9.2

管理 *IBM MQ*

附註

使用本資訊及其支援的產品之前，請先閱讀 第 463 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® MQ 6.2 版及所有後續版本與修訂版本。

當您將資訊傳送至 IBM 時，您授與 IBM 非專屬權限，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。

© Copyright International Business Machines Corporation 2007, 2025.

目錄

管理.....7
管理 IBM MQ 佇列管理程式及相關聯資源的方法..... 8
使用控制指令管理 IBM MQ..... 10
使用 MQSC 指令進行管理.....11
以互動方式執行 MQSC 指令..... 12
從文字檔執行 MQSC 指令..... 16
您可以在 z/OS 上從中發出 MQSC 指令的來源.....18
啟動時從 MQSC Script 自動配置.....19
具有特殊意義的同屬值及字元.....20
使用 PCF 指令自動化 IBM MQ 管理..... 20
IBM MQ 可程式指令格式簡介..... 21
使用 MQAI 來簡化 PCF 的使用.....31
使用 REST API 進行管理..... 65
「administrative REST API」入門.....66
使用 REST API 進行遠端管理.....70
REST API 時間戳記..... 74
REST API 錯誤處理..... 74
REST API 探索..... 76
REST API 國家語言支援.....77
REST API 版本..... 79
使用 Web 主控台進行管理.....80
Web 主控台入門.....80
「新建 Web 主控台」快速導覽.....82
在主控台類型之間切換..... 100
使用 IBM MQ Explorer 進行管理.....101
您可以使用 IBM MQ Explorer 執行的動作.....101
設定 IBM MQ Explorer.....103
使用 IBM MQ 工作列應用程式 (僅限 Windows)..... 108
IBM MQ 警示監視器應用程式 (僅限 Windows)..... 108
使用本端 IBM MQ 物件.....108
使用佇列管理程式.....109
停止 MQI 通道.....117
使用本端佇列..... 118
使用遠端佇列..... 126
使用別名佇列..... 127
使用模型佇列..... 129
使用無法傳送郵件的佇列.....130
使用管理主題..... 146
使用訂閱..... 149
使用服務..... 152
管理用於觸發的物件..... 158
在兩個系統之間使用 **dmpmqmsg** 公用程式.....160
使用遠端 IBM MQ 物件.....163
配置佇列管理程式以進行遠端管理.....163
管理用於遠端管理的指令伺服器.....167
在遠端佇列管理程式上發出 MQSC 指令.....168
編碼字集之間的資料轉換.....169
管理 Managed File Transfer..... 173
啟動 MFT 代理程式..... 173
列出 MFT 代理程式..... 177
停止 MFT 代理程式..... 178
啟動新的檔案傳送.....179

建立排定的檔案傳送 181
處理擱置中檔案傳送 182
觸發檔案傳送 183
監視正在進行的檔案傳送 184
在傳送日誌中檢視檔案傳送的狀態 185
監視 MFT 資源 187
使用檔案傳送範本 212
將資料從檔案傳送至訊息 214
將資料從訊息傳送至檔案 221
通訊協定橋接器 226
Connect:Direct 橋接器 245
從 IBM Integration Bus 使用 MFT 257
MFT 回復及重新啟動 257
設定回復停滯傳送的逾時 258
管理 MQ Telemetry 262
在 Linux 及 AIX 上配置遙測的佇列管理程式 263
在 Windows 上配置遙測的佇列管理程式 264
配置分散式佇列以將訊息傳送至 MQTT 用戶端 266
MQTT 用戶端識別、授權及鑑別 268
使用 TLS 進行遙測通道鑑別 273
遙測通道上的發佈保密 274
MQTT Java 用戶端及遙測通道的 TLS 配置 274
遙測通道 JAAS 配置 278
管理 AMQP 用戶端 280
檢視 AMQP 用戶端正在使用的 IBM MQ 物件 280
AMQP 用戶端識別、授權及鑑別 281
通道上的發佈隱私權 283
使用 TLS 設定 AMQP 用戶端 283
斷開 AMQP 用戶端與佇列管理器的連接 284
管理多重播送 284
開始使用多重播送 284
IBM MQ 多重播送主題拓蹼 285
控制多重播送訊息的大小 286
啟用多重播送傳訊的資料轉換 288
多重播送應用程式監視 288
多重播送訊息可靠性 289
進階多重播送作業 289
管理 IBM MQ for IBM i 292
使用 CL 指令管理 IBM MQ for IBM i 292
管理 IBM MQ for IBM i 的替代方式 304
IBM i 的工作管理 309
IBM i 上的可用性、備份、回復及重新啟動 314
靜止 IBM MQ for IBM i 350
管理 IBM MQ for z/OS 353
對 IBM MQ for z/OS 發出指令 354
IBM MQ for z/OS 公用程式 361
操作 IBM MQ for z/OS 363
撰寫程式以管理 IBM MQ for z/OS 381
在 z/OS 上管理 IBM MQ 資源 392
在 z/OS 上回復並重新啟動 423
IBM MQ 及 IMS 441
在 z/OS 上操作 Advanced Message Security 452
管理 IBM MQ Internet Pass-Thru 453
啟動和停止 MQIPT 453
使用指令行來管理 MQIPT 455
製作備份 460
效能調整 461

注意事項..... 463
程式設計介面資訊..... 464
商標..... 464

管理 IBM MQ

若要管理 IBM MQ 佇列管理程式及相關聯的資源，請從一組可用來啟動及管理這些資源的作業中選擇您偏好的方法。

關於這項作業

您可以在本端或遠端管理 IBM MQ 物件:

本端管理

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

如需相關資訊，請參閱 第 108 頁的『使用本端 IBM MQ 物件』。

遠端管理

IBM MQ 支援透過遠端管理從單一聯絡點進行管理。 遠端管理可讓您從本端系統發出在另一個系統上處理且也適用於 IBM MQ Explorer 的指令。 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。

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

如需相關資訊，請參閱 第 163 頁的『使用遠端 IBM MQ 物件』。

您可以使用許多不同的方法，在「IBM MQ」中建立及管理佇列管理程式及其相關資源。 這些方法包括指令行介面、圖形使用者介面及管理 API。

視您的平台而定，您可以使用不同的指令集來管理 IBM MQ :

- 第 8 頁的『IBM MQ 控制指令』
- 第 8 頁的『IBM MQ Script (MQSC) 指令』
- 第 8 頁的『可程式指令格式 (PCF)』
- administrative REST API
- IBM i 第 9 頁的『IBM i 控制語言 (CL)』

還有下列其他選項可用來建立及管理 IBM MQ 物件:

- Windows Linux 第 9 頁的『IBM MQ Explorer』
- 第 9 頁的『IBM MQ Console』
- Windows 第 9 頁的『Microsoft 叢集服務 (MSCS)』

z/OS 如需 IBM MQ for z/OS 上管理介面及選項的相關資訊，請參閱 第 353 頁的『管理 IBM MQ for z/OS』。

您可以使用 PCF 指令來自動化本端及遠端佇列管理程式的部分管理及監視作業。 在某些平台上使用「IBM MQ 管理介面 (MQAI)」也可以簡化這些指令。 如需自動化管理作業的相關資訊，請參閱 第 20 頁的『使用 PCF 指令自動化 IBM MQ 管理』。

相關概念

IBM MQ 技術概觀

相關工作

規劃

配置

© Copyright IBM Corp. 2007, 2025

相關參考
指令集比較

管理 IBM MQ 佇列管理程式及相關聯資源的方法

有數個不同的選項可用來管理 IBM MQ 佇列管理程式及相關聯的資源。

IBM MQ 控制指令

ALW

您可以使用控制指令，對佇列管理程式本身執行管理作業。

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

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

IBM MQ Script (MQSC) 指令

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

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

視指令上設定的旗標而定，您可以在三種模式下執行 **runmqsc** 指令:

- 驗證模式，其中會在本端佇列管理程式上驗證 MQSC 指令，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 指令可在所有平台上使用，包括 IBM i 及 z/OS 。 比較指令集中彙總了 MQSC 指令。

ALW 在 AIX, Linux, and Windows 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 可以建置在您從指令行執行的檔案中。 MQSC 指令可以傳送至遠端佇列管理程式。 如需完整資料，請參閱 第 16 頁的『從文字檔執行 MQSC 指令』。

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

附註: IBM i

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

如需使用 MQSC 指令的相關資訊，請參閱 第 11 頁的『使用 MQSC 指令進行管理』。

可程式指令格式 (PCF)

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

如需 PCF 的相關資訊，請參閱 第 21 頁的『IBM MQ 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。

administrative REST API

administrative REST API 提供 RESTful 介面，您可以用來管理 IBM MQ。 當您使用 administrative REST API 時，您在代表 IBM MQ 物件的 URL 上呼叫 HTTP 方法。 例如，您可以在下列 URL 上使用 HTTP 方法 GET 來要求 IBM MQ 安裝的相關資訊:

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

您可以搭配使用 administrative REST API 與程式設計語言的 HTTP/REST 實作，或使用 cURL 之類的工具或 REST 用戶端瀏覽器附加程式。

如需相關資訊，請參閱 administrative REST API

IBM i 控制語言 (CL)

IBM i

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

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

IBM MQ Explorer

Windows Linux

使用 IBM MQ Explorer，您可以執行下列動作:

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

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

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

如需相關資訊，請參閱 第 101 頁的『使用 IBM MQ Explorer 進行管理』。

您可以使用「IBM MQ Explorer 」來管理其他平台 (包括 z/OS) 上的遠端佇列管理程式。

IBM MQ Explorer 可以在產品安裝過程中安裝 (請參閱 安裝及解除安裝 IBM MQ) ， 也可以從 Fix Central 提供的獨立式 IBM MQ Explorer 下載項目安裝 (請參閱 在 Linux 和 Windows 上作為獨立式應用程式安裝及解除安裝 IBM MQ Explorer)。

IBM MQ Console

您可以使用 IBM MQ Console ， 從 Web 瀏覽器來管理 IBM MQ 。

如需相關資訊，請參閱 第 80 頁的『使用 Web 主控台進行管理』。

Microsoft 叢集服務 (MSCS)

Windows

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

請務必不要混淆 MSCS 意義上的叢集與 IBM MQ 叢集。 區別如下:

IBM MQ 叢集
這些是一個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
這些是電腦群組，它們連接在一起並配置成如果某個電腦失敗， MSCS 會執行 失效接手，將應用程式的狀態資料從失敗的電腦傳送至叢集中的另一部電腦，並在該處重新起始其作業。

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

ALW 使用控制指令管理 IBM MQ

控制指令提供在 AIX, Linux, and Windows 上執行一些 IBM MQ 管理作業的方法。

如果您想要發出控制指令，您的使用者 ID 必須是大部分控制指令的 mqm 群組成員。 如需此作業的相關資訊，請參閱 在 AIX, Linux, and Windows 上管理 IBM MQ 的權限。 此外，請注意環境特定資訊。 您的企業所使用的平台。

當使用在佇列管理程式上操作的控制指令時，您必須從與您使用的佇列管理程式相關聯的安裝中使用指令。

當使用在佇列管理程式上運作的控制指令時，如果該佇列管理程式已配置為使用 CHCKLOCL (REQUIRED) 的連線鑑別，則會觀察到連接失敗。

- 如果控制指令容許的話，請提供使用者 ID 和密碼。
- 請使用控制指令的 MQSC 對等項目 (如果有的話)。
- 當需要執行無法連接的控制指令時，請使用 -ns 選項來啟動佇列管理程式。

如需控制指令的完整清單，請參閱 IBM MQ 控制指令。

在 Windows 系統上使用控制指令

Windows

在 IBM MQ for Windows 中，您在命令提示字元輸入控制指令。

控制指令及其旗標不區分大小寫，但那些指令的引數 (例如佇列名稱及佇列管理程式名稱) 會區分大小寫。

例如，在指令中:

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

- 指令名稱可以大寫或小寫輸入，或兩者的混合。 這些都是有效的: crtmqm、 CRTMQM 和 CRTmqm。
- 旗標可以輸入為 -u、 -U、 /u 或 /U。
- SYSTEM.DEAD.LETTER.QUEUE 和 jupiter.queue.manager 必須完全如所示輸入。

在 AIX and Linux 系統上使用控制指令

Linux AIX

在 IBM MQ for AIX or Linux 系統中，您可以在 Shell 視窗中輸入控制指令。

在 UNIX and Linux 環境中，控制指令 (包括指令名稱本身、旗標及任何引數) 會區分大小寫。 例如，在指令中:

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

- 指令名稱必須是 crtmqm，而不是 CRTMQM。
- 旗標必須是 -u，而不是 -U。
- 無法傳送郵件的佇列稱為 SYSTEM.DEAD.LETTER.QUEUE。
- 引數指定為 jupiter.queue.manager，與 JUPITER.queue.manager 不同。

請小心輸入您在範例中看到的指令。

相關參考

IBM MQ 控制指令參照

使用 MQSC 指令進行管理

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。 MQSC 指令可在所有平台上使用。

關於這項作業

您發出 MQSC 指令的方式視您的平台而定:

- ALW 在 AIX, Linux, and Windows 上，您可以使用 **runmqsc** 指令向佇列管理程式發出 MQSC 指令。 您可以使用多種方式來執行 **runmqsc** 指令:
  - 以互動方式，從鍵盤發出指令。 請參閱 第 12 頁的『以互動方式執行 MQSC 指令』。
  - 從 ASCII 文字檔。 請參閱 第 16 頁的『從文字檔執行 MQSC 指令』。
  - 在遠端佇列管理程式上。 請參閱 第 163 頁的『使用遠端 IBM MQ 物件』。
- z/OS 在 z/OS 上，視指令而定，可以從許多來源發出 MQSC 指令。 如需相關資訊，請參閱 第 18 頁的『您可以在 z/OS 上從中發出 MQSC 指令的來源』。

MQSC 指令詳述於 MQSC 指令 區段中。

程序

- 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，參數通常可以任意順序出現; 如果參數具有對應值，則值必須直接出現在與它相關的參數之後。

  註: z/OS 在 z/OS 上，次要參數不必是秒。
- 關鍵字、括弧及值可以用任意數目的空白及逗點區隔。 語法圖中顯示的逗點一律可以取代為一或多個空白。 每一個參數前面必須至少有一個空白 (在主要參數後面) 在 z/OS 上除外 。
- 在指令的開頭或結尾，以及在參數、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。
- 包含空白、小寫字元或特殊字元的字串必須以單引號括住，除非下列其中一項為真:
  - 特殊字元是下列一或多個字元:
    - 句點 (.)
    - 正斜線 (/)
    - 底線 (_)
    - 百分比符號 (%)
  - z/OS 該指令是從 IBM MQ for z/OS 作業及控制台發出。
  - 字串是一個以星號結尾的同屬值。 (在 IBM i 上，這些必須以單引號括住)
  - 字串是單一星號，例如 TRACE (*) (在 IBM i 上，這些必須以單引號括住)
  - 字串是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。

- Multi
  在 多平台上，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。 當兩個不含空格的單引號解譯為零長度字串時，如果所使用的屬性是下列其中一個屬性，則例外:
  - TOPICSTR
  - SUB
  - USERDATA
  - SELECTOR
- z/OS
  在 z/OS 上，如果您想要以單引號括住空格，則必須將它輸入為 ('')。 不含字元 ('') 的字串與輸入 () 相同。
- 字串屬性中任何以 MQCHARV 類型為基礎的尾端空白 (例如: 選取器、子使用者資料) 都會被視為顯著，這表示 'abc ' 不等於 'abc'。
- 除非特別註明，否則左括弧後面接著右括弧 (中間沒有任何重要資訊) 無效。 例如，下列字串無效:

  ```
  NAME ( )
  ```
- 關鍵字不區分大小寫： AltER, alter 和 ALTER 都可以接受。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。
- 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 第 20 頁的『具有特殊意義的同屬值及字元』。

相關工作

解決 MQSC 指令的問題

相關參考

runmqsc (執行 MQSC 指令)

以互動方式執行 MQSC 指令

您可以使用指令視窗或 Shell，以互動方式使用 MQSC 指令。

開始之前

您可以設定執行 **runmqsc** 指令時所顯示的提示。 請參閱 第 13 頁的『設定 MQSC 命令提示字元』。

Linux AIX 當您在 UNIX and Linux 平台上以互動方式執行 MQSC 指令時， **runmqsc** 指令行支援指令恢復、指令完成及 Emacs 指令鍵。 請參閱第 15 頁的『AIX 和 Linux 上的 runmqsc 指令』。

關於這項作業

程序

1. 若要以互動方式使用 MQSC 指令，請開啟指令視窗或 Shell，並輸入下列指令:

   ```
   runmqsc QMgrName
   ```

   其中 *QMgrName* 指定您要處理 MQSC 指令的佇列管理程式名稱。 您可以將 *QMgrName* 保留空白，以處理預設佇列管理程式上的 MQSC 指令。

2. 視需要鍵入任何 MQSC 指令。 例如，若要建立稱為 ORANGE.LOCAL.QUEUE 的本端佇列，請輸入下列指令:

   ```
   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。
3. 輸入下列指令，以停止使用 MQSC 指令:

   ```
   end
   ```

   或者，您可以使用作業系統的 EOF 字元。

結果

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，下列訊息確認已建立佇列:

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

下列訊息指出您已發生語法錯誤:

```
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
```

這些訊息會傳送至標準輸出裝置。 如果您未正確輸入指令，請參閱指令的參考資訊，以尋找正確的語法。 請參閱 MQSC 指令。

相關工作

第 16 頁的『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，或反覆地使用特定的指令序列，您可以從文字檔重新導向 `stdin` 。 您也可以將輸出重新導向至檔案。

相關參考

runmqsc

設定 MQSC 命令提示字元

您可以使用 **MQPROMPT** 環境變數，將 MQSC 命令提示字元設為您選擇的提示字元。

程序

- 將 **MQPROMPT** 環境變數設為您選擇的提示。

  當以互動方式執行 **runmqsc** 主控台時，以及當從檔案或標準輸入裝置 (stdin) 將輸入重新導向至 **runmqsc** 時，都會插入提示。

您可以在命令提示字元中包含純文字，也可以使用與 IBM MQ 服務物件定義相同的 +VARNAME+ 表示法來插入環境變數。 如需相關資訊，請參閱 第 155 頁的『服務定義上的可取代插入項目』。

有許多其他可更換的插入項目 IBM MQ 提供，如下表所述。

可更換的插入項目	說明
MQ_HOST_NAME	系統的主機名稱
MQ_FILE_SEP	平台專用檔案分隔字元: – Linux AIX 在 AIX and Linux 系統上，MQ_FILE_SEP 是 /。 – Windows 在 Windows 系統上， MQ_FILE_SEP 的位置是 \
MQ_PATH_SEP	平台專用路徑分隔字元: – Linux AIX 在 AIX and Linux 系統上，MQ_PATH_SEP 是 :。 – Windows 在 Windows 系統上， MQ_PATH_SEP 的位置是 ;
MQ 日期時間	採用固定 YYYY-MM-DD hh:mm:ss.SSS 格式的本端系統日期和時間，例如: `2020-12-25 17:41:37.408`

附註:

– MQ 可更換插入項目值與 **runmqsc** 指令相關聯的 IBM MQ 安裝及主機系統相關。

– 展開插入時， MQPROMPT 限制為最多 256 個字元。 超出此值的 MQPROMPT 擴充會導致整個 MQPROMPT 字串被截斷而沒有擴充。

例如，若要將提示設為 MQSC，請輸入下列其中一個指令:

- Windows

```
set "MQPROMPT=MQSC"
```

- Linux AIX

```
export MQPROMPT="MQSC"
```

範例

下列範例顯示在 AIX 系統上設定 **MQPROMPT** 變數。 提示設定為顯示從相關聯系統環境變數取得的使用者名稱、佇列管理程式名稱，以及從 MQ 可更換插入項目取得的 IBM MQ 主機名稱:

```
sh> export MQPROMPT="+USER+ @ +QMNAME+ @ +MQ_HOST_NAME+> "
sh> runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2025.
Starting MQSC for queue manager MY.QMGR.

myuser @ MY.QMGR @ aix1> DISPLAY QMSTATUS
```

```
C:\ > set “MQPROMPT=+USERNAME+ @ +QMNAME+ @ +MQ_HOST_NAME+> “
C:\ > runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2025.
Starting MQSC for queue manager MY.QMGR.

myuser @ MY.QMGR @ WIN1> DISPLAY QMSTATUS
```

下列範例將時間戳記新增至上述 MQPROMPT 範例，取自 MQ 可更換的插入項目:

```
sh> export MQPROMPT="+MQ_DATE_TIME+ +USER+ @ +QMNAME+ @ +MQ_HOST_NAME+> "
sh> runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2025.
Starting MQSC for queue manager MY.QMGR.

2020-11-24 18:10:00.404 myuser @ MY.QMGR @ aix1> DISPLAY QMSTATUS
```

```
C:\ > set “MQPROMPT=+MQ_DATE_TIME+ +USERNAME+ @ +QMNAME+ @ +MQ_HOST_NAME+> “
C:\ > runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2025.
Starting MQSC for queue manager MY.QMGR.

2020-11-24 18:10:01.007 myuser @ MY.QMGR @ WIN1> DISPLAY QMSTATUS
```

Linux AIX AIX 和 Linux 上的 runmqsc 指令

AIX 和 Linux 上的 **runmqsc** 指令行支援指令恢復、指令完成及 Emacs 指令鍵。

可用的指令行編輯器功能如下:

- 使用上移鍵和下移鍵恢復先前輸入的指令
- 使用 Tab 鍵及空格鍵自動完成指令的下一個關鍵字
- Emacs 指令鍵或類似指令鍵功能

若要使用這些函數，必須安裝 curses 程式庫。 如果系統上未安裝 curses 程式庫，則 **runmqsc** 沒有指令行編輯器功能，且會在啟動 **runmqsc** 指令行時顯示一則訊息。 要安裝的 curses 程式庫名稱取決於 UNIX 平台:

- AIX 在 AIX 上，安裝 curses。
- 在 Linux 上，安裝 ncurses。

在 AIX 上安裝 ncurses 或 curses

註: 下列範例使用 Linux 的指示

執行下列指令，以尋找現有的 ncurses 套件:

```
rpm -qa | grep -i ncurses
```

必要的 ncurses 套件如下:

```
ncurses-term-6.1-7.20180224.el8.noarch
ncurses-6.1-7.20180224.el8.x86_64
ncurses-base-6.1-7.20180224.el8.noarch
ncurses-c++-libs-6.1-7.20180224.el8.x86_64
ncurses-libs-6.1-7.20180224.el8.x86_64
ncurses-compat-libs-6.1-7.20180224.el8.x86_64
ncurses-devel-6.1-7.20180224.el8.x86_64
```

您可以執行下列指令，來安裝上述文字中列出的所有必要 ncurses 套件:

```
yum install ncurses*
```

自訂 Emacs 按鍵連結

您可以自訂連結至指令的金鑰。 例如，您可以將按鍵連結至 vi 連結，而不是預設 Emacs 按鍵連結。

透過編輯儲存在起始目錄中的 .editrc 檔案來自訂金鑰。 如需相關資訊，請參閱 FreeBSD 線上指令說明中的 editrc 。

停用指令恢復、指令完成及 Emacs 指令鍵

您可以透過設定環境變數來停用指令恢復、指令完成及 Emacs 指令鍵。 將環境變數 **MQ_OVERRIDE_LIBEDIT_LOAD** 設為 TRUE。

當 **runmqsc** 顯示下列參考訊息時，可使用此環境變數作為暫行解決方法:

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

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，或反覆地使用特定的指令序列，您可以從文字檔重新導向 stdin 。 您也可以將輸出重新導向至檔案。

關於這項作業

runmqsc 指令的輸入取自 標準輸入裝置，也稱為 stdin。 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 (例如)。

runmqsc 指令的輸出會輸出至 標準輸出裝置，也稱為 stdout。 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。

當您使用下列指令時，可能想要將 MQSC 指令建置成 Script:

- z/OS CSQINP1、 CSQINP2 及 CSQINPX 起始設定資料集，或 z/OS 上的 CSQUTIL 批次公用程式。
- IBM i IBM i 上的 **STRMQM** 指令。
- ALW AIX, Linux, and Windows 上的 **runmqsc** 指令。

您可以使用 **MQPROMPT** 環境變數，將 MQSC 命令提示字元設為您選擇的提示字元。 如需相關資訊，請參閱第 13 頁的『設定 MQSC 命令提示字元』。

程序

1. 建立包含您要執行之 MQSC 指令的文字檔。
   - 為了在 IBM MQ 環境之間實現可攜性，請將 MQSC 指令檔中的行長度限制為 72 個字元。
   - 每個指令都必須從新的一行開始。
   - 系統不處理第一個位置中以星號 (*) 開頭的行。 這可用來將註解插入檔案中。
   - 空白行予以忽略。
   - 加號 (+) 指出指令從下一行的第一個非空白字元繼續執行。 如果您使用 + 來繼續執行指令，請記得在下一個參數 之前至少保留一個空白 (但在 z/OS 上不需要這樣做)。 當指令重新組合成單一字串時，會捨棄任何註解或空白行。
   - 減號 (-) ，表示將從下一行開始繼續執行指令。 當指令重新組合成單一字串時，會捨棄任何註解或空白行。
   - 內含在 Escape PCF (可程式化指令格式) 指令中的 MQSC 指令不能以加號或減號繼續。 整個指令必須包含在單一 Escape 指令中。 如需 PCF 指令的相關資訊，請參閱第 21 頁的『IBM MQ 可程式指令格式簡介』。
   - 在 多平台上，以及在 z/OS 上，對於從 CSQUTIL 批次公用程式發出的指令，您可以使用分號字元 (;) 來終止指令，即使您已在前一行結尾輸入加號 (+) 也一樣。
   - 行不得以鍵盤控制字元 (例如， Tab 鍵) 結尾。
   - 如果您從文字檔重新導向 stdin 以用戶端模式執行 **runmqsc** 指令，並提供 **-u** 旗標以提供認證，則 **runmqsc** 指令不會提示輸入密碼，而是從 stdin 讀取密碼。 您應該確定透過 stdin 提供的第一行資料是密碼。 作法是使用指令行工具 (例如 "echo" 或 "cat") ，並將後面接著 MQSC Script 的密碼傳遞至 **runmqsc** 指令 stdin。

- Windows 在 Windows 上，如果在指令 Script 中使用某些特殊字元 (例如井號 (£) 及邏輯 NOT (¬)) (例如，作為物件說明的一部分)，則它們在指令 (例如 **DISPLAY QLOCAL**) 的輸出中以不同方式顯示。
- 如需 MQSC 指令語法的相關資訊，請參閱 MQSC 指令。
- 您可以使用範例 MQSC 指令檔來協助您建立文字檔:

  amqscos0.tst
  範例程式所使用的物件定義。

  amqscic0.tst
  CICS 交易的佇列定義。

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

  Linux AIX 在 AIX and Linux 上，這些檔案位於 *MQ_INSTALLATION_PATH*/samp 目錄中。 *MQ_INSTALLATION_PATH* 代表 IBM MQ 安裝所在的高階目錄。

2. 請驗證本端佇列管理程式上的指令語法是否正確，而不執行指令。 在 **runmqsc** 指令上使用 **-v** 旗標。

   V 9.2.0

   - 從 IBM MQ 9.2.0 開始，使用 **-f** 選項來識別輸入文字檔名。 例如:

     ```
     runmqsc -f myprog.in -v QmgrName
     ```

   - For Long Term Support releases before IBM MQ 9.2.0, and Continuous Delivery releases before IBM MQ 9.1.4, use the < operator to direct the MQSC commands from the input text file to the command. 例如:

     ```
     runmqsc -v QmgrName < myprog.in
     ```

   傳回的報告類似於 第 18 頁的圖 2 中顯示的報告。

   您無法在驗證指令時指定遠端佇列管理程式。 也就是說，您無法指定 **-w** 旗標。

3. 當指令語法正確時，請移除 **-v** 旗標，然後重新執行 **runmqsc** 指令。

   V 9.2.0

   - 從 IBM MQ 9.2.0 執行 (例如) 下列指令:

     ```
     runmqsc -f myprog.in QmgrName
     ```

   - 若為 IBM MQ 9.2.0 之前的 Long Term Support 版本，以及 IBM MQ 9.1.4 之前的 Continuous Delivery 版本，請使用下列其中一個指令:
     - The < operator directs input from a text file. 例如，下列指令會執行文字檔 myprog.in 中包含的一連串指令:

       ```
       runmqsc QMgrName < myprog.in
       ```

     - > 運算子會將輸出導向至文字檔。 例如，下列指令會執行文字檔 myprog.in 中包含的一系列指令，並將其輸出至稱為 results.out 的檔案:

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

   第 18 頁的圖 1 顯示來自 MQSC 指令檔的擷取 myprog.in，而 第 18 頁的圖 2 顯示 results.out 中對應輸出的擷取。

範例

MQSC 指令以人類可讀的格式 (即 ASCII 文字) 撰寫。 下列範例是 MQSC 指令檔的擷取，其中顯示 MQSC 指令 **DEFINE QLOCAL**。

```
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 *1:* 從 *MQSC* 指令檔擷取

當 **runmqsc** 指令完成時，會傳回報告。 下列範例是從報告擷取:

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

圖 *2:* 從 *MQSC* 指令報告檔擷取

相關工作

第 13 頁的『設定 MQSC 命令提示字元』
您可以使用 **MQPROMPT** 環境變數，將 MQSC 命令提示字元設為您選擇的提示字元。

第 12 頁的『以互動方式執行 MQSC 指令』
您可以使用指令視窗或 Shell，以互動方式使用 MQSC 指令。

相關參考

runmqsc

z/OS 您可以在 z/OS 上從中發出 MQSC 指令的來源

MQSC 指令可以從各種來源發出，視指令而定。

可以從下列來源發出指令:

- z/OS 主控台或對等項目
- 起始設定輸入資料集 CSQINP1、 CSQINP2、CSQINPT 及 CSQINPX
- CSQUTIL 批次公用程式
- 適當授權的應用程式，將指令當作訊息傳送至 SYSTEM.COMMAND.INPUT 佇列

z/OS 如需進一步詳細資料，請參閱 第 354 頁的『對 IBM MQ for z/OS 發出指令』。

不過，並非所有指令都可以從所有這些來源發出。 指令可以根據是否可以從下列項目發出來分類:

1
CSQINP1

2
CSQINP2

C
z/OS 主控台

R
命令伺服器和命令佇列，透過 CSQUTIL、CSQINPT、CSQINPX 或應用程式等方式執行

在 MQSC 指令中的指令說明內，這些來源是透過在每一個指令說明中使用字元 1、2、C 及 R 來識別。

Multi V 9.2.0 啟動時從 MQSC Script 自動配置

從 IBM MQ 9.2.0 開始，您可以配置佇列管理程式，在每次啟動佇列管理程式時自動套用 MQSC Script 或 MQSC Script 集的內容。

您可以使用此功能來具有可修改的配置，並在下一次重新啟動佇列管理程式時自動重播。 例如，如果一或多個 Script 位於裝載磁碟機上，則可以具有集中式配置，在啟動時，會將最新版本套用至每個佇列管理程式。

這可能很有用的特定實務範例，是透過具有一組它們全部適用的單一配置，來確保統一叢集在叢集中的所有佇列管理程式上包含相同的定義。 如需此範例，請參閱 建立新的統一叢集。

開始之前

您可以使用：

1. 單一 Script ，並使用 MQSC 指令建立文字檔。
2. 一組 MQSC Script:
   - 識別將存在配置的目錄，以及
   - 在該目錄中，建立檔案，每一個檔案都具有副檔名 `.mqsc`，例如 `queues.mqsc`。

假設此 Script 會在每次啟動佇列管理程式時重新套用，則可以重播指令是很重要的。 例如， **DEFINE** 指令必須包含 **REPLACE** 字串，否則該指令會在第二個佇列管理程式啟動時顯示為失敗，因為該物件已存在。

請注意，在 MQSC Script 中，任何以 `*` 為字首的行都會被視為註解。

啟用 MQSC Script 的自動配置

您可以使用 **crtmqm** 指令的 **-ic** 旗標，並指向特定檔案或目錄，來配置新的佇列管理程式。 所提供的值儲存在 `qm.ini` 檔案中的 AutoConfig 段落下，作為屬性 **MQSCConfig**。

您可以透過新增指向有效檔案或目錄的 AutoConfig 段落屬性 **MQSCConfig**，來配置現有佇列管理程式以啟用自動 MQSC 配置。 例如：

```
AutoConfig:
   MQSCConfig=C:\mq_configuration\uniclus.mqsc
```

自動配置如何運作?

在佇列管理程式啟動期間， AutoConfig 段落屬性 **MQSCConfig** 所識別的配置會通過 **runmqsc** 驗證，以確保語法有效，然後以單一檔案形式儲存在佇列管理程式資料樹狀結構中 `autocfg` 目錄 `cached.mqsc`。

在中處理目錄中的多個檔案時，會按字母順序處理這些檔案，如果它包含 MQSC end 或 quit 指令，則會跳過該檔案的其餘內容。

在第一次啟動佇列管理程式期間，如果無法讀取檔案或目錄，或具有無效 MQSC 語法的檔案，則會阻止佇列管理程式啟動，並將適當的錯誤訊息傳送至主控台及佇列管理程式錯誤日誌。

在後續重新啟動時，如果所指向的檔案或目錄無法讀取或包含無效 MQSC 語法，則會使用先前快取的檔案，且寫入佇列管理程式錯誤日誌的訊息會強調顯示此情況。

V 9.2.2 在將 cached.mqsc 的內容套用至佇列管理程式時，當已套用所有 MQSC 指令時，會啟用佇列管理程式以供應用程式連接。 所套用配置的 **runmqsc** 日誌會儲存在佇列管理程式的 errors 目錄中，成為稱為 autocfgmqsc.LOG 的檔案。

此外，任何未順利完成的 MQSC 指令都會記載至佇列管理程式錯誤日誌，以識別指令失敗的原因。

具有特殊意義的同屬值及字元

部分字元，例如反斜線 (\) 及雙引號 (") 與 MQSC 指令一起使用時，字元具有特殊意義。 部分特殊字元可以與參數搭配使用，可以具有同屬值，但必須正確地指定。

在反斜線 (\) 和雙引號 (") 之前 具有 \的字元，亦即，如果您想要在文字中輸入 \ 或 " ，請輸入 \\ 或 \" 。

只要參數可以有同屬值，就會輸入以星號 (*) 結尾的參數，例如 ABC*。 同屬值表示所有以開頭的值; 因此 ABC* 表示所有以 ABC 開頭的值。 如果值中使用需要引號的字元，則星號必須放在引號內，因此 'abc*'。 星號必須是值中的最後一個或唯一字元。

同屬值中不接受問號 (?) 和冒號 (:)。

當您需要在欄位中使用任何這些特殊字元 (例如作為說明的一部分) 時，必須以單引號括住整個字串。

表 *1:* 具有特殊意義之字元的說明

字元	說明
	使用空白作為分隔字元。 除了以單引號 (') 括住的字串之外，多個空白相當於單一空白。 這些字串屬性中以 MQCHARV 類型為基礎的任何尾端空白都會被視為重要。
,	使用逗點作為分隔字元。 多個逗點相當於單一逗點，但以單引號 (') 括住的字串除外。
'	單引號指出字串的開頭或結尾。 IBM MQ 會保留以引號括住的所有字元，與輸入的字元完全相同。 計算字串長度時，不包括包含單引號。
''	在計算字串長度時， IBM MQ 會將字串內的單引號視為一個字元，且字串不會終止。
=	z/OS 在 z/OS 上，等號指出參數值的開頭，以逗點或空白結尾。
(	左括弧指出參數值或值清單的開頭。
)	右括弧指出參數值或值清單的結尾。
:	冒號表示內含的範圍。 例如 (1: 5) 表示 (1,2,3,4, 5)。 此表示法只能在 **TRACE** 指令中使用。
*	星號表示全部。 例如， **DISPLAY TRACE (*)** 表示顯示所有追蹤，而 **DISPLAY QUEUE (PAY*)** 表示顯示名稱以 PAY 開頭的所有佇列。

使用 PCF 指令自動化 IBM MQ 管理

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) 指令來自動執行本端及遠端佇列管理程式的管理作業。 本節假設您有管理 IBM MQ 物件的經驗。

PCF 指令

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

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

每一個 PCF 指令都是內嵌在 IBM MQ 訊息的應用程式資料部分中的資料結構。 每一個指令都會以與任何其他訊息相同的方式，使用 MQI 函數 MQPUT 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動作。

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

簡言之，以下是建立 PCF 指令訊息所需的部分內容:

訊息描述子
這是標準 IBM MQ 訊息描述子，其中:

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

應用程式資料
包含 PCF 訊息 (包括 PCF 標頭) ，其中:

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

如需 PCF 資料結構及其實作方式的完整說明，請參閱 第 21 頁的『IBM MQ 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。

IBM MQ 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，特別是當網路的大小和複雜性增加時。

下列項目支援「可程式指令格式」:

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

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，佇列建立及刪除。
- 效能監視。

  例如，佇列深度上限或訊息速率。
- 控制。

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

  透過網路的替代路徑定義。

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

何謂 PCF?

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

每一個佇列管理程式都有一個具有標準佇列名稱的管理佇列，且您的應用程式可以將 PCF 指令訊息傳送至該佇列。 每一個佇列管理程式也有一個指令伺服器，可處理來自管理佇列的指令訊息。 因此，網路中的任何佇列管理程式都可以處理 PCF 指令訊息，而且可以使用您指定的回覆佇列，將回覆資料傳回至應用程式。 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 IBM MQ 可程式指令格式

您可以在系統管理程式中使用 PCF 進行 IBM MQ 遠端管理。

本節包括:

- 第 22 頁的『PCF 指令訊息』
- 第 24 頁的『IBM MQ 中的 PCF 回應』
- z/OS 第 26 頁的『延伸回應』
- IBM MQ 物件的命名規則
- 第 27 頁的『IBM MQ 中 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。

Multi 在 多平台上， PCF 指令傳送至其中的佇列一律稱為 SYSTEM.ADMIN.COMMAND.QUEUE。

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

如何發出 PCF 指令訊息

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

註:

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

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

PCF 指令的訊息描述子

IBM MQ 訊息描述子完整記錄在 MQMD-訊息描述子中。

PCF 指令訊息在訊息描述子中包含下列欄位:

報告
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。

期限
任何有效值 (視需要)。

意見
設為 MQFB_NONE

Multi **編碼**
如果您要傳送至 IBM MQ for Multiplatforms 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。

Multi **CodedCharSetId**
如果您要傳送至 IBM MQ for Multiplatforms 系統，請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。

格式
設為 MQFMT_ADMIN。

優先順序
任何有效值 (視需要)。

持續性
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，或者可以指定 MQCI_NONE 以指出沒有相關性 ID。

ReplyToQ
接收回應的佇列名稱。

回覆目的地佇列管理程式
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

偏移
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的佇列， 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的佇列。 在呼叫之後，袋子的內容保持不變。

作為此呼叫的輸入，您必須提供:

- MQI 連線控點。
- 要放置訊息之佇列的物件控點。
- 訊息描述子。 如需訊息描述子的相關資訊，請參閱 MQMD-訊息描述子。
- 使用 MQPMO 結構放置訊息選項。 如需 MQPMO 結構的相關資訊，請參閱 MQPMO-Put-message 選項。
- 要轉換為訊息之工具袋的控點。

  註: 如果工具袋包含管理訊息，且使用 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訊息，並且可以使用 MQGET 呼叫而非 mqGetBag 呼叫來擷取訊息。 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。

IBM MQ 中的 *PCF* 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設定訊息 ID 和相關性 ID。

如果指令訊息的 PCF 標頭類型是 MQCFT_COMMAND ，則只會產生標準回應。 z/OS 以外的所有平台都支援這類指令。 較舊的應用程式在 z/OS 上不支援 PCF;「 IBM MQ Windows 探險家」是這類應用程式 (不過，IBM WebSphere MQ 6.0 或更新版本的 IBM 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。 此結構顯示在回應的定義中 (可程式指令格式的定義) 如 一律傳回。 對於這些回應，需要為回應中的物件命名以識別套用資料的物件的原因。

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。

格式
此欄位是 MQFMT_ADMIN。

編碼
設為 MQENC_NATIVE。

CodedCharSetId
設為 MQCCSI_Q_MGR。

持續性
與指令訊息中的相同。

優先順序
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接著一個以上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。

每一個這類訊息的開頭都是回應格式標頭，其 *CompCode* 值為 MQCC_FAILED，且 *Reason* 欄位會識別特定錯誤。 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* 欄位會識別錯誤， MQCFIN 結構中的 *Value* 欄位是錯誤指令之 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。 此訊息在標頭之後沒有參數結構。

資料回應

此回應包含對 inquire 指令的 OK 回應 (如先前所述)。 「確定」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。

z/OS 延伸回應

在 z/OS 上發出的指令會產生延伸回應。

延伸回應有三種類型:

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

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

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

延伸回應至少有兩個參數結構:

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

許多回應都有其他參數結構，下列各節會說明這些結構。

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

Inquire 指令的延伸回應

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

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

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

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

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

對 INQUIRE 以外指令的延伸回應

成功指令會產生訊息回應，其中標頭中 *CompCode* 欄位的值為 MQCC_OK，而 *Reason* 欄位的值為 MQRC_NONE。 一律至少有一則訊息; 它可能是參考資訊 (MQCFT_XR_MSG) 或摘要 (MQCFT_XR_SUMMARY)。 可能有其他參考資訊 (MQCFT_XR_MSG 類型) 訊息。 每一個參考訊息可能包含一些其他參數結構，以及指令的相關資訊; 請參閱個別指令說明，以瞭解可能發生的結構。

失敗的指令會產生錯誤訊息回應 (類型 MQCFT_XR_MSG)，其中標頭中 *CompCode* 欄位的值為 MQCC_FAILED，而 *Reason* 欄位會識別特定錯誤。 每一則訊息可能包含一些其他參數結構，以及錯誤的相關資訊: 請參閱可能發生之結構的個別錯誤說明。 可能會產生參考訊息回應。 可能會選擇性地有摘要回應 (MQCFT_XR_SUMMARY)， *CompCode* 值為 MQCC_FAILED， *Reason* 欄位值為 MQRCC_COMMAND_FAILED。

使用 CommandScope 對指令的延伸回應

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

起始回應集具有下列其他參數結構:

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

IBM MQ 中 *PCF* 指令的權限檢查

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

在處理指令的系統上執行檢查; 因此此使用者 ID 必須存在於目標系統上，且具有處理指令的必要權限。 如果訊息來自遠端系統，則在目標系統上實現現有 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。

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

IBM MQ for IBM i

IBM i

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

此外，還會針對特定 PCF 指令執行 IBM MQ 物件權限檢查，如 第 28 頁的表 2 所示。

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

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

- Ping 通道
- 變更通道
- 複製通道

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

IBM MQ for UNIX, Linux, and Windows

ALW

為了處理任何 PCF 指令，使用者 ID 必須具有目標系統上佇列管理程式物件的 *dsp* 權限。 此外，還會針對特定 PCF 指令執行 IBM MQ 物件權限檢查，如 第 28 頁的表 2 所示。

若要處理下列任何指令，使用者 ID 必須屬於群組 *mqm*。

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

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

IBM MQ Multiplatforms 的物件權限

Multi

表 *2:* 物件權限

指令	**IBM MQ 物件權限**	**類別權限 (適用於物件類型)**
變更鑑別資訊	dsp 和 chg	不適用
變更通道	dsp 和 chg	不適用
變更通道接聽器	dsp 和 chg	不適用
變更用戶端連線通道	dsp 和 chg	不適用
變更名單	dsp 和 chg	不適用
變更處理程序	dsp 和 chg	不適用

表 *2:* 物件權限 *(*繼續*)*		
指令	**IBM MQ 物件權限**	**類別權限 (適用於物件類型)**
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*	從*:* dsp 至*:* chg	crt
複製通道	dsp	crt
複製通道 (取代) 請參閱附註 *1*	從*:* dsp 至*:* chg	crt
複製通道接聽器	dsp	crt
複製通道接聽器 (取代) 請參閱附註 *1*	從*:* dsp 至*:* 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*	從*:* dsp 至*:* 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*	從*:* dsp 至*:* 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

表 *2:* 物件權限 *(*繼續*)*		
指令	**IBM MQ 物件權限**	**類別權限 (適用於物件類型)**
建立佇列 (取代) 請參閱附註 *1*	*(*系統預設佇列*)* dsp 至*:* dsp 及 chg	crt
建立服務	*(*系統預設佇列*)* dsp	crt
建立服務 (取代) 請參閱附註 *1*	*(*系統預設佇列*)* dsp *:* chg	crt
刪除鑑別資訊	dsp 和 dlt	不適用
刪除權限記錄	*(*佇列管理程式物件*)* chg 請參閱附註 *4*	請參閱附註 *4*
刪除通道	dsp 和 dlt	不適用
刪除通道接聽器	dsp 和 dlt	不適用
刪除用戶端連線通道	dsp 和 dlt	不適用
刪除名單	dsp 和 dlt	不適用
刪除處理程序	dsp 和 dlt	不適用
刪除佇列	dsp 和 dlt	不適用
刪除服務	dsp 和 dlt	不適用
查詢鑑別資訊	dsp	不適用
查詢權限記錄	請參閱附註 *4*	請參閱附註 *4*
查詢通道	dsp	不適用
查詢通道接聽器	dsp	不適用
查詢通道狀態 (適用於 **ChannelType** MQCHT_CLSSDR)	inq	不適用
查詢用戶端連線通道	dsp	不適用
查詢名單	dsp	不適用
查詢處理程序	dsp	不適用
查詢佇列	dsp	不適用
查詢佇列管理程式	請參閱附註 *3*	不適用
查詢佇列狀態	dsp	不適用
查詢服務	dsp	不適用
Ping 通道	ctrl	不適用
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

表 2: 物件權限 (繼續)		
指令	**IBM MQ 物件權限**	**類別權限 (適用於物件類型)**
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，當佇列管理程式已啟動且沒有 -a 參數時，指令伺服器會啟動。 如需相關資訊，請參閱 可程式化指令格式參照。
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和使用者的權限記錄。 請勿將此權限授與一般使用者或應用程式。

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

Multi 使用 MQAI 來簡化 PCF 的使用

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

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 更容易的方式來處理物件的內容 (或參數)。

使用 MQAI 的優點如下:

簡化 PCF 訊息的使用

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

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

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

更容易處理錯誤狀況

很難從 PCF 指令取得回覆碼。 MQAI 可讓程式更容易處理錯誤狀況。

在應用程式之間交換資料

應用程式資料以 PCF 格式傳送，並由 MQAI 壓縮及解壓縮。 如果您的訊息資料是由整數和字串所組成，您可以使用 MQAI 來利用 PCF 資料的 IBM MQ 內建資料轉換。 這可避免寫入資料轉換結束程式的需要。

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

使用 MQAI 的範例

下列範例程式示範如何使用 MQAI 來執行各種作業:

- amqsaicq.c: 建立本端佇列。
- amqsaiem.c: 使用簡式事件監視器在畫面上顯示事件。
- amqsailq.c: 列印所有本端佇列及其現行深度的清單。
- amqsaicl.c: 列印所有通道及其類型的清單。

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 IBM MQ 相同的程式庫。 如需如何建置 IBM MQ 應用程式的相關資訊，請參閱 建置程序化應用程式。

使用 MQAI 來配置 IBM MQ 的提示

MQAI 使用 PCF 訊息將管理指令傳送至指令伺服器，而不是直接處理指令伺服器本身。 您可以在 第 32 頁的『使用 MQAI 來配置 IBM MQ 的提示』中找到使用 MQAI 來配置 IBM MQ 的提示。

相關參考

IBM MQ 管理介面參照

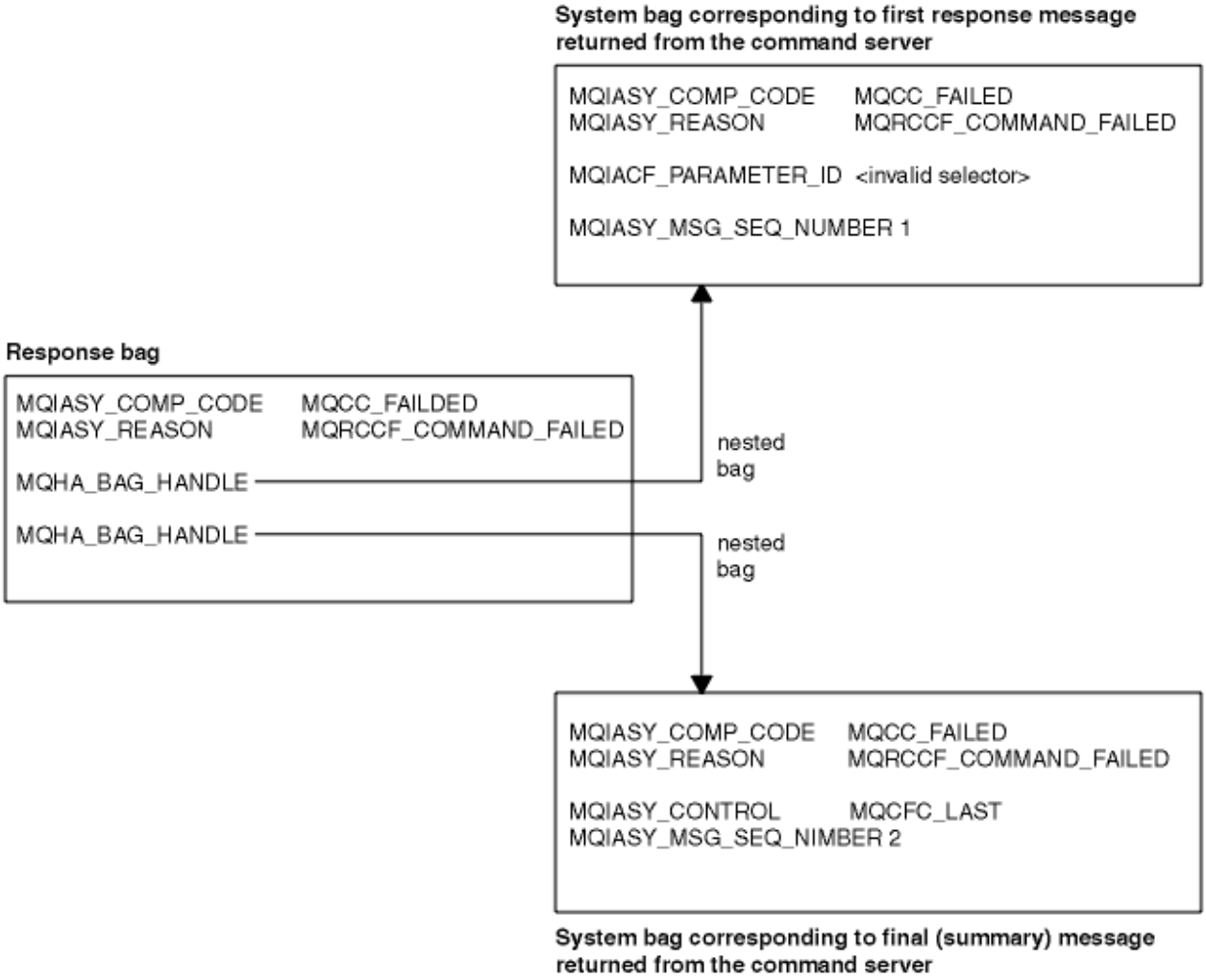
Multi 使用 MQAI 來配置 IBM MQ 的提示

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

- IBM MQ 中的字串會以空白填補至固定長度。 使用 C，通常可以提供以空值結尾的字串作為 IBM MQ 程式設計介面的輸入參數。
- 如果要清除字串屬性的值，請將它設為單一空白，而不是空字串。
- 請事先考量您要變更的屬性，並只查詢那些屬性。
- 某些屬性無法變更，例如佇列名稱或通道類型。 請確定您嘗試只變更那些可修改的屬性。 請參閱特定 PCF 變更物件的必要及選用參數清單。 請參閱 可程式指令格式的定義。
- 如果 MQAI 呼叫失敗，則會將失敗的部分詳細資料傳回回應工具袋。 然後可以在選取元 MQHA_BAG_HANDLE 可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詳細資訊。

  如需 MQExecute 的相關資訊，請參閱 第 64 頁的『使用 mqExecute 呼叫將管理指令傳送至 qm 指令伺服器』

  下圖顯示此實務範例:



Multi 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。

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 mqSetInteger 呼叫進行轉換。

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。

Multi 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 34 頁的圖 3 所示。

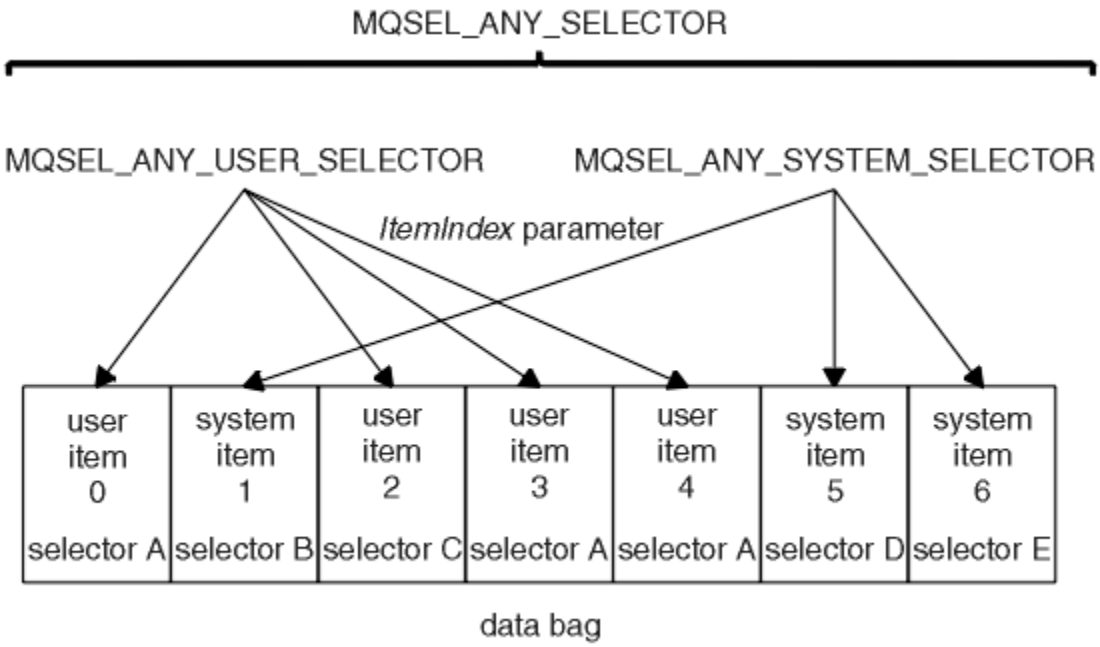


圖 *3:* 編製索引

在 第 34 頁的圖 3 中，下列索引配對可以參照使用者項目 3 (選取元 A):

- selector A (ItemIndex 1)
- MQSEL_ANY_USER_SELECTOR (ItemIndex 2)
- MQSEL_ANY_SELECTOR (itemIndex 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 第 61 頁的『變更工具袋內的資訊』 和 第 63 頁的『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，如果工具袋中有特定選取器的三個實例，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。

Multi *MQAI* 中的資料轉換處理

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 34 頁的表 3 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 *3: 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 3: 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
mqBufferToBagmqBufferToBag	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。

相關概念

第 169 頁的『編碼字集之間的資料轉換』
佇列管理程式可以將 IBM MQ 定義格式 (也稱為內建格式) 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。

第 171 頁的『ccsid_part2.tbl 檔案』
`ccsid_part2.tbl` 檔案用來提供其他 CCSID 資訊。 `ccsid_part2.tbl` 檔案會取代 IBM MQ 9.0 之前使用的 `ccsid.tbl` 檔案。

Multi 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 *4: PCF* 指令類型	
袋的型別	**MQIASY_TYPE 項目的起始值**
MQCBO_ADMIN_BAG	MQCFT_COMMAND
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 36 頁的表 4 所示。

表 *5: MQMD* 的 *format* 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 36 頁的表 5 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 36 頁的表 6 中所示的值。

表 *6:* 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 36 頁的表 5
Expiry	30 秒 (附註 第 37 頁的『1』)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	取決於工具袋 CCSID (附註 第 37 頁的『2』)
Format	請參閱 第 36 頁的表 5
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0

表 6: 訊息描述子值 (繼續)	
參數	**值**
ReplyToQ	請參閱附註 第 37 頁的『3』
ReplyToQMgr	Blank

附註:

1. 可以在 mqExecute 呼叫上使用 **OptionsBag** 參數來置換此值。 如需此作業的相關資訊，請參閱 mqExecute。
2. 請參閱 第 34 頁的『MQAI 中的資料轉換處理』。
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。

Multi 用於建立本端佇列的範例 C 程式 (amqsaicq.c)

範例 C 程式 `amqsaicq.c` 會使用 MQAI 建立本端佇列。

```
/******************************************************************************/
/*                                                                            */
/* Program name: AMQSAICQ.C                                                   */
/*                                                                            */
/* Description:  Sample C program to create a local queue using the           */
/*               IBM MQ Administration Interface (MQAI).                      */
/*                                                                            */
/* Statement:    Licensed Materials - Property of IBM                         */
/*                                                                            */
/*               84H2000, 5765-B73                                            */
/*               84H2001, 5639-B42                                            */
/*               84H2002, 5765-B74                                            */
/*               84H2003, 5765-B75                                            */
/*               84H2004, 5639-B43                                            */
/*                                                                            */
/*               (C) Copyright IBM Corp. 1999, 2025.                          */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* Function:                                                                  */
/*    AMQSAICQ is a sample C program that creates a local queue and is an     */
/*    example of the use of the mqExecute call.                               */
/*                                                                            */
/*     - The name of the queue to be created is a parameter to the program.   */
/*                                                                            */
/*     - A PCF command is built by placing items into an MQAI bag.            */
/*       These are:-                                                          */
/*            - The name of the queue                                         */
/*            - The type of queue required, which, in this case, is local.    */
/*                                                                            */
/*     - The mqExecute call is executed with the command MQCMD_CREATE_Q.      */
/*       The call generates the correct PCF structure.                        */
/*       The call receives the reply from the command server and formats into */
/*       the response bag.                                                    */
/*                                                                            */
/*     - The completion code from the mqExecute call is checked and if there  */
/*       is a failure from the command server then the code returned by the   */
/*       command server is retrieved from the system bag that is              */
/*       embedded in the response bag to the mqExecute call.                  */
/*                                                                            */
/* Note: The command server must be running.                                  */
/*                                                                            */
/*                                                                            */

/******************************************************************************/
/*                                                                            */
/* AMQSAICQ has 2 parameters - the name of the local queue to be created      */
/*                          - the queue manager name (optional)               */
/*                                                                            */
/******************************************************************************/
/******************************************************************************/
/* Includes                                                                   */
/******************************************************************************/
#include <stdio.h>
```

```
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

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

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

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

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

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

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


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

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

}
/*****************************************************************************/
/*                                                                           */
/* Function:    CreateLocalQueue                                             */
/* Description: Create a local queue by sending a PCF command to the command  */
/*              server.                                                      */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Handle to the queue manager                            */
/*                    Name of the queue to be created                        */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.     */
/*        The call generates the correct PCF structure.                      */
/*        The default options to the call are used so that the command is sent*/
/*        to the SYSTEM.ADMIN.COMMAND.QUEUE.                                 */
/*        The reply from the command server is placed on a temporary dynamic  */
/*        queue.                                                             */
```

```
/*        The reply is read from the temporary queue and formatted into the   */
/*        response bag.                                                       */
/*                                                                            */
/*        The completion code from the mqExecute call is checked and if there */
/*        is a failure from the command server then the code returned by the  */
/*        command server is retrieved from the system bag that is             */
/*        embedded in the response bag to the mqExecute call.                 */
/*                                                                            */
/******************************************************************************/
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG reason;                       /* reason code                    */
   MQLONG compCode;                     /* completion code                */
   MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute       */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute      */
   MQHBAG resultBag;                    /* result bag from mqExecute      */
   MQLONG mqExecuteCC;                  /* mqExecute completion code      */
   MQLONG mqExecuteRC;                  /* mqExecute reason code          */

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


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

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

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

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

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

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


   /******************************************************************************/
   /* Check the result from mqExecute call and find the error if it failed.   */
```

```
   /*****************************************************************/
   if ( compCode == MQCC_OK )
      printf("Local queue %s successfully created\n", qName);
   else
   {
      printf("Creation of local queue %s failed: Completion Code = %d
               qName, compCode, reason);
      if (reason == MQRCCF_COMMAND_FAILED)
      {
         /*****************************************************************/
         /* Get the system bag handle out of the mqExecute response bag.  */
         /* This bag contains the reason from the command server why the  */
         /* command failed.                                               */
         /*****************************************************************/
         mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &resultBag, &compCode,
                      &reason);
         CheckCallResult("Get the result bag handle", compCode, reason);

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

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


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

}
```

Multi 使用事件監視器顯示事件的範例 C 程式 (amqsaiem.c)

範例 C 程式 amqsaiem.c 使用 MQAI 示範基本事件監視器。

```
/**********************************************************************/
/*                                                                    */
/* Program name: AMQSAIEM.C                                           */
/*                                                                    */
/* Description:  Sample C program to demonstrate a basic event monitor*/
/*               using the IBM MQ Admin Interface (MQAI).             */
/* Licensed Materials - Property of IBM                               */
/*                                                                    */
/* 63H9336                                                            */
/* (c) Copyright IBM Corp. 1999, 2025. All Rights Reserved.           */
/*                                                                    */
/* US Government Users Restricted Rights - Use, duplication or        */
/* disclosure restricted by GSA ADP Schedule Contract with            */
/* IBM Corp.                                                          */
/**********************************************************************/
/*                                                                    */
/* Function:                                                          */
/*    AMQSAIEM is a sample C program that demonstrates how to write a simple  */
/*    event monitor using the mqGetBag call and other MQAI calls.     */
/*                                                                    */
/*    The name of the event queue to be monitored is passed as a parameter    */
/*    to the program. This would usually be one of the system event queues:-  */
/*            SYSTEM.ADMIN.QMGR.EVENT        Queue Manager events     */
/*            SYSTEM.ADMIN.PERFM.EVENT       Performance events       */
/*            SYSTEM.ADMIN.CHANNEL.EVENT     Channel events           */
/*            SYSTEM.ADMIN.LOGGER.EVENT      Logger events            */
/*                                                                    */
/*    To monitor the queue manager event queue or the performance event queue,*/
/*    the attributes of the queue manager need to be changed to enable        */
/*    these events. For more information about this, see Part 1 of the        */
/*    Programmable System Management book. The queue manager attributes can   */
/*    be changed using either MQSC commands or the MQAI interface.            */
/*    Channel events are enabled by default.                          */
/*                                                                    */
/* Program logic                                                      */
/*    Connect to the Queue Manager.                                   */
/*    Open the requested event queue with a wait interval of 30 seconds.      */
/*    Wait for a message, and when it arrives get the message from the queue  */
/*    and format it into an MQAI bag using the mqGetBag call.         */
/*    There are many types of event messages and it is beyond the scope of    */
/*    this sample to program for all event messages. Instead the program      */
/*    prints out the contents of the formatted bag.                   */
/*    Loop around to wait for another message until either there is an error  */
/*    or the wait interval of 30 seconds is reached.                  */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* AMQSAIEM has 2 parameters - the name of the event queue to be monitored    */
/*                           - the queue manager name (optional)      */
/*                                                                    */
/**********************************************************************

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

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

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

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

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

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

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

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

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

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

   return 0;

}

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

```
                callText, cc, rc);

}

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

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

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

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

   /****************************************************************************/
   /* Set the GMO options to control the action of the get message from the    */
   /* queue.                                                                   */
   /****************************************************************************/
   gmo.WaitInterval = 30000;               /* 30 second wait for message      */
   gmo.Options = MQGMO_WAIT + MQGMO_FAIL_IF_QUIESCING + MQGMO_CONVERT;
   gmo.Version = MQGMO_VERSION_2;          /* Avoid need to reset Message ID  */
   gmo.MatchOptions = MQMO_NONE;           /* and Correlation ID after every  */
                                  /* mqGetBag
   /****************************************************************************/
   /* If open fails, we cannot access the queue and must stop the monitor.     */
   /****************************************************************************/
   if (compCode != MQCC_OK)
      bQueueOK = 0;

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

     /*************************************************************************/
     /* Get the message from the event queue and convert it into the event    */
     /* bag.                                                                  */
     /*************************************************************************/
```

```
     mqGetBag(hConn, eventQueue, &md, &gmo, eventBag, &compCode, &reason);

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

        /*****************************************************************/
        /* If get fails because no message available then we have timed out, */
        /* so report this, otherwise report an error.                    */
        /*****************************************************************/
        if (reason == MQRC_NO_MSG_AVAILABLE)
        {
           printf("No more messages\n");
        }
        else
        {
           CheckCallResult("Get bag", compCode, reason);
        }
     }

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

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

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

} /* end of GetQEvents */

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

int PrintBag(MQHBAG dataBag)
{
    int errors;

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

    return errors;
}

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

```
/* Function: PrintBagContents                                                 */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* Input Parameters:  Bag Handle                                              */
/*                    Indentation level of bag                                */
/*                                                                            */
/* Output Parameters: None                                                    */
/*                                                                            */
/* Returns:           Number of errors found                                  */
/*                                                                            */
/* Logic: Count the number of items in the bag                                */
/*        Obtain selector and item type for each item in the bag.             */
/*        Obtain the value of the item depending on item type and display the */
/*        index of the item, the selector and the value.                      */
/*        If the item is an embedded bag handle then call this function again */
/*        to print the contents of the embedded bag increasing the            */
/*        indentation level.                                                  */
/*                                                                            */
/******************************************************************************/
int PrintBagContents(MQHBAG dataBag, int indent)
{
   /*************************************************************************/
   /* Definitions                                                           */
   /*************************************************************************/
   #define LENGTH 500                      /* Max length of string to be read*/
   #define INDENT 4                        /* Number of spaces to indent     */
                                           /* embedded bag display           */


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

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

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

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

          /*********************************************************************/
          /* First inquire the type of the item for each item in the bag       */
          /*********************************************************************/
          mqInquireItemInfo(dataBag,             /* Bag handle               */
                            MQSEL_ANY_SELECTOR,  /* Item can have any selector*/
                            i,                   /* Index position in the bag */
                            &selector,           /* Actual value of selector  */
                                                 /* returned by call          */
```

```
                    &itemType,          /* Actual type of item       */
                                        /* returned by call          */
                    &compCode,          /* Completion code           */
                    &reason);           /* Reason Code               */

  if (compCode != MQCC_OK)
     errors++;

  switch(itemType)
  {
  case MQITEM_INTEGER:
       /*************************************************************/
       /* Item is an integer. Find its value and display its index,   */
       /* selector and value.                                       */
       /*************************************************************/
       mqInquireInteger(dataBag,        /* Bag handle                */
                        MQSEL_ANY_SELECTOR, /* Allow any selector     */
                        i,              /* Index position in the bag */
                        &iValue,        /* Returned integer value
                        &compCode,      /* Completion code           */
                        &reason);       /* Reason Code               */

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

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

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


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

       /*************************************************************/
       /* The call can return a warning if the string is too long for */
       /* the output buffer and has been truncated, so only check     */
       /* explicitly for call failure.                              */
       /*************************************************************/
       if (compCode == MQCC_FAILED)
           errors++;
       else
       {
          /********************************************************/
          /* Remove trailing blanks from the string and terminate with*/
          /* a null. First check that the string should not have been */
          /* longer than the maximum buffer size allowed.          */
          /********************************************************/
          if (stringLength > LENGTH)
             trimLength = LENGTH;
          else
```

```
                    trimLength = stringLength;
                 mqTrim(trimLength, stringVal, stringVal, &compCode, &reason);
                 printf("%.*s  %-2d     %-4d     '%s' %d\n",
                         indent, blanks, i, selector, stringVal, ccsid);
              }
              break;

         case MQITEM_BYTE_STRING:
              /*****************************************************************/
              /* Item is a byte string. Obtain the byte string in a buffer,  */
              /* prepare the byte string for displaying and display the      */
              /* index, selector and string.                                   */
              /*****************************************************************/
              mqInquireByteString(dataBag,      /* Bag handle                  */
                                  MQSEL_ANY_SELECTOR, /* Allow any selector  */
                                  i,            /* Index position in the bag */
                                  LENGTH,       /* Maximum length of buffer  */
                                  byteStringVal, /* Buffer to receive string */
                                  &stringLength, /* Actual length of string  */
                                  &compCode,    /* Completion code           */
                                  &reason);     /* Reason Code

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

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

                 printf("'\n");
              }
              break;

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

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

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

Multi 用於查詢通道物件的範例 C 程式 (amqsaicl.c)

範例 C 程式 `amqsaicl.c` 使用 MQAI 查詢通道物件。

```
/*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAICL.C                                                  */
/*                                                                           */
/* Description:  Sample C program to inquire channel objects                 */
/*               using the IBM MQ Administration Interface (MQAI)            */
/*                                                                           */
/* <N_OCO_COPYRIGHT>                                                         */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 2008, 2025. 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);
}
```

Multi 用於查詢佇列及列印資訊的範例 C 程式 (amqsailq.c)

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

```
/********************************************************************************/
/*                                                                              */
/* Program name: AMQSAILQ.C                                                     */
/*                                                                              */
/* Description:  Sample C program to inquire the current depth of the local     */
/*               queues using the IBM MQ Administration Interface (MQAI)        */
/*                                                                              */
/* Statement:    Licensed Materials - Property of IBM                           */
/*                                                                              */
/*               84H2000, 5765-B73                                              */
/*               84H2001, 5639-B42                                              */
/*               84H2002, 5765-B74                                              */
/*               84H2003, 5765-B75                                              */
/*               84H2004, 5639-B43                                              */
/*                                                                              */
/*               (C) Copyright IBM Corp. 1999, 2025.                            */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* Function:                                                                    */
/*    AMQSAILQ is a sample C program that demonstrates how to inquire           */
/*    attributes of the local queue manager using the MQAI interface. In        */
/*    particular, it inquires the current depths of all the local queues.       */
/*                                                                              */
/*     - A PCF command is built by placing items into an MQAI administration    */
/*       bag.                                                                   */
/*       These are:-                                                            */
/*            - The generic queue name "*"                                      */
/*            - The type of queue required. In this sample we want to           */
/*              inquire local queues.                                           */
/*            - The attribute to be inquired. In this sample we want the        */
/*              current depths.                                                 */
/*                                                                              */
/*     - The mqExecute call is executed with the command MQCMD_INQUIRE_Q.       */
/*       The call generates the correct PCF structure.                          */
/*       The default options to the call are used so that the command is sent   */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                     */
/*       The reply from the command server is placed on a temporary dynamic     */
/*       queue.                                                                 */
/*       The reply from the MQCMD_INQUIRE_Q command is read from the            */
/*       temporary queue and formatted into the response bag.                   */
/*                                                                              */
/*     - The completion code from the mqExecute call is checked and if there    */
/*       is a failure from the command server, then the code returned by        */
/*       command server is retrieved from the system bag that has been          */
/*       embedded in the response bag to the mqExecute call.                    */
/*                                                                              */
/*     - If the call is successful, the depth of each local queue is placed     */
/*       in system bags embedded in the response bag of the mqExecute call.     */
/*       The name and depth of each queue is obtained from each of the bags     */
/*       and the result displayed on the screen.                                */
/*                                                                              */
/* Note: The command server must be running.                                    */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* AMQSAILQ has 1 parameter - the queue manager name (optional)                 */
/*                                                                              */
/********************************************************************************/

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

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

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

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


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

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

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

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

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

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

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

   /**************************************************************************/
   /* Send the command to find all the local queue names and queue depths.   */
```

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


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

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

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

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

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

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

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

  /***********************************************************************/
  /* If the command fails get the system bag handle out of the mqExecute   */
  /* response bag. This bag contains the reason from the command server    */
```

```
     /* why the command failed.                                           */
     /*********************************************************************/
     if (reason == MQRCCF_COMMAND_FAILED)
     {
       mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag, &compCode,
                    &reason);
       CheckCallResult("Get the result bag handle", compCode, reason);

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

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

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

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

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

Multi 資料工具袋和 MQAI

資料工具袋是使用「IBM MQ 管理介面 (MQAI)」來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。 將這些資料項目放入袋中時，會在袋中訂購這些資料項目。 這稱為插入順序。 每一個資料項目都包含一個 選取器，用於識別資料項目以及該資料項目的 值，該值可以是整數、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 59 頁的『MQAI 中可用的資料項目類型』 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 33 頁的『在 MQAI 中編製索引』 中說明。

  圖 1 顯示這些概念的階層。

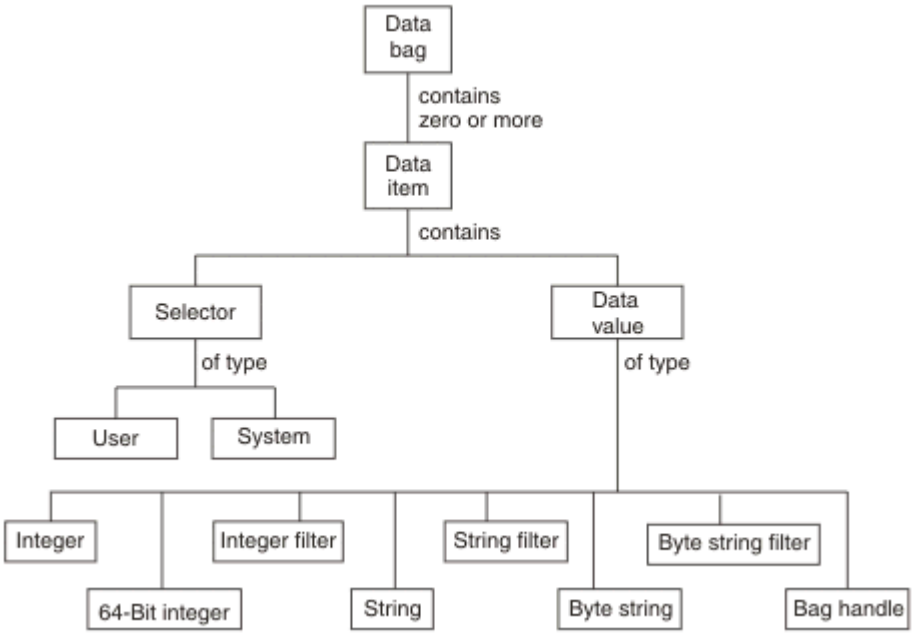


圖 *4: MQAI* 概念的階層

前一段已說明階層。

資料工具袋類型

根據您要執行的作業，您可以選擇要建立的資料工具袋類型:

使用者工具袋 (user bag)
用於使用者資料的簡式工具袋。

管理工具袋 (administration bag)
透過將管理訊息傳送至指令伺服器，為用來管理 IBM MQ 物件的資料所建立的工具袋。 管理工具袋會自動隱含某些選項，如 第 58 頁的『建立及刪除資料工具袋』 中所述。

指令工具袋 (command bag)
也會為管理 IBM MQ 物件的指令建立一個工具袋。 不過，與管理工具袋不同，指令工具袋不會自動暗示某些選項，雖然這些選項可用。 如需選項的相關資訊，請參閱 第 58 頁的『建立及刪除資料工具袋』。

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

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

使用資料工具袋本主題列出使用資料工具袋的不同方式:

使用資料工具袋

下列清單顯示使用資料袋的不同方式:

- 您可以建立及刪除資料工具袋 第 58 頁的『建立及刪除資料工具袋』。
- 您可以使用資料工具袋 第 59 頁的『使用 MQAI 放置及接收資料工具袋』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 60 頁的『使用 MQAI 將資料項目新增至工具袋』。
- 您可以在資料工具袋 第 60 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 61 頁的『在資料袋內查詢』內查詢。
- 您可以對資料工具袋 第 63 頁的『計數資料項目』內的資料項目進行計數。
- 您可以變更資料工具袋 第 61 頁的『變更工具袋內的資訊』內的資訊。
- 您可以清除資料工具袋 第 62 頁的『使用 mqClearBag 呼叫清除工具袋』。
- 您可以截斷資料工具袋 第 62 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 63 頁的『轉換袋子和緩衝器』。

Multi 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元的相鄰出現項目時，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 59 頁的『MQAI 中可用的資料項目類型』。
- 請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 mqCreateBag 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 mqDeleteBag 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 64 頁的『使用 mqExecute 呼叫將管理指令傳送至 qm 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。

Multi 使用 *MQAI* 放置及接收資料工具袋

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「IBM MQ 管理介面 (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 24 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。

Multi *MQAI* 中可用的資料項目類型

「IBM MQ 管理介面 (MQAI)」在建立資料工具袋時，會使用資料項目來移入資料工具袋。 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 59 頁的『系統項目及 MQAI』。

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 63 頁的『計數資料項目』.
- 第 63 頁的『刪除資料項目』.
- 第 60 頁的『使用 MQAI 將資料項目新增至工具袋』.
- 第 60 頁的『過濾及查詢資料項目』.

Multi 系統項目及 *MQAI*

「IBM MQ 管理介面 (MQAI)」可以將系統項目用於:

- 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內容的相關資訊，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 61 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 63 頁的『刪除資料項目』。)

Multi 使用 *MQAI* 將資料項目新增至工具袋

使用「IBM MQ 管理介面 (MQAI)」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 59 頁的『MQAI 中可用的資料項目類型』。

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 60 頁的圖 5 中。 項目由選取器識別。 通常一個選取元只會識別一個項目，但不一定如此。 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。

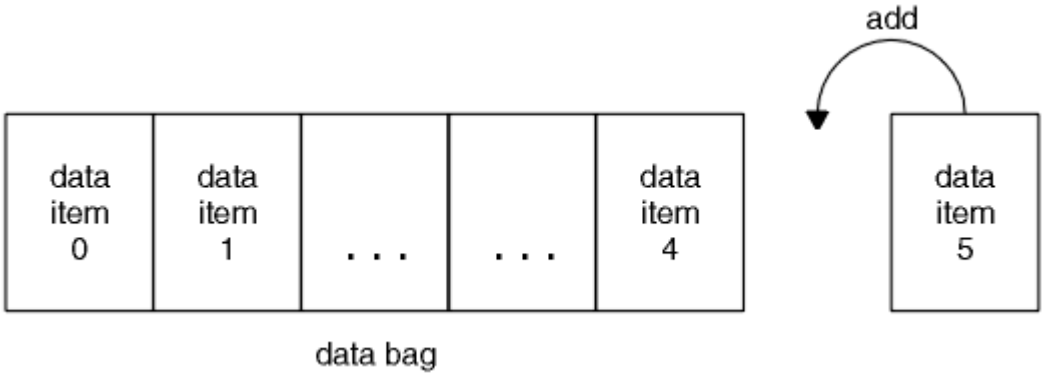


圖 *5:* 新增資料項目

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 59 頁的『系統項目及 MQAI』。

Multi 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。 通話專門用於管理目的，因此只能與管理袋搭配使用。 它可讓您指定要從 IBM MQ 查詢之屬性的選取元。

如需 mqAdd 查詢呼叫的完整說明，請參閱 mqAdd 查詢。

Multi 過濾及查詢資料項目

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

- 您可以使用 mqAddInteger 和 mqAddString 呼叫**過濾傳**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 21 頁的『IBM MQ 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 mqAddInquiry 呼叫**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, ...)
```

Multi 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。

Multi 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 61 頁的圖 6)。

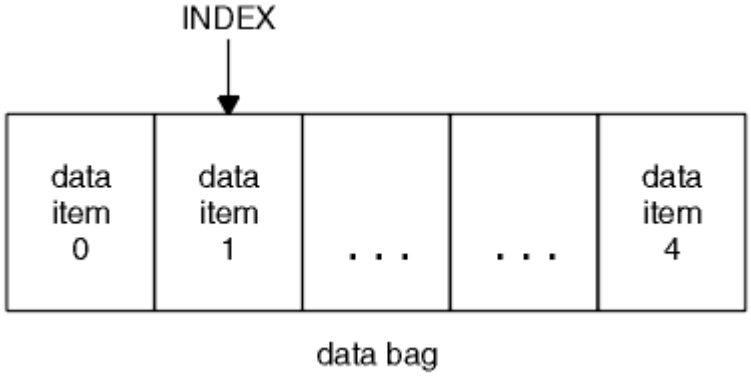


圖 *6:* 修改單一資料項目

2. 刪除所指定選取元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 62 頁的圖 7。) 特殊索引值容許取代參數的 **所有** 實例。

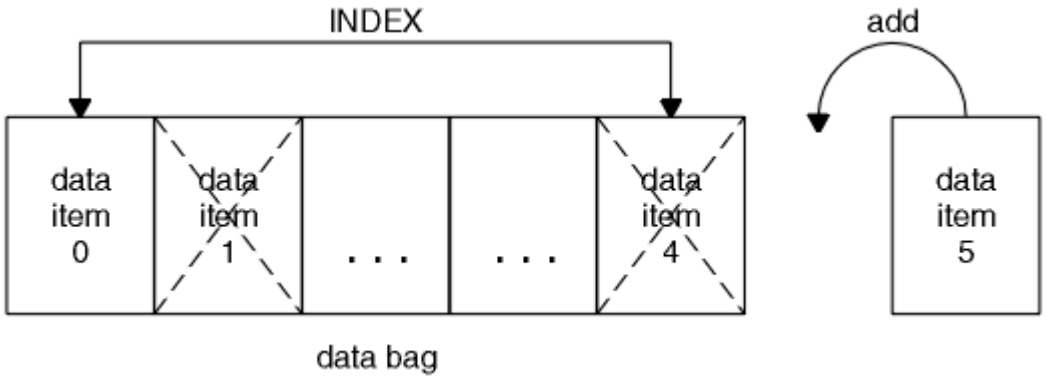


圖 7: 修改所有資料項目

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。

mqSetInteger 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。 mqSetString 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。 mqSetByteStringFilter 呼叫可讓您修改位元組字串過濾器項目。 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:

- mqSetInteger
- mqSetInteger64
- mqSetIntegerFilter
- mqSetString
- mqSetStringFilter
- mqSetByteString
- mqSetByteStringFilter

Multi 使用 *mqClearBag* 呼叫清除工具袋

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 mqClearBag 呼叫的完整說明，請參閱 mqClearBag。

Multi 使用 *mqTruncateBag* 呼叫截斷工具袋

mqTruncateBag 呼叫可透過從工具袋結尾刪除項目 (從最近新增的項目開始) 來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。

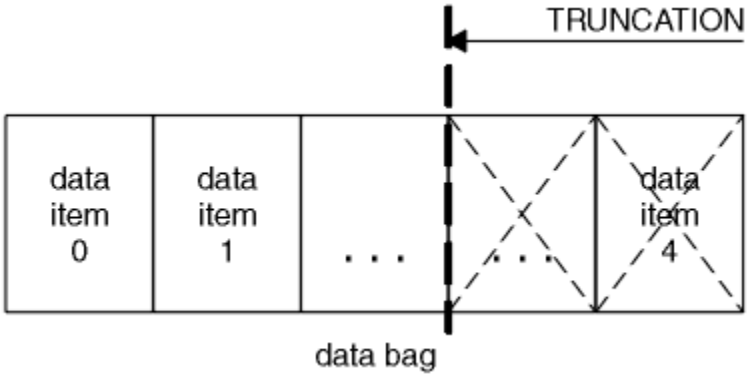


圖 8: 截斷工具袋

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

Multi 轉換袋子和緩衝器

若要在應用程式之間傳送資料，首先會將訊息資料放置在工具袋中。 然後，使用 mqBagToBuffer 呼叫將工具袋中的資料轉換為 PCF 訊息。 使用 MQPUT 呼叫將 PCF 訊息傳送至所需的佇列。 此圖 第 63 頁的圖 9 中顯示。 如需 mqBagToBuffer 呼叫的完整說明，請參閱 mqBagToBuffer。

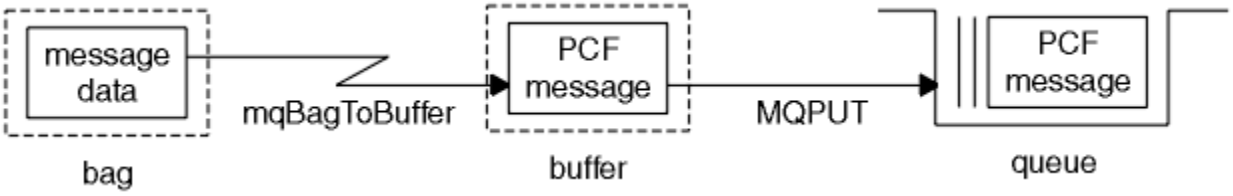


圖 *9:* 將工具袋轉換成 *PCF* 訊息

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 63 頁的圖 10 中顯示。 如需 mqBufferToBag 呼叫的完整說明，請參閱 mqBufferToBag。

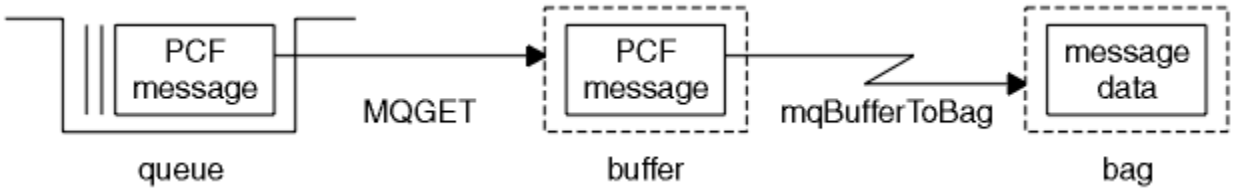


圖 *10:* 將 *PCF* 訊息轉換成工具袋表單

Multi 計數資料項目

mqCountItems 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 mqCountItems 呼叫的完整說明，請參閱 mqCountItems。

Multi 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。 如需詳細資訊，請參閱 第 63 頁的『使用 mqDeleteItem 呼叫從工具袋刪除資料項目』。
- 從工具袋中刪除所有使用者項目，亦即清除工具袋。 如需詳細資訊，請參閱 第 62 頁的『使用 mqClearBag 呼叫清除工具袋』。
- 從工具袋結尾刪除使用者項目，即截斷工具袋。 如需詳細資訊，請參閱 第 62 頁的『使用 mqTruncateBag 呼叫截斷工具袋』。

Multi 使用 *mqDeleteItem* 呼叫從工具袋刪除資料項目

mqDeleteItem 呼叫會從工具袋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 64 頁的圖 11。)

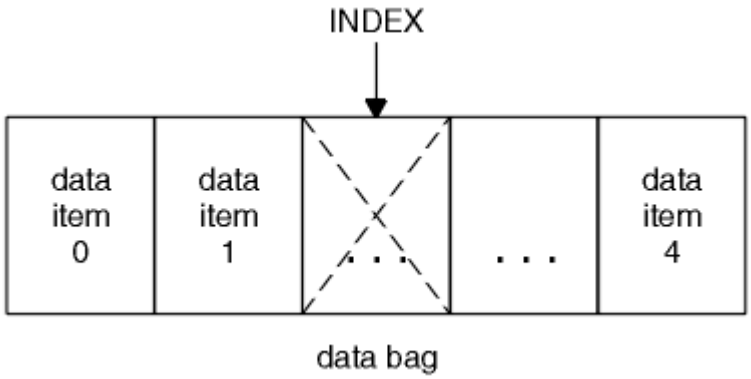


圖 *11:* 刪除單一資料項目

or

2. 指定選取元的所有出現項目。 (請參閱 第 64 頁的圖 12。)

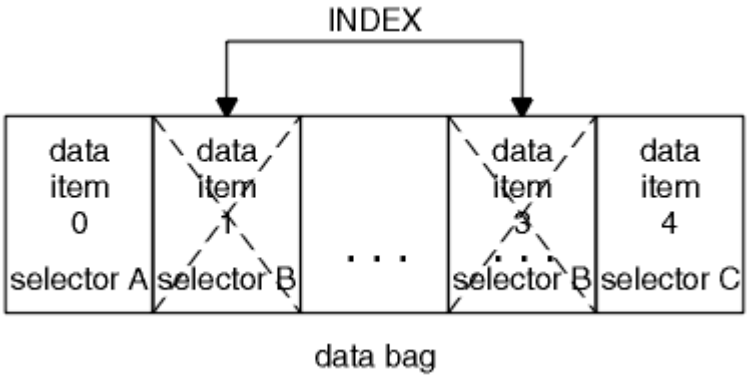


圖 *12:* 刪除所有資料項目

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， mqDeleteItem 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 mqDeleteItem 呼叫的完整說明，請參閱 mqDeleteItem。

Multi 使用 mqExecute 呼叫將管理指令傳送至 qm 指令伺服器

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

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

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

巢狀 的概念顯示在 第 65 頁的圖 13 中。

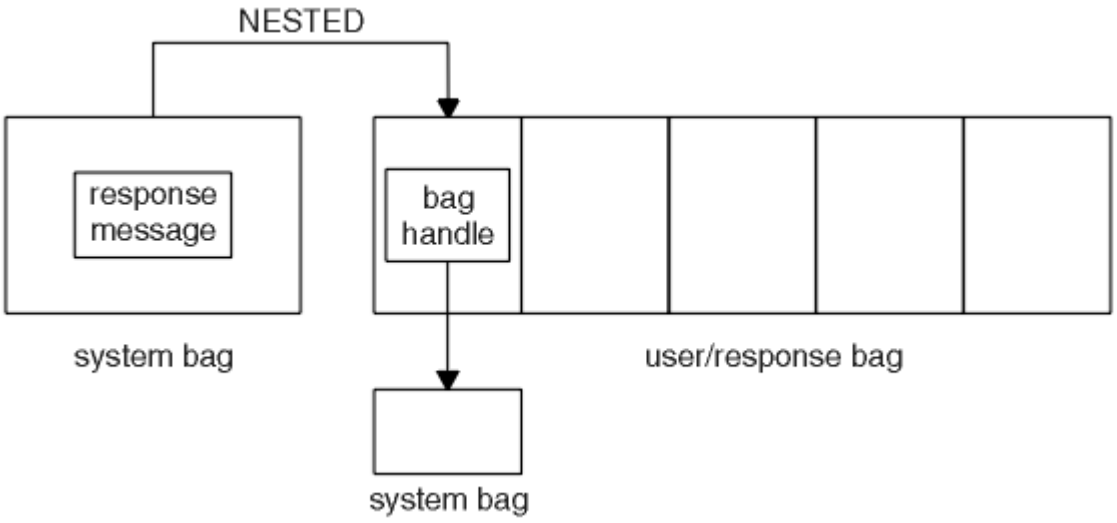


圖 *13:* 巢狀*(N)*

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。

下列控點是選用的:

- 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 REST API 進行管理

您可以使用 administrative REST API 來管理 IBM MQ 物件，例如佇列管理程式和佇列，以及 Managed File Transfer 代理程式和傳送。 資訊會以 JSON 格式傳送至 administrative REST API ，並從中接收。 這些 RESTful API 可協助您將 IBM MQ 管理內嵌至熱門 DevOps 及自動化工具。

開始之前

如需可用 REST 資源的相關參照資訊，請參閱 administrative REST API 參照。

程序

- 第 66 頁的『「administrative REST API」入門』
- 第 69 頁的『使用 administrative REST API』
- 第 70 頁的『使用 REST API 進行遠端管理』

- 第 74 頁的『REST API 時間戳記』
- 第 74 頁的『REST API 錯誤處理』
- 第 76 頁的『REST API 探索』
- 第 77 頁的『REST API 國家語言支援』

「administrative REST API」入門

快速開始使用 administrative REST API，並使用 cURL 來嘗試一些範例要求，以建立、更新、檢視及刪除佇列。

開始之前

為了讓您開始使用 administrative REST API，此作業中的範例具有下列需求:

- 這些範例使用 cURL 來提出 REST 要求，以顯示系統上佇列管理程式的相關資訊，以及建立佇列、更新、檢視及刪除佇列。 因此，若要完成此作業，您需要在系統上安裝 cURL 。
- 若要完成此作業，您必須是具有某些特權的使用者，如此才能使用 **dspmqweb** 指令:
  - z/OS 在 z/OS 上，您必須有權執行 **dspmqweb** 指令，並且具有 `mqwebuser.xml` 檔的寫入權。
  - Multi 在所有其他作業系統上，必須是 特許使用者。

IBM i 在 IBM i 上，指令應該在 QSHELL 中執行。

程序

1. 如果尚未配置 mqweb 伺服器以供 administrative REST API、 administrative REST API for MFT、messaging REST API 或 IBM MQ Console 使用，請配置 mqweb 伺服器。
   如需使用基本登錄為 mqweb 伺服器建立基本配置的相關資訊，請參閱 mqweb 伺服器的基本配置。
2. z/OS
   在 z/OS 上，設定 WLP_USER_DIR 環境變數，以便您可以使用 **dspmqweb** 指令。 透過輸入下列指令，將變數設定為指向您的 mqweb 伺服器配置:

   ```
   export WLP_USER_DIR=WLP_user_directory
   ```

   ，其中 *WLP_user_directory* 是傳遞至 `crtmqweb` 的目錄名稱。 例如:

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

   如需相關資訊，請參閱建立 mqweb 伺服器。
3. 輸入下列指令，確定 REST API URL :

   ```
   dspmqweb status
   ```

   以下步驟中的範例假設您的 REST API URL 是預設的 URL `https://localhost:9443/ibmmq/rest/v1/` 。 如果您的 URL 不同於預設值，請在下列步驟中使用您的 URL 替換預設值。
4. 使用 `mqadmin` 使用者的基本鑑別，在 `qmgr` 資源上試用 GET 要求:

   ```
   curl -k https://localhost:9443/ibmmq/rest/v2/admin/qmgr -X GET -u mqadmin:mqadmin
   ```

5. 使用 `mqsc` 資源來建立、顯示、變更及刪除佇列:

   此範例使用佇列管理程式 `QM1`。 請建立同名的佇列管理程式，或替換系統上現有的佇列管理程式。

   a) 在 `mqsc` 資源上提出 POST 要求，以建立本端佇列:

   在要求的內文中，新佇列的名稱會設為 `Q1`。 使用基本鑑別，並在 cURL REST 要求中設定具有任意值的 `ibm-mq-rest-csrf-token` HTTP 標頭。 POST、PATCH 及 DELETE 要求需要此額外標頭:

   ```
   curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
   mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json" --
   ```

```
data "{\"type\": \"runCommandJSON\", \"command\": \"define\", \"qualifier\": \"qlocal\",
\"name\": \"Q1\"}"
```

b) 在 mqsc 資源上提出 POST 要求，以檢視在步驟 第 66 頁的『5.a』中建立的本端佇列:

```
curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json" --
data "{\"type\": \"runCommandJSON\", \"command\": \"display\", \"qualifier\": \"qlocal\",
\"name\": \"Q1\"}"
```

c) 對 mqsc 資源發出 POST 要求，以更新佇列的說明:

```
curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json" --
data "{\"type\": \"runCommandJSON\", \"command\": \"alter\", \"qualifier\": \"qlocal\",
\"name\": \"Q1\", \"parameters\": {\"descr\": \"new description\" }}"
```

d) 對 mqsc 資源提出 POST 要求，以檢視新的佇列說明。 在要求內文中指定 **responseParameters** 屬性，以便回應包含說明欄位:

```
curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json" --
data "{\"type\": \"runCommandJSON\", \"command\": \"display\", \"qualifier\": \"qlocal\",
\"name\": \"Q1\", \"responseParameters\" : [\"descr\"]}"
```

e) 對 mqsc 資源提出 POST 要求以刪除佇列:

```
curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json" --
data "{\"type\": \"runCommandJSON\", \"command\": \"delete\", \"qualifier\": \"qlocal\",
\"name\": \"Q1\"}"
```

f) 對 mqsc 資源提出 POST 要求，以證明已刪除佇列:

```
curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json" --
data "{\"type\": \"runCommandJSON\", \"command\": \"display\", \"qualifier\": \"qlocal\",
\"name\": \"Q1\"}"
```

下一步

- 這些範例使用基本鑑別來保護要求。 您可以改為使用記號型鑑別或用戶端型鑑別。 如需相關資訊，請參閱使用用戶端憑證鑑別與 REST API 及 IBM MQ Console 及 使用記號型鑑別與 REST API。
- 進一步瞭解使用 administrative REST API 及使用查詢參數來建構 URL: 第 69 頁的『使用 administrative REST API』。
- 瀏覽可用的 administrative REST API 資源及所有可用的選用查詢參數的參照資訊: administrative REST API reference。
- 瞭解如何使用 administrative REST API 來管理遠端系統上的 IBM MQ 物件: 第 70 頁的『使用 REST API 進行遠端管理』。
- 瞭解如何將 administrative REST API 與 MFT 搭配使用:第 67 頁的『REST API for MFT 入門』。
- 探索 messaging REST API，這是 IBM MQ 傳訊的 RESTful 介面: 使用 REST API 進行傳訊。
- 探索 IBM MQ Console，瀏覽器型 GUI: 第 80 頁的『使用 Web 主控台進行管理』。

REST API for MFT 入門

快速開始使用 administrative REST API for Managed File Transfer ，並嘗試一些範例要求，以檢視 MFT 代理程式狀態，以及檢視傳送清單。

開始之前

- 這些範例使用 cURL 來傳送 REST 要求，以檢視傳送清單及檢視 MFT 代理程式狀態。 因此，若要完成此作業，您需要在系統上安裝 cURL 。
- 若要完成此作業，您必須是具有某些特權的使用者，如此才能使用 **dspmqweb** 指令:

– z/OS 在 z/OS 上，您必須有權執行 **dspmqweb** 指令，並且具有 mqwebuser.xml 檔的寫入權。

– Multi 在所有其他作業系統上，必須是 特許使用者。

程序

1. 確保已針對 administrative REST API for MFT 配置 mqweb 伺服器:
   - 如果尚未配置 mqweb 伺服器以供 administrative REST API、 administrative REST API for MFT、messaging REST API 或 IBM MQ Console 使用，請配置 mqweb 伺服器。如需使用基本登錄為 mqweb 伺服器建立基本配置的相關資訊，請參閱 mqweb 伺服器的基本配置。
   - 如果已配置 mqweb 伺服器，請確保已完成 mqweb 伺服器的基本配置 的步驟 8 ， 以啟用 administrative REST API for MFT。
2. z/OS
   在 z/OS 上，設定 WLP_USER_DIR 環境變數，以便您可以使用 **dspmqweb** 指令。 透過輸入下列指令，將變數設定為指向您的 mqweb 伺服器配置:
   ```
   export WLP_USER_DIR=WLP_user_directory
   ```
   ，其中 *WLP_user_directory* 是傳遞至 crtmqweb 的目錄名稱。 例如:
   ```
   export WLP_USER_DIR=/var/mqm/web/installation1
   ```
   如需相關資訊，請參閱建立 mqweb 伺服器。
3. 輸入下列指令，確定 REST API URL：
   ```
   dspmqweb status
   ```
   以下步驟中的範例假設您的 REST API URL 是預設的 URL `https://localhost:9443/ibmmq/rest/v1/` 。 如果您的 URL 不同於預設值，請在下列步驟中使用您的 URL 替換預設值。
4. 對 agent 資源提出 GET 要求，以傳回所有代理程式的基本詳細資料，包括名稱、類型及狀態:
   ```
   curl -k https://localhost:9443/ibmmq/rest/v2/admin/mft/agent/ -X GET -u mftadmin:mftadmin
   ```
5. 使用 **fteCreateTransfer** 指令建立要顯示的部分傳送。
   mqweb 伺服器會快取傳送的相關資訊，並在提出要求時傳回此資訊。 當 mqweb 伺服器重新啟動時，會重設此快取。 您可以檢視 console.log 及 messages.log 檔案，或在 z/OS 上查看已啟動作業的輸出，以查看伺服器是否已重新啟動。
6. 在 transfer 資源上發出 GET 要求，以傳回自 mqweb 伺服器啟動以來所進行最多四個傳送的詳細資料:
   ```
   curl -k https://localhost:9443/ibmmq/rest/v2/admin/mft/transfer?limit=4 -X GET -u
   mftadmin:mftadmin
   ```

下一步

- 這些範例使用基本鑑別來保護要求。 您可以改為使用記號型鑑別或用戶端型鑑別。 如需相關資訊，請參閱將記號型鑑別與 REST API 一起使用以及將用戶端憑證鑑別與 REST API 和 IBM MQ Console 一起使用。
- 進一步瞭解使用 administrative REST API 及使用查詢參數來建構 URL: 第 69 頁的『使用 administrative REST API』。
- 瀏覽可用的 administrative REST API for MFT 資源及所有可用的選用查詢參數的參照資訊: administrative REST API reference。
- 探索 messaging REST API，這是 IBM MQ 傳訊的 RESTful 介面: 使用 REST API 進行傳訊。
- 探索 IBM MQ Console，瀏覽器型 GUI: 第 80 頁的『使用 Web 主控台進行管理』。

使用 administrative REST API

當您使用 administrative REST API 時，您可以對代表各種 IBM MQ 物件 (例如佇列管理程式或佇列) 的 URL 呼叫 HTTP 方法。 HTTP 方法 (例如 POST) 代表要對 URL 所代表物件執行的動作類型。 動作的進一步相關資訊可以在 JSON 中作為 HTTP 方法有效負載的一部分提供，或在查詢參數中編碼。 可能傳回執行動作之結果的相關資訊作為 HTTP 回應的主體。

開始之前

在使用 administrative REST API 之前，請考量下列事項:

- 您必須向 mqweb 伺服器進行鑑別，才能使用 administrative REST API。 您可以使用 HTTP 基本鑑別、用戶端憑證鑑別或記號型鑑別來進行鑑別。 如需如何使用這些鑑別方法的相關資訊，請參閱 IBM MQ 主控台和 REST API 安全。
- REST API 區分大小寫。 例如，如果佇列管理程式稱為 `qmgr1`，則下列 URL 上的 HTTP GET 不會顯示資訊。

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

- 並非所有可以在 IBM MQ 物件名稱中使用的字元都可以直接在 URL 中編碼。 如果要正確編碼這些字元，您必須使用適當的 URL 編碼:
  - 正斜線/必須編碼為 %2F。
  - 百分比符號% 必須編碼為 %25。
- 由於某些瀏覽器的行為，請勿只使用句點或正斜線字元來命名物件。

關於這項作業

當您使用 REST API 對物件執行動作時，首先需要建構 URL 來代表該物件。 每一個 URL 都以字首開頭，說明要將要求傳送至哪一個主機名稱和埠。 URL 的其餘部分說明特定物件或一組物件 (稱為資源)。

要在資源上執行的動作會定義 URL 是否需要查詢參數。 它也定義所使用的 HTTP 方法，以及是否以 JSON 形式將其他資訊傳送至 URL 或從中傳回。 其他資訊可能構成 HTTP 要求的一部分，或作為 HTTP 回應的一部分傳回。

建構 URL 並建立選用的 JSON 有效負載以在 HTTP 要求中傳送之後，您可以將 HTTP 要求傳送至 IBM MQ。 您可以使用內建在您選擇的程式設計語言中的 HTTP 實作來傳送要求。 您也可以使用指令行工具 (例如 cURL 或 Web 瀏覽器或 Web 瀏覽器附加程式) 來傳送要求。

重要: 您至少必須執行步驟 第 69 頁的『1.a』 和 第 69 頁的『1.b』。

程序

1. 建構 URL:
   a) 輸入下列指令來決定字首 URL:

   ```
   dspmqweb status
   ```

   您要使用的 URL 包括 `/ibmmq/rest/` 詞組。
   b) 將資源新增至 URL 路徑。

   下列 IBM MQ 資源可用:

   - /admin/installation
   - /admin/qmgr
   - /admin/queue
   - /admin/subscription
   - /admin/channel
   - /action/qmgr/{qmgrname}/mqsc

下列 Managed File Transfer 資源可用:

- /admin/agent
- /admin/transfer
- /admin/monitor

例如，若要與佇列管理程式互動，請將 /qmgr 新增至字首 URL ，以建立下列 URL:

```
https://localhost:9443/ibmmq/rest/v2/admin/qmgr
```

c) 選擇性的: 將任何其他選用路徑區段新增至 URL。

在每一個物件類型的參照資訊中，可以在 URL 中以大括弧 { }來識別選用區段。

例如，將佇列管理程式名稱 QM1 新增至 URL ，以建立下列 URL:

```
https://localhost:9443/ibmmq/rest/v2/admin/qmgr/QM1
```

d) 選擇性的: 將選用查詢參數新增至 URL。

新增問號，? ， 變數名稱、等號 = ，以及 URL 的值或值清單。

例如，若要要求佇列管理程式 QM1 的所有屬性，請建立下列 URL:

```
https://localhost:9443/ibmmq/rest/v2/admin/qmgr/QM1?attributes=*
```

e) 將進一步選用查詢參數新增至 URL。

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

2. 在 URL 上呼叫相關 HTTP 方法。 指定任何選用的 JSON 有效負載，並提供適當的安全認證以進行鑑別。 例如:
   - 使用所選程式設計語言的 HTTP/REST 實作。
   - 使用 REST 用戶端瀏覽器附加程式或 cURL 之類的工具。

使用 REST API 進行遠端管理

您可以使用「 REST API 」來管理遠端佇列管理程式，以及與那些佇列管理程式相關聯的 IBM MQ 物件。 此遠端管理包括位於相同系統上，但與 mqweb 伺服器不在相同 IBM MQ 安裝中的佇列管理程式。 因此，您可以使用 REST API 來管理整個 IBM MQ 網路，只有一個執行 mqweb 伺服器的安裝架構。 若要管理遠端佇列管理程式，您必須配置 administrative REST API 閘道，以便在與 mqweb 伺服器相同的安裝中至少有一個佇列管理程式充當閘道佇列管理程式。 然後，您可以在 REST API 資源 URL 中指定遠端佇列管理程式，以執行指定的管理動作。

開始之前

您可以透過停用 administrative REST API 閘道來防止遠端管理。 如需相關資訊，請參閱 配置 administrative REST API 閘道。

若要使用 administrative REST API 閘道，必須符合下列條件:

- 必須配置並啟動 mqweb 伺服器。 如需配置及啟動 mqweb 伺服器的相關資訊，請參閱 第 66 頁的『「administrative REST API」入門』。
- 您要配置為閘道佇列管理程式的佇列管理程式必須與 mqweb 伺服器位於相同的安裝中。
- 您要管理的遠端佇列管理程式必須是 IBM MQ 8.0 或更新版本。
- 您必須確定在要求中指定的任何屬性對於您要將要求傳送至其中的系統而言都是有效的。 例如，如果閘道佇列管理程式位於 Windows 上，遠端佇列管理程式位於 z/OS 上，則無法要求針對 queue 資源上的 HTTP GET 要求傳回 dataCollection.statistics 屬性。
- 您必須確保在要求中指定的任何屬性對於您要將要求傳送至其中的 IBM MQ 層次有效。 例如，如果遠端佇列管理程式正在執行 IBM MQ 8.0，則無法要求針對 queue 資源上的 HTTP GET 要求傳回 extended.enableMediaImageOperations 屬性。

- 您必須使用下列其中一個支援的 REST 資源:
  - /queue
  - /subscription
  - /channel
  - /mqsc
  - /qmgr

    當您查詢遠端佇列管理程式時， /qmgr 資源只會傳回部分屬性: name、 status.started、 status.channelInitiatorState、 status.ldapConnectionState、 status.connectionCount 及 status.publishSubscribeState。

關於這項作業

若要使用 administrative REST API 閘道來管理遠端佇列管理程式，您必須準備佇列管理程式以進行遠端管理。 也就是說，您必須在閘道佇列管理程式與遠端佇列管理程式之間配置傳輸佇列、接聽器以及傳送端和接收端通道。 然後，您可以透過在資源 URL 中指定佇列管理程式，將 REST 要求傳送至遠端佇列管理程式。 透過使用 **setmqweb** 指令將 mqRestGatewayQmgr 屬性設為閘道佇列管理程式的名稱，或在隨要求一起傳送的標頭中傳送閘道佇列管理程式的名稱，來指定閘道佇列管理程式。 要求會透過閘道佇列管理程式傳送到遠端佇列管理程式。 傳回的回應的標頭指示用作閘道佇列管理程式的佇列管理程式。

程序

1. 配置閘道佇列管理程式與您要管理的遠端佇列管理程式之間的通訊。 這些配置步驟與透過 runmqsc 和 PCF 配置遠端管理所需的步驟相同。

   如需這些步驟的相關資訊，請參閱 第 163 頁的『配置佇列管理程式以進行遠端管理』。
2. 在遠端佇列管理程式上配置安全:

   a) 請確定遠端佇列管理程式執行所在的系統上存在相關使用者 ID。 遠端系統上必須存在的使用者 ID 取決於 REST API 使用者的角色:

   - 如果 REST API 使用者位於 MQWebAdmin 或 MQWebAdminRO 群組中，則啟動 mqweb 伺服器的使用者 ID 必須存在於遠端系統上。 在 IBM MQ Appliance 上，啟動 mqweb 伺服器的使用者為 mqsystem。
   - 如果 REST API 使用者位於 MQWebUser 群組中，則該 REST API 使用者 ID 必須存在於遠端系統上。

   b) 確保授與相關使用者 ID 必要的權限層次，以存取遠端佇列管理程式上的適當 REST API 資源:

   - 將訊息放置到 SYSTEM.ADMIN.COMMAND.QUEUE 的權限。
   - 將訊息放置到 SYSTEM.REST.REPLY.QUEUE 的權限。
   - 存取為遠端管理定義之傳輸佇列的權限。
   - 顯示佇列管理程式屬性的權限。
   - 執行 REST 要求的權限。 如需相關資訊，請參閱 REST API 資源參考主題的「安全需求」小節。
3. 配置使用哪個本端佇列管理程式作為閘道。 您可以配置預設閘道佇列管理程式、在 HTTP 標頭中指定閘道佇列管理程式，或使用兩種方法的組合:

   - 使用 **setmqweb** 指令來配置預設閘道佇列管理程式:

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

     其中 *qmgrName* 是閘道佇列管理程式的名稱。

     當下列兩個陳述式都成立時，會使用此閘道佇列管理程式:

     - REST 要求的 ibm-mq-rest-gateway-qmgr 標頭中未指定佇列管理程式。
     - 在 REST API 資源 URL 中指定的佇列管理程式不是本端佇列管理程式。
   - 透過將 HTTP 標頭 ibm-mq-rest-gateway-qmgr 設定為閘道佇列管理程式的名稱，在每個 REST 要求上配置閘道佇列管理程式。

4. 在資源 URL 中包含您要管理的遠端佇列管理程式名稱。
   例如，若要從遠端佇列管理程式 remoteQM 取得佇列清單，請使用下列 URL:

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

結果

ibm-mq-rest-gateway-qmgr 標頭會隨 REST 回應一起傳回。 此標頭指定哪個佇列管理程式用作閘道佇列管理程式。

如果您無法使用「 administrative REST API 」來管理遠端佇列管理程式:

- 請檢查遠端佇列管理程式是否在執行中。
- 請檢查指令伺服器是否在遠端系統上執行。
- 請檢查通道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。

範例

在下列範例中，兩部機器上有三個 IBM MQ 安裝架構。 在 Machine 1 上，有一個 Installation 1 和一個 Installation 2。 在 Machine 2 上，有一個 Installation 3。 mqweb 伺服器已配置給 Installation 1。 每一個安裝中都有單一佇列管理程式，且這些佇列管理程式已配置用於遠端管理。 也就是說，會配置並啟動下列接聽器、通道及佇列:

- 在佇列管理程式 QM1、 Installation 1、 Machine 1 上:
  - 傳送端通道 QM1.to.QM2
  - 接收端通道 QM2.to.QM1
  - 傳送端通道 QM1.to.QM3
  - 接收端通道 QM3.to.QM1
  - 傳輸佇列 QM2
  - 傳輸佇列 QM3
  - 在埠 1414 上配置的接聽器
- 在佇列管理程式 QM2 上，在 Installation 2 中，在 Machine 1 上:
  - 傳送端通道 QM2.to.QM1
  - 接收端通道 QM1.to.QM2
  - 傳輸佇列 QM1
  - 在埠 1415 上配置的接聽器
- 在佇列管理程式 QM3 上，在 Installation 3 中，在 Machine 2 上:
  - 傳送端通道 QM3.to.QM1
  - 接收端通道 QM1.to.QM3
  - 傳輸佇列 QM1
  - 預設接聽器

佇列 Qon2 定義在 QM2 上，而佇列 Qon3 定義在 QM3 上。

使用者 mquser 在兩部機器上定義，在 REST API 中獲授與 MQWebAdmin 角色，並獲授與權限來存取每一個佇列管理程式上的適當佇列。

setmqweb 指令用來將佇列管理程式 QM1 配置為預設閘道佇列管理程式。

下圖顯示此配置:

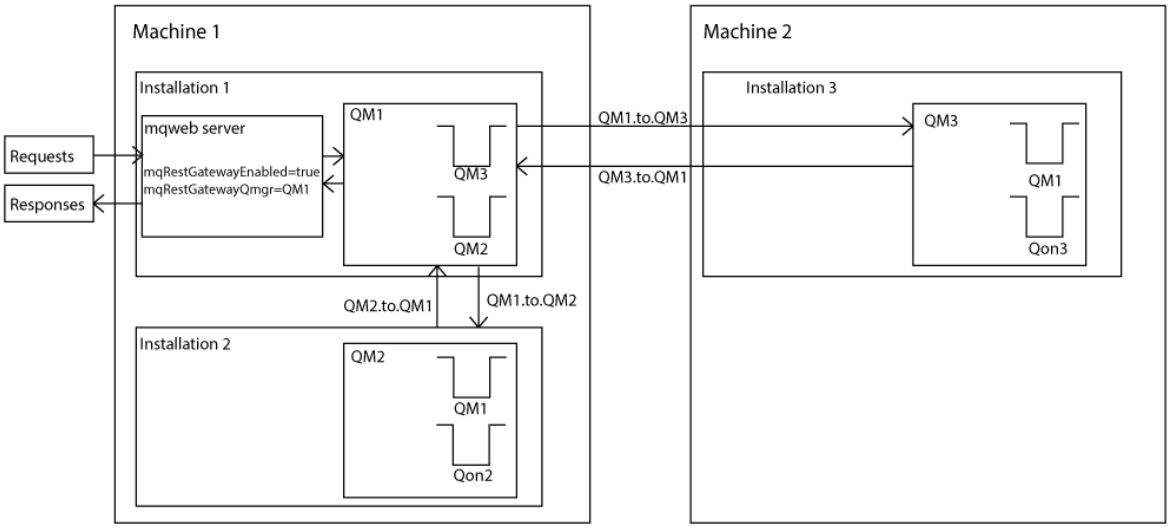


圖 *14:* 使用 *REST API* 進行遠端管理的範例配置圖。

下列 REST 要求會傳送至 mqweb 伺服器:

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

收到下列回應:

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

下列 REST 要求會傳送至 mqweb 伺服器:

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

收到下列回應:

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

```
        },
        "name" : "Qon3",
        "type" : "local"
    }]
}
```

REST API 時間戳記

當 administrative REST API 傳回日期和時間資訊時，它會以「世界標準時間 (UTC)」及設定格式傳回。

會以下列時間戳記格式傳回日期和時間:

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

例如， `2012-04-23T18:25:43.000Z`，其中 Z 指出時區是「世界標準時間 (UTC)」。

無法保證此時間戳記的正確性。 例如，如果 mqweb 伺服器未在與資源 URL 中指定的佇列管理程式相同的時區中啟動，則時間戳記可能不精確。 此外，如果需要調整「日光節約時間」，則時間戳記可能不正確。

REST API 錯誤處理

REST API 會傳回適當的 HTTP 回應碼 (例如 404 (找不到)) 及 JSON 回應來報告錯誤。 任何不在 200-299 範圍內的 HTTP 回應碼都視為錯誤。

錯誤回應格式

回應採用 UTF-8 編碼的 JSON 格式。 它包含巢狀 JSON 物件:

- 外部 JSON 物件，包含稱為 `error` 的單一 JSON 陣列。
- 陣列中的每一個元素都是 JSON 物件，代表錯誤的相關資訊。 每一個 JSON 物件都包含下列內容:

type
字串。

錯誤的類型。

messageId
字串。

MQWBnnnnX 格式之訊息的唯一 ID。 此 ID 具有下列元素:

MQWB
顯示訊息源自 IBM MQ Rest API 的字首。

nnnn
識別訊息的唯一號碼。

X
表示訊息嚴重性的單一字母:

- I，如果訊息純粹是參考資訊。
- W 如果訊息是問題的警告。
- E 表示訊息指出發生錯誤。
- S 如果訊息指出發生嚴重錯誤。

訊息
字串。

錯誤的說明。

說明
字串。

錯誤的說明。

action
字串。

可用來解決錯誤的步驟說明。

qmgrName

z/OS 此欄位僅適用於佇列管理程式是佇列共用群組成員的 z/OS 。 您必須已指定 **commandScope** 選用查詢參數或 **queueSharingGroupDisposition** 屬性。

字串。

發生錯誤的佇列管理程式名稱。

此欄位不適用於 messaging REST API。

completionCode

僅當 **type** 為 `pcf`、 `java` 或 `rest` 時，此欄位才可用。

數字。

與失敗相關聯的 MQ 完成碼。

reasonCode

僅當 **type** 為 `pcf`、 `java` 或 `rest` 時，此欄位才可用。

數字。

與失敗相關聯的 MQ 原因碼。

異常狀況

僅當 **type** 為 `java` 時，此欄位才可用。

陣列。

鏈結 Java 或 JMS 異常狀況的陣列。 異常狀況陣列的每一個元素都包含一個 **stackTrace** 字串陣列。

stackTrace 字串陣列包含分割成幾行的每一個異常狀況的詳細資料。

V 9.2.0 從 IBM MQ 9.1.2 開始，不再傳回此欄位。

佇列共用群組發生錯誤

z/OS

在佇列共用群組中，可以為某些指令指定 **commandScope** 的選用查詢參數。 這個參數可讓指令延伸到佇列共用群組中的其他佇列管理程式。 其中任何一個指令都可以獨立失敗，導致佇列共用群組的部分指令成功及部分指令失敗。

如果指令部分失敗，則會傳回 HTTP 錯誤碼 500。 對於產生失敗的每一個佇列管理程式，該失敗的相關資訊會以 `error` JSON 陣列中的元素形式傳回。 對於順利執行指令的每一個佇列管理程式，會以 `success` JSON 陣列中的元素形式傳回佇列管理程式的名稱。

範例

- 下列範例顯示嘗試取得不存在的佇列管理程式相關資訊的錯誤回應:

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

- z/OS 下列範例顯示嘗試刪除佇列共用群組中某些佇列管理程式不存在的佇列的錯誤回應:

```
"error" : [
  {
    "type": "rest",
    "messageId": "MQWB0037E",
    "message": "MQWB0037E: Could not find the queue 'missingQueue' - the queue manager reason
```

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

MFT 要求的錯誤

如果未啟用 MFT REST API 服務，且您呼叫 MFT REST API，則會收到下列異常狀況:

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

如果已啟用 MFT REST API 服務，且未在 `mqwebuser.xml` 檔案中設定協調佇列管理程式，則您會收到下列異常狀況:

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

REST API 探索

REST API 的 Documentation 在 IBM Documentation 內以及 Swagger 格式中提供。 Swagger 是用來記載 REST API 的常用方法。 透過在 mqweb 伺服器上啟用 API 探索特性，可以檢視 REST API 的 Swagger 文件。

開始之前

重要: 「API 探索」特性已穩定，您仍然可以使用此特性。 目前， IBM MQ 不支援使用 mpOpenAPI 特性。

您必須啟用 mqweb 伺服器的安全，才能使用 API 探索來檢視 Swagger 文件。 如需啟用安全所需之步驟的相關資訊，請參閱 IBM MQ Console 和 REST API 安全。

程序

1. 在下列其中一個目錄中找到 `mqwebuser.xml` 檔案:
   - ALW *MQ_DATA_PATH*/web/installations/*installationName*/servers/mqweb
   - z/OS *WLP_user_directory*/servers/mqweb

其中 *WLP_user_directory* 是執行 **crtmqweb** Script 以建立 mqweb 伺服器定義時指定的目錄。

2. 將適當的 XML 新增至 mqwebuser.xml 檔案:
   - 如果 <featureManager> 標籤存在於 mqwebuser.xml 檔案中，請在 <featureManager> 標籤內新增下列 XML:

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

   - 如果 <featureManager> 標籤不存在於 mqwebuser.xml 檔案中，請在 <server> 標籤內新增下列 XML:

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

3. 使用下列其中一種方法來檢視 Swagger 文件:
   - 在瀏覽器中輸入下列 URL，以顯示您可以瀏覽並試用 REST API 的網頁:

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

     除了鑑別每一個要求之外，您還必須包括每一個 POST、PATCH 或 DELETE 要求的 ibm-mq-rest-csrf-token 標頭。 此標頭的內容可以是任何字串，包括空白。

     這個要求標頭用來確認認證的擁有者正在使用用來鑑別要求的認證。 也就是說，記號是用來防止偽造跨網站要求攻擊。

   - 透過對下列 URL 發出 HTTP GET，擷取說明整個 REST API 的單一 Swagger 2 文件:

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

     此文件可用於您要以程式化方式導覽可用 API 的應用程式。

host
指定 REST API 可用的主機名稱或 IP 位址。

預設值為 localhost。

port
指定 administrative REST API 使用的 HTTP 埠號。

預設值為 9443。

如果主機名稱或埠號從預設值變更，您可以從 REST API URL 判定正確的值。 使用 **dspmqweb status** 指令來檢視 URL。

REST API 國家語言支援

REST API 以特定資格支援在 HTTP 要求中指定國家語言的能力。

背景

HTTP 標頭 容許在要求上指定特定行為，並在回應中提供其他資訊。

HTTP 標頭中包含要求以國家語言傳回資訊的能力。 可能的話， REST API 允許使用此標頭。

指定國家語言

在 ACCEPT-LANGUAGE HTTP 標頭中，可以提供一或多個語言標籤。 您可以選擇性地將等級與標籤相關聯，以容許指定依喜好設定排序的清單。 此頁面 提供原則的有用討論。

REST API 允許使用此標頭，從 ACCEPT-LANGUAGE 標頭中選取語言並以該語言傳回訊息。 當 ACCEPT-LANGUAGE 標頭不包含 REST API 可支援的語言時，會以預設語言傳回訊息。 這個預設語言對應於 REST API Web 伺服器的預設語言環境。

第 78 頁的『轉換哪些資料?』 一節說明轉換的資料。

指出回應的適用語言

來自 REST API 的回應上的 CONTENT-LANGUAGE HTTP 標頭指出傳回訊息的語言。

轉換哪些資料?

錯誤及參考訊息已翻譯，其他文字未翻譯。

- 從佇列管理程式傳回的資料不會轉換-例如，如果透過 REST API 執行 MQSC 指令，則佇列管理程式的回應會採用佇列管理程式的語言環境。
- REST API 產生的 (Swagger) 文件 (透過 `apiDiscovery` 特性公開) 採用英文。

支援哪些語言?

除了英文之外， REST API 錯誤及參考訊息也會翻譯成下列語言。

簡體中文
以語言標籤 `zh_CN` 表示

繁體中文
以語言標籤 `zh_TW` 表示

捷克文
以語言標籤 `cs` 表示

法文
以語言標籤 `fr` 表示

匈牙利文
以語言標籤 `hu` 表示

義大利文
以語言標籤 `it` 表示

日文
以語言標籤 `ja` 表示

韓文
以語言標籤 `ko` 表示

波蘭文
以語言標籤 `pl` 表示

(巴西) 葡萄牙文
以語言標籤 `pt_BR` 表示

西班牙文
以語言標籤 `es` 表示

範例

在範例中， Web 伺服器具有英文預設語言環境。

指定單一受支援語言
在要求標頭中， `ACCEPT-LANGUAGE` 設為 `fr`。 此設定指定法文是可翻譯文字的偏好語言。

在回應標頭中， `CONTENT-LANGUAGE` 設為 `fr`。 此設定指出回應中的錯誤及參考訊息是法文。

指定語言清單
在要求標頭中， `ACCEPT-LANGUAGE` 設為 `am, fr`。 此設定指定 Amharic 和法文是可翻譯文字的可接受語言，且 Amharic 是可翻譯文字的偏好語言。

在回應標頭中， `CONTENT-LANGUAGE` 設為 `fr`。 此設定指出回應中的錯誤及參考訊息是法文，因為 REST API 不支援 Amharic。

指定單一不受支援的語言
在要求標頭中， `ACCEPT-LANGUAGE` 設為 `am`。 此設定指定 Amharic 是可翻譯文字的偏好語言。

在回應標頭中， CONTENT-LANGUAGE 設為 en。 此設定指出回應中的錯誤及參考訊息是英文，因為 REST API 不支援 Amharic。

REST API 版本

REST API 版本號碼構成 REST 要求之基本 URL 的一部分。 例如，`https://localhost:9443/ibmmq/rest/v2/admin/installation`。 版本號碼是用來隔離用戶端與未來版本中可能引進之 REST API 的變更。

V 9.2.0 IBM MQ 9.2.0 引進 REST API 第 2 版。 此版本增加適用於 administrative REST API、messaging REST API 及 MFT REST API。 此版本增加會變更用於 REST API 的資源 URL。 第 2 版資源 URL 的 URL 字首是下列 URL:

```
https://host:port/ibmmq/rest/v2/
```

REST API 引進的部分變更可能會變更現有的 REST API 功能，因此使用 REST API 的用戶端可能需要更新。 為了防止這類變更強制更新用戶端， REST API 版本號碼會增加，且現有的功能會穩定在先前的數目。 可能會變更現有功能的新功能會以新版本號碼新增至 REST API 。 因此，用戶端可以在未更新的情況下繼續使用舊版 REST API 。

可能導致需要用戶端更新的 REST API 變更包括下列變更:

- 移除 JSON 中傳送至 REST API 或從中傳回之現有屬性的支援。
- 移除 URL、HTTP 動詞或標頭。 例如，如果重新命名 URL 或標頭，或使用不同的動詞。
- 將新的必要 JSON 屬性新增至傳送至現有 URL 的資料。
- 將新的必要 HTTP 標頭新增至傳送至現有 URL 的資料。
- 將新的必要查詢參數新增至現有 URL。

將這種類型的變更引進 Long Term Support (LTS) 版次中已存在的 REST API 函數時，這些第一項變更會增加 REST API 的版本號碼。 在 Continuous Delivery (CD) 版本內所做的任何後續變更，若需要變更使用 REST API 的用戶端，請使用新的版本號碼。

此版本號碼在後續的 CD 版本中保持不變，直到下一個 LTS 版本為止。 因此，在 LTS 版次之間，版本號碼最多會增加一次。

當版本號碼增加時，現有的 REST API 功能會穩定在舊版本號碼。 亦即，在 LTS 版本中提供的現有 REST API 功能仍在舊版本號碼中可用，但不會對該版本進行進一步變更。 任何新增至 REST API 的新功能都會新增至新的 REST API 版本。 不過，在版本增加之前的 CD 版本中，對 REST API 所做的任何新增都不保證包含在舊版 REST API 中。

現有用戶端可以繼續使用舊版本號碼的 REST API ，而不需要任何變更。 較舊版本的 REST API 可能已淘汰，且最終會移除。

部分變更不需要變更使用 REST API 的用戶端。 這些變更不會導致版本號碼增加。 因此，當引進這些類型的變更時，請確定使用 REST API 的任何用戶端都不需要更新。 REST API 的這些變更可能包括下列變更:

- 將新的 JSON 屬性新增至從 REST API 傳回的現有資料。
- 新增 URL。
- 將新的 HTTP 動詞新增至現有 URL。
- 將新的狀態碼新增至現有 URL。
- 將新的選用 JSON 屬性新增至傳送至現有 URL 的資料。
- 在現有 URL 上新增查詢參數。
- 將新標頭新增至傳送至現有 URL 的資料。
- 從 REST API 傳回新標頭。

對新 Continuous Delivery REST API 函數的變更

對於在 CD 版次中新增的 REST API 功能，對此新功能所做的任何變更，若可能需要變更 REST API 用戶端，則不會增加版本號碼。 也就是說，新功能可以在下一個 LTS 版次之前變更，而不增加版本號碼。 當 LTS 版本包含此功能時，任何可能需要變更 REST API 用戶端的後續變更都會增加版本號碼。

範例

1. 在 LTS 版次 X， REST API 是第 1 版。
2. 在 CD release X.0.1 中，新增了對新 URL 的支援。 這項變更不需要變更使用 REST API 的用戶端。 因此， REST API 仍為第 1 版。
3. 在 CD X.0.2 中，新增了對新 URL 的支援。 這項變更不需要變更使用 REST API 的用戶端。 因此， REST API 仍為第 1 版。
4. 在 LTS Y 版中， REST API 是第 1 版。
5. 在 CD release Y.0.1 中，會重新命名現有的 URL。 這項變更可能需要變更使用 REST API 的用戶端。 因此， REST API 的新版本會建立為第 2 版。 已重新命名的 URL 包括在 REST API 第 2 版中，以及所有現有的函數。 任何新增至 REST API 的新功能都會新增至第 2 版。 第 1 版在 LTS Y 版中保持穩定。
6. 在 CD release Y.0.2 中，會重新命名另一個現有的 URL。 由於 CD 版次 Y 中已增加版本，因此 REST API 會保留為第 2 版。 第 1 版在 LTS Y 版中保持穩定。
7. 在 LTS 發行版 Z， REST API 仍為第 2 版。 第 1 版在 LTS Y 版中保持穩定。

使用 Web 主控台進行管理

您可以使用 Web 主控台來執行基本管理作業。

V 9.2.0 從 IBM MQ 9.2.0 中，可以使用新的 Web 主控台 (稱為 New Web Console)，請參閱 第 82 頁的『「新建 Web 主控台」快速導覽』。

如果您想要的話，可以繼續在 Multiplatforms 上使用 Dashboard Web Console，請參閱 第 100 頁的『在主控台類型之間切換』。

註: 當您使用 Web 主控台時，請勿在任何佇列管理程式上停用指令伺服器。 如果佇列管理程式已停用指令伺服器，則 Web 主控台會變成無回應，且處理指令會長時間延遲。 對已停用指令伺服器的佇列管理程式所發出的任何指令都會逾時。

相關工作

V 9.2.0 追蹤新的 Web 主控台

追蹤儀表板 Web 主控台

V 9.2.0 Web 主控台入門

快速開始使用 Web 主控台。

開始之前

若要完成此作業，您必須是具有某些特權的使用者，如此才能使用 **dspmqweb** 指令:

- z/OS 在 z/OS 上，您必須有權執行 **dspmqweb** 指令，並且具有 `mqwebuser.xml` 檔的寫入權。
- Multi 在所有其他作業系統上，必須是 特許使用者。

IBM i 在 IBM i 上，指令應該在 QSHELL 中執行。

程序

1. 如果尚未配置 mqweb 伺服器以供 Web 主控台使用，請配置 mqweb 伺服器。
   如需使用基本登錄為 mqweb 伺服器建立基本配置的相關資訊，請參閱 mqweb 伺服器的基本配置。

2. z/OS
   在 z/OS 上，設定 WLP_USER_DIR 環境變數，以便您可以使用 **dspmqweb** 指令。 透過輸入下列指令，將變數設定為指向您的 mqweb 伺服器配置：

   ```
   export WLP_USER_DIR=WLP_user_directory
   ```

   ，其中 *WLP_user_directory* 是傳遞至 crtmqweb 的目錄名稱。 例如：

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

   如需相關資訊，請參閱建立 mqweb 伺服器。
3. 輸入下列指令來判斷 Web 主控台的 URI：

   ```
   dspmqweb status
   ```

   指令會產生類似下列的輸出：

   ```
   MQWB1124I: Server 'mqweb' is running.
      URLS:
      https://localhost:9443/ibmmq/rest/v1/
      https://localhost:9443/ibmmq/console/
   ```

   Web 主控台的 URI 以字尾 console/結尾。
4. 在瀏覽器中從 第 81 頁的『3』 輸入 URL，以連接至 Web 主控台。

   瀏覽器可能會產生安全異常狀況，因為 mqweb 伺服器提供的預設憑證不是授信憑證。 選擇繼續進行 Web 主控台。
5. 登入 Web 主控台。 使用使用者名稱 mqadmin 及密碼 mqadmin。

下一步

依預設， Web 主控台會使用記號型鑑別來鑑別使用者。 您也可以使用用戶端憑證鑑別。 如需相關資訊，請參閱 搭配使用用戶端憑證鑑別與 REST API 和 Web 主控台。

z/OS z/OS 的限制

使用 IBM MQ Console 來管理 z/OS 上的佇列管理程式時，下列限制適用。

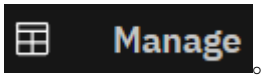
- 無法建立、刪除、啟動或停止 z/OS 上的佇列管理程式。
- z/OS 上的通道起始程式無法啟動或停止，且不會顯示通道起始程式狀態。
- 無法顯示或管理接聽器。
- 啟動、連線測試、解析及重設通道指令只能使用 CHLDISP (DEFAULT) 發出。
- 只能使用 QSGDISP (QMGR) 來建立新物件。
- 無法顯示或管理以 QSGDISP (GROUP) 定義的物件。
- 無法管理佇列管理程式安全。
- 無法監視系統資源使用情形。

相關概念

第 80 頁的『使用 Web 主控台進行管理』
您可以使用 Web 主控台來執行基本管理作業。

相關工作

第 85 頁的『使用本端佇列管理程式』

您可以從「管理」視圖的最上層建立、配置及控制本端佇列管理程式 

。

Multi IBM MQ for Multiplatforms 的限制

使用 IBM MQ Console 來管理 IBM MQ for Multiplatforms 上的佇列管理程式時，下列限制適用。

- 您無法使用 IBM MQ Console 來使用 AMQP 通道。
- 您無法使用 IBM MQ Console 來使用 MQTT 通道。

相關概念

第 80 頁的『使用 Web 主控台進行管理』
您可以使用 Web 主控台來執行基本管理作業。

相關工作

第 85 頁的『使用本端佇列管理程式』

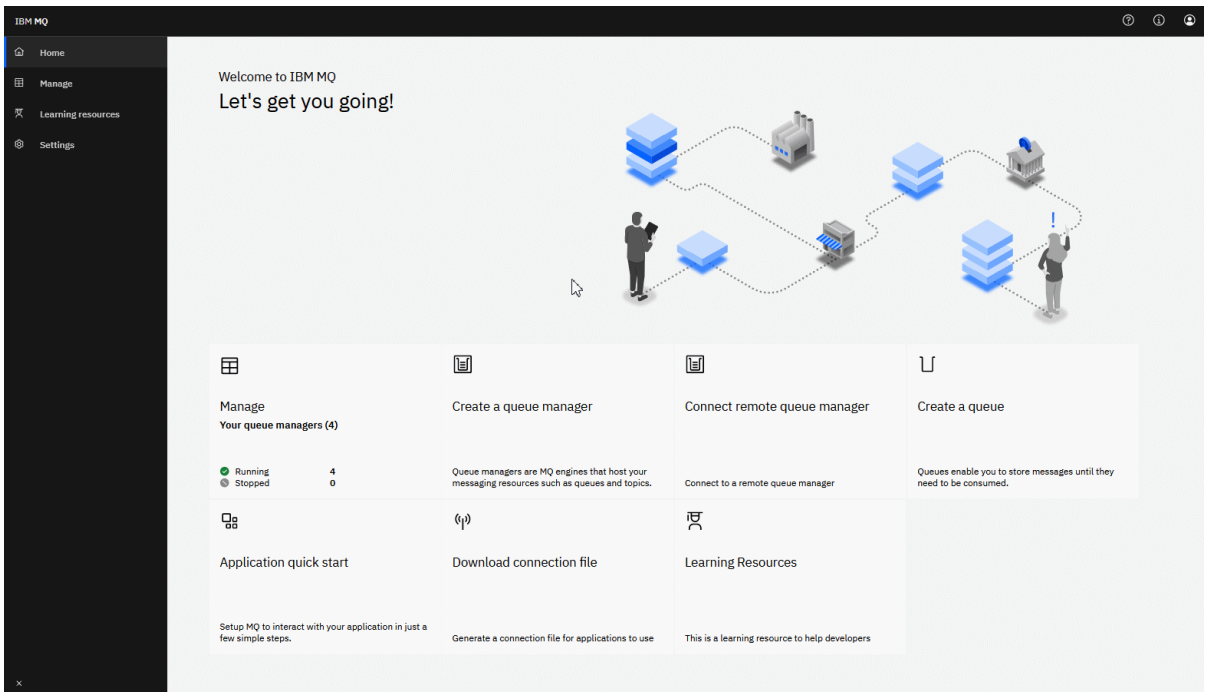
您可以從「管理」視圖的最上層建立、配置及控制本端佇列管理程式 Manage。

V 9.2.0 「新建 Web 主控台」快速導覽

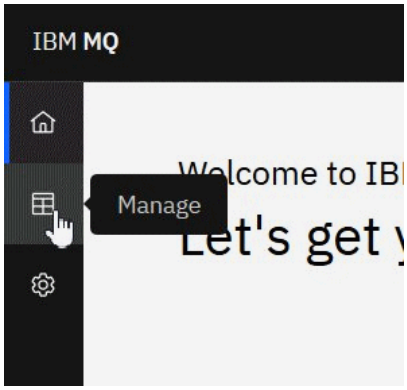
快速導覽可讓您開始使用 New Web Console。 有關其用法的更詳細主題如下。

New Web Console 是偏好的 Web 使用者介面，但如果您想要繼續使用現有主控台 (Dashboard Web Console) ，則可以切回該主控台，請參閱 第 100 頁的『在主控台類型之間切換』。

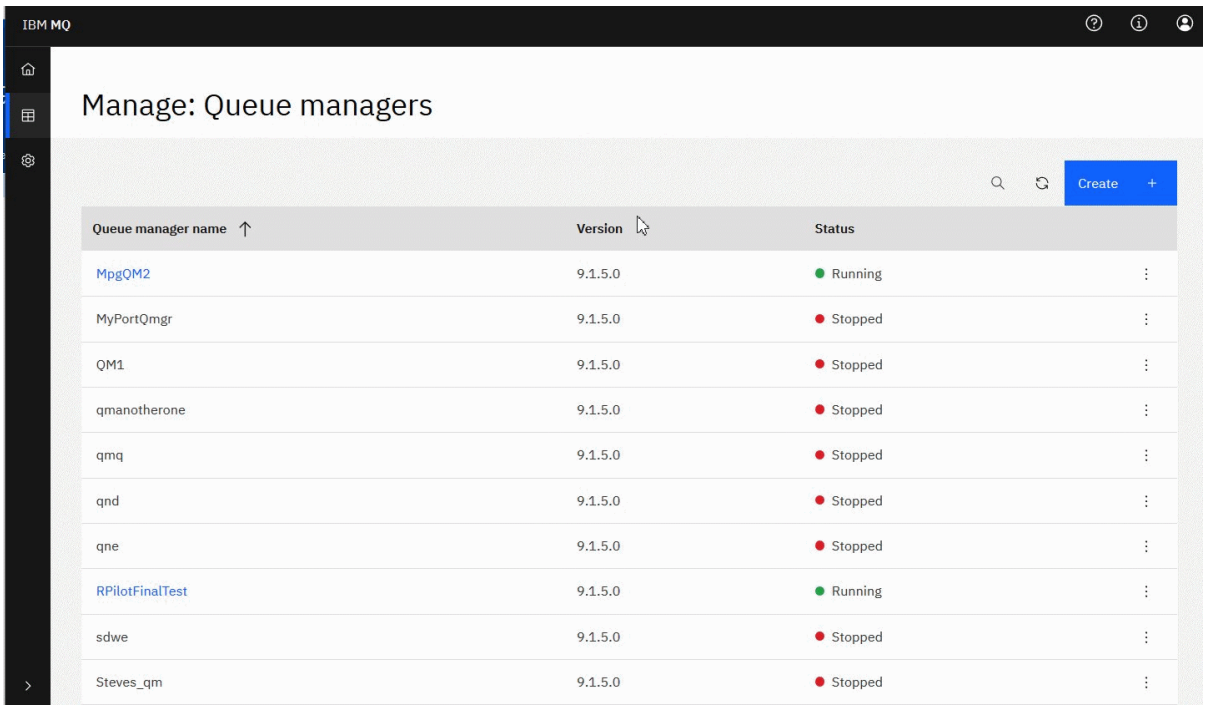
當您第一次登入 New Web Console 時，會將您帶至登陸頁面。 從這裡，您可以選擇管理現有佇列管理程式、建立佇列管理程式或佇列、導覽至部分教育主題，或在 IBM Documentation 中開啟 IBM MQ 產品資訊。 您也可以啟動應用程式快速入門，這會引導您完成快速且輕鬆地設定新佇列管理程式或現有佇列管理程式與應用程式之間傳訊的程序。



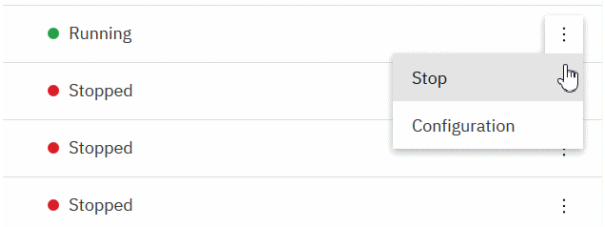
或者，您只要按一下管理圖示，即可立即開始管理 IBM MQ 物件。



管理視圖最初會顯示佇列管理程式及其現行狀態。 您也可以建立新的佇列管理程式，並連接遠端佇列管理程式。

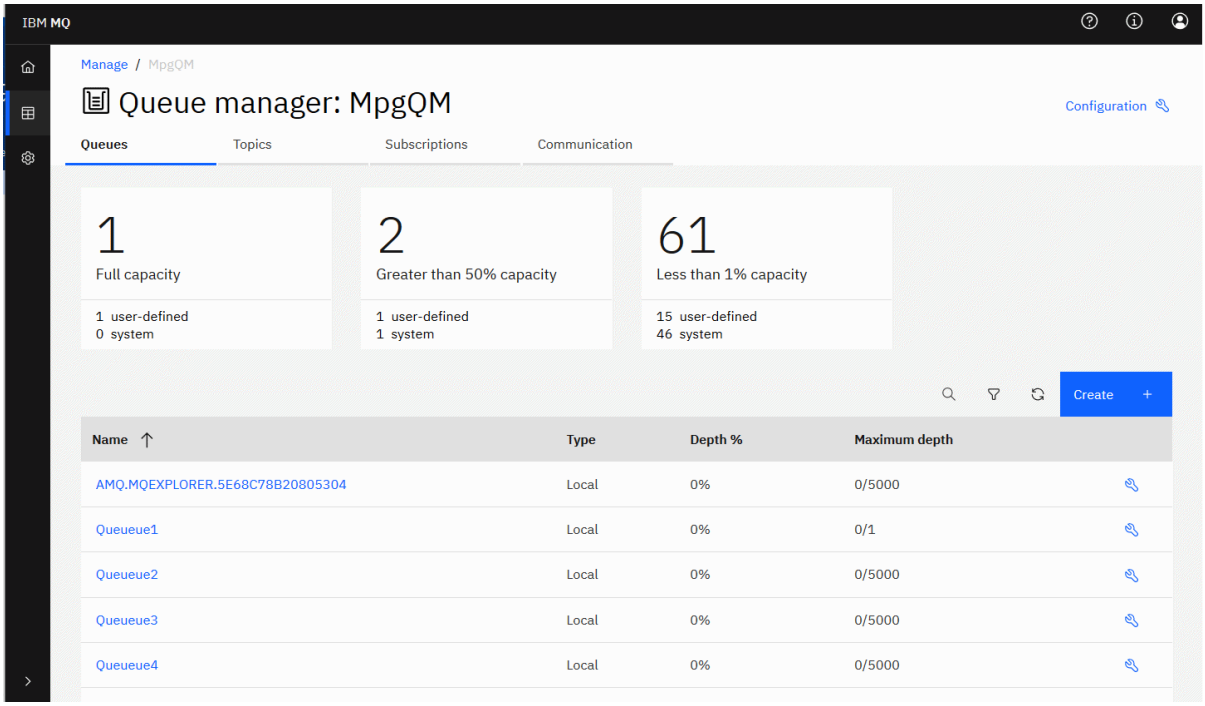


每一個佇列管理程式都有一個功能表，可讓您停止或配置執行中的佇列管理程式，或啟動或刪除已停止的佇列管理程式。



佇列管理程式的「權限記錄」、「鑑別資訊」物件及「通道鑑別」記錄位於佇列管理程式「**配置**」頁面的**安全**標籤上，您可以在其中建立及新增記錄。

按一下執行中佇列管理程式的名稱，以開啟其儀表板。



從佇列管理程式儀表板中，您可以完成下列動作:

在 **佇列** 標籤上:

- 建立新的佇列
- 配置現有佇列
- 按一下佇列名稱以檢視現有訊息並建立新訊息

在 **主題** 標籤上:

- 建立新的主題
- 配置現有主題
- 按一下主題名稱以檢視相符的訂閱

在 **訂閱** 標籤上:

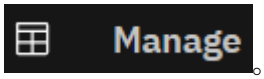
- 建立新的受管理或未受管理訂閱
- 配置現有訂閱

在 **通訊** 標籤上:

- listeners:
  - 啟動、停止及配置接聽器
  - 建立新的接聽器
- 佇列管理程式通道:
  - 啟動、停止、連線測試及配置通道
  - 建立新頻道
  - 重設通道 (從 **進階** 功能表項目)
  - 解決通道上不確定的訊息 (從 **進階** 功能表項目)

- 應用程式通道:
  - 啟動、停止、連線測試及配置通道
  - 建立新頻道
  - 重設通道 (從 **進階** 功能表項目)
  - 解決通道上不確定的訊息 (從 **進階** 功能表項目)

使用本端佇列管理程式

您可以從「管理」視圖的最上層建立、配置及控制本端佇列管理程式 

。

關於這項作業

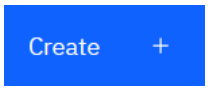
Multi 「管理」視圖會列出新增至的本端佇列管理程式 執行 IBM MQ Console 的 IBM MQ 安裝。 與相同系統上 IBM MQ 的不同安裝相關聯的佇列管理程式不會列出。

z/OS 在 z/OS 上， 「管理」視圖會列出與 IBM MQ Console 版本相同的佇列管理程式，以及在執行 MQ Console 的系統上定義的佇列管理程式。 不會列出與 MQ Console 不同版本的佇列管理程式。

您可以從清單中選取要使用的個別佇列管理程式。

註: IBM MQ Console 不支援抄寫的資料佇列管理程式 (RDQM)。

程序

- 如果要建立新的本端佇列管理程式，請執行下列動作:
  a) 按一下佇列管理程式清單視圖中的「建立」按鈕 

。
  b) 輸入新佇列管理程式的名稱。 名稱最多可以包含 48 個字元。 有效字元為字母和數字，以及 "."、"/"、 "_"和 "%" 字元。
  c) 選擇性的: 輸入可供佇列管理程式接聽的 TCP/IP 埠。 埠號不可超過 65535。
  d) 按一下**建立**。 即會建立並啟動新的佇列管理程式。
- 如果要啟動本端佇列管理程式，請執行下列動作:
  a) 在清單中尋找您要啟動的佇列管理程式。
  b) 從功能表 中選取 **開始** 。
- 如果要停止本端佇列管理程式，請執行下列動作:
  a) 從本端佇列管理程式小組件的清單中選取您要停止的佇列管理程式。
  b) 從功能表 中選取 **停止** 。
- 如果要刪除本端佇列管理程式，請執行下列動作:
  a) 如果佇列管理程式正在執行中，請停止它。
  b) 從功能表 中選取 **配置** ，然後選取 **刪除佇列管理程式**。
  c) 在確認視窗中輸入佇列管理程式的名稱，以確認您要刪除該佇列管理程式。 即會刪除佇列管理程式及所有相關聯的物件。
- 如果要檢視及編輯本端佇列管理程式的內容，請執行下列動作:
  a) 請確定佇列管理程式正在執行中，並在佇列管理程式清單中找到它。

b) 從功能表 ⋮ 中選取 **配置** 。

c) 確定已選取 **內容** 標籤。 檢視內容並視需要進行編輯。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。 如需內容的相關資訊，您可以在 佇列管理程式內容中檢視內容資訊。

- 如果要使用本端佇列管理程式的安全設定，請執行下列動作:

a) 請確定佇列管理程式正在執行中，並在佇列管理程式清單中選取它。

b) 從功能表 ⋮ 中選取 **配置** 。

c) 確定已選取 **安全** 標籤。

d) 您可以使用鑑別物件、授權記錄或通道鑑別物件。 如需相關資訊，請造訪下列主題:

- 第 86 頁的『使用鑑別資訊物件』
- 第 87 頁的『使用佇列管理程式權限記錄』
- 第 88 頁的『使用通道鑑別記錄』

V 9.2.0 使用鑑別資訊物件

您可以使用主控台來新增及刪除佇列管理程式上的鑑別資訊物件。 您也可以檢視及設定內容，以及管理物件的權限記錄。

關於這項作業

鑑別資訊視圖會列出特定佇列管理程式的鑑別資訊。 您可以從清單中選取要使用的個別鑑別資訊。

佇列管理程式鑑別資訊是 IBM MQ 支援傳輸層安全 (TLS) 的一部分。 這些物件包含在 LDAP 伺服器上使用 OCSP 或「憑證撤銷清冊 (CRL)」執行憑證撤銷檢查所需的定義，以及啟用使用者 ID 和密碼檢查所需的定義。

程序

- 如果要檢視佇列管理程式的鑑別資訊，請執行下列動作:

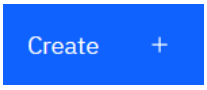
a) 請確定佇列管理程式正在執行中，並在佇列管理程式清單中選取它。

b) 從功能表 ⋮ 中選取 **配置** 。

c) 確定已選取 **安全** 標籤。

d) 從導覽畫面中選取 **鑑別資訊** 。

- 如果要新增鑑別資訊物件，請執行下列動作:

a) 按一下鑑別資訊清單視圖中的「建立」按鈕 

。

b) 指定鑑別資訊物件的名稱。 有效字元為字母和數字，以及 "."、 "/"、 "_"和 "%" 字元。

c) 指定鑑別資訊物件的類型。

d) 指定適用於物件類型的其他資訊:

- 對於 **CRL LDAP**，請指定 **LDAP 伺服器名稱**。 此名稱是執行 LDAP 伺服器之主機的主機名稱、IPv4 帶點十進位位址或 IPv6 十六進位表示法 (含選用埠號)。 您可以選擇性地為存取 LDAP 伺服器的使用者指定使用者名稱和密碼。
- 對於 **OCSP**，請指定 **OCSP 回應端 URL**。 此 URL 是用來檢查憑證撤銷的回應者 URL。 此值必須是包含 OCSP 回應端主機名稱及埠號的 HTTP URL。 如果 OCSP 回應端使用埠 80 (這是 HTTP 的預設值)，則可以省略埠號。 HTTP URL 定義在 RFC 1738 中。
- 對於 **IDPW OS**，雖然您可以選擇性地指定此鑑別類型的進一步選項，但沒有其他需求。

– 對於 **IDPW LDAP**，請指定 **LDAP 伺服器名稱** 及 **簡短使用者** 名稱。 LDAP 伺服器名稱是執行 LDAP 伺服器之主機的主機名稱、 IPv4 帶點十進位位址或 IPv6 十六進位表示法 (含選用埠號)。 簡短使用者名稱是 LDAP 使用者記錄中的欄位，用來作為連線的簡短名稱。 您可以選擇性地指定此鑑別類型的進一步選項。

e) 按一下**新增** 。

- 如果要刪除鑑別資訊物件，請執行下列動作:

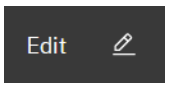
  a) 針對您要從清單中刪除的鑑別資訊物件，選取「扳手」圖示 。

  b) 在物件內容視圖中，按一下 **刪除鑑別資訊物件**。

  c) 按一下 **刪除**，確認您要刪除鑑別資訊物件。 已刪除物件。

- 如果要檢視及編輯鑑別資訊物件的內容，請執行下列動作:

  a) 針對您要從清單檢視的鑑別資訊物件，選取「扳手」圖示 。

  b) 若要編輯顯示的內容，請按一下「編輯」按鈕 

  c) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。

  d) 按一下**儲存**，以儲存變更。

- 如果要檢視及編輯鑑別資訊物件的權限記錄，請執行下列動作:

  a) 從清單中選取您要檢視其權限記錄之鑑別資訊物件的「扳手」圖示 。

  b) 選取 **安全** 標籤。

  c) 若要編輯或刪除現有權限記錄，請從功能表 中選取 **編輯** 或 **刪除** 。

  d) 若要新增權限記錄，請按一下 **新增** 按鈕

，提供新權限記錄的詳細資料，然後按一下 **建立**。

使用佇列管理程式權限記錄

您可以透過指定使用者或群組的權限記錄，來控制使用者和群組對佇列管理程式的存取權。

關於這項作業

您可以使用權限記錄，細部調整傳訊使用者或傳訊使用者群組對特定佇列管理程式的存取權。 權限記錄有兩種類型: 權限記錄控制一般權限，以及建立權限記錄控制哪些使用者和群組可以為佇列管理程式建立物件。

程序

- 如果要檢視佇列管理程式的權限記錄，請執行下列動作:

  a) 請確定佇列管理程式正在執行中，並在佇列管理程式清單中選取它。

  b) 從功能表 中選取 **配置** 。

  c) 確定已選取 **安全** 標籤。

  d) 從導覽畫面中選取 **權限記錄** 。 視圖會在兩個窗格中顯示權限記錄，可讓您使用一般權限記錄及建立權限記錄。

- 如果要新增一般權限記錄，請執行下列動作:

  a) 按一下權限記錄清單視圖中的「新增」按鈕

。

b) 選擇您要新增使用者或群組的權限記錄。
c) 指定您要新增其權限記錄的使用者或群組名稱 (權限記錄以此作為其名稱)
d) 選取您要授與的權限 (如需權限的相關資訊，請參閱 --)
e) 按一下**建立**。

- 如果要新增建立權限記錄，請執行下列動作:
  a) 按一下「建立權限記錄清單」視圖中的「新增」按鈕

。
  b) 選擇您要新增使用者或群組的權限記錄。
  c) 指定您要新增其權限記錄的使用者或群組名稱 (權限記錄以此作為其名稱)
  d) 選取您要授與建立權限的物件類型。
  e) 按一下**建立**。
- 如果要刪除權限記錄，請執行下列動作:
  a) 開啟您要刪除之權限記錄的功能表 ⋮ ，然後選取 **刪除**。
  b) 按一下 **刪除**，確認您要刪除鑑別資訊物件。 已刪除物件。
- 若要檢視及編輯權限記錄的內容，請執行下列動作:
  a) 開啟您要刪除之權限記錄的功能表 ⋮ ，然後選取 **編輯**。
  b) 視需要變更設定，然後按一下 **儲存** 以儲存變更。

V 9.2.0 使用通道鑑別記錄

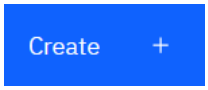
您可以使用 IBM MQ Console 來新增及刪除佇列管理程式上的通道鑑別記錄。 您也可以檢視及設定通道鑑別記錄的內容。

關於這項作業

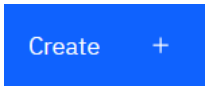
若要對授與在通道層次連接系統的存取權執行更精確的控制，您可以使用通道鑑別記錄。

若要施行安全，您可以使用封鎖通道鑑別記錄來封鎖對通道的存取。 您也可以使用位址對映通道鑑別記錄，以容許存取指定的使用者。 若要進一步瞭解通道鑑別記錄，請參閱 通道鑑別記錄。

程序

- 如果要檢視佇列管理程式的通道鑑別資訊，請執行下列動作:
  a) 請確定佇列管理程式正在執行中，並在佇列管理程式清單中選取它。
  b) 從功能表 ⋮ 中選取 **配置** 。
  c) 確定已選取 **安全** 標籤。
  d) 從導覽畫面中選取 **通道鑑別** 。
- 如果要新增通道鑑別記錄，請執行下列動作:
  a) 按一下通道鑑別資訊清單視圖中的「建立」按鈕 

。
  b) 選擇您要使用的規則類型。 選取一個 **容許**、 **封鎖**或 **警告**。
  c) 選擇您要配置通道鑑別規則的身分類型。 根據您選取的規則類型，可以使用不同的身分類型。
  d) 提供您正在指定之身分的必要資訊。 依預設，會顯示建議內容下限，供您提供值。 您可以選取 **顯示所有可用的選項**來檢視所有可用的內容。

e) 按一下「建立」按鈕 

，以建立通道鑑別記錄。

如需通道鑑別記錄可用設定的相關資訊，請參閱 通道鑑別記錄 及 SET CHLAUTH

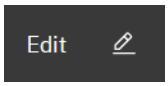
- 如果要刪除通道鑑別記錄，請執行下列動作:

  a) 按一下您要刪除之通道鑑別記錄旁的「扳手」圖示 。

  b) 在「編輯通道鑑別」視圖中，按一下 **刪除通道鑑別物件**。

  c) 按一下 **刪除**，確認您要刪除通道鑑別記錄。 通道鑑別記錄已刪除。
- 如果要檢視及編輯通道鑑別記錄的內容，請執行下列動作:

  a) 按一下您要編輯或檢視之通道鑑別記錄旁的「扳手」圖示 。 即會顯示內容。

  b) 按一下「編輯」按鈕 

  c) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。

  d) 按一下**儲存**，以儲存變更。

V 9.2.3 新增遠端佇列管理程式

您可以使用「 IBM MQ Console 」來管理在遠端系統上執行的佇列管理程式。

開始之前

您必須在遠端系統上準備佇列管理程式，以便可以從遠端管理它，請參閱 第 90 頁的『使用指令行將遠端佇列管理程式連接至 MQ Console』的步驟 第 90 頁的『1』 。

您也必須設定配置檔，以控制如何使用來自 IBM MQ Console 的遠端連線。 您可以使用 **setmqweb** 指令搭配 **remote** 參數來建立配置檔 (請參閱 配置遠端佇列管理程式連線行為 及 setmqweb)。 您無法直接編輯配置檔。

關於這項作業

您可以使用 JSON 格式的用戶端連線定義表 (CCDT) 來指定遠端連線詳細資料。 您可以使用文字編輯器來建立 JSON CCDT (請參閱 第 90 頁的『使用指令行將遠端佇列管理程式連接至 MQ Console』的步驟 第 91 頁的『2』)，也可以使用 IBM MQ Console 來建立 JSON CCDT。

或者，您可以在新增遠端佇列管理程式時直接指定連線詳細資料，以從 IBM MQ Console 建立 CCDT。

您也可以針對所有必要作業 (除了準備遠端佇列管理程式及建立 CCDT 之外) 使用指令行，將遠端佇列管理程式連接至 IBM MQ Console 。 請參閱 第 90 頁的『使用指令行將遠端佇列管理程式連接至 MQ Console』。

程序

- 如果要透過指定現有的 CCDT 來新增遠端佇列管理程式，請執行下列動作:

  a) 從「首頁」中，按一下 **連接遠端佇列管理程式**。

  b) 指定遠端佇列管理程式的名稱。

  c) 選擇性地指定佇列管理程式的唯一名稱。 如果您未指定唯一名稱，則實際名稱會與新增的字首 "remote-" 一起使用。

  d) 確定已選取 **使用 JSON CCDT 連接** 。

  e) 按一下 **瀏覽** ，然後選取包含您要使用之 JSON CCDT 的檔案。

  f) 按 **下一步** 以移至使用者頁面，並選擇性地指定使用者名稱及密碼，以連接至遠端佇列管理程式。 如果您未指定此資訊，則會從遠端連線配置檔取得鑑別資訊。

  g) 按 **下一步** 以移至「憑證」頁面。 如果 CCDT 指定 "transmissionSecurity" 資訊，則會使用此資訊。 您可以選擇性地貼上憑證 (作為 base64 編碼的公開金鑰) ，這會新增至廣域信任儲存庫。

憑證在新增至信任儲存庫之前，會暫時儲存在 *WLP_USER_DIR*/generated.crts/*uniqueName-qmgrName*.crt 中。 順利新增連線時，會從此位置刪除憑證。

h) 按 **下一步** 以檢視摘要頁面。 您可以使用 **上一步** 按鈕來重新造訪先前的頁面並進行更正。 如果您滿意此資訊，請按一下 **連接** 以連接遠端佇列管理程式。

- 如果要新增遠端佇列管理程式，並手動指定連線資訊，請執行下列動作:

a) 從「首頁」中，按一下 **連接遠端佇列管理程式**。

b) 指定遠端佇列管理程式的名稱。

c) 選擇性地指定佇列管理程式的唯一名稱。 如果您未指定唯一名稱，則實際名稱會與新增的字首 "remote-" 一起使用。

d) 選取 **手動登錄**。

e) 輸入連線將使用的用戶端連線通道名稱。

f) 指定遠端佇列管理程式執行所在的主機名稱。 如果偵測到遠端 MQ 安裝，則會顯示主機名稱，且您可以選取要連接的遠端佇列管理程式的主機。 在部分網路配置中，無法偵測遠端 MQ 實例。 在此情況下，請手動新增主機名稱及埠。

g) 按 **下一步** 以移至使用者頁面，並選擇性地指定使用者名稱及密碼，以連接至遠端佇列管理程式。 如果您未指定此資訊，則會從遠端連線配置檔取得鑑別資訊。

h) 按 **下一步** 以移至「憑證」頁面。 您可以從下拉清單中選取 SSL CipherSpec 。 您可以選擇性地貼上憑證 (作為 base64 編碼的公開金鑰) ，這會新增至廣域信任儲存庫。

憑證在新增至信任儲存庫之前，會暫時儲存在 *WLP_USER_DIR*/generated.crts/*uniqueName-qmgrName*.crt 中。 順利新增連線時，會從此位置刪除憑證。

i) 按 **下一步** 以檢視摘要頁面。 您可以使用 **上一步** 按鈕來重新造訪先前的頁面並進行更正。 如果您滿意此資訊，請按一下 **連接** 以連接遠端佇列管理程式。

您指定的連線資訊會寫入 Web 目錄中的 CCDT 檔案。 路徑為 *WLP_USER_DIR*/generated.ccdt/ccdt-*uniqueName*。

結果

遠端佇列管理程式會出現在「 IBM MQ Console」的遠端佇列管理程式清單中。 如果連線成功，您可以使用與使用本端佇列管理程式的物件相同的方式來管理遠端佇列管理程式的物件。

V 9.2.3 使用指令行將遠端佇列管理程式連接至 *MQ Console*

您可以在指令行上使用 **setmqweb remote** 指令，將遠端佇列管理程式連接至 MQ Console 。 遠端佇列管理程式可以是在與「 MQ Console」相同的系統上不同安裝中執行的佇列管理程式，也可以是在不同系統上執行的佇列管理程式。

開始之前

- 請確保 mqweb 伺服器已配置為容許遠端佇列管理程式連線至 MQ Console。 如需相關資訊，請參閱 配置遠端佇列管理程式連線行為。

程序

1. 配置本端佇列管理程式 QML 以接受遠端連線:

a) 建立伺服器連線通道，以容許從遠端管理佇列管理程式。

您可以使用 MQ Console 來建立伺服器連線通道，也可以在指令行上使用 **DEFINE CHANNEL** MQSC 指令。

例如，若要為遠端佇列管理程式 QM1 建立伺服器連線通道 QM1.SVRCONN ，請輸入下列指令:

```
runmqsc QM1
DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

如需 **DEFINE CHANNEL** 及可用選項的相關資訊，請參閱 DEFINE CHANNEL。

b) 請確定伺服器連線通道容許適當的使用者 ID 存取通道。 此使用者 ID 必須是在 MQ Console 建立遠端連線的系統上與 mqweb 伺服器一起啟動的使用者 ID。

您可以使用 MQ Console 來建立適當的權限記錄，也可以在指令行上使用 **SET CHLAUTH** MQSC 指令。

例如，若要授權使用者 `exampleUser` 存取遠端佇列管理程式 QM1 的 `QM1.SVRCONN`，請輸入下列指令:

```
SET CHLAUTH(QM1.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*') MCAUSER('exampleUser')
```

在此範例中，會設定 **address** 參數，讓 `exampleUser` 可以從任何 IP 位址進行連接。 您可以改為將 **address** 參數設為特定的 IP 位址。 例如，您可以限制只能存取 MQ Console 從遠端連接至佇列管理程式的 IP 位址。 如需此指令可用的選項相關資訊，請參閱 **SET CHLAUTH**。

c) ALW
建立接聽器以接受送入的網路連線:

您可以使用 MQ Console 來建立接聽器，也可以在指令行上使用 **DEFINE LISTENER** MQSC 指令。

例如，若要在埠 1414 上建立遠端佇列管理程式 QM1 的接聽器 `REMOTE.LISTENER`，請輸入下列指令:

```
DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
```

d) 確保接聽器在執行中:

您可以使用 MQ Console 來啟動接聽器，也可以在指令行上使用 **START LISTENER** MQSC 指令。

ALW 例如，在 AIX, Linux, and Windows 上，若要啟動佇列管理程式 QM1 的接聽器 `REMOTE.LISTENER`，請輸入下列指令:

```
START LISTENER(REMOTE.LISTENER)
```

z/OS 例如，在 z/OS 上，若要啟動接聽器，請輸入下列指令:

```
START LISTENER TRPTYPE(TCP) PORT(1414)
```

請注意，必須先啟動通道起始程式位址空間，然後才能在 z/OS 上啟動接聽器。

2. 建立包含遠端佇列管理程式連線資訊的 JSON CCDT 檔案:

- 使用與您要遠端連接之佇列管理程式相同的安裝相關聯的 MQ Console，從本端佇列管理程式定義產生 CCDT 檔案。

  從「**首頁**」畫面中，按一下 **下載連線檔案** 磚。
- 建立定義連線的 JSON 格式 CCDT 檔案。 如需建立 JSON 格式 CCDT 的相關資訊，請參閱 配置 JSON 格式 CCDT。

  CCDT 檔案必須包含 `name`、 `clientConnection` 及 `type` 資訊。 亦即， CCDT 檔案需要包含遠端佇列管理程式 (QM1) 的連線資料，且需要儲存在本端佇列管理程式 (QML) 的主機中，該主機具有 MQ Console。 總而言之，它是針對本端佇列管理程式 QML 連接至遠端佇列管理程式 QM1。

  您可以選擇性地包括其他資訊，例如 `transmissionSecurity` 資訊。 如需所有 CCDT 通道屬性定義的相關資訊，請參閱 CCDT 通道屬性定義的完整清單。

  下列範例顯示遠端佇列管理程式連線的基本 JSON CCDT 檔案。 它會將通道名稱設定為與步驟 第 90 頁的『1』中建立的範例伺服器連線通道相同的名稱，並將連線埠設定為與接聽器所使用埠相同的值。 連線主機會設為執行範例遠端佇列管理程式 QM1 之系統的主機名稱:

```
{
  "channel": [
    {
      "name": "QM1.SVRCONN",
      "clientConnection": {
        "connection": [
          {
```

```
            "host": "example.com",
            "port": 1414
          }
        ],
        "queueManager": "QM1"
      },
      "transmissionSecurity": {
        "cipherSpecification": "",
        "certificateLabel": "",
        "certificatePeerName": ""
      },
      "type": "clientConnection"
    }
  ]
}
```

3. 使用 **setmqweb remote** 指令，將遠端佇列管理程式資訊新增至 MQ Console 配置。 您必須使用與您想要從中檢視遠端佇列管理程式資訊之安裝相關聯的 **setmqweb** 指令。

   至少，若要將遠端佇列管理程式新增至「MQ Console」，您必須提供佇列管理程式名稱、佇列管理程式的唯一名稱 (以區分可能具有相同佇列管理程式名稱的其他遠端佇列管理程式)，以及佇列管理程式的 CCDT URL。 您可以指定數個其他選項，例如用於遠端佇列管理程式連線的使用者名稱及密碼，或信任儲存庫及金鑰儲存庫的詳細資料。 如需可使用 **setmqweb remote** 指令指定的完整參數清單，請參閱 setmqweb。

   例如，若要使用範例 CCDT 檔案新增範例遠端佇列管理程式 QM1，並指定用於連線的使用者名稱為 `exampleUser`，請輸入下列指令:

```
setmqweb remote add -uniqueName "MACHINEAQM1" -qmgrName "QM1" -ccdtURL
"c:\myccdts\ccdt.json" -username "exampleUser" -password "password"
```

結果

遠端佇列管理程式會出現在「IBM MQ Console」的遠端佇列管理程式清單中。 如果連線成功，您可以使用與使用本端佇列管理程式的物件相同的方式來管理遠端佇列管理程式的物件。

V 9.2.0 使用 IBM MQ 物件

每一個 IBM MQ 佇列管理程式都有數個不同類型的物件與其相關聯。

關於這項作業

您可以使用主控台來使用下列類型的 IBM MQ 物件:

- 佇列
- 主題
- 訂閱
- 通訊物件:
  - 接聽器
  - 佇列管理程式通道
  - 應用程式通道

程序

如果要使用 IBM MQ 物件，請執行下列動作:

1. 在佇列管理程式清單視圖中，按一下擁有您要使用之物件的佇列管理程式。
2. 按一下佇列、主題、訂閱或通訊標籤，以選取您要使用的物件類型。
3. 請參閱下列其中一個主題，以取得使用該類型物件的詳細指示。

V 9.2.0 使用佇列

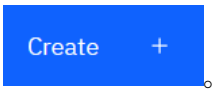
您可以在 **佇列** 標籤中檢視特定佇列管理程式的現有佇列。 您可以新增及刪除佇列、新增及清除佇列上的訊息、瀏覽訊息、檢視及設定佇列內容，以及管理佇列的權限記錄。

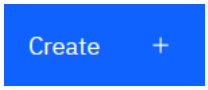
關於這項作業

佇列視圖會列出特定佇列管理程式的現有佇列。 您可以按一下佇列管理程式並選取 **佇列** 標籤，來存取佇列清單。 您可以從清單中選取要使用的個別佇列。

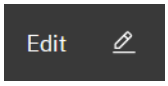
z/OS 您無法在 z/OS 上檢視或編輯佇列的權限記錄。

程序

- 如果要新增佇列，請執行下列動作:
  a) 在 **佇列** 標籤中，按一下「建立」按鈕 

。
  b) 選取您要建立的佇列類型:
     - 本端佇列-將訊息儲存在它所屬的佇列管理程式內。
     - 別名佇列-指向相同佇列管理程式上另一個佇列的指標。
     - 遠端佇列-指向不同佇列管理程式上另一個佇列的指標。
     - 模型佇列-建立動態佇列管理程式時所使用的佇列範本。
  c) 提供您正在建立之佇列類型的必要資訊。 依預設，會顯示建議內容下限，供您提供值。 您可以選取 **顯示所有可用的選項**來檢視所有可用的內容。
  d) 按一下**建立**。 即會建立新佇列。
- 如果要將訊息放到佇列中，請執行下列動作:
  a) 在佇列清單視圖的清單中，按一下您要在其中新增訊息的佇列。 您無法選取模型佇列。
  b) 按一下「建立」按鈕 

  c) 輸入您要放入佇列的訊息。
  d) 按一下**建立**。
- 如果要從佇列中清除訊息，請執行下列動作:
  a) 按一下您要從佇列清單中清除訊息的本端佇列。
  b) 按一下「動作」按鈕

，然後選取 **清除訊息**。
  c) 按一下 **清除訊息**，確認您要清除佇列。
- 若要瀏覽佇列上的訊息，請按一下佇列清單視圖中的佇列。 即會顯示該佇列上的訊息清單。
- 如果要刪除佇列，請執行下列動作:
  a) 按一下您要刪除之佇列旁的「扳手」圖示 。
  b) 在「編輯佇列」視圖中，按一下 **刪除佇列**。
  c) 按一下 **刪除**，確認您要刪除佇列。 已刪除佇列。
- 如果要檢視及編輯佇列的內容，請執行下列動作:
  a) 按一下您要編輯之佇列旁的「扳手」圖示 。
  b) 按一下「編輯」按鈕 

c) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。 如需內容的相關資訊，請參閱 MQ Explorer 文件中的 佇列內容

d) 按一下**儲存**，以儲存變更。

- 若要檢視及編輯佇列的權限記錄，請執行下列動作:

a) 按一下您要編輯其權限記錄之佇列旁的「扳手」圖示 。

b) 按一下**安全**標籤。

c) 依照佇列管理程式權限記錄的說明來使用權限記錄。 請參閱 第 87 頁的『使用佇列管理程式權限記錄』。

V 9.2.0 使用主題

您可以使用 IBM MQ Console 來新增和刪除主題，以及檢視和設定主題的內容。

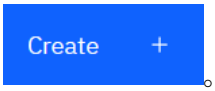
關於這項作業

主題視圖會列出特定佇列管理程式的現有主題。 您可以按一下佇列管理程式並選取 **主題** 標籤來存取主題清單。 您可以從清單中選取要使用的個別主題。

z/OS 您無法在 z/OS 上檢視或編輯主題的權限記錄。

程序

- 如果要新增主題，請執行下列動作:

a) 在 **主題** 標籤中，按一下「建立」按鈕 

。

b) 提供您要建立之主題的必要資訊。 依預設，會顯示建議內容下限，供您提供值。 您可以選取 **顯示所有可用的選項**來檢視所有可用的內容。

c) 按一下**建立**。 即會建立新主題。

- 如果要刪除主題，請執行下列動作:

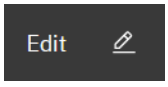
a) 按一下您要刪除之主題旁的「扳手」圖示 。

b) 在「編輯佇列」視圖中，按一下 **刪除主題**。

c) 按一下 **刪除**，確認您要刪除主題。 已刪除主題。

- 如果要檢視及編輯主題的內容，請執行下列動作:

a) 按一下您要編輯之主題旁的「扳手」圖示 。

b) 按一下「編輯」按鈕 

c) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。 如需內容的相關資訊，請參閱「 MQ 探險家」說明文件中的 主題內容 。

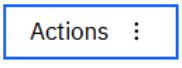
d) 按一下**儲存**，以儲存變更。

- 若要發佈主題的訊息，您必須至少有一個相符的訂閱。 必要的話，您可以建立測試訂閱:

a) 在主題清單中按一下您要發佈至其中的主題。

b) 您的下一個動作取決於這個主題是否有任何相符的訂閱:

如果沒有相符的訂閱:

a. 按一下「動作」按鈕 

，然後選取 **測試主題**。

b. 按一下「測試」按鈕 Test。 測試訊息會寫入測試訂閱。

如果主題有相符的訂閱:

a. 按一下訂閱名稱。

b. 按一下「建立」按鈕 Create +。

c. 輸入您要發佈的訊息。

d. 按一下「放置」按鈕 Put。 訊息會寫入所有符合的訂閱。

- 若要訂閱主題，請參閱 第 95 頁的『使用訂閱』:
- 如果要檢視及編輯主題的權限記錄，請執行下列動作:

a) 按一下您要編輯其權限記錄之主題旁邊的「扳手」圖示 。

b) 按一下**安全**標籤。

c) 如佇列管理程式權限記錄的說明，使用權限記錄，請參閱 第 87 頁的『使用佇列管理程式權限記錄』。

V 9.2.0 使用訂閱

您可以使用 IBM MQ Console 來新增和刪除訂閱，以及檢視和設定訂閱的內容。

關於這項作業

訂閱視圖會列出特定佇列管理程式的現有訂閱。 您可以按一下佇列管理程式並選取 **訂閱** 標籤來存取訂閱清單。 您可以從清單中選取要使用的個別訂閱。

如需訂閱的相關資訊，請參閱 訂閱者和訂閱 及 DEFINE SUB。

z/OS 您無法在 z/OS 上檢視或編輯訂閱的權限記錄。

程序

- 若要新增訂閱，請執行下列動作:

a) 在 **訂閱** 標籤中，按一下「建立」按鈕 Create +。

b) 選擇您要建立受管理還是未受管理的訂閱。

c) 提供您正在建立之訂閱的必要資訊。 依預設，會顯示建議內容下限，供您提供值。 您可以選取 **顯示所有可用的選項**來檢視所有可用的內容。

d) 按一下**建立**。 即會建立新的訂閱。

- 如果要刪除訂閱，請執行下列動作:

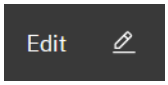
a) 按一下您要刪除之訂閱旁的「扳手」圖示 。

b) 在「編輯佇列」視圖中，按一下 **刪除訂閱**。

c) 按一下 **刪除**，以確認您要刪除訂閱。 已刪除訂閱。

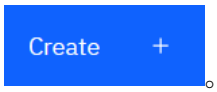
- 若要檢視及編輯訂閱的內容，請執行下列動作:

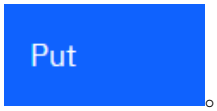
a) 按一下您要編輯之訂閱旁的「扳手」圖示 。

b) 按一下「編輯」按鈕 

c) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。

d) 按一下**儲存**，以儲存變更。

- 如果要針對訂閱所訂閱的主題發佈訊息，請執行下列動作:
  a) 在訂閱清單中按一下您要發佈至其主題的訂閱。
  b) 按一下「建立」按鈕 

。
  c) 輸入您要發佈的訊息。
  d) 按一下「放置」按鈕 

。 訊息會寫入符合您已發佈至的主題的所有訂閱。

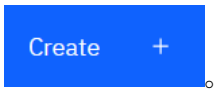
V 9.2.0 使用接聽器

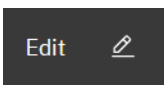
您可以使用 IBM MQ Console 來新增和刪除接聽器、啟動和停止接聽器、檢視和設定接聽器內容，以及管理接聽器的權限記錄。

關於這項作業

接聽器視圖會顯示特定佇列管理程式的現有接聽器。 您可以選取要使用的個別接聽器。

程序

- 如果要建立接聽器，請執行下列動作:
  a) 在 **通訊** 標籤中，確保顯示接聽器視圖，並按一下「建立」按鈕 

。
  b) 提供您正在建立之接聽器的必要資訊。 依預設，會顯示建議內容下限，供您提供值。 您可以選取 **顯示所有可用的選項**來檢視所有可用的內容。
  c) 按一下**建立**。 即會建立新的接聽器。
- 如果要啟動接聽器，請執行下列動作:
  a) 在清單中尋找您要啟動的接聽器。
  b) 從功能表 ⋮ 中選取 **開始** 。
- 如果要停止接聽器，請執行下列動作:
  a) 在清單中尋找您要啟動的接聽器。
  b) 從功能表 ⋮ 中選取 **停止** 。
- 如果要檢視及編輯接聽器的內容，請執行下列動作:
  a) 在清單中尋找接聽器。
  b) 從功能表 ⋮ 中選取 **配置** 。
  c) 確定已選取 **內容** 標籤。 若要編輯內容，請按一下「編輯」按鈕 

  d) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。 如需內容的相關資訊，請參閱「 MQ 探險家」說明文件中的 接聽器內容 。

e) 按一下**儲存**，以儲存變更。

- 若要檢視及編輯接聽器的權限記錄，請執行下列動作:
  a) 在清單中尋找接聽器。
  b) 從功能表 ⋮ 中選取 **配置** 。
  c) 按一下**安全**標籤。
  d) 如佇列管理程式權限記錄的說明，使用權限記錄，請參閱 第 87 頁的『使用佇列管理程式權限記錄』。
- 如果要刪除接聽器，請執行下列動作:
  a) 在清單中尋找接聽器。
  b) 從功能表 ⋮ 中選取 **配置** 。
  c) 按一下 **刪除接聽器**。

V 9.2.0 使用佇列管理程式通道

您可以使用 IBM MQ Console 搭配佇列管理程式通道: 您可以新增及刪除佇列管理程式通道、啟動及停止通道、重設及解析通道，以及連線測試通道。 您也可以檢視及設定佇列管理程式通道的內容，以及管理通道的權限記錄。

關於這項作業

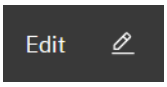
佇列管理程式通道是邏輯通訊鏈結，用於透過網路在佇列管理程式之間傳輸訊息。 佇列管理程式通道視圖包含一個畫面，顯示執行中通道數、重試次數及已停止通道數的快速視圖。

z/OS 您無法在 z/OS 上檢視或編輯通道的權限記錄。

程序

- 如果要新增佇列管理程式通道，請執行下列動作:
  a) 在 **通訊** 標籤中，確保顯示佇列管理程式通道視圖，並按一下「建立」按鈕 Create + 。
  b) 選取您要建立的佇列管理程式通道類型，然後按「下一步」按鈕 Next 。
  c) 提供您正在建立之通道的必要資訊。 依預設，會顯示建議內容下限，供您提供值。 您可以選取 **顯示所有可用的選項**來檢視所有可用的內容。
  d) 按一下**建立**。 即會以 **非作用中** 狀態建立新通道。
- 如果要啟動佇列管理程式通道，請執行下列動作:
  a) 在清單中尋找您要啟動的通道。
  b) 從功能表 ⋮ 中選取 **開始** 。
- 如果要停止佇列管理程式通道，請執行下列動作:
  a) 在清單中尋找您要停止的通道。
  b) 從功能表 ⋮ 中選取 **停止** 。
- 如果要檢視佇列管理程式通道的內容，請執行下列動作:
  a) 在清單中尋找通道。

b) 從功能表 ⋮ 中選取 **配置** 。

c) 確定已選取 **內容** 標籤。 若要編輯內容，請按一下「編輯」按鈕 

d) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。 如需內容的相關資訊，請參閱「 MQ 探險家」說明文件中的 通道內容 。

e) 按一下**儲存**，以儲存變更。

- 如果要重設佇列管理程式通道，請執行下列動作:
  a) 在清單中尋找通道。

  b) 從功能表 ⋮ 中選取 **進階** 。

  c) 在 **重設** 區段中，指定訊息序號。
  如果通道不會啟動，則您需要重設通道，因為兩端對於要傳送的下一個訊息序號不同意。 訊息序號指定該號碼。

  d) 按一下 **重設通道**。
- 如果要解析傳送端或伺服器通道，請執行下列動作:
  a) 在清單中尋找通道。

  b) 從功能表 ⋮ 中選取 **進階** 。

  c) 在 **解決** 區段中，按一下 **將訊息還原至傳輸佇列** 或 **捨棄訊息**，以選擇要確定還是取消現行訊息批次。
- 若要對佇列管理程式通道進行連線測試，請執行下列動作:
  a) 在清單中尋找通道。

  b) 從功能表 ⋮ 中選取 **連線測試** 。
- 如果要檢視及編輯佇列管理程式通道的權限記錄，請執行下列動作:
  a) 在清單中尋找通道。

  b) 從功能表 ⋮ 中選取 **配置** 。

  c) 按一下**安全**標籤。

  d) 如佇列管理程式權限記錄的說明，使用權限記錄，請參閱 第 87 頁的『使用佇列管理程式權限記錄』。
- 如果要刪除佇列管理程式通道，請執行下列動作:
  a) 在清單中尋找通道。

  b) 從功能表 ⋮ 中選取 **配置** 。

  c) 按一下 **刪除通道**。

V 9.2.0 使用應用程式通道

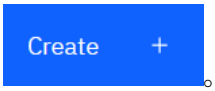
您可以使用 IBM MQ Console 來使用應用程式通道: 您可以新增及刪除通道、啟動及停止通道、重設及解析通道，以及連線測試通道。 您也可以檢視及設定應用程式通道的內容，以及管理通道的權限記錄。

關於這項作業

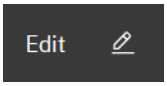
應用程式通道是邏輯通訊鏈結，應用程式使用該通訊鏈結，透過網路連接佇列管理程式。 應用程式通道視圖包括一個畫面，顯示執行中通道數、重試數及已停止通道數的快速視圖。

z/OS 您無法在 z/OS 上檢視或編輯通道的權限記錄。

程序

- 若要新增應用程式通道，請執行下列動作:
  a) 在 **通訊** 標籤中，確保顯示應用程式通道視圖，然後按一下「建立」按鈕 

。
  b) 按「下一個」按鈕

。
  c) 提供您正在建立之通道的必要資訊。 依預設，會顯示建議內容下限，供您提供值。 您可以選取 **顯示所有可用的選項**來檢視所有可用的內容。
  d) 按一下**建立**。 即會以 **非作用中** 狀態建立新通道。
- 如果要啟動應用程式通道，請執行下列動作:
  a) 在清單中尋找您要啟動的通道。
  b) 從功能表 ⋮ 中選取 **開始** 。
- 如果要停止應用程式通道，請執行下列動作:
  a) 在清單中尋找您要停止的通道。
  b) 從功能表 ⋮ 中選取 **停止** 。
- 如果要檢視應用程式通道的內容，請執行下列動作:
  a) 在清單中尋找通道。
  b) 從功能表 ⋮ 中選取 **配置** 。
  c) 確定已選取 **內容** 標籤。 若要編輯內容，請按一下「編輯」按鈕 

  d) 依需求編輯內容。 如果停用內容文字框，則內容是唯讀的，或只能從指令行編輯。 如需內容的相關資訊，請參閱「 MQ 探險家」說明文件中的 通道內容 。
  e) 按一下**儲存**，以儲存變更。
- 若要重設應用程式通道，請執行下列動作:
  a) 在清單中尋找通道。
  b) 從功能表 ⋮ 中選取 **進階** 。
  c) 在 **重設** 區段中，指定訊息序號。
  如果通道不會啟動，則您需要重設通道，因為兩端對於要傳送的下一個訊息序號不同意。 訊息序號指定該號碼。
  d) 按一下 **重設通道**。
- 如果要解析傳送端或伺服器通道，請執行下列動作:
  a) 在清單中尋找通道。
  b) 從功能表 ⋮ 中選取 **進階** 。
  c) 在 **解決** 區段中，按一下 **將訊息還原至傳輸佇列** 或 **捨棄訊息**，以選擇要確定還是取消現行訊息批次。
- 如果要對通道進行連線測試，請執行下列動作:
  a) 在清單中尋找通道。
  b) 從功能表 ⋮ 中選取 **連線測試** 。

- 若要檢視及編輯應用程式通道的權限記錄，請執行下列動作:
  a) 在清單中尋找通道。
  b) 從功能表 ⋮ 中選取 **配置** 。
  c) 按一下**安全**標籤。
  d) 如佇列管理程式權限記錄的說明，使用權限記錄，請參閱 第 87 頁的『使用佇列管理程式權限記錄』。
- 如果要刪除應用程式通道，請執行下列動作:
  a) 在清單中尋找通道。
  b) 從功能表 ⋮ 中選取 **配置** 。
  c) 按一下 **刪除通道**。

V 9.2.0 Web 主控台設定

您可以指定新 Web 主控台的一些一般設定。

按一下設定圖示 Settings ，以切換至 Web 主控台設定視圖。

使用設定來控制下列特性:

- 每十秒自動重新整理佇列管理程式一次。 此特性可以開啟或關閉。
- 是否顯示系統物件。 您可以針對所有物件類型指定此項目，或個別選取物件類型。
- 是否收集追蹤資訊。

z/OS Multi V 9.2.0 在主控台類型之間切換

您可以在 New Web Console (IBM MQ 9.2 的預設 Web 主控台) 與 Dashboard Web Console (舊版 IBM MQ 的 Web 主控台) 之間切換。

關於這項作業

您可以使用 **setmqweb** 指令在主控台之間切換。

如果您切換回使用 Dashboard Web Console，請參閱 IBM MQ 9.1 說明文件中的 儀表板 Web 主控台 。 以取得用法指示。

若要完成此作業，必須是具有特定專用權的使用者，讓您可以使用 **dspmqweb** 及 **setmqweb** 指令:

- z/OS 在 z/OS 上，您必須具有執行 **dspmqweb** 和 **setmqweb** 指令的權限，以及對 `mqwebuser.xml` 檔案的寫入權。
- Multi 在所有其他作業系統上，必須是 特許使用者。

小心: z/OS

在 z/OS 上發出 **setmqweb** 或 **dspmqweb** 指令之前，您必須先設定 WLP_USER_DIR 環境變數，讓變數指向您的 mqweb 伺服器配置。

若要這樣做，請發出下列指令:

```
export WLP_USER_DIR=WLP_user_directory
```

其中，*WLP_user_directory* 是傳遞至 crtmqweb 的目錄名稱。 例如:

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

如需相關資訊，請參閱建立 mqweb 伺服器。

程序

- 若要切換至 Dashboard Web Console，請使用下列指令:
  - 輸入下列指令，將 mqConsoleEnableUnsafe 行內內容設為 true:

    ```
    setmqweb properties -k mqConsoleEnableUnsafeInline -v true
    ```

    請注意，設定此內容會放寬 CSP 規則集以啟用不安全-行內，這可能會讓您的配置更不安全，但 Dashboard Web Console 需要此內容
  - 透過輸入下列指令，將 mqweb 伺服器切換為使用 Dashboard Web Console :

    ```
    setmqweb properties -k mqConsoleEarName -v com.ibm.mq.console
    ```
- 若要回復為 mq Web 伺服器 (包括 New Web Console) 的預設值，請使用下列指令:

  ```
  setmqweb properties -r
  ```
- 若要回復至 New Web Console，同時保留您對 mqweb 伺服器所做的任何其他自訂作業，請使用下列指令:
  - 輸入下列指令，將 mqConsoleEnableUnsafeInline 內容設為 false 以還原安全配置:

    ```
    setmqweb properties -k mqConsoleEnableUnsafeInline -v false
    ```
  - 透過輸入下列指令，將 mqweb 伺服器切換為使用 New Web Console :

    ```
    setmqweb properties -k mqConsoleEarName -v com.ibm.mq.webconsole
    ```

相關參考
setmqweb

Windows Linux 使用 IBM MQ Explorer 進行管理

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

IBM MQ for Windows 及 IBM MQ for Linux x86-64 提供稱為 IBM 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 MQ Explorer 執行的動作。

IBM MQ Explorer 可讓您透過將 IBM MQ Explorer 指向您感興趣的佇列管理程式及叢集，從執行 Windows 或 Linux x86-64 的電腦執行網路的本端或遠端管理。 它可以從遠端連接至在包括 z/OS 的任何受支援平台上執行的佇列管理程式，讓您可以從主控台檢視、探索及變更整個傳訊骨幹。

若要配置遠端 IBM MQ 佇列管理程式，以便 IBM MQ Explorer 可以管理它們，請參閱 第 103 頁的『IBM MQ Explorer 的必備軟體及定義』。

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

在 Linux 上，如果您有多個 Eclipse 安裝架構， IBM 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 MQ Explorer 。

在 Linux 上，若要順利啟動 IBM 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。

IBM MQ Explorer 可以在產品安裝過程中安裝 (請參閱 安裝及解除安裝 IBM MQ)，也可以從獨立式 IBM MQ Explorer 可從 Fix Central 下載 (請參閱 在 Linux 和 Windows 上作為獨立式應用程式安裝及解除安裝 IBM MQ Explorer)。 安裝。

Windows Linux 您可以使用 IBM MQ Explorer 執行的動作

您可以使用 IBM MQ Explorer，利用一系列「內容視圖」及「內容」對話框來執行管理作業。 您也可以撰寫一或多個 Eclipse 外掛程式來延伸 IBM MQ Explorer 。

IBM MQ Explorer 作業

使用 IBM MQ Explorer，您可以執行下列作業:

- 建立並 刪除 佇列管理程式 (僅在本端機器上)。
- 啟動和停止佇列管理程式 (僅在本端機器上)。
- 定義、顯示及變更 IBM MQ 物件 的定義，例如佇列及通道。
- 瀏覽佇列中的訊息。
- 啟動和停止通道。
- 檢視狀態資訊 ，關於通道、接聽器、佇列或服務物件。
- 檢視叢集中的佇列管理程式。
- 請檢查 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 建立新的佇列管理程式叢集 。
- 使用「新增佇列管理程式至叢集」精靈 將佇列管理程式新增至叢集 。
- 管理鑑別資訊物件，與「傳輸層安全 (TLS)」通道安全搭配使用。
- 建立及刪除通道起始程式、觸發監視器及接聽器。
- 啟動或停止 指令伺服器、 通道起始程式、 觸發監視器及 接聽器。
- 將特定服務設為 在佇列管理程式啟動時自動啟動。
- 修改佇列管理程式的內容。
- 變更本端預設佇列管理程式。
- 呼叫 **strmqikm** (ikeyman) GUI 以管理 TLS 憑證，建立憑證與佇列管理程式的關聯，以及配置和設定憑證儲存庫 (僅在本端機器上)。
- 從 IBM MQ 物件建立 JMS 物件及 JMS 物件中的 IBM MQ 物件。
- 針對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。

內容視圖和內容對話框

您可以使用一系列「內容視圖」及「內容」對話框來執行管理作業。

「內容」視圖
「內容視圖」是可以顯示下列內容的畫面:

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

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

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

延伸 IBM MQ Explorer

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

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

如需相關資訊，請參閱 延伸 MQ 探險家。

Windows Linux 決定是否使用 IBM MQ Explorer

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

您需要注意下列要點:

物件名稱
如果您使用 IBM MQ Explorer 對佇列管理程式及其他物件使用小寫名稱，當您使用 MQSC 指令來處理物件時，必須以單引號括住物件名稱，否則 IBM MQ 無法辨識它們。

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

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

設定 IBM MQ Explorer

本節概述設定 IBM MQ Explorer 所需的步驟。

- 第 103 頁的『IBM MQ Explorer 的必備軟體及定義』
- 第 103 頁的『的安全 IBM MQ Explorer』
- 第 106 頁的『在「IBM MQ Explorer」中顯示及隱藏佇列管理程式和叢集』
- 第 107 頁的『叢集成員資格和 IBM MQ Explorer』
- 第 108 頁的『IBM MQ Explorer 的資料轉換』

IBM MQ Explorer 的必備軟體及定義

在嘗試使用 IBM MQ Explorer 之前，請確定您滿足下列需求。

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

請檢查:

1. 指令伺服器正在每個遠端管理佇列管理程式上執行。
2. 必須在每個遠端佇列管理程式上執行適當的 TCP/IP 接聽器物件。 此物件可以是 IBM MQ 接聽器，也可以是在 AIX 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。

的安全 IBM MQ Explorer

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

使用 *IBM MQ Explorer* 的授權

任何使用者都可以使用 IBM MQ 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 IBM MQ Explorer 來執行本端管理作業，使用者必須具備執行管理作業的必要權限。 如果使用者是 mqm 群組的成員，則使用者有權執行所有本端管理作業。

若要使用「IBM MQ Explorer」連接至遠端佇列管理程式並執行遠端管理作業，執行「IBM MQ Explorer」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.MQEXPLORER.REPLY.MODEL
- 對佇列 SYSTEM.ADMIN.COMMAND.QUEUE
- 佇列 SYSTEM.ADMIN.COMMAND.QUEUE
- 執行所選動作的權限

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

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

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

此外，執行 IBM MQ Explorer 的使用者必須具有下列權限:

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

如需如何授與 IBM MQ 物件權限的相關資訊，請參閱 授與 AIX, Linux, and Windows 系統上 IBM MQ 物件的存取權。

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

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

從 *IBM MQ Explorer* 連接至遠端佇列管理程式的安全

您必須保護 IBM MQ Explorer 與每一個遠端佇列管理程式之間的通道安全。

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

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

搭配使用安全結束程式與 *IBM MQ Explorer*

您可以使用 IBM MQ Explorer 來指定預設安全結束程式及佇列管理程式特定的安全結束程式。

您可以定義預設安全結束程式，它可用於來自 IBM MQ Explorer 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 IBM MQ Explorer 來指定結束程式。 有關詳細資訊，請參閱 IBM MQ Explorer 說明。

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

「IBM 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 TLS 安全來保護 MQI 通道，則必須使用用戶端通道定義表來建立通道。

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

當您使用用戶端通道定義表建立通道時，可以使用「IBM MQ Explorer」，利用啟用 TLS 的 MQI 通道來連接遠端佇列管理程式，如 第 105 頁的『管理遠端佇列管理程式之系統上的作業』 及 第 105 頁的『管理 IBM 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及用戶端連線配對，並為兩個通道上伺服器連線的 *SSLCIPH* 屬性指定適當的值。 如需 *SSLCIPH* 屬性的相關資訊，請參閱 使用 TLS 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 MQ Explorer 的系統。
3. 在指定埠上啟動 TCP/IP 接聽器。
4. 將 CA 和個人 TLS 憑證都放在佇列管理程式的 SSL 目錄中:
   - AIX `/var/mqm/qmgrs/+QMNAME+/SSL` 適用於 AIX 或 Linux 系統。
   - Windows `C:\Program Files\IBM\MQ\qmgrs\+QMNAME+\SSL` 適用於 Windows 系統。

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

如需在 Windows 系統上使用 TLS 的詳細資訊，請參閱 在 AIX, Linux, and Windows 上使用 TLS。

管理 IBM MQ Explorer 之系統上的作業

在管理 IBM 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 MQ Explorer 使用 Java 金鑰儲存庫檔 (JKS) 來確保 TLS 安全，因此建立用於為 IBM MQ Explorer 配置 TLS 的金鑰儲存庫檔必須符合此要求。
2. 將 CA 憑證新增至前一個步驟中建立的金鑰資料庫。
3. 將佇列管理程式的個人憑證匯入金鑰資料庫。
4. 在 Windows 和 Linux 系統上，使用系統功能表、 `MQExplorer` 執行檔或 **strmqcfg** 指令來啟動 IBM MQ Explorer 。
5. 從 IBM MQ Explorer 工具列中，按一下 **視窗-> 喜好設定**，然後展開 **IBM 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 105 頁的『管理 IBM MQ Explorer 之系統上的作業』步驟 1 中所建立之 JKS 檔的名稱和密碼，然後按一下 **確定**。
6. 關閉「**喜好設定**」視窗，然後用滑鼠右鍵按一下 **佇列管理程式**。 按一下 **顯示/隱藏佇列管理程式**，然後在「**顯示/隱藏佇列管理程式**」畫面上按一下 **新增**。
7. 鍵入佇列管理程式的名稱，然後選取 **直接連接** 選項。 按「下一步」。
8. 選取 **使用用戶端通道定義表 (CCDT)**，並在管理遠端佇列管理程式的系統上，指定您在 第 105 頁的『管理遠端佇列管理程式之系統上的作業』 的步驟 2 中從遠端佇列管理程式傳送之通道表檔案的位置。
9. 按一下**完成**。 您現在可以從「IBM MQ Explorer」存取遠端佇列管理程式。

使用 *IBM MQ Explorer* 透過另一個佇列管理程式進行連接

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

在此情況下， IBM MQ Explorer 會將 PCF 指令訊息放置到中間佇列管理程式，並指定下列指令:

- 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。

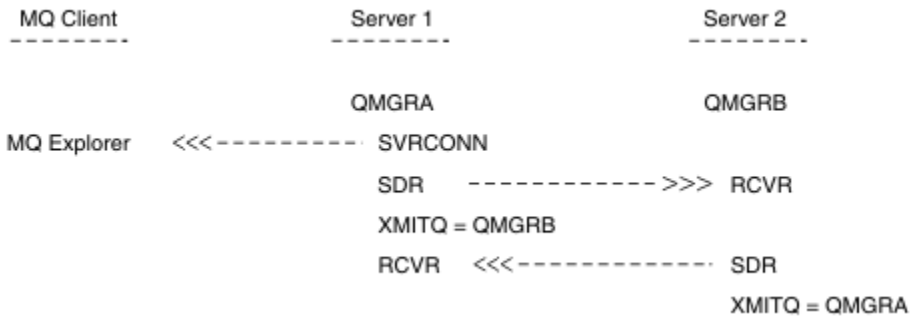
如果隨後使用連線透過中繼佇列管理程式連接至目標佇列管理程式，則 userId 會再次在訊息描述子 (MQMD) 的 *UserIdentifier* 參數中傳送。 為了讓目標佇列管理程式上的 MCA 接聽器接受此訊息，必須設定 MCAUSER 屬性，或 userId 必須已存在且具有放置權限。

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

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

建立與佇列管理程式的遠端管理連線。 驗證:

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



在此範例中:

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

在此狀況下，沒有從 IBM MQ Explorer 到 QMGRB 的直接連線; 與 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 163 頁的『配置佇列管理程式以進行遠端管理』。

在「IBM MQ Explorer」中顯示及隱藏佇列管理程式和叢集

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

如果要顯示遠端佇列管理程式，請執行下列動作:

1. 用滑鼠右鍵按一下 **佇列管理程式** 樹狀結構節點，然後選取 **顯示/隱藏佇列管理程式**。
2. 按一下**新增** 。 這時會顯示「顯示/隱藏佇列管理程式」畫面。

3. 在提供的欄位中輸入遠端佇列管理程式的名稱及主機名稱或 IP 位址。

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

4. 按一下**完成**。

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

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

如果要從這個畫面匯出遠端佇列管理程式清單，請執行下列動作:

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

如果要匯入遠端佇列管理程式的清單，請執行下列動作:

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

叢集成員資格和 IBM MQ Explorer

IBM MQ Explorer 需要屬於叢集成員之佇列管理程式的相關資訊。

如果佇列管理程式是叢集的成員，則會自動移入叢集樹狀結構節點。

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

使用此資訊， IBM MQ Explorer 可以:

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

在下列情況下，無法進行管理:

- 選擇的儲存庫變成無法使用。 IBM MQ Explorer 不會自動切換至替代儲存庫。
- 無法透過 TCP/IP 聯絡選擇的儲存庫。
- 選擇的儲存庫正在平台上執行的佇列管理程式上執行，且 IBM MQ Explorer 不支援指令層次。

可以管理的叢集成員可以是本端成員，或者如果可以使用 TCP/IP 來聯絡它們，則可以是遠端成員。 「 IBM MQ Explorer 」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。

IBM MQ Explorer 的資料轉換

IBM MQ Explorer 以 CCSID 1208 (UTF-8) 運作。 這可讓「IBM MQ Explorer」正確地顯示遠端佇列管理程式中的資料。 無論是直接連接至佇列管理程式，還是使用中繼佇列管理程式， IBM MQ Explorer 都需要將所有送入訊息轉換為 CCSID 1208 (UTF-8)。

如果您嘗試在 IBM MQ Explorer 與具有 IBM MQ Explorer 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 使用 IBM MQ 工作列應用程式 (僅限 Windows)

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

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

綠色
正常運作; 目前沒有警示

藍
不確定; IBM MQ 正在啟動或關閉

黃色
警示; 一個以上服務失敗或已失敗

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

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

Windows IBM MQ 警示監視器應用程式 (僅限 Windows)

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

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

從 IBM MQ 警示監視器中，您可以:

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

使用本端 IBM MQ 物件

您可以管理本端 IBM MQ 物件，以支援使用「訊息佇列介面 (MQI)」的應用程式。

關於這項作業

在此環境定義中，本端管理是指建立、顯示、變更、複製及刪除 IBM MQ 物件。

除了本節中說明的方法之外，您還可以使用 IBM MQ Explorer 來管理本端 IBM MQ 物件。 如需相關資訊，請參閱第 101 頁的『使用 IBM MQ Explorer 進行管理』。

程序

- 請使用下列主題中的資訊來協助您管理本端 IBM MQ 物件。
  - 使用 MQI 的應用程式
  - 第 11 頁的『使用 MQSC 指令進行管理』
  - 第 115 頁的『顯示及變更佇列管理程式屬性』
  - 第 118 頁的『使用本端佇列』
  - 第 127 頁的『使用別名佇列』
  - 第 129 頁的『使用模型佇列』
  - 第 152 頁的『使用服務』
  - 第 158 頁的『管理用於觸發的物件』

使用佇列管理程式

您可以使用控制指令來啟動和停止佇列管理程式。 您可以使用 MQSC 指令來顯示或變更佇列管理程式屬性。

相關工作

在 Multiplatforms 上建立佇列管理程式

Multi 啟動佇列管理程式

當您建立佇列管理程式時，必須啟動它，才能讓它處理指令或 MQI 呼叫。

關於這項作業

您可以使用 **strmqm** 指令來啟動佇列管理程式。 如需 **strmqm** 指令及其選項的說明，請參閱 strmqm。

Windows Linux 或者，在 Windows 和 Linux (x86 和 x86-64 平台) 系統上，您可以使用 IBM MQ Explorer 來啟動佇列管理程式。

Windows 在 Windows 上，當系統使用 IBM MQ Explorer 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 101 頁的『使用 IBM MQ Explorer 進行管理』。

程序

- 若要使用 **strmqm** 指令來啟動佇列管理程式，請輸入指令，後面接著您要啟動的佇列管理程式名稱。 例如，若要啟動稱為 QMB 的佇列管理程式，請輸入下列指令:

  ```
  strmqm QMB
  ```

  註: 您必須從與您使用的佇列管理程式相關聯的安裝中使用 **strmqm** 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。

  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。

- Windows Linux

  若要使用「IBM MQ Explorer」來啟動佇列管理程式，請完成下列步驟:

  a) 開啟 IBM MQ Explorer。

  b) 在 Navigator 視圖中，選取佇列管理程式。

  c) 按一下**啟動**。

結果

即會啟動佇列管理程式。

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

Multi 停止佇列管理程式

您可以使用 **endmqm** 指令來停止佇列管理程式。 此指令提供四種方式來停止佇列管理程式: 受控制或靜止的關機、立即關機、強制關機及等待關機。 或者，在 Windows 及 Linux 上，您可以使用「IBM MQ Explorer」來停止佇列管理程式。

關於這項作業

有四種方法可以使用 **endmqm** 指令來停止單一實例佇列管理程式:

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

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。

強制關機 (preemptive shutdown)
佇列管理程式會立即停止。 只有在異常情況下 (例如，佇列管理程式未因正常 **endmqm** 指令而停止時)，才使用這種類型的關機。

等待關機
這種類型的關閉相當於受控制的關閉，但只有在佇列管理程式停止之後，才會將控制傳回給您。

endmqm 指令會以停止單一實例佇列管理程式的相同方式，停止多重實例佇列管理程式的所有實例。 您可以在作用中實例或多重實例佇列管理程式的其中一個待命實例上發出 **endmqm** 。 不過，您必須在作用中實例上發出 **endmqm** ，以結束佇列管理程式。

V 9.2.0 從 IBM MQ 9.1.4 開始，您可以選擇在指定秒數的目標時間內結束佇列管理程式，如需詳細資料，請參閱 第 111 頁的『在目標時間內結束佇列管理程式』 。

如需 **endmqm** 指令及其選項的詳細說明，請參閱 endmqm。

提示: 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)

如果在嘗試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，但這次使用的參數會指定您需要的關機類型。

Windows Linux 作為使用 **endmqm** 指令的替代方案，在 Windows 和 Linux 上，您可以使用 IBM MQ Explorer 來停止佇列管理程式，以執行受控制或立即關閉。

程序

- 如果要使用 **endmqm** 指令來停止佇列管理程式，請輸入指令，後面接著適當的參數 (必要的話)，以及您要停止的佇列管理程式名稱。

  註: 您必須從與您使用的佇列管理程式相關聯的安裝中使用 **endmqm** 指令。 若要找出與佇列管理程式相關聯的安裝，請使用 **dspmq** 指令:

  ```
  dspmq -o installation
  ```

  - 若要執行受控制 (靜止) 關機，請輸入 **endmqm** 指令，如下列範例所示，它會停止稱為 QMB 的佇列管理程式:

    ```
    endmqm QMB
    ```

或者，輸入含有 **-c** 參數的 **endmqm** 指令 (如下列範例所示) 相當於 endmqm QMB 指令。

```
endmqm -c QMB
```

在這兩種情況下，控制會立即傳回給您，當佇列管理程式停止時不會通知您。 如果您要指令等到所有應用程式都已停止且佇列管理程式已結束之後才將控制權交還給您，請改用 **-w** 參數，如下列範例所示。

```
endmqm -w QMB
```

- 若要執行立即關閉，請輸入含有 **-i** 參數的 **endmqm** 指令，如下列範例所示:

```
endmqm -i QMB
```

- 若要執行強制關機，請輸入帶有 **-p** 參數的 **endmqm** 指令，如下列範例所示:

```
endmqm -p QMB
```

小心: 先佔式關機可能會對已連接的應用程式造成無法預期的結果。 除非使用一般 **endmqm** 指令來停止佇列管理程式的所有其他嘗試都失敗，否則請不要使用此選項。 ALW 如果先佔關機無法運作，請改為嘗試 第 112 頁的『手動停止佇列管理程式』 。

- 若要要求 自動用戶端重新連線，請輸入 **endmqm** 指令並指定 -r 參數。 此參數會重新建立用戶端與其佇列管理程式群組中其他佇列管理程式的連線功能。

  註: 使用預設 **endmqm** 指令來結束佇列管理程式不會觸發自動用戶端重新連線。
- 若要在關閉作用中實例之後傳送至多重實例佇列管理程式的待命實例，請在多重實例佇列管理程式的作用中實例上輸入 **endmqm** 指令並指定 -s 參數。
- 若要結束多重實例佇列管理程式的待命實例，並讓作用中實例繼續執行，請在多重實例佇列管理程式的待命實例上輸入 **endmqm** 指令並指定 -x 參數。

- Windows Linux

  在 Windows 和 Linux 上，若要使用 IBM MQ Explorer 來停止佇列管理程式，請完成下列步驟:

  a) 開啟 IBM MQ Explorer。

  b) 從「Navigator」視圖中選取佇列管理程式。

  c) 按一下**停止**。

  即會顯示「**結束佇列管理程式**」畫面。

  d) 選取 **已控制**或 **立即**。

  e) 按一下**確定**。

  佇列管理程式停止。

相關工作

將維護層次更新套用至 AIX 上的多重實例佇列管理程式

將維護層次更新套用至 Linux 上的多重實例佇列管理程式

將維護層次更新套用至 Windows 上的多重實例佇列管理程式

在目標時間內結束佇列管理程式

您可以在指定秒數的目標時間內結束佇列管理程式，不論是否岔斷基本佇列管理程式維護作業。

當您使用 **endmqm** 指令時，有兩種方法可以指定目標時間。 **-t** 選項容許完成必要的佇列管理程式維護作業，這可能會延長佇列管理程式結束的階段。 必要的話， **-tp** 選項會岔斷必要的佇列管理程式維護作業，以符合指定的目標時間。

當您指定目標時間時，關閉類型 **-w**、 **-i** 或 **-p** 指出開始關閉類型。

註: immediate 關機仍有順序，與 controlled 關機的差異主要在於任何執行中應用程式都靜止的方式。 immediate 關機仍會採取佇列壓縮之類的內部管理動作，以及持續保存可能耗時的 NPMCLASS (HIGH) 訊息，而時間限制關機會在它們干擾達到目標時間時退出這些額外動作。

佇列管理程式會根據需要呈報關閉類型，以嘗試符合目標時間。 例如：

- 從 **-w** 開始的 10 秒 **-t** 目標可能是 7 秒靜止，立即關閉佇列管理程式 2 秒 (包括內部管理)，然後立即關閉而不進一步內部管理：

```
endmqm  -w -t 10 queue_manager
```

- 10 秒 **-tp** 目標可能是 7 秒靜止中， 2 秒立即關閉佇列管理程式 (包括內部管理)， 1 秒立即關閉而不進一步內部管理，然後開始結束 IBM MQ 處理程序：

```
endmqm -c -tp 10 queue_manager
```

- 位於 **-i** 的兩秒 **-tp** 目標可能是佇列管理程式的一秒立即關閉 (包括內部管理)，一秒立即關閉 (不需要進一步內部管理)，然後開始結束 IBM MQ 處理程序：

```
endmqm -i -tp 2 queue_manager
```

- **-w** 上的一秒目標可能是 `wait` 上的 0.1 秒，例如，只要夠長，就會將 IBM MQ 回覆碼傳送至已連接的應用程式， 0.9 秒立即關閉佇列管理程式，包括內部管理，然後立即關閉而不進一步內部管理; 然後開始結束 IBM MQ 處理程序。

相關參考

endmqm (結束佇列管理程式)

ALW 手動停止佇列管理程式

如果停止及移除佇列管理程式的標準方法失敗，您可以嘗試手動停止佇列管理程式。

關於這項作業

停止佇列管理程式的標準方式是使用 **endmqm** 指令，如 第 110 頁的『停止佇列管理程式』中所述。 如果您無法以標準方式停止佇列管理程式，您可以嘗試手動停止佇列管理程式。 您執行此動作的方式取決於您使用的平台。

程序

- Windows
  若要在 Windows 上停止佇列管理程式，請參閱 第 112 頁的『在 Windows 上手動停止佇列管理程式』。
- Linux AIX
  若要在 AIX 或 Linux 上停止佇列管理程式，請參閱 第 113 頁的『在 AIX and Linux 上手動停止佇列管理程式』。

相關工作

在 Multiplatforms 上建立及管理佇列管理程式

相關參考

endmqm

Windows 在 *Windows* 上手動停止佇列管理程式

如果您無法使用 **endmqm** 指令來停止 Windows 上的佇列管理程式，您可以嘗試結束任何執行中的處理程序並停止 IBM MQ 服務，以手動停止佇列管理程式。

關於這項作業

提示: 「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 處理程序瀏覽器 (`procexp.exe`) 之類的工具，您可以從 Microsoft 網站 (http://www.microsoft.com) 下載該工具。

若要在「Windows」上停止佇列管理程式，請完成下列步驟。

程序

1. 使用「Windows 作業管理程式」來列出執行中處理程序的名稱 (ID)。
2. 使用「Windows 作業管理程式」或 **taskkill** 指令來結束處理程序，順序如下 (如果它們在執行中):

表 *7:* 執行時要停止的 *Windows* 處理程序	
處理程序名稱	**說明**
AMQZMUC0	重要程序管理程式
AMQZXMA0	執行控制器
AMQZFUMA	OAM 處理程序
AMQZLAA0	LQM 代理程式
AMQZLSA0	LQM 代理程式
AMQZMUF0	公用程式管理程式
AMQZMGR0	程序控制器
AMQZMUR0	可重新啟動的程序管理程式
AMQFQPUB	發佈訂閱程序
AMQFCXBA	分配管理系統工作者處理程序
AMQRMPPA	程序儲存區處理程序
AMQCRSTA	非執行緒回應者工作處理程序
AMQCRS6B	LU62 接收端通道及用戶端連線
AMQRRMFA	儲存庫處理程序 (適用於叢集)
AMQPCSEA	指令伺服器
RUNMQTRM	呼叫伺服器的觸發監視器
RUNMQDLQ	呼叫無法傳送郵件的佇列處理程式
RUNMQCHI	通道起始程式處理程序
RUNMQLSR	通道接聽器處理程序
AMQXSSVN	共用記憶體伺服器

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

Linux AIX 在 *AIX and Linux* 上手動停止佇列管理程式

如果您無法使用 **endmqm** 指令停止 AIX 或 Linux 上的佇列管理程式，則可以嘗試結束任何執行中的處理程序並停止 IBM MQ 服務，以手動停止佇列管理程式。

關於這項作業

若要在 AIX 或 Linux 上停止佇列管理程式，請完成下列步驟。

如果您手動停止佇列管理程式，則可能會取得 FFST，並將 FDC 檔案放置在 `/var/mqm/errors` 中。 這不應視為佇列管理程式中的問題報告。

即使您已使用手動停止佇列管理程式的方法來停止佇列管理程式，佇列管理程式也會正常重新啟動。

程序

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. 使用 **kill** 指令結束仍在執行中的任何佇列管理程式處理程序，並指定使用 **ps** 指令探索到的 PID。
若要結束處理程序，請使用 **kill -KILL <pid>** 或對等的 **kill -9 <pid>** 指令。

   您必須逐一完成您要結束的 PID ， 每次都會發出該指令。

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

   按下列順序結束處理程序:

表 *8:* 執行時要停止的 *AIX and Linux* 處理程序	
處理程序名稱	**說明**
amqzmuc0	重要程序管理程式
amqzxma0	執行控制器
阿姆克茲富馬	OAM 處理程序
amqzlaa0	LQM 代理程式
amqzlsa0	LQM 代理程式
amqzmuf0	公用程式管理程式
amqzmur0	可重新啟動的程序管理程式
amqzmgr0	程序控制器
amqfqpub	發佈訂閱程序
amqfcxba	分配管理系統工作者處理程序
amqrmppa	程序儲存區處理程序
amqcrsta	非執行緒回應者工作處理程序
amqcrs6b	LU62 接收端通道及用戶端連線
amqrrmfa	儲存庫處理程序 (適用於叢集)
amqpcsea	指令伺服器
runmqtrm	呼叫伺服器的觸發監視器
runmqdlq	呼叫無法傳送郵件的佇列處理程式
runmqchi	通道起始程式處理程序
runmqlsr	通道接聽器處理程序

註: 您可以使用 **kill -9** 指令來結束無法停止的處理程序。

Multi 重新啟動佇列管理程式

您可以使用 **strmqm** 指令來重新啟動佇列管理程式，或者在 Windows 及 Linux x86-64 系統上，您可以從 IBM MQ Explorer 重新啟動佇列管理程式。

關於這項作業

您可以使用 **strmqm** 指令來重新啟動佇列管理程式。 如需 **strmqm** 指令及其選項的說明，請參閱 strmqm。

Windows Linux 在 Windows 及 Linux x86-64 系統上，您可以使用與啟動佇列管理程式相同的 IBM MQ Explorer 來重新啟動佇列管理程式。

程序

- 若要使用 **strmqm** 指令來重新啟動佇列管理程式，請輸入指令，後面接著您要重新啟動的佇列管理程式名稱。

  例如，若要啟動稱為 `strmqm saturn.queue.manager` 的佇列管理程式，請輸入下列指令:

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

- Windows Linux

  若要使用「IBM MQ Explorer」來啟動佇列管理程式，請完成下列步驟:

  a) 開啟 IBM MQ Explorer。

  b) 在 Navigator 視圖中，選取佇列管理程式。

  c) 按一下**啟動**。

結果

佇列管理程式會重新啟動。

如果佇列管理程式重新啟動花費數秒以上的時間， IBM MQ 會發出參考訊息，間歇性地詳述啟動進度。

顯示及變更佇列管理程式屬性

您可以使用 MQSC 指令來顯示或變更佇列管理程式屬性。

關於這項作業

您可以使用 **DISPLAY QMGR** 指令來顯示佇列管理程式的佇列管理程式參數，以及使用 **ALTER QMGR** 指令來變更本端佇列管理程式的佇列管理程式參數。

程序

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

  ```
  DISPLAY QMGR
  ```

  下列範例顯示此指令的一般輸出:

  ```
  DISPLAY QMGR
       1 : DISPLAY QMGR
  AMQ8408: Display Queue Manager details.
     QMNAME(QM1)                             ACCTCONO(DISABLED)
     ACCTINT(1800)                           ACCTMQI(OFF)
     ACCTQ(OFF)                              ACTIVREC(MSG)
     ACTVCONO (DISABLED)                     ACTVTRC (OFF)
     ALTDATE(2012-05-27)                     ALTTIME(16.14.01)
     AUTHOREV(DISABLED)                      CCSID(850)
     CHAD(DISABLED)                          CHADEV(DISABLED)
     CHADEXIT( )                             CHLEV(DISABLED)
     CLWLDATA( )                             CLWLEXIT( )
     CLWLLEN(100)                            CLWLMRUC(999999999)
     CLWLUSEQ(LOCAL)                         CMDEV(DISABLED)
     CMDLEVEL(800)                           COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)
     CONFIGEV(DISABLED)                      CRDATE(2011-05-27)
     CRTIME(16.14.01)                        DEADQ()
     DEFXMITQ( )                             DESCR( )
     DISTL(YES)                              INHIBTEV(DISABLED)
     IPADDRV(IPV4)                           LOCALEV(DISABLED)
     LOGGEREV(DISABLED)                      MARKINT(5000)
     MAXHANDS(256)                           MAXMSGL(4194304)
     MAXPROPL(NOLIMIT)                       MAXPRTY(9)
     MAXUMSGS(10000)                         MONACLS(QMGR)
     MONCHL(OFF)                             MONQ(OFF)
  ```

```
   PARENT( )                               PERFMEV(DISABLED)
   PLATFORM(WINDOWSNT)                        PSRTYCNT(5)
   PSNPMSG(DISCARD)                        PSNPRES(NORMAL)
   PSSYNCPT(IFPER)                         QMID(QM1_2011-05-27_16.14.01)
   PSMODE(ENABLED)                         REMOTEEV(DISABLED)
   REPOS( )                                REPOSNL( )
   ROUTEREC(MSG)                           SCHINIT(QMGR)
   SCMDSERV(QMGR)                          SSLCRLNL( )
   SSLCRYP( )                              SSLEV(DISABLED)
   SSLFIPS(NO)                             SSLKEYR(C:\Program Files\IBM\WebSphere
MQ\Data\qmgrs\QM1\ssl\key)
   SSLRKEYC(0)                             STATACLS(QMGR)
   STATCHL(OFF)                            STATINT(1800)
   STATMQI(OFF)                            STATQ(OFF)
   STRSTPEV(ENABLED)                       SYNCPT
   TREELIFE(1800)                          TRIGINT(999999999)
```

註: SYNCPT 是唯讀佇列管理程式屬性。

ALL 參數是 **DISPLAY QMGR** 指令上的預設值。 它會顯示所有佇列管理程式屬性。 尤其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。

- 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。

  未在 **ALTER QMGR** 指令中指定的參數會導致那些參數的現有值保持不變。

相關工作

在 Multiplatforms 上建立佇列管理程式

相關參考

佇列管理程式的屬性

runmqsc (執行 MQSC 指令)

DISPLAY QMGR

ALTER QMGR

Multi 刪除佇列管理程式

您可以使用 **dltmqm** 指令來刪除佇列管理程式。 或者，在 Windows 及 Linux 系統上，您可以使用 IBM MQ Explorer 來刪除佇列管理程式。

開始之前

小心:

- 刪除佇列管理程式是一個重大步驟，因為您也會刪除與佇列管理程式相關聯的所有資源，包括所有佇列及其訊息和所有物件定義。 如果您使用 **dltmqm** 指令，則沒有可讓您改變主意的顯示提示; 當您按 Enter 鍵時，所有相關聯的資源都會遺失。
- Windows 在「Windows」上，刪除佇列管理程式也會從自動啟動清單中移除佇列管理程式 (如第 109 頁的『啟動佇列管理程式』中所述)。 當指令完成時，會顯示 IBM MQ queue manager ending 訊息; 系統不會告訴您佇列管理程式已刪除。

- 刪除叢集佇列管理程式並不會將它從叢集中移除。 如需相關資訊，請參閱 dltmqm 中的使用注意事項。

關於這項作業

您可以使用 **dltmqm** 指令來刪除佇列管理程式。 如需 **dltmqm** 指令及其選項的說明，請參閱 dltmqm。 請確定只有授信管理者才有權使用此指令。 (如需安全的相關資訊，請參閱 在 AIX, Linux, and Windows 上設定安全。)

Windows Linux 或者，在 Windows 和 Linux (x86 和 x86-64 平台) 系統上，您可以使用 IBM MQ Explorer 來刪除佇列管理程式。

程序

- 若要使用 **dltmqm** 指令來刪除佇列管理程式，請完成下列步驟:
  a) 停止佇列管理程式。
  b) 發出下列指令:

```
dltmqm QMB
```

  註: 您必須從與您使用的佇列管理程式相關聯的安裝中使用 **dltmqm** 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。

- Windows Linux
  若要使用「IBM MQ Explorer」來刪除佇列管理程式，請完成下列步驟:
  a) 開啟 IBM MQ Explorer。
  b) 在 Navigator 視圖中，選取佇列管理程式。
  c) 如果佇列管理程式未停止，請停止它。
  若要停止佇列管理程式，請在該佇列管理程式上按一下滑鼠右鍵，然後按一下 **停止**。
  d) 刪除佇列管理程式。
  如果要刪除佇列管理程式，請用滑鼠右鍵按一下它，然後按一下 **刪除**。

結果
已刪除佇列管理程式。

停止 MQI 通道

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

可以用三種模式來發出 STOP CHANNEL 指令，指出如何停止通道:

靜止
在處理任何現行訊息之後停止通道。

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

強制
立即停止通道。

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

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

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

相關概念

通道

使用本端佇列

本節包含可用來管理本端、模型及別名佇列的部分 MQSC 指令範例。

如需這些指令的詳細資訊，請參閱 MQSC 指令 。

相關參考

佇列的命名限制

其他物件的命名限制

使用 DEFINE QLOCAL 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。

關於這項作業

您可以使用 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值，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 是在系統安裝上建立的。

程序

- 若要建立本端佇列，請輸入 **DEFINE QLOCAL** 指令，如下列範例所示。
  在此範例中， **DEFINE QLOCAL** 指令定義名為 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)
```

附註:

1. 除了說明的值之外，範例中顯示的所有屬性值都是預設值。 包含這些範例是為了說明。 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 119 頁的『使用 DISPLAY QUEUE 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 120 頁的『使用 ALTER QLOCAL 或 DEFINE QLOCAL 變更本端佇列屬性』。

相關參考
DEFINE QLOCAL

使用 DISPLAY QUEUE 顯示預設物件屬性

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

關於這項作業

當您定義 IBM MQ 物件時，它會採用您未從預設物件指定的任何屬性。 例如，當您定義本端佇列時，佇列會從預設本端佇列 (稱為 SYSTEM.DEFAULT.LOCAL.QUEUE。 您可以使用 **DISPLAY QUEUE** 指令來查看這些屬性的確切內容。

程序

- 若要顯示本端佇列的預設物件屬性，請使用下列指令:

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

DISPLAY 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，亦即佇列上的訊息數目。 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。

相關參考
顯示佇列
DEFINE 佇列

使用 DEFINE QLOCAL 複製本端佇列定義

您可以在 **DEFINE QLOCAL** 指令上使用 **LIKE** 屬性來複製佇列定義。

關於這項作業

您可以搭配使用 **DEFINE** 指令與 **LIKE** 屬性，以建立與指定佇列具有相同屬性的佇列，而非系統預設本端佇列的佇列。 您也可以使用此形式的 **DEFINE** 指令來複製佇列定義，但會替換原始屬性的一或多項變更。

附註:

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)
```

程序

- 若要建立與指定佇列具有相同屬性的佇列，而不是系統預設本端佇列的佇列，請輸入 **DEFINE** 指令，如下列範例所示。

  輸入要複製的佇列名稱，與您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。

  此範例會建立與佇列 ORANGE.LOCAL.QUEUE，而不是系統預設本端佇列的 QUEUE:

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

- 若要複製佇列定義，但替換原始屬性的一或多項變更，請輸入 **DEFINE** 指令，如下列範例所示。

  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:

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

相關參考

DEFINE 佇列

使用 ALTER QLOCAL 或 DEFINE QLOCAL 變更本端佇列屬性

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

關於這項作業

您可以使用 **ALTER** 和 **DEFINE** 指令的 **REPLACE** 屬性，將現有的定義取代為指定的新定義。 使用 **ALTER** 與 **DEFINE** 之間的差異在於具有 **REPLACE** 的 **ALTER** 不會變更未指定的參數，但具有 **REPLACE** 的 **DEFINE** 會設定所有參數。

程序

- 若要變更佇列屬性，請使用 **ALTER** 指令或 **DEFINE** 指令，如下列範例所示。

  在這些範例中，佇列 ORANGE.LOCAL.QUEUE 會減少到 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
    ```

    此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。 因此，例如，如果先前禁止放置佇列，則會變更為啟用放置，因為已啟用放置是由佇列 SYSTEM.DEFAULT.LOCAL.QUEUE。

    如果您減少現有佇列上的訊息長度上限，則不會影響現有訊息。 不過，任何新訊息都必須符合新準則。

相關參考

ALTER 佇列
ALTER QLOCAL
DEFINE 佇列
DEFINE QLOCAL

使用 CLEAR QLOCAL 清除本端佇列

您可以使用 **CLEAR QLOCAL** 指令來清除本端佇列。

開始之前

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。

關於這項作業

如果您要使用 **CLEAR QLOCAL** 指令來清除本端佇列，則必須對本端佇列管理程式定義佇列名稱。

註: 沒有可讓您改變心意的提示; 一旦您按 Enter 鍵，訊息便會遺失。

程序

若要從本端佇列清除訊息，請使用 **CLEAR QLOCAL** ，如下列範例所示。

在此範例中，所有訊息都從稱為 MAGENTA.QUEUE:

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

相關參考

CLEAR QLOCAL

刪除具有 DELETE QLOCAL 的本端佇列

您可以使用 MQSC 指令 **DELETE QLOCAL** 來刪除本端佇列。

關於這項作業

如果佇列包含未確定的訊息，則無法刪除。

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，才能刪除該佇列。 然後會繼續刪除，即使指名的佇列中有已確定的訊息，也會清除這些訊息。

指定 **NOPURGE** 而非 **PURGE** ，可確保如果佇列包含任何已確定訊息，則不會刪除該佇列。

程序

- 若要刪除本端佇列，請使用 **DELETE QLOCAL** 指令，如下列範例所示。
  此範例會刪除佇列 PINK.QUEUE :

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

  此範例會刪除佇列 PINK.QUEUE 即使佇列中有已確定的訊息:

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

相關參考

DELETE QLOCAL

使用範例程式瀏覽佇列

IBM MQ 提供一個範例佇列瀏覽器，您可以用來查看佇列上訊息的內容。

關於這項作業

瀏覽器在下列位置以來源及執行檔格式提供，其中 *MQ_INSTALLATION_PATH* 代表安裝 IBM MQ 的高階目錄。

Windows 在 Windows 上，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

Linux AIX 在 AIX and Linux 上，檔名及路徑如下:

來源
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

執行檔
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

程序

- 若要執行範例程式，請輸入指令，如下列範例所示。
  範例程式需要兩個輸入參數，即要瀏覽訊息的佇列名稱，以及擁有該佇列的佇列管理程式。 例如:

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

結果

下列範例顯示此指令的一般結果:

```
AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 850
  Format : 'MQEVENT '
  Priority : 0  Persistence : 0
  MsgId : X'414D512073617475726E2E71756575650005D30033563DB8'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'saturn.queue.manager                          '
  ** Identity Context
  UserIdentifier : '            '
  AccountingToken :
   X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '                                             '
  ** Origin Context
  PutApplType    : '7'
  PutApplName    : 'saturn.queue.manager        '
  PutDate  : '19970417'    PutTime  : '15115208'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '104'

****  Message   ****

length - 104 bytes
```

```
00000000: 0700 0000 2400 0000 0100 0000 2C00 0000 '....→.....,...'
00000010: 0100 0000 0100 0000 0100 0000 AE08 0000 '................'
00000020: 0100 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030: 0000 0000 3000 0000 7361 7475 726E 2E71 '....0...saturn.q'
00000040: 7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager    '
00000050: 2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060: 2020 2020 2020 2020                     '                '

No more messages
MQCLOSE
MQDISC
```

相關參考

瀏覽器範例程式

啟用大型佇列

IBM MQ 支援大於 2 TB 的佇列。

Windows 在 Windows 系統上，不需要任何其他啟用，即可支援大型檔案。

Linux AIX 在 AIX and Linux 系統上，您需要明確啟用大型檔案支援，才能建立多個 GB 或 TB 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) 無法處理多個 GB 或 TB 的檔案。 在啟用大型檔案支援之前，請先檢查作業系統文件，以取得您使用之公用程式的限制相關資訊。

如需規劃佇列所需儲存體數量的相關資訊，請參閱 MQ 效能文件 ，以取得平台專用效能報告。

V 9.2.0 從 IBM MQ 9.1.5 開始，您可以使用本端及模型佇列上的新屬性來控制佇列檔的大小。 如需相關資訊，請參閱 第 123 頁的『修改 IBM MQ 佇列檔』。

Multi V 9.2.0 修改 IBM MQ 佇列檔

從 IBM MQ 9.2.0 開始，您可以使用本端及模型佇列上的屬性來控制佇列檔的大小。 您可以使用兩個佇列狀態屬性，來顯示佇列檔的現行大小，以及它目前能夠成長到的大小上限 (根據該檔案中目前使用的區塊大小)。

用來修改佇列檔的屬性

本端及模型佇列上的屬性為:

MAXFSIZE
表示佇列使用的佇列檔大小上限 (MB)。

如需進一步資訊，請參閱 MAXFSIZE 及 第 124 頁的『變更 IBM MQ 佇列檔的大小』 。

此屬性的 PCF 屬性是 MQIA_MAX_Q_FILE_SIZE; 請參閱 變更、複製及建立佇列。

佇列狀態上的兩個屬性如下:

CURFSIZE
顯示佇列檔的現行大小 (以 MB 為單位，四捨五入至最接近的 MB 數)。

如需相關資訊，請參閱 CURFSIZE 。

此屬性的 PCF 屬性是 MQIA_CUR_Q_FILE_SIZE。

CURMAXFS
指出佇列檔可以增長到的現行大小上限，如果佇列上使用中的現行區塊大小，則會四捨五入至最接近的 MB 數。

如需相關資訊，請參閱 CURMAXFS 。

此屬性的 PCF 屬性是 MQIA_CUR_MAX_FILE_SIZE。

如需這兩個 PCF 屬性的進一步資訊，請參閱 查詢佇列 及 查詢佇列 (回應) 。

您可以使用 MQSC 指令、 IBM MQ Explorer 及 administrative REST API 來設定及顯示這些屬性。

註: 您只能在 IBM MQ Console 中顯示 MAXFSIZE 的值; 您無法配置該值。

區塊大小及精度

佇列檔分成稱為區塊的區段。 若要增加佇列檔的大小上限，佇列管理程式可能需要變更佇列的區塊大小或精度。

如果以較大的 MAXFSIZE 值建立新定義的佇列，則會以適當的區塊大小建立佇列。 不過，如果現有佇列已增加其 MAXFSIZE 值 (例如，使用 ALTER QLOCAL 指令)，則可能需要容許清空佇列，佇列管理程式才能重新配置佇列。

如需相關資訊，請參閱 第 125 頁的『計算 IBM MQ 佇列檔可以儲存多少資料』。

小心: 部分檔案系統及作業系統對整個檔案系統的大小或個別檔案的大小有限制。 您應該檢查企業使用的系統限制。

相關參考

ALTER QUEUES

顯示佇列

DISPLAY QSTATUS

Multi V 9.2.0 變更 *IBM MQ* 佇列檔的大小

您可以增加或減少佇列檔的大小上限。

開始之前

在設定佇列檔的新大小之前，請使用 DISPLAY QLOCAL 指令來查看您要變更的佇列檔大小。 例如，發出下列指令:

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

您會收到下列輸出:

```
AMQ8409I: Display queue details
  QUEUE(SYSTEM.DEFAULT.LOCAL.QUEUE)        TYPE(QLOCAL)
  MAXFSIZE(DEFAULT)
```

其顯示佇列檔的大小上限是預設值 2,088,960 MB。

關於這項作業

下列程序教您如何:

- 減少佇列檔可成長至的大小上限。
- 增加佇列檔可成長至的大小上限。

小心: 您應該小心增加佇列檔的大小，而不考量應用程式的寫入方式及對效能的可能影響。 在非常大的佇列檔中隨機存取訊息可能會非常緩慢。

如果您考慮增加佇列檔的大小上限超過預設值，則應該小心使用訊息選取元，例如相關性 ID 及 IBM MQ classes for JMS 選取元字串。 較大的佇列檔更適合先進先出存取佇列。

個別佇列檔中有非常大量的資料，只能在配置成循環式記載的佇列管理程式上執行，或在個別佇列尚未啟用媒體影像處理的佇列管理程式上執行。

您不應限制 SYSTEM 佇列的大小，因為這可能會影響佇列管理程式的作業。

程序

1. 減少佇列檔大小上限
   a) 發出下列指令來建立稱為 SMALLQUEUE 的本端檔案，大小為 500 GB:

```
DEFINE QLOCAL(SMALLQUEUE) MAXFSIZE(512000)
     2 : DEFINE QLOCAL(SMALLQUEUE) MAXFSIZE(512000)
AMQ8006I: IBM MQ queue created
```

您會收到訊息: AMQ8006I:

註: 如果您配置的佇列值小於檔案中已存在的資料量，則無法將新訊息放入佇列。

如果應用程式嘗試將訊息放置到沒有足夠空間的佇列檔，則應用程式會收到回覆碼 MQRC_Q_SPACE_NOT_AVAILABLE。 當從佇列破壞性地讀取足夠的訊息時，應用程式可以開始將新訊息放入佇列。

2. 增加佇列檔大小上限。

a) 發出下列指令，以建立稱為 LARGEQUEUE 的本端檔案，大小為 5 TB:

```
DEFINE QLOCAL(LARGEQUEUE) MAXFSIZE(5242880)
     3 : DEFINE QLOCAL(LARGEQUEUE) MAXFSIZE(5242880)
AMQ8006I: IBM MQ queue created
```

Multi V 9.2.0 計算 *IBM MQ* 佇列檔可以儲存多少資料

可以儲存在佇列上的資料量受佇列分割成的個別區塊大小限制。

區塊大小及精度

預設區塊大小是 512 位元組。 若要支援大於 2 TB 的佇列檔，佇列管理程式將需要增加區塊大小。

當您配置佇列的 MAXFSIZE 時，會自動計算區塊大小，但如果佇列上已有訊息，則無法將已修訂的區塊大小套用至佇列。 佇列清空之後，佇列管理程式會自動修改區塊大小，以支援所配置的 MAXFSIZE。

DISPLAY QSTATUS 指令有一個新的屬性 CURMAXFS，可讓您確認佇列已修改為使用新的區塊大小。

在下列範例中， CURMAXFS 值 4177920 確認佇列檔目前能夠成長到大約 4 TB 大小。 如果佇列上配置的 MAXFSIZE 值大於 CURMAXFS 值，則佇列管理程式仍在等待佇列清空，然後再重新配置佇列檔的區塊大小。

```
DISPLAY QSTATUS(LARGEQUEUE)  CURMAXFS
     2 : DISPLAY QSTATUS(LARGEQUEUE)  CURMAXFS
AMQ8450I: Display queue status details
  QUEUE(LARGEQUEUE)                       TYPE(QUEUE)
  CURMAXFS(4177920)                       CURDEPTH(100000)
```

檢查佇列檔的大小

您可以使用 DISPLAY QSTATUS 指令中的 CURFSIZE 屬性來顯示磁碟上佇列檔的現行大小 (以 MB 為單位)。 在無法直接存取檔案系統的平台 (例如 IBM MQ Appliance) 上，這可能很有用。

```
DISPLAY QSTATUS(SMALLQUEUE)  CURFSIZE
     1 : DISPLAY QSTATUS(SMALLQUEUE)  CURFSIZE
AMQ8450I: Display queue status details
  QUEUE(SMALLQUEUE)                       TYPE(QUEUE)
  CURDEPTH(4024)                          CURFSIZE(10)
```

註: 當佇列中移除訊息時， CURFSIZE 屬性可能不會立即減少。

通常，只有在下列情況下，才會釋放佇列檔中的未用空間:

- 當沒有應用程式開啟佇列時
- 在 1000 次寫入佇列管理程式日誌之後，或
- 在佇列管理程式關閉時

相關參考

ALTER QUEUES

DISPLAY QSTATUS

使用遠端佇列

遠端佇列是遠端佇列的本端定義。 亦即，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。

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及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。)

在此之後，應用程式會發出 MQPUT 呼叫，將訊息放入此佇列。

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，必須將適當的傳輸佇列定義為具有 **Usage** 屬性的本端佇列，該屬性指定它是 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。

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，訊息才會到達其目的地。

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。

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。

小心: 發佈清單不支援使用指向主題物件的別名佇列。 如果別名佇列指向配送清單中的主題物件，則 IBM MQ 會傳回 MQRC_ALIAS_BASE_Q_TYPE_ERROR。

別名佇列所參照的佇列可以是下列任何一項:

- 本端佇列 (請參閱 第 118 頁的『使用 DEFINE QLOCAL 定義本端佇列』)。
- 遠端佇列的本端定義 (請參閱 第 126 頁的『使用遠端佇列』)。
- 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，而不需要變更應用程式，它會繼續使用別名。)

例如，假設已開發應用程式，將訊息放置在稱為 MY.ALIAS.QUEUE。 當它發出 MQOPEN 要求時，以及間接將訊息放在此佇列上時，它會指定此佇列的名稱。 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。

下列指令會定義針對應用程式 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 **DISPLAY QALIAS** 指令來顯示別名佇列的屬性:

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

使用 **ALTER QALIAS** 指令來變更別名所解析成的基本佇列名稱，其中 `force` 選項會強制變更，即使佇列已開啟:

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

使用 **DELETE QALIAS** 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。

相關參考

ALTER QALIAS
DEFINE QALIAS
DELETE QALIAS
分送清單

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 **DEFINE QMODEL** 指令，以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。 (永久佇列會在佇列管理程式重新啟動之間維護，暫時佇列則不會。) 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，方法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 **DISPLAY QUEUE** 指令來顯示模型佇列的屬性:

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

使用 **ALTER QMODEL** 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 **DELETE QMODEL** 指令來刪除此模型佇列:

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

相關參考
ALTER QMODEL
DEFINE QMODEL
DELETE QMODEL
顯示佇列

使用無法傳送郵件的佇列

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

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

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

無法傳送郵件的佇列沒有特殊需求，除了:

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

使用無法傳送郵件的佇列可能會影響遞送訊息的順序，因此您可能選擇不使用它們。 您可以設定 USEDLQ 通道屬性，以判定無法遞送訊息時是否使用無法傳送郵件的佇列。 此屬性可以配置為佇列管理程式的部分功能使用無法傳送郵件的佇列，而其他功能則不使用。 如需在不同 MQSC 指令上使用 USEDLQ 通道屬性的相關資訊，請參閱 DEFINE CHANNEL、 DISPLAY CHANNEL、 ALTER CHANNEL 及 DISPLAY CLUSQMGR。

IBM MQ 提供無法傳送郵件的佇列處理程式，可讓您指定如何處理或移除在無法傳送郵件的佇列上找到的訊息。 請參閱第 130 頁的『處理 IBM MQ 無法傳送郵件的佇列上的訊息』。

相關概念
無法傳送郵件的佇列
相關工作
未遞送訊息疑難排解
相關參考
ALTER QMGR
crtmqm (建立佇列管理程式)

處理 IBM MQ 無法傳送郵件的佇列上的訊息

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

訊息可以由佇列管理程式、訊息通道代理程式 (MCA) 及應用程式放置在 DLQ 上。 DLQ 上的所有訊息都必須以 無法傳送郵件的標頭 結構 MQDLH 作為字首。 佇列管理程式或訊息通道代理程式放置在 DLQ 上的訊息一

律具有此標頭; 將訊息放置在 DLQ 上的應用程式必須提供此標頭。 MQDLH 結構的 原因 欄位包含原因碼，可識別訊息在 DLQ 上的原因。

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

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

相關概念

無法傳送郵件的佇列

相關工作

未遞送訊息疑難排解

IBM i *IBM i 上無法傳送郵件的佇列處理程式*

何謂 IBM i 無法傳送郵件的佇列處理程式，以及如何呼叫它?

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

註: 通常最好避免在 DLQ 上放置訊息。 如需使用及避免 DLQ 的相關資訊，請參閱 第 130 頁的『使用無法傳送郵件的佇列』。

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

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

呼叫 DLQ 處理程式

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

- 作為指令提示中 STRMQMDLQ 的參數。 例如:

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

- 在規則表格中。 例如:

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

註: 規則表是可以採用任何名稱的來源實體檔內的成員。

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

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

STRMQMDLQ 指令會從規則表格取得其輸入。

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

相關概念

無法傳送郵件的佇列

相關工作

未遞送訊息疑難排解

IBM i *IBM i 上的 DLQ 處理程式規則表格*

無法傳送郵件的佇列處理程式規則表格定義 DLQ 處理程式如何處理到達 IBM i DLQ 的訊息。

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

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

規則表格中的每一個項目都包含一個以上關鍵字。

控制資料

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

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

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

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

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

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

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

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

YES
導致 DLQ 處理程式無限期等待。

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

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

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

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

註: 如果控制資料項目包含在規則表格中，則它必須是表格中的第一個項目。

IBM i *IBM i 上的 DLQ 規則 (型樣及動作)*

每一個 IBM i 無法傳送郵件的佇列規則的型樣及動作說明。

以下是 DLQ 處理程式規則表格中的規則範例:

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

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

本節說明您可以包含在規則中的關鍵字。 請注意下列項目:

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

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

IBM i *IBM i 上的 DLQ 型樣相符關鍵字*

範例中說明型樣相符關鍵字。 使用這些關鍵字來指定符合 IBM i 無法傳送郵件之佇列上的訊息的值。 所有型樣相符關鍵字都是選用的。

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

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

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

DESTQ (*QueueName*|* (預設值))
指定訊息的訊息佇列名稱。

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

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

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

FORMAT (*Format*|* (預設值))
訊息傳送者用來說明訊息資料格式的名稱。

MSGTYPE (*MsgType*|* (預設值))
DLQ 上訊息的訊息類型。

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

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

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

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

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

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

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

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

IBM i *IBM i* 上的 *DLQ* 動作關鍵字
使用這些無法傳送郵件的佇列動作關鍵字，可判定如何處理 IBM i 無法傳送郵件之佇列上的相符訊息。

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

捨棄
導致從 DLQ 中刪除訊息。

IGNORE
導致訊息保留在 DLQ 上。

重試
導致 DLQ 處理程式重試將訊息放入其目的地佇列。

轉遞
導致 DLQ 處理程式將訊息轉遞至 FWDQ 關鍵字上指定的佇列。

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

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

QueueName
訊息佇列的名稱。 FWDQ ('') 無效。

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

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

You can specify REPLYQ (?*) in the message pattern to avoid error messages, when a rule specifying FWDQ (&REPLYQ) matches a message with a blank *ReplyToQ* field.

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

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

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

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

''
FWDQM ('') 是預設值，用於識別本端佇列管理程式。

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

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

DEF
放置具有 DLQ 處理程式本身權限的訊息。

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

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

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

IBM i *IBM i 上的 DLQ 規則表格使用慣例*
IBM i 無法傳送郵件的佇列規則表格必須遵循關於其語法、結構及內容的特定慣例。

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

  例如:

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

  產生 'ABCD'。

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

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

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

這些參數無效，因為它們不包含任何有效字元:

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

- 萬用字元。 您可以使用問號 (?) 來取代任何單一字元，但尾端空白除外。 您可以使用星號 (*) 來取代零個以上相鄰字元。 在參數值中，星號 (*) 及問號 (?) 一**律** 解譯為萬用字元。
- 您無法在下列關鍵字的參數中包含萬用字元 :ACTION、HEADER、RETRY、FWDQ、FWDQM 及 PUTAUT。
- 當執行萬用字元相符時，參數值中的尾端空白，以及 DLQ 上訊息中的對應欄位中的尾端空白並不重要。 不過，引號中字串內的前導和內嵌空白對萬用字元相符很重要。
- 數值參數不能包含問號 (?) 萬用字元。 您可以併入星號 (*) 來取代整個數值參數，但不能併入星號作為數值參數的一部分。 例如，這些是有效的數值參數:

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

  不過， `MSGTYPE('2*')` 無效，因為它包含星號 (*) 作為數值參數的一部分。
- 數值參數必須在 0-999 999 999 的範圍內。 如果參數值在此範圍內，則會接受它，即使它目前在關鍵字相關的欄位中無效也一樣。 您可以對數值參數使用符號名稱。
- 如果字串值短於關鍵字相關的 MQDLH 或 MQMD 中的欄位，則該值會以空白填補欄位長度。 如果值 (排除星號) 比欄位長，則會診斷錯誤。 例如，這些都是 8 個字元欄位的有效字串值:

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

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

IBM i 在 *IBM i* 上處理 *DLQ* 規則表格

無法傳送郵件的佇列處理程式會在規則表格中搜尋其型樣符合 IBM i 無法傳送郵件之佇列上的訊息的規則。

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

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

註:

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

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

IBM i 確保在 *IBM i* 上處理所有 *DLQ* 訊息

無法傳送郵件的佇列處理程式會保留 IBM i DLQ 上已看到但未移除之所有訊息的記錄。

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

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

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

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

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

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

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

IBM i *IBM i* 上的 *DLQ* 處理程式規則表格範例

IBM i 上無法傳送郵件的佇列處理程式規則表格的程式碼範例。 此範例規則表格包含單一控制資料輸入及數個規則。

```
********************************************************************************
*    An example rules table for the STRMQMDLQ command        *
********************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* STRMQMDLQ, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to STRMQMDLQ,
* use the DLQ defined for the local queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.

* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)
```

```
* The AAAA corporation is always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never does things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it must be able
* to cope with the message being lost, so we can afford to
* discard the message.

PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)

* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We do not have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

呼叫 *DLQ* 處理程式

使用 runmqdlq 指令呼叫無法傳送郵件的佇列處理程式。 您可以用兩種方式來命名您要處理的 DLQ 及您要使用的佇列管理程式。

這兩種方式如下:

- 作為命令提示字元中 runmqdlq 的參數。 例如:

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

- 在規則表格中。 例如:

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

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

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

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

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

如需 runmqdlq 指令的相關資訊，請參閱 runmqdlq。

相關概念
無法傳送郵件的佇列
相關工作
未遞送訊息疑難排解

範例 *DLQ* 處理程式 ***amqsdlq***
除了使用 **runmqdlq** 指令所呼叫的無法傳送郵件的佇列處理程式之外， IBM MQ 還提供範例 DLQ 處理程式 **amqsdlq** 的來源，其函數類似於 **runmqdlq** 所提供的函數。

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

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

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

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

並在下列目錄中提供已編譯版本:

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

在 IBM MQ for UNIX 和 Linux 系統中， **amqsdlq** 的來源是在下列目錄中提供:

MQ_INSTALLATION_PATH/samp/dlq

並在下列目錄中提供已編譯版本:

MQ_INSTALLATION_PATH/samp/bin

V 9.2.3 包括範例程式的建置版本 (名為 **amqsdlqc**)。 您可以使用此選項，以用戶端模式連接至遠端佇列管理程式。 若要使用 **amqsdlqc**，您必須設定其中一個環境變數 **MQSERVER**、 **MQCHLLIB** 或 **MQCHLTAB**，以識別如何連接至佇列管理程式。 例如:

```
export MQSERVER="SYSTEM.DEF.SVRCONN/TCP/myappliance.co.uk(1414)"
```

DLQ 處理程式規則表格
無法傳送郵件的佇列處理程式規則表格定義 DLQ 處理程式如何處理到達 DLQ 的訊息。

規則表格中有兩種類型的項目:

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

規則表格中的每一個項目都包含一個以上關鍵字。

相關概念
無法傳送郵件的佇列
相關工作
未遞送訊息疑難排解

DLQ 控制資料
您可以在無法傳送郵件的佇列處理程式規則表格中包含控制資料項目中的關鍵字。

註:

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

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

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

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

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

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

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

YES
DLQ 處理程式無限期等待。

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

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

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

將控制資料併入規則表格的替代方案是提供 DLQ 及其佇列管理程式的名稱作為 runmqdlq 指令的輸入參數。 如果您在規則表格中同時指定一個值，並指定作為 runmqdlq 指令的輸入，則 runmqdlq 指令上指定的值優先。

在規則表中新增的所有控制資料屬性必須是表中的**第一個**項目。

DLQ 規則 *(*型樣及動作*)*
型樣相符關鍵字 (那些與無法傳送郵件的佇列上的訊息相符的關鍵字) 及動作關鍵字 (那些決定 DLQ 處理程式如何處理相符訊息的關鍵字) 的說明。 也提供範例規則。

型樣相符關鍵字

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

APPLIDAT（ *ApplIdentityData* | * （預設））
DLQ 上訊息的訊息描述子 MQMD 中指定的 *ApplIdentityData* 值。

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

APPLTYPE (*PutApplType* | * (預設))
PutApplType 值，在 DLQ 上訊息的訊息描述符 MQMD 中指定。

DESTQ (佇列名稱| * (預設))
指定訊息的訊息佇列名稱。

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

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

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

FORMAT (*Format*|* (預設值))
訊息傳送者用來說明訊息資料格式的名稱。

MSGTYPE（訊息類型| *（預設））
DLQ 上訊息的訊息類型。

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

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

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

原因（原因代碼| *（預設））
說明訊息放置到 DLQ 的原因碼。

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

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

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

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

動作關鍵字

用來說明如何處理相符訊息的動作關鍵字如下:

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

捨棄
從 DLQ 中刪除訊息。

IGNORE
在 DLQ 上保留訊息。

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

轉遞
將訊息轉遞至 FWDQ 關鍵字上指定的佇列。

您必須指定 ACTION 關鍵字。

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

QueueName
訊息佇列的名稱。 FWDQ ('') 無效。

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

&REPLYQ
從訊息描述符 MQMD 中的 *ReplyToQ* 欄位取得佇列名稱。

若要避免在指定 FWDQ (&REPLYQ) 的規則與帶有空白 *ReplyToQ* 欄位的訊息相符時出現錯誤訊息，請在訊息模式中指定 REPLYQ (?*)。

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

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

&DESTQM
從 MQDLH 結構中的 *DestQMgrName* 欄位取得佇列管理器名稱。

&REPLYQM
從訊息描述符 MQMD 中的 *ReplyToQMgr* 欄位取得佇列管理器名稱。

' '
FWDQM ('') 是預設值，用於識別本端佇列管理程式。

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

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

DEF
放置具有 DLQ 處理程式本身權限的訊息。

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

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

範例規則

以下是 DLQ 處理程式規則表格中的規則範例:

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

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

本節其餘部分會說明您可以在規則上使用的所有關鍵字。 請注意下列項目:

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

DLQ 規則表格使用慣例
無法傳送郵件的佇列處理程式規則表格的語法、結構及內容必須遵循這些慣例。

規則表格必須遵循下列慣例:

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

  例如:

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

  產生 'ABCD'， 以及

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

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

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

    這些參數無效，因為它們不包含任何有效字元:

    ```
    FORMAT('')
    FORMAT( )
    FORMAT()
    FORMAT
    ```
  - 萬用字元。 您可以使用問號 (?) 代替任何單一字元，但尾端空白除外; 您可以使用星號 (*) 代替零或多個相鄰字元。 在參數值中，星號 (*) 及問號 (?) 一**律** 解譯為萬用字元。
  - 在下列關鍵字的參數中不能包含萬用字元 :ACTION、HEADER、RETRY、FWDQ、FWDQM 及 PUTAUT。

- 當執行萬用字元相符時，參數值中的尾端空白，以及 DLQ 上訊息中的對應欄位中的尾端空白並不重要。不過，以單引號括住的字串內的前導和內嵌空白對萬用字元相符很重要。
- 數值參數不能包含問號 (?) 萬用字元。 您可以使用星號 (*) 來取代整個數值參數，但不能作為數值參數的一部分。 例如，這些是有效的數值參數:

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

  不過， `MSGTYPE('2*')` 無效，因為它包含星號 (*) 作為數值參數的一部分。
- 數值參數必須在 0-999 999 999 的範圍內。 如果參數值在此範圍內，則會接受它，即使它目前在關鍵字相關的欄位中無效也一樣。 您可以對數值參數使用符號名稱。
- 如果字串值短於關鍵字相關的 MQDLH 或 MQMD 中的欄位，則該值會以空白填補欄位長度。 如果值 (排除星號) 比欄位長，則會診斷錯誤。 例如，這些都是 8 個字元欄位的有效字串值:

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

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

如何處理 *DLQ* 規則表格

無法傳送郵件的佇列處理程式會在規則表格中搜尋型樣符合 DLQ 上訊息的規則。

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

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

註:

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

相關概念

無法傳送郵件的佇列

相關工作

未遞送訊息疑難排解

確定已處理所有 *DLQ* 訊息
無法傳送郵件的佇列處理程式會保留 DLQ 上已看到但未移除之所有訊息的記錄。

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

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

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

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

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

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

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

範例 *DLQ* 處理程式規則表格

runmqdlq 指令的無法傳送郵件的佇列規則表格範例，包含單一控制資料項目及數個規則。

```
*********************************************************************
*    An example rules table for the runmqdlq command        *
*********************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* runmqdlq, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to runmqdlq,
* use the DLQ defined for the local queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.
* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation are always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never do things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
```

```
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)
```

```
* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it should be able
* to cope with the message being lost, so we can afford to
* discard the message.  PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)
* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We don't have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

相關概念

無法傳送郵件的佇列

相關工作

未遞送訊息疑難排解

相關參考

runmqdlq (執行無法傳送郵件的佇列處理程式)

使用管理主題

使用 MQSC 指令來管理管理主題。

如需這些指令的詳細資訊，請參閱 MQSC 指令 。

相關概念

管理主題物件

第 147 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 147 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 148 頁的『變更管理主題屬性』
您可以使用 **ALTER TOPIC** 指令或 **DEFINE TOPIC** 指令與 **REPLACE** 屬性搭配，以兩種方式來變更主題屬性。

第 148 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 148 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 147 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 148 頁的『變更管理主題屬性』。

相關參考

DEFINE TOPIC

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)
```

您可以透過使用 **DISPLAY TOPIC** 指令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** 指令。 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)
```

相關參考
顯示主題
DISPLAY TPSTATUS

變更管理主題屬性

您可以使用 **ALTER TOPIC** 指令或 **DEFINE TOPIC** 指令與 **REPLACE** 屬性搭配，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; 所有其他屬性則維持相同。

- 使用 **DEFINE** 指令:

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

  這個指令會變更遞送至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。

相關參考
ALTER TOPIC
顯示主題

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 **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)
```

相關參考
DEFINE TOPIC

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。

例如:

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

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

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

相關參考
刪除主題

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:

管理者
由使用者以管理方式定義。

Proxy
內部建立的訂閱，用於在佇列管理程式之間遞送發佈。

API
以程式化方式建立，例如，使用 MQI MQSUB 呼叫。

如需這些指令的詳細資訊，請參閱 MQSC 指令 。

相關概念
第 149 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 150 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 151 頁的『變更本端訂閱屬性』
您可以使用 **ALTER SUB** 指令或 **DEFINE SUB** 指令與 **REPLACE** 屬性搭配，以兩種方式來變更訂閱屬性。

第 151 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 151 頁的『刪除本端訂閱』
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** 指令會定義稱為 ORANGE 的訂閱，且具有下列性質:

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

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

註:

- 訂閱與主題字串名稱不必相符。
- 除了目的地和主題字串的值之外，所有顯示的屬性值都是預設值。 它們僅在這裡顯示作為圖解。 如果您確定預設值是您想要的或尚未變更，則可以省略它們。 另請參閱第 150 頁的『顯示訂閱的屬性』。
- 如果您已在名稱為 ORANGE 的相同佇列管理程式上具有本端訂閱，則此指令會失敗。 如果您要改寫佇列的現有定義，請使用 **REPLACE** 屬性，但另請參閱 第 151 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。

相關參考

DEFINE SUB

顯示訂閱的屬性

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

例如，使用:

```
DISPLAY SUB(ORANGE)
```

您可以透過個別指定屬性來選擇性地顯示屬性。 例如:

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

此指令會顯示三個指定的屬性，如下所示:

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

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

顯示訂閱的替代方法是使用 SUBID:

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

此指令提供與之前相同的輸出:

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

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

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

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

會顯示下列輸出:

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

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

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

相關參考
DISPLAY SUB

變更本端訂閱屬性

您可以使用 **ALTER SUB** 指令或 **DEFINE SUB** 指令與 **REPLACE** 屬性搭配，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，還會變更提供其預設值的所有其他屬性。

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。

相關參考
ALTER SUB
DEFINE SUB

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)
```

相關參考
DEFINE SUB

刪除本端訂閱

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)
```

相關參考
DELETE SUB

檢查訂閱的訊息

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。

關於這項作業

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 150 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 121 頁的『使用範例程式瀏覽佇列』中說明的技術來檢視訊息。

相關參考

DISPLAY SBSTATUS

使用服務

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

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

服務物件可以是下列其中一種類型:

伺服器

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

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

指令

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

如果執行 MQSC 指令 **STOP SERVICE**，則在停止程式之前，不會執行任何檢查來判斷 MQSC 指令所啟動的程式 **START SERVICE** 是否仍在作用中。

相關參考

DISPLAY SVSTATUS
啟動服務
停止服務

定義服務物件

您可以使用 MQSC 指令 **DEFINE SERVICE** 來定義服務物件。

您需要定義的屬性如下:

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
指定服務程式的標準輸出 (stdout) 應該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 153 頁的『管理服務』
透過使用 **CONTROL** 參數，佇列管理程式可以自動啟動及停止服務物件的實例，或使用 MQSC 指令 **START SERVICE** 及 **STOP SERVICE** 來啟動及停止服務物件的實例。

相關參考
定義服務
DISPLAY SVSTATUS
啟動服務
停止服務

管理服務

透過使用 **CONTROL** 參數，佇列管理程式可以自動啟動及停止服務物件的實例，或使用 MQSC 指令 **START SERVICE** 及 **STOP SERVICE** 來啟動及停止服務物件的實例。

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 154 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。

停止服務
啟動服務

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。

您可以在其中新增環境變數的檔案

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
此檔案位於:

- Linux AIX AIX and Linux 系統上的 /var/mqm 。
- Windows 在 Windows 系統上安裝期間選取的資料目錄。

佇列管理程式範圍 service.env 檔
此檔案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:

- – Linux AIX 在 AIX and Linux 系統上: /var/mqm/qmgrs/QMNAME/service.env
  – Windows 在 Windows 系統上: C:\ProgramData\ IBM \ MQ\qmgrs\QMNAME\service.env

這兩個檔案都會處理 (如果有的話)，且佇列管理程式範圍檔案中的定義優先於機器範圍檔案中的那些定義。

可在 service.env 中指定的環境變數。

任何環境變數都可以在 service.env 中指定。 例如，如果 IBM MQ 服務執行一些指令，則在 service.env 檔案中設定 PATH 使用者變數可能很有用。 您設定變數的值不能是環境變數; 例如 CLASSPATH= *%CLASSPATH%* 不正確。 同樣地，在 Linux PATH= *$PATH* :/opt/mqm/bin 上，會產生非預期的結果。

CLASSPATH 必須大寫，且類別路徑陳述式只能包含文字。 部分服務 (例如遙測) 會設定自己的類別路徑。 會將定義在 service.env 中的 CLASSPATH 新增至該類別路徑。

檔案中定義的變數格式 service.env 是名稱/值變數配對的清單。 每一個變數都必須在新行上定義，且會在明確定義時採用每一個變數，包括空格。

service.env 範例

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                                *#
#* Licensed Materials - Property of IBM                             *#
#*                                                                  *#
#* 63H9336                                                          *#
#* (C) Copyright IBM Corporation 2005, 2025.                        *#
#*                                                                  *#
#* <NOC_COPYRIGHT>                                                  *#
#*                                                                  *#
#********************************************************************#
#********************************************************************#
#* Module Name: service.env                                         *#
#* Type       : IBM MQ service environment file                     *#
#* Function   : Define additional environment variables to be set   *#
#*              for SERVICE programs.                               *#
#* Usage      : <VARIABLE>=<VALUE>                                  *#
#*                                                                  *#
#********************************************************************#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE
```

相關參考

第 155 頁的『服務定義上的可取代插入項目』

在服務物件的定義中，可以替換記號。 執行服務程式時，會自動將替代的記號取代為其擴充文字。 替代記號可以從下列一般記號清單中取得，或從檔案 service.env 中定義的任何變數取得。

服務定義上的可取代插入項目

在服務物件的定義中，可以替換記號。 執行服務程式時，會自動將替代的記號取代為其擴充文字。 替代記號可以從下列一般記號清單中取得，或從檔案 service.env 中定義的任何變數取得。

以下是可在服務物件定義中用來替代記號的一般記號:

MQ_INSTALL_PATH
IBM MQ 的安裝位置。

MQ_DATA_PATH
IBM MQ 資料目錄的位置:

- Linux AIX 在 AIX and Linux 系統上， IBM MQ 資料目錄位置是 /var/mqm/
- Windows 在 Windows 系統上， IBM MQ 資料目錄的位置是在安裝 IBM MQ 期間選取的資料目錄

QMNAME
現行佇列管理程式名稱。

MQ 服務 _ 名稱
服務的名稱。

MQ_SERVER_PID
此記號只能由 **STOPARG** 和 **STOPCMD** 引數使用。

對於伺服器服務物件，此記號會取代為 **STARTCMD** 及 **STARTARG** 引數所啟動之處理程序的處理程序 ID。 否則，此記號將取代為 0。

MQ 佇列管理程式資料路徑
佇列管理程式資料目錄的位置。

MQ 佇列管理程式資料名稱
佇列管理程式的轉換名稱。 如需名稱轉換的相關資訊，請參閱 瞭解 IBM MQ 檔名。

若要使用可更換的插入項目，請將 + 字元內的記號插入任何 **STARTCMD**、 **STARTARG**、 **STOPCMD**、 **STOPARG**、 **STDOUT** 或 **STDERR** 字串中。 如需此範例，請參閱 第 156 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 **DEFINE SERVICE** MQSC 指令定義伺服器服務物件:

```
DEFINE SERVICE(S1) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+bin/runmqtrm') +
STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
STOPCMD('+MQ_INSTALL_PATH+bin/amqsstop') +
STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

   其中:

   +MQ_INSTALL_PATH+ 是代表安裝目錄的記號。
   +QMNAME+ 是代表佇列管理程式名稱的記號。
   ACCOUNTS.INITIATION.QUEUE 是起始佇列。
   amqsstop 是隨附於 IBM MQ 的範例程式，它會要求佇列管理程式中斷處理程序 ID 的所有連線。
   amqsstop 會產生 PCF 指令，因此指令伺服器必須在執行中。
   +MQ_SERVER_PID+ 是代表傳遞至停止程式之處理程序 ID 的記號。

   如需一般記號的清單，請參閱 第 155 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 **START SERVICE** MQSC 指令立即啟動伺服器服務物件的實例:

```
START SERVICE(S1)
```

3. 使用 **DISPLAY SVSTATUS** 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 **ALTER SERVICE** MQSC 指令:

```
ALTER SERVICE(S1) +
STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

   註: 在重新啟動服務之前，執行中服務將不會挑選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 **STOP SERVICE** 和 **START SERVICE** 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，並取得在 第 156 頁的『4』中所做的變更。

   註: 只有在服務定義中指定 **STOPCMD** 引數時，才能使用 MQSC 指令 **STOP SERVICE**。

相關參考

ALTER SERVICE
定義服務
DISPLAY SVSTATUS

啟動服務
停止服務

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 **DEFINE SERVICE** 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 AIX 或 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。

相關參考
定義服務

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 **DEFINE SERVICE** MQSC 指令定義指令服務物件:

```
DEFINE SERVICE(S3) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

其中:

logger 是隨 IBM MQ 提供的範例程式，可將項目寫入作業系統的系統日誌。
+QMNAME+ 是代表佇列管理程式名稱的記號。

相關參考
定義服務

傳遞引數的相關資訊

此範例顯示如何定義伺服器服務物件，以在啟動佇列管理程式時啟動稱為 runserv 的程式。

此範例以 Windows 樣式路徑分隔字元撰寫。

要傳遞至起始程式的其中一個引數是包含空格的字串。 此引數需要以單一字串傳遞。 為了達到此目的，會使用雙引號來定義指令服務物件，如下列指令所示:

1. 已使用 **DEFINE SERVICE** 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。

相關參考
定義服務

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. 已使用 **DEFINE SERVICE** 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。

相關參考
定義服務

管理用於觸發的物件

IBM MQ 可讓您在符合佇列上的特定條件時自動啟動應用程式。 例如，當佇列上的訊息數達到指定數目時，您可能想要啟動應用程式。 此機能稱為 觸發。 您必須定義支援觸發的物件。

在 使用觸發程式來啟動 IBM MQ 應用程式中詳細說明的觸發作業。

定義用於觸發的應用程式佇列

應用程式佇列是應用程式透過 MQI 進行傳訊所使用的本端佇列。 觸發需要在應用程式佇列上定義一些佇列屬性。

觸發本身由 **Trigger** 屬性 (MQSC 指令中的 TRIGGER) 啟用。 在此範例中，當本端佇列 MOTOR.INSURANCE.QUEUE，如下所示:

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
MAXMSGL (2000) +
DEFPSIST (YES) +
INITQ (MOTOR.INS.INIT.QUEUE) +
TRIGGER +
TRIGTYPE (DEPTH) +
TRIGDPTH (100)+
TRIGMPRI (5)
```

其中:

QLOCAL (MOTOR.INSURANCE.QUEUE)
是所定義應用程式佇列的名稱。

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
是程序定義的名稱，定義要由觸發監視器程式啟動的應用程式。

MAXMSGL (2000)
是佇列上訊息的長度上限。

DEFPSIST (YES)
指定依預設持續保存此佇列上的訊息。

INITQ (MOTOR.INS.INIT.QUEUE)
是佇列管理程式要放置觸發訊息的起始佇列名稱。

TRIGGER
是觸發程式屬性值。

TRIGTYPE (DEPTH)
指定當必要優先順序 (TRIGMPRI) 的訊息數達到 TRIGDPTH 中指定的數目時，產生觸發事件。

TRIGDPTH (100)
是產生觸發事件所需的訊息數目。

TRIGMPRI (5)
是佇列管理程式在決定是否產生觸發事件時要計算的訊息優先順序。 只會計算優先順序為 5 或以上的訊息。

定義起始佇列

當觸發事件發生時，佇列管理程式會將觸發訊息放置在應用程式佇列定義中指定的起始佇列上。 起始佇列沒有特殊設定，但您可以使用下列本端佇列 MOTOR.INS.INIT.QUEUE 用於指引:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
GET (ENABLED) +
NOSHARE +
NOTRIGGER +
MAXMSGL (2000) +
MAXDEPTH (1000)
```

定義處理程序

使用 DEFINE PROCESS 指令來建立程序定義。 程序定義會定義應用程式，以用來處理來自應用程式佇列的訊息。 應用程式佇列定義會命名要使用的處理程序，因此會將應用程式佇列與要用來處理其訊息的應用程式相關聯。 這是透過應用程式佇列 MOTOR.INSURANCE.QUEUE。 下列 MQSC 指令會定義必要的處理程序 MOTOR.INSURANCE.QUOTE.PROCESS，在下列範例中識別:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
DESCR ('Insurance request message processing') +
APPLTYPE (UNIX) +
APPLICID ('/u/admin/test/IRMP01') +
USERDATA ('open, close, 235')
```

其中:

MOTOR.INSURANCE.QUOTE.PROCESS
是程序定義的名稱。

DESCR ('Insurance request message processing')
說明與此定義相關的應用程式。 當您使用 DISPLAY PROCESS 指令時，會顯示此文字。 這可協助您識別處理程序執行的動作。 如果您在字串中使用空格，則必須以單引號括住字串。

APPLTYPE (UNIX)
是要啟動的應用程式類型。

APPLICID ('/u/admin/test/IRMP01')
是應用程式執行檔的名稱，指定為完整檔名。 在 Windows 系統中，一般 APPLICID 值會是 `c:\appl\test\irmp01.exe`。

USERDATA ('open, close, 235')
是使用者定義資料，可供應用程式使用。

顯示程序定義的屬性

請使用 DISPLAY PROCESS 指令來檢查定義的結果。 例如:

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)


24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
DESCR ('Insurance request message processing')
APPLICID ('/u/admin/test/IRMP01')
```

```
USERDATA (open, close, 235)
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
APPLTYPE (UNIX)
```

您也可以使用 MQSC 指令 ALTER PROCESS 來變更現有的程序定義，並使用 DELETE PROCESS 指令來刪除程序定義。

在兩個系統之間使用 dmpmqmsg 公用程式

dmpmqmsg 公用程式 (早期稱為 *qload*) 可讓您將佇列內容或其訊息複製或移動至檔案。

概觀

您使用 **dmpmqmsg** 建立的檔案可以視需要儲存，並在稍後的某個時間點用來將訊息重新載入回佇列。

重要:

1. 此檔案具有公用程式可理解的特定格式。 不過，檔案是人類可讀的，因此您可以在編輯器中更新它，然後再重新載入它。 如果您編輯檔案，則不得變更其格式。
2. 從 IBM MQ 9.1 開始， **dmpmqmsg** 公用程式隨附於 AIX, Linux, and Windows 的執行時期檔案集，因此可在 IBM MQ 伺服器及用戶端中使用。 在 IBM MQ 9.1 之前，公用程式僅隨附於伺服器套件。

可能的用途如下:

- 將佇列上的訊息儲存至檔案。 可能用於保存，稍後重新載入回佇列。
- 重新載入含有您先前儲存至檔案之訊息的佇列。
- 從佇列中移除舊訊息。
- 從儲存位置「重播」測試訊息，甚至在必要時維護訊息之間的正確時間。

小心: SupportPac MO03 使用 **-1** 參數來指定本端或用戶端連結。 **-1** 已取代為 **-c** 參數。

-P 現在用於字碼頁資訊，而非 **-c**。

如需指令及可用參數的進一步資訊，請參閱 dmpmqmsg 。

在 Linux 上使用 dmpmqmsg 公用程式的範例，使用 Windows 機器

您具有 Linux 機器上的佇列管理程式，該機器在佇列 (*Q1*) 中具有您要移至相同佇列管理程式中另一個佇列 (*Q2*) 的訊息。 您想要從 Windows 機器起始 **dmpmqmsg** 公用程式。

佇列 (*Q1*) 有四則訊息已使用範例 **amqsput** (本端佇列管理程式) 或 **amqsputc** (遠端佇列管理程式) 應用程式來新增。

在 Linux 機器上，您會看到:

```
display ql(Q1) CURDEPTH
        2 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(4)
```

設定 MQSERVER 環境變數以指向 Linux 中的佇列管理程式。 例如:

```
set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/veracruz.x.com(1414)
```

其中 *veracruz* 是機器的名稱。

執行 **dmpmqmsg** 公用程式以讀取佇列 *Q1*，並將輸出儲存在 `c:\temp\mqqload.txt` 中。

以遠端用戶端身分連接至在 Linux 主機及 MQSERVER 所建立埠中執行的佇列管理程式 *QM_VER*。 您可以使用下列屬性，作為遠端用戶端來達成連線: `-c`。

```
dmpmqmsg -m QM_VER -i Q1 -f c:\temp\mqqload.txt -c
Read      - Files:    0   Messages:       4   Bytes:           22
Written - Files:    1   Messages:       4   Bytes:           22
```

輸出檔 `c:\temp\mqqload.txt` 包含文字，使用 **dmpmqmsg** 公用程式瞭解的格式。

在 Windows 機器上，發出 **dmpmqmsg** 指令 (使用 `-o` 選項而非 `-i` 選項)，以從 Windows 機器上的檔案載入 Linux 機器上的佇列 (*Q2*):

```
dmpmqmsg -m QM_VER -o Q2 -f c:\temp\mqqload.txt -c
Read      - Files:    1   Messages:       4   Bytes:           22
Written - Files:    0   Messages:       4   Bytes:           22
```

在 Linux 機器上，請注意，現在佇列中有四則訊息已從檔案還原。

```
display ql(Q2) CURDEPTH
        6 : display ql(Q2) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q2)
TYPE(QLOCAL)
     CURDEPTH(4)
```

在 Linux 機器上，

從原始佇列中刪除訊息。

```
clear qlocal(Q1)
        4 : clear qlocal(Q1)
AMQ8022: IBM MQ queue cleared.
```

確認原始佇列上沒有其他訊息:

```
display ql(Q1) CURDEPTH
        5 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(0)
```

如需指令及其參數的說明，請參閱 dmpmqmsg。

相關概念

第 161 頁的『使用 dmpmqmsg 公用程式的範例』

您可以使用 **dmpmqmsg** 公用程式 (早期稱為 **qload**) 的簡單方式。 此公用程式會從 IBM MQ 8.0 納入產品中。

使用 dmpmqmsg 公用程式的範例

您可以使用 **dmpmqmsg** 公用程式 (早期稱為 **qload**) 的簡單方式。 此公用程式會從 IBM MQ 8.0 納入產品中。

先前 **qload** 公用程式是以 SupportPac MO03 提供。

將佇列卸載至檔案

在指令行上使用下列選項，將佇列上的訊息儲存至檔案:

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile
```

此指令會從佇列取得訊息副本，並將它們儲存在指定的檔案中。

將佇列卸載至一系列檔案

您可以在檔名中使用 `insert` 字元，將佇列卸載至一系列檔案。 在此模式中，每一則訊息都會寫入新檔案:

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile%n
```

此指令會將佇列卸載至檔案、 `myfile1`、 `myfile2`、 `myfile3` 等。

從檔案載入佇列

若要使用您儲存在 第 161 頁的『將佇列卸載至檔案』中的訊息重新載入佇列，請在指令行上使用下列選項:

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

此指令會將佇列卸載至檔案、 myfile1、 myfile2、 myfile3 等。

從一系列檔案載入佇列

您可以在檔名中使用 insert 字元，從一系列檔案載入佇列。 在此模式中，每一則訊息都會寫入新檔案:

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

此指令會將佇列載入至檔案、 myfile1、 myfile2、 myfile3 等。

將訊息從一個佇列複製到另一個佇列

將 第 161 頁的『將佇列卸載至檔案』中的 file 參數取代為另一個佇列名稱，並使用下列選項:

```
dmpmqmsg -m QM1 -i Q1 -o Q2
```

此指令容許將來自一個佇列的訊息複製到另一個佇列。

將前 100 則訊息從一個佇列複製到另一個佇列

使用前一個範例中的指令，並新增 -r#100 選項:

```
dmpmqmsg -m QM1 -i Q1 -o Q2 -r#100
```

將訊息從一個佇列移至另一個佇列

第 162 頁的『從檔案載入佇列』上的變異。 請注意使用僅瀏覽佇列的 **-i** (小寫) 與破壞性從佇列取得的 **-I** (大寫) 之間的區別:

```
dmpmqmsg -m QM1 -I Q1 -o Q2
```

將超過一天的訊息從一個佇列移至另一個佇列

此範例顯示使用年齡選擇。 可以選取早於、早於或在經歷時間範圍內的訊息。

```
dmpmqmsg -m QM1 -I Q1 -o Q2 -T1440
```

顯示目前在佇列上的訊息經歷時間

在指令行上使用下列選項:

```
dmpmqmsg -m QM1 -i Q1 -f stdout -dT
```

使用訊息檔案

從佇列卸載訊息之後 (如 第 161 頁的『將佇列卸載至檔案』)，您可能想要編輯檔案。

您也可能想要變更檔案的格式，以使用您在卸載佇列時未指定的其中一個顯示選項。

即使在卸載佇列之後，您也可以使用 **dmpmqmsg** 公用程式，將檔案重新處理成所需的格式。 在指令行上使用下列選項。

```
dmpmqmsg -f c:\oldfile -f c:\newfile -dA
```

如需指令及其參數的說明，請參閱 dmpmqmsg。

使用遠端 IBM MQ 物件

您可以使用 MQSC 指令、PCF 指令或 administrative REST API，來管理遠端佇列管理程式上的 IBM MQ 物件。 您必須先定義本端佇列管理程式與遠端佇列管理程式之間的傳輸佇列及通道，以便指令可以傳送至遠端佇列管理程式，以及本端佇列管理程式所接收的回應，然後才能使用上述任何方法。 或者，您可以配置佇列管理程式叢集，然後使用相同的遠端管理方法。

關於這項作業

若要準備佇列管理程式以進行遠端管理，您必須在本端佇列管理程式上配置下列物件:

- 接聽器。
- 具有遠端佇列管理程式名稱的傳輸佇列。
- 具有遠端佇列管理程式的連線詳細資料的傳送端通道。
- 與遠端佇列管理程式上的傳送端通道同名的接收端通道。

您也必須在遠端佇列管理程式上配置下列物件:

- 接聽器。
- 具有本端佇列管理程式名稱的傳輸佇列。
- 具有本端佇列管理程式連線詳細資料的傳送端通道。
- 與本端佇列管理程式上傳送端通道同名的接收端通道。

如需配置這些物件的相關資訊，請參閱 第 163 頁的『配置佇列管理程式以進行遠端管理』。

或者，您可以配置佇列管理程式叢集。 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。

如需配置叢集的相關資訊，請參閱 配置佇列管理程式叢集。

程序

- 如需如何管理遠端 IBM MQ 物件的相關資訊，請參閱下列子主題:
  - 第 163 頁的『配置佇列管理程式以進行遠端管理』
  - 第 167 頁的『管理用於遠端管理的指令伺服器』
  - 第 168 頁的『在遠端佇列管理程式上發出 MQSC 指令』
  - 第 169 頁的『編碼字集之間的資料轉換』

配置佇列管理程式以進行遠端管理

您可以使用 administrative REST API、MQSC 或 PCF 指令，從本端佇列管理程式管理遠端佇列管理程式。 遠端佇列管理程式可能位於相同系統、不同安裝架構中，或具有相同環境或不同 IBM MQ 環境的不同系統上。 您必須先在每一個佇列管理程式上建立傳送端和接收端通道、接聽器及傳輸佇列，才能從本端佇列管理程式遠端管理佇列管理程式。 這些通道及佇列可讓指令傳送至遠端佇列管理程式，以及在本端佇列管理程式上接收回應。 不論您要使用 administrative REST API、MQSC 或 PCF 指令，建立這些佇列及通道的程序都相同。

開始之前

- 下列程序使用範例佇列管理程式 `source.queue.manager` 及 `target.queue.manager`。 您必須在系統上建立並啟動這些佇列管理程式，才能遵循這些步驟，或在相關步驟中替換您自己的佇列管理程式名稱。

- 下列程序使用 TCP/IP 作為傳輸類型。 您必須知道這兩個系統的 IP 位址，才能完成這項作業。
- 下列程序會建立使用本端系統上網路埠 1818 及遠端系統上 1819 的接聽器。 您可以使用其他埠，但必須在適當的步驟中替換您的埠值。
- 您必須在本端或透過 Telnet 之類的網路機能來執行程序中的指令。

關於這項作業

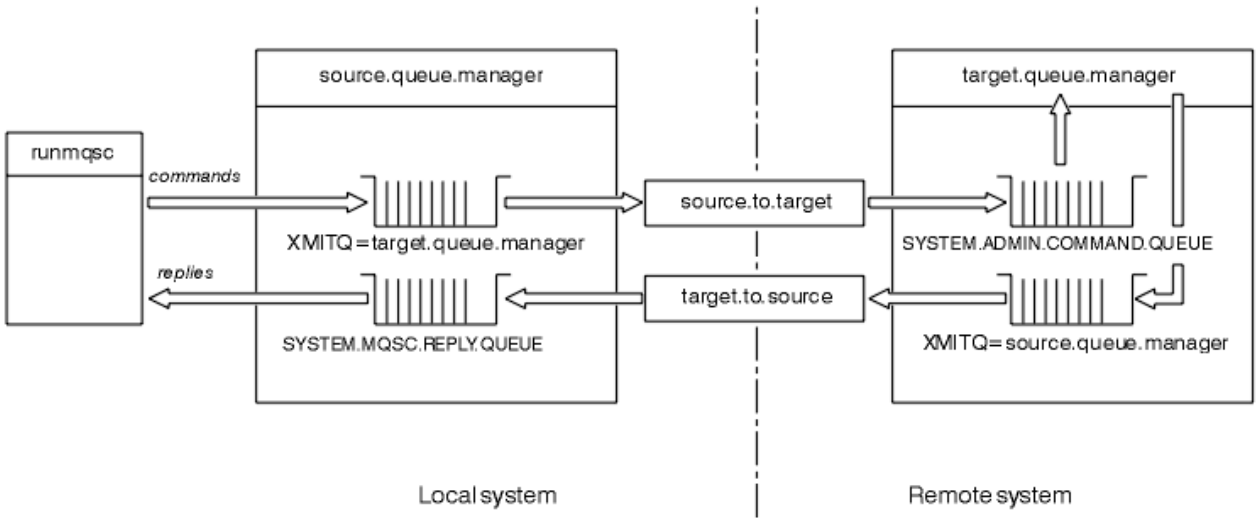


圖 *15:* 設定遠端管理的通道及佇列

第 164 頁的圖 15 顯示遠端管理所需的佇列管理程式、佇列及通道的配置:

- 物件 `source.queue.manager` 是來源佇列管理程式，您可以從中發出 administrative REST API、MQSC 或 PCF 指令，並將這些指令的結果傳回至其中。
- 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。
- 指令會放入與遠端佇列管理程式同名的傳輸佇列。 在此情況下，為 `target.queue.manager`。 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。
- 指令會由 `source.to.target` 通道傳送至遠端佇列管理程式上的 `SYSTEM.ADMIN.COMMAND.QUEUE`。 通道的每一端都有個別的定義。 一端是傳送端，另一端是接收端。 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。
- 指令輸出會放置在與從中傳送指令之本端佇列管理程式同名的遠端傳輸佇列上。 在此情況下，為 `source.queue.manager`。
- 輸出由 `target.to.source` 通道傳送至適當的回覆佇列，並由原始指令取得及輸出。

程序

1. 在遠端系統佇列管理程式上，確定指令佇列 `SYSTEM.ADMIN.COMMAND.QUEUE` 存在。 依預設，當建立佇列管理程式時，會建立此佇列。
2. 在遠端系統上，檢查指令伺服器是否正在佇列管理程式上執行。 如果指令伺服器不在執行中，則無法進行遠端管理:
   a) 針對佇列管理程式啟動 **runmqsc** 。 例如，對於佇列管理程式 `target.queue.manager`，請輸入下列指令:

   ```
   runmqsc target.queue.manager
   ```

   b) 輸入下列指令，以顯示指令伺服器的狀態:

   ```
   DISPLAY QMSTATUS CMDSERV
   ```

   c) 輸入下列指令，以結束 **runmqsc** :

```
end
```

d) 如果指令伺服器未啟動，請啟動它。 例如，對於佇列管理程式 target.queue.manager，請輸入下列指令:

```
strmqcsv target.queue.manager
```

3. 定義本端佇列管理程式上的通道、接聽器及傳輸佇列:

a) 針對佇列管理程式啟動 **runmqsc** 。 例如，對於佇列管理程式 source.queue.manager，請輸入下列指令:

```
runmqsc source.queue.manager
```

b) 定義傳送端通道。 這個傳送端通道必須與遠端佇列管理程式上的接收端通道同名。 例如，輸入下列 MQSC 指令，並將 **CONNAME** 的值取代為遠端佇列管理程式的 IP 位址及接聽器的埠號:

```
DEFINE CHANNEL ('source.to.target') +
CHLTYPE(SDR) +
CONNAME (localhost:1819) +
XMITQ ('target.queue.manager') +
TRPTYPE(TCP)
```

c) 定義接收端通道。 此接收端通道必須與遠端佇列管理程式上的傳送端通道同名。 例如，輸入下列指令:

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(RCVR) +
TRPTYPE(TCP)
```

d) 在本端佇列管理程式上定義接聽器。 例如，輸入下列指令:

```
DEFINE LISTENER ('source.queue.manager') +
TRPTYPE (TCP) +
PORT (1818)
```

e) 在本端佇列管理程式上定義傳輸佇列。 此傳輸佇列必須與遠端佇列管理程式同名。 例如，輸入下列指令:

```
DEFINE QLOCAL ('target.queue.manager') +
USAGE (XMITQ)
```

f) 啟動接聽器。 例如，輸入下列指令:

```
START LISTENER ('source.queue.manager')
```

g) 輸入下列指令，以結束 **runmqsc** :

```
end
```

4. 定義遠端佇列管理程式上的通道、接聽器及傳輸佇列:

a) 針對佇列管理程式啟動 **runmqsc** 。 例如，對於佇列管理程式 target.queue.manager，請輸入下列指令:

```
runmqsc target.queue.manager
```

b) 定義傳送端通道。 這個傳送端通道必須與本端佇列管理程式上的接收端通道同名。 例如，輸入下列 MQSC 指令，並將 **CONNAME** 的值取代為本端佇列管理程式的 IP 位址及接聽器的埠號:

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(SDR) +
CONNAME (localhost:1818) +
XMITQ ('source.queue.manager') +
TRPTYPE(TCP)
```

c) 定義接收端通道。 此接收端通道必須與本端佇列管理程式上的傳送端通道同名。例如，輸入下列指令:

```
DEFINE CHANNEL ('source.to.target') +
CHLTYPE(RCVR) +
TRPTYPE(TCP)
```

d) 定義接聽器。 例如，輸入下列指令：

```
DEFINE LISTENER ('target.queue.manager') +
TRPTYPE (TCP) +
PORT (1819)
```

e) 定義傳輸佇列。 此傳輸佇列必須與本端佇列管理程式同名。 例如，輸入下列指令：

```
DEFINE QLOCAL ('source.queue.manager') +
USAGE (XMITQ)
```

f) 啟動接聽器。 例如，輸入下列指令：

```
START LISTENER ('target.queue.manager')
```

g) 輸入下列指令，以結束 **runmqsc**：

```
end
```

5. 啟動本端系統上的傳送端通道：
   a) 針對佇列管理程式啟動 **runmqsc**。 例如，對於佇列管理程式 source.queue.manager，請輸入下列指令：

```
runmqsc source.queue.manager
```

   b) 啟動傳送端通道。 例如，輸入下列指令：

```
START CHANNEL ('source.to.target')
```

   c) 輸入下列指令，以結束 **runmqsc**：

```
end
```

6. 在遠端系統上啟動傳送端通道：
   a) 啟動佇列管理程式的 runmqsc。 例如，對於佇列管理程式 target.queue.manager，請輸入下列指令：

```
runmqsc target.queue.manager
```

   b) 啟動傳送端通道。 例如，輸入下列指令：

```
START CHANNEL ('target.to.source')
```

   c) 輸入下列指令，以結束 **runmqsc**：

```
end
```

7. 將 MQSC 指令從本端系統傳送至遠端佇列管理程式，以測試配置是否已順利完成：
   a) 從本端系統啟動遠端佇列管理程式的 **runmqsc**。 例如，輸入下列指令：

```
runmqsc -w 30 -m source.queue.manager target.queue.manager
```

   b) 輸入下列指令，以顯示遠端佇列管理程式上的佇列：

```
DISPLAY QUEUE (*)
```

   成功時，會顯示遠端佇列管理程式中的佇列清單。
   c) 如果這些步驟無法運作，請檢查兩個系統上的通道是否都處於執行中狀態。 如果通道不在執行中且未啟動，請檢查通道及傳輸佇列是否配置正確，以及指令伺服器是否在執行中。 例如，檢查是否為傳送

端通道指定正確的 CONNAME，以及傳輸佇列是否具有正確的名稱。 此外，請檢查佇列管理程式日誌，以取得可能有助於解決問題的安全異常狀況。

結果

您的佇列管理程式已配置為從本端系統遠端管理遠端佇列管理程式。

下一步

- 進一步瞭解使用 MQSC 指令進行遠端管理: 第 168 頁的『在遠端佇列管理程式上發出 MQSC 指令』
- 進一步瞭解使用 PCF 指令撰寫管理程式: 第 22 頁的『使用 IBM MQ 可程式指令格式』。
- 進一步瞭解如何使用 administrative REST API 進行遠端管理: 第 70 頁的『使用 REST API 進行遠端管理』。

管理用於遠端管理的指令伺服器

每一個佇列管理程式都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，並傳回完成碼或操作員訊息。 您可以啟動、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，指令伺服器是必要的。

開始之前

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。 如果指令伺服器自動啟動，則無法使用 strmqcsv 或 endmqcsv 指令來啟動及停止指令伺服器。 您可以使用 MQSC 指令 **ALTER QMGR** 來變更 **SCMDSERV** 屬性的值。 依預設，指令伺服器會自動啟動。

停止佇列管理程式也會結束與其相關聯的指令伺服器。

程序

- 顯示指令伺服器的狀態:
  a) 輸入下列指令，以針對適當的佇列管理程式啟動 **runmqsc**：

  ```
  runmqsc target.queue.manager
  ```

  其中 `target.queue.manager` 是要顯示指令伺服器的佇列管理程式。
  b) 輸入下列 MQSC 指令，以顯示指令伺服器狀態:

  ```
  DISPLAY QMSTATUS CMDSERV
  ```

  c) 輸入下列指令，以結束 **runmqsc**：

  ```
  end
  ```

- 如果指令伺服器未設定為自動啟動，請輸入下列指令來啟動指令伺服器:

  ```
  strmqcsv target.queue.manager
  ```

  其中 `target.queue.manager` 是正在啟動指令伺服器的佇列管理程式。
- 如果指令伺服器未設定為自動啟動，請輸入下列指令來停止指令伺服器:

  ```
  endmqcsv target.queue.manager
  ```

  其中 `target.queue.manager` 是正在停止指令伺服器的佇列管理程式。

  依預設，指令伺服器會以受控制的方式停止。 您可以將 `-i` 旗標新增至指令，以立即停止指令伺服器。

在遠端佇列管理程式上發出 MQSC 指令

在配置佇列管理程式進行遠端管理之後，您可以在本端系統上使用特定形式的 **runmqsc** 指令，在遠端佇列管理程式上執行 MQSC 指令。 每一個指令都會以 Escape PCF 傳送至指令佇列 SYSTEM.ADMIN.COMMAND.QUEUE，屬於遠端佇列管理程式。 會在 SYSTEM.MQSC.REPLY.QUEUE 佇列。

開始之前

您必須先完成 第 163 頁的『配置佇列管理程式以進行遠端管理』 中的步驟，以配置通道、傳輸佇列、接聽器及指令伺服器，然後才能使用 MQSC 指令從遠端管理佇列管理程式。

程序

1. 請確定指令伺服器正在遠端佇列管理程式上執行。

   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 167 頁的『管理用於遠端管理的指令伺服器』。
2. 在來源佇列管理程式上，您可以使用下列兩種方式之一來執行 MQSC 指令:

   - 以互動方式，使用下列指令來啟動 **runmqsc** :
     - z/OS 如果遠端佇列管理程式位於 z/OS 上，請輸入下列指令:

       ```
       runmqsc -w 30 -x -m source.queue.manager target.queue.manager
       ```

     - Multi 如果遠端佇列管理程式不在 z/OS 上，請輸入下列指令:

       ```
       runmqsc -w 30 -m source.queue.manager target.queue.manager
       ```

   - 從指令檔:

     a. 將要在遠端系統上執行的 MQSC 指令放置在文字檔中，每行一個指令。

     b. 使用 **runmqsc** 命令上的-v 標誌驗證本機佇列管理器上的 MQSC 命令。 -v 標誌檢查命令是否有效，但不運行它們。 請注意，如果部分指令適用於遠端佇列管理程式，但不適用於本端佇列管理程式，則它們可能會失敗:

     ```
     runmqsc -v source.queue.manager < myCmdFile.in > results.out
     ```

     myCmdFile.in 包含要檢查的 MQSC 指令，而 results.out 檔案包含指令的驗證結果。

     c. 透過輸入下列其中一個指令，在遠端佇列管理程式上執行指令檔:

     - z/OS 如果遠端佇列管理程式位於 z/OS 上，請輸入下列指令:

       ```
       runmqsc -w 30 -x -m source.queue.manager target.queue.manager < myCmdFile.in >
       results.out
       ```

     - Multi 如果遠端佇列管理程式不在 z/OS 上，請輸入下列指令:

       ```
       runmqsc -w 30 -m source.queue.manager target.queue.manager < myCmdFile.in >
       results.out
       ```

   使用的參數如下:

   -w 秒

   指定以間接模式執行 MQSC 指令，其中指令會放入指令伺服器輸入佇列並依序執行。

   變數 *seconds* 指定等待遠端佇列管理程式回應的時間長度 (以秒為單位)。 在此時間之後收到的任何回覆都會捨棄，但 MQSC 指令仍會在遠端佇列管理程式上執行。 當指令逾時時，會在本端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。

-x
指定遠端佇列管理程式是 z/OS 佇列管理程式。

-m *localQMgrName*
指定您要用來向遠端佇列管理程式提交指令的本端佇列管理程式名稱

下一步

如果您在遠端執行 MQSC 指令時遇到困難:

- 請檢查遠端佇列管理程式是否在執行中。
- 請檢查指令伺服器是否在遠端系統上執行。
- 請檢查通道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。
- 請確定從本端佇列管理程式傳送的要求對目標佇列管理程式有意義。 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 解決 MQSC 指令的問題。

編碼字集之間的資料轉換

佇列管理程式可以將 IBM MQ 定義格式 (也稱為內建格式) 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，支援在具有 ID (CCSID) 850 和 500 的編碼字集之間進行轉換，因為兩者都適用於西歐語言。

如需將 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 資料轉換處理中。

V 9.2.0 從 IBM MQ 9.2.0 開始， IBM MQ Appliance、 Windows、 Linux 和 macOS 支援在 CCSID 37 和 500 之間轉換。

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。

對於套用 `ccsid_part2.tbl` 的平台，如需進一步資訊，請參閱 第 172 頁的『指定預設資料轉換』 使用 `ccsid_part2.tbl` 。 第 170 頁的『預設資料轉換』中說明除套用 `ccsid_part2.tbl` 檔案的平台以外的其他平台上的預設資料轉換。

加強 Unicode 資料轉換支援

在 IBM MQ 9.0 之前，舊版產品不支援轉換包含超出「基本多語言平面」(U+FFFF 以上的字碼點) 之 Unicode 字碼點的資料。 Unicode 資料轉換支援僅限於以 Unicode 3.0 標準定義的字碼點，以 UTF-8 或 UCS-2(UTF-16 的 2 位元組固定寬度子集) 編碼。

從 IBM MQ 9.0 開始，產品支援在資料轉換中以 Unicode 8.0 標準定義的所有 Unicode 字元。 這包括 UTF-16 的完整支援，包括代理配對 (一對 2 位元組 UTF-16 字元，在 `X'D800'` 到 `X'DFFF'` 的範圍內，代表 U+FFFF 上方的 Unicode 字碼點)。

當一個 CCSID 中的預先編製字元對映至另一個 CCSID 中的結合字元序列時，也支援結合字元序列。

在部分平台上，已延伸與 Unicode 及 CCSID 1388、1390、1399、4933、5488 及 16884 之間的資料轉換，以支援目前定義給這些 CCSID 的所有字碼點，包括對映至 Unicode 增補平面中的字碼點的那些字碼點。

在 CCSID 1390、1399 和 16884 的情況下，這包括 JIS X 0213 (JIS2004) 標準中定義的字元。

還新增了與 Unicode 及六個新 CCSID 之間的轉換支援 (1374 到 1379)。

ccsid_part2.tbl 檔案

從 IBM MQ 9.0 提供其他檔案 `ccsid_part2.tbl`。

`ccsid_part2.tbl` 檔案優先於 `ccsid.tbl` 檔案，並且:

- 容許您新增或修改 CCSID 項目
- 指定預設資料轉換
- 指定不同指令層次的資料

`ccsid_part2.tbl` 僅適用於下列平台:

- Linux - 所有版本
- Windows

Windows 在 IBM MQ for Windows 上，依預設 `ccsid_part2.tbl` 位於 *MQDataRoot*`\conv\table` 目錄中。 此外，在 IBM MQ for Windows 上，它會記錄所有支援的字碼集。

Linux 在 IBM MQ for Linux 上， `ccsid_part2.tbl` 位於目錄 *MQDataRoot*`/conv/table` 中，且支援的字碼集保留在 IBM MQ 所提供的轉換表中。

雖然 `ccsid_part2.tbl` 檔案會取代舊版 IBM MQ 中使用的現有 `ccsid.tbl` 檔案，以提供其他 CCSID 資訊，但 `ccsid.tbl` 檔案會繼續由 IBM MQ 剖析，因此不得刪除。

如需相關資訊，請參閱 第 171 頁的『ccsid_part2.tbl 檔案』。

ccsid.tbl 檔案

在非 `ccsid_part2.tbl` 適用的平台上，檔案 `ccsid.tbl` 用於下列用途:

- AIX 在 AIX 上，作業系統會在內部保留支援的字碼集。
- 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 `ccsid.tbl` 中所記錄的資訊; 例如，如果未來版本的作業系統支援其他編碼字集，您可能想要這麼做。

預設資料轉換

從 IBM MQ 9.0 開始，下列平台上的預設資料轉換方法已變更:

- Linux - 所有版本
- Windows

如需進一步資訊，請參閱 第 172 頁的『指定預設資料轉換』 使用 `ccsid_part2.tbl` 。

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。

在非 `ccsid_part2.tbl` 適用的平台上，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:

- 如果不支援來源與目標 CCSID 之間的轉換，但來源與目標環境的 CCSID 都是 EBCDIC 或都是 ASCII ，則會將字元資料傳遞至目標應用程式而不進行轉換。

- 如果一個 CCSID 代表 ASCII 編碼字集，而另一個代表 EBCDIC 編碼字集，則 IBM MQ 會使用 ccsid.tbl 中定義的預設資料轉換 CCSID 來轉換資料。

註: 嘗試限制將字元轉換為在指定給訊息的編碼字集及預設編碼字集中具有相同編碼值的字元。 如果您只使用對 IBM MQ 物件名稱有效的字元集 (如 命名 IBM MQ 物件 中所定義) 你一般會滿足這個要求。 在日本使用的 EBCDIC CCSID 290、930、1279 及 5026 發生異常狀況，其中小寫字元具有與其他 EBCDIC CCSID 不同的代碼。

以使用者定義格式轉換訊息

佇列管理程式無法將使用者定義格式的訊息從一個編碼字集轉換成另一個編碼字集。 如果您需要以使用者定義格式轉換資料，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。

ccsid_part2.tbl 檔案

ccsid_part2.tbl 檔案用來提供其他 CCSID 資訊。 ccsid_part2.tbl 檔案會取代 IBM MQ 9.0 之前使用的 ccsid.tbl 檔案。

註: ccsid.tbl 檔案 (在 IBM MQ 9.0 之前用來提供其他 CCSID 資訊) 會繼續由 IBM MQ 剖析，且不應刪除。 不過， ccsid_part2.tbl 中的項目優先於 ccsid.tbl 中的其他項目。

您應該使用 ccsid_part2.tbl 而非 ccsid.tbl ，因為 ccsid_part2.tbl:

- 包含 Unicode 編碼值的支援。 從 IBM MQ 9.0 開始，產品支援資料轉換中以 Unicode 8.0 標準定義的所有 Unicode 字元，包括 UTF-16 的完整支援。 如需相關資訊，請參閱 第 169 頁的『編碼字集之間的資料轉換』。
- 可讓您指定 CCSID 項目的版本，以便項目僅適用於選取的指令層次。

您可以使用 ccsid_part2.tbl 檔案來執行下列動作:

- 新增或修改 CCSID 項目
- 指定預設資料轉換
- 指定不同指令層次的資料

ccsid_part2.tbl 檔案僅適用於下列平台:

- Linux - 所有版本
- Windows

ccsid_part2.tbl 檔案的位置視您的平台而定:

- 所有 Linux 版本上的 *MQDataRoot*/conv/table 目錄。
- Windows 上的 *MQDataRoot*\conv\table 目錄。

新增或修改 CCSID 項目

ccsid_part2.tbl 檔案中的項目具有下列格式:

```
<CCSID number> <Base CCSID> <DBCS CodePage> <SBCS CodePage>
<Type> <Encoding> <ACRI> <Name>
```

CCSID 1200 (UTF-16) 的範例項目如下:

```
1200 1200 1200 1200 3 8 0 UTF-16
```

註: 如需 ACRI 值的詳細資料，請參閱 `ccsid_part2.tbl` 檔案中的註解。

採用 `ccsid_part2.tbl` 格式:

類型可以相等:

1=SBCS

2=DBCS

3=MBCS

編碼可以相等:

1=EBCDIC

2 = ASCII

3 = ISO

4 = UCS-2

5 = UTF-8

6 = Euc

7 = GB18030

8 = UTF-16

9 = UTF-32

編輯檔案時，請執行下列動作:

- 可以使用行開頭的 # 符號來指定註解。 這可防止 IBM MQ 嘗試剖析行。
- 無法提供行內註解。
- 必須確定您未建立空白行。
- 不得在檔案結尾新增項目。

  應在 ACRI 表格資訊之前新增新的 CCSID 登錄。

指定預設資料轉換

如果兩個 CCSID 之間不支援轉換，您可以定義預設轉換 CCSID (用來在 ASCII 或類似與 EBCDIC CCSID 之間進行轉換)。

如果您啟用此功能，則預設轉換會用於傳輸及訊息標頭，也可以用於使用者資料轉換。

透過建立兩行類似下列內容來啟用預設轉換:

```
default       0      500     1     1    0
default       0      850     1     2    0
```

第一行將 EBCDIC CCSID 的預設值設為 500 ，第二行將 ASCII 及類似 CCSID 的預設值設為 850。

指定不同指令層次的資料

若要為 IBM MQ 的不同指令層次指定 CCSID 項目，請使用冒號符號，後面接著您希望下一區段適用的 IBM MQ 指令層次 (或指令層次)。

此數字代表佇列管理程式或用戶端必須在其中執行的指令層次下限。 例如，如果現行佇列管理程式是指令層次 900 ，且遇到 800 或 900 指令層次旗標，則會讀取 CCSID。

不過，層次 800 的佇列管理程式會忽略 900 區段中的任何 CCSID。

指定的指令層次適用於在指令層次旗標之後發現的所有 CCSID 項目，直到找到新的指令層次旗標為止。

如果您需要將指令層次設為所有指令層次，請指定數字零。

第一次剖析 `ccsid_part2.tbl` 時， IBM MQ 會將發現的所有 CCSID 視為適用於 IBM MQ 的所有指令層次。

只有在 IBM MQ 發現第一個指令層次旗標時，才會開始使用版本化。

下列程式碼 Snippet 顯示使用版本化的範例:

```
# Comment Block
# End of Comment Block
# Because no command level flag is specified and we're at the start of the file
# the following CCSIDs will be read on all versions
    819   819       0      819     1    3     0    ISO8859-1
    923   923       0      923     1    3     0    ISO8859-15
   1051  1051       0     1051     1    3     0    IBM-1051
# The colon :900 below shows that the CCSIDs after will only be for MQ cmd level 900 and above
:900
   8629   437       0      437     1    2     0    IBM-437
  12725   437       0      437     1    2     0    IBM-437
  16821   437       0      437     1    2     0    IBM-437
  20917   437       0      437     1    2     0    IBM-437
# The colon :0 below shows that the CCSIDs after will be for all version of MQ
:0
   4946   850       0      850     1    2     0    IBM-850
  33618   850       0      850     1    2     0    IBM-850
  61697   850       0      850     1    2     0    IBM-850
  61698   850       0      850     1    2     0    IBM-850
```

管理 Managed File Transfer

使用 Managed File Transfer 指令來管理 Managed File Transfer。 您也可以使用「IBM MQ Explorer」來執行部分管理作業。

將訊息放置在代理程式指令佇列中以啟動傳送

您也可以將檔案傳送訊息放置在來源代理程式的指令佇列上以啟動檔案傳送。
`SYSTEM.FTE.COMMAND.AGENT01` 是指令佇列名稱範例。 您必須確定訊息送達正確來源代理程式的指令佇列；如果代理程式收到的訊息與 XML 中的來源資訊不符，則會拒絕訊息。

The transfer request XML must conform to the `FileTransfer.xsd` schema and use the <request> element as the root element. 如需傳送要求訊息的結構及內容的相關資訊，請參閱檔案傳送要求訊息格式。 將傳送要求訊息放置在代理程式指令佇列上的作法視作業而定。 例如，您可以使用 IBM MQ Java API，以程式設計方式將訊息放置在佇列上。

啟動 MFT 代理程式

必須先啟動 Managed File Transfer 代理程式，才能將它用來進行檔案傳送。

關於這項作業

您可以從指令行啟動 Managed File Transfer Agent。 在此情況下，當您登出系統時，代理程式程序即會停止。

ALW 在 AIX, Linux, and Windows 上，您可以配置代理程式，讓它在您從系統登出時繼續執行並可繼續接收檔案傳送。

z/OS 在 z/OS 上，您可以將代理程式配置為從 JCL 以啟動作業的方式來啟動代理程式，而不需要互動式階段作業。

請注意，如果代理程式在執行時發生無法復原的錯誤，則會產生首次失敗資料擷取 (FDC) 並停止代理程式。

程序

- 若要從指令行啟動代理程式，請使用 **fteStartAgent** 指令。

如需相關資訊，請參閱 **fteStartAgent**。

- ALW
  若要配置代理程式以讓它在您從系統登出時繼續執行，請完成下列動作：
  - Windows 在 Windows 上，將代理程式配置為以 Windows 服務方式執行。 如需相關資訊，請參閱 第 174 頁的『以 Windows 服務方式啟動 MFT 代理程式』。
  - Linux AIX 在 AIX and Linux 上，使用 Script 檔將代理程式配置為在重新開機期間自動啟動。 如需相關資訊，請參閱 第 175 頁的『在 AIX and Linux 系統啟動時啟動 MFT 代理程式』。
- z/OS
  在 z/OS 上，將代理程式配置為從 JCL 以啟動作業的方式來啟動代理程式，而不需要互動式階段作業。 如需相關資訊，請參閱 第 176 頁的『在 z/OS 上啟動 MFT 代理程式』。

Windows 以 Windows 服務方式啟動 MFT 代理程式

您可以 Windows 服務方式啟動代理程式，則當您登出 Windows 時，代理程式會繼續執行且可以接收檔案傳送。

關於這項作業

在 Windows 上，當您從指令行啟動代理程式時，代理程式程序會使用您用來登入 Windows 的使用者名稱來執行。 當您登出系統時，代理程式程序即會停止。 若要防止代理程式停止，您可以配置代理程式以 Windows 服務方式執行。 以 Windows 服務方式執行，也可以讓您配置在 Windows 環境啟動或重新啟動時自動啟動代理程式。

請完成下列步驟，來啟動以 Windows 服務方式執行的代理程式。 您必須在其中一個支援的 Windows 版本上執行 Managed File Transfer，才能以 Windows 服務形式執行代理程式。 如需受支援環境的清單，請參閱 IBM MQ 的系統需求。

確切的步驟視您是否已建立代理程式或正在建立代理程式而定。 下列步驟說明這兩個選項。

程序

1. 如果您要建立 Managed File Transfer 代理程式，請使用 **fteCreateAgent**、 **fteCreateCDAgent** 或 **fteCreateBridgeAgent** 指令。 請指定 **-s** 參數，表示要以 Windows 服務方式執行代理程式。 在下列範例中，建立了代理程式 AGENT1，它具有代理程式佇列管理程式 QMGR1。 Windows 服務是以使用者名稱 `fteuser` 執行，它的關聯密碼是 `ftepassword`。

   ```
   fteCreateAgent -agentName AGENT1 -agentQMgr QMGR1 -s -su fteuser -sp ftepassword
   ```

   您可以在 **-s** 參數之後，選擇性地指定服務的名稱。 如果未指定名稱，則此服務會命名為 `mqmftAgent`*AGENTQMGR*，其中 *AGENT* 是您指定的代理程式名稱，而 *QMGR* 是代理程式佇列管理程式名稱。 在此範例中，服務的預設名稱是 `mqmftAgentAGENT1QMGR1`。

   註: 您使用 **-su** 參數指定的 Windows 使用者帳戶必須具有 **Log on as a service** 權限。 如需如何配置此項的相關資訊，請參閱 以 Windows 服務方式執行 MFT 代理程式或日誌程式的指引。

   有關更多信息，請參閱 fte 創建代理, fteCreateCDAgent:創建一個 Connect:Direct 橋樑代理， 或者 fteCreateBridgeAgent （建立並配置 MFT 協議橋代理）。
2. 如果您遵循前一個步驟來建立代理程式，請執行 **fteCreateAgent**、 **fteCreateCDAgent** 或 **fteCreateBridgeAgent** 指令所產生的 MQSC 指令。 這些指令會建立代理程式所需的 IBM MQ 佇列。

   例如，若為代理程式名為 *AGENT1*、代理程式佇列管理程式名為 *QMGR1* 及協調佇列管理程式名為 *COORDQMGR1*，請執行下列指令：

   ```
   runmqsc QMGR1 MQ_DATA_PATH\mqft\config\COORDQMGR1\agents\AGENT1\AGENT1_create.mqsc
   ```
3. 如果您未遵循先前的步驟建立代理程式，而是想要配置現有的代理程式以 Windows 服務方式執行，請先停止執行中的代理程式，然後修改其配置。

a) 下列範例使用代理程式名為 AGENT1。 請執行下列指令：

```
fteStopAgent AGENT1
```

b) 使用 **fteModifyAgent** 指令，配置代理程式以 Windows 服務方式執行：

```
fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword
```

有關更多信息，請參閱 fteModifyAgent:運行一個 MFT 代理人作為 Windows 服務。

4. 使用 **fteStartAgent** 指令來啟動代理程式。 或者，您也可以從 Windows 桌面開始功能表選取「控制台」中的「系統管理工具」，使用 Windows「服務」工具來啟動該服務。

```
fteStartAgent AGENT1
```

即使您登出 Windows，該服務也會繼續執行。 為了確保服務在關閉之後 Windows 重新啟動時也會重新啟動，依預設， Windows 服務工具中的 **啟動類型** 欄位會設為 **自動** 。 如果您不想要在 Windows 重新啟動時重新啟動服務，請將此變更為 **手動** 。

5. 選擇性的: 若要停止代理程式，請使用 fteStopAgent 指令，或使用 Windows「服務」工具。 例如，從指令行執行下列指令：

```
fteStopAgent AGENT1
```

- 當您以服務方式執行 **fteStopAgent** 指令時，不論您是否指定 **-i** 參數，該指令一律使用此參數執行。 **-i** 參數會立即停止代理程式，不會完成進行中的任何傳送。 這是因為 Windows 服務的限制所致。

下一步

如果您在啟動 Windows 服務時遇到問題，請參閱 以 Windows 服務方式執行 MFT 代理程式或日誌程式的指引。 這個主題也說明 Windows 服務日誌檔的位置。

Linux AIX 在 AIX and Linux 系統啟動時啟動 MFT 代理程式

Managed File Transfer Agent 可以配置成在 AIX and Linux 上系統啟動時啟動。 當您登出時，代理程式會繼續執行，並且可以接收檔案傳送。

當您使用下列其中一個 Managed File Transfer 指令 (**fteCreateAgent**、 **fteCreateCDAgent**、 或 **fteCreateBridgeAgent**) 建立並配置代理程式時，您可以使用只執行下列指令的 Script 檔，將它配置成在 AIX and Linux 機器上重新開機期間自動啟動:

```
su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
```

其中 *mq_install_root* 是必要 Managed File Transfer 安裝的根目錄，預設值為: /opt/mqm，而 *agent_name* 是要啟動的 Managed File Transfer Agent 的名稱。 此 Script 檔的使用會因特定作業系統而異。 例如， Linux 下還有其他可用的選項。

Linux

Linux

若是 Linux 系統，您有多種方式可以在系統開機程序期間啟動應用程式。 一般而言，請考慮執行下列步驟:

1. 建立一個稱為 /etc/rc.mqmft 的檔案，其內容如下:

```
#!/bin/sh
su -l mqmft_user"-c mq_install_root/bin/fteStartAgent agent_name"
```

其中 *mqmft_user* 是用來執行代理程式程序的使用者 ID。 此使用者 ID 必須是 mqm 群組的成員。

2. 使該檔案成為執行檔，例如:

```
chmod 755 /etc/rc.mqmft
```

3. 接下來，將下面這一行新增至 /etc/inittab:

```
mqmft:5:boot:/etc/rc.mqmft
```

在 Linux 上啟動期間啟動代理的其他方法包括將腳本行新增至/etc/rc.d/rc.local 文件，或在 Linux SuSe,上將腳本行新增至/etc/init.d/boot.local 文件。 您應該選取最適合您環境的方法。 以下是在支援的特定 Linux 發行套件啟動期間，啟動代理程式的其他方式的一些相關資訊:

SLES 10 及 11

若是 SUSE Linux Enterprise Server (SLES) 10 及 11 系統，請遵循下列步驟:

1. 以系統 root 使用者 ID 建立您專屬的 /etc/init.d/rc.rclocal 檔案。
2. 將下列這幾行新增至 rc.rclocal 檔案:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: rc.rclocal
# Required-Start: $network $syslog
# Required-Stop: $network $syslog
# Default-Stop: 0 1 2 6
# Description: MQMFT agent startup
### END INIT INFO
su -l mqmft_user"-c mq_install_root/bin/fteStartAgent agent_name"
```

3. 執行下列指令:

```
chmod 755 rc.rclocal

chkconfig --add rc.rclocal
```

在 Linux 上使用 systemd 啟動 Managed File Transfer 代理程式

Linux

執行下列程序:

1. 在 /etc/systemd/ 系統資料夾中建立檔案並將其命名，例如 <agentname>.service。 新增下列內容，其中 <agentname> 是 MFT_AGT_LNX_0。

```
# vi /etc/systemd/system/MFT_AGT_LNX_0.service
[Unit]
Description=IBM MQ MFT MFT_AGT_LNX_0
[Service]
ExecStart=/opt/mqm/bin/fteStartAgent MFT_AGT_LNX_0
ExecStop=/opt/mqm/bin/fteStopAgent MFT_AGT_LNX_0
Type=forking
User=mqm
Group=mqm
KillMode=none
```

2. 若要啟用服務，請執行下列指令:

```
# systemctl enable MFT_AGT_LNX_0
# systemctl daemon-reload
```

3. 若要啟動代理程式並檢查其狀態，請執行下列指令:

```
# systemctl start MFT_AGT_LNX_0
# systemctl status MFT_AGT_LNX_0
```

z/OS 在 z/OS 上啟動 MFT 代理程式

在 z/OS 上，除了從 z/OS UNIX System Services 階段作業執行 **fteStartAgent** 指令之外，您還可以從 JCL 以啟動作業形式啟動代理程式，而不需要互動式階段作業。

使用啟動的作業是因為它是以特定的使用者 ID 執行，且不受使用者登出影響。

註: 通常，據以執行啟動作業的管理使用者可能沒有登入專用權，並因此無法以執行代理程式的使用者身分登入 z/OS 系統。 無法對該代理程式發出 **fteStartAgent**、**fteStopAgent** 和 **fteSetAgentTraceLevel** 指令以及已指定 **-d** 參數的 **fteShowAgentDetails** 指令。

從 IBM MQ 9.0.2 及 IBM MQ 9.0.0 Fix Pack 1 中，您可以將代理程式內容 **adminGroup** 與 z/OS 上的 Managed File Transfer 代理程式搭配使用。 您可以定義安全管理員群組（例如 MFTADMIN），然後新增已啟動作業使用者 ID 和管理者 TSO ID 至此群組。 編輯代理程式內容檔，並將 **adminGroup** 內容設為此安全管理員群組的名稱。

```
adminGroup=MFTADMIN
```

此群組的成員隨後可以對正在當作已啟動作業執行的代理程式發出 **fteStartAgent**、**fteStopAgent** 和 **fteSetAgentTraceLevel** 以及已指定 **-d** 參數的 **fteShowAgentDetails** 指令。

如需相關資訊，請參閱 MFT agent.properties 檔案中的 **adminGroup** 內容。

作為 Java 應用程式，代理程式是一個 z/OS UNIX System Services 應用程式，您可以從針對代理程式產生的 Managed File Transfer 指令 PDSE 程式庫資料集，使用 BFGAGSTP 成員從 JCL 執行該應用程式。 如需如何建立 MFT 指令 PDSE 程式庫資料集以及針對必要代理程式自訂該資料集的相關資訊，請參閱 建立 MFT 代理程式或日誌程式指令資料集。

啟用代理程式以連接至遠端 z/OS 佇列管理程式

LTS

在下列實務範例中， z/OS 上的 MFT 代理程式可以使用用戶端連線連接至 z/OS 上的佇列管理程式:

- MFT 代理程式位於 IBM MQ 9.2.0 長期支援，已套用 APAR PH56722 ，且已與 IBM MQ Advanced for z/OS Value Unit Edition 或 IBM MQ Advanced for z/OS 的產品 ID (PID) 相關聯
- MFT 代理程式位於 IBM MQ 9.2.0 ，且已與 IBM MQ Advanced for z/OS Value Unit Edition 的 PID 相關聯

如需 IBM MQ 產品、其相關聯 PID 值及匯出分類的詳細資料，請參閱 IBM MQ 產品 ID 及匯出資訊 。

如需如何設定與 MFT 安裝相關聯之 PID 的相關資訊，請參閱 **fteSetProductID** 。

代理程式執行所在的 PID 會在代理程式啟動時顯示在日誌中。

註: z/OS 上以任何其他 PID 執行的 MFT 代理程式，只能使用連結模式連線來連接至本端佇列管理程式。

如果代理程式嘗試連接未在 z/OS 上執行的佇列管理程式，則會發出訊息 BFGQM1044E ，並結束代理程式啟動。

相關參考

第 178 頁的『在 z/OS 上停止 MFT 代理程式』
如果您從 JCL 以已啟動作業形式在 z/OS 上執行 Managed File Transfer Agent ，則除了 **fteStopAgent** 指令之外，代理程式還接受 z/OS 操作員指令 **MODIFY** 及 **STOP**。

MFT agent.properties 檔案

列出 MFT 代理程式

您可以使用指令行或 IBM MQ Explorer 來列出向特定佇列管理程式登錄的 Managed File Transfer 代理程式。

關於這項作業

若要使用指令行列出代理程式，請參閱 fteListAgents 指令。

若要使用 IBM MQ Explorer 來列出代理程式，請在「導覽器」視圖中按一下協調佇列管理程式名稱下的**代理程式**。

如果代理程式未由 **fteListAgents** 指令列出或未顯示在 IBM MQ Explorer 中，請使用下列主題中的診斷流程圖來尋找並修正問題: 如果 **fteListAgents** 指令未列出 MFT 代理程式，怎麼辦。

停止 MFT 代理程式

您可以從指令行停止 Managed File Transfer 代理程式。 停止代理程式時，代理程式開始靜止，但允許代理程式在停止之前完成現行檔案傳送。 您也可以在指令行指定 **-i** 參數，以立即停止代理程式。 代理程式停止之後，一直到重新啟動之前，您都無法使用該代理程式來傳送檔案。

開始之前

如果您想要檢查與佇列管理程式相關聯的代理程式名稱，您可以使用 IBM MQ Explorer 或指令行來列出代理程式，請參閱 fteListAgents 指令。

關於這項作業

若要從指令行停止代理程式，請參閱 fteStopAgent。

如果已將代理程式配置為以 Windows 服務執行，則執行 **fteStopAgent** 指令也會停止 Windows 服務。 或者，您也可以使用 Windows「服務」工具來停止服務，以停止代理程式。 如需相關資訊，請參閱 第 174 頁的『以 Windows 服務方式啟動 MFT 代理程式』主題。

z/OS 在 z/OS 上停止 MFT 代理程式

如果您從 JCL 以已啟動作業形式在 z/OS 上執行 Managed File Transfer Agent，則除了 **fteStopAgent** 指令之外，代理程式還接受 z/OS 操作員指令 **MODIFY** 及 **STOP**。

使用啟動的作業是因為它是以特定的使用者 ID 執行，且不受使用者登出影響。

註: 通常，據以執行啟動作業的管理使用者可能沒有登入專用權，並因此無法以執行代理程式的使用者身分登入 z/OS 系統。 無法對該代理程式發出 **fteStartAgent**、**fteStopAgent** 和 **fteSetAgentTraceLevel** 指令以及已指定 **-d** 參數的 **fteShowAgentDetails** 指令。

從 IBM MQ 9.0.2 及 IBM MQ 9.0.0 Fix Pack 1 中，您可以將代理程式內容 **adminGroup** 與 z/OS 上的 Managed File Transfer 代理程式搭配使用。 您可以定義安全管理員群組（例如 MFTADMIN），然後新增已啟動作業使用者 ID 和管理者 TSO ID 至此群組。 編輯代理程式內容檔，並將 **adminGroup** 內容設為此安全管理員群組的名稱。

```
adminGroup=MFTADMIN
```

此群組的成員隨後可以對正在當作已啟動作業執行的代理程式發出 **fteStartAgent**、**fteStopAgent** 和 **fteSetAgentTraceLevel** 以及已指定 **-d** 參數的 **fteShowAgentDetails** 指令。

如需相關資訊，請參閱 MFT agent.properties 檔案 中的 **adminGroup** 內容。

使用 z/OS MODIFY 指令 (F) 執行受管制的代理程式關閉

MODIFY 指令可讓您以受管制的方式停止代理程式，這是 **fteStopAgent** 指令的替代方案。 代理程式會完成目前正在進行的任何傳送，但代理程式不會啟動任何新的傳送。

例如：

```
F job_name,APPL=STOP
```

其中 *job_name* 是執行代理程式程序的工作。

使用 z/OS STOP 指令 (P) 執行立即的代理程式關閉

STOP 指令相等於使用 **fteStopAgent** 指令搭配 **-i** 參數執行的立即停止。 即使代理程式目前正在傳送檔案，代理程式也會立即停止。

例如：

```
P job_name
```

其中 *job_name* 是執行代理程式程序的工作。

相關參考

第 176 頁的『在 z/OS 上啟動 MFT 代理程式』

在 z/OS 上，除了從 z/OS UNIX System Services 階段作業執行 **fteStartAgent** 指令之外，您還可以從 JCL 以啟動作業形式啟動代理程式，而不需要互動式階段作業。

MFT agent.properties 檔案

啟動新的檔案傳送

您可以從「IBM MQ Explorer」或從指令行啟動新的檔案傳送，也可以選擇傳送單一檔案或以一個群組傳送多個檔案。

關於這項作業

若要從指令行啟動新的檔案傳送，請參閱 fteCreateTransfer 指令。

若要使用 IBM MQ Explorer 中的 **建立新的受管理檔案傳送** 精靈來啟動新的檔案傳送，請使用下列步驟:

程序

1. 在「導覽器」視圖中，按一下 **Managed File Transfer**。 **Managed File Transfer 中心**即會顯示在「內容」視圖中。
2. 所有協調佇列管理程式都會顯示在「導覽器」視圖中。 展開您要用於傳送的代理程式所登錄的協調佇列管理程式的名稱。 如果目前所連接的不是您想要用於傳送的協調佇列管理程式，請在「導覽器」視圖中的協調佇列管理程式名稱上按一下滑鼠右鍵，然後按一下**斷線**。 接著，用滑鼠右鍵按一下要使用的協調佇列管理程式名稱，並按一下**連接**。
3. 使用下列一種方法來啟動**建立新的 Managed File Transfer** 精靈:
   a) 在「導覽器」視圖中，在下列任何節點名稱上按一下滑鼠右鍵：相關的協調佇列管理程式、**傳送範本**、**傳送日誌**或**擱置傳送**。 然後，按一下**新建傳送**，以啟動精靈。
   b) 按一下**檔案** > **新建** > **其他** > **Managed File Transfer 精靈** > **新建傳送精靈**
4. 遵循精靈畫面上的指示操作。 每一個畫面也提供上下文相關說明。 若要存取 Windows 上的上下文相關說明，請按 F1 鍵。 在 Linux 上，按 Ctrl+F1 或 Shift+F1。

使用傳送定義檔

您可以指定傳送定義檔，它可用來建立檔案傳送。 此傳送定義檔是 XML 檔，它定義建立傳送需要的部分或全部資訊。

當您想要在單一傳送作業中指定多個來源檔案及多個目的地檔案時，傳送定義檔會很有幫助。 您可以使用傳送定義檔來提交複式檔案傳送。 您可以重複使用及共用傳送定義檔。

傳送定義檔可以使用兩種格式，這些格式大同小異，且都符合 FileTransfer.xsd 綱目。 您可以在 Managed File Transfer 安裝架構的 samples\schema 目錄中找到這個綱目。

下列是兩種支援的傳送定義檔格式:

- 傳送的來源和目的地檔案的定義。 此定義使用 **transferSpecifications** 元素作為根元素。
- 整個傳送的定義，包括來源和目的地檔案，以及來源和目的地代理程式。 此定義使用 **request** 元素作為根元素。
  - 此格式的檔案可以透過使用 **-gt** 參數的 **fteCreateTransfer** 指令產生。

下列範例顯示的傳送定義檔格式只指定傳送的來源和目的地檔案:

```
<?xml version="1.0" encoding="UTF-8"?>
<transferSpecifications xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <item checksumMethod="MD5" mode="text">
    <source recursive="false" disposition="leave">
      <file>textTransferTest.txt</file>
    </source>
    <destination type="directory" exist="overwrite">
```

```
      <file>c:\targetfiles</file>
    </destination>
  </item>
</transferSpecifications>
```

若要提交此格式的傳送定義檔，您必須在指令行上指定來源和目的地代理程式：

```
fteCreateTransfer -sa AGENT1 -sm agent1qm -da AGENT2 -dm agent2qm -td
  c:\definitions\example1.xml
```

下列範例是一個傳送定義檔格式，它指定傳送需要的全部資訊：

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="3.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>fteuser</userID>
    </originator>
    <sourceAgent agent="AGENT1" QMgr="agent1qm"/>
    <destinationAgent agent="AGENT2" QMgr="agent2qm"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\*.jpg</file>
        </source>
        <destination type="directory" exist="error">
          <file>/targetfiles/images</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

您可以在 **fteCreateTransfer** 指令上使用 **-gt** 參數，以產生此格式的檔案。 當您提交此格式的傳送定義檔時，不需要在指令行上指定任何其他項目：

```
fteCreateTransfer -td c:\definitions\example2.xml
```

除了傳送定義檔之外，您還可以傳入一般參數，以置換指令行的來源和目的地代理程式相關資訊。 例如：

```
fteCreateTransfer -da AGENT9 -dm agent9qm -td c:\definitions\example2.xml
```

此範例使用指令行選項，將傳送定義檔內定義的目的地代理程式置換成 **AGENT9**，並將傳送定義檔內定義的目的地佇列管理程式置換成 **agent9qm**。

上述兩種格式都包含一個以上的 <item> 元素。 For further information about the <item> element, see 檔案傳送要求訊息格式. 這些傳送項目各自定義一對來源和目的地檔案，以及控制傳送行為的其他屬性。 例如，您可以指定下列行為：

- 傳送是否使用總和檢查
- 傳送是文字還是二進位
- 傳送完成之後是否刪除來源檔案
- 是否改寫目的地檔案（如果該檔案存在的話）

使用傳送定義檔的優點是，您可以指定指令行無法提供的其他選項。 例如，當您執行「訊息至檔案」的傳送時，可使用傳送定義檔來指定 groupId 屬性。 此屬性指定從佇列中讀取的訊息的 IBM MQ 群組 ID。 傳送定義檔的另一個優點是，您可以為每一對檔案指定不同的選項。 例如，您可以指定是否使用總和檢查，或者是以文字模式還是以二進位模式逐一傳送檔案。 如果您使用指令行，則每一個傳送檔案都套用相同的選項。

例如：

```
  <item checksumMethod="none" mode="binary">
    <source disposition="leave">
      <file>c:\sourcefiles\source1.doc</file>
    </source>
```

```
    <destination type="file" exist="error">
      <file>c:\destinationfiles\destination1.doc</file>
    </destination>
  </item>

  <item checksumMethod="MD5" mode="text">
    <source disposition="delete">
      <file>c:\sourcefiles\source2.txt</file>
    </source>
    <destination type="file" exist="overwrite">
      <file encoding="UTF8" EOL="CRLF">c:\destinationfiles\destination2.txt</file>
    </destination>
  </item>

  <item checksumMethod="none" mode="text">
    <source recursive="false" disposition="leave">
      <file>c:\originfiles\source3.txt</file>
    </source>
    <destination type="file" exist="overwrite">
      <file>c:\targetfiles\destination3.txt</file>
    </destination>
  </item>
```

z/OS 您可以使用項目，將檔案從分散式系統傳送至 z/OS 系統：

```
z/OS
  <item checksumMethod="none" mode="text">
    <source recursive="false" disposition="leave">
      <file>textTransferTest.txt</file>
    </source>
    <destination type="dataset" exist="overwrite">
      <file encoding="IBM-1047">//TEXT.TRANS.TEST</file>
    </destination>
  </item>
```

z/OS 此範例是以文字模式，將 `textTransferTest.txt` 檔從來源代理程式傳送至目的地代理程式的資料集 `//TEXT.TRANS.TEST`。 此傳送將來源資料從來源代理程式的預設編碼（未指定任何來源編碼屬性）轉換成字碼頁：IBM-1047。

建立排定的檔案傳送

您可以從 IBM MQ Explorer 或從指令行排定新的檔案傳送。 排定的傳送可以包含群組中的單一檔案或多個檔案。 您可以執行排定的檔案傳送一次，或多次重複傳送。

關於這項作業

您可以設定執行檔案傳送排程一次，或按下列間隔執行：

- 每分鐘
- 每小時
- 每天一次
- 每週一次
- 每月
- 每年

然後，您可以指定在下列情況下停止執行：

- 在定義的時間及日期
- 在定義的執行次數之後

或者，您可以指定不斷地繼續執行。

如果排定的傳送每天在同一時間執行，請使用代理程式內容檔中的 **adjustScheduleTimeForDaylightSaving** 屬性，來調整時鐘變更時排程發生的時間。 如需相關資訊，請參閱 MFT agent.properties 檔案 。

若要使用指令行建立新的排定檔案傳送，請針對 fteCreateTransfer 指令使用排程參數 (**-tb**、 **-ss**、 **-oi**、 **-of**、 **-oc** 及 **-es**)。

若要使用 IBM MQ Explorer 中的 **建立新的受管理檔案傳送** 精靈來建立新的排定檔案傳送，請使用下列步驟:

程序

1. 在「導覽器」視圖中，按一下 **Managed File Transfer**。 **Managed File Transfer 中心**即會顯示在「內容」視圖中。
2. 所有協調佇列管理程式都會顯示在「導覽器」視圖中。 展開您要用於傳送的代理程式所登錄的協調佇列管理程式的名稱。 如果目前所連接的不是您想要用於傳送的協調佇列管理程式，請在「導覽器」視圖中的協調佇列管理程式名稱上按一下滑鼠右鍵，然後按一下**斷線**。 接著，用滑鼠右鍵按一下要使用的協調佇列管理程式名稱，並按一下**連接**。
3. 使用下列其中一種方法啟動 **建立新的 Managed File Transfer** 精靈:
   a) 在「導覽器」視圖中，在下列任何節點名稱上按一下滑鼠右鍵：相關的協調佇列管理程式、**傳送範本**、**傳送日誌**或**擱置傳送**。 然後，按一下**新建傳送**，以啟動精靈。
   b) 按一下**檔案 > 新建 > 其他 > Managed File Transfer 精靈 > 新建傳送精靈**
4. 遵循精靈畫面上的指示操作。 請確保已選取**啟用排程傳送**勾選框，然後在**排程**標籤上輸入您的排程明細。 如果沒有可能會影響傳送的問題，排定的檔案傳送會在排程開始時間的一分鐘內啟動。 例如，網路或代理程式可能發生問題，造成排定的傳送無法開始。 每一個畫面都提供有上下文相關說明。 若要存取 Windows 上的上下文相關說明，請按 F1 鍵。 在 Linux 上，按 Ctrl+F1 或 Shift+F1。

結果

如需排程檔案傳送所涉及之訊息的相關資訊，請參閱 排程檔案傳送日誌訊息格式。

處理擱置中檔案傳送

您可以透過「IBM MQ Explorer」來檢視擱置的排定檔案傳送。 **擱置傳送**視窗會顯示所有已向您目前所連接的協調佇列管理程式進行登錄的擱置傳送。

關於這項作業

若要檢視尚未開始的排定檔案傳送的狀態，請使用下列步驟:

程序

1. 在「導覽器」視圖中展開 **Managed File Transfer**。 **Managed File Transfer 中心**即會顯示在「內容」視圖中。
2. 所有協調佇列管理程式都會顯示在「導覽器」視圖中。 展開您用於排定傳送的協調佇列管理程式的名稱。 如果您想要變更所連接的協調佇列管理程式，請在「導覽器」視圖中您想要使用的協調佇列管理程式名稱上按一下滑鼠右鍵，然後再按一下**連接**。
3. 按一下**擱置傳送**。 「**擱置傳送**」視窗即會顯示在「內容」視圖中。
4. 「**擱置傳送**」視窗會顯示下列關於排定檔案傳送的詳細資料:
   a) **名稱** 排定檔案傳送的號碼。 會自動指派此號碼。
   b) **來源** 來源代理程式的名稱。
   c) **來源檔案** 要在其主機系統上傳送的檔案名稱。
   d) **目的地** 目的地代理程式的名稱。
   e) **目的地檔案** 檔案傳送至目的地系統之後的檔案名稱。
   f) **排定的開始時間（選取的時區）** 檔案傳送排定開始的時間和日期，以管理者選取的時區表示。 若要變更顯示的時區，請按一下 **視窗 > 喜好設定 > IBM MQ Explorer > Managed File Transfer**，然後從**時區：**清單中選取替代時區。 按一下**確定**。
   g) **重複間隔** 如果您選擇要重複排定的傳送，此即為您要重複執行傳送的指定間隔，以數字表示。

h) **重複類型** 如果您選擇要重複排定的傳送，此即為您為檔案傳送指定的重複間隔類型。 類型可以是下列其中一個值：**分鐘**、**小時**、**日**、**週**、**月**或**年**。

i) **重複截止** 如果您選擇要重複排定的傳送，此即為您想要結束重複檔案傳送的詳細時間。 例如，指定的日期和時間，或在指定的次數之後。

結果

若要重新整理**擱置傳送**視窗所顯示的內容，請按一下「內容」視圖工具列上的「重新整理」按鈕 。

若要取消擱置的檔案傳送，請在特定傳送上按一下滑鼠右鍵，然後按一下**取消**。 取消傳送後將會完全捨棄檔案傳送要求。

觸發檔案傳送

您可以設定檔案傳送的特定觸發條件，必須符合這些條件才會進行傳送。 如果不符合觸發條件，則不會進行檔案傳送，而且會選擇性地提交日誌訊息，以記錄未進行傳送的事實。 然後，會捨棄檔案傳送要求。 例如，您可以設定只有當來源代理程式所在系統上之指定檔案超出指定的大小時，或特定指定檔案存在於來源代理程式所在的系統上時，才會進行檔案傳送。 您可以從 IBM MQ Explorer 或從指令行設定觸發的檔案傳送。

關於這項作業

您可以持續監視資源直到滿足觸發條件。 如需資源監視的進一步相關資訊，請參閱：第 187 頁的『監視 MFT 資源』。

您可以設定三個不同觸發條件。 這些條件如下所示：

- 如果特定檔案存在於與來源代理程式相同的系統上
- 如果特定檔案不存在於與來源代理程式相同的系統上
- 如果特定檔案超出來源代理程式所在系統上的特定大小（此大小可以用位元組、KB、MB 或 GB 表示）。 這些度量單位使用 2^{10} 慣例，例如，1KB 等於 1024 個位元組，1MB 等於 1024KB。

之前清單中的觸發類型有兩種結合方式：

- 若為單一條件，您可以在來源代理程式所在系統上指定多個檔案。 如果任何一個指定的檔案符合該條件（布林運算子 OR），就會觸發傳送。
- 您可以指定多個條件。 只有在符合所有條件時（布林運算子 AND），才會觸發傳送。

您也可以結合觸發的傳送與排定的傳送。 如需相關資訊，請參閱建立排定的檔案傳送。 在此情況下，會在排程應該開始時評估觸發條件，若為重複排程，則在每次排程應該開始時評估觸發條件。

在通訊協定橋接器代理程式上，並不支援觸發的傳送。

若要使用指令行建立觸發檔案傳送，請在 fteCreateTransfer 指令上使用 **-tr** 參數。

若要使用 IBM MQ Explorer 中的 **建立新的受管理檔案傳送** 精靈來建立排定的檔案傳送，請使用下列步驟：

程序

1. 在「導覽器」視圖中，按一下 **Managed File Transfer**。 **Managed File Transfer 中心**即會顯示在「內容」視圖中。
2. 所有協調佇列管理程式都會顯示在「導覽器」視圖中。 展開您用於排定傳送的協調佇列管理程式的名稱。 如果您想要變更所連接的協調佇列管理程式，請在「導覽器」視圖中您想要使用的協調佇列管理程式名稱上按一下滑鼠右鍵，然後再按一下**連接**。
3. 使用下列一種方法來啟動**建立新的 Managed File Transfer** 精靈：
   a) 在「導覽器」視圖中，在下列任何節點名稱上按一下滑鼠右鍵：相關的協調佇列管理程式、**傳送範本**、**傳送日誌**或**擱置傳送**。 然後，按一下**新建傳送**，以開啟精靈。
   b) 按一下**檔案** > **新建** > **其他** > **Managed File Transfer 精靈** > **新建傳送精靈**

4. 遵循精靈畫面上的指示操作。 確定您在**觸發**標籤上選取**啟用已觸發的傳送**勾選框，並完成該標籤上的欄位以設定觸發。 每一個畫面都提供有上下文相關說明。 若要存取 Windows 上的上下文相關說明，請按 F1 鍵。 在 Linux 上，按 **Ctrl+F1** 或 Shift+F1。

監視正在進行的檔案傳送

您可以使用 IBM MQ Explorer 中的 **受管理檔案傳送-目前傳送進度** 標籤來監視進行中的檔案傳送。 這可以是從「IBM MQ Explorer」或指令行啟動的檔案傳送。 此標籤也會在排定的傳送開始時顯示該傳送的進度。

關於這項作業

如果您要使用「IBM MQ Explorer」來監視與遠端系統上協調佇列管理程式相關聯之傳送，請遵循第 185 頁的『配置 IBM MQ Explorer 以監視遠端協調佇列管理程式』主題中的指示。

在您停止並重新啟動「IBM MQ Explorer」之後，先前的檔案傳送資訊並不會保留下來。 在重新啟動時，即會從**目前傳送進度**標籤中清除舊傳送項目的相關資訊。 當 IBM MQ Explorer 開啟時，您可以隨時使用 **移除已完成的傳送** 來清除已完成的傳送。

程序

在使用「IBM MQ Explorer」或指令行啟動新的檔案傳送之後，您可以在**目前傳送進度**標籤中監視傳送進度。 每一個進行中的傳送都會顯示下列資訊：

a) **來源**。 用來從來源系統傳送檔案的代理程式名稱。

b) **目的地**。 用來在目的地系統上接收檔案的代理程式名稱。

c) **現行檔案**。 目前所傳送的檔案名稱。 個別檔案中已傳送的部分會以 B、KiB、MiB 顯示。 GiB 或 TiB 連同檔案大小總計放在括弧內。 顯示的測量單位視檔案大小而定。

B 是指每秒的位元組數。 KiB/s 是每秒的 kibibyte，1 kibibyte 等於 1024 個位元組。 MiB/s 是每秒的 mebibyte，1 mebibyte 等於 1,048,576 個位元組。 GiB/s 是每秒的 gibibyte，1 gibibyte 等於 1,073,741,824 個位元組。 TiB/s 是每秒的 tebibyte，1 tebibyte 等於 1,099,511,627,776 個位元組。

d) **檔案數**。 如果您傳送多個檔案，則此數代表整個檔案群組的傳送已進行至何處。

e) **進度**。 進度列會以百分比顯示目前的檔案傳送已完成多少。

f) **速率**。 以 KiB/s 表示檔案傳送的速率（每秒的 kibibyte，1 kibibyte 等於 1024 個位元組。）

g) **開始時間（選取的時區）**。 檔案傳送開始的時間，以管理者所選取的時區顯示。 若要變更顯示的時區，請按一下 **視窗 > 喜好設定 > IBM MQ Explorer > Managed File Transfer**，然後從**時區：**清單中選取替代時區。 按一下**確定**。

如果在傳送檔案的過程中傳送進入回復狀態，則開始時間將會更新，以反映檔案傳送回復的時間。

結果

此標籤會自動定期重新整理其資訊，但若要強制重新整理**目前傳送進度**標籤中的顯示內容，請按一下「內容」視圖工具列上的**重新整理**。

若要刪除**目前傳送進度**標籤中的檔案傳送，請按一下「內容」視圖工具列上的**移除已完成的傳送**。 按此按鈕只會從標籤中移除檔案傳送詳細資料，並不會停止或取消現行或排定的傳送。

如果您想要在關閉**目前傳送進度**標籤後返回該處，只要按一下**視窗 > 顯示視圖 > 其他 > 其他 > Managed File Transfer - 目前傳送進度**，即可顯示該標籤。 按一下**確定**。

下一步

此外，可以開發用於自訂檔案監視的應用程式。 這可以透過建立適當 Managed File Transfer 管理主題的訂閱來達成（以程式化方式或系統管理方式），這樣監視器應用程式主可以接收該主題上發佈的 Managed File Transfer 檔案傳送活動。 如需訂閱主題及發佈訊息格式的相關資訊，請參閱 檔案傳送進度訊息範例。

相關工作

第 185 頁的『配置 IBM MQ Explorer 以監視遠端協調佇列管理程式』

使用「IBM MQ Explorer」，可監視與遠端系統上所執行之協調佇列管理程式相關聯的檔案傳送。 在 IBM WebSphere MQ 7.5 版或更新版本中，您需要能夠執行「IBM MQ Explorer」的系統。 您必須安裝「IBM MQ Explorer」元件，才能連接遠端協調佇列管理程式。

第 185 頁的『在傳送日誌中檢視檔案傳送的狀態』
您可以使用 IBM MQ Explorer 中的 **傳送日誌** 來檢視檔案傳送的詳細資料。 這些可能是從指令行或 IBM MQ Explorer 啟動的傳送。 您也可以自訂在**傳送日誌**中顯示的內容。

配置 IBM MQ Explorer 以監視遠端協調佇列管理程式

使用「IBM MQ Explorer」，可監視與遠端系統上所執行之協調佇列管理程式相關聯的檔案傳送。 在 IBM WebSphere MQ 7.5 版或更新版本中，您需要能夠執行「IBM MQ Explorer」的系統。 您必須安裝「IBM MQ Explorer」元件，才能連接遠端協調佇列管理程式。

關於這項作業

假設：有權透過配置佇列管理程式以允許遠端連線，來連接至遠端協調佇列管理程式。

如需如何配置此項的相關資訊，請參閱 使用通道鑑別以用戶端模式連接至佇列管理程式 及 管理 MFT 特定資源的權限。

若要監視未執行 Windows 或 Linux 的系統上代理程式間的佇列管理程式及檔案傳送，請使用下列步驟將「IBM MQ Explorer」配置為連接至遠端系統：

程序

1. 啟動本端的「IBM MQ Explorer」。
2. 當「IBM MQ Explorer」載入後，請用滑鼠右鍵按一下 **Managed File Transfer** 資料夾，然後選取**新建配置**。
3. 繼續執行精靈，接著選取「協調佇列管理程式」與「指令佇列管理程式」，然後定義配置的名稱。
4. 按一下**完成**，以完成定義。
5. 定義完成時，用滑鼠右鍵按一下定義，然後選取**連接**。

結果

現在啟動「IBM MQ Explorer」，並使用它來監視與協調佇列管理程式相關聯之 Managed File Transfer 網路的傳送活動。

相關工作

第 184 頁的『監視正在進行的檔案傳送』
您可以使用 IBM MQ Explorer 中的 **受管理檔案傳送-目前傳送進度** 標籤來監視進行中的檔案傳送。 這可以是從「IBM MQ Explorer」或指令行啟動的檔案傳送。 此標籤也會在排定的傳送開始時顯示該傳送的進度。

第 185 頁的『在傳送日誌中檢視檔案傳送的狀態』
您可以使用 IBM MQ Explorer 中的 **傳送日誌** 來檢視檔案傳送的詳細資料。 這些可能是從指令行或 IBM MQ Explorer 啟動的傳送。 您也可以自訂在**傳送日誌**中顯示的內容。

在傳送日誌中檢視檔案傳送的狀態

您可以使用 IBM MQ Explorer 中的 **傳送日誌** 來檢視檔案傳送的詳細資料。 這些可能是從指令行或 IBM MQ Explorer 啟動的傳送。 您也可以自訂在**傳送日誌**中顯示的內容。

程序

1. 在「導覽器」視圖中展開 **Managed File Transfer**，然後展開您想要檢視傳送日誌的協調佇列管理程式的名稱。
2. 在「導覽器」視圖中按一下**傳送日誌**。 「**傳送日誌**」即會顯示在「內容」視圖中。
3. 「**傳送日誌**」視窗會顯示下列關於檔案傳送的詳細資料：
   a) **來源** 來源檔案所在系統上的代理程式名稱。

b) **目的地** 您想要傳送檔案至其中的目的地系統上的代理程式名稱。

c) **完成狀態** 檔案傳送的狀態。 其狀態可以是下列其中一個值：「已啟動」、「進行中」、「成功」、「部分成功」、「已取消」或「失敗」。

d) **擁有者** 主機上提交傳送要求的使用者 ID。

e) **開始時間（選取的時區）** Managed File Transfer 代理程式接受檔案傳送要求的時間和日期，以管理者選取的時區表示。 若要變更顯示的時區，請按一下**視窗** > **喜好設定** > **IBM MQ Explorer** > **Managed File Transfer**，然後從**時區：**清單中選取替代時區。 按一下**確定**。

f) **狀態記錄（選取的時區）** （依預設，不顯示此直欄。 您可以使用**配置傳送日誌直欄** 視窗來選擇要顯示此直欄。） 記錄完成狀態的日期和時間，以管理者選取的時區表示。

g) **工作名稱** 使用者使用 **fteCreateTransfer** 的 **-jn** 參數或在 Ant Script 中指定的 ID

h) **傳送 ID** 檔案傳送的唯一 ID。

i) **Connect: Direct** 會列出有關**程序號碼**、**程序名稱**、**主要節點**、**次要節點**、**來源類型**和**目的地類型**的詳細資料。

結果

註: 在 APAR IC99545 中，IBM MQ 8.0.0 Fix Pack 1 已變更「傳送日誌」的內部格式。 因此，如果 IBM MQ Explorer 升級至 V8.0.0.1 或更新版本，然後還原至 V8.0.0.0，則對於 IBM MQ Explorer 處於 V8.0.0.1 時發生的傳送，不會顯示任何審核 XML。 在「**內容**」視窗中，這些傳送的 XML 畫面只包含空文字框。

若要檢視已完成傳送的進一步詳細資料，請按一下加號 (+) 來展開您感興趣的傳送。 然後，您可以查看該傳送中包含的所有來源及目的地檔案名稱。 不過，如果傳送目前正在進行，而且是由許多檔案組成，則您只能檢視到目前為止已傳送的檔案。

若要重新整理**傳送日誌**中顯示的內容，請按一下「內容」視圖工具列上的**重新整理**按鈕 。 在您停止並重新啟動 IBM MQ Explorer 之後，「傳送日誌」中的檔案傳送資訊會保留在日誌中。 如果您想要刪除日誌中所有已完成的檔案傳送，請按一下「內容」視圖工具列上的**移除已完成的傳送** 。

若要從日誌中刪除個別的已完成檔案傳送，請在該傳送上按一下滑鼠右鍵，然後按一下**刪除**。 如果您刪除傳送，它不會停止或取消進行中或已排定的傳送；您只會刪除已儲存的歷程資料。

若要將傳送的唯一 ID 複製到剪貼簿，請在該傳送上按一下滑鼠右鍵，然後按一下**複製 ID**。

利用蹦現功能表的**內容**動作，即可存取傳送的 meta 資料及完整的審核 XML。

相關工作

第 184 頁的『監視正在進行的檔案傳送』
您可以使用 IBM MQ Explorer 中的 **受管理檔案傳送-目前傳送進度** 標籤來監視進行中的檔案傳送。 這可以是從「IBM MQ Explorer」或指令行啟動的檔案傳送。 此標籤也會在排定的傳送開始時顯示該傳送的進度。

第 186 頁的『配置傳送日誌』
您可以在 IBM MQ Explorer 中配置要顯示哪些資訊，以及如何在 **傳送日誌** 中顯示資訊。

第 258 頁的『設定回復停滯傳送的逾時』
您可以針對套用至來源代理程式所有傳送的已停滯檔案傳送，設定傳送回復逾時。 您也可以設定個別傳送的傳送回復逾時。 如果您設定特定時間量 (以秒為單位)，在此期間，來源代理程式會持續嘗試回復已停滯的檔案傳送，且當代理程式達到逾時時，傳送不會成功，則傳送會失敗。

配置傳送日誌

您可以在 IBM MQ Explorer 中配置要顯示哪些資訊，以及如何在 **傳送日誌** 中顯示資訊。

關於這項作業

若要重新排列**傳送日誌**中直欄的順序，請按一下要移動的直欄標題，然後將直欄拖曳至新位置。 新直欄順序僅保留到您下一次停止並重新啟動「IBM MQ Explorer」為止。

若要過濾**傳送日誌**中的項目，請在**過濾顯示的日誌項目**欄位中輸入字串。 若要將所有項目還原回日誌，請刪除您在欄位中輸入的字串。 您可以在此欄位中使用任何有效的 Java 正規表示式。 如需相關資訊，請參閱 MFT 使用的正規表示式。

若要自訂「傳送日誌」中顯示的直欄，請使用**配置傳送日誌直欄** 。 按照下列步驟啟動並使用**配置傳送日誌直欄**視窗。

程序

1. 確保您已在「內容」視圖中開啟**傳送日誌**。 按一下「內容」視圖工具列上的**配置傳送日誌直欄** 。 **配置傳送日誌直欄**視窗即會開啟。
2. 若要自訂**傳送日誌**的視圖，請針對您要顯示或隱藏的直欄選取或清除個別勾選框。 您可以按一下**全選**，然後按一下**確定**以選取所有勾選框，或按一下**取消全選**，然後按一下**確定**以清除所有勾選框。

相關工作

第 184 頁的『監視正在進行的檔案傳送』
您可以使用 IBM MQ Explorer 中的 **受管理檔案傳送-目前傳送進度** 標籤來監視進行中的檔案傳送。 這可以是從「IBM MQ Explorer」或指令行啟動的檔案傳送。 此標籤也會在排定的傳送開始時顯示該傳送的進度。

第 185 頁的『在傳送日誌中檢視檔案傳送的狀態』
您可以使用 IBM MQ Explorer 中的 **傳送日誌** 來檢視檔案傳送的詳細資料。 這些可能是從指令行或 IBM MQ Explorer 啟動的傳送。 您也可以自訂在**傳送日誌**中顯示的內容。

監視 MFT 資源

您可以監視 Managed File Transfer 資源；例如，佇列或目錄。 滿足此資源的條件時，資源監視器會啟動作業，例如檔案傳送。 您可以在 IBM MQ Explorer 的 Managed File Transfer 外掛程式中使用 **fteCreateMonitor** 指令或「 **監視器** 」視圖來建立資源監視器。

關於這項作業

Managed File Transfer 資源監視使用下列術語：

資源監視器
資源監視器是以預先定義的定期間隔輪詢資源 (例如目錄或佇列) 的處理程序，以查看資源內容是否已變更。 如果發現變更，則比較內容與此監視器的條件組。 如果符合，則開始執行此監視器的作業。

資源
資源監視器在每一個輪詢間隔檢查的系統資源，以與觸發條件相互比較。 受監視資源包括佇列、目錄或巢狀目錄結構。

條件及觸發條件
條件是評估的表示式 (通常是針對受監視資源的內容)。 如果表示式評估為 true，則條件會組成整體觸發條件。

觸發條件是在滿足所有條件時滿足的整體條件。 滿足觸發條件時，作業可繼續執行。

作業
作業是在滿足觸發條件或條件集時啟動的作業。 支援的作業包含檔案傳送及指令呼叫。

觸發檔案
觸發檔案是放置在監視目錄中的檔案，指出作業 (通常是傳送) 可以開始。 例如，可能指出所有要處理的檔案已送達已知位置，可進行傳送或處理。 觸發檔案名稱可用來指定要傳送的檔案（使用變數替代）。 如需相關資訊，請參閱 第 196 頁的『使用變數替代來自訂 MFT 資源監視器作業』。

觸發檔案也稱為備妥檔案或執行檔案。 不過，在此文件中，它通常稱為觸發檔案。

通訊協定橋接器代理程式或 Connect:Direct 橋接器代理程式上不支援資源監視。

MFT 資源監視概念

Managed File Transfer 資源監視特性的主要概念概觀。

資源監視器

您可以使用 **fteCreateMonitor** 指令來建立資源監視器，這會從指令行建立並啟動新的資源監視器。 資源監視器與 Managed File Transfer 代理程式相關聯，且只有在該代理程式執行時才會作用中。 當監視代理程式停止時，資源監視器也會停止。 當建立資源監視器時，如果代理程式已在執行中，則資源監視器會立即啟動。 監視代理程式也必須是資源監視器所起始之作業的來源代理程式。

資源監視器名稱在其代理程式內必須是唯一的。 資源監視器名稱長度必須至少為一個字元，且不得包含星號 (*)、百分比 (%) 或問號 (?) 字元。 系統會忽略提供資源監視器名稱的大小寫，並將資源監視器名稱轉換為大寫。 如果您嘗試建立名稱已存在的資源監視器，則會忽略要求，並將嘗試記載至資源監視器日誌主題。

註: 您無法使用包含排定傳送的作業定義來建立資源監視器。

V 9.2.2 若為 Long Term Support，以及 Continuous Delivery 之前 IBM MQ 9.2.2，停止資源監視器的唯一方法是停止執行監視作業的代理程式。 若要重新啟動資源監視器，您必須完全重新啟動代理程式。 從 IBM MQ 9.2.2 開始，您可以啟動及停止資源監視器，而不需要停止或重新啟動代理程式。 如需相關資訊，請參閱 第 190 頁的『啟動及停止資源監視器』。

可以在代理程式上建立且全部以相同優先順序執行的資源監視器數目沒有限制。 請考慮重疊的受監視資源、衝突的觸發條件及資源輪詢頻率的含意。

重疊資源監視器可能導致:

- 來源位置/項目的可能競用。
- 相同來源項目的可能重複傳送要求。
- 由於來源項目衝突，傳送發生非預期的錯誤或失敗。

如果多個監視器掃描相同位置，並且可以在相同項目上觸發，則您可能最終會遇到兩個不同監視器針對相同項目提交受管理傳送要求的問題。

資源監視器會在每一個輪詢間隔期間之後查看資源的內容。 資源的內容會與觸發條件相互比較，如果滿足這些條件，則會呼叫與資源監視器相關聯的作業。

作業會非同步地啟動。 如果有條件相符，且已啟動作業，則資源監視器會繼續輪詢資源內容的進一步變更。 因此，舉例來說，如果因為一個稱為 `reports.go` 的檔案到達監視目錄中而發現相符項目，則作業會啟動一次。 在下次輪詢間隔時，即使該檔案仍存在，作業就不會再次啟動。 不過，如果檔案被刪除，然後再次放入目錄中，或檔案已更新（亦即前次修改日期屬性變更），則下一次檢查觸發條件時會導致再次呼叫作業。

在 IBM MQ 9.1.5 之前，如果資源監視器執行輪詢所花費的時間超過輪詢間隔，則這表示下一個輪詢會在現行輪詢完成且兩者之間沒有間隙時立即開始，這可能會影響資源監視器將工作提交至代理程式的速度。 如果在第一次輪詢期間找到的項目在第二次輪詢發生時仍然存在，則這可能會導致效能問題。

V 9.2.0 從 IBM MQ 9.1.5 開始，資源監視器會使用 ScheduledExecutor 服務，並只在完成前一個輪詢加上配置的輪詢間隔時間之後，才起始下一個輪詢。 這表示輪詢間隔之間一律會有差距，如果輪詢時間超過輪詢間隔，則不會在前一個輪詢之後立即開始另一個輪詢。

V 9.2.0 從 IBM MQ 9.1.3 開始，如果檔案無法傳送，您可以清除資源監視器歷程，以容許提交另一個傳送要求，而不需要刪除檔案並重新將它放置在目錄中，或更新檔案以變更其前次修改日期屬性。 清除歷程很有用，例如，在需要傳送檔案但無法修改檔案的情況下。 如需相關資訊，請參閱 第 210 頁的『正在清除資源監視器歷程』。

資源

Managed File Transfer 中的資源監視器可以輪詢下列兩種類型資源的內容:

目錄或巢狀目錄結構
一種常見的情況是監視目錄中是否有觸發檔案。 外部應用程式可能處理多個檔案，並將它們放在已知的來源目錄。 應用程式完成處理之後，表示可透過將觸發檔案放入受監視的位置以準備傳送或處理檔案。 Managed File Transfer 資源監視器可以偵測觸發檔案，並起始將這些檔案從來源目錄傳送至另一個 Managed File Transfer Agent 。

依預設，會監視指定的目錄。 如果也要檢查子目錄，請在 **fteCreateTransfer** 指令中設定遞迴層次。

兩個監視目錄範例，如下所示：

- 監視觸發檔案 (例如 `trigger.file`)，然後傳送萬用字元 (例如 `*.zip`)。
- 監視 `*.zip`，然後傳送 `${FilePath}`（例如，觸發傳送的檔案）。 如需變數替代的相關資訊，請參閱第 196 頁的『使用變數替代來自訂 MFT 資源監視器作業』。

註: 請不要建立監視器來監視 `*.zip` 又傳送 `*.zip`。 此監視器會對系統上的每個 .zip 檔開始進行 `*.zip` 的傳送。 亦即，監視器會為 `*.zip` 產生 * 次傳送。

如需建立資源監視器以監視目錄的範例，請參閱第 194 頁的『監視目錄及使用變數替代』。

IBM MQ 佇列
監視佇列的範例是外部應用程式可能正在產生訊息，並將它們放置在具有相同群組 ID 的已知佇列上。 當應用程式完成將訊息放置在佇列上後，它會指出是完整群組。 Managed File Transfer 資源監視器可以偵測完整的訊息群組，並起始將訊息群組從來源佇列傳送至檔案。 如需建立資源監視器來監視佇列的範例，請參閱第 196 頁的『範例：配置 MFT 資源』。

註: 每個佇列只能指定一個監視器。 如果指定多個監視器來輪詢 IBM MQ 佇列，將發生無法預期的行為。

不支援監視資料集。

條件及觸發條件

當資源包含的值符合其他一些字串或型樣時，即符合條件。 條件可以是下列其中一項：

- 檔案名稱（型樣）相符
- 檔案名稱（型樣）不符
- 檔案大小
- 在許多次輪詢中，如果檔案大小維持相同，則符合

檔案名稱比對可以表示為：

- 字串完全相符
- 簡式萬用字元比對，如 搭配使用萬用字元與 MFT 中所述
- 正規表示式相符

也可以使用萬用字元或 Java 正規表示式來識別絕不相符的檔名，從檔案名稱比對作業中排除檔案名稱。

偵測到相符的檔案時，會保留前次修改時間的時間戳記。 如果後續輪詢偵測到檔案已變更，則再次滿足觸發條件時，作業就會啟動。 如果在檔案不存在時偵測條件，且監視目錄中沒有任何檔案符合檔案名稱型樣，則會啟動作業。 如果後來新增至目錄中的檔案符合檔案名稱型樣，則只有在該檔案被刪除時才會啟動作業。

作業

Managed File Transfer 支援下列兩種作業，可讓您配置為由資源監視器啟動：

檔案傳送作業
檔案傳送作業的定義方式與任何其他檔案傳送相同。 產生監視器所需的作業 XML 最有用的作法是執行 fteCreateTransfer 指令並指定 **-gt** 參數。 此指令會產生 XML 文件形式的作業定義，包括傳送規格。 然後，您可以在 fteCreateMonitor 指令上傳遞作業 XML 文件的名稱作為 **-mt** 參數的值。 **fteCreateMonitor** 執行時會讀取作業 XML 文件。 執行 **fteCreateMonitor** 之後，監視器不會再使用作業 XML 檔案的任何變更。

使用檔案傳送作業時，您可以選取要將多少觸發條件分批放入作業中。 預設是一個觸發條件啟動一個作業。 您可以執行 fteCreateMonitor 指令並搭配 **-bs** 選項，以選取要整批放入一個作業中的觸發條件數目。

指令作業
指令作業可以執行 Ant Script、呼叫可執行程式或執行 JCL 工作。 如需相關資訊，請參閱第 191 頁的『配置 MFT 監視作業以啟動指令及 Script』。

觸發檔案

您可以在資源監視器中使用觸發檔案的內容，定義要在單一傳送要求中傳送的檔案集。 每次偵測到相符的觸發檔案時，系統都會剖析其內容，以取得來源檔案路徑，並選擇性地取得目的地檔案路徑。 然後，這些檔案路徑會用來定義您指定的作業傳送 XML 檔（以單一傳送要求形式提交至代理程式）中的檔案項目。 資源監視器的定義會決定是否啟用觸發內容。

每一個觸發檔案的格式是在每一行文字上顯示要傳送的單一檔案路徑。 預設的行格式為單一來源檔案路徑，或是以逗點區隔的來源及目的地檔案路徑。

如需相關資訊與範例，請參閱第 203 頁的『使用觸發檔』。

啟動及停止資源監視器

若為 Long Term Support，以及 Continuous Delivery 之前 IBM MQ 9.2.2，停止資源監視器的唯一方法是停止執行監視作業的代理程式。 若要重新啟動資源監視器，您必須完全重新啟動代理程式。 如需相關資訊，請參閱第 173 頁的『啟動 MFT 代理程式』及第 178 頁的『停止 MFT 代理程式』。

V 9.2.2 從 IBM MQ 9.2.2 開始，您可以使用 **fteStartMonitor** 及 **fteStopMonitor** 指令來啟動及停止資源監視器，而不需要停止或重新啟動代理程式。 例如，在下列情況下，這很有用:

- 如果代理程式有多個資源監視器，且只有部分資源監視器發生錯誤，但其餘資源監視器仍正常運作，因此您只想要重新啟動失敗的資源監視器。
- 如果您想要停止資源監視器來執行某些維護工作，或如果在特定時間內不需要資源監視器，且您不想要它執行不必要的動作，則會耗用寶貴的系統資源。

V 9.2.2 如需相關資訊，請參閱 啟動 MFT 資源監視器 及 停止 MFT 資源監視器。

V 9.2.2

表 9: 資源監視器的行為，視執行的指令而定

指令	資源監視器的行為
V 9.2.2 **fteStartMonitor**	如果代理程式正在執行中，則會啟動資源監視器 (如果目前已停止)。
V 9.2.2 **fteStopMonitor**	如果代理程式正在執行中，則資源監視器會停止 (如果目前已啟動)。
fteStartAgent	在代理程式啟動，不論先前對 **fteStopMonitor** 的呼叫為何的過程中，會啟動資源監視器。
fteStopAgent	任何執行中的資源監視器都會停止。

備份及還原資源監視器

您可以備份已經定義的資源監視器，以便未來可重複使用它們。 您可以使用不同選項，如下所示:

- 搭配使用 **fteCreateMonitor** 指令與 **-ox** 參數可將資源監視器配置匯出至 XML 檔，搭配 **-ix** 參數可透過從 XML 檔匯入資源監視器配置來還原資源監視器。
- 搭配使用 **fteListMonitors** 指令與 **-ox** ，可將單一資源監視器的定義匯出至 XML 檔案。
- 搭配使用 **fteListMonitors** 指令與 **-od** ，可將多個資源監視器定義匯出至指定的目錄。 每個資源監視器定義皆會儲存至個別 XML 檔。 您也可以使用 **-od** 選項，將單一資源監視器定義匯出至指定的目錄。

如需相關資訊，請參閱 第 209 頁的『備份及還原 MFT 資源監視器』。

資源監視器記載

從 IBM MQ 9.1.0 開始，Managed File Transfer 會併入資源監視器記載。 如需相關資訊，請參閱 第 205 頁的『記載 MFT 資源監視器』。

相關概念

第 196 頁的『使用變數替代來自訂 MFT 資源監視器作業』
滿足作用中資源監視器的觸發條件時，會呼叫所定義的作業。 除了每次以相同的目的地代理程式或相同的目的地檔案名稱呼叫傳送或指令作業之外，您還可以在執行時期修改作業定義。 您可以將變數名稱插入作業定義 XML 中，來達成此目的。 當監視器判斷已滿足觸發條件，且作業定義包含變數名稱時，它會以變數值替換變數名稱，然後呼叫該作業。

相關工作

第 191 頁的『配置 MFT 監視作業以啟動指令及 Script』
資源監視器不限於將檔案傳送作為其關聯的作業執行。 您也可以配置監視器以從監視代理程式呼叫其他指令，包括可執行程式、Ant Script 或 JCL 工作。 若要呼叫指令，請編輯監視作業定義 XML，以併入具有對應指令呼叫參數的多個指令元素，如引數及內容。

第 196 頁的『範例：配置 MFT 資源』
您可以搭配使用 **-mq** 參數與 **fteCreateMonitor** 指令，將 IBM MQ 佇列指定為資源監視器要監視的資源。

第 201 頁的『監視佇列及使用變數替代』
您可以使用 **fteCreateMonitor** 指令來監視佇列，並將訊息從受監視佇列傳送至檔案。 在第一個要從受監視佇列讀取的訊息中，任何 IBM MQ 訊息內容的值都可代入作業 XML 定義中，並用來定義傳送行為。

相關參考

fteCreateMonitor：建立 MFT 資源監視器

fteListMonitors：列出 MFT 資源監視器

fteDeleteMonitor：刪除 MFT 資源監視器

配置 MFT 監視作業以啟動指令及 Script

資源監視器不限於將檔案傳送作為其關聯的作業執行。 您也可以配置監視器以從監視代理程式呼叫其他指令，包括可執行程式、Ant Script 或 JCL 工作。 若要呼叫指令，請編輯監視作業定義 XML，以併入具有對應指令呼叫參數的多個指令元素，如引數及內容。

關於這項作業

您要監視代理程式呼叫的可執行程式、 Ant Script 或 JCL 工作的檔案路徑必須包含在監視代理程式的 commandPath 中。 如需指令路徑內容的相關資訊，請參閱 commandPath MFT 內容。

您可以按下列其中一種方式建立作業定義 XML 文件：

- 根據 FileTransfer.xsd 綱目手動建立作業定義 XML 文件。
- 使用產生的 XML 文件作為作業定義的基礎。

無論想要傳送作業還是指令作業，作業定義必須以 <request> 根元素開頭。 <request> 的子元素必須為 <managedTransfer> 或 <managedCall>。 在具有要執行的單一指令或 Script 時，您通常要選擇 <managedCall>；如果您希望作業併入檔案傳送並選擇性地併入最多四個指令呼叫，則選擇 <managedTransfer>。

程序

- 若要根據 FileTransfer.xsd 綱目手動建立作業定義 XML 文件，請參閱 第 191 頁的『根據綱目手動建立作業定義 XML』。
- 若要透過修改產生的文件來建立作業定義，請編輯 **fteCreateTransfer -gt** 參數所產生的 XML 文件。 如需相關資訊，請參閱第 193 頁的『修改產生的文件來建立作業定義文件』。

根據綱目手動建立作業定義 *XML*

您可以根據綱目 FileTransfer.xsd 手動建立作業定義 XML 檔案。

關於這項作業

綱目 FileTransfer.xsd 可在 *MQ_INSTALLATION_PATH*/mqft/samples/schema 中找到。 如需此綱目的相關資訊，請參閱 檔案傳送要求訊息格式。

範例

下列範例顯示儲存為 cleanuptask.xml，的作業定義 XML 文件範例，它使用 <managedCall> 元素來呼叫稱為 RunCleanup.xml 的 Ant Script。 RunCleanup.xml Ant Script 必須位於監視代理程式的 commandPath 上。

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedCall>
     <originator>
       <hostName>hostName</hostName>
       <userID>userID</userID>
       <mqmdUserID>mqmdUserID</mqmdUserID>
     </originator>
     <agent QMgr="QM1" agent="AGENT1"/>
     <reply QMGR="QM1">reply</reply>
     <transferSet priority="1">
       <metaDataSet>
         <metaData key="name1">value1</metaData>
       </metaDataSet>
       <call>
           <command name="RunCleanup.xml" type="antscript" retryCount="2"
          retryWait="30" successRC="0">
           <target>check_exists</target>
           <target>copy_to_archive</target>
           <target>rename_temps</target>
             <target>delete_files</target>
           <property name="trigger.filename" value="${FileName}"/>
           <property name="trigger.path" value="${FilePath}"/>
           </command>
       </call>
     </transferSet>
     <job>
       <name>JOBCLEAN1</name>
     </job>
   </managedCall>
</request>
```

<agent> 元素指定在其 commandPath 上使用具名 Ant Script 所配置的 Managed File Transfer Agent 。

<call><command>... 結構定義您要執行的執行檔或 Script。 此指令會採用選用的 type 屬性，此屬性可能具有下列其中一個值：

antscript
在個別 JVM 中執行 Ant Script。

executable
呼叫可執行程式。

jcl
呼叫 JCL 工作。

os4690background
此值必須適用於在 4690 作業系統上執行的 IBM MQ 8.0 或更早的管理式檔案傳輸代理。 不再支援此作業系統，且不再使用此值。

如果您省略 type 屬性，將使用預設值 executable。

name 屬性指定您要執行的 Ant Script、執行檔或 JCL 工作的名稱，不含任何路徑資訊。 代理程式會在代理程式的 agent.properties 檔案中的 commandPath 內容指定的位置中，搜尋 Script 或程式。

retrycount 屬性會在程式未傳回成功回覆碼時，指定再次嘗試呼叫程式的次數。 指派給此屬性的值不得為負數。 如果未指定 retrycount 屬性，則會使用預設值零。

retrywait 屬性指定在再次嘗試呼叫程式之前等待的時間（以秒為單位）。 指派給此屬性的值不得為負數。 如果未指定 retrywait 屬性，則會使用預設值零。

successrc 屬性是用於決定何時順利執行程式呼叫的表示式。 指令的程序回覆碼會使用此表示式進行評估。 值可能由多個表示式結合垂直線 (|) 字元（表示布林 OR）或 '&' 符號字元（表示布林 AND）組成。 每一個表示式可以是下列其中一種表示式類型：

- 一個數字，表示程序回覆碼與此數之間的「相等」測試。

- 一個以大於字元 (>) 為字首的數字，表示數字與程序回覆碼之間的大於測試。
- 一個以小於字元 (<) 為字首的數字，表示數字與程序回覆碼之間的小於測試。
- 一個以驚嘆號字元 (!) 為字首的數字，表示數字與程序回覆碼之間的不等於測試。 例如：>2&<7&!5|0|14 將下列回覆碼解譯為成功回覆碼：0、3、4、6、14。 其餘全都會解譯為不成功回覆碼。

如果未指定 successrc 屬性，則會使用預設值零。 這表示只有在回覆碼為零時，才會判定指令已順利執行。

對於 Ant Script ，您通常會指定 <target> 和 <property> 元素。 <target> 元素值必須符合 Ant Script 中的目標名稱。

對於可執行程式，您可以指定 <argument> 元素。 巢狀引數元素指定傳遞至作為程式呼叫一部分呼叫的程式。 按遇到引數元素的順序透過引數元素指定的值建立程式引數。 您可以指定零個以上引數元素作為程式呼叫的巢狀元素。

管理者使用包含 <managedCall> 元素的作業定義 XML 文件來照常定義及啟動監視器。 例如：

```
fteCreateMonitor -ma AGENT1 -mm QM1 -md /monitored -mn MONITOR01 -mt
   /tasks/cleanuptask.xml -pi 30 -pu seconds -tr match,*.go
```

傳送定義 XML 文件的路徑必須在您從中執行 **fteCreateMonitor** 指令的本端檔案系統上（在此範例中為 /tasks/cleanuptask.xml）。 cleanuptask.xml 文件僅用於建立資源監視器。 cleanuptask.xml 文件參照的任何作業（Ant Scripts 或 JCL 工作）必須在監視代理程式的指令路徑中。 當符合監視觸發條件時，使用監視器中的實際值替代作業定義 XML 中的任何變數。 例如，在傳送至代理程式的要求訊息中，使用 /monitored/cleanup.go 取代 ${FilePath}。 要求訊息放在代理程式指令佇列上。 指令處理器偵測到要求適用於程式呼叫，並啟動指定的程式。 如果呼叫 antscript 類型的指令，則會啟動新的 JVM ，且 Ant 作業會在新的 JVM 下執行。 如需使用變數替代的相關資訊，請參閱使用變數替代值來自訂作業。

相關概念

第 196 頁的『使用變數替代來自訂 MFT 資源監視器作業』
滿足作用中資源監視器的觸發條件時，會呼叫所定義的作業。 除了每次以相同的目的地代理程式或相同的目的地檔案名稱呼叫傳送或指令作業之外，您還可以在執行時期修改作業定義。 您可以將變數名稱插入作業定義 XML 中，來達成此目的。 當監視器判斷已滿足觸發條件，且作業定義包含變數名稱時，它會以變數值替換變數名稱，然後呼叫該作業。

相關參考

檔案傳送要求訊息格式

commandPath MFT 內容

修改產生的文件來建立作業定義文件

您可以透過修改 **fteCreateTransfer** 的 **-gt** 選項產生的 XML 文件，來建立監視作業定義文件。

關於這項作業

產生的文件具有 <request>，後面接著 <managedTransfer> 元素。 若要將此作業定義轉換為有效的 <managedCall> 結構，請遵循下列步驟：

程序

1. 將 <managedTransfer> 的開始和結束標籤取代為 <managedCall> 標籤。
2. 移除任何 <schedule> 元素及子節點。
3. 將 <sourceAgent> 的開始和結束標籤取代為 <agent>，以與監視代理程式配置詳細資料相符。
4. 移除 <destinationAgent> 和 <trigger> 元素。
5. 移除 <item> 元素。
6. 移除任何 preSourceCall、 postSourceCall、 preDestinationCall 或 postDestinationCall 元素。
7. 在 <transferSet> 元素內插入新的 <call>...</call> 結構。 此結構包含指令定義，如下列範例所示：

```
<call>
          <command name="RunCleanup.xml" type="antscript" retryCount="2"
         retryWait="30" successRC="0">
          <target>check_exists</target>
          <target>copy_to_archive</target>
          <target>rename_temps</target>
            <target>delete_files</target>
          <property name="trigger.filename" value="${FileName}"/>
          <property name="trigger.path" value="${FilePath}"/>
          </command>
       </call>
```

範例

您也可以保留 <managedTransfer> 元素（包括所有檔案傳送詳細資料），並插入最多四個指令呼叫。 在此情況下，您可以在 <metaDataSet> 與 <item> 元素之間插入下列呼叫元素的任何選項：

preSourceCall
在啟動傳送之前，呼叫來源代理程式上的程式。

postSourceCall
在完成傳送之後，呼叫來源代理程式上的程式。

preDestinationCall
在啟動傳送之前，呼叫目的地代理程式上的程式。

postDestinationCall
在完成傳送之後，呼叫目的地代理程式上的程式。

其中每一個元素都會採用 <command> 元素結構，如較早範例所述。 FileTransfer.xsd 綱目定義各種呼叫元素使用的類型。

下列範例顯示作業定義文件中的 preSourceCall、postSourceCall、preDestinationCall 及 postDestinationCall：

```
⋮
    <transferSet priority="1">
       <metaDataSet>
         <metaData key="key1">value1</metaData>
       </metaDataSet>
       <preSourceCall>
         <command name="send.exe" retryCount="0" retryWait="0" successRC="0"
           type="executable">
           <argument>report1.pdf</argument>
           <argument>true</argument>
         </command>
       </preSourceCall>
       <postSourceCall>
         <command name="//DO_IT.JCL" retryCount="0" retryWait="0" successRC="0"
           type="jcl">
           <argument>argument</argument>
         </command>
       </postSourceCall>
       <preDestinationCall>
         <command name="ant_script.xml" retryCount="0" retryWait="0" successRC="0"
           type="antscript">
           <target>step1</target>
           <property name="name" value="value"/>
         </command>
       </preDestinationCall>
       <postDestinationCall>
         <command name="runit.cmd" retryCount="0" retryWait="0" successRC="0" />
       </postDestinationCall>
       <item checksumMethod="none" mode="binary">
⋮
```

您可以在傳送中混用不同類型的指令。 引數、目標及內容元素是選用的。

監視目錄及使用變數替代

您可以使用 **fteCreateMonitor** 指令來監視目錄。 替代變數的值可替換至作業 XML 定義中，並用來定義傳送行為。

關於這項作業

在此範例中，來源代理程式稱為 AGENT_HOP。 AGENT_HOP 所監視的目錄稱為 /test/monitored。 代理程式每隔 5 分鐘會輪詢目錄一次。

在 .zip 檔寫入至目錄後，將檔案寫入目錄的應用程式會在相同的目錄中撰寫一個觸發檔案。 觸發檔案的名稱與 .zip 檔相同，但副檔名不同。 例如，在 file1.zip 檔寫入至目錄後，即會在該目錄中撰寫 file1.go 檔。 資源監視器會監視該目錄中符合型樣 *.go 的檔案，然後使用變數替代要求傳送關聯的 .zip 檔。

程序

1. 建立作業 XML，以定義監視器被觸發後所執行的作業。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>blue.example.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP" />
     <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <file>/test/monitored/${fileName{token=1}{separator=.}}.zip</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/out/${fileName{token=1}{separator=.}}.zip</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

   取代為與觸發檔案相關聯值的變數，會以**粗體**強調顯示。 此作業 XML 會儲存至 /home/USER1/task.xml 檔。

2. 建立用來監視目錄 /test/monitored 的資源監視器。

   提交下列指令：

```
fteCreateMonitor -ma AGENT_HOP -mm QM_HOP -md /test/monitored
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr match,*.go -pi 5 -pu minutes
```

3. 使用者或程式會將 jump.zip 檔寫入 /test/monitored 目錄，然後再將 jump.go 檔寫入該目錄。
4. jump.go 檔的存在會觸發監視器。 代理程式會將觸發檔案的相關資訊代入作業 XML 中。

   這會導致作業 XML 轉換成：

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>blue.example.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP" />
     <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <file>/test/monitored/jump.zip</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/out/jump.zip</file>
         </destination>
       </item>
     </transferSet>
```

```
    </managedTransfer>
</request>
```

結果

執行作業 XML 所定義的傳送。 AGENT_HOP 會從 /test/monitored 目錄中讀取 jump.zip 檔，且該檔案會傳送至 AGENT_SKIP 執行所在系統上稱為 /out/jump.zip 的檔案。

相關概念

第 196 頁的『使用變數替代來自訂 MFT 資源監視器作業』

滿足作用中資源監視器的觸發條件時，會呼叫所定義的作業。 除了每次以相同的目的地代理程式或相同的目的地檔案名稱呼叫傳送或指令作業之外，您還可以在執行時期修改作業定義。 您可以將變數名稱插入作業定義 XML 中，來達成此目的。 當監視器判斷已滿足觸發條件，且作業定義包含變數名稱時，它會以變數值替換變數名稱，然後呼叫該作業。

相關工作

第 191 頁的『配置 MFT 監視作業以啟動指令及 Script』

資源監視器不限於將檔案傳送作為其關聯的作業執行。 您也可以配置監視器以從監視代理程式呼叫其他指令，包括可執行程式、Ant Script 或 JCL 工作。 若要呼叫指令，請編輯監視作業定義 XML，以併入具有對應指令呼叫參數的多個指令元素，如引數及內容。

相關參考

fteCreateMonitor：建立 MFT 資源監視器

範例：配置 MFT 資源

您可以搭配使用 **-mq** 參數與 **fteCreateMonitor** 指令，將 IBM MQ 佇列指定為資源監視器要監視的資源。

關於這項作業

在此範例中，要監視的資源是佇列 *MONITORED_QUEUE*。 此佇列必須位於監視代理程式的佇列管理程式 *QM_NEPTUNE* 上。 監視佇列的條件是有完整訊息群組存在。 符合條件時所要執行的作業，定義於 task.xml 檔中。

註: 請不要建立多個資源監視器來監視個別佇列。 這麼做可能會導致無法預期的行為。

程序

請鍵入下列指令：

```
fteCreateMonitor -ma AGENT_NEPTUNE -mn myMonitor -mm QM_NEPTUNE -mq MONITORED_QUEUE
  -mt task.xml -tr completeGroups -pi 5 -pu minutes
```

監視器每五分鐘會檢查佇列一次，以查看條件 completeGroups 是否成立。 如果佇列上有一個以上完整群組存在，監視器即會逐一針對各個完整群組執行 task.xml 檔中定義的作業。

使用變數替代來自訂 MFT 資源監視器作業

滿足作用中資源監視器的觸發條件時，會呼叫所定義的作業。 除了每次以相同的目的地代理程式或相同的目的地檔案名稱呼叫傳送或指令作業之外，您還可以在執行時期修改作業定義。 您可以將變數名稱插入作業定義 XML 中，來達成此目的。 當監視器判斷已滿足觸發條件，且作業定義包含變數名稱時，它會以變數值替換變數名稱，然後呼叫該作業。

小心: 變數名稱不區分大小寫。

用於替代的變數只可用於正數觸發條件。 只有 match 及 fileSize 觸發條件會替換變數。 如果使用 noMatch 條件，且作業定義中有替代變數名稱，則不會呼叫作業，且監視器會產生回覆碼 110 及錯誤訊息 BFGDM0060E。

如果受監視資源是佇列

在第一個要從受監視佇列讀取的訊息中，任何 IBM MQ 訊息內容的值都可代入作業 XML 定義中。

使用者定義訊息內容的字首為 `usr.`，但在變數名稱中不包括此字首。 變數名稱必須以大括弧 {} 括住，且前面必須是錢幣符號 ($) 字元。

例如，`${destFileName}` 取代為要從來源佇列中讀取的第一個訊息的 `usr.destFileName` 訊息內容的值。 如需相關資訊，請參閱 MFT 從來源佇列上的訊息讀取的 MQ 訊息內容 及 第 201 頁的『監視佇列及使用變數替代』。

如果變數未定義為訊息內容，監視器會報告 BFGDM0060E 錯誤並傳回回覆碼 110 (監視器作業變數替代失敗)。 此外，代理程式還會將下列錯誤訊息寫入其事件日誌 (`outputN.log`):

```
BFGDM0113W: Trigger failure for <monitor name> for reason BFGDM0060E: A monitor task could
not complete as a variable substitution <variable name> was not present.
```

如果針對監視器啟用中等或詳細資源監視器記載，則監視器會將下列訊息寫入代理程式的資源監視器事件日誌 (`resmoneventN.log`):

```
BFGDM0060E: A monitor task could not complete as a variable substitution
<variable name> was not present.
```

如需資源監視器記載的相關資訊，請參閱 第 205 頁的『記載 MFT 資源監視器』 。

下表顯示依預設會提供的替代變數。 例如，`${AGENTNAME}` 取代為資源監視器代理程式的名稱。

表 *10:* 依預設提供的替代變數	
變數	**說明**
AGENTNAME	資源監視器代理程式的名稱。
QUEUENAME	所監視的佇列名稱。
ENCODING	佇列上第一個訊息或群組中第一個訊息的字元編碼。
MESSAGEID	佇列上的第一個訊息或群組中的第一個訊息的 IBM MQ 訊息 ID。
GROUPID	群組的 IBM MQ 群組 ID，如果只找到一個訊息，則為訊息 ID。 只有在您是監視完整群組時，才需要設定此變數。
CurrentTimeStamp	以監視器觸發時的當地時間為基礎的時間戳記。 代理程式的時間戳記值是唯一的。
CurrentTimeStampUTC	以監視器觸發時的世界標準時間時區的時間為基礎的時間戳記。 代理程式的時間戳記值是唯一的。

如果受監視資源是目錄

下表顯示作業 XML 定義中可替換的變數名稱集。

表 *11:* 可替換的變數	
變數	**說明**
FilePath	觸發檔案的完整路徑名稱。
FileName	觸發程式的檔案名稱部分。
LastModifiedTime	觸發檔案前次修改時間。 此時間以代理程式執行所在之時區的當地時間表示，並且格式化為 ISO 8601 時間。
LastModifiedDate	觸發檔案前次修改日期。 此日期以代理程式執行所在之時區的當地日期表示，並且格式化為 ISO 8601 日期。
LastModifiedTimeUTC	觸發檔案前次修改時間。 此時間以轉換為世界標準時間時區的當地時間表示，並且格式化為 ISO 8601 時間

表 *11:* 可替換的變數 *(*繼續*)*	
變數	**說明**
LastModifiedDateUTC	觸發檔案前次修改日期。 此日期以轉換為世界標準時間時區的當地日期表示，並且格式化為 ISO 8601 日期。
AgentName	資源監視器代理程式的名稱。
CurrentTimeStamp	以監視器觸發時的當地時間為基礎的時間戳記。 代理程式的時間戳記值是唯一的。
CurrentTimeStampUTC	以監視器觸發時的世界標準時間時區的時間為基礎的時間戳記。 代理程式的時間戳記值是唯一的。

如果受監視資源是觸發檔案

下表顯示當資源監視器使用觸發檔案的內容來判斷需要傳送的檔案時，可以替代的變數名稱集。

表 *12:* 使用觸發檔案時可以替代的變數	
變數	**說明**
contentSource	原始檔的完整路徑名稱。
contentDestination	目的地檔案的完整路徑名稱。

變數名稱必須以錢幣符號 ($) 字元開頭，並以大括弧 {} 括住。例如， `${FilePath}` 會取代為相符觸發檔案的完整檔案路徑。

有兩個特殊關鍵字可套用至變數名稱，以進一步精準化。 它們是：

記號
要替換的記號索引（左邊從 1 開始，右邊從 -1 開始）

separator
將變數值記號化的單一字元。 預設值是 AIX and Linux 平台上的正斜線字元 (/)，或 Windows 平台上的反斜線字元 (\)，但分隔字元可以是任何可出現在變數值中的有效字元。

如果 separator 關鍵字指定在變數名稱中，變數名稱會根據分隔字元分割成多個記號。

指派給 token 關鍵字的值則用作索引，選取要用來取代變數名稱的記號。 記號索引相對於變數中的第一個字元，從 1 開始。 如果未指定記號關鍵字，則會插入整個變數。

在訊息 XML 中，代入代理程式名稱的任何值是以不區分大小寫的方式來處理。 所有 Managed File Transfer Agent 名稱都是大寫。 如果在訊息 XML 中，將 `Paris` 值代入代理程式屬性，此值會解譯為對代理程式 `PARIS` 的參照。

相關概念
第 198 頁的『範例: 資源監視器定義的變數替代』
使用 XML 和 IBM MQ Explorer 來替代資源監視器定義的變數範例。

相關工作
變數替代導致多個檔案傳送至單一檔名時應該如何處理

範例: 資源監視器定義的變數替代

使用 XML 和 IBM MQ Explorer 來替代資源監視器定義的變數範例。

顯示變數替代如何運作的範例

假設相符觸發程式檔案的檔案路徑在 Windows 上是 `c:\MONITOR\REPORTS\Paris\Report2009.doc`，在 AIX and Linux 平台上是 `/MONITOR/REPORTS/Paris/Report2009.doc`，則會替換變數，如下表所示。

表 *13:* 如何替換變數	
變數規格	**在變數替代之後**
${FilePath}	Windows :c:\MONITOR\REPORTS\Paris\Report2009.doc AIX and Linux :/MONITOR/REPORTS/Paris/Report2009.doc
${FilePath{token=1}{separator=.}}	Windows :c:\MONITOR\REPORTS\Paris\Report2009 AIX and Linux :/MONITOR/REPORTS/Paris/Report2009
${FilePath{token=2}{separator=.}}	Windows :doc AIX and Linux :doc
${FilePath{token=3}}	Windows 報告 AIX and Linux : 巴黎

您也可以指定負數記號索引，來選取相對於變數最後字元的記號（如下表中所示）。 表格中的範例使用相同的變數值， c:\MONITOR\REPORTS\Paris\Report2009.doc (在 Windows 上) 和 /MONITOR/REPORTS/Paris/Report2009.doc (在 AIX and Linux 上)。

表 *14:* 使用負數記號索引的範例	
變數規格	**在變數替代之後**
${FilePath}	Windows :c:\MONITOR\REPORTS\Paris\Report2009.doc AIX and Linux :/MONITOR/REPORTS/Paris/Report2009.doc
${FilePath{token=-2}{separator=.}}	Windows :c:\MONITOR\REPORTS\Paris\Report2009 AIX and Linux :/MONITOR/REPORTS/Paris/Report2009
${FilePath{token=-2}{separator=\}}	Windows : 巴黎 AIX and Linux : 巴黎
${FilePath{token=-4}}	Windows :MONITOR AIX and Linux :MONITOR

用於替代的變數僅適用於下列正觸發條件及 noSizeChange 選項，這是正觸發條件規則的例外情況:

- match
- fileSize
- noSize **變更**

如果使用 noMatch 條件，且作業定義中有替代變數名稱，則不會呼叫作業，且監視器會產生回覆碼 110 及錯誤訊息 BFGDM0060E。

使用 XML 的範例

下列範例作業定義 XML 使用監視器代理程式名稱作為傳送的來源代理程式 (Paris)，使用檔案路徑中倒數第二個目錄名稱作為傳送的目的地代理程式名稱 (Report2009)，並將傳送的檔案重新命名為觸發檔案名稱的根再加上副檔名 .rpt。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="${AgentName}" QMgr="QM1" />
     <destinationAgent agent="${FilePath{token=-2}}" QMgr="QMD" />
     <transferSet>
       <item mode="binary" checksumMethod="MD5">
         <source recursive="false" disposition="leave">
           <file>c:/incoming/reports/summary/report.doc</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/${FileName{token=1}{separator=.}}.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

這會導致作業 XML 轉換成：

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT1" QMgr="QM1" />
     <destinationAgent agent="Paris" QMgr="QMD" />
     <transferSet>
       <item mode="binary" checksumMethod="MD5">
         <source recursive="false" disposition="leave">
           <file>c:/incoming/reports/summary/report.doc</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/Report2009.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

<destinationAgent> 元素之 agent 屬性中的變數 ${FilePath{token=-2}} 會取代為 Paris 一值。此值會以不區分大小寫的方式來處理，並解譯為對代理程式 PARIS 的參照。

使用 IBM MQ Explorer 的範例

透過「IBM MQ Explorer」建立資源監視器時，一旦指定監視器內容及觸發條件，即可將傳送項目新增至監視器。 下列範例示範如何在 "**新增傳送項目畫面**" 中使用 ${FilePath} 和 ${FileName} 變數，來自訂資源監視器相符項所產生的傳送。

範例 1

若要在符合觸發條件時將來源檔案簡單地傳送至另一個位置，則可使用 ${FilePath} 變數：

- 將來源**檔名**設定為 ${FilePath}。
- 從目的地的**類型**下拉功能表中選取**目錄**。
- 將目的地**檔名**設定為您要將來源檔案傳送至其中的位置，例如，這可以是 C:\MFT\out\。

範例 2

若要將來源檔案傳送至另一個位置並變更檔案的副檔名，則可將 ${FileName} 變數與 ${FilePath} 變數一起使用：

在下列範例中，假設來源檔案的檔案路徑等於 C:\MONITOR\REPORTS\Paris\Report2009.doc：

- 將來源**檔名**設定為 ${FilePath}。
- 將目的地**檔名**設定為您要將來源檔案傳送至其中的位置，後接 ${FileName{token=1}{separator=.}}，再接檔案的新副檔名。 例如，這可以是 C:\MFT\out\${FileName{token=1}{separator=.}}.rpt，它將等於包含來源檔案名稱的 C:\MFT\out\Report2009.rpt。

範例 3

若要使用來源檔案的部分檔案路徑來決定傳送目的地，則可將 ${FilePath} 變數與記號及分隔字元規格一起使用。

在下列範例中，假設來源檔案的檔案路徑等於 C:\MONITOR\REPORTS\Paris\Report2009.doc。

您可以使用部分來源檔案路徑來決定檔案目的地。 使用檔案路徑範例 C:\MONITOR\REPORTS\Paris\Report2009.doc，如果要根據來源檔案的位置（即本範例中的 Paris）將檔案傳送至某個資料夾，則可執行下列動作：

- 將來源**檔名**設定為 ${FilePath}。
- 將目的地**檔名**設定為每個位置的資料夾所處目的地，然後附加檔案路徑的目的地部分及檔名。 例如，這可以是 C:\MFT\out\${FilePath{token=-2}{separator=\}}\${FileName}，它將等於包含來源檔案名稱的 C:\MFT\out\Paris\Report2009.doc。

相關概念

第 196 頁的『使用變數替代來自訂 MFT 資源監視器作業』
滿足作用中資源監視器的觸發條件時，會呼叫所定義的作業。 除了每次以相同的目的地代理程式或相同的目的地檔案名稱呼叫傳送或指令作業之外，您還可以在執行時期修改作業定義。 您可以將變數名稱插入作業定義 XML 中，來達成此目的。 當監視器判斷已滿足觸發條件，且作業定義包含變數名稱時，它會以變數值替換變數名稱，然後呼叫該作業。

相關工作

變數替代導致多個檔案傳送至單一檔名時應該如何處理

監視佇列及使用變數替代

您可以使用 **fteCreateMonitor** 指令來監視佇列，並將訊息從受監視佇列傳送至檔案。 在第一個要從受監視佇列讀取的訊息中，任何 IBM MQ 訊息內容的值都可代入作業 XML 定義中，並用來定義傳送行為。

關於這項作業

在此範例中，來源代理程式稱為 AGENT_VENUS，其連接至 QM_VENUS。 AGENT_VENUS 所監視的佇列稱為 START_QUEUE，位於 QM_VENUS 上。 代理程式每隔 30 分鐘會輪詢佇列一次。

當完整訊息群組寫入佇列時，監視器作業即會將該訊息群組傳送至眾多目的地代理程式之一（全部都連接至佇列管理程式 QM_MARS）的檔案上。 訊息群組傳送目的地的檔案名稱，由群組中第一個訊息上的 IBM MQ 訊息內容 usr.fileName 所定義。 訊息群組傳送目的地的代理程式名稱，由群組中第一個訊息上的 IBM MQ 訊息內容 usr.toAgent 所定義。 如果未設定 usr.toAgent 標頭，目的地代理程式將會使用預設值 AGENT_MAGENTA。

當您指定 useGroups="true"時，如果未同時指定 groupId="${GROUPID}"，則傳送只會取得佇列上的第一則訊息。 比方說，例如您使用變數替代產生 fileName，a.txt 的內容因此有可能不正確。 這是因為 fileName 是由監視器所產生，但傳送實際上取用的並不是應產生 fileName 檔案的訊息。

程序

1. 建立作業 XML，以定義監視器被觸發後所執行的作業。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
     <destinationAgent agent="${toAgent}" QMgr="QM_MARS" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/${fileName}.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

取代為 IBM MQ 訊息標頭值的變數，會以**粗體**強調顯示。 此作業 XML 會儲存至 /home/USER1/task.xml 檔。

2. 建立用來監視佇列 START_QUEUE 的資源監視器。
   提交下列指令：

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. 使用者或程式將訊息群組寫入佇列 START_QUEUE。
   此群組中的第一個訊息具有下列 IBM MQ 訊息內容集：

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. 在寫入完整群組時，會觸發監視器。 代理程式將 IBM MQ 訊息內容代入作業 XML 中。
   這會導致作業 XML 轉換成：

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
     <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/larmer.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

結果

即會執行作業 XML 所定義的傳送。 AGENT_VENUS 從 START_QUEUE 讀取的完整訊息群組，會寫入 AGENT_VIOLET 執行時所在系統上稱為 /reports/larmer.rpt 的檔案。

下一步

將每一個訊息傳送至個別檔案

如果要監視佇列並讓每則訊息傳送至個別檔案，您可以使用與本主題中先前所述方式相似的技術。

1. 依照先前說明來建立監視器，並在 **fteCreateMonitor** 指令上指定 **-tr completeGroups** 參數。
2. 在作業 XML 中指定下列內容：

```
<queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
```

不過，當您將訊息放入來源佇列時，請不要將它們放置在 IBM MQ 群組中。 為每一個訊息新增 IBM MQ 訊息內容。 例如，您可以為每一個訊息指定具有唯一檔案名稱值的 usr.filename 內容。 這實際上會讓 Managed File Transfer Agent 將來源佇列上的每一則訊息都視為個別群組。

配置訊息轉為檔案傳送的監視器重試行為

如果資源監視器所觸發的「訊息轉為檔案」傳送失敗，而將觸發監視器的訊息群組保留在佇列上，該傳送將會根據後續的輪詢間隔進行重新提交。 重新提交傳送的次數受限於監視代理程式的 **monitorGroupRetryLimit** 內容。

關於這項作業

每次觸發新的「訊息轉為檔案」傳送時，都會為傳送作業產生新的傳送 ID。

如果代理程式重新啟動，則即使已觸發傳送的次數已超出 agent.properties 檔案中的 **monitorGroupRetryLimit** 值，監視器仍會再次觸發傳送。 **monitorGroupRetryLimit** 內容的值，是監視器在訊息群組仍存在於佇列上的情況下，再次觸發「訊息轉為檔案」傳送的次數上限。 此內容的預設值是 10。 此內容值可設為任何正整數值或 -1。 如果將此內容值指定為 -1，監視器就會不限次數地重新觸發傳送，直到不符合觸發條件為止。

如果傳送嘗試導致已觸發傳送的次數超出 **monitorGroupRetryLimit** 值，則代理程式會將錯誤寫入其事件日誌。

單一訊息會被視為單一群組，且只要訊息仍在佇列上，或是傳送的已觸發次數小於 **monitorGroupRetryLimit** 的值，就會在每一個輪詢間隔重新觸發傳送。

若要設定監視代理程式的 **monitorGroupRetryLimit** 內容，請執行下列步驟：

程序

1. 使用 **fteStopAgent** 指令停止監視代理程式。
2. 編輯監視代理程式的 agent.properties 檔案，以包含下列行：

```
monitorGroupRetryLimit=number_of_retries
```

   agent.properties 檔案位於 *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*monitoring_agent_name* 目錄中。
3. 使用 **fteStartAgent** 指令啟動監視代理程式。

相關工作

第 196 頁的『範例：配置 MFT 資源』
您可以搭配使用 **-mq** 參數與 **fteCreateMonitor** 指令，將 IBM MQ 佇列指定為資源監視器要監視的資源。

使用觸發檔

您可以在資源監視器中使用觸發檔案的內容，定義要在單一傳送要求中傳送的檔案集。 每次偵測到相符的觸發檔案時，系統都會剖析其內容，以取得來源檔案路徑，並選擇性地取得目的地檔案路徑。 然後，這些檔案路徑會用來定義您指定的作業傳送 XML 檔（以單一傳送要求形式提交至代理程式）中的檔案項目。 資源監視器的定義會決定是否啟用觸發內容。

您可以在建立監視器時指定 **-tc**（觸發內容）參數，以啟用檔案內容觸發。 此 **-tc** 參數僅適用於檔案觸發選項 match 及 noSizeChange。 如需建立監視器的相關資訊，請參閱 **fteCreateMonitor**: 建立 MFT 資源監視器。

使用觸發程式內容檔時，每一行的預設格式為:

- 單一原始檔路徑，或
- 來源檔案路徑和目的地檔案路徑，以逗點區隔

其中空格字元是作為檔案路徑的一部分來處理。 您可以在 **fteCreateMonitor** 指令上指定 **-tcr** 和 **-tcc** 參數，來變更預設行格式。 如需相關資訊，請參閱第 204 頁的『進階選項』。

剖析觸發檔案後，即會產生檔案路徑清單，並將其套用至您所指定的傳送作業 XML。 如同所有監視器，傳送作業 XML 的格式是以 **fteCreateTransfer** 指令產生、並定義了單一項目或檔案的完整傳送作業 XML。 單一項目必須使用替代變數 ${contentSource}（並選擇性地使用 ${contentDestination}）來取代來源及目的地檔案路徑。 監視器會展開傳送作業 XML，以在觸發檔案中併入每一行（檔案路徑）的檔案項目。

-bs 參數無法用在檔案內容觸發上，因為 **-tc** 參數表示每一個觸發檔案各有一個傳送要求。

範例

下列範例所定義的監視器會觸發以 trig 結尾的檔案，並讀取該檔案中的檔案路徑。

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -dd /file/destdir ${contentSource}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
```

fteCreateTransfer 指令會針對原始檔路徑為 ${contentSource}的單一檔案建立一個稱為 task.xml 的檔案。 例如:

```
<item checksumMethod="MD5" mode="binary">
    <source disposition="leave" recursive="false">
          <file>${contentSource}</file>
    </source>
</item>
```

fteCreateMonitor 指令會在 /home/trigdir 目錄中掃描以 trig 結尾的檔案，並使用這些內容，為該觸發檔案中的所有路徑建立以 task.xml 為基礎的單一傳送要求。 觸發檔案的格式必須是一行顯示一個檔案路徑（僅限來源）、且不使用逗點分隔字元。 例如:

```
/home/file/first.txt
/home/file/second.txt
/home/different/third.txt
⋮
```

所有檔案都會遞送至具有其檔名（而非其檔案路徑）的 /file/destdir 目錄中，亦即，/home/file/first.txt 會遞送至 /file/destdir/first.txt。

或者，如果您將 **fteCreateTransfer** 指令中的 **-dd /file/destdir** 參數變更為 -df ${contentDestination}，並將觸發檔案的內容格式變更為 *source file path,destination file path*，則可以為相同目的地代理程式定義不同的目的地路徑。 例如:

```
/home/file/first.txt,/home/other/sixth.txt
```

然後，目的地位置會變成 /home/other/sixth.txt。

替代變數可以記號化。 例如，您可以使用 ${ contentDestination{token=-1 }} 將檔案名稱部分與提供的路徑分開。 因此，如果 **fteCreateTransfer** 目的地定義為 -df /file/destdir/${contentDestination{token=-1}}，則 /home/file/first.txt 的新目的地為 /file/destdir/sixth.txt。

進階選項

您可以使用 **-tcr** *regex* 參數來變更觸發檔案內容的預設行格式。 請提供符合必要行格式、並可提供一個或是兩個擷取群組的正規表示式。 第一個擷取群組是來源，第二個擷取群組（選用）是目的地。 例如:

- 來源與目的地路徑會以連字號區隔:

```
((?:[^-])+)-((?:[^-])+)
```

  在此範例中，分隔字元定義於三個位置，而連字號 - 的三個實例全都可變更為任何其他字元。 請確實跳出任何特殊字元。
- 來源與目的地路徑會以含有尾端空格的逗點區隔。 以 # 記號表示的註解會被忽略。

```
((?:[^,])+),((?:[^,])+) *(?:#.*)+
```

  檔案路徑不能包含 # 記號。 一般而言，項目如下: `/home/source/from.txt,/home/destination/to.txt # some comment`。

如果您使用 **-tcr** 參數，請確定正規表示式已經過妥善設計及測試，因此表示式能夠偵測出錯誤並正確剖析觸發檔案。

您可以使用 **-tcc destSrc** 參數將擷取順序反轉。 如果您指定此參數，則第一個擷取群組會是目的地檔案路徑，第二個群組會是來源檔案路徑。

錯誤的處理方式

空的觸發檔案
如果觸發檔案是空的，則不會執行檔案傳送。 亦即，監視器會建立傳送要求，但不會指定檔案項目。

觸發檔案出現錯誤
如果觸發檔案中有任何項目無法根據預期的格式進行剖析，則不會產生傳送要求。 此時會發佈監視器錯誤日誌，同時將錯誤記載到事件日誌中。 觸發檔案會標示為已處理，且監視器在檔案更新之前都不會再嘗試處理檔案。

傳送作業 XML 不符
傳送作業 XML 必須符合觸發檔案，亦即，如果傳送作業 XML 同時具有 ${contentSource} 和 ${contentDestination}，則該監視器的所有觸發檔案都必須具有來源及目的地檔案路徑，反之亦然。 在第一種情況下，如果觸發檔案僅提供來源檔案路徑，則監視器會報告 ${contentDestination} 替代失敗。

範例

下列範例說明觸發檔案內容僅含有來源檔案路徑的基本內容觸發:

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -dd /file/destdir ${contentSource}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
```

-tcr 參數會定義兩個由任意字元序列組成的擷取群組，這些字元以空格字元區隔。 **-tcc destSrc** 參數及選項會指出要以目的地在前、來源在後的格式處理擷取群組。

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -df ${contentDestination} $
{contentSource}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
    -tcr "((?:[^ ])+) ((?:[^ ])+)" -tcc destSrc
```

記載 MFT 資源監視器

您可以使用記載來取得資源監視器的診斷資訊。

關於這項作業

您可以使用 **fteSetAgentLoglevel** 指令或 `agent.properties` 檔案來控制資源監視器記載，以將記載用於資源監視器。

請注意，現有追蹤點仍將用於擷取資訊。

資源監視器日誌會寫入名為 `resmonevent`*N*`.log` 的檔案，其中 *N* 代表數字; 例如 `resmonevent0.log`。事件日誌檔會記錄監視器輪詢資源 (例如目錄或佇列) 時發生的數個動作。

小心: 一個代理程式的所有資源監視器都會寫入相同的日誌檔。

如需 `resmonevent`*N*`.log` 檔案的部分範例輸出，請參閱 如果 MFT 目錄資源監視器未觸發檔案，怎麼辦。

下表列出了資源監視器寫入日誌檔的事件類型。 第三個直欄說明擷取每一個事件所需的記載層次，其中最低層次是 INFO ，而最高層次是 VERBOSE。

請注意，設定較高的記載層次，也會寫入較低層次的事件。 例如，將記載層次設為 MODERATE 也會寫入 INFO 層次事件，但不會寫入 VERBOSE 層次事件。

號碼	事件	記載層次	說明
1	已建立監視器	INFO	已建立資源監視器。
2	已刪除監視器	INFO	已刪除資源監視器。
3	監視器已停止	INFO	已停止資源監視器。
4	監視器已啟動	INFO	已啟動資源監視器。
5	開始輪詢	INFO	資源監視器已開始新的輪詢週期。
6	結束輪詢	INFO	資源監視器輪詢週期已結束。
7	型樣相符	VERBOSE	在觸發監視器目錄上找到與所指定型樣相符的檔案，或者在佇列中找到與所指定型樣相符的訊息。
8	型樣不符	VERBOSE	在觸發監視器目錄上找到不相符的檔案，或者在佇列中找到與所指定型樣不相符的訊息。
9	傳送要求	INFO	資源監視器已起始傳送。
10	目錄太深	VERBOSE	資源監視器所監視的目錄包含更多要輪詢的子目錄，較資源監視器配置中指定的數目大。
11	檔案已鎖定	MODERATE	資源監視器所監視的觸發檔已由其他程序鎖定。
12	檔案太小	MODERATE	觸發檔小於資源監視器配置中指定的大小。
13	檔案大小不穩定	MODERATE	變更觸發檔的頻率超出資源監視器配置的預期。
14	輪詢次數太多	MODERATE	資源監視器輪詢不穩定觸發檔的次數太多。
15	相符的項目數	INFO	在資源監視器所輪詢的目錄中找到的觸發檔總數。
16	傳送項目	INFO	傳送要求中的項目總數。
17	已產生 FDC	MODERATE	資源監視器已產生異常狀況。
18	傳送要求	INFO	由資源監視器提交的傳送要求。
19	啟動監視器失敗	MODERATE	資源監視器啟動失敗。
20	已清除歷程	INFO	已清除監視歷程資訊。
21	清除監視器歷程失敗	INFO	嘗試清除監視歷程資訊失敗。
22	傳送 ID	INFO	監視器已提交傳送要求的 ID。
23	批次處理	INFO	相符項目的傳送要求總數: *N*，其中 *N* 是數字。

程序

- 若要使用 **fteSetAgentLoglevel** 來開啟及關閉資源監視器記載，請參閱 fteSetAgentLogLevel 以取得 **logMonitor** 參數的說明，及不同選項之用法的範例。
- 若要使用 agent.properties 檔案來控制資源監視器記載，請參閱 MFT agent.properties 檔案，以取得可讓您執行下列記載活動之其他內容的說明:
  - 開啟或關閉記載功能
  - 限制每個日誌檔的大小
  - 限制資源監視器可以產生的日誌數

範例

下列訊息範例會在佇列管理程式 MFTDEMO 上，為代理程式 HA2 設定 verbose 層次記載功能:

```
<?xml version="1.0"?>
<log:log version="6.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xmlns:log="https://www.ibm.com/log">
    <log:originator>
      <log:request>
        <log:hostName>192.168.7.1</log:hostName>
        <log:userID>johndoe</log:userID>
      </log:request>
    </log:originator>
    <log:endpoint agent="HA2" QMgr="MFTDEMO"/>
    <log:logMonitor>MON1="verbose"</log:logMonitor>
</log:log>
```

相關參考

fteSetAgentLogLevel 指令
MFT agent.properties 檔案

V 9.2.2 啟動 MFT 資源監視器

從 IBM MQ 9.2.2 開始，您可以使用 **fteStartMonitor** 指令來啟動資源監視器，而不需要停止或重新啟動代理程式。

開始之前

如果透過將 agent.properties 檔案中的 **authorityChecking** 屬性設為 true 已啟用使用者權限管理，則您必須具有 **監視** 或 **監視作業** 權限才能啟動資源監視器。 如需使用者權限管理的相關資訊，請參閱限制 MFT 代理程式動作的使用者權限。

關於這項作業

您可以從已安裝 Managed File Transfer 指令元件的任何系統執行 **fteStartMonitor** 指令，這表示您可以從任何位置啟動資源監視器，而且不限於擁有資源監視器的代理程式執行所在的系統。 如需此指令的必要及選用參數的相關資訊，請參閱 fteStartMonitor (啟動 MFT 資源監視器)。

程序

- 若要在執行 **fteStartMonitor** 指令之前或之後找出代理程式的狀態，請搭配使用 **fteListMonitors** 指令與 **-v** 參數，如下列範例所示:

  ```
  fteListMonitors -ma monitoring_agent_name -v
  ```

- 若要在相同機器上執行的代理程式中啟動資源監視器，請輸入 **fteStartMonitor** 指令，如下所示:

  ```
  fteStartMonitor -mn monitor_name -ma agent_name
  ```

- 若要在不同機器上執行的代理程式中啟動資源監視器，請輸入 **fteStartMonitor** 指令，如下所示:

```
fteStartMonitor -mn monitor_name -ma agent_name -mm AgentQueueManager
```

如果指令佇列管理程式也是監視代理程式的代理程式佇列管理程式，則 **-mm** 參數是選用的，否則您必須使用 **-mm** 參數來指定代理程式佇列管理程式。

結果

如果代理程式正在執行中，則會啟動資源監視器 (如果目前已停止)。 該指令會輸出下列訊息，並在代理程式的 output0.log 中記載事件。

```
BFGCL0816I: 已發出啟動代理程式 'agent_name' 的資源監視器 'monitor_name' 的要求。
BFGCL0251I: 要求已順利完成。
```

如需指令無法啟動資源監視器時所輸出訊息的相關資訊，請參閱 fteStartMonitor (啟動 MFT 資源監視器)。

相關概念

第 187 頁的『MFT 資源監視概念』
Managed File Transfer 資源監視特性的主要概念概觀。

相關工作

第 208 頁的『停止 MFT 資源監視器』
從 IBM MQ 9.2.2 開始，您可以停止資源監視器，而不需要使用 **fteStopMonitor** 指令停止或重新啟動代理程式。

相關參考

fteStart 監視器 (啟動 MFT 資源監視器)

V 9.2.2 停止 MFT 資源監視器

從 IBM MQ 9.2.2 開始，您可以停止資源監視器，而不需要使用 **fteStopMonitor** 指令停止或重新啟動代理程式。

開始之前

如果透過將 agent.properties 檔案中的 **authorityChecking** 屬性設為 true 已啟用使用者權限管理，則您必須具有 **監視器** 或 **監視器作業** 權限才能停止資源監視器。 如需使用者權限管理的相關資訊，請參閱 限制 MFT 代理程式動作的使用者權限。

關於這項作業

您可以從已安裝 Managed File Transfer 指令元件的任何系統執行 **fteStopMonitor** 指令，這表示您可以從任何位置停止資源監視器，而且不限於擁有資源監視器的代理程式執行所在的系統。 如需此指令的必要及選用參數的相關資訊，請參閱 fteStopMonitor (停止 MFT 資源監視器)。

當資源監視器停止時，它會將訊息寫入代理程式的資源監視器事件日誌 resmoneventnumber.log。 如果使用 **fteStopMonitor** 指令停止資源監視器，則訊息包括發出停止要求的使用者名稱:

```
使用者 '<mquser_id>' 已停止資源監視器
```

如果資源監視器的代理程式重新啟動，則會自動啟動資源監視器，即使先前已使用 **fteStopMonitor** 指令停止資源監視器也一樣。

代理程式會循序而非平行處理停止監視要求，因此，例如，如果代理程式收到停止監視 M1 的要求，然後又收到快速連續停止監視 M2 的另一個要求，則會先停止 M1 ，然後再嘗試停止 M2。

程序

- 若要在執行 **fteStopMonitor** 指令之前或之後找出代理程式的狀態，請搭配使用 **fteListMonitors** 指令與 **-v** 參數，如下列範例所示:

  ```
  fteListMonitors -ma monitoring_agent_name -v
  ```

- 若要在相同機器上執行的代理程式中停止資源監視器，請輸入 **fteStopMonitor** 指令，如下所示:

```
fteStopMonitor -mn monitor_name -ma agent_name
```

- 若要在不同機器上執行的代理程式中停止資源監視器，請輸入 **fteStopMonitor** 指令，如下所示:

```
fteStopMonitor -mn monitor_name -ma agent_name -mm AgentQueueManager
```

如果指令佇列管理程式也是監視代理程式的代理程式佇列管理程式，則 **-mm** 參數是選用的，否則您必須使用 **-mm** 參數來指定代理程式佇列管理程式。

結果

如果代理程式正在執行中，則資源監視器會停止 (如果目前已啟動)。 該指令會輸出下列訊息，並在代理程式的 output0.log 中記載事件。

```
BFGCL0813I: 已發出停止代理程式 'SOURCE' 之資源監視器 'MNTR' 的要求。
BFGCL0251I: 要求已順利完成。
```

如需指令無法停止資源監視器時所輸出訊息的相關資訊，請參閱 fteStopMonitor (停止 MFT 資源監視器)

相關概念

第 187 頁的『MFT 資源監視概念』
Managed File Transfer 資源監視特性的主要概念概觀。

相關工作

第 207 頁的『啟動 MFT 資源監視器』
從 IBM MQ 9.2.2 開始，您可以使用 **fteStartMonitor** 指令來啟動資源監視器，而不需要停止或重新啟動代理程式。

相關參考

fteStop 監視器 (停止 MFT 資源監視器)

備份及還原 MFT 資源監視器

您可以透過下列方式來備份要供未來使用的資源監視器：將其定義匯出至 XML 檔，隨後即可匯入該檔案以從備份建立新的資源監視器。

關於這項作業

您可能需要備份先前已經定義的資源監視器，以便未來可重複使用其定義，例如，在不同基礎架構中或在資源監視器因佇列管理程式問題而需要重建時重建資源監視器。

使用 **fteCreateMonitor** 指令或包含 **-ox** 參數的 **fteListMonitors** 指令，即可備份單一資源管理程式定義。 在這兩種情況下，皆可透過將資源管理程式定義匯出至 XML 檔來備份資源管理程式定義。 然後即可使用 **fteCreateMonitor** 指令的 **-ix** 參數，從 XML 檔匯入定義以建立新的資源管理程式。

使用 **-ox** 參數時，一次只能備份一個資源監視器定義。

從 IBM MQ 9.1 開始，**-od** 參數已新增至 **fteListMonitors** 指令。 透過指定此參數，即可將多個資源監視器定義成批匯出至指定目錄，以便一次備份多個資源監視器。 每一個資源監視器定義都會儲存至個別 XML 檔案，其名稱格式為 *agent name*.*monitor name*.xml。

如果您具有大量要備份的資源監視器，則 **-od** 參數特別有用，因為您僅需執行 **fteListMonitors -od** 指令一次，而不必針對每個資源定義個別地執行 **fteListMonitors -ox** 指令，亦不必針對每個資源監視器使用個別 Script 來執行 **fteListMonitors -ox** 指令。

程序

- 若要將一個資源監視器的定義匯出至 XML 檔以對其進行備份，請使用下列任一指令:
  - 包含 **-ox** 參數的 **fteCreateMonitor** 指令。
  - 包含 **-ox** 參數的 **fteListMonitors** 指令。

使用 **-ox** 參數時，您還必須指定 **-ma** 及 **-mn** 參數，如下列範例中所示：

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -ox filename1.xml
```

- 若要透過將多個資源監視器定義匯出至指定目錄中的 XML 檔案來備份它們，請搭配使用 **fteListMonitors** 指令與 **-od** 參數，如下列範例所示：

```
fteListMonitors -od /usr/mft/resmonbackup
```

成批備份資源監視器時，您必須指定有效的目標目錄。 不指定目標路徑會導致下列範例中所示的錯誤訊息：

```
BFGCL0762E: Output directory not specified. 請指定有效的路徑來重新執行指令。
```

-od 參數不得與 **-ox** 參數結合使用，否則會顯示下列錯誤訊息：

```
BFGCL0761E: It is not valid to specify both the '-od' and '-ox' parameters together.
```

您可以定義要包含在備份中的一組特定資源監視器。 例如，透過使用 **-ma** 參數來指定代理程式名稱，即可備份該代理程式的所有資源監視器，如下列範例中所示：

```
fteListMonitors -ma AGENT1 -od /usr/mft/resmonbackup
```

您亦可透過在定義用於比對代理程式名稱及/或監視器名稱的型樣時納入星號字元 (*) 來使用萬用字元比對。 下列範例會備份名稱符合指定型樣，且所處代理程式名稱符合指定型樣的所有資源監視器：

```
fteListMonitors -ma AGENT* -mn MON* -od /usr/mft/resmonbackup
```

當此指令正在執行時，它會顯示下列進度報告訊息：

```
A total of number matching resource monitor definitions found.
index of number resource monitor definitions saved to file system.
```

如果您使用的是詳細選項，則仍會顯示執行中總計，但並非顯示

```
index of number resource monitor definitions saved to file system
```

此指令會顯示正在儲存的監視器定義的名稱，例如：

```
BFGCL0762I: 代理程式 'XFERAGENT' 的監視器 'FILEMON' 定義已儲存為 FILEMON.XFERAGENT.XML 至檔案系統。
```

- 若要透過將特定代理程式匯出至指定目錄中的 XML 檔案，來備份特定代理程式的一個資源監視器，請搭配使用 **fteListMonitors** 指令與 **-od** 參數：

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -od /usr/mft/resmonbackup
```

除了輸出檔名的格式為 *agent name.monitor name*.xml 以外，使用 **-od** 參數來備份單一資源監視器，類似於使用 **-ox** 參數。

- 若要從備份還原資源監視器定義，請使用包含 **-ix** 參數的 **fteCreateMonitor** 指令，如下列範例中所示：

```
fteCreateMonitor -ix file name
```

如需如何使用 **-od** 參數的其他範例，請參閱 fteListMonitors: list MFT resource monitors。

相關參考

fteCreateMonitor：建立 MFT 資源監視器

fteListMonitors：列出 MFT 資源監視器

V 9.2.0 正在清除資源監視器歷程

您可以清除資源監視器的歷程，以便可以針對由於失敗而先前未傳送的檔案提交另一個檔案傳送要求。 若要清除資源監視器歷程，您可以使用 **fteClearMonitorHistory** 指令或 IBM MQ Explorer。

開始之前

如果已透過將 agent.properties 檔案中的 **authorityChecking** 屬性設為 true 來啟用使用者權限管理，則清除監視歷程的使用者必須具有適當的權限，如下表所示。

表 15: 執行 ***fteClearMonitorHistory*** 指令所需的使用者權限		
使用者清除監視器歷程	**MFT 存取權**	**必要權限**
與建立資源監視器的使用者相同的使用者。	監視器	BROWSE on SYSTEM.FTE.AUTHMON1.<*monitor_agent_name*> 此權限與建立或刪除資源監視器所需的權限相同。
建立資源監視器之使用者以外的任何使用者。	監視作業	SET on SYSTEM.FTE.AUTHPS1.<*agent_name*> 此權限與刪除資源監視器所需的權限相同。

如需使用者權限管理的相關資訊，請參閱 限制 MFT 代理程式動作的使用者權限。

如果沒有必要權限的使用者嘗試清除資源監視器歷程，則 **fteClearMonitorHistory** 指令會輸出錯誤訊息，並在代理程式的 `output0.log` 檔案中記載失敗。 如需相關資訊，請參閱 fteClearMonitorHistory: 清除資源監視器歷程。

關於這項作業

如果檔案傳送已起始，且由於任何原因而無法傳送檔案，則資源監視器不會在下一次輪詢中再次選取此檔案進行傳送，因為監視器歷程指出在先前的輪詢中看到檔案，且自那時以來未修改過該檔案 (請參閱 第 187 頁的『MFT 資源監視概念』)。

在 IBM MQ 9.1.3 之前，如果檔案無法傳送，則只能在下列情況下再次起始檔案傳送: 刪除檔案然後再次放置在目錄中，或更新檔案以變更前次修改日期屬性，或重建資源監視器本身。

不過，從 IBM MQ 9.1.3，您可以使用 **fteClearMonitorHistory** 指令或使用 IBM MQ Explorer 來清除資源監視器歷程。 清除歷程容許提交無法傳送之檔案的另一個傳送要求，而不需要刪除檔案，然後重新放入目錄中，或更新檔案以變更其前次修改日期屬性 (例如，在需要傳送檔案但無法修改檔案的情況下)。 能夠清除資源監視器的歷程也表示不需要重建資源監視器，就可以針對無法傳送的檔案提交另一個傳送要求。

z/OS z/OS 上 Managed File Transfer 隨附的範例 V 9.2.0 SCSQFCMD 成員包含 JCL Script，可清除監視器的歷程。

程序

- 若要使用 **fteClearMonitorHistory** 指令來清除資源監視器歷程，請以下列格式輸入指令:

  ```
  fteClearMonitorHistory -p <configuration> -ma <agent name> -mn <monitor name> -w 1000
  ```

  只需要 **-ma** 和 **-mn** 參數。 所有其他參數都是選用參數。 如需如何使用 **fteClearMonitorHistory** 指令的相關資訊 (包括範例)，請參閱 fteClearMonitorHistory: 清除資源監視器歷程。

  如果順利清除歷程，指令會輸出下列訊息:

  ```
  BFGCL0780I: 已發出要求清除代理程式 'agent name' 之資源監視器 'monitor name' 的歷程。
  BFGCL0251I: 要求已順利完成。
  ```

  並在代理程式的 `output0.log` 檔案中記載成功。

  如果嘗試清除資源監視器歷程失敗， **fteClearMonitorHistory** 會輸出錯誤訊息，並將失敗記錄在代理程式的 `output0.log` 檔案中。
- 若要使用 IBM MQ Explorer MFT 外掛程式中的資源監視器視圖來清除資源監視器歷程，請在資源監視器上按一下滑鼠右鍵，然後從下拉功能表中選取 **清除歷程** 。

  如果順利清除歷程，則會顯示下列訊息:

  ```
  BFGUI00171: 已順利清除資源監視器歷程。
  ```

如果嘗試清除歷程失敗，則會顯示錯誤訊息。 例如：

```
BFGUI0016E 無法清除指定資源監視器的歷程-原因 2059
```

使用檔案傳送範本

您可以使用檔案傳送範本來儲存一般檔案傳送設定，以進行重複或複式傳送。 請使用 **fteCreateTemplate** 指令，從指令行建立傳送範本，或者使用 IBM MQ Explorer，利用**為受管理檔案傳送建立新範本**精靈建立傳送範本，或選取**將傳送設定儲存為範本**勾選框，在建立檔案傳送時儲存範本。 **傳送範本**視窗會顯示您在 Managed File Transfer 網路中建立的所有傳送範本。

關於這項作業

若要從指令行建立傳送範本，請使用 fteCreateTemplate 指令。 然後，當您想要提交您在指令行上建立的傳送範本時，請按一下 IBM MQ Explorer 中的 **提交** 。

若要在 IBM MQ Explorer 中檢視傳送範本，請使用下列步驟：

程序

1. 在「導覽器」視圖中展開 **Managed File Transfer**。 **Managed File Transfer 中心**即會顯示在「內容」視圖中。
2. 所有協調佇列管理程式都會列在「導覽器」視圖中。 展開您用於排定傳送的協調佇列管理程式的名稱。 如果您想要變更所連接的協調佇列管理程式，請在「導覽器」視圖中您想要使用的協調佇列管理程式名稱上按一下滑鼠右鍵，然後再按一下**連接**。
3. 按一下**傳送範本**。 「**傳送範本**」視窗即會顯示在「內容」視圖中。
4. 「**傳送範本**」視窗列出關於檔案傳送的下列詳細資料：
   a) **名稱** 檔案傳送範本的名稱。
   b) **來源** 用來從來源系統傳送檔案的代理程式名稱。
   c) **來源檔案** 要在其主機系統上傳送的檔案名稱。
   展開傳送範本資訊，以檢視此欄位。
   d) **目的地** 用來在目的地系統上接收檔案的代理程式名稱。
   e) **目的地檔案** 檔案傳送至目的地系統之後的檔案名稱。
   展開傳送範本資訊，以檢視此欄位。
   f) **排定的開始時間（選取的時區）** 排定檔案傳送開始的時間和日期，以管理者使用的時區表示。 若要變更顯示的時區，請按一下**視窗** > **喜好設定** > **IBM MQ Explorer** > **Managed File Transfer**，然後從**時區：** 清單中選取替代時區。 按一下**確定**。
   g) **觸發事件** 觸發檔案傳送開始的事件類型。 類型可以是下列其中一值：**存在**、**不存在**或**超出**。

結果

若要重新整理**傳送範本**視窗所顯示的內容，請按一下「內容」視圖工具列上的「重新整理」按鈕 。

若要提交傳送範本並開始在範本中定義的傳送，請在範本名稱上按一下滑鼠右鍵，然後按一下**提交**。

若要變更傳送範本，請在範本名稱上按一下滑鼠右鍵，然後按一下**編輯**。 原始範本中包括的所有檔案都會列為傳送群組的一部分，即使它們並未併入為原始範本的群組的一部分。 如果您想要移除範本中的某個檔案，必須從群組中選取檔案規格，然後按一下**移除已選取的項目**。 如果您想要將檔案規格新增至範本中，請使用範本畫面中的欄位，然後按一下**新增至群組**按鈕。 完成編輯時，系統會提示您為編輯的範本指定新名稱。

若要從傳送範本中建立檔案傳送，請在範本名稱上按一下滑鼠右鍵，然後按一下**編輯為新建傳送**。

若要建立傳送範本的副本，請在範本名稱上按一下滑鼠右鍵，然後按一下**複製**。 複製的傳送範本會使用與原始範本相同的名稱自動儲存，並加上 "(copy)" 字樣。

若要刪除傳送範本，請在範本名稱上按一下滑鼠右鍵，然後按一下**刪除**。

相關工作

第 213 頁的『使用 IBM MQ Explorer 建立檔案傳送範本』

您可以從「IBM MQ Explorer」或從指令行建立檔案傳送範本。 然後，您可以使用該範本，利用範本詳細資料建立新的檔案傳送，或提交範本以啟動檔案傳送。

相關參考

fteCreateTemplate：建立新的檔案傳送範本

fteListTemplates

fteDeleteTemplates

使用 IBM MQ Explorer 建立檔案傳送範本

您可以從「IBM MQ Explorer」或從指令行建立檔案傳送範本。 然後，您可以使用該範本，利用範本詳細資料建立新的檔案傳送，或提交範本以啟動檔案傳送。

關於這項作業

若要從指令行建立檔案傳送範本，請使用 fteCreateTemplate 指令。

若要使用 IBM MQ Explorer 中的 **建立受管理檔案傳送的新範本** 精靈來建立檔案傳送範本，請使用下列步驟:

程序

1. 在「導覽器」視圖中，按一下 **Managed File Transfer**。 **Managed File Transfer 中心**即會顯示在「內容」視圖中。
2. 所有協調佇列管理程式都會顯示在「導覽器」視圖中。 展開您用於排定傳送的協調佇列管理程式的名稱。 如果您想要變更所連接的協調佇列管理程式，請在「導覽器」視圖中您想要使用的協調佇列管理程式名稱上按一下滑鼠右鍵，然後再按一下**連接**。
3. 啟動**建立 Managed File Transfer 的新範本**精靈，方法是用滑鼠右鍵按一下**傳送範本**，然後按一下**新建範本**。
4. 遵循精靈畫面上的指示操作。 每一個畫面都提供有上下文相關說明。 若要存取 Windows 上的上下文相關說明，請按 F1 鍵。 在 Linux 上，按 Ctrl+F1 或 Shift+F1。

   如果您已建立包含所有必要傳送詳細資料的範本，請確保已選取**傳送摘要**頁面上的**將傳送設定儲存為範本**勾選框（如果尚未選取此勾選框）。 亦請在「名稱」欄位中輸入範本的名稱。 如果您建立的範本尚未包含所有必要的傳送詳細資料，系統會自動勾選**將傳送設定儲存為範本**勾選框。

相關工作

第 212 頁的『使用檔案傳送範本』

您可以使用檔案傳送範本來儲存一般檔案傳送設定，以進行重複或複式傳送。 請使用 **fteCreateTemplate** 指令，從指令行建立傳送範本，或者使用 IBM MQ Explorer，利用**為受管理檔案傳送建立新範本**精靈建立傳送範本，或選取**將傳送設定儲存為範本**勾選框，在建立檔案傳送時儲存範本。 **傳送範本**視窗會顯示您在 Managed File Transfer 網路中建立的所有傳送範本。

相關參考

fteCreateTemplate：建立新的檔案傳送範本

fteListTemplates

fteDeleteTemplates

備份檔案傳送範本定義

檔案傳送範本包含 XML 文件，定義傳送的來源及目的地檔案規格。 您可以使用此 XML 檔案作為 **fteCreateTemplate** 指令的輸入，以重建檔案傳送範本。

關於這項作業

若要備份包含傳送範本之來源及目的地檔案規格的 XML 文件，請使用 fteCreateTransfer 指令 指令或 IBM MQ Explorer。 若要建立傳送範本 XML 格式的備份檔，請使用下列步驟:

程序

- 方法一: 在 fteCreateTransfer 指令上使用 **-gt** 參數，以產生新檔案的傳送範本 XML 訊息。
- 方法二: 使用 IBM MQ Explorer 來建立範本。
  當您到達「 傳送範本摘要 」頁面時:
  a) 複製 要求訊息 *XML* 預覽。
  b) 將此傳送範本 XML 訊息儲存至新檔案。
- 方法三: 使用 IBM MQ Explorer 來備份現有的範本。
  a) 移至 **受管理檔案傳送 > 佇列管理程式名稱 > 傳送範本**。
  b) 在「傳送」畫面中，強調顯示需要備份的範本，按一下滑鼠右鍵並從蹦現功能表中選取 **編輯** 。
  c) 按 **下一步** ，直到您到達「 傳送範本摘要 」頁面。
  d) 複製 要求訊息 *XML* 預覽。
  e) 將此傳送範本 XML 訊息儲存至新檔案。

結果

您可以使用上述其中一種方法所建立的傳送範本 XML 訊息檔案，作為 fteCreateTemplate 指令的輸入。 如需如何使用此指令的詳細資料，請參閱 **fteCreateTemplate** 指令。

相關參考
fteCreate 範本指令
fteListtTm 平板指令

將資料從檔案傳送至訊息

您可以使用 Managed File Transfer 的「檔案轉為訊息」特性，將資料從一個檔案傳送至 IBM MQ 佇列上的單一訊息或多則訊息。

若要執行「檔案轉為訊息」和「訊息轉為檔案」的傳送，傳送的來源和目的地代理程式都必須為 IBM WebSphere MQ 7.5 或更新版本，或是 IBM WebSphere MQ File Transfer Edition 7.0.3 或更新版本。 如需「訊息轉為檔案」傳送的相關資訊，請參閱第 221 頁的『將資料從訊息傳送至檔案』。

檔案到訊息傳送的目的地代理程式，不得為通訊協定橋接器代理程式或 Connect:Direct 橋接器代理程式。

您可以將檔案資料傳送至 IBM MQ 訊息資料。 IBM MQ 訊息可由應用程式讀取及使用。 支援下列類型的「檔案轉為訊息」傳送:

- 從單一檔案至單一訊息。 該訊息未設定 IBM MQ 群組 ID。
- 從單一檔案至多則訊息，方法是將檔案分割為給定長度的訊息。 這些訊息都具有相同的 IBM MQ 群組 ID。
- 從單一檔案至多則訊息，方法是使用 Java 正規表示式定界字元來分割文字檔。 這些訊息都具有相同的 IBM MQ 群組 ID。
- 從單一檔案至多則訊息，方法是使用十六進位定界字元來分割二進位檔。 這些訊息都具有相同的 IBM MQ 群組 ID。

  如果您要以位元組序列作為定界字元來分割二進位檔，請使用 **fteCreateTransfer** 指令的 **-sqdb** 參數。 如需相關資訊，請參閱 -sqdb 參數。

依預設，透過「檔案轉為訊息」傳送所建立的訊息都是持續性訊息。 這些訊息可以設定為非持續性訊息，或者設定為使用目的地佇列所定義的持續性值。

如果您指定將一個檔案分割為多則訊息，則從該檔案建立的所有訊息都將具有相同的 IBM MQ 群組 ID。 如果您沒有指定將一個檔案分割為多則訊息，則只會從該檔案建立一則訊息，而且該訊息未設定 IBM MQ 群組 ID。

如果您要將檔案傳送至大型訊息，或許多小型訊息，則可能需要變更某些 IBM MQ 或 Managed File Transfer 內容。 如需的相關資訊，請參閱 設定 MQ 屬性及與訊息大小相關聯的 MFT 內容的指引。

註: 如果目的地佇列是叢集佇列或叢集佇列的別名，而且未將代理程式內容 enableClusterQueueInputOutput 設為 true，則在將檔案傳送至佇列時，您會收到一則錯誤訊息。 如需相關資訊，請參閱 如果目的地佇列是叢集佇列或叢集佇列的別名，怎麼辦

配置代理程式以執行「檔案轉為訊息」傳送

依預設，代理程式無法執行檔案轉為訊息或「訊息轉為檔案」傳送。 若要啟用此功能，必須將代理程式內容 enableQueueInputOutput 設定為 true。 為了能夠寫入 IBM MQ 叢集佇列，您還必須將代理程式內容 enableClusterQueueInputOutput 設為 true。

關於這項作業

如果嘗試對未將 enableQueueInputOutput 內容設為 ture 的目的地代理程式執行檔案到訊息傳送，則傳送就會失敗。 發佈至協調佇列管理程式的傳送日誌訊息包含下列訊息:

```
BFGIO0197E: An attempt to write to a queue was rejected by the destination agent. The
agent must have enableQueueInputOutput=true set in the agent.properties file to
support transferring to a queue.
```

若要讓代理程式能夠對佇列進行寫入及讀取，請執行下列步驟:

程序

1. 使用 **fteStopAgent** 指令停止目的地代理程式。
2. 編輯 agent.properties 檔案，以包含 enableQueueInputOutput=true 行。
   agent.properties 檔案位於 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*destination_agent_name* 目錄中。
3. 選擇性的: 編輯 agent.properties 檔案，以包含 enableClusterQueueInputOutput=true 行。
   agent.properties 檔案位於 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*destination_agent_name* 目錄中。
4. 使用 **fteStartAgent** 指令啟動目的地代理程式。

範例：將單一檔案傳送至單一訊息

透過將 **-dq** 參數與 **fteCreateTransfer** 指令搭配使用，您可以將佇列指定為檔案傳送的目的地。 來源檔案必須小於目的地佇列上設定的訊息長度上限。 目的地佇列所在的佇列管理程式與目的地佇列所連接的佇列管理程式不必相同，但這兩個佇列管理程式必須能夠進行通訊。

關於這項作業

來源檔案名為 /tmp/single_record.txt，與來源代理程式 AGENT_NEPTUNE 位於同一系統。 來源代理程式 AGENT_NEPTUNE 使用佇列管理程式 QM_NEPTUNE。 目的地代理程式為 AGENT_VENUS，並且此代理程式連接至佇列管理程式 QM_VENUS。 目的地佇列 RECEIVING_QUEUE 位於佇列管理程式 QM_MERCURY 上。 QM_MERCURY 與佇列管理程式 QM_VENUS 位於相同的 IBM MQ 網路，而且 QM_VENUS 可以存取 QM_MERCURY。

程序

鍵入下列指令:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -dm QM_VENUS
                  -dq RECEIVING_QUEUE@QM_MERCURY /tmp/single_record.txt
```

如果目的地佇列所在的佇列管理程式不同於目的地佇列所使用的佇列管理程式，則必須以下列格式指定 **-dq** 參數的值：*queue_name*@*queue_manager_name*。 如果未在該值中指定 @*queue_manager_name*，則目的地代理程式會假設目的地佇列是位於目的地代理程式佇列管理程式上。 enableClusterQueueInputOutput 代理程式內容設為 true 屬異常狀況。 在此情況下，目的地代理程式將使用標準 IBM MQ 解析程序來判定放置佇列的位置。

來源代理程式 AGENT_NEPTUNE 會從 /tmp/single_record.txt 檔案讀取資料，然後將該資料傳送至目的地代理程式 AGENT_VENUS。 目的地代理程式 AGENT_VENUS 會將該資料傳送至 RECEIVING_QUEUE@QM_MERCURY 佇列上的持續性訊息。 該訊息未設定 IBM MQ 群組 ID。

範例：依長度將單一檔案分割為多則訊息

您可以使用 **fteCreateTransfer** 指令的 **-qs** 參數，將檔案分割成多則 IBM MQ 訊息。 檔案會被分割為多個固定長度的區段，每個區段會寫入個別訊息。

關於這項作業

來源檔案名為 /tmp/source.file，大小為 36KB。 該來源檔案與來源代理程式 AGENT_NEPTUNE 位於同一系統。 來源代理程式 AGENT_NEPTUNE 連接至佇列管理程式 QM_NEPTUNE。 目的地代理程式為 AGENT_MERCURY，此代理程式連接至佇列管理程式 QM_MERCURY。 目的地佇列 RECEIVING_QUEUE 還位於佇列管理程式 QM_MERCURY 上。 傳送會將該來源檔案分割為大小為 1KB 的區段，然後將每個區段寫入 RECEIVING_QUEUE 上的訊息。

程序

鍵入下列指令:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
                  -dq RECEIVING_QUEUE -qs 1K /tmp/source.file
```

來源代理程式 AGENT_NEPTUNE 會從 /tmp/source.file 檔案讀取資料，然後將該資料傳送至目的地代理程式 AGENT_MERCURY。 目的地代理程式 AGENT_MERCURY 會將資料寫入佇列 RECEIVING_QUEUE@QM_MERCURY 上的 36 個 1KB 持續性訊息。 這些訊息都具有相同的 IBM MQ 群組 ID，而且群組中的最後一則訊息已設定 IBM MQ LAST_MSG_IN_GROUP 旗標。

範例：使用正規表示式定界字元將一個文字檔分割為多則訊息

透過在給定的 Java 正規表示式的每個相符項之處分割單一文字檔，可以將該檔案傳送至多則訊息。 若要這樣做，需使用 **fteCreateTransfer** 指令的 **-dqdt** 參數。

關於這項作業

檔案會被分割為多個可變長度的區段，每個區段會寫入個別訊息。 對於文字檔，會在檔案中文字符合給定正規表示式的每個點來分割檔案。 來源檔案名為 /tmp/names.text，並且具有下列內容:

```
Jenny Jones,John Smith,Jane Brown
```

指定該檔案分割點的正規表示式為逗點字元 (,)。

該來源檔案與連接至佇列管理程式 QM_NEPTUNE 的來源代理程式 *AGENT_NEPTUNE* 位於同一系統。 目的地佇列 *RECEIVING_QUEUE* 位於佇列管理程式 *QM_MERCURY* 上。 *QM_MERCURY* 也是目的地代理程式 *AGENT_MERCURY* 所使用的佇列管理程式。 傳送會將該來源檔案分割為多個區段，並將每個區段寫入 *RECEIVING_QUEUE* 上的訊息。

程序

鍵入下列指令:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
                  -dq RECEIVING_QUEUE -t text -dqdp postfix -dqdt "," /tmp/names.text
```

來源代理程式 AGENT_NEPTUNE 會從 /tmp/names.text 檔案讀取資料，然後將該資料傳送至目的地代理程式 AGENT_MERCURY。 目的地代理程式 AGENT_MERCURY 會將資料寫入 *RECEIVING_QUEUE* 佇列上的三個持續訊息。 這些訊息都具有相同的 IBM MQ 群組 ID，而且群組中的最後一則訊息已設定 IBM MQ LAST_MSG_IN_GROUP 旗標。

訊息中的資料如下所示。

- 第一則訊息：

```
Jenny Jones
```

- 第二則訊息：

```
John Smith
```

- 第三則訊息：

```
Jane Brown
```

範例：使用正規表示式定界字元分割文字檔，並在訊息中包含定界字元

透過在給定的 Java 正規表示式的每個相符項之處分割單一文字檔，可以將該檔案傳送至多則訊息，並在產生的訊息中包含正規表示式相符項。 若要這樣做，需使用 **fteCreateTransfer** 指令的 **-dqdt** 及 **-qi** 參數。

關於這項作業

將單一文字檔傳送至佇列上的多則訊息。 檔案會被分割為多個可變長度的區段，每個區段會寫入個別訊息。 對於文字檔，會在檔案中文字符合給定正規表示式的每個點來分割檔案。 來源檔案名為 `/tmp/customers.text`，並且具有下列內容：

```
Customer name: John Smith
Customer contact details: john@example.net
Customer number: 314

Customer name: Jane Brown
Customer contact details: jane@example.com
Customer number: 42

Customer name: James Jones
Customer contact details: jjones@example.net
Customer number: 26
```

指定該檔案分割點的正規表示式為 `Customer\snumber:\s\d+`，相符項為 "Customer number:" 後面接任何位數的文字。 在指令行上指定的正規表示式必須以雙引號括住，以避免指令 Shell 評估正規表示式。 該正規表示式是當成 Java 正規表示式進行評估。 如需相關資訊，請參閱 MFT 使用的正規表示式。

依預設，正規表示式可以比對的字元數為五個。 此範例中所使用的正規表示式會比對大於五個字元的字串。 若要啟用大於五個字元的比對，請編輯代理程式內容檔，以包含 **maxDelimiterMatchLength** 內容。

依預設，訊息中不會包含符合正規表示式的文字。 若要在訊息中包含符合正規表示式的文字（如此範例中所示），請使用 **-qi** 參數。 來源檔案與連接至佇列管理程式 QM_NEPTUNE 的來源代理程式 AGENT_NEPTUNE 位於同一系統。 目的地佇列 RECEIVING_QUEUE 位於佇列管理程式 QM_MERCURY 上。 QM_MERCURY 也是目的地代理程式 AGENT_MERCURY 所使用的佇列管理程式。 傳送會將該來源檔案分割為多個區段，並將每個區段寫入 RECEIVING_QUEUE 上的訊息。

程序

1. 使用以下指令停止目的地代理程式：

```
fteStopAgent AGENT_MERCURY
```

2. 將以下行新增至 AGENT_MERCURY 的代理程式內容檔：

```
maxDelimiterMatchLength=25
```

註： 增加 **maxDelimiterMatchLength** 值可能會降低效能。

3. 使用以下指令啟動目的地代理程式：

```
fteStartAgent AGENT_MERCURY
```

4. 鍵入下列指令：

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
 -dq RECEIVING_QUEUE
     text -dqdt "Customer\snumber:\s\d+" -qi -dqdp postfix /tmp/customers.text
```

來源代理程式 AGENT_NEPTUNE 會從 /tmp/customers.text 檔案讀取資料，然後將該資料傳送至目的地代理程式 AGENT_MERCURY。 目的地代理程式 AGENT_MERCURY 會將資料寫入 RECEIVING_QUEUE 佇列上的三個持續性訊息。 這些訊息都具有相同的 IBM MQ 群組 ID，而且群組中的最後一則訊息已設定 IBM MQ LAST_MSG_IN_GROUP 旗標。

訊息中的資料如下所示。

- 第一則訊息:

```
Customer name: John Smith
Customer contact details: john@example.net
Customer number: 314
```

- 第二則訊息:

```
Customer name: Jane Brown
Customer contact details: jane@example.com
Customer number: 42
```

- 第三則訊息:

```
Customer name: James Jones
Customer contact details: jjones@example.net
Customer number: 26
```

範例：設定有關「檔案轉為訊息」傳送的 IBM MQ 訊息內容

您可以在 **fteCreateTransfer** 指令上使用 **-qmp** 參數，以指定是否在由傳送寫入目的地佇列的第一則訊息上設定 IBM MQ 訊息內容。 IBM MQ 訊息內容可讓應用程式選取要處理的訊息，或在不存取 IBM MQ 訊息描述子 (MQMD) 或 MQRFH2 標頭的情況下擷取訊息的相關資訊。

關於這項作業

在 **fteCreateTransfer** 指令中包含 -qmp true 參數。 在此範例中，提交該指令的使用者的 MQMD 使用者 ID 為 larmer。

程序

鍵入下列指令:

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM -qmp true
      -t text /tmp/source_file.txt
```

由目的地代理程式 AGENT_SATURN 寫入佇列 MY_QUEUE(位於佇列管理程式 MyQM 上) 的第一則訊息的 IBM MQ 訊息內容設為下列值:

```
usr.WMQFTETransferId=414cbaedefa234889d999a8ed09782395ea213ebbc9377cd
usr.WMQFTETransferMode=text
usr.WMQFTESourceAgent=AGENT_JUPITER
usr.WMQFTEDestinationAgent=AGENT_SATURN
usr.WMQFTEFileName=source_file.txt
usr.WMQFTEFileSize=1024
usr.WMQFTEFileLastModified=1273740879040
usr.WMQFTEFileIndex=0
usr.WMQFTEMqmdUser=larmer
```

範例：設定有關「檔案轉為訊息」傳送的使用者定義內容

使用者定義 meta 資料設定為由傳送寫入目的地佇列的第一則訊息的 IBM MQ 訊息內容。 IBM MQ 訊息內容可讓應用程式選取要處理的訊息，或在未存取 IBM MQ 訊息描述子 (MQMD) 或 MQRFH2 標頭的情況下擷取訊息的相關資訊。

關於這項作業

在 **fteCreateTransfer** 指令中包含參數 `-qmp true` 和 `-md account=123456` ，以將 RFH2 標頭中的 `usr.account` 內容設為 `123456` 。

程序

鍵入下列指令：

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM
                  -qmp true -md account=123456 /tmp/source_file.txt
```

除了 IBM MQ 訊息內容標準集外，由目的地代理程式 `AGENT_SATURN` 寫入佇列 `MY_QUEUE`（位於佇列管理程式 `MyQM` 上）的第一則訊息的訊息標頭中還會設定使用者定義內容。 標頭設定為下列值：

```
usr.account=123456
```

字首 `usr` 會新增至使用者定義 meta 資料名稱的開頭。

範例：新增「檔案轉為訊息」傳送的使用者定義訊息內容

將 Managed File Transfer 用於訊息轉為檔案的受管理傳送時，您可以針對產生的訊息併入使用者定義的訊息內容。

關於這項作業

您可以使用下列任何方法來定義自訂訊息內容：

- 指定傳送要求的 **-md** 參數。 如需相關資訊，請參閱 第 219 頁的『範例：設定有關「檔案轉為訊息」傳送的使用者定義內容』。
- 使用 Ant 作業; 您可以使用 fte: filecopy 或 fte:filemove。 下列範例說明 fte:filecopy 作業：

```
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">
<!-- Initialise the properties used in this script.-->

<target name="init" description="initialise task properties">
               <property name="src.file" value="/home/user/file1.bin"/>
               <property name="dst.queue" value="TEST.QUEUE@qm2"/>
               <fte:uuid property="job.name" length="8"
prefix="copyjob#"/>
</target>
<target name="step1" depends="init" description="transfer file">

<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
             rcproperty="copy.result">

<fte:metadata>
<fte:entry name="fileName" value="${FileName}"/>
</fte:metadata>

<fte:filespec srcfilespec="${src.file}" dstqueue="${dst.queue}"
dstmsgprops="true"/>

</fte:filecopy>

</target>
</project>
```

- 使用資源監視器及變數替代。 下列範例顯示部分傳送作業 XML：

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor
xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="5.00"
xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinit ion ./Monitor.xsd">
 <name>METADATA</name>
 <pollInterval units="minutes">5</pollInterval>
 <batch maxSize="5"/>
 <agent>AGENT1</agent>
 <resources>
   <directory recursionLevel="0">e:\temp</directory>
 </resources>
 <triggerMatch>
   <conditions>
     <allOf>
       <condition>
         <fileMatch>
           <pattern>*.txt</pattern>
         </fileMatch>
       </condition>
     </allOf>
   </conditions>
 </triggerMatch>
 <tasks>
   <task>
     <name/>
     <transfer>
       <request version="5.00"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
         <managedTransfer>
           <originator>
             <hostName>mqjason.raleigh.ibm.com.</hostName>
             <userID>administrator</userID>
           </originator>
           <sourceAgent QMgr="AGENTQM" agent="AGENT1"/>
           <destinationAgent QMgr="AGENTQM" agent="AGENT2"/>
           <transferSet priority="0">
             <metaDataSet>
               <metaData key="FileName">${FileName}</metaData>
             </metaDataSet>
             <item checksumMethod="MD5" mode="text">
               <source disposition="delete" recursive="false">
                 <file>${FilePath}</file>
               </source>
               <destination type="queue">
                 <queue persistent="true"
setMqProps="true">TEST.QUEUE@AGENTQM</queue>
               </destination>
             </item>
           </transferSet>
           <job>
             <name>Metadata_example</name>
           </job>
         </managedTransfer>
       </request>
     </transfer>
   </task>
 </tasks>
 <originator>
   <hostName>mqjason.raleigh.ibm.com.</hostName>
   <userID>administrator</userID>
 </originator>
</monitor:monitor>
```

相關工作

第 218 頁的『範例：設定有關「檔案轉為訊息」傳送的 IBM MQ 訊息內容』

您可以在 **fteCreateTransfer** 指令上使用 **-qmp** 參數，以指定是否在由傳送寫入目的地佇列的第一則訊息上設定 IBM MQ 訊息內容。 IBM MQ 訊息內容可讓應用程式選取要處理的訊息，或在不存取 IBM MQ 訊息描述子 (MQMD) 或 MQRFH2 標頭的情況下擷取訊息的相關資訊。

相關參考

fte: filecopy Ant task

fte: filemove Ant task

檔案轉為訊息傳送失敗

在代理程式已啟動將檔案資料寫入目的地佇列作業後，如果「檔案轉為訊息」傳送失敗，代理程式會將一則訊息寫入佇列，以指示使用訊息的應用程式發生失敗。

如果發生失敗，已寫入目的地佇列的訊息將：

- 為空白
- 具有與代理程式先前寫入目的地佇列的訊息相同的 IBM MQ 群組 ID
- 已設定 IBM MQ LAST_MSG_IN_GROUP 旗標
- 包含其他 IBM MQ 訊息內容（如果已啟用訊息內容）。 如需相關資訊，請參閱 MFT 在寫入目的地佇列的訊息上設定的 MQ 訊息內容主題。

範例

透過執行以下指令來要求傳送：

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq RECEIVING_QUEUE
                  -qmp true -qs 1K /tmp/source1.txt
```

source1.txt 檔案為 48KB。 傳送會將此檔案分割為 1KB 的訊息，並將這些訊息寫入目的地佇列 RECEIVING_QUEUE。

在傳送進行時，代理程式將 16 則訊息寫入 RECEIVING_QUEUE 後，來源代理程式發生失敗。

代理程式會將空白訊息寫入 RECEIVING_QUEUE。 除了訊息內容標準集外，空白訊息還具有以下訊息內容集：

```
usr.WMQFTEResultCode = 40
usr.WMQFTESupplement = BFGTR0036I: The transfer failed to complete successfully.
```

V 9.2.0.2 V 9.2.2 從 IBM MQ 9.2.0 Fix Pack 2 和 IBM MQ 9.2.2 開始，當來自檔案的傳送失敗時，由於定界字元大小檢查錯誤，只會傳送一則空訊息。 此外，如果傳送失敗的原因是定界字元超出目的地代理程式上的設定大小，則會將使用者內容新增至此訊息。

將資料從訊息傳送至檔案

Managed File Transfer 的「訊息轉為檔案」特性，可讓您從 IBM MQ 佇列的一個以上訊息中，將資料傳送至檔案、資料集（在 z/OS 上）或使用者檔案空間。 如果您具有可建立或處理 IBM MQ 訊息的應用程式，則可以使用 Managed File Transfer 的「訊息轉為檔案」功能，將這些訊息傳送至您 Managed File Transfer 網路中任何系統上的檔案。

如需「檔案轉為訊息」傳送的相關資訊，請參閱第 214 頁的『將資料從檔案傳送至訊息』。

小心：「訊息轉為檔案」傳送的來源代理程式不可以是通訊協定橋接器代理程式或 Connect:Direct 橋接器代理程式。

您可以將 IBM MQ 訊息資料傳送至檔案。 下列是受支援的「訊息轉為檔案」傳送類型：

- 從單一訊息至單一檔案
- 從多個訊息至單一檔案
- 從具有相同 IBM MQ 群組 ID 的多個訊息至單一檔案。
- 從多個訊息至單一檔案，包括寫入至檔案之每一個訊息中的資料之間的文字或二進位定界字元。

如果您要從大型訊息或許多小型訊息傳送檔案，則可能必須變更某些 IBM MQ 或 Managed File Transfer 內容。 如需的相關資訊，請參閱 設定與訊息大小相關聯的 MQ 屬性及 MFT 內容的指引。

從 IBM MQ 9.1.0 開始，在「訊息轉為檔案」傳送中，來源代理程式會瀏覽來自來源佇列的訊息，與舊版 IBM MQ 中的破壞性 GET 不同。 瀏覽完所有訊息（如果使用了訊息分組，則為群組中）並將資料寫入目的地檔案之後，就會從來源佇列中移除訊息。 這可在萬一傳送失敗或取消時，使訊息仍保留在來源佇列中。 由於這項變更，因此必須隨 GET 權限一起提供 BROWSE 權限，才能執行「訊息轉為檔案」的傳送。

從 IBM MQ 9.0.0 Fix Pack 2 及 IBM MQ 9.0.4，會更新 Managed File Transfer ，以還原先前由 APAR IT18213 移除的傳送 ID 與傳送要求 XML 有效負載內 groupId 屬性值的比較檢查。 如果這兩個 ID 相同，則在「訊息轉為檔案」傳送的輸入佇列中第一次進行 MQGET 嘗試時，來源代理程式會使用此 ID 作為訊息 ID 比對選項（相對於群組 ID 比對選項）。

配置代理程式以執行訊息轉為檔案傳送

依預設，代理程式無法執行訊息與檔案之間的來回傳送。 若要啟用此功能，必須將代理程式內容 enableQueueInputOutput 設為 ture。

關於這項作業

如果您嘗試從 enableQueueInputOutput 內容未設為 true 的來源代理程式中執行「訊息轉為檔案」的傳送，傳送將會失敗。 發佈至協調佇列管理程式的傳送日誌訊息包含下列訊息：

```
BFGIO0197E: An attempt to read from a queue was rejected by the source agent.
The agent must have enableQueueInputOutput=true set in the agent.properties file
to support transferring from a queue.
```

若要讓代理程式能夠對佇列進行寫入及讀取，請執行下列步驟：

程序

1. 使用 **fteStopAgent** 指令來停止來源代理程式。
2. 編輯 agent.properties 檔案，以包含 enableQueueInputOutput=true 行。
   agent.properties 檔案位於 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*source_agent_name* 目錄中。
3. 使用 **fteStartAgent** 指令來啟動來源代理程式。

範例：從佇列傳送至單一檔案

您可以搭配使用 **-sq** 參數與 **fteCreateTransfer** 指令，將 IBM MQ 佇列指定為檔案傳送的來源。

關於這項作業

來源資料內含在佇列 START_QUEUE 上的三個訊息中。 此佇列必須位於來源代理程式的佇列管理程式 *QM_NEPTUNE* 上。

程序

鍵入下列指令：

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE
                  -da AGENT_VENUS -df /out/three_to_one.txt
                  -sq START_QUEUE
```

佇列 *START_QUEUE* 上的訊息中資料，會被寫入 AGENT_VENUS 執行所在系統上的 /out/three_to_one.txt 檔。

範例：將訊息群組從佇列傳送至單一檔案

您可以搭配使用 **-sq** 及 **-sqgi** 參數與 **fteCreateTransfer** 指令，在 IBM MQ 佇列上指定單一完整群組作為檔案傳送的來源。

關於這項作業

在此範例中，佇列 START_QUEUE 上有十個訊息。 此佇列必須位於來源代理程式的佇列管理程式 *QM_NEPTUNE* 上。 前三個訊息屬於 IBM MQ 群組 ID 為 41424b3ef3a2202020202020202020202020202020201111 的群組；此群組並不是完整群組。 接下來

的五個訊息屬於 IBM MQ 群組 ID 為 41424b3ef3a2202020202020202020202020202020202222 的群組；此群組是完整群組。 其餘的兩個訊息屬於 IBM MQ 群組 ID 為 41424b3ef3a2202020202020202020202020202020203333 的群組；此群組是完整群組。

程序

鍵入下列指令：

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS
                  -df /out/group.txt -sqgi -sq START_QUEUE
```

訊息中屬於佇列 *START_QUEUE* 上第一個完整群組 (具有 IBM MQ 群組 ID 41424b3ef3a2202020202020202020202020202020202222 的群組) 的資料會寫入 AGENT_VENUS 執行所在系統上的 /out/group.txt 檔案。

範例：在每一個訊息中的資料前面插入文字定界字元

在文字模式下從來源佇列傳送至檔案時，您可以搭配使用 **fteCreateTransfer** 指令與 **-sq**、**-sqdt** 及 **-sqdp** 參數，以指定在個別訊息中的資料前面插入文字定界字元。

關於這項作業

在此範例中，佇列 START_QUEUE 上有四個訊息。 此佇列位於來源代理程式的佇列管理程式 *QM_NEPTUNE* 上。 要在每一則訊息的資料前面插入的文字定界字元，可以用 Java 文字字串來表示，例如：\n\u002D\u002D\u002D\n。

程序

鍵入下列指令：

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -df /out/output.txt
                  -t text -sqdt "\n\u002D\u002D\u002D\n" -sqdp prefix -sq START_QUEUE
```

來源代理程式 AGENT_NEPTUNE 會分別為 START_QUEUE 上四個訊息中的資料加上前置的文字定界字元。 這項資料會寫入目的地檔案 /out/output.txt 中。

範例：在每一個訊息中的資料後面插入二進位定界字元

在二進位模式下從來源佇列傳送至檔案時，您可以搭配使用 **fteCreateTransfer** 指令與 **-sq**、**-sqdb** 及 **-sqdp** 參數，以指定在個別訊息中的資料後面插入二進位定界字元。

關於這項作業

在此範例中，佇列 START_QUEUE 上有三個訊息。 此佇列位於來源代理程式的佇列管理程式 *QM_NEPTUNE* 上。 要在每一個訊息中的資料後面插入的二進位定界字元，必須使用以逗點區隔的十六進位位元組清單表示，例如：x34,xE7,xAE。

程序

鍵入下列指令：

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -df /out/binary.file
                  -sqdp postfix -sqdb x34,xE7,xAE -sq START_QUEUE
```

來源代理程式 AGENT_NEPTUNE 會分別為 START_QUEUE 上三個訊息中的資料加上二進位定界字元。 這項資料會寫入目的地檔案 /out/binary.file 中。

監視佇列及使用變數替代

您可以使用 **fteCreateMonitor** 指令來監視佇列，並將訊息從受監視佇列傳送至檔案。 在第一個要從受監視佇列讀取的訊息中，任何 IBM MQ 訊息內容的值都可代入作業 XML 定義中，並用來定義傳送行為。

關於這項作業

在此範例中，來源代理程式稱為 AGENT_VENUS，其連接至 QM_VENUS。 AGENT_VENUS 所監視的佇列稱為 START_QUEUE，位於 QM_VENUS 上。 代理程式每隔 30 分鐘會輪詢佇列一次。

當完整訊息群組寫入佇列時，監視器作業即會將該訊息群組傳送至眾多目的地代理程式之一（全部都連接至佇列管理程式 QM_MARS）的檔案上。 訊息群組傳送目的地的檔案名稱，由群組中第一個訊息上的 IBM MQ 訊息內容 `usr.fileName` 所定義。 訊息群組傳送目的地的代理程式名稱，由群組中第一個訊息上的 IBM MQ 訊息內容 `usr.toAgent` 所定義。 如果未設定 `usr.toAgent` 標頭，目的地代理程式將會使用預設值 AGENT_MAGENTA。

當您指定 `useGroups="true"`時，如果未同時指定 `groupId="${GROUPID}"`，則傳送只會取得佇列上的第一則訊息。 比方說，例如您使用變數替代產生 fileName，`a.txt` 的內容因此有可能不正確。 這是因為 fileName 是由監視器所產生，但傳送實際上取用的並不是應產生 fileName 檔案的訊息。

程序

1. 建立作業 XML，以定義監視器被觸發後所執行的作業。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
     <destinationAgent agent="${toAgent}" QMgr="QM_MARS" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/${fileName}.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

   取代為 IBM MQ 訊息標頭值的變數，會以**粗體**強調顯示。 此作業 XML 會儲存至 `/home/USER1/task.xml` 檔。

2. 建立用來監視佇列 START_QUEUE 的資源監視器。
   提交下列指令：

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. 使用者或程式將訊息群組寫入佇列 START_QUEUE。
   此群組中的第一個訊息具有下列 IBM MQ 訊息內容集：

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. 在寫入完整群組時，會觸發監視器。 代理程式將 IBM MQ 訊息內容代入作業 XML 中。
   這會導致作業 XML 轉換成：

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
```

```
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
     <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/larmer.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

結果

即會執行作業 XML 所定義的傳送。 AGENT_VENUS 從 START_QUEUE 讀取的完整訊息群組，會寫入 AGENT_VIOLET 執行時所在系統上稱為 /reports/larmer.rpt 的檔案。

下一步

將每一個訊息傳送至個別檔案

如果要監視佇列並讓每則訊息傳送至個別檔案，您可以使用與本主題中先前所述方式相似的技術。

1. 依照先前說明來建立監視器，並在 **fteCreateMonitor** 指令上指定 **-tr completeGroups** 參數。
2. 在作業 XML 中指定下列內容：

```
<queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
```

不過，當您將訊息放入來源佇列時，請不要將它們放置在 IBM MQ 群組中。 為每一個訊息新增 IBM MQ 訊息內容。 例如，您可以為每一個訊息指定具有唯一檔案名稱值的 usr.filename 內容。 這實際上會讓 Managed File Transfer Agent 將來源佇列上的每一則訊息都視為個別群組。

範例：使用 IBM MQ 訊息內容阻止訊息轉為檔案傳送

您可以將 usr.UserReturnCode IBM MQ 訊息內容設定為非零值，讓「訊息轉為檔案」的傳送無法執行。 您也可以設定 usr.UserSupplement IBM MQ 訊息內容，指定無法執行之原因的增補資訊。

關於這項作業

在此範例中，佇列 INPUT_QUEUE 與 /home/user/output.file 檔之間正在進行傳送。

使用者會建立訊息，並將其放置於佇列 INPUT_QUEUE 上。 來源代理程式採用佇列 INPUT_QUEUE 中的訊息，並將傳送資料傳送至目的地代理程式。 目的地代理程式則將此資料寫入 /home/user/output.file 檔中。

將訊息寫入佇列 INPUT_QUEUE 的使用者想要停止進行中的傳送，並且刪除任何已寫入目的地檔案的資料。

程序

1. 使用者將訊息寫入具有下列 IBM MQ 訊息內容集的佇列 INPUT_QUEUE：

```
usr.UserReturnCode=1
usr.UserSupplement="Cancelling transfer - sent wrong data."
```

2. 來源代理程式讀取這些 IBM MQ 訊息內容，並停止處理來自該佇列的訊息。 目的地代理程式刪除任何已寫入目的地目錄的檔案資料。
3. 來源代理程式將傳送日誌訊息傳送至協調佇列管理程式，回報傳送失敗。
   訊息包含下列資訊：

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
               ID="414d5120514d312020202020202020207e970d4920008702" agentRole="sourceAgent"
             xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
  <action time="2008-11-02T21:28:09.593Z">progress</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
            version="6.1 build 7601 Service Pack 1" />
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
            version="6.1 build 7601 Service Pack 1" />
  </destinationAgent>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1   </mqmdUserID>
  </originator>
  <transferSet index="0" size="1"
               startTime="2008-11-02T21:28:09.281Z"
               total="1">
    <item mode="binary">
      <source>
        <queue>INPUT_QUEUE@QM1</queue>
      </source>
      <destination exist="error">
        <file>/home/user/output.file</file>
      </destination>
      <status resultCode="1">
        <supplement>Cancelling transfer - sent wrong data.</supplement>
      </status>
    </item>
  </transferSet>
</transaction>
```

通訊協定橋接器

通訊協定橋接器可供 Managed File Transfer (MFT) 網路存取儲存在 MFT 網路之外的檔案伺服器上的檔案（在本端網域或遠端位置）。 此檔案伺服器可使用 FTP、FTPS 或 SFTP 網路通訊協定。 每一個檔案伺服器需要至少一個專用的代理程式。 專用代理程式即所謂的通訊協定橋接器代理程式。 橋接器代理程式可以與多個檔案伺服器互動。

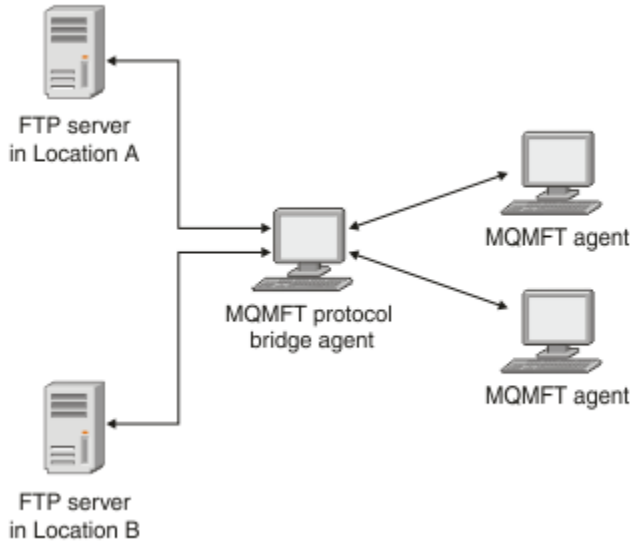
通訊協定橋接器可提供作為 Managed File Transfer 的服務元件的一部分。 在連接至不同檔案伺服器的單一 MFT 系統上，您可以有多個專用的代理程式。

您可以使用通訊協定橋接器代理程式，將檔案同時傳送至多個端點。 MFT 提供稱為 `ProtocolBridgeProperties.xml` 的檔案，您可以編輯此檔案，定義您想要將檔案傳送至哪些不同的通訊協定檔案伺服器。 **fteCreateBridgeAgent** 指令為您將預設通訊協定檔案伺服器的詳細資料新增至 `ProtocolBridgeProperties.xml`。 此檔案以 通訊協定橋接器內容檔格式說明。

您可以使用通訊協定橋接器代理程式來執行下列動作：

- 使用 FTP、FTPS 或 SFTP 從 MFT 網路將檔案上傳至遠端伺服器。
- 使用 FTP、FTPS 或 SFTP 從遠端伺服器將檔案下載至 MFT 網路

註: 通訊協定橋接器代理程式僅支援允許以絕對檔案路徑存取檔案的 FTP、FTPS 或 SFTP 伺服器。 如果在傳送要求中指定了相對檔案路徑，則通訊協定橋接器代理程式將會嘗試根據用來登入通訊協定伺服器的起始目錄，將相對路徑轉換成絕對檔案路徑。 通訊協定橋接器代理程式不支援只能根據現行目錄來存取檔案的通訊協定伺服器。



此圖顯示不同位置上的兩個 FTP 伺服器。 這些 FTP 伺服器用來與 Managed File Transfer 代理程式交換檔案。 通訊協定橋接器代理程式位於 FTP 伺服器與 MFT 網路的其餘部分之間，且配置為與這兩個 FTP 伺服器進行通訊。

除了通訊協定橋接器代理程式之外，請確定您在 MFT 網路中還有另一個代理程式。 通訊協定橋接器代理程式只是 FTP、FTPS 或 SFTP 伺服器的橋接器，不會將傳送的檔案寫入本端磁碟。 如果要對 FTP、FTPS 或 SFTP 伺服器來回傳送檔案，您必須使用通訊協定橋接器代理程式作為檔案傳送的目的地或來源（代表 FTP、FTPS 或 SFTP 伺服器），並以另一個標準代理程式作為對應的來源或目的地。

使用通訊協定橋接器傳送檔案時，橋接器必須有權讀取您要傳送的檔案所在的來源或目的地目錄。 比方說，如果您想要從只具有執行權限 (d--x--x--x) 的目錄 `/home/fte/bridge` 傳送檔案，則您嘗試從這個目錄執行的任何傳送都會失敗，並產生下列錯誤訊息：

```
BFGBR0032E: Attempt to read filename from the protocol file server
  has failed with server error 550. Failed to open file.
```

配置通訊協定橋接器代理程式

通訊協定橋接器代理程式如同標準 MFT 代理程式。 您可以使用 **fteCreateBridgeAgent** 指令來建立通訊協定橋接器代理程式。 您可以使用 `ProtocolBridgeProperties.xml` 檔案 (在 通訊協定橋接器內容檔格式中說明) 來配置通訊協定橋接器代理程式。 如果您使用舊版，請使用 進階代理程式內容: 通訊協定橋接器及 進階代理程式內容: 通訊協定橋接器代理程式記載中說明的特定通訊協定橋接器內容來配置代理程式。 對於所有版本，您也可以如第 234 頁的『對映檔案伺服器的認證』中所述配置認證對映。 為特定通訊協定檔案伺服器配置通訊協定橋接器代理程式之後，該代理程式即可為此用途專用。

通訊協定橋接器回復

如果因為檔案伺服器無法使用，導致通訊協定橋接器代理程式無法連接至檔案伺服器，在檔案伺服器變成可用之前，所有檔案傳送要求都會排入佇列中。 如果因為代理程式使用錯誤認證，導致通訊協定橋接器代理程式無法連接至檔案伺服器，則傳送會失敗，且傳送日誌訊息會反映此錯誤。 如果通訊協定橋接器代理程式由於任何原因而結束，則所有要求的檔案傳送都會保留，並於通訊協定橋接器重新啟動之後繼續進行。

在檔案傳送期間，檔案通常以暫存檔形式寫入目的地，然後在傳送完成時重新命名。 但是，如果傳送目的地是配置為限制寫入的通訊協定檔案伺服器（使用者可以將檔案上傳至通訊協定檔案伺服器，但無法以任何方式變更這些已上傳的檔案；實際上，使用者只能寫入一次），已傳送的檔案將直接寫入目的地。 這表示如果傳送期間發生問題，目的地通訊協定檔案伺服器上會留下未完整寫入的檔案，Managed File Transfer 無法刪除或編輯這些檔案。 在此情況下，傳送會失敗。

使用 ProtocolBridgeProperties.xml 檔定義通訊協定檔案伺服器的內容

使用 Managed File Transfer 在代理程式配置目錄中提供的 ProtocolBridgeProperties.xml 檔案，定義您要在其中來回傳送檔案的一個以上通訊協定檔案伺服器的內容。

關於這項作業

fteCreateBridgeAgent 指令在代理程式配置目錄 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* 中建立 ProtocolBridgeProperties.xml 檔案。 如果執行此指令時指定預設通訊協定檔案伺服器，則此指令也會在檔案中建立此伺服器的項目。

BFGCL0392I 訊息會提供 ProtocolBridgeProperties.xml 檔案的位置。

```
<?xml version="1.0" encoding="IBM-1047"?>
<!--
This ProtocolBridgeProperties.xml file determines the protocol servers that will be accessed by
the
MQMFT protocol bridge agent.

Each protocol server is defined using either a <tns:ftpServer>, <tns:ftpsServer>, or
<tns:sftpServer>
element - depending on the protocol used to communicate with the server.  When the protocol
bridge agent participates in a managed file transfer it will determine which server to used
based on
the prefix (if any) present on the file path.  For example a file path of 'server1:/home/user/
file.txt' would
be interpreted as a request to transfer /home/user/file.txt using 'server1'.  The server name
is compared
to the 'name' attribute of each <tns:ftpServer>, <tns:ftpsServer> or <tns:sftpServer> element
in this
XML document and the first match is used to determine which protocol server the protocol bridge
agent will connect to.  If no match is found then the managed file transfer operation will fail.

If a file path is not prefixed with a server name, for example '/home/user/file.txt' then this
XML
document can specify a default server to use for the managed file transfer.  To specify a
default server use the <tns:defaultServer> element as the first element inside the
<tns:serverProperties>
element.  The default server will be used whenever the protocol bridge agent participates in
a managed file transfer for file names which do not specify a prefix.

An optional <tns:limits> element can be specified within each server definition. This element
contains
attributes that govern the amount of resources used by each defined server.

An optional <tns:credentialsFile> element can be specified within each serverProperties
definition. This
element contains a path to a file containing credentials to be used when connecting to defined
servers.

An example ProtocolBridgeProperties.xml file is as follows:

<?xml version="1.0" encoding="UTF-8"?>
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
ProtocolBridgeProperties.xsd">

    <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml" />

    <tns:defaultServer name="myFTPserver" />

    <tns:ftpServer name="myFTPserver" host="windows.hursley.ibm.com" port="1234"
platform="windows"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF-8"
        listFormat="unix" limitedWrite="false">

        <tns:limits maxListFileNames="100" maxListDirectoryLevels="999999999"
            maxReconnectRetry="2"  reconnectWaitPeriod="10"
            maxSessions="60" socketTimeout="30" />

    </tns:ftpServer>

    <tns:ftpsServer name="myFTPSserver" host="unix.hursley.ibm.com" platform="unix"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF8"
        listFormat="unix" limitedWrite="false" ftpsType="explicit"
```

```
        trustStore="C:\FTE\keystores\myFTPSserver\FTPSKeyStore.jks"
trustStorePassword="password">

        <tns:limits maxReconnectRetry="10" connectionTimeout="10"/>

    </tns:ftpsServer>

    <tns:sftpServer name="mySFTPserver" host="windows.hursley.ibm.com" platform="windows"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF-8"
        limitedWrite="false">

        <tns:limits connectionTimeout="60"/>

    </tns:sftpServer>

</tns:serverProperties>

This example shows the outermost <tns:serverProperties> element which must exist for the
document to
be valid, an optional <tns:defaultServer> element, as well as definitions for an FTP, FTPS and
SFTP server.

The attributes of the <tns:ftpServer>, <tns:ftpsServer> and <tns:sftpServer> elements determine
the
characteristics of the connection established to the server.  These attributes correspond to
the command
line parameters for the 'fteCreateBridgeAgent' command.

The following attributes are valid for all of the <tns:ftpServer>, <tns:ftpsServer> and
<tns:sftpServer>
elements: name, host, port, platform, fileEncoding, limitedWrite and controlEncoding.

The following attributes are valid for the <tns:ftpServer> and <tns:ftpsServer> elements:
timezone, locale,
listFormat, listFileRecentDateFormat, listFileOldDateFormat, and monthShortNames.

The following attributes are valid for the <tns:ftpServer> element only: passiveMode

The following attributes are valid for the <tns:ftpsServer> element only: ftpsType, trustStore,
trustStorePassword,
trustStoreType, keyStore, keyStorePassword, keyStoreType, ccc, protFirst, auth, and
connectTimeout.

The following attributes are valid for the <tns:limits> element within all of the
<tns:ftpServer>, <tns:ftpsServer>
and <tns:sftpServer> elements: maxListFileNames, maxListDirectoryLevels, maxReconnectRetry,
reconnectWaitPeriod,
maxSessions and socketTimeout

-->
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
ProtocolBridgeProperties.xsd">

    <!-- By default the location of the credentials file is in the home directory of the user
that started the -->
    <!-- protocol bridge agent. If you wish to specify a different location use the
credentialsFile element to -->
    <!-- do this. For
example:                                                                                    -->
    <!-- <tns:credentialsFile path="/test/
ProtocolBridgeCredentials.xml"/>                                    -->

    <tns:defaultServer name="WINMVSCA.HURSLEY.IBM.COM" />
    <tns:ftpServer name="WINMVSCA.HURSLEY.IBM.COM" host="WINMVSCA.HURSLEY.IBM.COM"
platform="unix"
                timeZone="Europe/London" locale="en-GB" fileEncoding="US-ASCII"
                listFormat="unix" limitedWrite="false"  />

    <!-- Define servers here -->
</tns:serverProperties>
```

此指令可以產生下列訊息：BFGCL0532I:

為了讓代理程式運作，必須手動建立其他認證檔。
By default this file is called ProtocolBridgeCredentials.xml and is located in the home
directory of the user who starts the agent. For example,if this user started the agent
the location would be: $HOME/ProtocolBridgeCredentials.xml

如果您使用認證檔:

1. 請參閱下列文字以取得有關如何建立認證檔的進一步資訊。
2. 認證檔必須位於權限受限的目錄中。 例如，其他使用者必須沒有讀取權。
3. 在 $HOME 環境變數中為已啟動代理程式的使用者 ID 指定認證檔目錄位置，或編輯 ProtocolBridgeProperties.xml 檔案並在下列行中指定位置：

```
<tns:credentialsFile path="/test/ProtocolBridgeCredentials.xml"/>
```

如果您想要新增其他非預設通訊協定伺服器，請編輯該檔案來定義其內容。 此範例新增額外一部 FTP 伺服器。

註: 通訊協定橋接器代理程式不支援檔案鎖定。 這是因為 Managed File Transfer 不支援檔案伺服器上的檔案鎖定機制。

程序

1. 在檔案中插入下列幾行來定義通訊協定檔案伺服器作為 <tns:serverProperties> 的子元素：

```
<tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234"
   platform="windows"
                            timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
                           listFormat="unix" limitedWrite="false" >
<tns:limits maxListFileNames="10" maxListDirectoryLevels="500"/>
```

2. 然後，變更屬性的值：
   - name 是通訊協定檔案伺服器的名稱
   - host 是通訊協定檔案伺服器的主機名稱或 IP 位址
   - port 是通訊協定檔案伺服器的埠號
   - platform 是執行通訊協定檔案伺服器的平台
   - timeZone 是執行通訊協定檔案伺服器的時區
   - locale 是通訊協定檔案伺服器上使用的語言
   - fileEncoding 是通訊協定檔案伺服器的字元編碼
   - listFormat 是通訊協定檔案伺服器所傳回的檔案清單格式
   - limitedWrite 決定寫入檔案伺服器時是否遵循預設模式，亦即建立暫存檔，然後在傳送完成時重新命名該檔案。 若為配置為唯寫的檔案伺服器，則會直接以其最終名稱建立檔案。 此內容值可以是 true 或 false。 如果是通訊協定橋接器代理程式，則 limitedWrite 屬性會與 doNotUseTempOutputFile 代理程式內容一起使用。 如果您想要使用暫存檔，則不得設定 doNotUseTempOutputFile 的值，而且必須將 limitedWrite 的值設為 false。 其他任何設定組合意味著不會使用暫存檔。
   - maxListFileNames 是在通訊協定檔案伺服器上的目錄中掃描檔案名稱時收集的名稱數目上限。
   - maxListDirectoryLevels 是在通訊協定檔案伺服器上的目錄中掃描檔案名稱時遞迴的目錄層次數目上限。

   如需這些屬性的詳細資料 (包括它們是必要屬性還是選用屬性及其預設值)，請參閱 通訊協定橋接器內容檔格式。

相關參考

通訊協定橋接器內容檔格式

MFT 所使用的正規表示式

查閱通訊協定檔案伺服器內容: ProtocolBridgePropertiesExit2

如果您有大量通訊協定檔案伺服器，則可以實作 com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2 介面來查閱在傳送時參照的通訊協定檔案伺服器內容。 您可以實作此介面，這更勝於維護 ProtocolBridgeProperties.xml 檔。

關於這項作業

Managed File Transfer 提供一個查閱通訊協定檔案伺服器內容的使用者結束程式範例。 如需相關資訊，請參閱 第 231 頁的『使用範例使用者結束程式來查閱通訊協定檔案伺服器內容』。

任何查閱通訊協定橋接器內容的使用者結束程式都必須實作 `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2` 介面。 如需相關資訊，請參閱 ProtocolBridgePropertiesExit.java 介面。

您可以類似其他使用者結束程式的方式，將多個通訊協定伺服器內容結束程式鏈結在一起。 會根據在代理程式內容檔中使用 `protocolBridgePropertiesExitClasses` 內容指定的順序來呼叫結束程式。 個別傳回所有起始設定方法，如果有一個以上傳回 false 值，代理程式就不會啟動。 此錯誤報告於代理程式事件日誌中。

對所有結束程式的 getProtocolServerProperties 方法，只傳回一個整體結果。 如果該方法傳回內容物件作為結果碼，則此值是傳回的結果，且不會呼叫後續結束程式的 getProtocolServerProperties 方法。 如果該方法傳回空值作為結果碼，則會呼叫下一個結束程式的 getProtocolServerProperties 方法。 如果沒有後續的結束程式，則傳回空值結果。 出現空值的整體結果碼，代表通訊協定橋接器代理程式查閱失敗。

建議您使用 ProtocolBridgePropertiesExit2.java 介面，但如果需要 ProtocolBridgePropertiesExit.java 介面的相關資訊，請參閱第 232 頁的『查閱通訊協定檔案伺服器內容: ProtocolBridgePropertiesExit』。

若要執行結束程式，請完成下列步驟:

程序

1. 編譯通訊協定伺服器內容使用者結束程式。
2. 建立包含已編譯結束程式及其套件結構的 Java 保存檔 (JAR)。
3. 將包含結束程式類別的 JAR 檔放在通訊協定橋接器代理程式的 `exits` 目錄。 此目錄位於 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* 目錄中。
4. 編輯通訊協定橋接器代理程式的內容檔，以併入 `protocolBridgePropertiesExitClasses` 內容。 以此內容的值而言，請指定以逗點區隔的類別清單，這些類別實作通訊協定橋接器伺服器內容使用者結束程式。 會根據此清單中指定的順序來呼叫結束程式類別。 如需相關資訊，請參閱 MFT agent.properties 檔案。
5. 您可以選擇性地指定 `protocolBridgePropertiesConfiguration` 內容。 您對此內容指定的值會以字串形式傳遞給 `protocolBridgePropertiesExitClasses` 指定的結束程式類別的 initialize() 方法。 如需相關資訊，請參閱 MFT agent.properties 檔案。

使用範例使用者結束程式來查閱通訊協定檔案伺服器內容

Managed File Transfer 提供一個查閱通訊協定檔案伺服器內容的使用者結束程式範例。

關於這項作業

MQ_INSTALLATION_PATH/mqft/samples/protocolBridge 目錄及 範例通訊協定橋接器內容使用者結束程式主題中提供了查閱通訊協定橋接器內容的範例使用者結束程式。

`SamplePropertiesExit2.java` 結束程式會讀取包含通訊協定伺服器內容的內容檔。 此內容檔中每一個項目的格式如下所示:

```
serverName=type://host:port
```

內容檔的位置是從通訊協定橋接器代理程式內容 `protocolBridgePropertiesConfiguration` 取得。

若要執行使用者結束程式範例，請完成下列步驟:

程序

1. 編譯 `SamplePropertiesExit2.java` 檔案。
2. 建立包含已編譯結束程式及其套件結構的 JAR 檔。

3. 將 JAR 檔放在 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent*/exits 目錄中。
4. 編輯 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/agent.properties 檔案以包含此行：

   ```
   protocolBridgePropertiesExitClasses=SamplePropertiesExit2
   ```

5. 在 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent* 目錄中建立通訊協定橋接器內容檔，例如 protocol_bridge_properties.properties。 編輯此檔案，以併入此格式的項目：

   ```
   serverName=type://host:port
   ```

6. 編輯 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent*/agent.properties 檔案以包含此行：

   ```
   protocolBridgePropertiesConfiguration=MQ_DATA_PATH/mqft/config/coordination_queue_manager/
   agents/bridge_agent/protocol_bridge_properties.properties
   ```

   您必須使用 protocol_bridge_properties.properties 檔的絕對路徑。
7. 使用 **fteStartAgent** 指令來啟動通訊協定橋接器代理程式。

相關概念

第 226 頁的『通訊協定橋接器』

通訊協定橋接器可供 Managed File Transfer (MFT) 網路存取儲存在 MFT 網路之外的檔案伺服器上的檔案（在本端網域或遠端位置）。 此檔案伺服器可使用 FTP、FTPS 或 SFTP 網路通訊協定。 每一個檔案伺服器需要至少一個專用的代理程式。 專用代理程式即所謂的通訊協定橋接器代理程式。 橋接器代理程式可以與多個檔案伺服器互動。

相關參考

ProtocolBridgePropertiesExit.java 介面

通訊協定橋接器內容使用者結束程式範例

MFT agent.properties 檔案

fteCreateBridgeAgent（建立及配置 MFT 通訊協定橋接器代理程式）

查閱通訊協定檔案伺服器內容: *ProtocolBridgePropertiesExit*

如果您有大量通訊協定檔案伺服器，則可以實作 com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit 介面來查閱在傳送時參照的通訊協定檔案伺服器內容。

關於這項作業

您可以實作 com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit 介面，以優先於維護 ProtocolBridgeProperties.xml 檔案。 建議您使用 ProtocolBridgePropertiesExit2.java 介面，但同時也支援 ProtocolBridgePropertiesExit.java 介面。 如果您已從 IBM WebSphere MQ File Transfer Edition 實作 ProtocolBridgePropertiesExit.java 介面，則可以在 IBM WebSphere MQ 7.5 或更新版本中使用該介面。 ProtocolBridgePropertiesExit2.java 中的 getCredentialLocation 方法會使用 ProtocolBridgeCredentials.xml 檔案的預設位置，這是您的起始目錄。

註: IBMWebSphereMQ File Transfer Edition (FTE) 不再是支援的產品。 如果要從 FTE 移轉至 IBM MQ 中的 Managed File Transfer 元件，請參閱 移轉 Managed File Transfer。

任何查閱通訊協定橋接器內容的使用者結束程式都必須實作 com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit 介面:

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
```

```
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     * @return {@code true} if the initialization is successful and {@code
     *         false} if unsuccessful. If {@code false} is returned from an exit
     *         the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Obtains a set of properties for the specified protocol server name.
     * <p>
     * The returned {@link Properties} must contain entries with key names
     * corresponding to the constants defined in
     * {@link ProtocolServerPropertyConstants} and in particular must include an
     * entry for all appropriate constants described as required.
     *
     * @param protocolServerName
     *            The name of the protocol server whose properties are to be
     *            returned. If a null or a blank value is specified, properties
     *            for the default protocol server are to be returned.
     * @return The {@link Properties} for the specified protocol server, or null
     *         if the server cannot be found.
     */
    public Properties getProtocolServerProperties(
            final String protocolServerName);

    /**
     * Invoked once when a protocol bridge agent is shut down. It is intended to
     * release any resources that were allocated by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     */
    public void shutdown(final Map<String, String> bridgeProperties);

}
```

您可以類似其他使用者結束程式的方式，將多個通訊協定伺服器內容結束程式鏈結在一起。 會根據在代理程式內容檔中使用 protocolBridgePropertiesExitClasses 內容指定的順序來呼叫結束程式。 個別傳回所有起始設定方法，如果有一個以上傳回 false 值，代理程式就不會啟動。 此錯誤報告於代理程式事件日誌中。

對所有結束程式的 getProtocolServerProperties 方法，只傳回一個整體結果。 如果該方法傳回內容物件作為結果碼，則此值是傳回的結果，且不會呼叫後續結束程式的 getProtocolServerProperties 方法。 如果該方法傳回空值作為結果碼，則會呼叫下一個結束程式的 getProtocolServerProperties 方法。 如果沒有後續的結束程式，則傳回空值結果。 出現空值的整體結果碼，代表通訊協定橋接器代理程式查閱失敗。

程序

若要執行結束程式，請完成下列步驟：

1. 編譯通訊協定伺服器內容使用者結束程式。
2. 建立包含已編譯結束程式及其套件結構的 Java 保存檔 (JAR)。
3. 將包含結束類別的 JAR 檔放置在通訊協定橋接器代理程式的 exits 目錄中。

   此目錄位於 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* 目錄中。

4. 編輯通訊協定橋接器代理程式的內容檔，以併入 protocolBridgePropertiesExitClasses 內容。
   以此內容的值而言，請指定以逗點區隔的類別清單，這些類別實作通訊協定橋接器伺服器內容使用者結束程式。 會根據此清單中指定的順序來呼叫結束程式類別。 如需相關資訊，請參閱 MFT agent.properties 檔案。
5. 您可以選擇性地指定 protocolBridgePropertiesConfiguration 內容。
   您對此內容指定的值會以字串形式傳遞給 protocolBridgePropertiesExitClasses 指定的結束程式類別的 initialize() 方法。 如需相關資訊，請參閱 MFT agent.properties 檔案。

對映檔案伺服器的認證

使用通訊協定橋接器代理程式的預設認證對映功能，或是撰寫您專屬的使用者結束程式，將 Managed File Transfer 中的使用者認證對映至檔案伺服器上的使用者認證。 Managed File Transfer 提供一個使用者結束程式範例，可執行使用者認證對映。

使用 *ProtocolBridgeCredentials.xml* 檔來對映檔案伺服器的認證

使用通訊協定橋接器代理程式的預設認證對映功能，將 Managed File Transfer 中的使用者認證對映至檔案伺服器上的使用者認證。 Managed File Transfer 提供 XML 檔案，您可以編輯此檔案以包括自己的認證資訊。

關於這項作業

ProtocolBridgeCredentials.xml 檔必須由使用者手動建立。 依預設，此檔案的位置是啟動通訊協定橋接器代理程式之使用者的起始目錄，但此檔案可儲存於檔案系統上可供代理程式存取的任何位置上。 若要指定不同的位置，請將<credentialsFile>元素新增至 ProtocolBridgeProperties.xml 檔案。 例如，

```
<tns:credentialsFile path="/example/path/to/ProtocolBridgeCredentials.xml"/>
```

請先編輯該檔案，併入主機、使用者及認證資訊，以設定認證對映，然後才能使用通訊協定橋接器代理程式。 如需相關資訊和範例，請參閱 通訊協定橋接器認證檔案格式。

z/OS 當您使用 IBM WebSphere MQ 7.5 或更早版本在 z/OS 平台上建立 ProtocolBridgeCredentials.xml 檔案時，必須先設定檔案標籤，然後再編輯檔案。 執行下列指令，將檔案標示為具有 ASCII 內容：

```
chtag -t -c ISO8859-1 ProtocolBridgeCredentials.xml
```

註: z/OS 在 z/OS 上，您可以將通訊協定橋接器認證檔儲存在資料集上，使用者可在此資料集中指定 .xml 檔案的名稱。

程序

1. • 編輯 <tns:server name="*server name*"> 這一行，將 name 屬性的值變更為 ProtocolBridgeProperties.xml 檔中的伺服器名稱。

   為 IBM WebSphere MQ File Transfer Edition 7.0.4 及更舊版本所建立的通訊協定橋接器代理程式，沒有 ProtocolBridgeProperties.xml 檔案（或是相關的使用者結束程式），因此對 IBM WebSphere MQ File Transfer Edition 7.0.4 Fix Pack 1 以及更新版本而言，會自動指派伺服器的主機名稱作為伺服器名稱。 Therefore, if you use an updated ProtocolBridgeCredentials.xml file with <server> entries, a name corresponding to the server's host name will match.

   您可以使用 pattern 屬性來指定使用包含萬用字元或正規表示式的伺服器名稱。 例如，

   ```
   <tns:server name="serverA*" pattern="wildcard">
   ```

2. 將使用者 ID 及認證資訊，作為 <tns:server> 的子元素插入檔案中。
   您可以在檔案中插入下列一個或多個元素：

   • 如果通訊協定檔案伺服器是 FTP、FTPS 或 SFTP 伺服器，則可以使用密碼來鑑別要求傳送的使用者。將下列字行插入檔案中：

```
<tns:user name="FTE User ID"
  serverUserId="Server User ID"
  serverPassword="Server Password">
</tns:user>
```

然後，變更屬性的值。

- name 是一個 Java 正規表示式，用來比對與 MFT 傳送要求相關聯的 MQMD 使用者 ID
- serverUserId 是傳遞給通訊協定檔案伺服器作為登入使用者 ID 的值。 如果未指定 serverUserId 屬性，則改用與 MFT 傳送要求相關聯的 MQMD 使用者 ID
- serverPassword 是與 serverUserId 相關聯的密碼。

name 屬性可以包含 Java 正規表示式。 認證對映程式會試圖將 MFT 傳送要求的 MQMD 使用者 ID 與此正規表示式相比對。 通訊協定橋接器代理程式會試圖將 MQMD 使用者 ID 與 <tns:user> 元素之 name 屬性中的正規表示式相比對（依元素存在於檔案中的順序進行）。 找到相符的項目時，通訊協定橋接器代理程式就不會再尋找是否還有其他相符的項目。 如果找到相符項目，即會將對應的 serverUserId 及 serverPassword 值傳遞給通訊協定檔案伺服器，作為登入使用者 ID 及密碼。 MQMD 使用者 ID 比對須區分大小寫。

- 如果通訊協定檔案伺服器是 SFTP 伺服器，您可以使用公開和私密金鑰來鑑別要求傳送的使用者。 在檔案中插入下列這幾行，並變更屬性的值。 <tns:user> 元素可以包含一或多個 <tns:privateKey> 元素。

```
<tns:user name="FTE User ID"
    serverUserId="Server User ID"
    hostKey="Host Key">
    <tns:privateKey associationName="association"
      keyPassword="Private key password">
    Private key file text
    </tns:privateKey>
</tns:user>
```

- name 是一個 Java 正規表示式，用來比對與 MFT 傳送要求相關聯的 MQMD 使用者 ID
- serverUserId 是傳遞給通訊協定檔案伺服器作為登入使用者 ID 的值。 如果未指定 serverUserId 屬性，則改用與 MFT 傳送要求相關聯的 MQMD 使用者 ID
- hostKey 是登入時預計會從伺服器傳回的金鑰
- key 是 serverUserId 的私密金鑰
- keyPassword 是金鑰的密碼，用來產生公開金鑰
- associationName 是用以識別追蹤及記載用途的值

name 屬性可以包含 Java 正規表示式。 認證對映程式會試圖將 MFT 傳送要求的 MQMD 使用者 ID 與此正規表示式相比對。 通訊協定橋接器代理程式會試圖將 MQMD 使用者 ID 與 <tns:user> 元素之 name 屬性中的正規表示式相比對（依元素存在於檔案中的順序進行）。 找到相符的項目時，通訊協定橋接器代理程式就不會再尋找是否還有其他相符的項目。 如果找到相符項目，則會使用對應的 serverUserId 及 key 值，向通訊協定檔案伺服器鑑別 MFT 使用者。 MQMD 使用者 ID 比對須區分大小寫。

如需對通訊協定橋接器代理程式使用私密金鑰的相關資訊，請參閱第 237 頁的『範例：如何配置通訊協定橋接器代理程式對 UNIX SFTP 伺服器使用私密金鑰認證』。

註: z/OS

當傳送要求寫入指令佇列時，如果來源代理程式指令佇列位於 z/OS 或 IBM i 系統上，MQMD 使用者 ID 可能轉換為大寫。 因此，相同原始使用者的 MQMD 使用者 ID 可能以原本的大小寫傳到認證結束程式或轉換為大寫傳到認證結束程式，端視傳送要求中指定的來源代理程式而定。 預設認證對映結束程式須區分大小寫以比對提供的 MQMD 使用者 ID，而您在對映檔案中可能必須允許此行為。

使用結束類別對映檔案伺服器的認證

如果您不想要使用通訊協定橋接器代理程式的預設認證對映功能，您可以撰寫專屬的使用者結束程式，將 Managed File Transfer 中的使用者認證對映至檔案伺服器上的使用者認證。 如果您配置認證對映使用者結束程式，它們會取代預設認證對映功能。

關於這項作業

Managed File Transfer 提供一個使用者結束程式範例，可執行使用者認證對映。 如需相關資訊，請參閱 第 236 頁的『使用範例通訊協定橋接器認證使用者結束程式』。

用來對映通訊協定橋接器認證的使用者結束程式，必須實作下列其中一個介面：

- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit`，可讓通訊協定橋接器代理程式與一部預設通訊協定檔案伺服器來回傳送檔案
- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2`，可讓您與多個端點來回傳送檔案。

`com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2` 介面包含與 `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit` 相同的函數，還包括延伸函數。 有關更多信息，請參閱 ProtocolBridgeCredentialExit .java 介面和 ProtocolBridgeCredentialExit2.java 介面。

認證結束程式可以採取類似其他使用者結束程式的方式鏈結在一起。 結束程式依照它們在代理程式內容檔中使用 `protocolBridgeCredentialConfiguration` 內容指定的順序依序呼叫。 個別傳回所有起始設定方法，如果有一個以上傳回 false 值，代理程式就不會啟動。 此錯誤報告於代理程式事件日誌中。

對所有結束程式的 mapMQUserId 方法，只會傳回一個整體結果，如下所示：

- 如果該方法傳回 USER_SUCCESSFULLY_MAPPED 或 USER_DENIED_ACCESS 值作為結果碼，則此值是傳回的結果，且不會呼叫後續結束程式的 mapMQUserId 方法。
- 如果該方法傳回 NO_MAPPING_FOUND 值作為結果碼，則會呼叫下一個結束程式的 mqMQUserId 方法。
- 如果沒有後續的結束程式，則傳回 NO_MAPPING_FOUND 結果。
- 橋接器代理程式將整體結果碼 USER_DENIED_ACCESS 或 NO_MAPPING_FOUND 視為傳送失敗。

若要執行結束程式，請完成下列步驟：

程序

1. 編譯通訊協定橋接器認證使用者結束程式。
2. 建立包含已編譯結束程式及其套件結構的 Java 保存檔 (JAR)。
3. 將包含結束類別的 JAR 檔放在橋接器代理程式的 `exits` 目錄。 此目錄位於 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* 目錄中。
4. 編輯通訊協定橋接器代理程式的內容檔，以併入 `protocolBridgeCredentialExitClasses` 內容。 以此內容的值而言，請指定以逗點區隔的類別清單，這些類別實作通訊協定橋接器認證結束常式。 會根據此清單中指定的順序來呼叫結束程式類別。 如需相關資訊，請參閱 MFT agent.properties 檔案。
5. 編輯通訊協定橋接器代理程式的內容檔，以併入：

   ```
   exitClassPath=IBM MQ
   installation_directory\mqft\config\configuration_queue_manager\agents\protocol_bridge_agent_n
   ame\exits\SampleCredentialExit.jar
   ```

   代理程式的 `agent.properties` 檔案位於 *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*bridge_agent_name* 目錄中。

   如果變更 `agent.properties` 檔案，則必須重新啟動代理程式，以讓變更生效。
6. 您可以選擇性地指定 `protocolBridgeCredentialConfiguration` 內容。 您對此內容指定的值會以「字串」物件形式傳遞給 `protocolBridgeCredentialExitClasses` 指定的結束類別的 initialize() 方法。 如需相關資訊，請參閱 MFT agent.properties 檔案。
7. 使用 **fteStartAgent** 指令來啟動通訊協定橋接器代理程式。

使用範例通訊協定橋接器認證使用者結束程式
Managed File Transfer 提供一個使用者結束程式範例，可執行使用者認證對映。

關於這項作業

MQ_INSTALLATION_PATH/mqft/samples/protocolBridge 目錄及主題 範例通訊協定橋接器認證使用者結束程式中提供範例通訊協定橋接器認證結束程式。 此範例是基於 com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit 介面。

SampleCredentialExit.java 結束程式會讀取內容檔，此內容檔將與傳送要求相關聯的 MQMD 使用者 ID 對映至伺服器使用者 ID 及伺服器密碼。 此內容檔的位置是從通訊協定橋接器代理程式內容 protocolBridgeCredentialConfiguration 中取得。

若要執行使用者結束程式範例，請完成下列步驟：

程序

1. 編譯 SampleCredentialExit.java 檔。
2. 建立包含已編譯結束程式及其套件結構的 JAR 檔。
3. 將 JAR 檔放在 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/exits 目錄中。
4. 編輯 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/agent.properties 檔案以包含此行：

   ```
   protocolBridgeCredentialExitClasses=SampleCredentialExit
   ```

5. 編輯通訊協定橋接器代理程式的內容檔，以併入：

   ```
   exitClassPath=IBM MQ
   installation_directory\mqft\config\configuration_queue_manager\agents\protocol_bridge_agent_n
   ame\exits\SampleCredentialExit.jar
   ```

   代理程式的 agent.properties 檔案位於 *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*agent_name* 目錄中。

   如果變更 agent.properties 檔案，則必須重新啟動代理程式，以讓變更生效。
6. 在 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent* 目錄中建立認證內容檔 (credentials.properties)，並編輯它以包含此格式的項目：

   ```
   mqUserId=serverUserId,serverPassword
   ```

7. 編輯 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/agent.properties 檔案以包含此行：

   ```
   protocolBridgeCredentialConfiguration=MQ_DATA_PATH/mqft/
   config/coordination_queue_manager/agents/bridge_agent_name/credentials.properties
   ```

   您必須使用 credentials.properties 檔的絕對路徑。
8. 使用 **fteStartAgent** 指令來啟動通訊協定橋接器代理程式。

範例：如何配置通訊協定橋接器代理程式對 UNIX SFTP 伺服器使用私密金鑰認證

此範例示範如何產生及配置 ProtocolBridgeCredentials.xml 檔。 此範例是一般例子，詳細資料可能隨平台而不同，但原則不變。

關於這項作業

程序

1. 產生公開和私密金鑰，以用來向 SFTP 伺服器進行鑑別。

   例如，在 Linux 主機系統上，您可以使用作為 'openssh' 套件一部分提供的工具 **ssh-keygen** 來建立公開/私密金鑰組。

依預設，在沒有引數的情況下， **ssh-keygen** 指令會提示輸入兩個金鑰檔的位置和通行詞組，其預設為名稱:

```
id_rsa      <-- Private key
id_rsa.pub  <-- Public key
```

小心: 如果您使用最新版本 OpenSSH,中的 **ssh-keygen** 命令（例如 RHEL 8 提供的命令），則所使用的密鑰格式與協定橋代理不相容，並且向 SFTP 伺服器的傳輸嘗試會失敗並顯示下列訊息:

```
BFGBR0216E: Authentication to protocol server 'sftp.host.address' failed
because of invalid private key.
```

若要建立與這些較新版本的 OpenSSH,請使用 **ssh-keygen** 指令的下列參數指定金鑰格式:

```
ssh-keygen -m PEM
```

然後， `id_rsa` 私密金鑰的內容具有下列項目的第一行和最後一行:

```
-----BEGIN RSA PRIVATE KEY-----
... ... ...
-----END RSA PRIVATE KEY-----
```

與通訊協定橋接器代理程式相容。

2. 將 `id_rsa.pub` 檔案的整個內容複製到 SFTP 伺服器上 SFTP 使用者的 `~/.ssh/authorized_keys` 檔案。

   請確保已適當地設定此檔案及 `~/.ssh` 目錄上的檔案許可權，以便 SFTP 伺服器容許金鑰鑑別。 這些許可權通常如下:

   ```
   ~/.ssh                    Mode 700
   ~/.ssh/authorized_keys    Mode 600
   ```

3. Managed File Transfer 需要使用 MD5 演算法產生的主機 ssh 指紋。 執行下列其中一個指令，以取得 SFTP 伺服器的主機 ssh 指紋。

   - 若為 Red Hat® Enterprise Linux 6.x 版及更舊版本，以及 Linux Ubuntu 14.04，請執行下列指令:

     ```
     ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
     ```

   - 從 Red Hat Enterprise Linux 7.x、 Linux Ubuntu 16.04 及 SuSE Linux 12.4 開始，依預設， ssh-keygen 指令會使用 SHA256 演算法來產生 ssh 指紋。 若要使用 MD5 演算法產生 ssh 指紋，請執行下列指令:

     ```
     ssh-keygen -l -E MD5 -f /etc/ssh/ssh_host_rsa_key.pub
     ```

     指令的輸出將類似於下列範例:

     ```
     2048 MD5:64:39:f5:49:41:10:55:d2:0b:81:42:5c:87:62:9d:27 no comment (RSA)
     ```

     僅擷取輸出的十六進位部分，以用作 `ProtocolBridgeCredentials.xml` 檔案中的 hostKey (請參閱步驟 第 238 頁的『4』)。 因此，在此範例中，您將擷取 `64:39:f5:49:41:10:55:d2:0b:81:42:5c:87:62:9d:27`。

4. 在通訊協定橋接器代理程式系統上，編輯 `ProtocolBridgeCredentials.xml` 檔。 將下列範例中以斜體顯示的值替換成您自己的值:

   ```
   <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
   xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
    ProtocolBridgeCredentials.xsd ">

   <tns:agent name="Agent_name">

   <tns:server name="SFTP_name">
   ```

```
<tns:user name="mq_User_ID" serverUserId="SFTP_user_ID"
  hostKey="ssh_host_finger">
<tns:privateKey associationName="name" keyPassword="pass_phrase">
 Complete contents of the id_rsa file including the entries
 -----BEGIN RSA PRIVATE KEY-----

 -----END RSA PRIVATE KEY-----
</tns:privateKey>
</tns:user>

</tns:server>
</tns:agent>
</tns:credentials>
```

其中:

- *Agent_name* 是通訊協定橋接器代理程式的名稱。
- *SFTP_host_name* 是 `ProtocolBridgeProperties.xml` 檔中顯示的 SFTP 伺服器名稱。
- *mq_User_ID* 是與傳送要求相關聯的 MQMD 使用者 ID。
- *SFTP_user_ID* 是步驟 2 中使用的 SFTP 使用者 ID。 它是傳遞至 SFTP 伺服器作為登入使用者 ID 的值。
- *ssh_host_finger* 是步驟 3 所收集的指紋。
- *name* 是可指定作為追蹤及記載用途的名稱。
- *pass_phrase* 是您在步驟 1 中於 ssh-keygen 中提供的密碼詞組。
- *id_rsa* 檔案的完整內容 是從步驟 1 產生之 `id_rsa` 檔案的完整內容。 若要防止連線錯誤，請確保包括下列兩個項目:

  ```
  -----BEGIN RSA PRIVATE KEY-----

  -----END RSA PRIVATE KEY-----
  ```

  您可以複製 <tns:privatekey> 元素來新增其他金鑰。

5. 啟動通訊協定橋接器代理程式（如果尚未啟動）。 或者，通訊協定橋接器代理程式會定期輪詢 `ProtocolBridgeCredentials.xml` 檔來取得變更。

將通訊協定橋接器配置用於 FTPS 伺服器

採取您配置 FTP 伺服器時的類似作法來配置 FTPS 伺服器：為伺服器建立橋接器代理程式，定義伺服器內容，然後對映使用者認證。

關於這項作業

若要配置 FTPS 伺服器，請完成下列步驟:

程序

1. 使用 **fteCreateBridgeAgent** 指令為 FTPS 伺服器建立通訊協定橋接器代理程式。 適用於 FTP 的參數也適用於 FTPS，但 FTPS 另外還有三個必要的特定參數:
   a) **-bt** 參數。 指定 FTPS 作為此參數的值。
   b) 信任儲存庫檔案的 **-bts** 參數。 此指令假設只需要伺服器鑑別，因此您必須指定信任儲存庫檔案的位置。

   依預設是使用 **fteCreateBridgeAgent** 指令來配置 FTPS 通訊協定的明確格式，但您可以變更通訊協定橋接器內容檔來配置隱含格式。 通訊協定橋接器一律以被動模式連接至 FTPS 伺服器。

   如需 **fteCreateBridgeAgent** 指令的相關資訊，請參閱 fteCreateBridgeAgent (建立及配置 MFT 通訊協定橋接器代理程式)。

   如果您需要如何建立信任儲存庫檔案的相關指示，請參閱 IBM Developer 文章：在 IBM WebSphere MQ File Transfer Edition 中配置 Secure Sockets Layer 連線功能，或請參閱 Oracle keytool 文件中的 keytool 相關資訊。

註: IBMWebSphereMQ File Transfer Edition (FTE) 不再是支援的產品。 如果要從 FTE 移轉至 IBM MQ 中的 Managed File Transfer 元件，請參閱 移轉 Managed File Transfer。

2. Define the FTPS server properties within an <ftpsServer> element in the protocol bridge properties file: `ProtocolBridgeProperties.xml`. 如需相關資訊，請參閱 第 228 頁的『使用 ProtocolBridgeProperties.xml 檔定義通訊協定檔案伺服器的內容』。 您也可以透過編輯通訊協定橋接器內容檔來啟用用戶端鑑別。 如需所有配置選項的詳細資料，請參閱 通訊協定橋接器內容檔格式。
3. 透過使用通訊協定橋接器代理程式的預設認證對映功能，或是撰寫您專屬的使用者結束程式，將 Managed File Transfer 中的使用者認證對映至 FTPS 伺服器上的使用者認證。 如需相關資訊，請參閱 第 234 頁的『對映檔案伺服器的認證』。
4. 依預設，信任儲存庫檔案會配置為 JKS 格式；如果您想要變更格式，請編輯通訊協定橋接器內容檔。

範例

以下顯示通訊協定橋接器內容檔中的 FTPS 伺服器項目範例:

```
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
       xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
       ProtocolBridgeProperties.xsd">
    <tns:defaultServer name="ftpsserver.mycompany.com" />

    <tns:ftpsServer name="ftpsserver.mycompany.com" host="ftpsserver.mycompany.com" port="990"
platform="windows"
       timeZone="Europe/London" locale="en_US" fileEncoding="UTF8"
       listFormat="unix" limitedWrite="false"
       trustStore="c:\mydirec\truststore.jks" />

    <!-- Define servers here -->
</tns:serverProperties>
```

下一步

如需支援及不支援的 FTPS 通訊協定組件的相關資訊，請參閱 通訊協定橋接器支援 FTPS 伺服器。

V 9.2.1 限制檔案傳送至個別檔案伺服器數目的實務範例及範例

已修訂的通訊協定橋接器代理程式如何使用 **maxActiveDestinationTransfers** 及 **failTransferWhenCapacityReached** 屬性，以及一些範例。

根據 maxActiveDestinationTransfers 值顯示通訊協定橋接器代理程式運作的實務範例

情境 1

通訊協定橋接器代理程式的 `ProtocolBridgeProperties.xml` 檔案包含兩個檔案伺服器定義:

- 您尚未設定廣域 **maxActiveDestinationTransfers** 屬性。
- 您尚未同時在 `fileServerA` 和 `FileServerB` 上設定 **maxActiveDestinationTransfers** 屬性。
- 您已將通訊協定橋接器代理程式 **maxDestinationTransfers** 屬性設為預設值。

如果您已將通訊協定橋接器代理程式 **maxDestinationTransfers** 屬性設為預設值 25 ，則:

- 目的地代理程式會開始處理兩個傳送至 `fileServerA` 的受管理傳送。
- 兩個傳送都已完成。

  此時，用戶端會意識到 `fileServerA` 已失敗，並在 `ProtocolBridgeProperties.xml` 檔案中為 `fileServerA` 設定下列值:

  maxActiveDestinationTransfers = 0
  failTransferWhenCapacityReached = true
- `fileServerA` 的另一個傳送會到達， `fileServerB` 的幾個傳送則會到達:

根據前一個步驟中設定的內容，受管理傳送至 fileServerA 會被拒絕並標示為失敗，而 fileServerB 的傳送則會在標準現有流程中處理。

- 一段時間之後，用戶端發現 fileServerA 正在重新執行，因此用戶端會移除或註銷先前在 ProtocolBridgeProperties.xml 中新增的值。 新的受管理傳送抵達 fileServerA ，並在標準現有流程中處理。

實務範例 2

- 您已設定檔案伺服器的 **maxActiveDestinationTransfers** 屬性，但未設定 **failTransferWhenCapacityReached** 屬性。
- 通訊協定橋接器代理程式充當此數目受管理傳送至檔案伺服器的目的地代理程式。
- **maxActiveDestinationTransfers** 屬性的值會減少 1。

通訊協定橋接器代理程式會動態更新其配置，並將 **maxActiveDestinationTransfers** 設為仍作用中的新值。 進行中的受管理傳送不受此更新影響，且容許完成。

實務範例 3

通訊協定橋接器代理程式的 ProtocolBridgeProperties.xml 檔案包含兩個檔案伺服器定義:

- 您尚未設定廣域 **maxActiveDestinationTransfers** 屬性。
- 您尚未設定 **failTransferWhenCapacityReached** 屬性。
- 您已在 fileServerA 上將 **maxActiveDestinationTransfers** 設為 1。
- 您尚未在 fileServerB 上設定 **maxActiveDestinationTransfers** 屬性。

如果通訊協定橋接器代理程式將 **maxDestinationTransfers** 屬性設為 5:

- 從通訊協定橋接器代理程式至 fileServerA 的作用中目的地傳送數目上限為 1 (雖然目的地代理程式具有 5 個目的地傳送插槽，但只有 1 個可用於 fileServerA 的受管理傳送)。

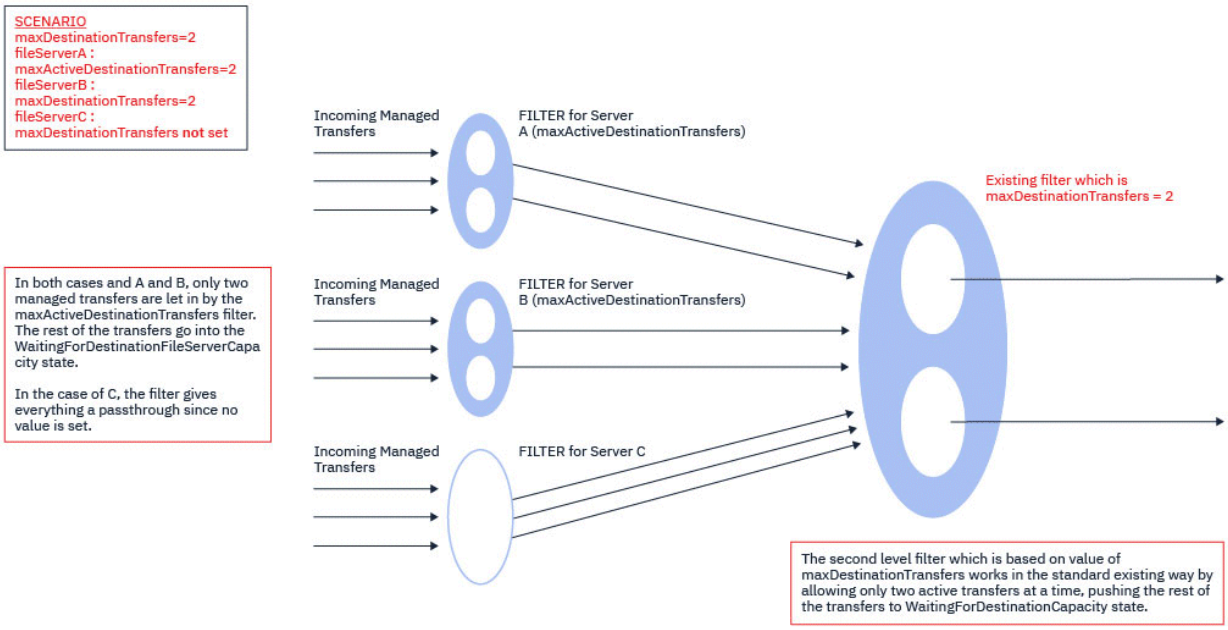
  當 fileServerA 失敗時，這很有用。 再次執行 fileServerA 之後，**maxActiveDestinationTransfers** 的值可以增加至 5 ，以允許所容許目的地傳送的完整容量。
- 從通訊協定橋接器代理程式至 fileServerB 的作用中目的地傳送數目上限為 5。

  因為未針對此檔案伺服器設定 **maxActiveDestinationTransfers** ，所以通訊協定橋接器代理程式可以使用其所有 5 個目的地傳送插槽來對其進行受管理傳送。

實務範例 4

在下圖中:

- 在 agent.properties 檔中，您已將 **maxDestinationTransfers** 屬性設為 2。
- 您已在 fileServerA 上將 **maxActiveDestinationTransfers** 設為 2。
- 您已在 fileServerB 上將 **maxActiveDestinationTransfers** 屬性設為 2。
- 您尚未在 fileServerC 上設定 **maxActiveDestinationTransfers** 屬性。



如圖所示， **maxActiveDestinationTransfers** 和 **maxDestinationTransfers** 屬性彼此獨立。

會檢查每一部伺服器的 **maxActiveDestinationTransfers** 值。 根據此值，容許進一步繼續傳送，或將傳送推送至 **WaitingForDestinationFileServerCapacity** 狀態。

然後容許的傳送會通過針對 **maxDestinationTransfers** 的現有標準檢查流程。

實務範例 5

小心: 在設定 **maxActiveDestinationTransfers** 屬性的值時應該小心，因為您必須記住 **maxDestinationTransfers** 屬性的值。

如果您不這麼做，則可能會發生下列文字中所說明的狀況:

- 您尚未設定廣域 **maxActiveDestinationTransfers** 屬性的值。
- 您已在 `agent.properties` 檔案中設定值 **maxDestinationTransfers**= 2。
- 您已在 `fileServerA` 上設定值 **maxActiveDestinationTransfers**= 2。
- 您尚未在 `fileServerB` 上設定 **maxActiveDestinationTransfers** 的值。

假設發生下列一連串事件:

- 通訊協定橋接器代理程式會接收將檔案傳送至 `fileServerA` 的要求。 通訊協定橋接器代理程式目前未執行任何動作，因此它接受此受管理傳送要求。

  現在，傳送插槽看起來如下:

  - Destination Transfers: 1
  - `fileServerA` 的目的地傳送: 1
  - `fileServerB` 的目的地傳送: 0
- 現在，通訊協定橋接器代理程式會接收另一個要求，以作為涉及 `fileServerA` 之受管理傳送的目的地代理程式。 再一次，它會接受此要求，因此傳送時段看起來如下:

  - Destination Transfers: 2
  - `fileServerA` 的目的地傳送: 2
  - `fileServerB` 的目的地傳送: 0

代理程式中的兩個 `Destination Transfer` 插槽現在已被佔用，因此在 `fileServerA` 的其中一個傳送完成之前，代理程式無法參與任何其他受管理傳送。

- 稍後 fileServerA 會失敗，這會導致兩個受管理傳送進入回復。 在此期間，這些受管理傳送所使用的 Destination transfer 插槽仍在使用中。
- 接下來，通訊協定橋接器代理程式會接收將檔案傳送至 fileServerB 的要求。 Destination Transfers for fileServerB 插槽中有此傳送的空間，不過，代理程式的所有 Destination Transfer 插槽都在使用中，因此傳送會放入待辦事項中，以便稍後可以重試。

因此，在至少一個傳送至 fileServerA 的傳送完成並釋放其 Destination Transfer 插槽之前，會封鎖傳送至 fileServerB 。

若要防止發生此狀況，請執行下列動作:

- 請將檔案伺服器上的 **maxActiveDestinationTransfers** 值設為小於 **maxDestinationTransfers** 值，以便保留可用插槽，或
- 在所有端點伺服器之間平均分配 **maxActiveDestinationTransfers** 屬性的值。

基於 maxActiveDestinationTransfers 屬性值的通訊協定橋接器代理程式行為

註: 在下表列出的所有錯誤案例中，如果 **maxActiveDestinationTransfers** 屬性設為無效的值，則通訊協定橋接器代理程式會假設未設定此屬性。

maxActiveDestinationTransfers	樣本值	說明
未指定	未指定	轉帳照常進行。 * ftp * 端點的傳送數目沒有限制。
已指定	0	不容許傳送至此特定 * ftp * 端點。
負值	-1	output0.log 值 -1 中記載的錯誤對非負整數無效。 通訊協定橋接器代理程式假設未設定屬性。
非整數值	abc	output0.log 值 abc 中記載的錯誤對整數無效。 通訊協定橋接器代理程式假設未設定屬性。
清空	""	屬性 maxActiveDestinationTransfers 的值 '' 不適用於非負整數。
已指定	5	對於此 * ftp * 端點，只容許在任何時間點執行五個作用中傳送。 根據 failTransferWhenCapacityReached 屬性的值，會重試或拒絕過多傳送。

maxActiveDestinationTransfers 與 failTransferWhenCapacityReached 屬性組合的通訊協定橋接器代理程式行為

failTransferWhenCapacity 已達到值	maxActiveDestinationTransfers 值	結果
否	3	此端點伺服器容許三個作用中傳送。 將重試任何其他傳送。
True	3	此端點伺服器容許三個作用中傳送。 任何其他傳送都會被拒絕並標示為失敗。
未指定	3	若為 failTransferWhenCapacity，則會考量預設值 **false** 。

failTransferWhenCapacity 已達到值	maxActiveDestinationTransfers 值	結果
		結果是容許三個作用中傳送至此端點伺服器。 將重試任何其他傳送。
布林值以外的值	已指定	在 output.log 中記載了錯誤。 為 failTransferWhenCapacityReached 指定的值不是布林值。 會考量 failTransferWhenCapacityReached 的預設值。

maxDestinationTransfers 與 failTransferWhenCapacityReached 屬性組合的通訊協定橋接器代理程式行為

failTransferWhenCapacity 已達到值	maxDestination 傳送值	結果
True	10	當並行作用中傳送數達到 10 時，通訊協定橋接器代理程式會導致第 11 個受管理傳送失敗。
否	10	現有行為。 當並行作用中傳送數達到 10 時，第 11 個受管理傳送會排入佇列等待釋放插槽。
未指定	10	現有行為

錯誤訊息

現有訊息:

BFGS0082I
當通訊協定橋接器代理程式拒絕傳送時，當通訊協定橋接器代理程式已執行 **maxDestinationTransfers** 屬性中定義的傳送數目上限時，記載於來源代理程式的 output0.log 檔案中。

新訊息:

BFGSS0085I
當通訊協定橋接器代理程式拒絕並重試受管理傳送時，會記載在來源代理程式的 output0.log 檔中。

BFGSS0086I
當通訊協定橋接器代理程式拒絕並重試受管理傳送，且目的地項目不包含檔案伺服器名稱時，記載在來源代理程式的 output0.log 檔案中

BFGSS0084E
當通訊協定橋接器代理程式拒絕且超出 **maxActiveDestinationTransfers** 屬性中指定的並行傳送數目上限時，會記載在 Explorer 及 audit.xml 檔案中，並將受管理傳送標示為失敗。

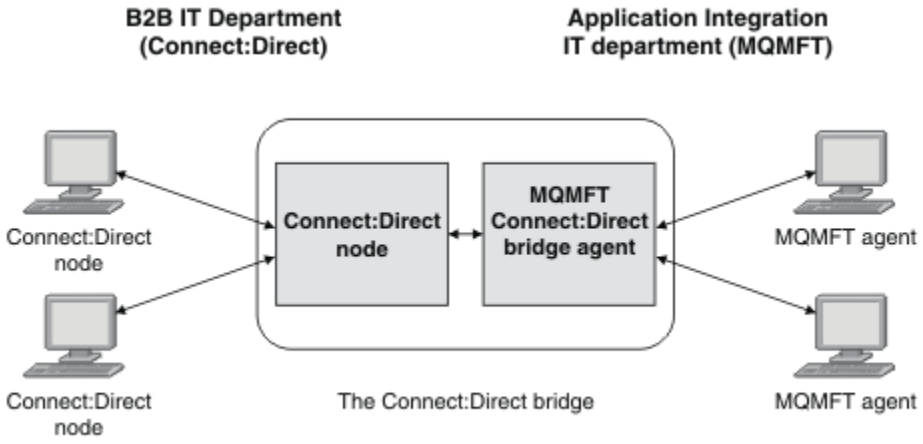
BFGSS0087E
當通訊協定橋接器代理程式拒絕且超出 **maxActiveDestinationTransfers** 屬性中指定的目的地傳送數目上限時，會記載在 Explorer 及 audit.xml 檔案中，並將受管理傳送標示為失敗。

BFGSS0088W
當 **maxActiveDestinationTransfers** 屬性的值超出 **maxDestinationTransfers** 屬性的值時，會記載在 output0.log 中。

BFGSS0089I
當目的地通訊協定橋接器代理程式所使用的來源代理程式不在 IBM MQ 9.2.1 或更新版本時，會記載在目的地通訊協定橋接器代理程式的 output0.log 檔中。

Connect:Direct 橋接器

您可以從現有 IBM Sterling Connect:Direct 網路來回傳送檔案。 使用 Connect:Direct 橋接器（Managed File Transfer 元件），可在 MFT 與 IBM Sterling Connect:Direct 之間傳送檔案。



此圖顯示兩個部門 (B2B IT 部門和「應用程式整合 IT」部門) 之間的 MFT Connect:Direct 橋接器。 B2B IT 部門使用 Connect:Direct，從公司的事業夥伴來回傳送檔案。 「應用程式整合 IT」部門使用 IBM MQ 作為其傳訊基礎架構，因此最近已選擇 Managed File Transfer 作為其檔案傳送解決方案。

透過使用 MFT Connect:Direct 橋接器，這兩個部門即可在 B2B IT 部門中的 Connect:Direct 網路與「應用程式整合 IT」部門中的 MFT 網路之間傳送檔案。 Connect:Direct 橋接器是 Managed File Transfer 的元件，其中包括與 Connect:Direct 節點通訊的 MFT 代理程式。 MFT 代理程式專用於與 Connect:Direct 節點之間的傳送，因此也稱為 Connect:Direct 橋接器代理程式。

Connect:Direct 橋接器可提供作為 Managed File Transfer 的服務和代理程式元件的一部分，並可用於下列作業：

1. 使用 Managed File Transfer 指令起始一個或多個檔案的傳送（從 MFT 代理程式至 Connect:Direct 節點）。
2. 使用 Managed File Transfer 指令起始一個或多個檔案的傳送（從 Connect:Direct 節點至 MFT 代理程式）。
3. 使用 Managed File Transfer 指令起始會啟動使用者定義 Connect:Direct 程序的檔案傳送。
4. 使用 Connect:Direct 程序提交 MFT 檔案傳送要求。

Connect:Direct 橋接器可以將檔案僅傳送至 Connect:Direct 節點，或是僅從該節點傳送檔案。 Connect:Direct 橋接器也可以僅在 Connect:Direct 程序提交的傳送過程中，將檔案傳送至其本端檔案系統，或是從該系統傳送檔案。

z/OS 您可以使用 Connect:Direct 橋接器，與 z/OS 系統上 Connect:Direct 節點上的資料集進行來回傳送。 相較於僅涉及 Managed File Transfer 代理程式的資料集傳送，此傳送在行為上存在一些差異。 如需相關資訊，請參閱 z/OS 與 Connect:Direct 節點之間來回傳送資料集。

支援的平台

Connect:Direct 橋接器由 MFT Connect:Direct 橋接器代理程式，以及 Connect:Direct 節點組成。 此代理程式在 Windows 和 Linux for x86-64 上受到支援。 在 IBM Sterling Connect:Direct for Windows 和 IBM

Sterling Connect:Direct for UNIX 支援的平台上支援此節點。 如需建立 Connect:Direct 橋接器代理程式及配置 Connect:Direct 節點以讓代理程式與之通訊的相關指示，請參閱 配置 Connect:Direct 橋接器。

Connect:Direct 橋接器可以與 Connect:Direct 節點之間來回傳送檔案，這些節點是在 Connect:Direct for Windows、Connect:Direct for UNIX z/OS 或 Connect:Direct for z/OS 服務的安裝中執行。 如需受支援 Connect:Direct 版本的詳細資料，請參閱 IBM MQ 系統需求網頁。

組成 Connect:Direct 橋接器的代理程式及節點必須在同一系統上，或可以存取同一檔案系統（例如透過共用 NFS 裝載）。 此檔案系統用於在涉及 Connect:Direct 橋接器的檔案傳送期間，將檔案暫時儲存在 **cdTmpDir** 參數所定義的目錄中。 Connect:Direct 橋接器代理程式及 Connect:Direct 橋接器節點，必須能使用相同路徑名稱，對此目錄進行定址。 例如，如果代理程式和節點是在不同的 Windows 系統上，則這些系統必須使用相同的磁碟機代號來裝載共用檔案系統。 下列配置容許代理程式及節點使用相同路徑名稱：

- 代理程式和節點是在同一個系統上，它可能執行 Windows 或 Linux for x86-64
- 代理程式在 Linux for x86-64 上，節點在 AIX 上
- 代理程式是在一個 Windows 系統上，而節點是在另一個 Windows 系統上

下列配置不容許代理程式及節點使用相同路徑名稱：

- 代理程式在 Linux for x86-64 上，節點在 Windows 上
- 代理程式是在 Windows 上，而節點是在 UNIX 上

規劃 Connect:Direct 橋接器安裝時，請考量此限制。

將檔案傳送至 Connect:Direct 節點

您可以使用 Connect:Direct 橋接器，將檔案從 Managed File Transfer 代理程式傳送至 Connect:Direct 節點。 透過下列方式，將 Connect:Direct 節點指定為傳送目的地：將 Connect:Direct 橋接器代理程式指定為目的地代理程式，並以 *connect_direct_node_name*:*file_path* 格式指定目的地檔案。

開始之前

在傳送檔案之前，您必須配置 Connect:Direct 橋接器，這是 Managed File Transfer 的元件。 如需相關資訊，請參閱 配置 Connect:Direct 橋接器。

關於這項作業

在此範例中，Connect:Direct 橋接器代理程式稱為 CD_BRIDGE。 來源代理程式稱為 FTE_AGENT，並且可以是任何 WMQFTE 版本。 目的地 Connect:Direct 節點稱為 CD_NODE1。 要傳送的檔案位於 FTE_AGENT 所在系統上的 `/home/helen/file.log` 檔案路徑中。 該檔案將傳送至執行 CD_NODE1 的系統上的 `/files/data.log` 檔案路徑中。

程序

1. 將 fteCreate 傳送指令與 **-df** (目的地檔案) 參數的值 (格式為 *connect_direct_node_name*:*file_path*) 及 **-da** (目的地代理程式) 參數的值 (指定為 Connect:Direct 橋接器代理程式的名稱) 搭配使用。

   註: *connect_direct_node_name* 指定的 Connect:Direct 節點是您要將檔案傳送至其中的節點，而不是作為 Connect:Direct 橋接器一部分運作的 Connect:Direct 節點。

   ```
   fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
     -df CD_NODE1:/files/data.log /home/helen/file.log
   ```

   如需相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。
2. 來源代理程式 FTE_AGENT 將檔案傳送至 Connect:Direct 橋接器代理程式 CD_BRIDGE。 該檔案暫時儲存在執行 Connect:Direct 橋接器代理程式所在系統上，由 cdTmpDir 代理程式內容所定義的位置中。 Connect:Direct 橋接器代理程式將該檔案傳送至 Connect:Direct 節點 CD_NODE1。

相關概念

第 245 頁的『Connect:Direct 橋接器』

您可以從現有 IBM Sterling Connect:Direct 網路來回傳送檔案。 使用 Connect:Direct 橋接器（Managed File Transfer 元件），可在 MFT 與 IBM Sterling Connect:Direct 之間傳送檔案。

相關工作

第 247 頁的『從 Connect:Direct 節點傳送檔案』
您可以使用 Connect:Direct 橋接器，將檔案從 Connect:Direct 節點傳送至 Managed File Transfer Agent 。 您可以透過下列方式，將 Connect:Direct 節點指定為傳送來源：將 Connect:Direct 橋接器代理程式指定為來源代理程式，並以 *connect_direct_node_name*:*file_path* 格式指定來源規格。

相關參考

MFT `agent.properties` 檔案

從 Connect:Direct 節點傳送檔案

您可以使用 Connect:Direct 橋接器，將檔案從 Connect:Direct 節點傳送至 Managed File Transfer Agent 。 您可以透過下列方式，將 Connect:Direct 節點指定為傳送來源：將 Connect:Direct 橋接器代理程式指定為來源代理程式，並以 *connect_direct_node_name*:*file_path* 格式指定來源規格。

開始之前

在傳送檔案之前，您必須配置 Connect:Direct 橋接器，這是 Managed File Transfer 的元件。 請參閱 配置 Connect:Direct 橋接器。

關於這項作業

在此範例中，Connect:Direct 橋接器代理程式稱為 CD_BRIDGE。 目的地代理程式稱為 FTE_AGENT，它可以是任何 Managed File Transfer 版本。 來源 Connect:Direct 節點稱為 CD_NODE1。 要傳送的檔案位於 CD_NODE1 所在系統上的 `/home/brian/in.file` 檔案路徑中。 該檔案將傳送至執行 FTE_AGENT 的系統上的 `/files/out.file` 檔案路徑中。

程序

搭配使用 **fteCreateTransfer** 指令與來源規格的值 (格式為 *connect_direct_node_name*:*file_path*)，以及指定為 Connect:Direct 橋接器代理程式名稱的 **-sa** 參數值。

註: *connect_direct_node_name* 指定的 Connect:Direct 節點是您要從中傳送檔案的節點，而不是作為 Connect:Direct 橋接器一部分運作的 Connect:Direct 節點。 例如：

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_AGENT
                  -df /files/out.file CD_NODE1:/home/brian/in.file
```

如需相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。

結果

Connect:Direct 橋接器代理程式 CD_BRIDGE 要求來自 Connect:Direct 節點 CD_NODE1 的檔案。 Connect:Direct 節點將該檔案傳送至 Connect:Direct 橋接器。 在從 Connect:Direct 節點傳送該檔案時，Connect:Direct 橋接器會將該檔案暫時儲存在 `cdTmpDir` 代理程式內容所定義的位置中。 將該檔案從 Connect:Direct 節點傳送至 Connect:Direct 橋接器完畢時，Connect:Direct 橋接器會將該檔案傳送至目的地代理程式 FTE_AGENT，然後從暫存位置刪除該檔案。

相關概念

第 245 頁的『Connect:Direct 橋接器』
您可以從現有 IBM Sterling Connect:Direct 網路來回傳送檔案。 使用 Connect:Direct 橋接器（Managed File Transfer 元件），可在 MFT 與 IBM Sterling Connect:Direct 之間傳送檔案。

相關參考

MFT `agent.properties` 檔案

z/OS 將資料集傳送至 z/OS 上的 Connect:Direct 節點

您可以使用位於 Windows 或 Linux 系統上的 Connect:Direct 橋接器，將資料集從 z/OS 上的 Managed File Transfer 代理程式傳送至 z/OS 上的 Connect:Direct 節點。

開始之前

在傳送檔案之前，您必須配置 Connect:Direct 橋接器，這是 Managed File Transfer 的元件。 請參閱 配置 Connect:Direct 橋接器。

關於這項作業

在此範例中，**-df** 參數用於指定傳送目的地。 **-df** 參數適合在傳送的來源代理程式為任何版本的 Managed File Transfer 時使用。 如果來源代理程式是 IBM WebSphere MQ File Transfer Edition 7.0.4 或更新版本，則您可以改用 **-ds** 參數。 來源代理程式稱為 FTE_ZOS1 ，並且是 IBM WebSphere MQ File Transfer Edition 7.0.3 代理程式。 Connect:Direct 橋接器代理程式稱為 CD_BRIDGE，它位於 Linux 系統上。 目的地 Connect:Direct 節點稱為 CD_ZOS2。 來源代理程式和目的地 Connect:Direct 節點都位於 z/OS 系統上。 要傳送的資料集位於 FTE_ZOS1 所在系統上的 `//FTEUSER.SOURCE.LIB` 中。 該資料集將傳送至 CD_ZOS2 所在系統上的 `//CDUSER.DEST.LIB` 資料集。

註: IBMWebSphereMQ File Transfer Edition (FTE) 不再是支援的產品。 如果要從 FTE 移轉至 IBM MQ 中的 Managed File Transfer 元件，請參閱 移轉 Managed File Transfer。

程序

1. 搭配使用 fteCreateTransfer 指令與 **-df** 參數的值，格式如下: *connect_direct_node_name*:*data_set_name*;*attributes* ，以及指定為 Connect:Direct 橋接器代理程式名稱的 **-da** (目的地代理程式) 參數值。

   connect_direct_node_name 指定的 Connect:Direct 節點是您要將資料集傳送至其中的節點，而不是作為 Connect:Direct 橋接器一部分運作的 Connect:Direct 節點。

   由 *data_set_name* 指定的資料集名稱必須是絕對（而非相對）名稱。 Connect:Direct 不會以使用者名稱作為資料集名稱的字首。

   ```
   fteCreateTransfer -sa FTE_ZOS1 -sm QM_ZOS
                     -da CD_BRIDGE -dm QM_BRIDGE
                     -df CD_ZOS2://'CDUSER.DEST.LIB;BLKSIZE(8000);LRECL(80)'
                     //'FTEUSER.SOURCE.LIB'
   ```

   如需相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。
2. 來源代理程式 FTE_ZOS1 將資料集中的資料，傳送至 Connect:Direct 橋接器代理程式 CD_BRIDGE。 該資料將作為純文字檔，暫時儲存在執行 Connect:Direct 橋接器代理程式的系統上，由 cdTmpDir 代理程式內容所定義的位置中。 Connect:Direct 橋接器代理程式將該資料傳送至 Connect:Direct 節點 CD_ZOS2。 當此傳送完成時，即會從執行 Connect:Direct 橋接器代理程式的系統刪除該純文字檔。

相關概念

第 245 頁的『Connect:Direct 橋接器』
您可以從現有 IBM Sterling Connect:Direct 網路來回傳送檔案。 使用 Connect:Direct 橋接器（Managed File Transfer 元件），可在 MFT 與 IBM Sterling Connect:Direct 之間傳送檔案。

相關工作

z/OS 與 Connect:Direct 節點來回傳送資料集

相關參考

z/OS 不得與 MFT 一起使用的 BPXWDYN 內容

將多個檔案傳送至 Connect:Direct 節點

您可以使用 Connect:Direct 橋接器，將多個檔案從 Managed File Transfer Agent 傳送至 Connect:Direct 節點。 若要將 Connect:Direct 節點用作多個檔案傳送目的地，請將 Connect:Direct 橋接器代理程式指定為目的地代理程式，並以 *connect_direct_node_name*:*directory_path* 格式指定目的地目錄。

開始之前

在傳送檔案之前，您必須配置 Connect:Direct 橋接器，這是 Managed File Transfer 的元件。 請參閱 配置 Connect:Direct 橋接器。

關於這項作業

在此範例中，來源代理程式稱為 FTE_AGENT。 Connect:Direct 橋接器代理程式稱為 CD_BRIDGE。 目的地 Connect:Direct 節點稱為 CD_NODE1。 要傳送的檔案為 FTE_AGENT 所在系統上的 `/home/jack/data.log`、`/logs/log1.txt` 及 `/results/latest`。 這些檔案將傳送至執行 CD_NODE1 的系統上的 `/in/files` 目錄。

程序

將 fteCreateTransfer 指令與 **-dd**（目的地目錄）參數的值（格式為 *connect_direct_node_name*:*directory_path*）搭配使用。 將 **-da**（目的地代理程式）參數的值，指定為 Connect:Direct 橋接器代理程式的名稱。

註: *connect_direct_node_name* 指定的 Connect:Direct 節點是您要將檔案傳送至其中的節點，而不是作為 Connect:Direct 橋接器一部分運作的 Connect:Direct 節點。

```
fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
                  -dd CD_NODE1:/in/files /home/jack/data.log
                  /logs/log1.txt /results/latest
```

如需相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。

結果

來源代理程式 FTE_AGENT 將第一個檔案，傳送至 Connect:Direct 橋接器代理程式 CD_BRIDGE。 Connect:Direct 橋接器代理程式暫時將檔案儲存在 cdTmp 目錄內容所定義的位置。 將該檔案從來源代理程式完全傳送至 Connect:Direct 橋接器後，Connect:Direct 橋接器代理程式會將該檔案，傳送至 cdNode 代理程式內容所定義的 Connect:Direct 節點。 此節點將該檔案傳送至目的地 Connect:Direct 節點 CD_NODE1。 當兩個 Connect:Direct 節點之間的傳送完成時，Connect:Direct 橋接器代理程式會從暫存位置刪除該檔案。 對於每個指定的來源檔案，將會重複執行此程序。

相關概念

第 245 頁的『Connect:Direct 橋接器』

您可以從現有 IBM Sterling Connect:Direct 網路來回傳送檔案。 使用 Connect:Direct 橋接器（Managed File Transfer 元件），可在 MFT 與 IBM Sterling Connect:Direct 之間傳送檔案。

相關工作

第 246 頁的『將檔案傳送至 Connect:Direct 節點』

您可以使用 Connect:Direct 橋接器，將檔案從 Managed File Transfer 代理程式傳送至 Connect:Direct 節點。 透過下列方式，將 Connect:Direct 節點指定為傳送目的地：將 Connect:Direct 橋接器代理程式指定為目的地代理程式，並以 *connect_direct_node_name*:*file_path* 格式指定目的地檔案。

第 250 頁的『使用萬用字元將多個檔案傳送至 Connect:Direct』

若要將多個檔案從 Managed File Transfer 代理程式傳送至 Connect:Direct 節點，請使用 Connect:Direct 橋接器。 您可以在提供給 **fteCreateTransfer** 指令的來源規格中使用萬用字元。 如同涉及萬用字元的所有 Managed File Transfer 傳送一樣，只有檔案路徑的最後部分才能包含萬用字元。 例如，`/abc/def*` 是有效的檔案路徑，而 `/abc*/def` 無效。

第 247 頁的『從 Connect:Direct 節點傳送檔案』

您可以使用 Connect:Direct 橋接器，將檔案從 Connect:Direct 節點傳送至 Managed File Transfer Agent 。 您可以透過下列方式，將 Connect:Direct 節點指定為傳送來源：將 Connect:Direct 橋接器代理程式指定為來源代理程式，並以 *connect_direct_node_name*:*file_path* 格式指定來源規格。

第 250 頁的『從 Connect:Direct 節點傳送多個檔案』

您可以使用 Connect:Direct 橋接器，將多個檔案從 Connect:Direct 節點傳送至 Managed File Transfer Agent 。 可以透過下列方式將 Connect:Direct 節點指定為多個檔案傳送的來源：將 Connect:Direct 橋接器代理程式指定為來源代理程式，並以 *connect_direct_node_name*:*file_path* 格式指定一個以上的來源規格。

相關參考

MFT agent.properties 檔案

z/OS 從 Connect:Direct 節點傳送多個檔案

您可以使用 Connect:Direct 橋接器，將多個檔案從 Connect:Direct 節點傳送至 Managed File Transfer Agent 。 可以透過下列方式將 Connect:Direct 節點指定為多個檔案傳送的來源：將 Connect:Direct 橋接器代理程式指定為來源代理程式，並以 *connect_direct_node_name*:*file_path* 格式指定一個以上的來源規格。

開始之前

在傳送檔案之前，您必須配置 Connect:Direct 橋接器，這是 Managed File Transfer 的元件。 請參閱 配置 Connect:Direct 橋接器。

關於這項作業

在此範例中，Connect:Direct 橋接器代理程式稱為 CD_BRIDGE。 目的地代理程式稱為 FTE_Z，它是在 z/OS 系統上執行。 來源 Connect:Direct 節點稱為 CD_NODE1。 要傳送的檔案為 CD_NODE1 所在系統上的 /in/file1、/in/file2 及 /in/file3 檔案路徑中。 這些檔案將傳送至執行 FTE_Z 的系統上的已分割資料集 //OBJECT.LIB。

程序

搭配使用 fteCreateTransfer 指令與來源規格的值 (格式為 *connect_direct_node_name*:*file_path*)，以及指定為 Connect:Direct 橋接器代理程式名稱的 **-sa** 參數值。

註: *connect_direct_node_name* 指定的 Connect:Direct 節點是您要從中傳送檔案的節點，而不是作為 Connect:Direct 橋接器一部分運作的 Connect:Direct 節點。

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_Z
                  -dp //'OBJECT.LIB' CD_NODE1:/in/file1
                  CD_NODE1:/in/file2 CD_NODE1:/in/file3
```

如需相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。

結果

Connect:Direct 橋接器代理程式 CD_BRIDGE 要求來自 Connect:Direct 節點 CD_NODE1 的第一個檔案。 Connect:Direct 節點將該檔案傳送至 Connect:Direct 橋接器。 在從 Connect:Direct 節點傳送該檔案時，Connect:Direct 橋接器會將該檔案暫時儲存在 cdTmpDir 代理程式內容所定義的位置中。 將該檔案從 Connect:Direct 節點傳送至 Connect:Direct 橋接器完畢時，Connect:Direct 橋接器會將該檔案傳送至目的地代理程式 FTE_Z，然後從暫存位置刪除該檔案。 對於每個指定的來源檔案，將會重複執行此程序。

相關概念

第 245 頁的『Connect:Direct 橋接器』

您可以從現有 IBM Sterling Connect:Direct 網路來回傳送檔案。 使用 Connect:Direct 橋接器（Managed File Transfer 元件），可在 MFT 與 IBM Sterling Connect:Direct 之間傳送檔案。

相關參考

MFT agent.properties 檔案

使用萬用字元將多個檔案傳送至 Connect:Direct

若要將多個檔案從 Managed File Transfer 代理程式傳送至 Connect:Direct 節點，請使用 Connect:Direct 橋接器。 您可以在提供給 **fteCreateTransfer** 指令的來源規格中使用萬用字元。 如同涉及萬用字元的所有 Managed File Transfer 傳送一樣，只有檔案路徑的最後部分才能包含萬用字元。 例如，/abc/def* 是有效的檔案路徑，而 /abc*/def 無效。

開始之前

在傳送檔案之前，您必須配置 Connect:Direct 橋接器，這是 Managed File Transfer 的元件。 如需相關資訊，請參閱 配置 Connect:Direct 橋接器。

關於這項作業

在此範例中，來源代理程式稱為 FTE_AGENT，Connect:Direct 橋接器代理程式稱為 CD_BRIDGE。 目的地 Connect:Direct 節點稱為 CD_NODE1。 要傳送的檔案為 FTE_AGENT 所在系統上的 `/reports` 目錄中。 僅會傳送名稱以 `report` 開頭，後面接著兩個字元且字尾為 `.log` 的檔案。 例如，將會傳送 `/reports/report01.log` 檔案，但不會傳送 `/reports/report1.log` 檔案。 這些檔案將傳送至執行 CD_NODE1 的系統上的 `/home/fred` 目錄。

程序

1. 將 fteCreateTransfer 指令與 **-dd**（目的地目錄）參數的值（格式為 *connect_direct_node_name*:*directory_path*）搭配使用。 對於 **-da**（目的地代理程式）參數，請指定 Connect:Direct 橋接器代理程式。

   註: *connect_direct_node_name* 指定的 Connect:Direct 節點是您要將檔案傳送至其中的節點，而不是作為 Connect:Direct 橋接器一部分運作的 Connect:Direct 節點。

   ```
   fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
                     -dd CD_NODE1:/home/fred "/reports/report??.log"
   ```

   如需相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。
2. 來源代理程式 FTE_AGENT 將符合 `/reports/report??.log` 型樣的第一個檔案，傳送至 Connect:Direct 橋接器代理程式 CD_BRIDGE。 Connect:Direct 橋接器代理程式暫時將檔案儲存在 cdTmp 目錄內容所定義的位置。 將該檔案從來源代理程式完全傳送至 Connect:Direct 橋接器後，Connect:Direct 橋接器代理程式會將該檔案，傳送至 cdNode 代理程式內容所定義的 Connect:Direct 節點。 此節點將該檔案傳送至目的地 Connect:Direct 節點 CD_NODE1。 當兩個 Connect:Direct 節點之間的傳送完成時，Connect:Direct 橋接器代理程式會從暫存位置刪除該檔案。對於符合 `/reports/report??.log` 萬用字元型樣的每個來源檔案，將會重複執行此程序。

   註: 符合 `/reports/report??.log` 型樣的檔案清單，會因來源代理程式 FTE_AGENT 所在系統的作業系統而異。

   - 如果來源代理程式位於含有 Windows 作業系統的系統上，則型樣相符將不區分大小寫。 此型樣將符合 `/reports` 目錄中，檔名格式為 `report` 後面接著兩個字元且字尾為 `.log`（不論字母大小寫為何）的所有檔案。 例如，`Report99.Log` 便是相符項。
   - 如果來源代理程式位於含有 Linux 或 UNIX 作業系統的系統上，則型樣相符將區分大小寫。 此型樣僅符合 `/reports` 目錄中，檔名格式為 `report` 後面接著兩個字元且字尾為 `.log` 的檔案。 例如，`reportAB.log` 是相符項，但 `reportAB.LOG` 及 `Report99.Log` 不是相符項。

回復及重新啟動從 Connect:Direct 節點來回進行的傳送

在傳送期間，Managed File Transfer 可能會無法連接至 IBM Sterling Connect:Direct 節點；比方說，例如該節點無法使用。 Managed File Transfer 會嘗試回復傳送，否則傳送會失敗並產生錯誤訊息。

如果 Connect:Direct 節點無法使用

如果 Connect:Direct 節點無法使用（例如，由於網路或電源中斷），Managed File Transfer 會按下列方式回復檔案傳送:

- 在此傳送要求中，如果 Managed File Transfer 先前未順利連接至 Connect:Direct 節點，則會在 **cdMaxConnectionRetries** 及 **recoverableTransferRetryInterval properties** 的值所決定的時間長度內重試傳送。 這些內容指定在 Connect:Direct 橋接器代理程式的 `agent.properties` 檔案中。 在失敗嘗試次數達到 **cdMaxConnectionRetries property** 的值之後，傳送失敗並產生錯誤訊息。 依預設，將會以 60 秒為間隔，無限地嘗試傳送。
- 在執行此傳送要求的過程中，如果 Managed File Transfer 先前曾順利連接至 Connect:Direct 節點，則會在 **cdMaxPartialWorkConnectionRetries** 及 **recoverableTransferRetryInterval** 內容值所決定的時間長度內再次嘗試傳送。 當失敗的嘗試次數達到 **cdMaxPartialWorkConnectionRetries** 內容值後，傳送會失敗並產生錯誤訊息。 依預設，將會以 60 秒為間隔，無限地嘗試傳送。

- 對於某些類型的 Connect:Direct 節點故障 (例如強制停止節點)，當節點回復時， Connect:Direct 處理程序會進入 `Held Due to Error`（HE）狀態。 在節點回復之後， Managed File Transfer 會自動回復與檔案傳送相關且狀態為 HE 的任何 Connect:Direct 處理程序。
- 如果傳送失敗，將會從管理 Connect:Direct 橋接器的系統，刪除與該傳送相關的所有暫存檔。 這些暫存檔的位置由 **cdTmpDir** 內容定義。
- 如果是從 Managed File Transfer 傳送至 Connect:Direct，且將來源處置方式指定為刪除，則在傳送失敗時不會刪除來源檔案。

如果 Connect:Direct 節點使用者認證無效

如果 Managed File Transfer 因 Connect:Direct 節點拒絕使用者認證而無法連接至該節點，傳送會失敗並產生錯誤訊息。 在此狀況下，請檢查您是否已提供適用於 Connect:Direct 節點的正確使用者認證。 如需相關資訊，請參閱 對映 Connect:Direct 的認證。

如果 Connect:Direct 橋接器代理程式無法使用

如果 Connect:Direct 橋接器代理程式無法使用，所有進行中檔案傳送的回復方式，都與標準 Managed File Transfer 傳送相同。 如需相關資訊，請參閱第 257 頁的『MFT 回復及重新啟動』。

相關概念

第 245 頁的『Connect:Direct 橋接器』
您可以從現有 IBM Sterling Connect:Direct 網路來回傳送檔案。 使用 Connect:Direct 橋接器（Managed File Transfer 元件），可在 MFT 與 IBM Sterling Connect:Direct 之間傳送檔案。

第 257 頁的『MFT 回復及重新啟動』
如果代理程式或佇列管理程式因故無法使用，例如因為電源或網路故障，在這些情況下，Managed File Transfer 會回復，如下所示:

相關工作

配置 Connect:Direct 橋接器

相關參考

MFT `agent.properties` 檔案

透過檔案傳送要求提交使用者定義的 Connect:Direct 程序

您可以提交傳送要求，以經由 Connect:Direct 橋接器代理程式進行傳送。在檔案傳送過程中，該代理程式會呼叫使用者定義的 Connect:Direct 程序。

依預設，當您提交檔案傳送要求以經由 Connect:Direct 橋接器進行傳送時，Connect:Direct 橋接器代理程式會產生 Connect:Direct 程序，用於將檔案傳送至遠端 Connect:Direct 節點，或是從該節點傳送檔案。

不過，您可以配置 Connect:Direct 橋接器代理程式，改為使用 `ConnectDirectProcessDefinition.xml` 檔案來呼叫使用者定義的 Connect:Direct 處理程序。

`ConnectDirectProcessDefinition.xml` 檔案

fteCreateCDAgent 指令在代理程式配置目錄 *MQ_DATA_PATH*`/mqft/config/`*coordination_queue_manager*`/agents/`*cd_bridge_agent_name* 中建立 `ConnectDirectProcessDefinitions.xml` 檔案。 您必須先編輯此檔案以設定程序定義，然後才能從 Connect:Direct 橋接器代理程式，呼叫使用者定義的 Connect:Direct 程序。

此檔案會定義一個以上的程序集，其中包括一個以上在傳送過程中呼叫的 Connect:Direct 程序的位置。 每個程序集均包括許多條件。 如果傳送滿足程序集的所有條件，則會使用此程序集來指定該傳送呼叫的 Connect:Direct 程序。 如需相關資訊，請參閱第 253 頁的『使用 ConnectDirectProcessDefinition.xml 檔案指定要啟動的 Connect:Direct 程序』。

本質符號變數

您可以使用 Managed File Transfer 定義的本質符號變數，將值代入使用者定義的 Connect:Direct 程序。 為了遵循 Connect:Direct 命名慣例，Managed File Transfer 使用的所有本質符號變數均會具有下列格式：%FTE 後接五個大寫英數字元。

建立程序以將檔案從 Connect:Direct 節點傳送至 Connect:Direct 橋接器系統時，您必須在 Connect:Direct 程序中，將本質變數 %FTETFILE 用作 TO FILE 的值。 建立程序以將檔案從 Connect:Direct 橋接器系統傳送至 Connect:Direct 節點時，您必須在 Connect:Direct 程序中，將本質變數 %FTEFFILE 用作 FROM FILE 的值。 這些變數包含 Connect:Direct 橋接器代理程式在 Managed File Transfer 網路中進行傳入及傳出時，所用的暫存檔路徑。

如需本質符號變數的相關資訊，請參閱 Connect:Direct 產品文件。

範例 Connect:Direct 程序

Managed File Transfer 提供範例 Connect:Direct 程序。 這些範例位於下列目錄中：*MQ_INSTALLATION_PATH*/mqft/samples/ConnectDirectProcessTemplates。

使用 *ConnectDirectProcessDefinition.xml* 檔案指定要啟動的 *Connect:Direct* 程序

指定要在 Managed File Transfer 傳送中啟動哪個 Connect:Direct 處理程序。 Managed File Transfer 提供 XML 檔案，您可以編輯此檔案以指定程序定義。

關於這項作業

fteCreateCDAgent 指令在代理程式配置目錄 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name* 中建立 ConnectDirectProcessDefinitions.xml 檔案。 您必須先編輯此檔案以設定程序定義，然後才能從 Connect:Direct 橋接器代理程式，呼叫使用者定義的 Connect:Direct 程序。

對於您要指定在透過 Connect:Direct 橋接器進行傳送過程中呼叫的每個程序，請執行下列步驟：

程序

1. 定義您要讓 Connect:Direct 橋接器代理程式，在傳送過程中呼叫的 Connect:Direct 程序，然後將程序範本儲存在檔案中。
2. 在文字編輯器中開啟 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name*/ConnectDirectProcessDefinitions.xml 檔案。
3. 建立 <processSet> 元素。
4. 在 <processSet> 元素內，建立 <condition> 元素。
5. 在 <condition> 元素內，建立一個以上元素來定義傳送要求必須符合的條件，以呼叫您在步驟 1 中定義的 Connect:Direct 程序。 這些元素可以是 <match> 元素或 <defined> 元素。
   - 使用 <match> 元素可指定變數的值必須符合某個型樣。 請建立具有下列屬性的 <match> 元素：
     - variable - 要比較其值的變數名稱。 此變數為本質符號。 如需相關資訊，請參閱 與使用者定義的 Connect:Direct 處理程序搭配使用的替代變數。
     - value - 要與指定變數值進行比較的型樣。
     - 選用項目： pattern - value 屬性值所用的型樣類型。 此型樣類型可以是 wildcard 或 regex。此為選用屬性，預設值為 wildcard。
   - 使用 <defined> 元素可指定必須為變數定義值。 請建立具有下列屬性的 <defined> 元素：
     - variable - 必須為其定義值的變數名稱。 此變數為本質符號。 如需相關資訊，請參閱 與使用者定義的 Connect:Direct 處理程序搭配使用的替代變數。

   在 <condition> 元素內指定的條件，可使用邏輯 AND 結合在一起。 必須符合所有條件，Connect:Direct 橋接器代理程式才能呼叫此 <processSet> 元素指定的程序。 如果您沒有指定 <condition> 元素，此程序集將符合所有傳送。
6. 在 <processSet> 元素內，建立 <process> 元素。

7. 在 <process> 元素內，建立 <transfer> 元素。

   transfer 元素可指定 Connect:Direct 橋接器代理程式，在傳送過程中呼叫的 Connect:Direct 程序。 請建立具有下列屬性的 <transfer> 元素：

   - process- -您在步驟 1 中定義之 Connect:Direct 處理程序的位置。 這個檔案的位置是以絕對路徑或相對於 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name* 目錄來指定。

結果

在搜尋條件相符項時，Connect:Direct 橋接器代理程式會從檔案開頭開始搜尋到檔案結尾。 找到的第一個相符項，即為所用的相符項。

相關工作

配置 Connect:Direct 橋接器

相關參考

Connect:Direct 程序定義檔格式

fteCreateCDAgent：建立 Connect:Direct 橋接器代理程式

在 *Managed File Transfer* 所呼叫的 *Connect:Direct* 程序中使用本質符號變數

您可以在程序定義中使用本質符號變數，從 Managed File Transfer 傳送呼叫使用者定義的 Connect:Direct 程序，並將資訊從傳送傳入 Connect:Direct 程序。

關於這項作業

此範例使用本質符號變數，將資訊從 Managed File Transfer 傳送中傳入使用者定義的 Connect:Direct 程序。 如需 Managed File Transfer 所使用本質符號變數的相關資訊，請參閱 與使用者定義 Connect:Direct 處理程序搭配使用的替代變數。

在此範例中，檔案將從 Managed File Transfer Agent 傳送至 Connect:Direct 橋接器節點。 該傳送的第一部分由 Managed File Transfer 執行。 該傳送的第二部分由使用者定義的 Connect:Direct 程序執行。

程序

1. 建立使用本質符號變數的 Connect:Direct 程序。

```
%FTEPNAME PROCESS
    SNODE=%FTESNODE
    PNODEID=(%FTEPUSER,%FTEPPASS)
    SNODEID=(%FTESUSER,%FTESPASS)

COPY001 COPY
    FROM (
        FILE=%FTEFFILE
        DISP=%FTEFDISP
    )
    TO (
        FILE=%FTETFILE
        DISP=%FTETDISP
    )
PEND
```

2. 將此程序儲存至位於下列位置的文字檔：*MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent*/Example.cdp
3. 編輯 ConnectDirectProcessDefinition.xml 檔案，以包含呼叫您在步驟 1 中建立的 Connect:Direct 程序的規則。

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
               xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://wmqfte.ibm.com/
                ConnectDirectProcessDefinitions ConnectDirectProcessDefinitions.xsd">


    <tns:processSet>
        <tns:condition>
            <tns:match variable="%FTESNODE" value="TOBERMORY" pattern="wildcard" />
        </tns:condition>
```

```
        <tns:process>
            <tns:transfer process="Example.cdp" />
        </tns:process>
    </tns:processSet>

</tns:cdprocess>
```

在此範例中，如果提交至 Connect:Direct 橋接器代理程式的傳送要求，以 TOBERMORY 作為其來源或目的地 Connect:Direct 節點，則會呼叫 `Example.cdp` Connect:Direct 程序。

4. 提交滿足您在步驟 3 中於 `ConnectDirectProcessDefinition.xml` 檔案內所定義條件的檔案傳送要求。

   例如，

   ```
   fteCreateTransfer -sa ORINOCO -da CD_BRIDGE
                     -sm QM_WIMBLEDON -dm QM_COMMON
                     -de overwrite -df TOBERMORY:/home/bulgaria/destination.txt
                     -sd leave c:\bungo\source.txt
   ```

   在此範例中，目的地 Connect:Direct 節點為 TOBERMORY。 此節點是該傳送中的次要節點，因此 %FTESNODE 的值設定為 TOBERMORY。 此指令會比對在 `ConnectDirectProcessDefinition.xml` 檔案中設定的條件。

5. Managed File Transfer 將來源檔案傳送至與 Connect:Direct 橋接器代理程式所處相同系統上的暫存位置。
6. Connect:Direct 橋接器代理程式根據傳送要求中的資訊以及配置資訊，設定本質符號變數的值。

   本質符號變數會設定為下列值：

   - %FTEPNAME=*process_name* - 此值是 Connect:Direct 橋接器代理程式產生的 8 字元程序名稱。
   - %FTESNODE=TOBERMORY - 此值將根據 **fteCreateTransfer** 指令的 **-df** 參數來設定。
   - %FTEPUSER,=*primary_node_user* - 此資訊將從 `ConnectDirectCredentials.xml` 檔案取得。
   - %FTEPPASS=*primary_node_user_password* - 此資訊將從 `ConnectDirectCredentials.xml` 檔案取得。
   - %FTESUSER,=*secondary_node_user* - 此資訊將從 `ConnectDirectCredentials.xml` 檔案取得。
   - %FTESPASS=*secondary_node_user_password* - 此資訊將從 `ConnectDirectCredentials.xml` 檔案取得。
   - %FTEFFILE =*temporary_location* - 此值是與 Connect:Direct 橋接器代理程式所處相同系統上的檔案暫存位置。
   - %FTEFDISP=leave - 此值將根據 **fteCreateTransfer** 指令的 **-sd** 參數來設定。
   - %FTETFILE=`/home/bulgaria/destination.txt` - 此值將根據 **fteCreateTransfer** 指令的 **-df** 參數來設定。
   - %FTETDISP=overwrite - 此值將根據 **fteCreateTransfer** 指令的 **-de** 參數來設定。
7. Connect:Direct 程序在 Connect:Direct 橋接器節點上啟動。 Connect:Direct 會將檔案從 Connect:Direct 橋接器系統上的暫存位置，傳送至 Connect:Direct 節點 TOBERMORY 執行所在系統上的目的地 `/home/bulgaria/destination.txt` 。

相關概念

第 252 頁的『透過檔案傳送要求提交使用者定義的 Connect:Direct 程序』
您可以提交傳送要求，以經由 Connect:Direct 橋接器代理程式進行傳送。在檔案傳送過程中，該代理程式會呼叫使用者定義的 Connect:Direct 程序。

相關參考

與使用者定義的 Connect:Direct 程序搭配使用的替代變數

使用 Connect:Direct 程序提交 Managed File Transfer 傳送要求

您可以透過 Connect:Direct 程序，將傳送要求提交至 Connect:Direct 橋接器代理程式。 Managed File Transfer 提供可在 Connect:Direct 處理程序中從 **RUN TASK** 陳述式呼叫的指令。

Managed File Transfer 提供下列可與 Connect:Direct 程序搭配使用的指令：

ftetag
在 **ftebxfer** 或 **ftecxfer** 指令之前的步驟中指定此指令，可以建立傳送所需的審核資訊。 此指令會將傳送的來源規格視為參數。 如需來源規格格式的相關資訊，請參閱 **fteCreateTransfer**: 開始新的檔案傳送。

ftebxfer
如果要向其提交傳送要求的佇列管理程式，與提交指令的 Connect:Direct 節點位於相同系統上，請指定此指令以建立檔案傳送要求。 此指令會採用與 **fteCreateTransfer** 指令相同的參數。 如需這些參數的相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。 此指令還具有以下一個其他參數：

-qmgrname
必要項目。 要向其提交該指令的佇列管理程式名稱。

ftecxfer
如果要向其提交傳送要求的佇列管理程式，與提交指令的 Connect:Direct 節點位於不同系統上，請指定此指令以建立檔案傳送要求。 此指令會採用與 **fteCreateTransfer** 指令相同的參數。 如需參數的相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送。 此指令還具有以下三個其他參數：

-qmgrname
必要項目。 要向其提交該指令的佇列管理程式名稱。

-connname
必要項目。 要向其提交該指令的佇列管理程式的主機及埠（以 IBM MQ CONNAME 格式指定）。 例如，`host.example.com(1337)`。

-channelname
選用項目。 用來連接至要向其提交該指令的佇列管理程式的通道名稱。 如果未指定此參數，則會使用預設值 SYSTEM.DEF.SVRCONN。

相關工作
第 256 頁的『使用「Connect:Direct 要求端」建立及提交呼叫 Managed File Transfer 的 Connect:Direct 程序』
「Connect:Direct 要求者」是一個圖形使用者介面，可用來建立及提交呼叫 Managed File Transfer 的 Connect:Direct 程序。

相關參考
範例：呼叫 MFT 指令的 Connect:Direct 程序檔案

使用「*Connect:Direct* 要求端」建立及提交呼叫 *Managed File Transfer* 的 *Connect:Direct* 程序

「Connect:Direct 要求者」是一個圖形使用者介面，可用來建立及提交呼叫 Managed File Transfer 的 Connect:Direct 程序。

關於這項作業

此作業說明如何建立 Connect:Direct 處理程序，以呼叫 Managed File Transfer **ftecxfer** 指令或 **ftebxfer** 指令。 當提交傳送要求的目標佇列管理程式與提交指令的 Connect:Direct 節點位於不同的系統時，請使用 **ftecxfer** 指令。 當提交傳送要求的目標佇列管理程式與提交指令的 Connect:Direct 節點位於相同系統時，請使用 **ftebxfer** 指令。 **ftecxfer** 指令會與傳送來源代理程式的代理程式佇列管理程式建立用戶端連線。 在呼叫 **ftecxfer** 指令之前，您必須呼叫 **ftetag** 指令，並為其傳遞來源規格資訊。 讓您可以使用與從 Managed File Transfer 起始的傳送相同的方式，來記載及審核程序。

程序

1. 啟動「Connect:Direct 要求端」。
2. 在畫面的**節點**標籤中，選取用作程序主要節點的 Connect:Direct 節點。
3. 選取**檔案** > **新建** > **程序**。 「**程序內容**」視窗即會開啟。
4. 在**名稱：**欄位中，鍵入程序的名稱。
5. 從 **Snode** > **名稱：**清單中選取次要節點。
6. 從 **Snode** > **作業系統：**清單中選取次要節點的作業系統。

7. 選擇性的: 在此視窗中完成您所需的任何進一步資訊。
8. 按一下**確定**。 「**程序內容**」視窗即會關閉。
9. 建立執行 Managed File Transfer **ftetag** 指令的陳述式。
   a) 在「**程序**」視窗中，用滑鼠右鍵按一下 **End** 陳述式。
   b) 選取**插入 > 執行作業**。 「**執行作業陳述式**」視窗即會開啟。
   c) 在**標籤：**欄位中，鍵入 Tag。
   d) 在 **選用參數或指令** 欄位中，鍵入 pgm(*MQ_INSTALLATION_PATH*/bin/ftetag) args(*source_specification*)。 如需 *source_specification* 格式的相關資訊，請參閱 **fteCreateTransfer**: 開始新的檔案傳送。
   e) 按一下**確定**。 「**執行作業陳述式**」視窗即會關閉。
10. 建立執行 Managed File Transfer **ftecxfer** 或 **ftebxfer** 指令的陳述式。
    a) 在「**程序**」視窗中，用滑鼠右鍵按一下 **End** 陳述式。
    b) 選取**插入 > 執行作業**。 「**執行作業陳述式**」視窗即會開啟。
    c) 在**標籤：**欄位中，鍵入 Transfer。
    d) 在 **選用參數或指令** 欄位中，輸入 pgm(*MQ_INSTALLATION_PATH*/bin/ftecxfer) args(*parameters*) 或 pgm(*MQ_INSTALLATION_PATH*/bin/ftebxfer) args(*parameters*)，視您選擇的指令而定。 **ftecxfer** 和 **ftebxfer** 指令所使用的參數，與 **fteCreateTransfer** 指令所使用的參數相同，但 **ftecxfer** 和 **ftebxfer** 另有一些專用參數。 如需相關資訊，請參閱 **fteCreateTransfer**: 啟動新的檔案傳送 和 第 255 頁的『使用 Connect:Direct 程序提交 Managed File Transfer 傳送要求』。
    e) 按一下**確定**。 「**執行作業陳述式**」視窗即會關閉。
11. 選擇性的: 建立您需要的任何其他陳述式。
12. 提交程序。
    a) 在「**程序**」視窗中按一下滑鼠右鍵。
    b) 選取**提交**。 「**Connect:Direct 連接**」視窗即會開啟。
    c) 輸入用於執行程序的使用者名稱及密碼。
    d) 按一下**確定**。

相關概念

第 255 頁的『使用 Connect:Direct 程序提交 Managed File Transfer 傳送要求』
您可以透過 Connect:Direct 程序，將傳送要求提交至 Connect:Direct 橋接器代理程式。 Managed File Transfer 提供可在 Connect:Direct 處理程序中從 **RUN TASK** 陳述式呼叫的指令。

從 IBM Integration Bus 使用 MFT

您可以使用 FTEOutput 和 FTEInput 節點，從 IBM Integration Bus 使用 Managed File Transfer 。

- 使用 FTEInput 節點，在應用 Managed File Transfer 的網路上傳送檔案，然 後在 Integration Bus 流程期間處理該檔案。
- 使用 FTEOutput 節點，將 Integration Bus 流程輸出檔案，傳輸到網路上的其他位置。

與分配管理系統代理程式來回傳送檔案的代理程式，可以是任何層次的 Managed File Transfer。

如需相關資訊，請參閱 IBM Integration Bus 產品說明文件。

MFT 回復及重新啟動

如果代理程式或佇列管理程式因故無法使用，例如因為電源或網路故障，在這些情況下，Managed File Transfer 會回復，如下所示:

- 一般而言，如果檔案在傳送時中發生問題，在解決問題後，Managed File Transfer 會回復並重新啟動該檔案傳送。
- 如果傳送過程中的檔案被刪除或變更，而代理程式或佇列管理程式在當時無法使用，則傳送會失敗，且傳送日誌中會出現一個訊息提供失敗的詳細資料。

- 如果代理程式程序在檔案傳送期間失敗，則重新啟動代理程式後，就會繼續傳送。
- 如果代理程式與其代理程式佇列管理程式的連線中斷，代理程式會一直等待嘗試重新連接至佇列管理程式。 當代理程式順利重新連接至其佇列管理程式時，現行傳送會繼續進行。
- 如果代理程式因故而停止，則任何與代理程式相關聯的資源監視器都會停止輪詢。 當代理程式回復時，監視器也會重新啟動，資源輪詢會回復。
- 對於來源處置為**刪除**的檔案傳送，如果在所有資料從來源代理程式傳送至目的地代理程式之後進行回復，則來源檔會在刪除之前解除鎖定。 這個解除鎖定動作表示來源檔在檔案刪除之前可能遭到修改。 因此會將刪除來源檔視為不安全的作業，而顯示下列警告：

```
BFGTR0075W: The source file has not been deleted because it is possible that the source file
was modified after the source file was transferred.
```

  在此情況下，請驗證來源檔的內容是否未經修改，然後再手動刪除來源檔。

您可以在「IBM MQ Explorer」中檢查傳送的狀態。 如果任何傳送顯示為 `Stalled`，則您可能需要採取更正動作，因為停滯狀態表示代理程式或傳送所涉及的兩個代理程式之間發生問題。

相關工作

第 258 頁的『設定回復停滯傳送的逾時』

您可以針對套用至來源代理程式所有傳送的已停滯檔案傳送，設定傳送回復逾時。 您也可以設定個別傳送的傳送回復逾時。 如果您設定特定時間量 (以秒為單位)，在此期間，來源代理程式會持續嘗試回復已停滯的檔案傳送，且當代理程式達到逾時時，傳送不會成功，則傳送會失敗。

設定回復停滯傳送的逾時

您可以針對套用至來源代理程式所有傳送的已停滯檔案傳送，設定傳送回復逾時。 您也可以設定個別傳送的傳送回復逾時。 如果您設定特定時間量 (以秒為單位)，在此期間，來源代理程式會持續嘗試回復已停滯的檔案傳送，且當代理程式達到逾時時，傳送不會成功，則傳送會失敗。

關於這項作業

從 IBM MQ 9.1 開始，您可以將傳送回復逾時設定為套用至來源代理程式的所有傳送，方法是將傳送回復逾時參數新增至代理程式的 `agent.properties` 檔案。 您也可以從指令行設定個別傳送的傳送回復逾時，或使用 IBM MQ Explorer 或使用 Apache Ant 作業來設定個別傳送的傳送回復逾時。 如果在 `agent.properties` 檔案中設定了傳送回復逾時值，則個別傳送的設定傳送回復逾時值會置換 `agent.properties` 檔案中的值。

傳送回復逾時有三個選項：

- 代理程式會繼續嘗試回復已停滯的傳送，直到順利完成為止。 這與未設定傳送回復逾時時代理程式的預設行為相同。
- 代理程式會在進入回復時立即將傳送標示為失敗。
- 在傳送標示為失敗之前，代理程式會持續重試停滯的傳送達指定的時間量。

設定檔案傳送回復逾時是選用的。 如果您未設定它，則傳送會遵循預設行為。 這與 IBM MQ 9.1 之前 Managed File Transfer 來源代理程式的預設行為相同，代理程式會繼續嘗試回復已停滯的傳送，直到傳送成功為止。

相關概念

第 257 頁的『MFT 回復及重新啟動』

如果代理程式或佇列管理程式因故無法使用，例如因為電源或網路故障，在這些情況下，Managed File Transfer 會回復，如下所示：

傳送回復逾時概念

您可以設定時間量（以秒為單位），在此期間內，來源代理程式會一直嘗試回復已停止的檔案傳送。 如果代理程式達到重試間隔的逾時值時傳送未順利完成，則傳送失敗。

回復逾時優先順序

透過 **fteCreateTransfer**、 **fteCreateTemplate** 或 **fteCreateMonitor** 指令指定，或使用 IBM MQ Explorer 指定，或在 **fte:filespec** 巢狀元素中指定，個別傳送的傳送回復逾時值，優先於在來源代理程式的 agent.properties 檔案中為 **transferRecoveryTimeout** 參數指定的值。

例如，如果啟動 **fteCreateTransfer** 指令時沒有 **-rt** 參數和值配對，則來源代理程式 AGENT1 會檢查 agent.properties 檔案中是否有 **transferRecoveryTimeout** 值，以判定回復逾時行為:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

如果 agent.properties 檔案中的 **transferRecoveryTimeout** 參數未設定或設為 -1，代理程式會遵循預設行為，並嘗試回復傳送，直到成功為止。

不過，如果 **fteCreateTransfer** 指令包括 **-rt** 參數，則此參數的值優先於 agent.properties 檔案中的值，並用作傳送的回復逾時設定:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt 21600 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

回復逾時計數器

回復逾時計數器是在傳送進入回復狀態時啟動。 傳送日誌訊息會透過主題字串 Log/agent_name/transfer_ID 發佈至 SYSTEM.FTE 主題，以指出傳送狀態已變更為回復中，以及狀態變更時的來源代理程式時鐘時間。 如果在設定的重試間隔內回復傳送，但未達到回復逾時 (counter<=recovery timeout)，則計數器將重設為 0，並準備在傳送進入回復時重新啟動。

如果計數器達到為回復逾時設定的最大值 (counter==recovery timeout)，則回復傳送將停止，且來源代理程式會報告此傳送失敗。 此類型的傳送失敗 (由傳送達到回復逾時的事實所導致) 由訊息碼 RECOVERY TIMEOUT (69)指出。 另一個傳送日誌訊息已發佈至 SYSTEM.FTE 主題，主題字串為 Log/agent_name/transfer_ID，指出傳送失敗，並包括訊息、回覆碼及來源代理程式事件日誌。 在回復期間發生下列任何事件時，會以訊息更新來源代理程式的事件日誌:

- 當回復逾時參數設為大於 -1 的值時，傳送會進入回復。 代理程式的事件日誌將更新，以指出 **TransferId** 的回復計時器開始，以及來源代理程式在起始回復逾時處理之前等待的時間量。
- 回復傳送時，會以新訊息更新來源代理程式的事件日誌，以指出回復中的 **TransferId** 。
- 當回復傳送逾時時，會更新來源代理程式的事件日誌，以指出在回復時因回復逾時而失敗的 **TransferId** 。

這些日誌訊息讓使用者（訂閱者和日誌程式）能夠識別因為傳送回復逾時而失敗的傳送。

回復逾時的計數器一律位於來源代理程式。 但是，如果目的地代理程式無法及時接收到來源代理程式的資訊，它會向來源代理程式傳送要求，將此傳送置於回復狀態。 對於設定了回復逾時選項的傳送，來源代理程式會在接收到目的地代理程式的要求時啟動回復逾時計數器。

對於不使用回復逾時選項的傳送、失敗的傳送和部分完成的傳送，仍需要手動處理。

對於向多個檔案發出單一傳送要求的傳送集，其中某些檔案已順利完成，但有一個檔案僅局部完成，仍會標示為失敗的傳送，因為傳送沒有按照預期完成。 在傳送局部完成的檔案時，來源代理程式可能已逾時。

確定目的地代理程式和檔案伺服器已經準備妥當，並處於可接受檔案傳送的狀態。

您必須再次為整個集發出傳送要求，但為了避免因初次傳送嘗試後還有一些檔案留在目的地而導致有問題，發出新要求時您可以指定**改寫現有檔案**選項。 這可確保前次傳送嘗試的不完整檔案集在新的傳送中予以清除，然後再重新寫入至目的地。

V 9.2.0 從 IBM MQ 9.1.5 開始，在起始傳送嘗試失敗之後，不再需要手動移除留在目的地的組件檔。 如果設定傳送的傳送回復逾時，如果傳送回復逾時，則來源代理程式會將傳送移至 RecoveryTimedOut 狀態。 重新同步化傳送之後，目的地代理程式會移除在傳送期間建立的任何組件檔，並將完成訊息傳送至來源代理程式。

追蹤資料和訊息

會基於診斷目的而包含追蹤點。 會記載回復逾時值、重試間隔開始、回復期間開始及計數器重設，以及傳送是否逾時及失敗。 萬一發生問題或出現非預期的行為，您可以收集來源代理程式輸出日誌和追蹤檔案，並在 IBM 支援人員要求時提供它們，以協助進行疑難排解。

在下列情況下，訊息會通知您:

- 傳送進入回復 (BFGTR0081I)
- 傳送已終止，因為它從回復中逾時 (BFGSS0081E)
- 在回復之後回復 Atransfer (BFGTR0082I)

相關概念

第 257 頁的『MFT 回復及重新啟動』
如果代理程式或佇列管理程式因故無法使用，例如因為電源或網路故障，在這些情況下，Managed File Transfer 會回復，如下所示:

設定一個來源代理程式之所有傳送的傳送回復逾時

您可以透過將 **transferRecoveryTimeout** 參數新增至 agent.properties 檔案，來設定適用於來源代理程式所有傳送的傳送回復逾時。

關於這項作業

若要設定適用於來源代理程式之所有傳送的樹狀結構傳送回復逾時，請將 **transferRecoveryTimeout** 的參數及值配對新增至 agent.properties 檔案。

transferRecoveryTimeout 參數有三個選項:

-1
代理程式繼續嘗試回復已停止的傳送，直到傳送順利完成為止。 使用此選項相當於代理程式在未設定此內容時的預設行為。

0
一旦進入回復，代理程式即停止檔案傳送。

>0
代理程式繼續嘗試回復已停止的傳送，直至達到已指定的正整數值所設定的時間量（以秒為單位）為止。

只有在重新啟動代理程式之後，您對 agent.properties 檔案所做的任何變更才會生效。

必要的話，您可以針對個別傳送置換 agent.properties 檔案中的傳送回復逾時值。 如需相關資訊，請參閱 第 261 頁的『設定個別傳送的傳送回復逾時』。

程序

- 若要指定代理程式繼續嘗試回復已停滯的傳送，直到順利完成為止，請將傳送回復逾時值設為 -1，如下列範例所示:

  ```
  transferRecoveryTimeout=-1
  ```

- 若要指定代理程式在進入回復時立即將傳送標示為失敗，請將傳送回復逾時值設為 0，如下列範例所示:

  ```
  transferRecoveryTimeout=0
  ```

- 若要指定代理程式在將傳送標示為失敗之前持續重試停滯的傳送達給定的時間量，請針對您要代理程式持續重試的時間量 (以秒為單位) 設定傳送回復逾時值。

  例如，將傳送回復逾時值設為 21600 表示代理程式在進入回復時，會持續嘗試回復傳送六小時:

  ```
  transferRecoveryTimeout=21600
  ```

  此參數的最大值為 999999999。

設定個別傳送的傳送回復逾時

您可以從指令行或使用 IBM MQ Explorer，或使用 Apache Ant 作業，來設定個別傳送的傳送回復逾時。 如果在 agent.properties 檔案中設定了傳送回復逾時值，則設定個別傳送的傳送回復逾時值會置換在 agent.properties 檔案中設定的值。

關於這項作業

在下列情況下，您可以設定個別傳送的傳送回復逾時參數:

- 使用 **fteCreateTransfer** 指令或使用 IBM MQ Explorer 來建立傳送。
- 使用 **fteCreateTemplate** 指令或使用 IBM MQ Explorer 來建立傳送範本。
- 使用 **fteCreateMonitor** 指令或使用 IBM MQ Explorer 來建立資源監視器。
- 使用 fte: filecopy 或 fte: filemove Ant 作業複製或移動檔案。

如果您設定個別傳送的傳送回復逾時值，則此值會置換 agent.properties 檔案中設定的傳送回復逾時值 (請參閱第 260 頁的『設定一個來源代理程式之所有傳送的傳送回復逾時』)。

程序

- 若要使用 **fteCreateTransfer** 或 **fteCreateTemplate** 指令來設定傳送回復逾時，請為 **-rt** 參數指定適當的選項:

  -1
  代理程式繼續嘗試回復已停止的傳送，直到傳送順利完成為止。 使用此選項相當於代理程式在未設定此內容時的預設行為。

  0
  一旦進入回復，代理程式即停止檔案傳送。

  >0
  代理程式會繼續嘗試在指定的時間量 (以秒為單位) 內回復停滯的傳送。

  fteCreateTransfer 指令的範例

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt -1 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt 0 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt 21600 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

  fteCreateTemplate 指令的範例

```
fteCreateTemplate -tn "payroll accounts monthly report template" -rt -1 -sa PAYROLL -sm
QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

```
fteCreateTemplate -tn "payroll accounts monthly report template" -rt 0 -sa PAYROLL -sm
QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

```
fteCreateTemplate -tn "payroll accounts monthly report template" -rt 21600 -sa PAYROLL -sm
QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

  fteCreateMonitor 指令沒有 **-rt** 參數。 如果您使用 **fteCreateTransfer** 指令設定 **-rt** 參數，並同時設定 **-gt** 參數，則回復逾時參數會包含在 XML 文件中，並具有您執行 **fteCreateTransfer** 指令

時所產生的傳送定義。 然後，當您執行 **fteCreateMonitor** 指令時，資源監視器會使用此 XML 文件。 在下列範例中，傳送回復逾時詳細資料將包含在 task.xml 檔案中:

```
fteCreateMonitor -ma AgentName -md C:\mqmft\monitors -mn Monitor_Name -mt task.xml -tr
"fileSize>=5MB,*.zip"
```

- 若要使用 IBM MQ Explorer 「新建傳送」、「新建監視器」或「新建範本」精靈頁面來設定傳送回復逾時，請在 **傳送回復逾時** (秒) 欄位中選取必要選項:

  為來源代理程式
  如果您選取 **作為來源代理程式**，則會使用 agent.properties 檔案中的 **transferRecoveryTimeout** 參數值 (如果已設定的話)，否則會套用傳送回復逾時的預設行為。

  數值清單框
  如果您在數值清單框中輸入以秒為單位的時間，代理程式會繼續嘗試在指定的時間量內回復停滯的傳送。

  無
  如果您選取 **無**，則不會設定任何傳送回復逾時，代理程式會繼續嘗試回復已停滯的傳送，直到傳送成功為止。

- 使用 Ant 作業來設定回復逾時。 包括 **transferRecoveryTimeout** 選項和值，以及用於移動或複製檔案的 **fte:filecopy** 或 **fte:filemove** 元素，例如:

  fte:filecopy 的範例

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="copy.result" transferRecoveryTimeout="0">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filecopy>
```

  fte:filemove 的範例

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src=agent1@qm1 dst="agent2@qm2"
              rcproperty="move.result" transferRecoveryTimeout=”21600”>

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filemove>
```

Windows Linux AIX

管理 MQ Telemetry

使用 IBM MQ Explorer 或在指令行管理 MQ Telemetry。 使用瀏覽器來配置遙測通道、控制遙測服務，以及監視連接至 IBM MQ 的 MQTT 用戶端。 使用 JAAS、TLS 和 IBM MQ 物件權限管理程式來配置 MQ Telemetry 的安全。

使用 IBM MQ Explorer 管理

使用瀏覽器來配置遙測通道、控制遙測服務，以及監視連接至 IBM MQ 的 MQTT 用戶端。 使用 JAAS、TLS 和 IBM MQ 物件權限管理程式來配置 MQ Telemetry 的安全。

使用指令行管理

MQ Telemetry 可以在指令行 使用 MQSC 指令完全管理。

MQ Telemetry 文件也有範例 Script，示範 IBM MQ Telemetry Transport v3 用戶端應用程式的基本用法。

在使用之前，請先閱讀並瞭解 IBM MQ Telemetry Transport 範例程式 中的範例。

相關概念

MQ Telemetry

相關參考

MQXR 內容

Linux AIX 在 Linux 及 AIX 上配置遙測的佇列管理程式

請遵循下列步驟來手動配置 MQ Telemetry 。 如果您只需要使用訪客使用者 ID 的簡式配置，您可以改為在 IBM MQ Explorer 中執行 MQ Telemetry 支援精靈。

開始之前

如果您只需要簡式配置，請考量使用 IBM MQ Explorer 中的 MQ Telemetry 支援。 此支援包括精靈及範例指令程序 `sampleMQM`。 這些資源會使用訪客使用者 ID 來設定起始配置。 請參閱 使用 IBM MQ Explorer 驗證 MQ Telemetry 的安裝 和 IBM MQ Telemetry Transport 範例程式。

如果您需要使用不同鑑別方法的更複雜配置，請使用此作業中的步驟。 請從下列起始步驟開始:

1. 如需如何安裝 IBM MQ 及 MQ Telemetry 特性的相關資訊，請參閱 MQ Telemetry 的安裝考量。
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 IBM MQ Explorer 來配置。 如果您決定直接編輯檔案，請先停止佇列管理程式，再進行變更。 請參閱 MQXR 內容。

關於這項作業

請遵循此作業中的步驟，使用不同的授權架構來手動配置 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。

   如果 `SYSTEM.MQTT.TRANSMIT.QUEUE` 不存在，則會在第一次啟動遙測 (MQXR) 服務時自動建立它，並設為使用訪客使用者 ID。 不過，這項作業會將 MQ Telemetry 配置成使用不同的授權架構。 針對此作業，在啟動遙測 (MQXR) 服務之前，請先建立 `SYSTEM.MQTT.TRANSMIT.QUEUE` 並配置其存取權。

   請執行下列指令:

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc qMgr
   ```
3. 設定預設傳輸佇列。

   如果 `SYSTEM.MQTT.TRANSMIT.QUEUE` 是預設傳輸佇列，則更容易將訊息直接傳送至 MQTT 用戶端。 否則，您必須為接收 IBM MQ 訊息的每個用戶端新增遠端佇列定義; 請參閱 第 267 頁的『將訊息直接傳送至用戶端』。 請注意，變更預設傳輸佇列可能會干擾您現有的配置。

   第一次啟動遙測 (MQXR) 服務時，它不會將 `SYSTEM.MQTT.TRANSMIT.QUEUE` 設為佇列管理程式的預設傳輸佇列。 若要配置此設定，您可以變更預設傳輸佇列內容。 您可以使用 IBM MQ Explorer 或執行下列指令來執行此動作:

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```
4. 遵循 第 269 頁的『授權 MQTT 用戶端存取 IBM MQ 物件』 中的程序來建立一或多個使用者 ID。 使用者 ID 具有發佈、訂閱及傳送發佈至 MQTT 用戶端的權限。
5. 安裝遙測 (MQXR) 服務。

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

   另請參閱 第 264 頁的『建立 SYSTEM.MQXR.SERVICE』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 MQ Explorer，配置遙測通道以接受來自 MQTT 用戶端的連線。

   遙測通道必須配置成其身分是步驟 第 263 頁的『4』中所定義的其中一個使用者 ID。

   另請參閱 DEFINE CHANNEL (MQTT)。
8. 執行範例用戶端來驗證配置。

   若要讓範例用戶端使用遙測通道，該通道必須授權用戶端發佈、訂閱及接收發佈。 依預設，範例用戶端會連接至埠 1883 上的遙測通道。 另請參閱 IBM MQ Telemetry Transport 程式範例。

建立 SYSTEM.MQXR.SERVICE

使用 **runMQXRService** 指令來建立 SYSTEM.MQXR.SERVICE。

LTS
```
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')
```

V 9.2.4
```
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+" -sf "[DEFAULT]"') +
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')
```

註: V 9.2.4 從 IBM MQ 9.2.4 開始， **-sf** 旗標會提供認證金鑰檔名稱，其中包含要用來加密 TLS 通道通行詞組的金鑰。 如需相關資訊，請參閱 MQTT TLS 通道的通行詞組加密。

Windows 在 Windows 上配置遙測的佇列管理程式

請遵循下列步驟來手動配置 MQ Telemetry 。 如果您只需要使用訪客使用者 ID 的簡式配置，您可以改為在 IBM MQ Explorer 中執行 MQ Telemetry 支援精靈。

開始之前

如果您只需要簡式配置，請考量使用 IBM MQ Explorer 中的 MQ Telemetry 支援。 此支援包括精靈及範例指令程序 sampleMQM。 這些資源會使用訪客使用者 ID 來設定起始配置。 請參閱 使用 IBM MQ Explorer 驗證 MQ Telemetry 的安裝 和 IBM MQ Telemetry Transport 範例程式。

如果您需要使用不同鑑別方法的更複雜配置，請使用此作業中的步驟。 請從下列起始步驟開始:

1. 如需如何安裝 IBM MQ 及 MQ Telemetry 特性的相關資訊，請參閱 MQ Telemetry 的安裝考量。
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 IBM MQ Explorer 來配置。 如果您決定直接編輯檔案，請先停止佇列管理程式，再進行變更。 請參閱 MQXR 內容。

關於這項作業

請遵循此作業中的步驟，使用不同的授權架構來手動配置 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。

   如果 SYSTEM.MQTT.TRANSMIT.QUEUE 不存在，則會在第一次啟動遙測 (MQXR) 服務時自動建立它，並設為使用訪客使用者 ID。 不過，這項作業會將 MQ Telemetry 配置成使用不同的授權架構。 針對此作業，在啟動遙測 (MQXR) 服務之前，請先建立 SYSTEM.MQTT.TRANSMIT.QUEUE 並配置其存取權。

   請執行下列指令:

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```
3. 設定預設傳輸佇列。

   如果 SYSTEM.MQTT.TRANSMIT.QUEUE 是預設傳輸佇列，則更容易將訊息直接傳送至 MQTT 用戶端。 否則，您必須為接收 IBM MQ 訊息的每個用戶端新增遠端佇列定義; 請參閱 第 267 頁的『將訊息直接傳送至用戶端』。 請注意，變更預設傳輸佇列可能會干擾您現有的配置。

   第一次啟動遙測 (MQXR) 服務時，它不會將 SYSTEM.MQTT.TRANSMIT.QUEUE 設為佇列管理程式的預設傳輸佇列。 若要配置此設定，您可以變更預設傳輸佇列內容。 您可以使用 IBM MQ Explorer 或執行下列指令來執行此動作:

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```
4. 遵循 第 269 頁的『授權 MQTT 用戶端存取 IBM MQ 物件』 中的程序來建立一或多個使用者 ID。 使用者 ID 具有發佈、訂閱及傳送發佈至 MQTT 用戶端的權限。
5. 安裝遙測 (MQXR) 服務。

   ```
   type installMQXRService_win.mqsc | runmqsc qMgr
   ```

   另請參閱 第 266 頁的『建立中 SYSTEM.MQXR.SERVICE』 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 MQ Explorer，配置遙測通道以接受來自 MQTT 用戶端的連線。

   遙測通道必須配置成其身分是步驟 第 265 頁的『4』 中所定義的其中一個使用者 ID。

   另請參閱 DEFINE CHANNEL (MQTT)。
8. 執行範例用戶端來驗證配置。

   若要讓範例用戶端使用遙測通道，該通道必須授權用戶端發佈、訂閱及接收發佈。 依預設，範例用戶端會連接至埠 1883 上的遙測通道。 另請參閱 IBM MQ Telemetry Transport 程式範例。

建立中 SYSTEM.MQXR.SERVICE

使用 **runMQXRService** 指令來建立 SYSTEM.MQXR.SERVICE。

V 9.2.4

```
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+\." -sf "[DEFAULT]"') +
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')
```

註: V 9.2.4 從 IBM MQ 9.2.4 開始， **-sf** 旗標會提供認證金鑰檔名稱，其中包含要用來加密 TLS 通道通行詞組的金鑰。 如需相關資訊，請參閱 MQTT TLS 通道的通行詞組加密。

Windows Linux AIX 配置分散式佇列以將訊息傳送至 MQTT 用戶端

IBM MQ 應用程式可以透過發佈至用戶端所建立的訂閱，或直接傳送訊息，來傳送 MQTT v3 用戶端訊息。 無論使用哪種方法，都會將訊息放置在 SYSTEM.MQTT.TRANSMIT.QUEUE 上，並由遙測 (MQXR) 服務傳送至用戶端。 有數種方法可以在 SYSTEM.MQTT.TRANSMIT.QUEUE 上放置訊息。

發佈訊息以回應 MQTT 用戶端訂閱

遙測 (MQXR) 服務會代表 MQTT 用戶端建立訂閱。 用戶端是任何符合用戶端所傳送訂閱之發佈的目的地。 遙測服務會將相符發佈轉遞回用戶端。

MQTT 用戶端以佇列管理程式形式連接至 IBM MQ ，並將其佇列管理程式名稱設為 ClientIdentifier。 要傳送至用戶端的發佈目的地是傳輸佇列 SYSTEM.MQTT.TRANSMIT.QUEUE。 遙測服務會使用目標佇列管理程式名稱作為特定用戶端的金鑰，將 SYSTEM.MQTT.TRANSMIT.QUEUE 上的訊息轉遞至 MQTT 用戶端。

遙測 (MQXR) 服務會使用 ClientIdentifier 作為佇列管理程式名稱來開啟傳輸佇列。 遙測 (MQXR) 服務會將佇列的物件控點傳遞至 MQSUB 呼叫，以轉遞符合用戶端訂閱的發佈。 在物件名稱解析中，ClientIdentifier 會建立為遠端佇列管理程式名稱，且傳輸佇列必須解析為 SYSTEM.MQTT.TRANSMIT.QUEUE。 使用標準 IBM MQ 物件名稱解析， *ClientIdentifier* 會如下解析; 請參閱 第 267 頁的表 16。

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 *16: MQTT* 佇列管理程式別名的名稱解析					
	輸入		**輸出**		
ClientIdentifier	**佇列管理程式名稱**	**佇列名稱**	**佇列管理程式名稱**	**佇列名稱**	**傳輸佇列**
不符合任何項目	*ClientIdentifier*	未定義	*ClientIdentifier*	未定義	預設傳輸佇列。 SYSTEM.MQTT.TRANSMIT.QUEUE
符合名為 *ClientIdentifier* 的佇列管理程式別名	*ClientIdentifier*	未定義	*ClientIdentifier*	未定義	SYSTEM.MQTT.TRANSMIT.QUEUE

如需名稱解析的進一步相關資訊，請參閱 名稱解析。

任何 IBM MQ 程式都可以發佈至相同的主題。 發佈會傳送至其訂閱者，包括訂閱主題的 MQTT v3 用戶端。

如果在叢集中建立具有屬性 CLUSTER(*clusterName*)的管理主題，則叢集中的任何應用程式都可以發佈至用戶端; 例如:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

圖 *16:* 在 *Windows* 上定義叢集主題

註: 請勿提供叢集屬性給 SYSTEM.MQTT.TRANSMIT.QUEUE 。

MQTT 用戶端訂閱者和發佈者可以連接至不同的佇列管理程式。 訂閱者和發佈者可以是相同叢集的一部分，也可以透過發佈/訂閱階層來連接。 發佈會使用 IBM MQ 從發佈者遞送至訂閱者。

將訊息直接傳送至用戶端

除了用戶端建立訂閱並接收符合訂閱主題的發佈之外，另一種方法是將訊息直接傳送至 MQTT v3 用戶端。 MQTT V3 用戶端應用程式無法直接傳送訊息，但其他應用程式 (例如 IBM MQ 應用程式) 可以。

IBM MQ 應用程式必須知道 MQTT v3 用戶端的 ClientIdentifier 。 因為 MQTT v3 用戶端沒有佇列，所以目標佇列名稱會傳遞至 MQTT v3 應用程式用戶端 messageArrived 方法作為主題名稱。 例如，在 MQI 程式中，建立一個物件描述符，其中客戶端為 ObjectQmgrName：

```
MQOD.ObjectQmgrName = ClientIdentifier ;
MQOD.ObjectName = name ;
```

圖 *17:* 將訊息傳送至 *MQTT v3* 用戶端目的地的 *MQI* 物件描述子

如果使用 JMS 撰寫應用程式，請建立點對點目的地; 例如:

```
javax.jms.Destination jmsDestination =
(javax.jms.Destination)jmsFactory.createQueue
("queue://ClientIdentifier/name");
```

圖 *18: JMS* 目的地，用於將訊息傳送至 *MQTT v3* 用戶端

若要將自發訊息傳送至 MQTT 用戶端，請使用遠端佇列定義。 遠端佇列管理程式名稱必須解析為用戶端的 ClientIdentifier。 傳輸佇列必須解析為 SYSTEM.MQTT.TRANSMIT.QUEUE ; 請參閱 第 268 頁的表 17。 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 17: *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 無狀態及有狀態階段作業。 當用戶端重新連接至階段作業時，會將它轉遞至用戶端。

如果您傳送非持續訊息，它會以 **最多一次** 服務品質 QoS=0 傳送至用戶端。 如果您直接將持續訊息傳送至用戶端，依預設會使用 **正好一次** 服務品質 QoS=2 來傳送該持續訊息。 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。

Windows Linux AIX MQTT 用戶端識別、授權及鑑別

遙測 (MQXR) 服務會使用 MQTT 通道，代表 MQTT 用戶端發佈或訂閱 IBM MQ 主題。 IBM MQ 管理者會配置用於 IBM MQ 授權的 MQTT 通道身分。 管理者可以定義通道的一般身分，或者使用已連接至通道的用戶端的 Username 或 ClientIdentifier。

遙測 (MQXR) 服務可以使用用戶端提供的 Username 或者使用用戶端憑證來鑑別用戶端。 則使用用戶端提供的密碼來鑑別 Username。

彙總： 用戶端識別是用戶端身分的選項。 根據環境定義，用戶端由 ClientIdentifier、Username、管理者建立的一般用戶端身分或用戶端憑證識別。 用於確實性檢查的用戶端 ID 不一定是用於授權的 ID。

MQTT 用戶端程式會設定使用 MQTT 通道傳送至伺服器的 **使用者名稱** 及 **密碼**。 他們也可以設定加密及鑑別連線所需的 TLS 內容。 管理者會決定是否鑑別 MQTT 通道，以及如何鑑別通道。

若要授權 MQTT 用戶端存取 IBM MQ 物件，請授權用戶端的 ClientIdentifier 或 Username，或授權一般用戶端身分。 若要允許用戶端連接至 IBM MQ，請鑑別 Username 或使用用戶端憑證。 配置 JAAS 以鑑別 Username，並配置 TLS 以鑑別用戶端憑證。

如果您在用戶端設定 **密碼**，請使用 VPN 加密連線，或將 MQTT 通道配置為使用 TLS，以保持密碼專用。

管理用戶端憑證非常困難。 因此，如果與密碼鑑別相關聯的風險可以接受，則一般會使用密碼鑑別來鑑別用戶端。

如果可以安全地管理和儲存用戶端憑證，則可以依賴於憑證鑑別。 不過，在使用遙測的環境類型中，極少能夠安全地管理憑證。 而是使用用戶端憑證鑑別裝置，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。

Windows Linux AIX MQTT 用戶端身分及授權

使用用戶端 ID、 Username 或一般用戶端身分來授權存取 IBM MQ 物件。

IBM MQ 管理者有三個選項，可用來選取 MQTT 通道的身分。 管理者在定義或修改用戶端所使用的 MQTT 通道時進行選擇。 身分用於授與 IBM MQ 主題的存取權。 選擇順序如下:

1. 用戶端 ID (請參閱 USECLNTID)。
2. 管理者為通道提供的身分 (通道的 MCAUSER)。 請參閱 MCAUSER)。
3. 如果上述任一選項都不適用，則從 MQTT 用戶端傳遞的 **使用者名稱** (Username 是 MqttConnectOptions 類別的屬性。 必須在用戶端連接至服務之前將其設定。 其預設值為空值)。

避免麻煩: 之後會參照此程序所選擇的身分，例如 DISPLAY CHSTATUS (MQTT) 指令作為用戶端的 MCAUSER。 請注意，這不一定與選項 (2) 中所參照通道的 MCAUSER 相同。

使用 IBM MQ **setmqaut** 指令來選取與 MQTT 通道相關聯的身分已授權使用哪些物件及哪些動作。 例如，下列程式碼授權通道身分 MQTTClient，由佇列管理程式 QM1 的管理者提供:

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

Windows Linux AIX 授權 *MQTT* 用戶端存取 *IBM MQ* 物件

遵循下列步驟，以授權 MQTT 用戶端發佈及訂閱 IBM MQ 物件。 下列步驟遵循四個替代存取控制型樣。

開始之前

MQTT 用戶端在連接至遙測通道時，會獲指派身分，以存取 IBM MQ 中的物件。 「IBM MQ 管理者」會使用 IBM MQ Explorer 來配置遙測通道，以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。

不論使用哪一種類型，已安裝的授權服務必須將身分定義給 IBM MQ 作為主體。 Windows 或 Linux 上的預設授權服務稱為「物件權限管理程式 (OAM)」。 如果您是使用 OAM，則身分必須定義為使用者 ID。

使用身分為用戶端或用戶端集合提供發佈或訂閱 IBM MQ 中所定義主題的許可權。 如果 MQTT 用戶端已訂閱主題，請使用身分來授與它接收產生的發佈資訊的許可權。

很難管理具有數萬個 MQTT 用戶端的系統，每個用戶端都需要個別存取權。 其中一個解決方案是定義一般身分，並將個別 MQTT 用戶端與其中一個一般身分相關聯。 定義所需數量的一般身分，以定義不同的許可權組合。 另一個解決方案是撰寫您自己的授權服務，它比作業系統更容易與數以千計的使用者打交道。

您可以使用 OAM，以兩種方式將 MQTT 用戶端結合成一般身分:

1. 定義多個遙測通道，每一個遙測通道都具有管理者使用「IBM MQ 探險家」配置的不同使用者 ID。 使用不同 TCP/IP 埠號連接的用戶端會與不同的遙測通道相關聯，並獲指派不同的身分。
2. 定義單一遙測通道，但讓每一個用戶端從一小組使用者 ID 中選取 **使用者名稱** 。 管理者會配置遙測通道，以選取用戶端 **使用者名稱** 作為其身分。

在此作業中，遙測通道的身分稱為 *mqttUser*，不論其設定方式為何。 如果用戶端集合使用不同的身分，請使用多個 *mqttUsers*，每一個用戶端集合一個。 由於作業使用 OAM，因此每一個 *mqttUser* 都必須是使用者 ID。

關於這項作業

在這項作業中，您可以選擇四個存取控制型樣，以符合特定需求。 型樣在存取控制的精度方面有所不同。

- 第 269 頁的『無存取控制』
- 第 270 頁的『粗略存取控制』
- 第 270 頁的『中階存取控制』
- 第 270 頁的『精細存取控制』

模型的結果是將 *mqttUsers* 許可權集指派給發佈和訂閱 IBM MQ，以及從 IBM MQ 接收發佈。

無存取控制
MQTT 用戶端會獲授與 IBM MQ 管理權限，且可以對任何物件執行任何動作。

程序

1. 建立使用者 ID *mqttUser*，以作為所有 MQTT 用戶端的身分。
2. 將 *mqttUser* 新增至 mqm 群組; 請參閱 在 Windows 上新增使用者至群組，或 在 Linux 上建立及管理群組

粗略存取控制
MQTT 用戶端有權發佈及訂閱，以及將訊息傳送至 MQTT 用戶端。 他們沒有執行其他動作或存取其他物件的權限。

程序

1. 建立使用者 ID *mqttUser*，以作為所有 MQTT 用戶端的身分。
2. 授權 *mqttUser* 發佈及訂閱所有主題，以及將發佈傳送至 MQTT 用戶端。

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

中階存取控制
MQTT 用戶端分成不同的群組，以發佈及訂閱不同的主題集，以及將訊息傳送至 MQTT 用戶端。

程序

1. 在發佈/訂閱主題樹狀結構中建立多個使用者 ID、 *mqttUsers* 及多個管理主題。
2. 將不同的 *mqttUsers* 授權給不同的主題。

```
setmqaut -m qMgr -t topic -n topic1 -p mqttUserA -all +pub +sub
setmqaut -m qMgr -t topic -n topic2 -p mqttUserB -all +pub +sub
```

3. 建立群組 *mqtt*，並將所有 *mqttUsers* 新增至群組。
4. 授權 *mqtt* 將主題傳送至 MQTT 用戶端。

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

精細存取控制
MQTT 用戶端會納入現有的存取控制系統中，授權群組對物件執行動作。

關於這項作業

使用者 ID 會指派給一或多個作業系統群組，視它需要的授權而定。 如果 IBM MQ 應用程式發佈及訂閱與 MQTT 用戶端相同的主題空間，請使用此模型。 群組稱為 Publish *X*、 Subscribe *Y* 及 mqtt

Publish *X*
Publish *X* 群組的成員可以發佈至 *topicX*。

Subscribe *Y*
Subscribe *Y* 群組的成員可以訂閱 *topicY*。

mqtt
mqtt 群組的成員可以將發佈傳送至 MQTT 用戶端。

程序

1. 在發佈/訂閱主題樹狀結構中，建立多個配置給多個管理主題的群組 Publish *X* 和 Subscribe *Y*。
2. 建立群組 mqtt。
3. 建立多個使用者 ID *mqttUsers*，並視他們獲授權執行的動作而定，將使用者新增至任何群組。
4. 授權不同的 Publish *X* 及 Subscribe *X* 群組給不同的主題，並授權 *mqtt* 群組將訊息傳送至 MQTT 用戶端。

```
setmqaut -m qMgr -t topic -n topic1 -p Publish X -all +pub
setmqaut -m qMgr -t topic -n topic1 -p Subscribe X -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Windows Linux AIX MQTT 使用密碼進行用戶端鑑別

使用用戶端密碼鑑別 Username。 用於鑑別用戶端的身分，可以不同於用於授權用戶端發佈至和訂閱主題的身分。

遙測 (MQXR) 服務會使用 JAAS 來鑑別用戶端 Username。 JAAS 使用 MQTT 用戶端提供的 **密碼** 。

IBM MQ 管理者透過配置用戶端所連接的 MQTT 通道，來決定是否鑑別 **使用者名稱**，或完全不鑑別。 可以將用戶端指派給不同通道，而且可以配置每個通道以使用不同的方法鑑別其用戶端。 如果使用 JAAS，則您可以配置哪些方法必須鑑別用戶端，哪些方法可以選擇性地鑑別用戶端。

選擇用於鑑別的身分不會影響選擇用於授權的身分。 為了方便管理，您可能會設定用於授權的一般身分，但是鑑別要使用該身分的每個使用者。 下列程序概述的步驟，用於鑑別要使用一般身分的個別使用者：

1. IBM MQ 管理者使用 IBM MQ Explorer 將 MQTT 通道身分設為任何名稱，例如 MQTTClientUser。
2. IBM MQ 管理者授權 MQTTClient 發佈和訂閱任何主題：

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

3. 在連接至伺服器之前， MQTT 用戶端應用程式開發者會建立 MqttConnectOptions 物件，並設定 Username 和 Password 。
4. 安全開發者建立 JAAS LoginModule，以利用 Password 鑑別 Username，並將它併入 JAAS 配置檔。
5. IBM MQ 管理者會配置 MQTT 通道，以使用 JAAS 來鑑別用戶端的 UserName 。

Windows Linux AIX 使用 TLS 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。

透過向用戶端提供私有簽署的數位證書，您可以向 WebSphere MQ 驗證 MQTT 用戶端的身份。 WebSphere MQ 管理者可以強制 MQTT 用戶端使用 TLS 向佇列管理程式鑑別其自己。 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 TLS 的用戶端鑑別依賴於具有密碼的用戶端。 對於自簽憑證，密碼是用戶端的私密金鑰，否則是憑證管理中心提供的金鑰。 金鑰用於簽章用戶端的數位憑證。 擁有對應公開金鑰的任何人都可以驗證該數位憑證。 可以信任憑證，如果這些憑證已鏈結，則可以透過憑證鏈回溯追蹤至授信主要憑證。 用戶端驗證會將用戶端所提供憑證鏈中的所有憑證傳送至伺服器。 伺服器會檢查憑證鏈，直到它找到信任的憑證為止。 授信憑證是從自簽憑證產生的公用憑證，或者是一般由憑證管理中心發出的主要憑證。 在最後一個選用步驟中，可以將信任的憑證與「現用」的憑證撤銷清冊相互比較。

授信憑證可能由憑證管理中心發出，並且已經包含在 JRE 憑證儲存庫中。 它可以是自簽憑證，也可以是已作為授信憑證新增至遙測通道金鑰儲存庫的任何憑證。

註: 遙測通道具有結合的金鑰儲存庫/信任儲存庫，其中保留一個以上遙測通道的私密金鑰，以及鑑別用戶端所需的任何公用憑證。 因為 SSL 通道必須具有金鑰儲存庫，所以它與通道信任儲存庫是同一個檔案，永不參照 JRE 憑證儲存庫。 言下之意，如果用戶端鑑別需要 CA 主要憑證，您必須將主要憑證放置於通道的金鑰儲存庫中，即使 JRE 憑證儲存庫中已經存在 CA 主要憑證也是如此。 永不參照 JRE 憑證儲存庫。

考慮用戶端鑑別打算應對的威脅，以及用戶端和伺服器在應對威脅時所扮演的角色。 單獨鑑別用戶端憑證不足以防止未獲授權存取系統。 如果其他使用者已在使用該用戶端裝置，則該用戶端裝置不需使用憑證持有者的權限就可以運作。 切勿依賴單一防禦措施來防範意外攻擊。 至少使用雙重因數的鑑別方法，以及補充關於擁有憑證的私密資訊知識。 例如，使用 JAAS，並且使用伺服器發出的密碼鑑別用戶端。

用戶端憑證的主要威脅是落入不適合的人手中。 憑證保存在用戶端上受密碼保護的金鑰儲存庫中。 系統是如何將憑證放入金鑰儲存庫中？ MQTT 用戶端如何取得金鑰儲存庫的密碼？ 密碼保護的安全程度如何？ 遙測裝置通常易於移除，然後可被私下入侵。 裝置硬體是否必須具有防竄改功能？ 配送和保護用戶端憑證非常困難，這稱為金鑰管理問題。

次要威脅是無意中誤用裝置來存取伺服器。 比方說，如果 MQTT 應用程式被竄改，則它可能會使用已鑑別的用戶端身分，利用伺服器配置中的缺點。

若要使用 SSL 來鑑別 MQTT 用戶端，請配置遙測通道及用戶端。

相關概念

第 272 頁的『使用 TLS 進行 MQTT 用戶端鑑別的遙測通道配置』
IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 TLS 通道已配置成使用通行詞組來保護對金鑰檔的存取權。 如果 TLS 通道未定義通行詞組或金鑰檔，則通道不會接受 TLS 連線。

相關工作

使用 TLS 進行用戶端鑑別的 MQTT 用戶端配置

Windows Linux AIX 使用 *TLS* 進行 *MQTT* 用戶端鑑別的遙測通道配置

IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 TLS 通道已配置成使用通行詞組來保護對金鑰檔的存取權。 如果 TLS 通道未定義通行詞組或金鑰檔，則通道不會接受 TLS 連線。

將 TLS 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它們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，公開簽署的憑證會儲存在與私密簽署憑證相同的金鑰檔中，而不是儲存在個別信任儲存庫中。

伺服器會使用它所擁有的所有公用憑證及密碼組合，來驗證它所傳送之任何用戶端憑證的簽章。 伺服器會驗證金鑰鏈。 佇列管理程式可以配置成根據憑證撤銷清冊來測試憑證。 佇列管理程式撤銷名稱清單內容是 `SSLCRLNL`。

如果伺服器金鑰儲存庫中的憑證已驗證用戶端傳送的任何憑證，則會鑑別用戶端。

IBM MQ 管理者可以配置相同的遙測通道，以使用 JAAS 來檢查用戶端的 `UserName` 或 `ClientIdentifier` (具有用戶端 **密碼**)。

您可以對多個遙測通道使用相同的金鑰儲存庫。

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的驗證會向伺服器鑑別用戶端。 數位憑證僅用於由 IBM MQ 進行鑑別。 它不是用來驗證用戶端的 TCP/IP 位址，或設定授權或帳戶的用戶端身分。 伺服器採用的用戶端身分是用戶端的 `Username` 或 `ClientIdentifier`，或 IBM MQ 管理者所建立的身分。

您也可以使用 TLS 密碼組合進行用戶端鑑別。 如果您計劃使用 SHA-2 密碼組合，請參閱第 273 頁的『搭配使用 SHA-2 密碼組合與 MQTT 通道的系統需求』。

相關概念

第 273 頁的『使用 TLS 進行通道鑑別的遙測通道配置』
IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 TLS 通道已配置成使用通行詞組來保護對金鑰檔的存取權。 如果 TLS 通道未定義通行詞組或金鑰檔，則通道不會接受 TLS 連線。

相關參考

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)
CipherSpecs 和 CipherSuites

Windows Linux AIX 使用 TLS 進行遙測通道鑑別

MQTT 用戶端與佇列管理程式之間的連線一律由 MQTT 用戶端起始。 MQTT 用戶端一律是 SSL 用戶端。 伺服器的用戶端鑑別和 MQTT 用戶端的伺服器鑑別皆為選用項目。

除非將用戶端配置為使用支援匿名連線的 CipherSpec，否則它一律會嘗試鑑別伺服器。 如果鑑別失敗，則不會建立連線。

作為使用 SSL 的替代方案，一些類型的「虛擬私密網路 (VPN)」（例如 IPsec），會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 MQTT 用戶端連接至遙測通道。

使用 SSL 的伺服器鑑別會鑑別作為要傳送機密資訊目標的伺服器。 用戶端會針對放置在其信任儲存庫或 JRE `cacerts` 儲存庫中的憑證，執行符合從伺服器傳送之憑證的檢查。

JRE 憑證儲存庫是 JKS 檔案 `cacerts`。 它位於 `JRE InstallPath\lib\security\`。 安裝後，它的預設密碼為 `changeit`。 您可以將信任的憑證儲存在 JRE 憑證儲存庫或用戶端信任儲存庫中。 不能同時使用這兩個儲存庫。 如果您想要將用戶端信任的公用憑證與其他 Java 應用程式使用的憑證分開，請使用用戶端信任儲存庫。 如果您想要對用戶端上執行的所有 Java 應用程式使用一般憑證儲存庫，請使用 JRE 憑證儲存庫。 如果決定使用 JRE 憑證儲存庫，請檢閱它所包含的憑證，以確保您信任這些憑證。

透過提供不同的信任提供者，您可以修改 JSSE 配置。 您可以自訂信任提供者，以對憑證執行不同的檢查。 在部分已使用 MQTT 用戶端的 OGSi 環境中，環境會提供不同的信任提供者。

若要使用 TLS 來鑑別遙測通道，請配置伺服器和用戶端。

Windows Linux AIX 使用 TLS 進行通道鑑別的遙測通道配置

IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 TLS 通道已配置成使用通行詞組來保護對金鑰檔的存取權。 如果 TLS 通道未定義通行詞組或金鑰檔，則通道不會接受 TLS 連線。

將伺服器的數位憑證 (使用其私密金鑰簽署) 儲存在遙測通道將在伺服器上使用的金鑰儲存庫中。 如果您要將金鑰鏈傳輸至用戶端，請將任何憑證儲存在金鑰儲存庫的金鑰鏈中。 使用 IBM MQ 瀏覽器來配置遙測通道，以使用 TLS。 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 TLS 遙測通道埠號預設為 8883。

您也可以使用 TLS 密碼組合進行通道鑑別。 如果您計劃使用 SHA-2 密碼組合，請參閱第 273 頁的『搭配使用 SHA-2 密碼組合與 MQTT 通道的系統需求』。

相關概念

第 272 頁的『使用 TLS 進行 MQTT 用戶端鑑別的遙測通道配置』
IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 TLS 通道已配置成使用通行詞組來保護對金鑰檔的存取權。 如果 TLS 通道未定義通行詞組或金鑰檔，則通道不會接受 TLS 連線。

相關參考

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)
CipherSpecs 和 CipherSuites

Windows Linux AIX 搭配使用 SHA-2 密碼組合與 MQTT 通道的系統需求

如果您使用支援 SHA-2 密碼組合的 Java 版本，您可以使用這些組合來保護 MQTT (遙測) 通道及用戶端應用程式的安全。

對於包括遙測 (MQXR) 服務的 IBM MQ 8.0 ， 最低 Java 版本為 Java 7 from IBM ， SR6。 依預設，從 IBM SR4 開始， Java 7 中支援 SHA-2 密碼組合。 因此，您可以搭配使用 SHA-2 密碼組合與遙測 (MQXR) 服務，以保護 MQTT (遙測) 通道的安全。

如果您使用不同的 JRE 來執行 MQTT 用戶端，則需要確定它也支援 SHA-2 密碼組合。

相關概念

遙測 (MQXR) 服務

第 273 頁的『使用 TLS 進行通道鑑別的遙測通道配置』

IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 TLS 通道已配置成使用通行詞組來保護對金鑰檔的存取權。 如果 TLS 通道未定義通行詞組或金鑰檔，則通道不會接受 TLS 連線。

相關參考

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

Windows Linux AIX 遙測通道上的發佈保密

使用 TLS 來加密透過連線的傳輸，可保護透過遙測通道以任一方向傳送之 MQTT 發佈資訊的隱私權。

連接至遙測通道的 MQTT 用戶端會使用 TLS，以使用對稱金鑰加密法來保護通道上所傳輸發佈的隱私權。 因為不會鑑別端點，所以您不能信任單獨使用的通道加密。 將保密安全與伺服器或交互鑑別結合使用。

作為使用 SSL 的替代方案，一些類型的「虛擬私密網路 (VPN)」（例如 IPsec），會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 MQTT 用戶端連接至遙測通道。

對於會加密通道和鑑別伺服器的一般配置，請參閱第 273 頁的『使用 TLS 進行遙測通道鑑別』。

在不鑑別伺服器的情況下加密 TLS 連線會對中間人攻擊公開連線。 雖然您交換的資訊可以防竊聽，但是您不知道與您交換資訊的對象是誰。 除非您控制網路，否則您就會面臨別人截取您的 IP 傳輸，以及假冒端點的問題。

您可以使用支援匿名 TLS 的 Diffie-Hellman 金鑰交換 CipherSpec 來建立已加密 TLS 連線，而無需鑑別伺服器。 在用戶端與伺服器之間共用且用來加密 TLS 傳輸的主要密碼，是在不交換私密簽署伺服器憑證的情況下建立的。

由於匿名連線不安全，大部分 TLS 實作不會預設為使用匿名 CipherSpecs。 如果遙測通道接受 TLS 連線的用戶端要求，則通道必須具有受通行詞組保護的金鑰儲存庫。 依預設，由於 TLS 實作不使用匿名 CipherSpecs，金鑰儲存庫必須包含用戶端可以鑑別的私密簽章憑證。

如果您使用匿名 CipherSpec，則伺服器金鑰儲存庫必須存在，但是它不必包含任何私密簽章的憑證。

建立加密連線的另一種方法，是利用您自己的實作，取代用戶端上的信任提供者。 您的信任提供者不會鑑別伺服器憑證，但是會加密連線。

小心: 將 TLS 與 MQTT 搭配使用時，您可以使用大型訊息，不過，這樣做可能會影響效能。 MQTT 已針對處理小型訊息 (大小通常介於 1KB 與 1MB 之間) 進行最佳化。

Windows Linux AIX MQTT Java 用戶端及遙測通道的 TLS 配置

配置 TLS 以鑑別遙測通道及 MQTT Java 用戶端，並加密它們之間的訊息傳送。 MQTT Java 用戶端使用 Java Secure Socket Extension (JSSE) 來使用 TLS 連接遙測通道。 作為使用 SSL 的替代方案，一些類型的「虛擬私密網路 (VPN)」（例如 IPsec），會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 MQTT 用戶端連接至遙測通道。

您可以配置 Java MQTT 用戶端與遙測通道之間的連線，以使用透過 TCP/IP 的 TLS 通訊協定。 安全取決於您配置 TLS 以使用 JSSE 的方式。 您可以配置三個不同的安全等級，從最安全的配置開始:

1. 只允許信任的 MQTT 用戶端連接。 僅將 MQTT 用戶端連接至授信遙測通道。 加密用戶端與佇列管理程式之間的訊息; 請參閱 第 271 頁的『使用 TLS 進行 MQTT 用戶端鑑別』
2. 僅將 MQTT 用戶端連接至授信遙測通道。 加密用戶端和佇列管理程式之間的訊息；請參閱第 273 頁的『使用 TLS 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 274 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 TLS 連線的配置方式。 JSSE 配置參數組織成三個集：

1. MQ Telemetry 通道
2. MQTT Java 用戶端
3. JRE

使用「IBM MQ 探險家」來配置遙測通道參數。 在 MqttConnectionOptions.SSLProperties 屬性中設定 MQTT Java 用戶端參數。 在用戶端和伺服器上，透過編輯 JRE 安全目錄中的檔案，修改 JRE 安全參數。

MQ Telemetry channel

使用 IBM MQ Explorer 來設定所有遙測通道 TLS 參數。

ChannelName

在所有通道上，ChannelName 都是必要的參數。

通道名稱識別與特定埠號相關聯的通道。 命名通道以協助您管理 MQTT 用戶端集。

PortNumber

在所有通道上，PortNumber 都是選用參數。 對於 TCP 通道，它預設為 1883 ，對於 TLS 通道，它預設為 8883 。

與此通道相關聯的 TCP/IP 埠號。 透過指定為通道定義的埠，將 MQTT 用戶端連接至通道。 如果通道具有 TLS 內容，則用戶端必須使用 TLS 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 TLS 頻道的必要參數。 對於 TCP 通道，必須省略。

KeyFileName 是包含您提供的數位憑證的 Java 金鑰庫的路徑。 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 TLS 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 是選用的 TLS 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，用戶端必須使用 TLS 連線到伺服器，並對伺服器進行身份驗證。 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 是選用性 TLS 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 TLS 連線的 CipherSpec 的名稱。

遙測服務及 MQTT 用戶端會從每一端啟用的所有 CipherSpecs 協議一般 CipherSpec 。 如果在連線的任一端或全部兩端指定特定 CipherSpec，則它必須與另一端的 CipherSpec 相符。

透過將其他提供者新增至 JSSE，安裝其他密碼。

Federal Information Processing Standards (FIPS)

FIPS 是一項選用設定。 依預設不會對其設定。

在佇列管理程式的「內容」畫面中，或者使用 **runmqsc**，可以設定 SSLFIPS。 SSLFIPS 指定是否僅使用通過 FIPS 認證的演算法。

名單

撤銷名單是一項選用設定。 依預設不會對其設定。

在佇列管理程式的「內容」畫面中，或者使用 **runmqsc**，可以設定 SSLCRLNL。 SSLCRLNL 指定用於提供憑證撤銷位置的鑑別資訊物件名單。

不會使用其他設定 TLS 內容的佇列管理程式參數。

MQTT Java 用戶端

在 MqttConnectionOptions .SSLProperties 中為 Java 客戶端設定 TLS 屬性；例如：

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

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

在 JRE 中配置 Java 安全的其他層面，這些層面會影響用戶端及伺服器上 TLS 的行為。 Windows 上的配置檔位於 *Java Installation Directory*\jre\lib\security 中。 如果您使用的是 IBM MQ 隨附的 JRE，則路徑如下表中所示：

表 *18: JRE TLS* 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
AIX 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 IBM MQ 指令 strmqkm 來開啟 cacerts 。 使用密碼 changeit，將 cacerts 作為 JKS 檔案開啟。 修改密碼以維護檔案的安全。

配置安全類別

使用 java.security 檔案以登錄其他安全提供者和其他預設安全內容。

權限

使用 java.policy 檔案來修改授與資源的權限。 javaws.policy 會授與 javaws.jar 的權限

加密強度

一些 JRE 提供強度減弱的加密。 如果您無法將金鑰匯入至金鑰儲存庫，則原因可能是加密的強度減弱。 請嘗試使用 **strmqikm** 指令來啟動 **ikeyman** ，或從 IBM 開發人員套件、安全資訊下載強大但適用範圍有限的檔案。

重要: 您所在的國家/地區可能會對加密軟體的進口、佔有、使用或轉口至其他國家/地區施加限制。 在下載或使用未限定政策檔案之前，您必須檢查您所在國家/地區的法律。 請檢查其進口、佔有、使用和轉口加密軟體的相關法規和政策，判斷是否允許該軟體。

修改信任提供者以允許用戶端連接至所有伺服器

此範例說明如何新增信任提供者，並從 MQTT 用戶端程式碼參照它。 範例不會執行用戶端或伺服器鑑別。 產生的 TLS 連線會加密而不進行鑑別。

第 278 頁的圖 19 中的程式碼 Snippet 會設定 MQTT 用戶端的 AcceptAllProviders 信任提供者和信任管理程式。

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

圖 *19: 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 *20: 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 *21: 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 *22: AcceptAllX509TrustManager.java*

Windows Linux AIX 遙測通道 JAAS 配置

配置 JAAS 以鑑別用戶端傳送的 Username。

IBM MQ 管理者會使用 JAAS 來配置哪些 MQTT 通道需要用戶端鑑別。 指定要執行 JAAS 鑑別的每個通道的 JAAS 配置名稱。 通道可以全部使用相同的 JAAS 配置，也可以使用不同的 JAAS 配置。 配置定義在 *WMQData directory*\qmgrs*qMgrName*\mqxr\jaas.config 中。

jaas.config 檔案依 JAAS 配置名稱組織。 在每個配置名稱下面，是「登入」配置清單；請參閱 第 279 頁的『範例 jaas.config 檔案』。

JAAS 提供四個標準「登入」模組。 標準 NT 和 UNIX 「登入」模組值有限。

JndiLoginModule
針對在 JNDI (Java 命名和目錄介面) 下配置的目錄服務進行鑑別。

Krb5LoginModule
使用 Kerberos 通訊協定進行鑑別。

NTLoginModule
使用現行使用者的 NT 安全資訊進行鑑別。

UnixLoginModule
使用現行使用者的 UNIX 安全資訊進行鑑別。

使用 NTLoginModule 或 UnixLoginModule 的問題是遙測 (MQXR) 服務以 mqm 身分執行，而不是 MQTT 通道的身分。 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。 MQ Telemetry 隨附範例 JAASLoginModule.java 。 它是 javax.security.auth.spi.LoginModule 介面的實作。 可以使用它來開發您自己的鑑別方法。

您提供的所有新 LoginModule 類別都必須在遙測 (MQXR) 服務的類別路徑上。 請勿將類別放在類別路徑中的 IBM MQ 目錄中。 建立您自己的目錄，並定義遙測 (MQXR) 服務的完整類別路徑。

透過在 service.env 檔案中設定類別路徑，可以擴增遙測 (MQXR) 服務使用的類別路徑。 CLASSPATH 必須大寫，並且類別路徑陳述式只能包含文字。 不能在 CLASSPATH 中使用變數；例如 CLASSPATH=%CLASSPATH% 就是不正確的。 遙測 (MQXR) 服務會設定其專屬類別路徑。 會將定義在 service.env 中的 CLASSPATH 新增至該類別路徑。

遙測 (MQXR) 服務提供兩個回呼，針對連接至 MQTT 通道的用戶端傳回 **使用者名稱** 及 **密碼** 。 **使用者名稱** 和 **密碼** 設定在 MqttConnectOptions 物件中。 請參閱第 279 頁的『範例 JAASLoginModule.Login() 方法』，以取得如何存取 Username 和 Password 的範例。

範例 jaas.config 檔案

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";
};
```

範例 JAASLoginModule.Login() 方法

JAAS 登入模組的範例，編碼為接收 MQTT 用戶端提供的 **使用者名稱** 和 **密碼** 。

```
public boolean login()
throws javax.security.auth.login.LoginException {
javax.security.auth.callback.Callback[] callbacks =
new javax.security.auth.callback.Callback[2];
callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
callbacks[1] = new javax.security.auth.callback.PasswordCallback(
"PasswordCallback", false);
try {
callbackHandler.handle(callbacks);
String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
.getName();
char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
.getPassword();
 // Accept everything.
if (true) {
loggedIn = true;
} else
```

```
throw new javax.security.auth.login.FailedLoginException("Login failed");

principal= new JAASPrincipal(username);

} catch (java.io.IOException exception) {
throw new javax.security.auth.login.LoginException(exception.toString());
} catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
throw new javax.security.auth.login.LoginException(exception.toString());
}

return loggedIn;
}
```

相關工作

解決問題：Telemetry 服務未呼叫 JAAS 登入模組

相關參考

AuthCallback MQXR 類別

管理 AMQP 用戶端

您可以使用 IBM MQ Explorer 或在指令行管理 AMQP 用戶端。 使用「瀏覽器」來配置通道，並監視連接至 IBM MQ 的 AMQP 用戶端。 使用 TLS 及 JAAS 來配置 AMQP 用戶端的安全。

開始之前

如需在平台上安裝 AMQP 的相關資訊，請參閱 選擇要安裝的項目。

使用 IBM MQ Explorer 管理

使用「探險家」來配置 AMQP 通道，並監視連接至 IBM MQ 的 AMQP 用戶端。 您可以使用 TLS 及 JAAS 來配置 AMQP 用戶端的安全。

使用指令行管理

您可以 使用 MQSC 指令在指令行管理 AMQP 用戶端。

檢視 AMQP 用戶端正在使用的 IBM MQ 物件

您可以檢視 AMQP 用戶端正在使用的不同 IBM MQ 資源，例如連線及訂閱。

連線

啟動 AMQP 服務時，會建立新的 Hconns 並連接至佇列管理程式。 當 AMQP 用戶端發佈訊息時，會使用此 Hconns 儲存區。 您可以使用 **DISPLAY CONN** 指令來檢視 Hconns。 例如：

```
DISPLAY CONN(*) TYPE(CONN) WHERE (APPLDESC LK 'IBM MQ Advanced Message Queuing Protocol*')
```

此指令也會顯示任何用戶端特定的 Hconns。 具有空白用戶端 ID 屬性的 Hconns 是儲存區中使用的 Hconns

當 AMQP 用戶端連接至 AMQP 通道時，新的 Hconn 會連接至佇列管理程式。 此 Hconn 用於針對 AMQP 用戶端所建立的訂閱非同步使用訊息。 您可以使用 **DISPLAY CONN** 指令來檢視特定 AMQP 用戶端所使用的 Hconn。 例如：

```
DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_abcd1234')
```

用戶端建立的訂閱

當 AMQP 用戶端訂閱主題時，會建立新的 IBM MQ 訂閱。 訂閱名稱包括下列資訊：

- 用戶端的名稱。 如果用戶端加入共用訂閱，則會使用共用名稱
- 用戶端訂閱的主題型樣
- 字首。 如果用戶端建立非共用訂閱，則字首為 `private`；如果用戶端加入共用訂閱，則字首為 `share`

若要檢視特定 AMQP 用戶端正在使用的訂閱，請執行 **DISPLAY SUB** 指令並過濾 private 字首:

```
DISPLAY SUB(':private:*')
```

若要檢視多個用戶端正在使用的共用訂閱，請執行 **DISPLAY SUB** 指令並過濾 share 字首:

```
DISPLAY SUB(':share:*')
```

因為共用訂閱可供多個 AMQP 用戶端使用，所以您可能想要檢視目前從共用訂閱耗用訊息的用戶端。 您可以透過列出目前在訂閱佇列上開啟控點的 Hconns 來執行此動作。 若要檢視目前使用共用的用戶端，請完成下列步驟:

1. 尋找共用訂閱用作目的地的佇列名稱。 例如:

```
DISPLAY SUB(':private:recv_e298452:public') DEST
   5 : DISPLAY SUB(':private:recv_e298452:public') DEST
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120514D312020202020202020707E0A565C2D0020)
   SUB(:private:recv_e298452:public)
   DEST(SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
```

2. 執行 **DISPLAY CONN** 指令，以尋找在該佇列上開啟的控點:

```
DISPLAY CONN(*) TYPE(HANDLE) WHERE  (OBJNAME
EQ SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
   21 : DISPLAY CONN(*) TYPE(HANDLE) WHERE(OBJNAME EQ
SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)

AMQ8276: Display Connection details.
   CONN(707E0A56642B0020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(HANDLE)

   OBJNAME(SYSTEM.BASE.TOPIC)      OBJTYPE(TOPIC)

   OBJNAME(SYSTEM.MANAGED.DURABLE.560A7E7020002961)
   OBJTYPE(QUEUE)
```

3. 針對每一個控點，檢視已開啟控點的 AMQP 用戶端 ID:

```
DISPLAY CONN(707E0A56642B0020) CLIENTID
   23 : DISPLAY CONN(707E0A56642B0020) CLIENTID

AMQ8276: Display Connection details.
   CONN(707E0A56642B0020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(CONN)
   CLIENTID(recv_8f02c9d)
DISPLAY CONN(707E0A565F290020) CLIENTID
   24 : DISPLAY CONN(707E0A565F290020) CLIENTID
AMQ8276: Display Connection details.
   CONN(707E0A565F290020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(CONN)
   CLIENTID(recv_86d8888)
```

AMQP 用戶端識別、授權及鑑別

與其他 IBM MQ 用戶端應用程式一樣，您可以透過多種方式來保護 AMQP 連線的安全。

您可以使用下列安全特性來保護 IBM MQ 的 AMQP 連線安全:

- 通道鑑別記錄
- 連線鑑別
- 通道 MCA 使用者配置
- IBM MQ 權限定義
- TLS 連線功能

從安全角度來看，建立連線包含下列兩個步驟:

- 決定連線是否應繼續
- 決定應用程式假設稍後進行權限檢查的 IBM MQ 身分

下列資訊概述不同的 IBM MQ 配置，以及 AMQP 用戶端嘗試建立連線時所完成的步驟。 並非所有 IBM MQ 配置都使用說明的所有步驟。 例如，部分配置不會將 TLS 用於公司防火牆內的連線，部分配置會使用 TLS，但不會使用用戶端憑證進行鑑別。 許多環境不使用自訂或自訂 JAAS 模組。

建立連線

下列步驟說明 AMQP 用戶端正在建立連線時所發生的情況。 這些步驟決定連線是否繼續，以及應用程式假設進行權限檢查的 IBM MQ 身分:

1. 如果用戶端開啟與 IBM MQ 的 TLS 連線並提供憑證，則佇列管理程式會嘗試驗證用戶端憑證。
2. 如果用戶端提供使用者名稱及密碼認證，則佇列管理程式會接收 AMQP SASL 訊框，並檢查 MQ CONNAUTH 配置。
3. 會檢查 MQ 通道鑑別規則 (例如， IP 位址及 TLS 憑證 DN 是否有效)
4. 除非通道鑑別規則另有決定，否則會主張通道 MCAUSER。
5. 如果已配置 JAAS 模組，則會呼叫它
6. MQ CONNECT 權限檢查套用至產生的 MQ 使用者 ID。
7. 以假設的 IBM MQ 身分建立連線。

發佈訊息

下列步驟說明 AMQP 用戶端發佈訊息時所發生的情況。 這些步驟決定連線是否繼續，以及應用程式假設進行權限檢查的 IBM MQ 身分:

1. AMQP 鏈結連接訊框到達佇列管理程式。 會針對連線期間所建立的 MQ 使用者身分，檢查指定主題字串的 IBM MQ 發佈權限。
2. 訊息已發佈至指定的主題字串。

訂閱主題型樣

下列步驟說明 AMQP 用戶端訂閱主題型樣時發生的情況。 這些步驟決定連線是否繼續，以及應用程式假設進行權限檢查的 IBM MQ 身分:

1. AMQP 鏈結連接訊框到達佇列管理程式。 會針對連線期間所建立的 MQ 使用者身分，檢查指定主題型樣的 IBM MQ 訂閱權限。
2. 已建立訂閱。

AMQP 用戶端身分及授權

使用 AMQP 用戶端 ID、AMQP 使用者名稱或在通道或通道鑑別規則中定義的一般用戶端身分，以授權存取 IBM MQ 物件。

管理者在定義或修改 AMQP 通道、配置佇列管理程式 CONNAUTH 設定或定義通道鑑別規則時做出選擇。 身分用於授與 IBM MQ 主題的存取權。 根據下列各項來進行選擇:

1. 通道 USECLNTID 屬性。
2. 佇列管理程式 CONNAUTH 規則的 ADOPTCTX 屬性。
3. 通道上定義的 MCAUSER 屬性。
4. 相符通道鑑別規則的 USERSRC 屬性。

避免麻煩: 之後會參照此處理程序所選擇的身分，例如 DISPLAY CHSTATUS (AMQP) 指令，作為用戶端的 MCAUSER。 請注意，這不一定與選項 (2) 中所參照通道的 MCAUSER 相同。

使用 IBM MQ **setmqaut** 指令來選取與 AMQP 通道相關聯的身分所授權使用的物件及動作。 例如，下列指令會授權佇列管理程式 QM1 的管理者所提供的通道身分 AMQPClient:

```
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPClient -all +pub +sub
```

和

```
setmqaut -m QM1 -t qmgr -p AMQPClient -all +connect
```

使用密碼進行 AMQP 用戶端鑑別

使用用戶端密碼鑑別 AMQP 用戶端使用者名稱。 您可以使用不同於用來授權用戶端發佈及訂閱主題之身分的身分來鑑別用戶端。

AMQP 服務可以使用 MQ CONNAUTH 或 JAAS 來鑑別用戶端使用者名稱。 如果已配置其中一個，則 MQ CONNAUTH 配置或 JAAS 模組會驗證用戶端提供的密碼。

下列程序概述針對本端 OS 使用者及密碼鑑別個別使用者的範例步驟，如果成功，則採用一般身分 AMQPUser:

1. IBM MQ 管理者會使用「IBM MQ 探險家」將 AMQP 通道 MCAUSER 身分設為任何名稱，例如 AMQPUser。
2. IBM MQ 管理者授權 AMQPUser 發佈及訂閱任何主題:

   ```
   setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPUser -all +pub +sub +connect
   ```

3. IBM MQ 管理者會配置 IDPWOS CONNAUTH 規則，以檢查用戶端提供的使用者名稱及密碼。 CONNAUTH 規則應該設定 CHCKCLNT (REQUIRED) 及 ADOPTCTX (NO)。

註: 建議您使用通道鑑別規則，並將 MCAUSER 通道屬性設為沒有專用權的使用者，以容許對佇列管理程式連線有更多控制權。

通道上的發佈隱私權

透過使用 TLS 來加密透過連線的傳輸，可保護透過任一方向透過 AMQP 通道傳送之 AMQP 發佈的隱私權。

連接至 AMQP 通道的 AMQP 用戶端會使用 TLS 來保護通道上使用對稱金鑰加密法所傳輸發佈的隱私權。 因為不會鑑別端點，所以您不能信任單獨使用的通道加密。 將保密安全與伺服器或交互鑑別結合使用。

作為使用 TLS 的替代方案，部分類型的「虛擬專用網路 (VPN)」(例如 IPsec) 會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 AMQP 用戶端連接至 AMQP 通道。

在不鑑別伺服器的情況下加密 TLS 連線會對中間人攻擊公開連線。 雖然您交換的資訊可以防竊聽，但是您不知道與您交換資訊的對象是誰。 除非您控制網路，否則您就會面臨別人截取您的 IP 傳輸，以及假冒端點的問題。

您可以使用支援匿名 TLS 的 Diffie-Hellman 金鑰交換 CipherSpec 來建立已加密 TLS 連線，而無需鑑別伺服器。 在用戶端與伺服器之間共用且用來加密 TLS 傳輸的主要密碼，是在不交換私密簽署伺服器憑證的情況下建立的。

因為匿名連線不安全，大部分 TLS 實作不會預設為使用匿名 CipherSpec。 如果 AMQP 通道接受 TLS 連線的用戶端要求，則通道必須具有受通行詞組保護的金鑰儲存庫。 依預設，由於 TLS 實作不使用匿名 CipherSpec ，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。

使用 TLS 設定 AMQP 用戶端

您可以將 AMQP 用戶端設定為使用 TLS 來保護網路中流動的資料並驗證用戶端連接到的佇列管理器的身份。

要在 AMQP 用戶端到 AMQP 通道的連接中使用 TLS，您必須確保已設定佇列管理員使用 TLS。 在佇列管理器上設定 TLS 描述如何設定佇列管理器從中讀取 TLS 憑證的金鑰庫。

當佇列管理器配置了金鑰庫後，您必須在用戶端將連線到的 AMQP 通道上設定 TLS 屬性。 AMQP 通道有四個與 TLS 配置相關的屬性，如下所示：

SSLCAUTH
SSLCAUTH 屬性用於指定佇列管理器是否應要求 AMQP 用戶端提供用戶端憑證來驗證其身分。

SSLCIPH
SSLCIPH 屬性指定通道套用於對 TLS 流中的資料進行編碼的密碼。

SSLPEER
SSLPEER 屬性用於指定用戶端憑證必須符合的專有名稱 (DN)（如果允許連線）。

CERTLABL
CERTLBL 指定佇列管理器應提供給客戶端的憑證。 佇列管理器的金鑰庫可以包含多個憑證。 此屬性允許您指定用於連接到此通道的憑證。 如果未指定 CERTLBL，則使用佇列管理器金鑰儲存庫中的憑證以及與佇列管理器 CERTLBL 屬性對應的標籤。

使用 TLS 屬性配置 AMQP 通道後，必須使用下列命令重新啟動 AMQP 服務：

```
STOP SERVICE(SYSTEM.AMQP.SERVICE)
START SERVICE(SYSTEM.AMQP.SERVICE)
```

當 AMQP 用戶端連接到受 TLS 保護的 AMQP 通道時，用戶端會驗證佇列管理器提供的憑證的身份。 為此，您必須使用包含佇列管理器憑證的信任庫來設定 AMQP 用戶端。 執行此操作的步驟會因您使用的 AMQP 用戶端而異。 有關各種 AMQP 用戶端和 API 的詳細資訊，請參閱各自的 AMQP 用戶端文件。

斷開 AMQP 用戶端與佇列管理器的連接

如果要中斷 AMQP 用戶端與佇列管理器的連接，請執行 PURGE CHANNEL 指令或停止與 AMQP 用戶端的連線。

- 運行 **PURGE CHANNEL** 命令。 例如：

  ```
  PURGE CHANNEL(MYAMQP) CLIENTID('recv_28dbb7e')
  ```

- 或者，透過完成以下步驟停止 AMQP 用戶端正在使用的連線以中斷客戶端連線：

  1. 透過執行 **DISPLAY CONN** 命令來尋找客戶端正在使用的連線。 例如：

     ```
     DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_28dbb7e')
     ```

     命令輸出如下：

     ```
     DISPLAY CONN(*)  TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
       40 : DISPLAY CONN(*) TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
     AMQ8276: Display  Connection details.
       CONN(707E0A565F2D0020)
       EXTCONN(414D5143514D31202020202020202020)
       TYPE(CONN)
       CLIENTID(recv_28dbb7e)
     ```

  2. 停止連接。 例如：

     ```
     STOP CONN(707E0A565F2D0020)
     ```

管理多重播送

使用此資訊來瞭解「IBM MQ 多重播送」管理作業，例如減少多重播送訊息的大小及啟用資料轉換。

開始使用多重播送

使用此資訊來開始使用 IBM MQ Multicast 主題及通訊資訊物件。

關於這項作業

IBM MQ Multicast 傳訊透過將主題對映至群組位址，使用網路來遞送訊息。 下列作業是快速測試所需的 IP 位址和埠是否已正確配置多重播送傳訊的方法。

建立多重播送的 COMMINFO 物件

通訊資訊 (COMMINFO) 物件包含與多重播送傳輸相關聯的屬性。 如需 COMMINFO 物件參數的相關資訊，請參閱 DEFINE COMMINFO。

請使用下列指令行範例來定義多重播送的 COMMINFO 物件:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

其中 *MC1* 是 COMMINFO 物件的名稱， *group address* 是您的群組多重播送 IP 位址或 DNS 名稱， *port number* 是要傳輸的埠 (預設值為 1414)。

會建立稱為 *MC1* 的新 COMMINFO 物件; 此名稱是您在下一個範例中定義 TOPIC 物件時必須指定的名稱。

建立 TOPIC 物件以進行多重播送

主題是發佈/訂閱訊息中所發佈資訊的主旨，而主題是透過建立 TOPIC 物件來定義。 TOPIC 物件有兩個參數，可定義它們是否可以與多重播送一起使用。 這些參數為: **COMMINFO** 和 **MCAST**。

- **COMMINFO** 此參數指定多重播送通訊資訊物件的名稱。 如需 COMMINFO 物件參數的相關資訊，請參閱 DEFINE COMMINFO。
- **MCAST** 此參數指定主題樹狀結構中的這個位置是否容許多重播送。

使用下列指令行範例來定義多重播送的 TOPIC 物件:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

即會建立稱為 *ALLSPORTS* 的新 TOPIC 物件。 它具有主題字串 *Sports*，其相關通訊資訊物件稱為 *MC1* (在前一個範例中定義 COMMINFO 物件時指定的名稱) ，並且已啟用多重播送。

測試多重播送發佈/訂閱

建立 TOPIC 和 COMMINFO 物件之後，可以使用 amqspubc 範例和 amqssubc 範例來測試它們。 如需這些範例的相關資訊，請參閱 發佈/訂閱範例程式。

1. 開啟兩個指令行視窗; 第一個指令行用於 amqspubc 發佈範例，第二個指令行用於 amqssubc 訂閱範例。
2. 在指令行 1 輸入下列指令:

   ```
   amqspubc Sports QM1
   ```

   其中 *Sports* 是先前範例中所定義 TOPIC 物件的主題字串，而 *QM1* 是佇列管理程式的名稱。
3. 在指令行 2 輸入下列指令:

   ```
   amqssubc Sports QM1
   ```

   其中 體育 和 *QM1* 與步驟 第 285 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; amqssubc 範例 (在埠上接聽來自指定位址的發佈) 會在指令行 2 輸出 `Hello world`。

IBM MQ 多重播送主題拓蹼

使用此範例來瞭解 IBM MQ Multicast 主題拓蹼。

IBM MQ 多重播送支援需要每個子樹狀結構在總計階層內都有自己的多重播送群組及資料串流。

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。 IP 位址的完整多重播送範圍是 224.0.0.0 至 239.255.255.255，但是其中部分位址已保留。 如需保留位址清單，請聯絡您的系統管理者，或參閱 https://www.iana.org/assignments/multicast-addresses 以取得相關資訊。 建議您使用 239.0.0.0 至 239.255.255.255 範圍內的本端範圍多重播送位址。

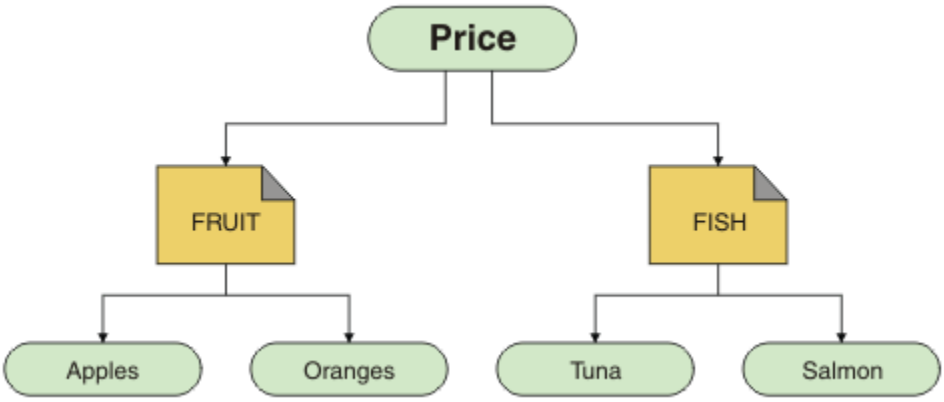
在下圖中，有兩個可能的多重播送資料串流:

```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX)
DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

其中 *239.XXX.XXX.XXX* 和 *239.YYY.YYY.YYY* 是有效的多重播送位址。

這些主題定義用來建立主題樹狀結構，如下圖所示:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)
DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



每一個多重播送通訊資訊 (COMMINFO) 物件都代表不同的資料串流，因為它們的群組位址不同。 在此範例中， FRUIT 主題定義為使用 COMMINFO 物件 MC1 ， FISH 主題定義為使用 COMMINFO 物件 MC2 ， 並且 Price 節點沒有多重播送定義。

IBM MQ 多重播送的主題字串限制為 255 個字元。 此限制表示必須小心處理樹狀結構內節點及葉節點的名稱; 如果節點及葉節點的名稱太長，則主題字串可能會超出 255 個字元，並傳回 2425 (0979) (RC2425) :MQRC_TOPIC_STRING_ERROR 原因碼。 建議讓主題字串盡可能簡短，因為較長的主題字串可能會對效能造成不利影響。

控制多重播送訊息的大小

使用此資訊來瞭解 IBM MQ 訊息格式，並減少 IBM MQ 訊息的大小。

IBM MQ 訊息具有一些相關聯的屬性，這些屬性包含在訊息描述子中。 對於小型訊息，這些屬性可能代表大部分資料流量，並且可能對傳輸速率產生重大不利影響。 IBM MQ Multicast 可讓使用者配置隨訊息一起傳輸這些屬性 (如果有的話)。

訊息屬性 (非主題字串) 的存在取決於 COMMINFO 物件是否指出必須傳送它們。 如果未傳輸屬性，接收端應用程式會套用預設值。 預設 MQMD 值不一定與 MQMD_DEFAULT 值相同，並在 第 287 頁的表 19 中說明。

COMMINFO 物件包含 MCPROP 屬性，可控制有多少 MQMD 欄位及使用者內容與訊息一起流動。 透過將此屬性的值設為適當的層次，您可以控制「IBM MQ 多重播送」訊息的大小:

MCPROP
多重播送內容控制與訊息一起傳送的 MQMD 內容及使用者內容數。

ALL
會傳輸 MQMD 的所有使用者內容及所有欄位。

回覆
只傳輸使用者內容，以及處理訊息回覆的 MQMD 欄位。 這些內容如下:

- MsgType
- MessageId
- CorrelId
- ReplyToQ

- ReplyToQmgr

使用者
只傳輸使用者內容。

無
不傳輸任何使用者內容或 MQMD 欄位。

COMPAT
此值會導致以相容模式將訊息傳輸至 RMM，這容許與現行 XMS 應用程式及 IBM Integration Bus RMM 應用程式進行一些交互作業。

V 9.2.0 V 9.2.0 XMS .NET 多播訊息傳遞（使用 RMM ）已從 IBM MQ 9.2 中棄用，並將在 XMS .NET 的未來版本中刪除。

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。

下表顯示根據 MCPROP (本節先前說明) 的值傳送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 *19:* 傳訊屬性及其與多重播送的關係

屬性	使用多重播送時的動作	如果未傳輸，則為預設值
TopicString	一律併入	不適用
MQMQ StrucId	未傳輸	不適用
MQMD 版本	未傳輸	不適用
報告	如果不是預設值則併入	0
MsgType	如果不是預設值則併入	MQMT_DATAGRAM
期限	如果不是預設值則併入	0
意見	如果不是預設值則併入	0
編碼	如果不是預設值則併入	MQENC_NORMAL (equiv)
CodedCharSetId	如果不是預設值則併入	1208
格式	如果不是預設值則併入	MQRFH2
優先順序	如果不是預設值則併入	4
持續性	如果不是預設值則併入	MQPER_NOT_PERSISTENT
MsgId	如果不是預設值則併入	空值
CorrelId	如果不是預設值則併入	空值
BackoutCount	如果不是預設值則併入	0
ReplyToQ	如果不是預設值則併入	Blank
回覆目的地佇列管理程式	如果不是預設值則併入	Blank
UserIdentifier	如果不是預設值則併入	Blank
AccountingToken	如果不是預設值則併入	空值
PutAppIType	如果不是預設值則併入	MQAT_JAVA
PutApp 名稱	如果不是預設值則併入	Blank
PutDate	如果不是預設值則併入	Blank
PutTime	如果不是預設值則併入	Blank

表 19: 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**
ApplOriginData	如果不是預設值則併入	Blank
GroupID	已排除	不適用
MsgSeqNumber	已排除	不適用
偏移	已排除	不適用
MsgFlags	已排除	不適用
OriginalLength	已排除	不適用
UserProperties	已包括	不適用

相關參考
ALTER COMMINFO
DEFINE COMMINFO

啟用多重播送傳訊的資料轉換

使用此資訊來瞭解 IBM MQ Multicast 傳訊的資料轉換如何運作。

IBM MQ Multicast 是一種共用的無連線通訊協定，因此每一個用戶端都無法提出特定要求來進行資料轉換。每個訂閱相同多重播送串流的用戶端都會接收相同的二進位資料; 因此，如果需要 IBM MQ 資料轉換，則會在每個用戶端本端執行轉換。

在混合平台安裝中，可能是大部分用戶端需要的資料格式不是傳輸應用程式的原始格式。 在此情況下，可以使用多重播送 COMMINFO 物件的 **CCSID** 及 **ENCODING** 值來定義訊息傳輸的編碼，以提高效率。

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

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

除了這些格式之外，您還可以定義自己的格式，並使用 MQDXP-資料轉換結束程式參數 資料轉換結束程式。

如需程式設計資料轉換的相關資訊，請參閱 MQI for 多重播送傳訊中的資料轉換。

如需資料轉換的相關資訊，請參閱 資料轉換。

如需資料轉換結束程式及 ClientExitPath 的相關資訊，請參閱 用戶端配置檔的 ClientExit 路徑段落。

多重播送應用程式監視

使用此資訊來瞭解管理及監視 IBM MQ Multicast。

多重播送資料流量的現行發佈者及訂閱者的狀態 (例如，傳送及接收的訊息數，或遺失的訊息數) 會定期從用戶端傳輸至伺服器。 收到狀態時， COMMINFO 物件的 COMMEV 屬性會指定佇列管理程式是否將事件訊息放置在 SYSTEM.ADMIN.PUBSUB.EVENT。 事件訊息包含收到的狀態資訊。 此資訊是尋找問題來源的寶貴診斷輔助。

使用 MQSC 指令 **DISPLAY CONN** 可顯示連接至佇列管理程式之應用程式的連線資訊。 如需 **DISPLAY CONN** 指令的相關資訊，請參閱 DISPLAY CONN。

使用 MQSC 指令 **DISPLAY TPSTATUS** 來顯示發佈者和訂閱者的 狀態。 如需 **DISPLAY TPSTATUS** 指令的相關資訊，請參閱 DISPLAY TPSTATUS。

COMMEV 及多重播送訊息可靠性指示器

可靠性指示器與 COMMINFO 物件的 **COMMEV** 屬性一起使用，是監視 IBM MQ Multicast 發佈者和訂閱者的關鍵元素。 可靠性指示器（發布或 **MCASTREL** 狀態命令返回的 **MCASTREL** 字段）是一個 IBM MQ 指示器，說明沒有錯誤的傳輸的百分比。 傳輸錯誤的潛在原因包括使用者緩慢、網路忙碌及網路中斷。 **COMMEV** 控制是否針對使用 COMMINFO 物件所建立的多重播送控點，產生事件訊息，並設為下列三個可能值之一:

已停用
不會寫入事件訊息。

ENABLED
一律寫入事件訊息，並在 COMMINFO **MONINT** 參數中定義頻率。

異常狀況
如果訊息可靠性低於可靠性臨界值，則會寫入事件訊息。 90% 或更小的訊息可靠性層次指出網路配置可能有問題，或一個以上「發佈/訂閱」應用程式執行太慢:

- 值 **MCASTREL(100,100)** 表示在短期或長期時間範圍內沒有任何問題。
- 值 **MCASTREL(80,60)** 表示 20% 的訊息目前有問題，但它也比長期值 60 有所改善。

即使佇列管理程式的單點播送連線中斷，用戶端仍可能繼續傳輸及接收多重播送資料流量，因此資料可能過期。

多重播送訊息可靠性

使用此資訊來瞭解如何設定 IBM MQ Multicast 訂閱及訊息歷程。

克服多重播送傳輸失敗的關鍵元素是 IBM MQ 對已傳輸資料 (要保留在鏈結傳輸端的訊息歷程) 的緩衝。 此處理程序表示在放置應用程式程序中不需要緩衝訊息，因為 IBM MQ 提供可靠性。 此歷程的大小是透過通訊資訊 (COMMINFO) 物件來配置，如下列資訊中所述。 較大的傳輸緩衝意味著在需要時需要重新傳輸的傳輸歷程更多，但由於多重播送的本質，無法支援 100% 的保證遞送。

IBM MQ Multicast 訊息歷程在通訊資訊 (COMMINFO) 物件中由 **MSGHIST** 屬性控制:

MSGHIST
此值是系統保留來處理在 ACK (負值確認通知) 情況下重新傳輸的訊息歷程數量 (以 KB 為單位)。

值 0 會提供最低可靠性層次。 預設值為 100 KB。

IBM MQ Multicast 新訂閱歷程在通訊資訊 (COMMINFO) 物件中由 **NSUBHIST** 屬性控制:

NSUBHIST
新訂閱者歷程控制加入發佈串流的訂閱者是否收到目前所有可用的資料，或只收到訂閱後的發佈。

無
NONE 值會導致轉送器僅傳輸從訂閱時間開始的發佈。 NONE 是預設值。

ALL
ALL 值會導致轉送器重新傳輸已知的主題歷程。 在某些情況下，此狀況會對保留的發佈提供類似的行為。

註: 如果因為重新傳輸所有主題歷程而有大型主題歷程，則使用 ALL 值可能會對效能造成不利影響。

相關參考
DEFINE COMMINFO
ALTER COMMINFO

進階多重播送作業

使用此資訊來瞭解進階 IBM MQ Multicast 管理作業，例如配置 .ini 檔案以及與 IBM MQ LLM 的交互作業能力。

如需多重播送安裝中的安全考量，請參閱 多重播送安全。

在多重播送與非多重播送發佈/訂閱網域之間橋接

使用此資訊來瞭解當非多重播送發佈者發佈至已啟用 IBM MQ 多重播送的主題時所發生的情況。

如果非多重播送發佈者發佈至定義為 **MCAST** 已啟用且 **BRIDGE** 已啟用的主題，則佇列管理程式會透過多重播送直接將訊息傳輸出至任何可能正在接聽的訂閱者。 多重播送發佈者無法發佈至未啟用多重播送的主題。

透過設定主題物件的 **MCAST** 及 **COMMINFO** 參數，可以啟用多重播送現有主題。 如需這些參數的相關資訊，請參閱 起始多重播送概念 。

COMMINFO 物件 **BRIDGE** 屬性控制來自未使用多重播送之應用程式的發佈。 如果 **BRIDGE** 設為 ENABLED ，且主題的 **MCAST** 參數也設為 ENABLED，則來自未使用多重播送之應用程式的發佈會橋接至所使用的應用程式。 如需 **BRIDGE** 參數的相關資訊，請參閱 DEFINE COMMINFO。

配置「多重播送」的 .ini 檔

使用此資訊來瞭解 `.ini` 檔案中的 IBM MQ 多重播送欄位。

可以在 `ini` 檔案中進行其他 IBM MQ 多重播送配置。 您必須使用的特定 `ini` 檔案取決於應用程式類型:

- 用戶端: 配置 *MQ_DATA_PATH* `/mqclient.ini` 檔案。
- 佇列管理程式: 配置 *MQ_DATA_PATH* `/qmgrs/`*QMNAME*`/qm.ini` 檔案。

其中 *MQ_DATA_PATH* 是 IBM MQ 資料目錄 (`/var/mqm/mqclient.ini`) 的位置， *QMNAME* 是套用 `.ini` 檔案的佇列管理程式名稱。

`.ini` 檔案包含用來細部調整 IBM MQ 多重播送行為的欄位:

```
Multicast:
Protocol       = IP | UDP
IPVersion       = IPv4 | IPv6 | ANY | BOTH
LimitTransRate    = DISABLED | STATIC | DYNAMIC
TransRateLimit    = 100000
SocketTTL       = 1
Batch         = NO
Loop         = 1
Interface       = <IPaddress>
FeedbackMode     = ACK | NACK | WAIT1
HeartbeatTimeout   = 20000
HeartbeatInterval   = 2000
```

通訊協定

UDP
在此模式中，會使用 UDP 通訊協定來傳送封包。 然而，網路元素無法像在 IP 模式中一樣在多重播送中提供協助。 封包格式仍與 PGM 相容。 這是預設值。

IP
在此模式中，轉送器會傳送原始 IP 封包。 具有 PGM 支援的網路元素可協助進行可靠的多重播送封包配送。 此模式與 PGM 標準完全相容。

IPVersion

IPv4
僅使用 IPv4 通訊協定進行通訊。 這是預設值。

IPv6
僅使用 IPv6 通訊協定進行通訊。

ANY
視可用的通訊協定而定，使用 IPv4 及/或 IPv6 進行通訊。

BOTH
支援使用 IPv4 和 IPv6 進行通訊。

LimitTransRate

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。 發送器的發送速率不會超過 TransRateLimit 參數指定的速率。

動態
發射機根據從接收機獲得的反饋來調整其傳輸速率。 在這種情況下，傳輸速率限制不能大於 TransRateLimit 參數所指定的值。 傳送器嘗試達到最佳傳輸速率。

TransRateLimit
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:

- 否 訊息不會批次處理，會立即傳送。
- *YES* 訊息已批次處理。

迴圈
將值設為 1 可啟用多重播送迴圈。 多重播送迴圈定義傳送的資料是否迴圈回主電腦。

介面
多重播送資料流量在其上流動之介面的 IP 位址。 如需相關資訊及疑難排解，請參閱: 在非多重播送網路上測試多重播送應用程式 及 針對多重播送資料流量設定適當的網路

FeedbackMode

NACK
負確認通知的意見。 這是預設值。

ACK
正面確認通知的意見。

WAIT1
由正面確認通知所提供的回饋，其中轉送器只會等待來自任何接收端的 1 個 ACK。

HeartbeatTimeout
活動訊號逾時 (毫秒)。 值 0 表示主題的一或多個接收端不會產生活動訊號逾時事件。 預設值為 20000。

HeartbeatInterval
活動訊號間隔 (毫秒)。 值 0 表示不傳送活動訊號。 活動訊號間隔必須遠小於 **HeartbeatTimeout** 值，以避免錯誤活動訊號逾時事件。 預設值為 2000。

多重播送與 IBM MQ 低延遲傳訊的交互作業能力

使用此資訊來瞭解 IBM MQ 多重播送與 IBM MQ 低延遲傳訊 (LLM) 之間的交互作業能力。

對於使用 LLM 的應用程式而言，基本有效負載傳送是可能的，而另一個應用程式則使用多重播送來雙向交換訊息。 雖然多重播送使用 LLM 技術，但 LLM 產品本身並未內嵌。 因此，可以同時安裝 LLM 和 IBM MQ Multicast ，並分別操作和服務這兩個產品。

與多重播送通訊的 LLM 應用程式可能需要傳送及接收訊息內容。 IBM MQ 訊息內容及 MQMD 欄位會以 LLM 訊息內容傳輸，並具有下表所示的特定 LLM 訊息內容碼:

表 *20: IBM MQ* 訊息內容至 *IBM MQ LLM* 內容對映

IBM MQ 內容 (property)	**IBM MQ LLM 內容類型**	**LLM 內容類型**	**LLM 內容碼**
MQMD.Report	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1001
MQMD.MsgType	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1002
MQMD.Expiry	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1003
MQMD.Feedback	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1004
MQMD.Encoding	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1005
MQMD.CodedCharSetId	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1006
MQMD.Format	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1007

表 20: IBM MQ 訊息內容至 IBM MQ LLM 內容對映 (繼續)			
IBM MQ 內容 (property)	**IBM MQ LLM 內容類型**	**LLM 內容類型**	**LLM 內容碼**
MQMD.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: IBM MQ 低延遲傳訊。

IBM i 管理 IBM MQ for IBM i

介紹可讓您在 IBM i 上管理 IBM MQ 的方法。

管理作業包括建立、啟動、變更、檢視、停止及刪除叢集、處理程序及 IBM MQ 物件 (佇列管理程式、佇列、名稱清單、處理程序定義、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件)。

如需如何管理 IBM MQ for IBM i 的詳細資料，請參閱下列鏈結:

- 第 292 頁的『使用 CL 指令管理 IBM MQ for IBM i』
- 第 304 頁的『管理 IBM MQ for IBM i 的替代方式』
- 第 309 頁的『IBM i 的工作管理』

相關概念

第 314 頁的『IBM i 上的可用性、備份、回復及重新啟動』
使用此資訊來瞭解 IBM MQ for IBM i 如何使用 IBM i 日誌登載支援來協助其備份及還原策略。

瞭解 IBM MQ for IBM i 佇列管理程式檔案庫名稱

相關工作

變更 IBM i 上的配置資訊

在 IBM i 上設定安全

相關參考

第 350 頁的『靜止 IBM MQ for IBM i』
本節說明如何靜止 (循序結束) IBM MQ for IBM i。

第 131 頁的『IBM i 上無法傳送郵件的佇列處理程式』
何謂 IBM i 無法傳送郵件的佇列處理程式，以及如何呼叫它?

判斷 IBM MQ for IBM i 應用程式的問題

IBM i 上的可安裝服務及元件

系統及預設物件

IBM i 使用 CL 指令管理 IBM MQ for IBM i

使用此資訊來瞭解 IBM MQ IBM i 指令。

大部分 IBM MQ 指令群組 (包括與佇列管理程式、佇列、主題、通道、名稱清單、程序定義及鑑別資訊物件相關聯的指令) 都可以使用相關 **WRK*** 指令來存取。

該集中的主體指令是 **WRKMQM**。 例如，此指令可讓您顯示系統上所有佇列管理程式的清單，以及狀態資訊。 或者，您可以針對每一個項目使用各種選項，來處理所有佇列管理程式特定的指令。

從 **WRKMQM** 指令中，您可以選取每一個佇列管理程式的特定區域 (例如，使用通道、主題或佇列)，並從中選取個別物件。

記錄 IBM MQ 應用程式定義

當您建立或自訂 IBM MQ 應用程式時，保留所建立之所有 IBM MQ 定義的記錄非常有用。 此記錄可用於:

- 回復目的
- 維護
- 推出 IBM MQ 應用程式

您可以使用下列兩種方式之一來記錄 IBM MQ 應用程式定義:

1. 建立 CL 程式以產生伺服器的 IBM MQ 定義。
2. 建立 MQSC 文字檔作為 SRC 成員，以使用跨平台 IBM MQ 指令語言來產生 IBM MQ 定義。

如需定義佇列物件的進一步詳細資料，請參閱 第 11 頁的『使用 MQSC 指令進行管理』 及 第 22 頁的『使用 IBM MQ 可程式指令格式』。

相關參考

IBM MQ for IBM i CL 指令參照

IBM i 開始使用 IBM MQ for IBM i 之前，請使用 CL 指令

使用此資訊來啟動 IBM MQ 子系統並建立本端佇列管理程式。

開始之前

請確定 IBM MQ 子系統正在執行中 (使用指令 STRSBS QMQM/QMQM)，且未保留與該子系統相關聯的工作佇列。 依預設，在檔案庫 QMQM 中， IBM MQ 子系統及工作佇列都命名為 QMQM。

關於這項作業

使用 IBM i 指令行啟動佇列管理程式

程序

1. 從 IBM i 指令行發出 CRTMQM 指令來建立本端佇列管理程式。

   當您建立佇列管理程式時，您可以選擇讓該佇列管理程式成為預設佇列管理程式。 如果省略佇列管理程式名稱參數 (MQMNAME)，則預設佇列管理程式 (只能有一個) 是 CL 指令適用的佇列管理程式。
2. 從 IBM i 指令行發出 STRMQM 指令，以啟動本端佇列管理程式。

   如果佇列管理程式啟動花費數秒以上的時間， IBM MQ 會顯示間歇性詳細說明啟動進度的狀態訊息。 如需這些訊息的相關資訊，請參閱 訊息及原因碼。

下一步

您可以從 IBM i 指令行發出 ENDMQM 指令來停止佇列管理程式，並從 IBM i 指令行發出其他 IBM MQ 指令來控制佇列管理程式。

遠端佇列管理程式無法從遠端啟動，但必須由本端操作員在其系統中建立並啟動。 但有遠端作業機能 (在 IBM MQ for IBM i 之外) 可啟用這類作業的情況例外。

本端佇列管理者無法停止遠端佇列管理程式。

註: 在靜止 IBM MQ 系統的過程中，您必須靜止作用中佇列管理程式。 這說明於第 350 頁的『靜止 IBM MQ for IBM i』。

IBM i 建立 IBM MQ for IBM i 物件

使用此資訊來瞭解為 IBM i 建立 IBM MQ 物件的方法。

開始之前

下列作業建議從指令行使用 IBM MQ for IBM i 的各種方式。

關於這項作業

有兩種線上方法可建立 IBM MQ 物件:

程序

1. 使用 Create 指令，例如: **Create MQM Queue** 指令: **CRTMQMQ**
2. 使用「使用 MQM 物件」指令，後面接著 F6，例如: **Work with MQM Queues** 指令: **WRKMQMQ**

下一步

如需所有指令的清單，請參閱 IBM MQ for IBM i CL 指令。

註: 可以從「訊息佇列管理程式指令」功能表提交所有 MQM 指令。 若要顯示此功能表，請在指令行上鍵入 GO CMDMQM，然後按 Enter 鍵。

當您從此功能表中選取指令時，系統會自動顯示提示畫面。 若要顯示您直接在指令行上鍵入之指令的提示畫面，請在按下 Enter 鍵之前按 F4。

使用 *CRTMQMQ* 指令建立本端佇列

程序

1. 在指令行上鍵入 CHGMQM，然後按 F4 鍵。
2. 在 **建立 MQM 佇列畫面**上，在 Queue name 欄位中鍵入您要建立的佇列名稱。 若要指定大小寫混合的名稱，請以單引號括住名稱。
3. 在 Queue type 欄位中鍵入 *LCL。
4. 除非您使用預設佇列管理程式，否則請指定佇列管理程式名稱，然後按 Enter 鍵。 您可以使用新值改寫任何值。 向前捲動以查看進一步的欄位。 用於叢集的選項位於選項清單的結尾。
5. 當您變更任何值時，請按 Enter 鍵以建立佇列。

使用 *WRKMQMQ* 指令建立本端佇列

程序

1. 在指令行上鍵入 WRKMQMQ。
2. 輸入佇列管理程式的名稱。
3. 如果您要顯示提示畫面，請按 F4。 提示畫面可指定同屬佇列名稱或佇列類型，以減少顯示的佇列數目。
4. 按下 Enter，即會顯示「**使用 MQM 佇列**」畫面。 您可以使用新值來改寫任何值。 向前捲動以查看進一步的欄位。 用於叢集的選項位於選項清單的結尾。
5. 按 F6 以建立新的佇列; 這會將您帶到 **CRTMQMQ** 畫面。 如需如何建立佇列的指示，請參閱 第 294 頁的『使用 CRTMQMQ 指令建立本端佇列』。 當您已建立佇列時，會再次顯示 **使用 MQM 佇列** 畫面。 當您按 F5=Refresh 時，新佇列會新增至清單中。

變更佇列管理程式屬性

關於這項作業

若要變更在 **CHGMQM** 指令上指定之佇列管理程式的屬性，請指定您要變更的屬性及值。 例如，使用下列選項來變更 jupiter.queue.manager 的屬性:

程序

在指令行上鍵入 **CHGMQM**，然後按 F4 鍵。

結果

此指令會變更使用的無法傳送郵件的佇列，並啟用禁止事件。

IBM i 在 IBM i 上使用本端佇列

本節包含可用來管理本端佇列的部分指令範例。 也可以使用 **WRKMQMQ 指令畫面**中的選項來使用所有顯示的指令。

定義本端佇列

對於應用程式，本端佇列管理程式是應用程式所連接的佇列管理程式。 本端佇列管理程式所管理的佇列被認為是該佇列管理程式的本端佇列。

使用指令 **CRTMQMQ QTYPE *LCL** 來建立本端佇列的定義，以及建立稱為佇列的資料結構。 您也可以修改預設本端佇列的佇列性質。

在此範例中，我們定義的佇列 orange.local.queue 指定為具有下列性質:

- 它會啟用取得、停用放置，並以先進先出 (FIFO) 為基礎來運作。
- 它是 一般 佇列，亦即它不是起始佇列或傳輸佇列，且不會產生觸發訊息。
- 佇列深度上限為 1000 則訊息; 訊息長度上限為 2000 個位元組。

下列指令會在預設佇列管理程式上執行此動作:

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL)
TEXT('Queue for messages from other systems')
PUTENBL(*NO)
GETENBL(*YES)
TRGENBL(*NO)
MSGDLYSEQ(*FIFO)
MAXDEPTH(1000)
MAXMSGLEN(2000)
USAGE(*NORMAL)
```

註:

1. USAGE *NORMAL 指出此佇列不是傳輸佇列。
2. 如果相同佇列管理程式上已有名稱為 orange.local.queue 的本端佇列，則此指令會失敗。 如果您要改寫佇列的現有定義，請使用 REPLACE *YES 屬性，但另請參閱 第 296 頁的『變更本端佇列屬性』。

定義無法傳送郵件的佇列

每一個佇列管理程式都必須有一個本端佇列作為無法傳送郵件的佇列，以便儲存無法遞送至其正確目的地的訊息，以供稍後擷取。 您必須明確告知佇列管理程式無法傳送郵件的佇列。 您可以在 **CRTMQM** 指令上指定無法傳送郵件的佇列來執行此動作，也可以稍後使用 **CHGMQM** 指令來指定一個佇列。 您也必須先定義無法傳送郵件的佇列，才能使用它。

產品隨附稱為 SYSTEM.DEAD.LETTER.QUEUE 的無法傳送郵件佇列範例。 當您建立佇列管理程式時，會自動建立此佇列。 必要的話，您可以修改此定義。 不需要重新命名它，但如果您喜歡，可以重新命名。

無法傳送郵件的佇列沒有特殊需求，除了:

- 它必須是本端佇列。
- 其 MAXMSGL (訊息長度上限) 屬性必須啟用佇列，以容納佇列管理程式 **加上** 無法傳送郵件的標頭 (MQDLH) 大小所必須處理的最大訊息。

IBM MQ 提供無法傳送郵件的佇列處理程式，可讓您指定如何處理或移除在無法傳送郵件的佇列上找到的訊息。 如需進一步資訊，請參閱第 131 頁的『IBM i 上無法傳送郵件的佇列處理程式』。

顯示預設物件屬性

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

```
DSPMQMQ QNAME(SYSTEM.DEFAULT.LOCAL.QUEUE) MQMNAME(MYQUEUEMANAGER)
```

複製本端佇列定義

您可以使用 CPYMQMQ 指令來複製佇列定義。 例如:

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

此指令會建立與原始佇列 orange.local.queue 具有相同屬性的佇列，而不是系統預設本端佇列的佇列。

您也可以使用 **CPYMQMQ** 指令來複製佇列定義，但替換原始屬性的一或多項變更。 例如:

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('third.queue') MQMNAME(MYQUEUEMANAGER)
MAXMSGLEN(1024)
```

此指令會將佇列 orange.local.queue 的屬性複製到佇列 third.queue，但指定新佇列上的訊息長度上限為 1024 個位元組，而不是 2000 個位元組。

註: 當您使用 **CPYMQMQ** 指令時，只會複製佇列屬性，不會複製佇列上的訊息。

變更本端佇列屬性

您可以使用 **CHGMQMQ** 指令或具有 REPLACE *YES 屬性的 **CPYMQMQ** 指令，以兩種方式變更佇列屬性。 在 第 295 頁的『定義本端佇列』中，您已定義佇列 orange.local.queue。 例如，如果您需要將此佇列上的訊息長度上限增加到 10,000 個位元組。

- 使用 **CHGMQMQ** 指令:

  ```
  CHGMQMQ QNAME('orange.local.queue') MQMNAME(MYQUEUEMANAGER) MAXMSGLEN(10000)
  ```

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

  ```
  CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL) MQMNAME(MYQUEUEMANAGER)
  MAXMSGLEN(10000) REPLACE(*YES)
  ```

  此指令不僅會變更訊息長度上限，還會變更所有其他屬性，這些屬性會提供其預設值。 現在佇列已啟用放置，而先前已禁止放置。 啟用放置是由佇列 SYSTEM.DEFAULT.LOCAL.QUEUE 指定的預設值，除非您已變更它。

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

清除本端佇列

若要從本端佇列 magenta.queue 中刪除所有訊息，請使用下列指令:

```
CLRMQMQ QNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

在下列情況下，您無法清除佇列:

- 在同步點下，有未確定的訊息已放置於佇列上。
- 應用程式目前已開啟佇列。

刪除本端佇列

使用指令 **DLTMQMQ** 來刪除本端佇列。

如果佇列上有未確定的訊息，或佇列正在使用中，則無法刪除。

啟用大型佇列

IBM MQ 支援大於 2 GB 的佇列。 如需如何啟用 IBM i 以支援大型檔案的相關資訊，請參閱作業系統文件。

IBM i 產品資訊可在 IBM Documentation 中找到。

部分公用程式可能無法處理大於 2 GB 的檔案。 在啟用大型檔案支援之前，請先檢查作業系統說明文件，以取得這類支援的限制相關資訊。

IBM i 在 IBM i 上使用別名佇列

本節包含可用來管理別名佇列的部分指令範例。 也可以使用 **WRKMQMQ 指令畫面**中的選項來使用所有顯示的指令。

別名佇列 (有時稱為佇列別名) 提供重新導向 MQI 呼叫的方法。 別名佇列不是實際佇列，而是解析為實際佇列的定義。 別名佇列定義包含由 TGTQNAME 屬性指定的目標佇列名稱。

當應用程式在 MQI 呼叫中指定別名佇列時，佇列管理程式會在執行時期解析實際佇列名稱。

例如，已開發應用程式將訊息放置在稱為 `my.alias.queue` 的佇列上。 當此佇列提出 **MQOPEN** 要求時，它會指定此佇列的名稱; 如果它將訊息放置在此佇列上，則會間接指定此佇列的名稱。 應用程式不知道佇列是別名佇列。 對於每一個使用此別名的 MQI 呼叫，佇列管理程式會解析實際佇列名稱，該名稱可以是本端佇列，也可以是在此佇列管理程式中定義的遠端佇列。

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

定義別名佇列

下列指令會建立別名佇列:

```
CRTMQMQ QNAME('my.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
MQMNAME(MYQUEUEMANAGER)
```

此指令會將指定 `my.alias.queue` 的 MQI 呼叫重新導向至佇列 `yellow.queue`。 指令不會建立目標佇列; 如果在執行時期佇列 `yellow.queue` 不存在， MQI 呼叫會失敗。

如果您變更別名定義，則可以將 MQI 呼叫重新導向至另一個佇列。 例如:

```
CHGMQMQ QNAME('my.alias.queue') TGTQNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

此指令會將 MQI 呼叫重新導向至另一個佇列 `magenta.queue`。

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

- 應用程式 ALPHA 可以在 `yellow.queue` 上放置訊息，但不容許從它取得訊息。
- 應用程式測試版可以從 `yellow.queue` 取得訊息，但不容許在其中放置訊息。

您可以使用下列指令來執行此動作:

```
/* This alias is put enabled and get disabled for application ALPHA */

CRTMQMQ QNAME('alphas.alias.queue') QTYPE(*ALS)     TGTQNAME('yellow.queue')
PUTENBL(*YES) GETENBL(*NO) MQMNAME(MYQUEUEMANAGER)

/* This alias is put disabled and get enabled for application BETA */

CRTMQMQ QNAME('betas.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
PUTENBL(*NO) GETENBL(*YES) MQMNAME(MYQUEUEMANAGER)
```

A 會在其 MQI 呼叫中使用佇列名稱 alphas.alias.queue ;BETA 會使用佇列名稱 betas.alias.queue。 它們都存取相同的佇列，但使用不同的方式。

當您定義別名佇列時，可以使用 REPLACE *YES 屬性，就像您將這些屬性與本端佇列搭配使用一樣。

搭配使用其他指令與別名佇列

您可以使用適當的指令來顯示或變更別名佇列屬性。 例如：

```
* Display the alias queue's attributes */

DSPMQMQ QNAME('alphas.alias.queue') MQMNAME(MYQUEUEMANAGER)

/* ALTER the base queue name, to which the alias resolves. */
/* FORCE = Force the change even if the queue is open. */

CHQMQMQ QNAME('alphas.alias.queue') TGTQNAME('orange.local.queue') FORCE(*YES)
MQMNAME(MYQUEUEMANAGER)
```

IBM i 在 IBM i 上使用模型佇列

本節包含可用來管理模型佇列的部分指令範例。 也可以使用 **WRKMQMQ 指令畫面**中的選項來使用所有顯示的指令。

如果佇列管理程式從應用程式接收 MQI 呼叫，並指定已定義為模型佇列的佇列名稱，則會建立動態佇列。 建立佇列時，佇列管理程式會產生新動態佇列的名稱。 模型佇列是一個範本，指定從它建立的任何動態佇列的屬性。

模型佇列為應用程式提供方便的方法來建立佇列，因為它們是必要項目。

定義模型佇列

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

```
CRTMQMQ QNAME('green.model.queue') QTYPE(*MDL) DFNTYPE(*PERMDYN)
```

此指令會建立模型佇列定義。 從 DFNTYPE 屬性中，從這個範本建立的實際佇列是永久動態佇列。 未指定的屬性會自動從 SYSYTEM.DEFAULT.MODEL.QUEUE 預設佇列複製。

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

搭配使用其他指令與模型佇列

您可以使用適當的指令來顯示或變更模型佇列的屬性。 例如：

```
/* Display the model queue's attributes */

DSPMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('green.model.queue')

/* ALTER the model queue to enable puts on any */
/* dynamic queue created from this model. */

CHGMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('blue.model.queue') PUTENBL(*YES)
```

IBM i 在 IBM i 上使用觸發

使用此資訊來瞭解觸發及程序定義。

IBM MQ 提供在符合佇列上的特定條件時自動啟動應用程式的機能。 條件的一個範例是佇列上的訊息數達到指定的數目時。 此機能稱為 觸發 ，並在 觸發通道中詳細說明。

什麼是觸發?

佇列管理程式會將某些條件定義為構成觸發事件。 如果佇列已啟用觸發，且發生觸發事件，則佇列管理程式會將觸發訊息傳送至稱為起始佇列的佇列。 起始佇列上的觸發訊息存在指出已發生觸發事件。

佇列管理程式所產生的觸發訊息不會持續存在。 這會減少記載 (從而增進效能) ，並在重新啟動期間將重複項減至最少，從而縮短重新啟動時間。

何謂觸發監視器?

處理起始佇列的程式稱為觸發監視器應用程式，其功能是讀取觸發訊息，並根據觸發訊息中包含的資訊採取適當的動作。 通常此動作會啟動其他應用程式，以處理導致產生觸發訊息的佇列。 從佇列管理程式的觀點來看，觸發監視器應用程式沒有特殊的功能-它是從佇列 (起始佇列) 讀取訊息的另一個應用程式。

變更觸發監視器的工作提交屬性

以指令 **STRMQMTRM** 提供的觸發監視器會使用系統預設工作說明 QDFTJOBD 提交每一個觸發訊息的工作。 這有一些限制，因為提交的工作一律稱為 QDFTJOBD ，且具有預設工作說明的屬性，包括檔案庫清單 *SYSVAL。 IBM MQ 提供置換這些屬性的方法。 例如，可以自訂提交的工作，以具有更有意義的工作名稱，如下所示:

1. 在工作說明中指定您想要的說明，例如記載值。
2. 指定觸發程序中所使用程序定義的「環境資料」:

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOBD)')
```

「觸發監視器」會使用指定的說明來執行 SBMJOB 。

透過在程序定義的「環境資料」中指定適當的關鍵字和值，可以置換 SBMJOB 的其他屬性。 唯一例外是 CMD 關鍵字，因為此屬性由觸發監視器填入。 用於指定程序定義的「環境資料」，其中要變更工作名稱和說明的指令範例如下:

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOB)
JOB(TRIGGER)')
```

定義用於觸發的應用程式佇列

應用程式佇列是應用程式透過 MQI 進行傳訊所使用的本端佇列。 觸發需要在應用程式佇列上定義一些佇列屬性。 觸發本身由 TRGENBL 屬性啟用。

在此範例中，當本端佇列 motor.insurance.queue 上有 100 則優先順序為 5 或更高的訊息時，會產生觸發事件，如下所示:

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.insurance.queue') QTYPE(*LCL)
PRCNAME('motor.insurance.quote.process') MAXMSGLEN(2000)
DFTMSGPST(*YES) INITQNAME('motor.ins.init.queue')
TRGENBL(*YES) TRGTYPE(*DEPTH) TRGDEPTH(100) TRGMSGPTY(5)
```

其中參數為:

MQMNAME(MYQUEUEMANAGER)
佇列管理程式的名稱。

QNAME('motor.insurance.queue')
正在定義的應用程式佇列名稱。

PRCNAME('motor.insurance.quote.process')
要由觸發監視器程式啟動的應用程式名稱。

MAXMSGLEN(2000)
佇列上訊息的長度上限。

DFTMSGPST(*YES)
依預設，此佇列上的訊息會持續保存。

INITQNAME('motor.ins.init.queue')
佇列管理程式要放置觸發訊息的起始佇列名稱。

TRGENBL(*YES)
觸發屬性值。

TRGTYPE(*DEPTH)
當必要優先順序 (**TRGMSGPTY**) 的訊息數時，會產生觸發事件 達到 **TRGDEPTH** 中指定的數目。

TRGDEPTH(100)
產生觸發事件所需的訊息數。

TRGMSGPTY(5)
在決定是否產生觸發事件時，由佇列管理程式計算的訊息優先順序。 只會計算優先順序為 5 或以上的訊息。

定義起始佇列

當觸發事件發生時，佇列管理程式會將觸發訊息放置在應用程式佇列定義中指定的起始佇列上。 起始佇列沒有特殊設定，但您可以使用本端佇列 motor.ins.init.queue 的下列定義作為指引:

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.ins.init.queue') QTYPE(*LCL)
GETENBL(*YES) SHARE(*NO) TRGTYPE(*NONE)
MAXMSGL(2000)
MAXDEPTH(1000)
```

建立程序定義

使用 **CRTMQMPRC** 指令來建立程序定義。 程序定義會將應用程式佇列與要處理來自佇列之訊息的應用程式相關聯。 這是透過應用程式佇列 motor.insurance.queue 上的 PRCNAME 屬性來完成。 下列指令會建立此範例中所識別的必要處理程序 motor.insurance.quote.process:

```
CRTMQMPRC MQMNAME(MYQUEUEMANAGER) PRCNAME('motor.insurance.quote.process')
TEXT('Insurance request message processing')
APPTYPE(*OS400) APPID(MQTEST/TESTPROG)
USRDATA('open, close, 235')
```

其中參數為:

MQMNAME(MYQUEUEMANAGER)
佇列管理程式的名稱。

PRCNAME('motor.insurance.quote.process')
程序定義的名稱。

TEXT('Insurance request message processing')
與此定義相關的應用程式說明。 當您使用 **DSPMQMPRC** 指令時，會顯示此文字。 這可協助您識別處理程序執行的動作。 如果您在字串中使用空格，則必須以單引號括住字串。

APPTYPE(*OS400)
要啟動的應用程式類型。

APPID(MQTEST/TESTPROG)
應用程式執行檔的名稱，指定為完整檔名。

USRDATA('open, close, 235')
使用者定義資料，可供應用程式使用。

顯示程序定義

使用 **DSPMQMPRC** 指令來檢查定義的結果。 例如:

```
MQMNAME(MYQUEUEMANAGER) DSPMQMPRC('motor.insurance.quote.process')
```

您也可以使用 **CHGMQMPRC** 指令來變更現有的程序定義，並使用 **DLTMQMPRC** 指令來刪除程序定義。

IBM i IBM i 上兩個 IBM MQ 系統之間的通訊

此編碼範例說明如何使用 CL 指令來設定兩個 IBM MQ for IBM i 系統，以便它們可以彼此通訊。

系統稱為 SYSTEMA 和 SYSTEMB，所用的通訊協定是 TCP/IP。

執行下列程序:

1. 在 SYSTEMA 上建立佇列管理程式，並將它稱為 QMGRA1。

```
CRTMQM   MQMNAME(QMGRA1) TEXT('System A - Queue +
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

2. 啟動此佇列管理程式。

```
STRMQM   MQMNAME(QMGRA1)
```

3. 在 SYSTEMA 上定義您需要將訊息傳送至 SYSTEMB 上佇列管理程式的 IBM MQ 物件。

```
/* Transmission queue */
CRTMQMQ  QNAME(XMITQ.TO.QMGRB1) QTYPE(*LCL) +
MQMNAME(QMGRA1) TEXT('Transmission Queue +
to QMGRB1') MAXDEPTH(5000) USAGE(*TMQ)

/* Remote queue that points to a queue called TARGETB        */
/* TARGETB belongs to queue manager QMGRB1 on SYSTEMB        */
CRTMQMQ  QNAME(TARGETB.ON.QMGRB1) QTYPE(*RMT) +
MQMNAME(QMGRA1) TEXT('Remote Q pointing +
at Q TARGETB on QMGRB1 on Remote System +
SYSTEMB') RMTQNAME(TARGETB) +
RMTMQMNAME(QMGRB1) TMQNAME(XMITQ.TO.QMGRB1)

/* TCP/IP sender channel to send messages to the queue manager on SYSTEMB*/
CRTMQMCHL CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*SDR) +
MQMNAME(QMGRA1) TRPTYPE(*TCP) +
TEXT('Sender Channel From QMGRA1 on +
SYSTEMA to QMGRB1 on SYSTEMB') +
CONNAME(SYSTEMB) TMQNAME(XMITQ.TO.QMGRB1)
```

4. 在 SYSTEMB 上建立佇列管理程式，並將它稱為 QMGRB1。

```
CRTMQM   MQMNAME(QMGRB1) TEXT('System B - Queue +
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

5. 在 SYSTEMB 上啟動佇列管理程式。

```
STRMQM   MQMNAME(QMGRB1)
```

6. 在 SYSTEMA 上定義從佇列管理程式接收訊息所需的 IBM MQ 物件。

```
/* Local queue to receive messages on */
CRTMQMQ  QNAME(TARGETB) QTYPE(*LCL) MQMNAME(QMGRB1) +
TEXT('Sample Local Queue for QMGRB1')

/* Receiver channel of the same name as the sender channel on SYSTEMA */
CRTMQMCHL CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*RCVR) +
MQMNAME(QMGRB1) TRPTYPE(*TCP) +
TEXT('Receiver Channel from QMGRA1 to +
QMGRB1')
```

7. 最後，在 SYSTEMB 上啟動 TCP/IP 接聽器，以便可以啟動通道。 此範例使用預設埠 1414。

```
STRMQMLSR MQMNAME(QMGRB1)
```

現在，您已準備好在 SYSTEMA 與 SYSTEMB 之間傳送測試訊息。 使用其中一個提供的範例，將一系列訊息放置到 SYSTEMA 上的遠端佇列。

在 SYSTEMA 上啟動通道，方法是使用指令 **STRMQMCHL**，或使用指令 **WRKMQMCHL** 並針對傳送端通道輸入啟動要求 (選項 14)。

通道應該進入 RUNNING 狀態，且訊息會傳送至 SYSTEMB 上的 TARGETB 佇列。

發出下列指令來檢查您的訊息:

```
WRKMQMMSG QNAME(TARGETB) MQMNAME(QMGRB1).
```

IBM i IBM i 上的資源定義範例

此範例包含 AMQSAMP4 範例 IBM i CL 程式。

```
/*********************************************************************/
/*                                                                   */
/* Program name: AMQSAMP4                                            */
/*                                                                   */
/* Description: Sample CL program defining MQM queues                */
/*       to use with the sample programs                             */
/*       Can be run, with changes as needed, after                   */
/*       starting the MQM                                            */
/*                                                                   */
/* <N_OCO_COPYRIGHT>                                              */
/* Licensed Materials - Property of IBM                              */
/*                                                                   */
/* 63H9336                                                           */
/* (c) Copyright IBM Corp. 1993, 2025. All Rights Reserved.          */
/*                                                                   */
/* US Government Users Restricted Rights - Use, duplication or       */
/* disclosure restricted by GSA ADP Schedule Contract with           */
/* IBM Corp.                                                         */
/* <NOC_COPYRIGHT>                                                */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/* Function:                                                         */
/*                                                                   */
/*                                                                   */
/*  AMQSAMP4 is a sample CL program to create or reset the           */
/*  MQI resources to use with the sample programs.                   */
/*                                                                   */
/*  This program, or a similar one, can be run when the MQM          */
/*  is started - it creates the objects if missing, or resets        */
/*  their attributes to the prescribed values.                       */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*  Exceptions signaled: none                                        */
/*  Exceptions monitored: none                                       */
/*                                                                   */
/*  AMQSAMP4 takes a single parameter, the Queue Manager name        */
/*                                                                   */
/*********************************************************************/
QSYS/PGM PARM(&QMGRNAME)

/*********************************************************************/
/*  Queue Manager Name Parameter                                     */
/*********************************************************************/
QSYS/DCL VAR(&QMGRNAME) TYPE(*CHAR)

/*********************************************************************/
/*    EXAMPLES OF DIFFERENT QUEUE TYPES                              */
/*                                                                   */
/*  Create local, alias and remote queues                            */
/*                                                                   */
/*  Uses system defaults for most attributes                         */
/*                                                                   */
/*********************************************************************/
/*  Create a local queue   */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.LOCAL')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Sample local queue') /* description */+
SHARE(*YES)         /* Shareable  */+
DFTMSGPST(*YES) /* Persistent messages OK */

/*  Create an alias queue  */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.ALIAS')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*ALS) REPLACE(*YES)         +
+
TEXT('Sample alias queue')         +
DFTMSGPST(*YES) /* Persistent messages OK */+
```

```
TGTQNAME('SYSTEM.SAMPLE.LOCAL')

/*  Create a remote queue - in this case, an indirect reference  */
/*   is made to the sample local queue on OTHER queue manager  */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.REMOTE')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*RMT) REPLACE(*YES)         +
+
TEXT('Sample remote queue')/* description */+
DFTMSGPST(*YES) /* Persistent messages OK */+
RMTQNAME('SYSTEM.SAMPLE.LOCAL')       +
RMTMQMNAME(OTHER)  /* Queue is on OTHER  */

/*  Create a transmission queue for messages to queues at OTHER  */
/*  By default, use remote node name                */
CRTMQMQ   QNAME('OTHER') /* transmission queue name */+
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES) +
TEXT('Transmision queue to OTHER') +
USAGE(*TMQ)  /* transmission queue   */

/*********************************************************************/
/*    SPECIFIC QUEUES AND PROCESS USED BY SAMPLE PROGRAMS           */
/*                                                                   */
/*  Create local queues used by sample programs                      */
/*  Create MQI process associated with sample initiation queue       */
/*                                                                   */
/*********************************************************************/
/*  General reply queue  */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.REPLY')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('General reply queue')         +
DFTMSGPST(*NO) /* Not Persistent     */

/*  Queue used by AMQSINQ4                     */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.INQ')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSINQ4')          +
SHARE(*YES)        /* Shareable  */+
DFTMSGPST(*NO) /* Not Persistent     */+
+
TRGENBL(*YES) /* Trigger control on   */+
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.INQPROCESS')     +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Queue used by AMQSSET4                     */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.SET')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSSET4')          +
SHARE(*YES)        /* Shareable */ +
DFTMSGPST(*NO)/* Not Persistent     */ +
+
TRGENBL(*YES) /* Trigger control on   */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.SETPROCESS')     +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Queue used by AMQSECH4                     */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.ECHO')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSECH4')          +
SHARE(*YES)        /* Shareable */ +
DFTMSGPST(*NO)/* Not Persistent     */ +
+
TRGENBL(*YES) /* Trigger control on   */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')    +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Initiation Queue used by AMQSTRG4, sample trigger process   */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.TRIGGER') +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)  +
```

```
TEXT('Trigger queue for sample programs')

/*  MQI Processes associated with triggered sample programs    */
/*                                 */
/*****  Note - there are versions of the triggered samples ******/
/*****   in different languages - set APPID for these   ******/
/*****   process to the variation you want to trigger   ******/
/*                                 */
CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.INQPROCESS')     +
MQMNAME(&QMGRNAME)             +
REPLACE(*YES)                +
+
TEXT('Trigger process for AMQSINQ4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSINQ4')  /* C          +
/*    APPID('QMQM/AMQ0INQ4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3INQ4')  /* RPG - ILE */

CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.SETPROCESS')     +
MQMNAME(&QMGRNAME)             +
REPLACE(*YES)                +
+
TEXT('Trigger process for AMQSSET4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSSET4')  /* C   */    +
/*    APPID('QMQM/AMQ0SET4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3SET4')  /* RPG - ILE */

CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')    +
MQMNAME(&QMGRNAME)             +
REPLACE(*YES)                +
+
TEXT('Trigger process for AMQSECH4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSECH4')  /* C   */    +
/*    APPID('QMQM/AMQ0ECH4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3ECH4')  /* RPG - ILE */

/*********************************************************************/
/*                                                                   */
/*  Normal return.                                                   */
/*                                                                   */
/*********************************************************************/
SNDPGMMSG  MSG('AMQSAMP4 Completed creating sample +
objects for ' *CAT &QMGRNAME)
RETURN
ENDPGM

/*********************************************************************/
/*                                                                   */
/* END OF AMQSAMP4                                                   */
/*                                                                   */
/*********************************************************************/
```

IBM i 管理 IBM MQ for IBM i 的替代方式

使用 CL 指令是管理 IBM MQ for IBM i 的偏好方法。 不過，您可以使用各種其他管理方法，包括 MQSC 指令、PCF 指令及遠端管理。

您通常使用 IBM i CL 指令來管理 IBM MQ for IBM i。 如需這些指令的概觀，請參閱 第 292 頁的『使用 CL 指令管理 IBM MQ for IBM i』。

您可以使用 IBM MQ 檢測事件來監視佇列管理程式的作業。 如需 IBM MQ 設備測試事件以及如何使用它們的相關資訊，請參閱 設備測試事件 。

您可以使用下列子主題中說明的任何管理方法，作為使用 IBM i CL 指令的替代方案:

IBM i IBM i 上的本端及遠端管理

您可以在本端或遠端管理 IBM MQ for IBM i 物件。

本端管理 表示對您在本端系統上定義的任何佇列管理程式執行管理作業。 在 IBM MQ 中，您可以將此視為本端管理，因為不涉及任何 IBM MQ 通道，即通訊由作業系統管理。 若要執行此類型的作業，您必須登入遠端系統並從該處發出指令，或建立可以為您發出指令的處理程序。

IBM MQ 支援透過所謂 遠端管理從單一點進行管理。 遠端管理包括將可程式化指令格式 (PCF) 控制訊息傳送至目標佇列管理程式上的 SYSTEM.ADMIN.COMMAND.QUEUE 。

有多種產生 PCF 訊息的方法。 它們是:

1. 使用 PCF 訊息寫入程式。 請參閱 第 306 頁的『在 IBM i 上使用 PCF 指令進行管理』。
2. 使用 MQAI 撰寫程式，以送出 PCF 訊息。 請參閱 第 31 頁的『使用 MQAI 來簡化 PCF 的使用』。
3. 使用 IBM MQ for Windows 隨附的「 IBM MQ 檔案總管」，可讓您使用圖形使用者介面 (GUI) 並產生正確的 PCF 訊息。 請參閱第 307 頁的『將 IBM MQ Explorer 與 IBM MQ for IBM i 搭配使用』。
4. 使用 **STRMQMMQSC** 可間接將指令傳送至遠端佇列管理程式。 請參閱 第 305 頁的『在 IBM i 上使用 MQSC 指令進行管理』。

例如，您可以發出遠端指令來變更遠端佇列管理程式上的佇列定義。

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

IBM i 在 IBM i 上使用 MQSC 指令進行管理

使用此資訊來瞭解 MQSC 指令，以及如何使用它們來管理 IBM MQ for IBM i。

IBM MQ Script (MQSC) 指令以人類可讀的格式 (即 EBCDIC 文字) 撰寫。 您可以使用 MQSC 指令來管理佇列管理程式物件，包括佇列管理程式本身、佇列、程序定義、名稱清單、通道、用戶端連線通道、接聽器、服務、主題及鑑別資訊物件。

您可以使用 **STRMQMMQSC** IBM MQ CL 指令，對佇列管理程式發出 MQSC 指令。 此方法僅是批次方法，從伺服器檔案庫系統中的來源實體檔取得其輸入。 此來源實體檔的預設名稱為 QMQSC。

小心: 請勿使用 QTEMP 程式庫作為 STRMQMQSC 的來源程式庫，因為 QTEMP 程式庫的使用受到限制。 您必須使用另一個檔案庫作為指令的輸入檔。

IBM MQ for IBM i 未提供稱為 QMQSC 的原始檔。 若要處理 MQSC 指令，您必須發出下列指令，在您選擇的程式庫中建立 QMQSC 原始檔:

```
CRTSRCPF FILE(MYLIB/QMQSC) RCDLEN(240) TEXT('IBM MQ - MQSC Source')
```

MQSC 來源保留在此原始檔內的成員中。 若要使用成員，請輸入下列指令:

```
WRKMBRPDM MYLIB/QMQSC
```

您現在可以新增成員並維護現有成員

您也可以透過發出 RUNMQSC 或下列指令，以互動方式輸入 MQSC 指令:

1. 鍵入佇列管理程式名稱，並按 Enter 鍵以存取 **WRKMQM** 結果畫面。
2. 在此畫面上選取 F23=More options 。
3. 在「第 306 頁的圖 23」顯示的畫面上，針對作用中佇列管理程式選取選項 26。

若要結束這類 MQSC 階段作業，請鍵入 end。

第 306 頁的圖 23 是 MQSC 指令檔的擷取，顯示 MQSC 指令 (DEFINE QLOCAL) 及其屬性。

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

圖 *23:* 從 *MQSC* 指令檔 *myprog.in* 解壓縮

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

MQSC 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。

註:

1. MQSC 檔案的格式不取決於其在檔案系統中的位置。
2. MQSC 屬性名稱限制為 8 個字元。
3. MQSC 指令可在其他平台上使用，包括 z/OS。

如需每一個 MQSC 指令及其語法的說明，請參閱 MQSC 指令。

IBM i 在 IBM i 上使用 PCF 指令進行管理

IBM MQ 可程式指令格式 (PCF) 指令的目的是容許將管理作業程式設計到管理程式中。 如此一來，您可以從程式建立佇列和程序定義，以及變更佇列管理程式。

PCF 指令涵蓋 MQSC 指令所提供的相同函數範圍。 不過，與 MQSC 指令不同， PCF 指令及其回覆不是您可以讀取的文字格式。

您可以撰寫程式，從單一節點向網路中的任何佇列管理程式發出 PCF 指令。 以此方式，您可以將管理作業集中化及自動化。

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

簡言之，這些是應用程式設計師必須指定以建立 PCF 指令訊息的部分內容:

訊息描述子
這是標準 IBM MQ 訊息描述子，其中:

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

應用程式資料
包含 PCF 訊息 (包括 PCF 標頭) ，其中:

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

Escape PCF 是在訊息文字內包含 MQSC 指令的 PCF 指令。 您可以使用 PCF 將指令傳送至遠端佇列管理程式。 如需進一步資訊，請參閱第 31 頁的『使用 MQAI 來簡化 PCF 的使用』。

如需 PCF 資料結構及其實作方式的完整說明，請參閱 指令及回應的結構。

IBM i 將 IBM MQ Explorer 與 IBM MQ for IBM i 搭配使用

使用此資訊可使用 IBM MQ Explorer 來管理 IBM MQ for IBM i 。

IBM MQ for Windows (x86 平台) 及 IBM MQ for Linux (x86 及 x86-64 平台) 提供稱為「IBM MQ 探險家」的管理介面，以執行管理作業作為使用 CL、控制或 MQSC 指令的替代方案。

IBM MQ Explorer 可讓您透過將 IBM MQ Explorer 指向您感興趣的佇列管理程式及叢集，從執行 Windows (x86 平台) 或 Linux (x86 及 x86-64 平台) 的電腦執行網路的本端或遠端管理。

使用 IBM MQ Explorer，您可以:

- 啟動和停止佇列管理程式 (僅在本端機器上)。
- 定義、顯示及變更 IBM MQ 物件 (例如佇列、主題及通道) 的定義。
- 瀏覽佇列上的訊息。
- 啟動和停止通道。
- 檢視通道的狀態資訊。
- 檢視叢集中的佇列管理程式。
- 請檢查以查看哪些應用程式、使用者或通道已開啟特定佇列。
- 使用「**建立新的叢集**」精靈來建立新的佇列管理程式叢集。
- 使用「**將佇列管理程式新增至叢集**」精靈，將佇列管理程式新增至叢集。
- 管理與傳輸層安全 (TLS) 通道安全搭配使用的鑑別資訊物件。

使用線上指引，您可以:

- 定義及控制各種資源，包括佇列管理程式、佇列、通道、程序定義、用戶端連線通道、接聽器、主題、服務、名稱清單及叢集。
- 啟動或停止佇列管理程式及其相關聯的處理程序。
- 檢視工作站上或其他工作站中的佇列管理程式及其相關聯物件。
- 檢查佇列管理程式、叢集及通道的狀態。

在嘗試使用 IBM MQ Explorer 來管理伺服器機器上的 IBM MQ 之前，請確定您已滿足下列需求。 請檢查:

1. 針對由 CL 指令 **STRMQMCSVR** 在伺服器上啟動的任何受管理佇列管理程式，正在執行指令伺服器。
2. 每個遠端佇列管理程式都有適當的 TCP/IP 接聽器。 這是由 **STRMQMLSR** 指令啟動的 IBM MQ 接聽器。
3. 伺服器連線通道 (稱為 SYSTEM.ADMIN.SVRCONN) 存在於每一個遠端佇列管理程式上。 您必須自行建立此通道。 對於每一個受管理的遠端佇列管理程式，這是必要的。 沒有它，無法進行遠端管理。
4. 驗證 SYSTEM.MQEXPLORER.REPLY.MODEL 佇列是否存在。

IBM i 在 IBM i 上管理用於遠端管理的指令伺服器

使用此資訊來瞭解 IBM MQ for IBM i 指令伺服器的遠端管理。

每一個佇列管理程式都可以有相關聯的指令伺服器。 指令伺服器會處理來自遠端佇列管理程式的任何送入指令，或來自應用程式的 PCF 指令。 它會將指令呈現給佇列管理程式進行處理，並根據指令的原點傳回完成碼或操作員訊息。

對於涉及 PCF 的所有管理、MQAI 以及遠端管理而言，指令伺服器是必要的。

註: 對於遠端管理，您必須確定目標佇列管理程式正在執行中。 否則，包含指令的訊息無法離開從中發出指令的佇列管理程式。 相反地，這些訊息會在提供遠端佇列管理程式的本端傳輸佇列中排入佇列。 如果可能的話，請避免此狀況。

有個別控制指令可用來啟動及停止指令伺服器。 您可以使用「IBM MQ 檔案總管」來執行下列各節中說明的作業。

啟動和停止指令伺服器

若要啟動指令伺服器，請使用下列 CL 指令:

```
STRMQMCSVR MQMNAME('saturn.queue.manager')
```

其中 saturn.queue.manager 是正在啟動指令伺服器的佇列管理程式。

若要停止指令伺服器，請使用下列其中一個 CL 指令:

1. ```
   ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*CNTRLD)
   ```

   以執行受管制的停止，其中 saturn.queue.manager 是正在停止指令伺服器的佇列管理程式。 這是預設選項，表示可以省略 OPTION(*CNTRLD)。
2. ```
   ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*IMMED)
   ```

   執行立即停止，其中 saturn.queue.manager 是要停止指令伺服器的佇列管理程式。

顯示指令伺服器的狀態

對於遠端管理，請確定目標佇列管理程式上的指令伺服器正在執行中。 如果它不在執行中，則無法處理遠端指令。 任何包含指令的訊息都會排入目標佇列管理程式的指令佇列 SYSTEM.ADMIN.COMMAND.QUEUE 中。

若要顯示佇列管理程式的指令伺服器狀態 (在這裡稱為 saturn.queue.manager)， CL 指令為:

```
DSPMQMCSVR MQMNAME('saturn.queue.manager')
```

在目標機器上發出此指令。 如果指令伺服器正在執行中，則會出現 第 308 頁的圖 24 中顯示的畫面:

```
Display MQM Command Server (DSPMQMCSVR)

Queue manager name . . . . . > saturn.queue.manager

MQM Command Server Status. . . . > RUNNING




F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

圖 *24:* 顯示 *MQM* 指令伺服器畫面

IBM i 執行 Web 主控台指令

您必須依照下列文字中的說明來配置環境， Web 主控台相關的 Qshell 指令才能在 IBM MQ for IBM i 上正確執行。

關於這項作業

當 Qshell 啟動時，它會根據工作的 CCSID 來起始設定處理指令的內部表格。 若要讓 Web 主控台相關的 Qshell 指令在 IBM i 上正確執行，您必須配置環境。

將 LANG 環境變數設為語言環境物件的路徑名稱，即可設定語言環境。 例如，若要設定美式英文的語言環境， LANG 環境變數設定如下:

```
LANG=/QSYS.LIB/EN_US.LOCALE
```

在 Qshell 中，您可以透過發出指令集來檢查設定，以列出所有環境變數。 通常是 LANG 可能會影響執行時期環境的語言環境。 它也可能有 LC_ALL。

若要正確執行 Qshell 指令，語言環境設定必須與您的工作設定一致。

程序

使用 CL 指令 DSPJOB JOB (JobNumber/USERProfile/JobName)

a) 選取選項 2 以顯示工作定義屬性。

b) 下列屬性應該與 LANG 或 LC_ALL 環境設定一致

- 語言 ID
- 國家或地區 ID
- 編碼字集 ID

例如，如果

```
LANG=/QSYS.LIB/FR_FR.LOCALE
```

您的工作屬性應該為:

- 語言 ID..... 法國
- 國家或地區 ID..... FR
- 編碼字集 ID..... 297

下一步

如需國家語言支援的相關資訊，請參閱 IBM Documentation 主題 國家語言支援 (NLS) 考量。

IBM i IBM i 的工作管理

此資訊說明 IBM MQ 處理工作要求的方式，並詳細說明可用於設定優先順序及控制與 IBM MQ 相關聯之工作的選項。

警告

除非您完全瞭解 IBM i 和 IBM MQ 工作管理的概念，否則請勿變更 IBM MQ 工作管理物件。

如需子系統及工作說明的相關資訊，請參閱 IBM i 產品說明文件中的 工作管理 。 請特別注意 啟動工作 及 批次工作上的小節。

IBM MQ for IBM i 納入 IBM i UNIX 環境及 IBM i 執行緒。 請 **不要** 對 Integrated File System (IFS) 中的物件進行任何變更。

在正常作業期間， IBM MQ 佇列管理程式會啟動一些批次工作來執行不同的作業。 依預設，這些批次工作在安裝 IBM MQ 時所建立的 QMQM 子系統中執行。

工作管理是指自訂 IBM MQ 作業以從系統取得最佳效能，或簡化管理的程序。

例如，您可以:

- 變更工作的執行優先順序，使一個佇列管理程式比另一個佇列管理程式更能回應。
- 將一些工作的輸出重新導向至特定輸出佇列。
- 讓特定類型的所有工作在特定子系統中執行。
- 將錯誤隔離至子系統。

工作管理是透過建立或變更與 IBM MQ 工作相關聯的工作說明來執行。 您可以配置下列項目的工作管理:

- 整個 IBM MQ 安裝。
- 個別佇列管理程式。

- 個別佇列管理程式的個別工作。

IBM i IBM i 的 IBM MQ 作業

這是 IBM MQ for IBM i 工作的表格，以及各工作的簡要說明。

當佇列管理程式正在執行時，您會看到下列部分或所有批次工作在 IBM MQ 子系統中的 QMQM 使用者設定檔下執行。 這些工作在 第 310 頁的表 21 中有簡要說明。

您可以使用「**使用佇列管理程式** (WRKMQM)」畫面上的選項 22，檢視連接至佇列管理程式的所有工作。 您可以使用 WRKMQMLSR 指令來檢視接聽器。

表 *21: IBM MQ* 作業。

工作名稱	函數
AMQZMUC0	公用程式管理程式。 此工作執行重要佇列管理程式公用程式，例如日誌登載鏈管理程式。
AMQZXMA0	執行控制器，它是佇列管理程式所啟動的第一個工作。 它會處理 MQCONN 要求，並啟動代理程式處理程序來處理 IBM MQ API 呼叫。
AMQZFUMA	物件權限管理程式 (OAM)。
AMQZLAA0	針對使用 MQCNO_STANDARD_BINDING 連接至佇列管理程式的應用程式執行大部分工作的佇列管理程式代理程式。
AMQZLSA0	佇列管理程式代理程式。
AMQZMUF0	公用程式管理程式
AMQZMGR0	處理程序控制器。 此工作用於啟動及管理接聽器和服務。
AMQZMUR0	公用程式管理程式。 此工作執行重要佇列管理程式公用程式，例如日誌登載鏈管理程式。
AMQFQPUB	已將發佈/訂閱常駐程式排入佇列。
AMQFCXBA	分配管理系統工作者工作。
RUNMQBRK	分配管理系統控制工作。
AMQRMPPA	通道處理程序儲存區工作。
AMQCRSTA	TCP/IP 呼叫的通道回應者。
AMQCRS6B	LU62 接收端通道及用戶端連線 (請參閱附註)。
AMQRRMFA	叢集的儲存庫管理程式。
AMQCLMAA	非執行緒 TCP/IP 接聽器。
AMQPCSEA	處理 PCF 及遠端管理要求的 PCF 指令處理器。
RUNMQTRM	觸發監視器。
RUNMQDLQ	無法傳送郵件的佇列處理程式。
RUNMQCHI	通道起始程式。
RUNMQCHL	針對每一個傳送端通道啟動的傳送端通道工作。
RUNMQLSR	執行緒 TCP/IP 接聽器。
AMQRCMLA	通道 MQSC 及 PCF 指令處理器。

註: LU62 接收端工作在通訊子系統中執行，並從用來啟動工作的遞送及通訊項目中取得其執行時期內容。 如需相關資訊，請參閱 起始結束 (接收端) 。

IBM i IBM i 上的工作管理物件

安裝 IBM MQ 時，會在 QMQM 程式庫中提供各種物件，以協助進行工作管理。 這些物件是 IBM MQ 工作在其自己的子系統中執行所需的物件。

提供兩個 IBM MQ 批次工作的範例工作說明。 如果未提供 IBM MQ 工作的特定工作說明，則它會以預設工作說明 QMQMJOBD 執行。

當您安裝 IBM MQ 時所提供的工作管理物件會列在 第 311 頁的表 22 中，而針對佇列管理程式所建立的物件會列在 第 311 頁的表 23 中。

註: 工作管理物件可以在 QMQM 程式庫中找到，佇列管理程式物件可以在佇列管理程式程式庫中找到。

表 *22:* 工作管理物件

姓名	類型	說明
AMQZLAA0	*JOBD	IBM MQ 代理程式處理程序所使用的工作說明
AMQZLSA0	*JOBD	隔離的連結佇列管理程式代理程式
AMQZXMA0	*JOBD	IBM MQ 執行控制器使用的工作說明
QMQM	*SBSD	在其中執行所有 IBM MQ 工作的子系統
QMQM	*JOBQ	連接至所提供子系統的工作佇列
QMQMJOBD	*JOBD	預設 IBM MQ 工作說明，在沒有工作的特定工作說明時使用
QMQMMSG	*MSGQ	IBM MQ 工作的預設訊息佇列。
QMQMRUN20	*CLS	高優先順序 IBM MQ 工作的類別說明
QMQMRUN35	*CLS	中優先順序 IBM MQ 工作的類別說明
QMQMRUN50	*CLS	低優先順序 IBM MQ 工作的類別說明

表 *23:* 為佇列管理程式建立的工作管理物件

姓名	類型	說明
AMQA000000	*JRNRCV	本端異動日誌接收器
AMQAJRN	*JRN	本端異動日誌
AMQJRNINF	*USRSPC	以佇列管理程式啟動及媒體回復所需的最新異動日誌接收器更新的使用者空間。 應用程式可以查詢此使用者空間，以判斷哪些異動日誌接收器需要保存，哪些可以安全刪除。
AMQAJRNMSG	*MSGQ	本端日誌登載訊息佇列
AMQCRC6B	*PGM	啟動 LU6.2 連線的程式
AMQRFOLD	*FILE	已移轉佇列管理程式通道定義檔
QMQMMSG	*MSGQ	佇列管理程式訊息佇列

IBM i IBM MQ 如何在 IBM i 上使用工作管理物件

此資訊說明 IBM MQ 使用工作管理物件的方式，並提供配置範例。

小心: 請勿變更 QMQM 子系統中的工作佇列登錄設定，以依優先順序限制子系統中容許的工作數目。 如果您嘗試這樣做，則可以在提交必要的 IBM MQ 工作之後停止它們執行，並導致佇列管理程式啟動失敗。

若要瞭解如何配置工作管理，您必須先瞭解 IBM MQ 如何使用工作說明。

用來啟動工作的工作說明控制工作的許多屬性。 例如:

- 將工作排入佇列的工作佇列，以及工作執行所在的子系統。
- 用來啟動工作的遞送資料，以及工作用於其執行時期參數的類別。
- 工作用於列印檔案的輸出佇列。

啟動 IBM MQ 工作的程序可以分為三個步驟:

1. IBM MQ 選取工作說明。

   IBM MQ 使用下列技術來判定要用於批次工作的工作說明:

   a. 請在佇列管理程式檔案庫中尋找與工作同名的工作說明。 如需佇列管理程式庫的進一步詳細資料，請參閱 瞭解 IBM MQ for IBM i 佇列管理程式庫名稱 。

   b. 在佇列管理程式檔案庫中尋找預設工作說明 QMQMJOBD。

   c. 請在 QMQM 程式庫中尋找與工作同名的工作說明。

   d. 使用 QMQM 程式庫中的預設工作說明 QMQMJOBD。

2. 工作已提交至工作佇列。

   依預設，已設定 IBM MQ 提供的工作說明，將工作放入檔案庫 QMQM 中的工作佇列 QMQM。 QMQM 工作佇列會連接至所提供的 QMQM 子系統，因此依預設，工作會在 QMQM 子系統中開始執行。

3. 工作進入子系統並完成遞送步驟。

   當工作進入子系統時，會使用工作說明上指定的遞送資料來尋找工作的遞送登錄。

   遞送資料必須符合 QMQM 子系統中定義的其中一個遞送項目，且這會定義工作使用哪些提供的類別 (QMQMRUN20、 QMQMRUN35 或 QMQMRUN50)。

註: 如果 IBM MQ 工作似乎未啟動，請確定子系統正在執行中，且未保留工作佇列。

如果您已修改 IBM MQ 工作管理物件，請確定所有項目都有正確關聯。 例如，如果您在工作說明上指定 QMQM/QMQM 以外的工作佇列，請確定已針對子系統 (即 QMQM) 執行 ADDJOBQE。

您可以使用下列工作表作為範例，為 第 310 頁的表 21 中所記載的每一個工作建立工作說明:

```
What is the queue manager library name? ___________
Does job description AMQZXMA0 exist in the queue manager library? Yes   No
Does job description QMQMJOBD exist in the queue manager library? Yes   No
Does job description AMQZXMA0 exist in the QMQM library?     Yes   No
Does job description QMQMJOBD exist in the QMQM library?     Yes   No
```

如果您對所有這些問題都回答「否」，請在 QMQM 程式庫中建立廣域工作說明 QMQMJOBD。

IBM MQ 訊息佇列

在每一個佇列管理程式檔案庫中建立 IBM MQ 訊息佇列 QMQMMSG。 當佇列管理程式工作結束，且 IBM MQ 將訊息傳送至佇列時，作業系統訊息會傳送至此佇列。 例如，報告啟動時需要哪些異動日誌接收器。 請將此訊息佇列中的訊息數保持在可管理的大小，以便更容易監視。

IBM i IBM i 的預設系統範例

這些範例顯示在佇列管理程式啟動時提交部分標準工作時，未修改的 IBM MQ 安裝如何運作。

首先，會啟動 AMQZXMA0 執行控制器工作。

1. 針對佇列管理程式 TESTQM 發出 **STRMQM** 指令。
2. IBM MQ 會先搜尋佇列管理程式庫 QMTESTQM ，找出工作說明 AMQZXMA0，然後再搜尋工作說明 QMQMJOBD。

   這兩個工作說明都不存在，因此 IBM MQ 會在產品程式庫 QMQM 中尋找工作說明 AMQZXMA0 。 此工作說明已存在，因此會使用它來提交工作。

3. 工作說明使用 IBM MQ 預設工作佇列，因此會將工作提交至工作佇列 QMQM/QMQM。
4. AMQZXMA0 工作說明上的遞送資料是 QMQMRUN20，因此系統會在子系統遞送項目中搜尋符合該資料的項目。

依預設，序號為 9900 的遞送登錄具有符合 QMQMRUN20 的比較資料，因此會以該遞送登錄上定義的類別 (也稱為 QMQMRUN20) 來啟動工作。

5. QMQM/QMQMRUN20 類別已將執行優先順序設為 20 ，因此 AMQZXMA0 工作以與系統上大部分互動式工作相同的優先順序在子系統 QMQM 中執行。

IBM i 在 IBM i 上配置工作管理範例

使用此資訊來瞭解如何變更及建立 IBM MQ 工作說明，以變更 IBM MQ 工作的執行時期屬性。

IBM MQ 工作管理彈性的關鍵在於 IBM MQ 搜尋工作說明的兩層方式:

- 如果您在佇列管理程式檔案庫中建立或變更工作說明，則那些變更會置換 QMQM 中的廣域工作說明，但這些變更是本端變更，且只會影響該特定佇列管理程式。
- 如果您在 QMQM 程式庫中建立或變更廣域工作說明，除非針對個別佇列管理程式在本端置換，否則這些工作說明會影響系統上的所有佇列管理程式。

1. 下列範例會增加個別佇列管理程式之通道控制工作的優先順序。

   若要讓儲存庫管理程式及通道起始程式工作 AMQRRMFA 及 RUNMQCHI 儘快針對佇列管理程式 TESTQM 執行，請執行下列步驟:

   a. 使用您要在佇列管理程式檔案庫中控制的 IBM MQ 處理程序名稱，建立 QMQM/QMQMJOBD 工作說明的本端重複項。 例如:

   ```
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
   NEWOBJ(RUNMQCHI)
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
   NEWOBJ(AMQRRMFA)
   ```

   b. 變更工作說明上的遞送資料參數，以確保工作使用 QMQMRUN20 類別。

   ```
   CHGJOBD JOBD(QMTESTQM/RUNMQCHI) RTGDTA('QMQMRUN20')
   CHGJOBD JOBD(QMTESTQM/AMQRRMFA) RTGDTA('QMQMRUN20')
   ```

   現在佇列管理程式 TESTQM 的 AMQRRMFA 及 RUNMQCHI 工作:

   - 使用佇列管理程式檔案庫中新的本端工作說明
   - 以優先順序 20 執行，因為當工作進入子系統時，會使用 QMQMRUN20 類別。

2. 下列範例定義 QMQM 子系統的新執行優先順序類別。

   a. 透過發出下列指令，在 QMQM 程式庫中建立重複的類別，以容許其他佇列管理程式存取該類別:

   ```
   CRTDUPOBJ OBJ(QMQMRUN20) FROMLIB(QMQM) OBJTYPE(*CLS) TOLIB(QMQM)
   NEWOBJ(QMQMRUN10)
   ```

   b. 發出下列指令，將類別變更為具有新的執行優先順序:

   ```
   CHGCLS CLS(QMQM/QMQMRUN10) RUNPTY(10)
   ```

   c. 發出下列指令，將新的類別定義新增至子系統:

   ```
   ADDRTGE SBSD(QMQM/QMQM) SEQNBR(8999) CMPVAL('QMQMRUN10') PGM(QSYS/QCMD)
   CLS(QMQM/QMQMRUN10)
   ```

   註: 您可以為遞送序號指定任何數值，但這些值必須依序排列。 此序號告訴子系統要在遞送登錄中搜尋遞送資料相符項的順序。

   d. 發出下列指令，將本端或廣域工作說明變更為使用新的優先順序類別:

   ```
   CHGJOBD JOBD(QMQMlibname/QMQMJOBD) RTGDTA('QMQMRUN10')
   ```

   現在與 QMlibraryname 相關聯的所有佇列管理程式工作都使用執行優先順序 10。

3. 下列範例會在其自己的子系統中執行佇列管理程式

   若要讓佇列管理程式 TESTQM 的所有工作在 QBATCH 子系統中執行，請執行下列步驟:

a. 使用指令在佇列管理程式檔案庫中建立 QMQM/QMQMJOBD 工作說明的本端副本

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
```

b. 變更工作說明上的工作佇列參數，以確定工作使用 QBATCH 工作佇列。

```
CHGJOBD JOBD(QMTESTQM/QMQMJOBD) JOBQ(*LIBL/QBATCH)
```

註: 工作佇列與子系統說明相關。 如果您發現工作保留在工作佇列上，請驗證工作佇列定義已定義在 SBSD 上。 對子系統使用 DSPSBSD 指令，並採用選項 6，工作佇列登錄。

現在佇列管理程式 TESTQM 的所有工作:

- 在佇列管理程式檔案庫中使用新的本端預設工作說明
- 提交至工作佇列 QBATCH。

若要確保正確遞送工作並設定其優先順序，請執行下列動作:

- 在子系統 QBATCH 中建立 IBM MQ 工作的遞送登錄，或
- 不論使用何種遞送資料，都依賴呼叫 QCMD 的全面遞送登錄。

僅當工作佇列 QBATCH 的最大作用中工作選項設定為 *NOMAX 時，此選項才有效。 系統預設值為 1。

4. 下列範例會建立另一個 IBM MQ 子系統

a. 發出下列指令，在 QMQM 程式庫中建立重複的子系統:

```
CRTDUPOBJ OBJ(QMQM) FROMLIB(QMQM) OBJTYPE(*SBSD) TOLIB(QMQM) NEWOBJ(QMQM2)
```

b. 發出下列指令來移除 QMQM 工作佇列:

```
RMVJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM)
```

c. 發出下列指令，為子系統建立新的工作佇列:

```
CRTJOBQ JOBQ(QMQM/QMQM2) TEXT('Job queue for IBM MQ Queue Manager')
```

d. 發出下列指令，將工作佇列登錄新增至子系統:

```
ADDJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM2) MAXACT(*NOMAX)
```

e. 發出下列指令，在佇列管理程式庫中建立重複的 QMQMJOBD:

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMlibraryname)
```

f. 發出下列指令，變更工作說明以使用新的工作佇列:

```
CHGJOBD JOBD(QMlibraryname/QMQMJOBD) JOBQ(QMQM/QMQM2)
```

g. 發出下列指令來啟動子系統:

```
STRSBS SBSD(QMQM/QMQM2)
```

註:

a. 您可以在任何檔案庫中指定子系統。 如果基於任何原因重新安裝產品或取代 QMQM 程式庫，則會移除您所做的任何變更。

b. 與 QMlibraryname 相關聯的所有佇列管理程式工作現在在子系統 QMQM2 下執行。

IBM i IBM i 上的可用性、備份、回復及重新啟動

使用此資訊來瞭解 IBM MQ for IBM i 如何使用 IBM i 日誌登載支援來協助其備份及還原策略。

在閱讀本節之前，您必須先熟悉標準 IBM i 備份及回復方法，以及使用 IBM i 上的異動日誌及其相關異動日誌接收器。 如需這些主題的相關資訊，請參閱 備份及回復。

若要瞭解備份及回復策略，您首先需要瞭解 IBM MQ for IBM i 如何在 IBM i 檔案系統及整合檔案系統 (IFS) 中組織其資料。

IBM MQ for IBM i 會將其資料保留在每一個佇列管理程式實例的個別檔案庫中，以及 IFS 檔案系統中的串流檔中。

佇列管理程式特定檔案庫包含日誌登載、異動日誌接收器及控制佇列管理程式工作管理所需的物件。 IFS 目錄及檔案包含 IBM MQ 配置檔、 IBM MQ 物件的說明，以及它們包含的資料。

在將這些物件套用至適當的物件 之前 ，這些物件的每一項變更都會記錄在異動日誌中 (可透過系統失效來回復)。 這會透過重播日誌中所記錄的資訊來回復這類變更。

您可以將「 IBM MQ for IBM i 」配置成使用不同伺服器上的多個佇列管理程式實例，以在伺服器或佇列管理程式失敗時提供更高的佇列管理程式可用性，並加快回復速度。

IBM i IBM i 上的佇列管理程式日誌

使用此資訊來瞭解 IBM MQ for IBM i 如何在其作業中使用日誌登載來控制本端物件的更新。

每一個佇列管理程式檔案庫都包含該佇列管理程式的日誌登載，且日誌登載具有名稱 QM *GRLIB*/AMQ *A* JRN，其中 QM *GRLIB* 是佇列管理程式檔案庫的名稱，而 *A* 是單一實例佇列管理程式的字母 A ，對於佇列管理程式實例而言是唯一的。

QM *GRLIB* 採用名稱 QM，後面接著唯一格式的佇列管理程式名稱。 例如，名為 TEST 的佇列管理程式具有名為 QMTEST 的佇列管理程式庫。 使用 **CRTMQM** 指令建立佇列管理程式時，可以指定佇列管理程式檔案庫。

異動日誌具有相關的異動日誌接收器，其中包含要登載的資訊。 接收器是只能附加資訊且最終會填滿的物件。

異動日誌接收器會使用過期資訊的寶貴磁碟空間。 不過，您可以將資訊放在永久儲存體中，以將這個問題降到最低。 在任何特定時間，都會將一個異動日誌接收器連接至異動日誌。 如果異動日誌接收器達到其預定的臨界值大小，則會將它分離並由新的異動日誌接收器取代。 當您使用 **CRTMQM** 及 **THRESHOLD** 參數建立佇列管理程式時，可以指定異動日誌接收器的臨界值。

與本端 IBM MQ for IBM i 異動日誌相關聯的異動日誌接收器存在於每一個佇列管理程式檔案庫中，並採用如下的命名慣例:

```
AMQ Arnnnnn
```

其中

A
是字母 A-Z。 對於單一實例佇列管理程式，它是 A 。 它會因多重實例佇列管理程式的不同實例而有所不同。

NNNNN
是順序中下一個異動日誌加 1 的十進位 00000 to 99999 。

r
是十進位 0 to 9，每次還原接收端時都會增加 1。

日誌的順序是根據日期。 不過，下一個異動日誌的命名是根據下列規則:

1. AMQArnnnnn 會移至 AMQAr(nnnnn+1)，並在 nnnnn 到達 99999 時折返。 例如， AMQA099999 會移至 AMQA000000，而 AMQA999999 會移至 AMQA900000。
2. 如果具有規則 1 所產生名稱的日誌已存在，則訊息 CPI7OE3 會傳送至 QSYSOPR 訊息佇列，且自動接收端切換會停止。

   繼續使用目前連接的接收端，直到您調查問題並手動連接新的接收端為止。
3. 如果順序中沒有可用的新名稱 (亦即，所有可能的異動日誌名稱都在系統上) ，則您需要執行下列兩項:
   a. 刪除不再需要的日誌 (請參閱 第 319 頁的『IBM i 上的日誌登載管理』)。
   b. 使用 (**RCDMQMIMG**) 將異動日誌變更記錄至最新的異動日誌接收器 然後重複前一個步驟。 這容許重複使用舊的異動日誌接收器名稱。

AMQAJRN 異動日誌會使用 MNGRCV(*SYSTEM) 選項，讓作業系統在達到臨界值時自動變更異動日誌接收器。 如需系統如何管理接收端的相關資訊，請參閱 *IBM i* 備份及回復。

異動日誌接收器的預設臨界值為 100,000 KB。 當您建立佇列管理程式時，可以將此值設為較大的值。 LogReceiverSize 屬性的初始值寫入 mqs.ini 檔案的 LogDefaults 節。

當異動日誌接收器延伸超出其指定的臨界值時，會分離接收器並建立新的異動日誌接收器，繼承前一個接收器的屬性。 當系統自動連接新的異動日誌接收器時，會忽略在建立佇列管理程式之後對 LogReceiverSize 或 LogASP 屬性所做的變更

如需配置系統的進一步詳細資料，請參閱 變更 IBM i 上的配置資訊 。

如果您需要在建立佇列管理程式之後變更異動日誌接收器的大小，請使用下列指令建立新的異動日誌接收器，並將其擁有者設為 QMQM：

```
CRTJRNRCV JRNRCV(QM GRLIB/AMQ Arnnnnn) THRESHOLD(xxxxxx) +
TEXT('MQM LOCAL JOURNAL RECEIVER')
CHGOBJOWN OBJ(QM GRLIB/AMQ Arnnnnn) OBJTYPE(*JRNRCV) NEWOWN(QMQM)
```

其中

QMGRLIB
是佇列管理程式檔案庫的名稱

A
是實例 ID (通常是 A)。

rnnnnn
是先前說明的命名順序中的下一個異動日誌接收器

xxxxxx
是新的接收端臨界值 (KB)

註: 接收端的大小上限由作業系統控管。 若要檢查此值，請查看 **CRTJRNRCV** 指令上的 THRESHOLD 關鍵字。

現在，使用下列指令將新的接收器連接至 AMQAJRN 日誌登載:

```
CHGJRN JRN(QMGRLIB/AMQ A JRN) JRNRCV(QMGRLIB/AMQ Annnnnn)
```

如需如何管理這些異動日誌接收器的詳細資料，請參閱 第 319 頁的『IBM i 上的日誌登載管理』 。

IBM i *IBM i 上的佇列管理程式日誌登載使用情形*

使用此資訊來瞭解 IBM MQ for IBM i 如何在其作業中使用日誌登載來控制本端物件的更新。

訊息佇列的持續更新分兩個階段進行。 代表更新的記錄會先寫入日誌登載，然後更新佇列檔。

因此，異動日誌接收器會變得比佇列檔更最新。 為了確保從一致點開始重新啟動處理程序， IBM MQ 會使用檢查點。

檢查點是指異動日誌中說明的記錄與佇列中的記錄相同的時間點。 檢查點本身是由重新啟動佇列管理程式所需的日誌登載記錄系列所組成。 例如，檢查點時所有作用中交易 (即工作單元) 的狀態。

IBM MQ 會自動產生檢查點。 當佇列管理程式啟動及關閉時，以及在記載特定數目的作業之後，即會採用它們。

您可以對佇列管理程式上的所有物件發出 RCDMQMIMG 指令並顯示結果，以強制佇列管理程式取得檢查點，如下所示:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME) DSPJRNDTA(*YES)
```

當佇列處理進一步訊息時，檢查點記錄會變成與佇列的現行狀態不一致。

當 IBM MQ 重新啟動時，它會在日誌中尋找最新的檢查點記錄。 此資訊保留在每個檢查點結束時更新的檢查點檔案中。 檢查點記錄代表日誌與資料之間的最新一致性點。 來自此檢查點的資料用來重建在檢查點時存在的佇列。 當重新建立佇列時，即會向前播放日誌，讓佇列回到系統失效或關閉之前的狀態。

若要瞭解 IBM MQ 如何使用日誌登載，請考量在佇列管理程式 TEST 中名為 TESTQ 的本端佇列的情況。 這由 IFS 檔案代表:

```
/QIBM/UserData/mqm/qmgrs/TEST/queues
```

如果指定的訊息放置在此佇列上，然後從佇列中擷取，則發生的動作會顯示在圖 第 317 頁的圖 25 中。

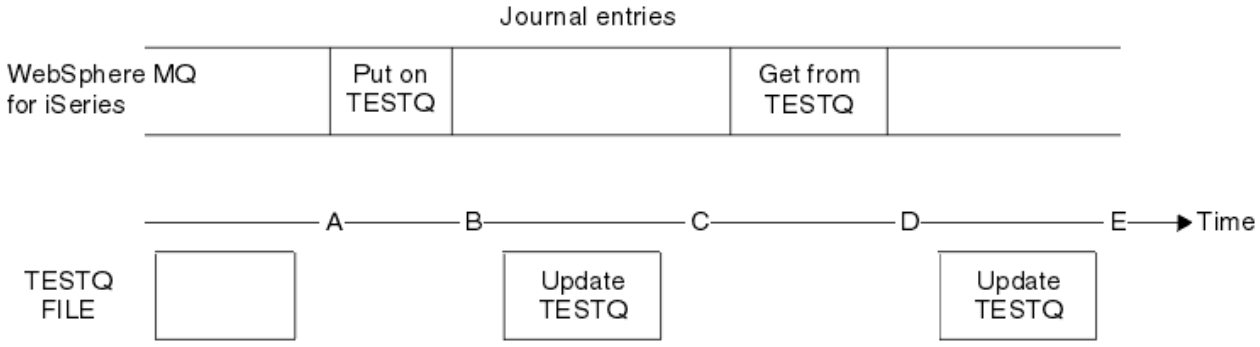


圖 *25:* 更新 *MQM* 物件時的事件順序

圖表中顯示的五個點 A 到 E 代表定義下列狀態的時間點:

A
佇列的 IFS 檔案表示法與異動日誌中包含的資訊一致。

B
異動日誌項目會寫入佇列上定義 Put 作業的異動日誌。

C
對佇列進行適當的更新。

D
日誌登載項目會寫入佇列中定義 Get 作業的日誌登載。

E
對佇列進行適當的更新。

IBM MQ for IBM i 回復功能的關鍵在於使用者可以將 TESTQ 的 IFS 檔案表示法儲存為時間 A，隨後透過還原已儲存的物件並從時間 A 開始重播日誌登載中的項目，以回復 TESTQ 的 IFS 檔案表示法 (即時間 E)。

在系統失效之後， IBM MQ for IBM i 會使用此策略來回復持續訊息。 IBM MQ 會記住異動日誌接收器中的特定項目，並確保在啟動時從此時開始重播異動日誌中的項目。 此啟動項目會定期重新計算，因此 IBM MQ 只需要在下一次啟動時執行必要的重播下限。

IBM MQ 提供物件的個別回復。 與物件相關的所有持續性資訊都會記錄在本端 IBM MQ for IBM i 日誌登載中。 任何變成損壞或毀損的 IBM MQ 物件都可以從保留在異動日誌中的資訊完全重建。

如需系統如何管理接收端的相關資訊，請參閱 第 314 頁的『IBM i 上的可用性、備份、回復及重新啟動』。

IBM i *IBM i 上的媒體映像檔*

在 IBM i 上，媒體映像檔是記錄在日誌中的 IBM MQ 物件的完整副本。 部分毀損或損壞的物件可以從其媒體映像檔自動回復。

長持續時間的 IBM MQ 物件可以代表大量日誌登載項目，回到建立它的點。 為了避免此情況， IBM MQ for IBM i 具有物件的媒體影像概念。

此媒體影像是記錄在異動日誌中的 IBM MQ 物件完整副本。 如果取得物件的影像，則可以透過重播從此影像開始的異動日誌項目來重建物件。 異動日誌中代表每一個 IBM MQ 物件的重播點的項目稱為其媒體回復項目。 IBM MQ 會追蹤:

- 每一個佇列管理程式物件的媒體回復項目。
- 此集合內最舊的項目 (如需詳細資料，請參閱 第 319 頁的『IBM i 上的日誌登載管理』 中的錯誤訊息 AMQ7462 。

會定期取得 *CTLG 物件及 *MQM 物件的影像，因為這些物件對於佇列管理程式重新啟動非常重要。

其他物件的影像會在方便時拍攝。 依預設，當使用具有參數 ENDCCTJOB (*YES) 的 **ENDMQM** 指令關閉佇列管理程式時，會取得所有物件的影像。 對於非常大的佇列管理程式，這項作業可能需要相當長的時間。 如果您需要快速關閉，請指定參數 RCDMQMIMG (*NO) 與 ENDCCTJOB (*YES)。 在這種情況下，建議您使用下列指令，在重新啟動佇列管理程式之後，將完整媒體映像檔記錄在日誌登載中:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME)
```

IBM MQ 會自動記錄物件的影像，如果它發現可由日誌中的小項目緊密說明物件的方便點。 不過，某些物件 (例如，一致包含大量訊息的佇列) 可能永遠不會發生這種情況。

請使用 IBM MQ 指令 RCDMQMIMG，它可讓您手動取得所選取物件的影像，而不是讓最舊媒體回復項目的日期持續一段不必要的長時間。

從媒體映像檔回復

如果發現部分物件已毀損或損壞，則 IBM MQ 會自動從其媒體映像檔回復這些物件。 尤其是在一般佇列管理程式啟動期間，這會套用至特殊 *MQM 及 *CTLG 物件。 如果在前次關閉佇列管理程式時有任何同步點交易未完成，則也會自動回復任何受影響的佇列，以完成啟動作業。

您必須使用 IBM MQ 指令 RCRMQMOBJ 手動回復其他物件。 此指令會重播日誌登載中的項目，以重建 IBM MQ 物件。 如果 IBM MQ 物件損壞，則唯一有效的動作是刪除它或透過此方法重建它。 不過請注意，無法以這種方式回復非持續訊息。

IBM i *IBM MQ for IBM i 上的檢查點*

在不同時間會採用檢查點，以提供已知的一致起始點來進行回復。

處理程序 AMQZMUC0 內的檢查點執行緒負責取得下列點的檢查點:

- 佇列管理程式啟動 (STRMQM)。
- 佇列管理程式關閉 (ENDMQM)。
- 自前次檢查點以來已經過一段時間 (預設期間為 30 分鐘)，且自前次檢查點以來已寫入日誌記錄數下限 (預設值為 100)。
- 在寫入一些日誌記錄之後。 預設值為 10 000。
- 在超出異動日誌臨界值大小且已自動建立新的異動日誌接收器之後。
- 使用下列項目取得完整媒體映像檔時:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME) DSPJRNDTA(*YES)
```

IBM i IBM MQ for IBM i 資料的備份

使用此資訊來瞭解每一個佇列管理程式的兩種 IBM MQ 備份類型。

對於每一個佇列管理程式，有兩種類型的 IBM MQ 備份可供考量:

- 資料及日誌登載備份。

  若要確保這兩組資料一致，請僅在關閉佇列管理程式之後執行此動作。
- 日誌登載備份。

  您可以在佇列管理程式處於作用中狀態時執行此動作。

對於這兩種方法，您都需要尋找佇列管理程式 IFS 目錄及佇列管理程式檔案庫的名稱。 您可以在 IBM MQ 配置檔 (mqs.ini) 中找到這些檔案。 如需相關資訊，請參閱 QueueManager 段落。

請使用下列程序來執行兩種類型的備份:

特定佇列管理程式的資料及日誌登載備份

> **註: 當佇列管理程式在執行中，請勿使用「作用中時儲存」要求。 除非已確定或回復具有擱置變更的所有確定定義，否則無法完成此類要求。 如果在佇列管理程式處於作用中狀態時使用此指令，則通道連線可能不會正常結束。 請一律使用下列程序。**

1. 使用下列指令建立空的異動日誌接收器:

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. 使用 **RCDMQMIMG** 指令來記錄所有 IBM MQ 物件的 MQM 映像檔，然後使用下列指令強制執行檢查點:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 結束通道並確定佇列管理程式不在執行中。 如果佇列管理程式正在執行中，請使用 **ENDMQM** 指令停止它。
4. 發出下列指令來備份佇列管理程式檔案庫:

```
SAVLIB LIB(QMTEST)
```

5. 發出下列指令來備份佇列管理程式 IFS 目錄:

```
SAV DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/test'))
```

特定佇列管理程式的日誌登載備份

因為所有相關資訊都保留在異動日誌中，只要您在某個時間執行完整儲存，就可以透過儲存異動日誌接收器來執行部分備份。 這些會記錄自完整備份之後的所有變更，並透過發出下列指令來執行:

1. 使用下列指令建立空的異動日誌接收器:

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. 使用 **RCDMQMIMG** 指令來記錄所有 IBM MQ 物件的 MQM 映像檔，然後使用下列指令強制執行檢查點:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 使用下列指令儲存異動日誌接收器:

```
SAVOBJ OBJ(AMQ*) LIB(QMTEST) OBJTYPE(*JRNRCV) .........
```

簡式備份策略是每週執行 IBM MQ 程式庫的完整備份，並執行每日日誌登載備份。 這當然取決於您如何為企業設定備份策略。

IBM i *IBM i 上的日誌登載管理*

作為備份策略的一部分，請注意異動日誌接收器。 由於各種原因，從 IBM MQ 檔案庫中移除異動日誌接收器非常有用:

- 釋放空間; 這會套用至所有異動日誌接收器
- 啟動時增進效能 (STRMQM)
- 增進重建物件的效能 (RCRMQMOBJ)

在刪除異動日誌接收器之前，您必須小心擁有備份副本，且不再需要異動日誌接收器。

異動日誌接收器 在 已從異動日誌分離並儲存之後，只要它們在回復作業需要時可供還原，即可從佇列管理程式檔案庫中移除它們。

日誌登載管理的概念顯示在 第 320 頁的圖 26 中。

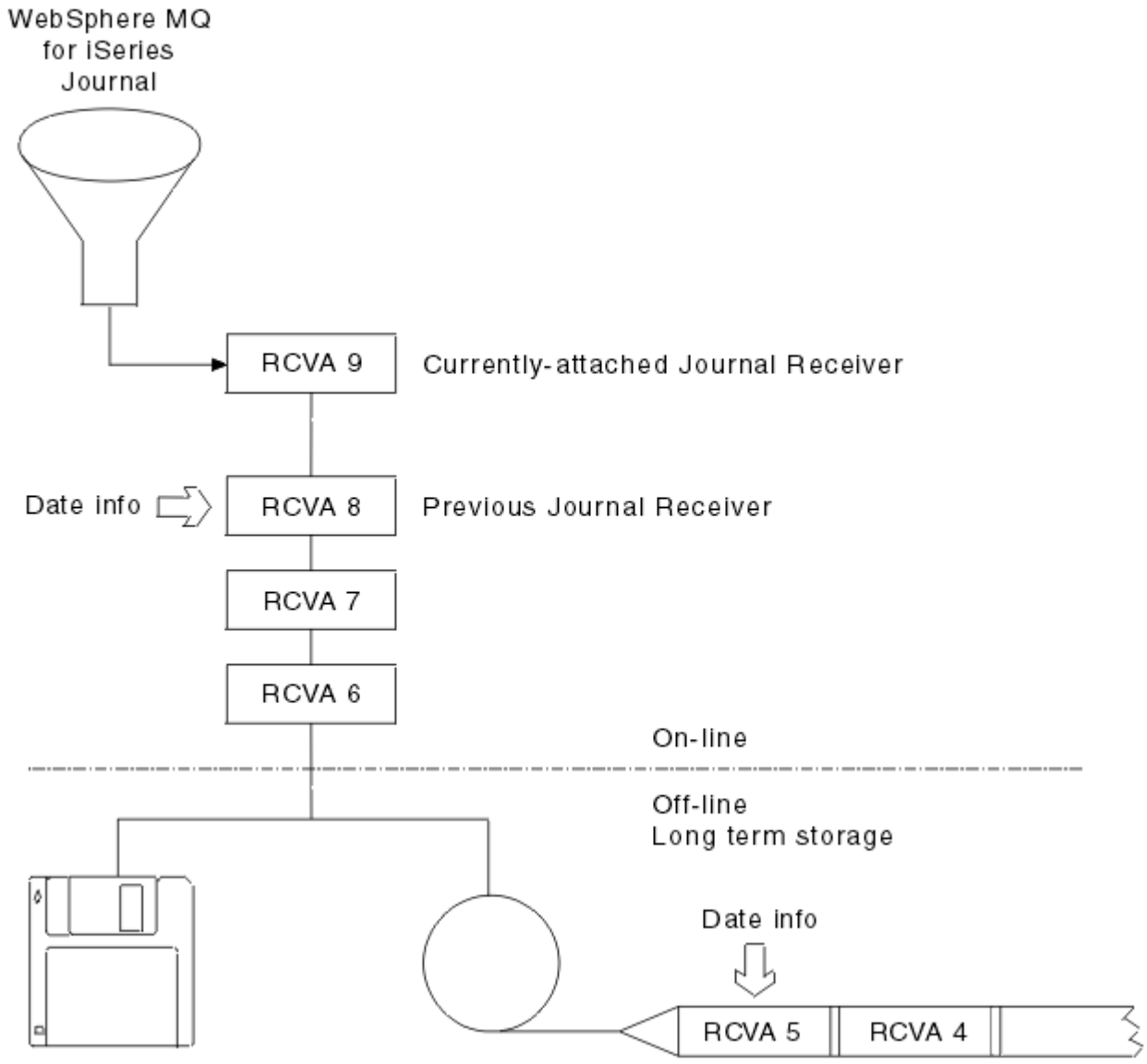


圖 *26: IBM i* 上的日誌登載

請務必瞭解日誌登載 IBM MQ 可能需要回溯多久，以判定何時可以從佇列管理程式檔案庫中移除已備份的日誌登載接收端，以及何時可以捨棄備份本身。

IBM MQ 會向佇列管理程式訊息佇列 (佇列管理程式檔案庫中的 QMQMMSG) 發出兩則訊息，以協助判斷此時間。 當啟動時，當它變更本端異動日誌接收器，且您使用 RCDMQIMG 來強制檢查點時，即會發出這些訊息。 這兩個訊息如下:

AMQ7460
啟動回復點。 此訊息定義啟動項目的日期和時間， IBM MQ 會在啟動回復通過時從中重播日誌登載。 如果包含此記錄的異動日誌接收器在 IBM MQ 檔案庫中可用，則此訊息也會包含包含此記錄的異動日誌接收器名稱。

AMQ7462
最舊的媒體回復登錄。 此訊息定義用來從其媒體映像檔重建物件的最舊項目的日期和時間。

所識別的異動日誌接收器是所需要的最舊異動日誌接收器。 不再需要任何其他具有較舊建立日期的 IBM MQ 異動日誌接收器。 如果只顯示星星，則您需要從指示的日期還原備份，以判定哪一個是最舊的異動日誌接收器。

當記載這些訊息時， IBM MQ 也會將使用者空間物件寫入只包含一個項目的佇列管理程式檔案庫: 需要保留在系統上的最舊異動日誌接收器名稱。 此使用者空間稱為 AMQJRNINF ，且資料會以下列格式寫入:

```
JJJJJJJJJJLLLLLLLLLLYYYYMMDDHHMMSSmmm
```

其中:

JJJJJJJJJJ
是 IBM MQ 仍需要的最舊接收端名稱。

LLLLLLLLLL
是異動日誌接收器檔案庫名稱。

YYYY
是 IBM MQ 需要的最舊日誌登載項目的年份。

MM
是 IBM MQ 需要的最舊日誌登載項目的月份。

DD
是 IBM MQ 需要的最舊日誌登載項目的日期。

HH
是 IBM MQ 需要的最舊日誌登載項目的小時。

SS
是 IBM MQ 需要的最舊異動日誌項目的秒數。

mmm
是 IBM MQ 需要的最舊日誌登載項目的毫秒數。

從系統中刪除最舊的異動日誌接收器時，此使用者空間包含星號 (*) 作為異動日誌接收器名稱。

註: 定期執行 RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) 可以節省 IBM MQ 的啟動時間，並減少為了回復而需要儲存及還原的本端異動日誌接收器數目。

除非 IBM MQ for IBM i 正在對啟動或重建物件執行回復傳遞，否則它不會參照異動日誌接收器。 如果發現其需要的異動日誌不存在，則會向佇列管理程式訊息佇列 (QMQMMSG) 發出訊息 AMQ7432 ，報告完成回復傳遞所需的異動日誌項目的時間及日期。

如果發生此情況，請從備份中還原在此日期之後分離的所有異動日誌接收器，以容許回復傳遞成功。

保留包含啟動項目的異動日誌接收器，以及佇列管理程式檔案庫中可用的任何後續異動日誌接收器。

保留包含最舊 Media Recovery Entry 及任何後續異動日誌接收器的異動日誌接收器 (隨時可用) ，以及存在於佇列管理程式檔案庫或備份中。

當您強制檢查點時:

- 如果 AMQ7460 中所指名的異動日誌接收器未進階，則表示需要確定或回復不完整的工作單元。
- 如果未進階 AMQ7462 中所命名的異動日誌接收器，則表示有一個以上損壞的物件。

IBM i 在 *IBM i* 上還原完整佇列管理程式 (資料及日誌登載)

使用此資訊可從備份或遠端機器還原一或多個佇列管理程式。

如果您需要從備份回復一或多個 IBM MQ 佇列管理程式，請執行下列步驟。

1. 靜止 IBM MQ 佇列管理程式。
2. 尋找最新備份集，由最新完整備份及後續備份的異動日誌接收器組成。
3. 發出下列指令，從完整備份執行 RSTLIB 作業，將 IBM MQ 資料檔案庫還原至完整備份時的狀態:

   ```
   RSTLIB LIB(QMQRLIB1) .........
   RSTLIB LIB(QMQRLIB2) .........
   ```

   如果異動日誌接收器部分儲存在一個異動日誌備份中，且完整儲存在後續的備份中，則只還原完全儲存的異動日誌接收器。 依時間順序個別還原日誌。

4. 使用下列指令執行 RST 作業，將 IBM MQ IFS 目錄還原至 IFS 檔案系統:

```
RST DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/testqm')) ...
```

5. 啟動訊息佇列管理程式。 這會重播自完整備份以來寫入的所有日誌登載記錄，並將所有 IBM MQ 物件還原至日誌登載備份時的一致狀態。

如果您要在不同機器上還原完整佇列管理程式，請使用下列程序來還原佇列管理程式檔案庫中的所有項目。 (我們使用 TEST 作為範例佇列管理程式名稱。)

1. CRTMQM TEST
2. DLTLIB LIB(QMTEST)
3. RSTLIB SAVLIB(QMTEST) DEV(*SAVF) SAVF(QMGRLIBSAV)
4. 刪除下列 IFS 檔案:

```
/QIBM/UserData/mqm/qmgrs/TEST/QMQMCHKPT
/QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMQMOBJCAT
/QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMANAGER
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.AUTH.DATA.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CHANNEL.INITQ/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.COMMAND.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.REPOSITORY.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.TRANSMIT.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.PENDING.DATA.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.ADMIN.COMMAND.QUEUE/q
```

5. STRMQM TEST
6. RCRMQMOBJ OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(TEST)

IBM i 在 *IBM i* 上還原特定佇列管理程式的異動日誌接收器

使用此資訊來瞭解還原異動日誌接收器的不同方式。

最常見的動作是如果後續回復功能再次需要已移除的接收器，則將已備份的異動日誌接收器還原至佇列管理程式檔案庫。

這是一項簡式作業，需要使用標準 IBM i RSTOBJ 指令來還原異動日誌接收器:

```
RSTOBJ OBJ(QMQMDATA/AMQA000005) OBJTYPE(*JRNRCV) .........
```

可能需要還原一系列異動日誌接收器，而不是單一接收器。 例如， AMQA000007 是 IBM MQ 程式庫中最舊的接收端，且 AMQA000005 和 AMQA000006 都需要還原。

在此情況下，請依反向時間順序個別還原接收端。 這並不總是必要的，但卻是良好的實踐。 在嚴重狀況下，您可能需要使用 IBM i 指令 WRKJRNA ，將已還原的異動日誌接收器與異動日誌建立關聯。

還原異動日誌時，系統會以異動日誌接收器順序中的新名稱自動建立連接的異動日誌接收器。 不過，產生的新名稱可能與您需要還原的異動日誌接收器相同。 需要人為介入來克服此問題; 依序建立新的名稱異動日誌接收器，並在還原異動日誌接收器之前建立新的異動日誌。

例如，考量已儲存的異動日誌 AMQAJRN 及下列異動日誌接收器的問題:

- AMQA000000
- AMQA100000
- AMQA200000
- AMQA300000
- AMQA400000
- AMQA500000
- AMQA600000

- AMQA700000
- AMQA800000
- AMQA900000

將異動日誌 AMQAJRN 還原至佇列管理程式檔案庫時，系統會自動建立異動日誌接收器 AMQA000000。 此自動產生的接收器與您要還原且無法還原的其中一個現有異動日誌接收器 (AMQA000000) 衝突。

解決方案是:

1. 手動建立下一個異動日誌接收器 (請參閱 第 315 頁的『IBM i 上的佇列管理程式日誌』):

```
CRTJRNRCV JRNRCV(QMQRLIB/AMQA900001) THRESHOLD(XXXXX)
```

2. 使用異動日誌接收器手動建立異動日誌:

```
CRTJRN JRN(QMGRLIB/AMQAJRN) MNGRCV(*SYSTEM) +
JRNRCV(QMGRLIB/AMQA9000001) MSGQ(QMGRLIB/AMQAJRNMSG)
```

3. 將本端異動日誌接收器 AMQA000000 還原為 AMQA900000。

IBM i IBM i 上的多重實例佇列管理程式

多重實例佇列管理程式透過在作用中伺服器失敗時自動切換至待用伺服器，來改善可用性。 作用中及待用伺服器是相同佇列管理程式的多重實例；它們共用相同的佇列管理程式資料。 如果作用中實例失敗，您需要將其日誌登載傳送至接管的待命資料庫，以便佇列管理程式可以重建其佇列。

配置您正在執行多重實例佇列管理程式的 IBM i 系統，以便在作用中佇列管理程式實例失敗時，它所使用的日誌登載可供接管的待命實例使用。 您可以設計自己的配置及管理作業，讓作用中實例的日誌登載可供接管的實例使用。 如果您不想遺失訊息，您的設計必須確保待命日誌與失敗點的作用中日誌一致。 您可以從兩個配置中的其中一個來調整您的設計，這些配置會在後續主題中以維護一致性的範例來說明。

1. 將日誌登載從執行作用中佇列管理程式實例的系統鏡映至執行待命實例的系統。
2. 將異動日誌放置在可從執行作用中實例的系統轉移到待命實例的「獨立輔助儲存區 (IASP)」中。

第一個解決方案不需要其他硬體或軟體，因為它使用基本 ASP。 第二個解決方案需要切換式 IASP ，其需要 IBM i 叢集作業支援，可作為個別計價的 IBM i 授權產品 5761-SS1 選項 41。

IBM i *IBM i* 上的可靠性和可用性

多重實例佇列管理程式旨在改善應用程式的可用性。 技術和實體限制意味著您需要不同的解決方案，以滿足災難回復、備份佇列管理程式及連續作業的需求。

在配置可靠性和可用性時，您會交換大量因素，產生四個不同的設計點:

災難回復
已針對毀損所有本端資產的重大災難之後的回復最佳化。

IBM i 上的災難回復通常基於 IASP 的地理鏡映。

備份
在本地化失敗 (通常是人為錯誤或某些無法預期的技術問題) 之後針對回復進行最佳化。

「 IBM MQ 」提供備份佇列管理程式，以定期備份佇列管理程式。 您也可以使用佇列管理程式日誌登載的非同步抄寫來改善備份的貨幣。

可用性
已針對快速還原作業進行最佳化，在可預見的技術故障 (例如伺服器或磁碟故障) 之後出現幾乎不中斷的服務。

回復通常以分鐘為單位來測量，偵測有時需要比回復處理程序更長的時間。 多重實例佇列管理程式可協助您配置 可用性。

連續作業
已針對提供不中斷服務進行最佳化。

連續操作解決方案必須解決偵測問題，幾乎總是涉及透過多個系統提交相同的工作，並使用第一個結果，或如果正確性是主要考量，則至少比較兩個結果。

多重實例佇列管理程式可協助您配置 可用性。 一次有一個佇列管理程式實例在作用中。 根據系統的配置、載入及調整方式，切換至待命實例所需的時間從 10 秒以上到 15 分鐘以上。

如果與可重新連接的 IBM MQ MQI clients 搭配使用，則多重實例佇列管理程式可以提供幾乎不中斷的服務外觀，這些可繼續處理，而無需應用程式瞭解佇列管理程式中斷; 請參閱主題 自動化用戶端重新連線。

IBM i *IBM i* 上高可用性解決方案的元件

使用多重實例佇列管理程式來建構高可用性解決方案，方法是針對佇列管理程式資料提供健全的網路儲存體、佇列管理程式日誌登載的日誌抄寫或健全的 IASP 儲存體，並使用配置為可重新啟動佇列管理程式服務之應用程式的可重新連接用戶端。

多重實例佇列管理程式會回復在另一部伺服器上啟動另一個佇列管理程式實例，以回應佇列管理程式失敗的偵測。 若要完成其啟動，實例需要存取網路儲存體中的共用佇列管理程式資料，以及其本端佇列管理程式日誌登載副本。

若要建立高可用性解決方案，您需要管理佇列管理程式資料的可用性、本端佇列管理程式日誌登載的貨幣，以及建置可重新連接的用戶端應用程式，或將應用程式部署為佇列管理程式服務，以在佇列管理程式回復時自動重新啟動。 IBM MQ classes for Java 不支援自動重新連接用戶端。

佇列管理程式資料

將佇列管理程式資料放置在共用、高可用性及可靠的網路儲存體上，可能是使用 RAID 層次 1 磁碟或更高層次。 檔案系統需要符合多重實例佇列管理程式之共用檔案系統的需求; 如需共用檔案系統需求的相關資訊，請參閱 共用檔案系統的需求。 「網路檔案系統 4」(NFS4) 是符合這些需求的通訊協定。

佇列管理程式日誌登載

您還需要配置佇列管理程式實例所使用的 IBM i 日誌登載，以便待命實例能夠將其佇列管理程式資料還原至一致狀態。 對於不中斷服務，這表示當作用中實例失敗時，您必須將日誌登載還原至其狀態。 與備份或災難回復解決方案不同，將異動日誌還原至較早的檢查點是不夠的。

您無法在網路儲存體上的多個 IBM i 系統之間實際共用日誌登載。 若要在失敗點將佇列管理程式日誌登載還原為一致狀態，您需要將失敗時作用中佇列管理程式實例的本端實體日誌登載，傳送至已啟動的新實例，或在執行中待命實例上維護日誌登載的鏡映。 鏡映異動日誌是與屬於失敗實例的本端異動日誌保持完全同步的遠端異動日誌抄本。

三個配置是設計如何管理多重實例佇列管理程式的日誌登載的起點。

1. 使用從作用中實例 ASP 到待命實例 ASP 的同步化日誌登載抄寫 (日誌登載鏡映)。
2. 將您已配置為保留佇列管理程式日誌登載的 IASP 從作用中實例傳送至接管作為作用中實例的待命實例。
3. 使用已同步的次要 IASP 鏡映。

如需將佇列管理程式資料放入 iASP 的相關資訊，請參閱 IBM MQ IBM i CRTMQM 指令的 ASP 選項。

另請參閱 IBM Documentation 中 IBM i 資訊中的 高可用性 。

應用程式

若要建置用戶端以在待命佇列管理程式回復時自動重新連接至佇列管理程式，請使用 MQCONNX 將應用程式連接至佇列管理程式，並在 **MQCNO 選項** 欄位中指定 MQCNO_RECONNECT_Q_MGR 。 如需設計用戶端應用程式以進行回復的相關資訊，請參閱 高可用性範例程式 ，以取得使用可重新連接用戶端的三個範例程式，以及 應用程式回復 。

IBM i 在 *IBM i* 上使用 *NetServer* 建立佇列管理程式資料的網路共用

在 IBM i 伺服器上建立網路共用，以儲存佇列管理程式資料。 設定來自兩部伺服器 (即將管理佇列管理程式實例) 的連線，以存取網路共用。

開始之前

- 此作業需要三部 IBM i 伺服器。 網路共用定義在其中一部伺服器 GAMMA 上。 其他兩部伺服器 (ALPHA 和 BETA) 將連接至 GAMMA。
- 在所有三部伺服器上安裝 IBM MQ 。
- 安裝 System i Navigator; 請參閱 System i Navigator。

關於這項作業

- 在 GAMMA 上建立佇列管理程式目錄，並設定使用者設定檔 QMQM 及 QMQMADM 的正確所有權及許可權。 在 GAMMA 上安裝 IBM MQ 可輕鬆建立目錄及許可權。
- 使用 System i Navigator 來建立與 GAMMA 上佇列管理程式資料目錄的共用。
- 在 ALPHA 和 BETA 上建立指向共用的目錄。

程序

1. 在 GAMMA 上，建立目錄以管理佇列管理程式資料，並以 QMQM 使用者設定檔作為擁有者，並以 QMQMADM 作為主要群組。

   提示:

   建立具有正確許可權的目錄的快速可靠方法是在 GAMMA 上安裝 IBM MQ 。

   稍後，如果您不想在 GAMMA 上執行 IBM MQ ，請解除安裝 IBM MQ。 解除安裝之後，目錄 /QIBM/UserData/mqm/qmgrs 會保留在 GAMMA 上，擁有者為 QMQM 使用者設定檔，以及 QMQMADM 主要群組。

   作業會使用 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 目錄來進行共用。
2. 啟動 System i Navigator **新增連線** 精靈，並連接至 GAMMA 系統。
   a) 按兩下 Windows 桌面上的 **System i Navigator** 圖示。
   b) 按一下 **是** 以建立連線。
   c) 遵循 **新增連線** 精靈中的指示，並建立從 IBM i 系統到 GAMMA 的連線。

      GAMMA 的連線會新增至 **我的連線**。
3. 在 GAMMA 上新增檔案共用。
   a) 在「 **System i Navigator** 」視窗中，按一下 My Connections/GAMMA/File Systems 中的 File Shares 資料夾。
   b) 在「 **我的作業** 」視窗中，按一下 **管理 IBM i NetServer 共用**。

      即會在桌面上開啟新視窗 **IBM i NetServer -GAMMA**，並顯示共用物件。
   c) 用滑鼠右鍵按一下 Shared Objects 資料夾> **檔案** > **新建** > **檔案**。

      即會開啟新視窗 **IBM i NetServer 檔案共用-GAMMA**。
   d) 例如，為共用提供名稱 WMQ 。
   e) 將存取控制設為 Read/Write。
   f) 瀏覽至您先前建立的 /QIBM/UserData/mqm/qmgrs 目錄，以選取 **路徑名稱** ，然後按一下 **確定**。

      IBM i NetServer 檔案共用-GAMMA 視窗會關閉，且 WMQ 會列在共用物件視窗中。
4. 在共用物件視窗中，用滑鼠右鍵按一下 **WMQ** 。 按一下 **檔案** > **許可權**。

   會針對物件 /QIBM/UserData/mqm/qmgrs 開啟 **Qmgrs 許可權-GAMMA** 視窗。
   a) 請檢查 QMQM 的下列許可權 (如果尚未設定的話):

      ```
      Read
      Write
      Execute
      Management
      ```

Existence
Alter
Reference

b) 請檢查 QMQMADM 的下列許可權 (如果尚未設定的話):

Read
Write
Execute
Reference

c) 新增您要授與 /QIBM/UserData/mqm/qmgrs 許可權的其他使用者設定檔。

例如，您可以將預設使用者設定檔 (公用) Read 及 Execute 許可權提供給 /QIBM/UserData/mqm/qmgrs。

5. 請檢查所有獲授與 GAMMA 上 /QIBM/UserData/mqm/qmgrs 存取權的使用者設定檔，其密碼與存取 GAMMA 的伺服器上的密碼相同。

特別是，請確保要存取共用之其他伺服器上的 QMQM 使用者設定檔，與 GAMMA 上的 QMQM 使用者設定檔具有相同的密碼。

提示: 按一下「System i Navigator」中的 My Connections/GAMMA/Users and Groups 資料夾，以設定密碼。 或者，使用 **CHFUSRPRF** 和 **CHGPWD** 指令。

結果

確認您可以使用共用從其他伺服器存取 GAMMA。 如果您正在執行其他作業，請檢查您是否可以使用路徑 /QNTC/GAMMA/WMQ 從 ALPHA 和 BETA 存取 GAMMA。 如果 /QNTC/GAMMA 目錄不存在於 ALPHA 或 BETA 上，則您必須建立該目錄。 視 NetServer 網域而定，在建立目錄之前，您可能必須先 IPL ALPHA 或 BETA。

```
CRTDIR DIR('/QNTC/GAMMA')
```

當您檢查是否可以從發出指令的 ALPHA 或 BETA 存取 /QNTC/GAMMA/WMQ 時， CRTMQM MQMNAME('QM1') MQMDIRP('/QNTC/GAMMA/WMQ') 會在 GAMMA 上建立 /QIBM/UserData/mqm/qmgrs/QM1 。

下一步

遵循作業 第 335 頁的『在 IBM i 上使用日誌登載鏡映及 NetServer 建立多重實例佇列管理程式』 或 第 338 頁的『在 IBM i 上使用 NetServer 及日誌登載鏡映將單一實例佇列管理程式轉換為多重實例佇列管理程式』 中的步驟來建立多重實例佇列管理程式。

IBM i *IBM i 上的失效接手效能*

偵測佇列管理程式實例所花費的時間已失敗，然後在待命資料庫上回復處理所花費的時間可能會根據配置而在數十秒至 15 分鐘之間或更長。 在設計及測試高可用性解決方案時，效能需要成為主要考量。

在決定是配置多重實例佇列管理程式來使用日誌登載抄寫還是使用 IASP 時，需要權衡一些優點和缺點。 鏡映需要佇列管理程式同步寫入遠端異動日誌。 從硬體觀點來看，這不需要影響效能，但從軟體觀點來看，寫入遠端異動日誌所涉及的路徑長度大於僅寫入本端異動日誌所涉及的路徑長度，這可能會在某種程度上降低執行中佇列管理程式的效能。 不過，當待命佇列管理程式接管時，在失敗之前從作用中實例所維護的遠端日誌登載同步化其本端日誌登載的延遲，通常與 IBM i 偵測 IASP 並將其傳送至執行佇列管理程式待命實例的伺服器所花費的時間相比較小。 IASP 傳送時間可以多達 10 到 15 分鐘，而不是在秒內完成。 IASP 傳送時間取決於將 IASP 傳送至待命系統時需要 轉接 的物件數，以及需要合併的存取路徑或索引大小。

當待命佇列管理程式接管時，在失敗之前從作用中實例所維護的遠端日誌中同步化其本端日誌登載的延遲，通常與 IBM i 偵測獨立 ASP 並將其傳送至執行佇列管理程式待命實例的伺服器所花費的時間相比較小。 獨立 ASP 傳送時間可以多達 10 到 15 分鐘，而不是在秒內完成。 獨立 ASP 傳送時間取決於當獨立 ASP 傳送至待命系統時需要 轉接 的物件數，以及需要合併的存取路徑或索引大小。

不過，傳送日誌不是影響待命實例完全回復所花費時間的唯一因素。 您也需要考量網路檔案系統釋放佇列管理程式資料鎖定所花費的時間，這些資料會向待命實例發出信號以嘗試繼續其啟動，以及從日誌登載回復佇

列所花費的時間，以便實例能夠重新開始處理訊息。 這些其他延遲來源都會增加啟動待命實例所花費的時間。 切換時間總計由下列元件組成:

失敗偵測時間
NFS 釋放佇列管理程式資料鎖定所花費的時間，以及待命實例繼續其啟動程序所花費的時間。

傳送時間
如果是 HA 叢集，則為 IBM i 將 IASP 從管理作用中實例的系統傳送至待命實例所花費的時間; 如果是日誌登載抄寫，則為使用來自遠端抄本的資料更新待命資料庫的本端日誌登載所花費的時間。

重新啟動時間
新作用中佇列管理程式實例從其還原日誌中的最新檢查點重建其佇列及回復處理訊息所花費的時間。

註:

如果已接管的待命實例配置為同步抄寫至先前作用中的實例，則啟動可能會延遲。 如果遠端日誌位於管理先前作用中實例的伺服器上，且伺服器失敗，則新的已啟動實例可能無法抄寫至其遠端日誌登載。

等待同步回應的預設時間是一分鐘。 您可以配置抄寫逾時之前的延遲上限。 或者，您可以配置待命實例，以開始使用非同步抄寫至失敗的作用中實例。 稍後，當失敗實例再次在待命上執行時，您會將切換至同步抄寫。 相同的考量也適用於使用同步獨立 ASP 鏡映。

您可以針對這些元件進行個別基準線測量，以協助您評量失效接手的整體時間，並將要使用的配置方法納入決策中。 在做出最佳配置決策時，您也需要考量相同伺服器上的其他應用程式如何進行失效接手，以及是否有備份或災難回復處理程序已使用 IASP。

透過調整叢集配置，可以縮短 IASP 傳送時間:

1. 叢集中跨系統的使用者設定檔應該具有相同的 GID 和 UID ，而不需要轉接處理程序來變更 UID 和 GID。
2. 將系統及基本使用者磁碟儲存區中的資料庫物件數目縮至最小，因為需要合併這些物件，以建立磁碟儲存區群組的交互參照表格。
3. 如需進一步的效能提示，請參閱 IBM 紅皮書: *Implementing PowerHA for IBM i(SG24-7405*)。

使用基本 ASP、異動日誌鏡映及小型配置的配置應該以數十秒的時間順序切換。

IBM i 結合 *IBM i* 叢集功能與 *IBM MQ* 叢集作業的概觀

在 IBM i 上執行 IBM MQ ，並利用 IBM i 叢集作業功能，可以提供比僅使用 IBM MQ 叢集作業更完整的「高可用性」解決方案。

若要具有此功能，您需要設定:

1. IBM i 機器上的叢集; 請參閱 第 327 頁的『IBM i 叢集』
2. 您將佇列管理程式移至其中的獨立輔助儲存區 (IASP); 請參閱 第 327 頁的『獨立輔助儲存區 (IASP)』
3. 叢集資源群組 (CRG); 請參閱 第 328 頁的『裝置叢集資源群組』，您在其中定義:
   - 回復網域
   - IASP
   - 結束程式; 請參閱 第 328 頁的『裝置 CRG 跳出程式』

IBM i 叢集

IBM i 叢集是邏輯上鏈結在一起的實例 (即 IBM i 電腦或分割區) 集合。

此分組的目的是容許備份每一個實例，消除單一失敗點，並增加應用程式及資料備援。 建立叢集之後，可以配置各種叢集資源群組 (CRG) 類型來管理叢集中的應用程式、資料及裝置。

如需進一步資訊，請參閱 建立叢集 及 建立叢集 (CRTCLU) 指令。

獨立輔助儲存區 (IASP)

IASP 是一種使用者 ASP 類型，可作為單一層次儲存體的延伸。 它是儲存體的一部分，由於其獨立於系統儲存體，因此可以輕鬆地操作，而不需要對系統進行 IPL。

IASP 可以輕鬆切換至另一個作業系統實例，或抄寫至另一個作業系統實例上的目標 IASP。 可以使用兩種方法在實例之間切換 IASP:

- 第一種方法需要使用「高速鏈結 (HSL)」迴圈來連接叢集中的所有電腦，以及包含 IASP 的切換式磁碟直立式主機。
- 第二個方法要求作業系統實例必須是可在分割區之間切換輸入/輸出處理器 (IOP) 之相同 IBM i 電腦上的分割區。 不需要特殊硬體即可抄寫 IASP。 透過網路使用 TCP/IP 執行抄寫。

如需相關資訊，請參閱 配置裝置 ASP (CFGDEVASP) 指令。

裝置叢集資源群組

叢集資源群組 (CRG) 有多種類型。 如需不同可用 CRG 類型的相關資訊，請參閱 叢集資源群組。

本主題集中於裝置 CRG。 裝置 CRG:

- 說明及管理裝置資源，例如獨立輔助儲存區 (IASP)。
- 定義叢集節點的回復網域
- 指派裝置，以及
- 指派將處理叢集事件的結束程式。

回復網域表示將哪些叢集節點視為主要節點。 其餘節點會被視為備份。 備份節點也會在回復網域中排序，指定哪個節點是第一個備份、第二個備份等等，視回復網域中有多少節點而定。

如果主要節點失敗，跳出程式會在回復網域中的所有節點上執行。 然後，在第一個備份上執行的跳出程式可以進行必要的起始設定，以讓此節點成為新的主要節點。

如需相關資訊，請參閱 建立裝置 CRG 及 建立叢集資源群組 (CRTCRG) 指令。

裝置 CRG 跳出程式

當回復網域定義的其中一個節點中發生事件時，作業系統叢集資源服務會呼叫裝置 CRG 跳出程式; 例如，失效接手或切換事件。

當叢集的主要節點失敗且 CRG 與它們所管理的所有資源一起切換時，會發生失效接手事件，而當特定的 CRG 從主要節點手動切換至備份節點時，會發生切換事件。

不論任何一種方式，跳出程式都會負責起始設定及啟動在前一個主要節點上執行的所有程式，這會將第一個備份節點轉換成新的主要節點。

例如，使用 IBM MQ 時，結束程式應該負責啟動 IBM MQ 子系統 (QMQM) 及佇列管理程式。 佇列管理程式應該配置成自動啟動接聽器和服務，例如觸發監視器。

IBM i 上可從 IBM MQ 9.1 取得範例結束程式 AMQSCRG4。

切換式 IASP 配置

IBM MQ 可以設定為利用 IBM i 的叢集功能。 若要執行此作業:

1. 在資料中心系統之間建立 IBM i 叢集
2. 將佇列管理程式移至 IASP。

   第 329 頁的『將佇列管理程式移至獨立輔助儲存區或從獨立輔助儲存區移除佇列管理程式』 包含一些範例程式碼，可協助您執行這項作業。
3. 您需要建立 CRG ，以定義回復網域、IASP 及跳出程式。

   第 329 頁的『配置裝置叢集資源群組』 包含一些範例程式碼，可協助您執行這項作業。

相關概念

第 345 頁的『獨立 ASP 及高可用性』
獨立 ASP 可讓應用程式及資料在伺服器之間移動。 獨立 ASP 的彈性表示它們是部分 IBM i 高可用性解決方案的基礎。 在考量對佇列管理程式異動日誌使用 ASP 或獨立 ASP 時，您應該考量根據獨立 ASP 的其他高可用性配置。

IBM i 配置裝置叢集資源群組

設定裝置叢集資源群組 (CRG) 的範例程式。

關於這項作業

在下列範例中，請注意:

- [PRIMARY SITE NAME] 和 [BACKUP SITE NAME] 可以是八個字元或更少的任何兩個不同字串。
- [PRIMARY IP] 和 [BACKUP IP] 是用於鏡映的 IP。

程序

1. 識別叢集的名稱。
2. 識別 CRG 跳出程式名稱及檔案庫。
3. 決定此 CRG 要定義的主要節點及備份節點名稱。
4. 識別要由此 CRG 管理的 IASP，並確定已在主要節點下建立它。
5. 使用下列指令在備份節點中建立裝置說明:

```
CRTDEVASP DEVD([IASP NAME]) RSRCNAME([IASP NAME])
```

6. 使用下列指令，將接管 IP 位址新增至所有節點:

```
ADDTCPIFC INTNETADR(' [TAKEOVER IP]') LIND([LINE DESC])
SUBNETMASK('[SUBNET MASK]') AUTOSTART(*NO)
```

7. 使用下列指令，僅在主要節點中啟動接管 IP 位址:

```
STRTCPIFC INTNETADR('[TAKEOVER IP')
```

8. 選擇性的: 如果您的 IASP 是可切換的，請呼叫此指令:

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG( [CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT
NAME])
USRPRF([EXIT PROFILE]) RCYDMN(( [PRIMARY NODE] *PRIMARY) ([BACKUP NAME] *BACKUP))
EXITPGMFMT(EXTP0200) CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]')
```

9. 選擇性的: 如果要鏡映 IASP，請呼叫下列指令:

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG([CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT NAME])
USRPRF([EXIT PROFILE]) RCYDMN(([PRIMARY NODE] *PRIMARY *LAST [PRIMARY SITE NAME] ('[PRIMARY
IP]'))
[BACKUP NAME] *BACKUP *LAST [BACKUP SITE NAME] ('[BACKUP IP]'))) EXITPGMFMT(EXTP0200)
CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]'))
```

IBM i 將佇列管理程式移至獨立輔助儲存區或從獨立輔助儲存區移除佇列管理程式

將佇列管理程式移至獨立輔助儲存區 (IASP) 的範例程式，以及從 IASP 移除佇列管理程式的指令。

關於這項作業

在下列範例中，請注意:

- [MANAGER NAME] 是佇列管理程式的名稱。
- [IASP NAME] 是 IASP 的名稱。
- [MANAGER LIBRARY] 是佇列管理程式檔案庫的名稱。
- [MANAGER DIRECTORY] 是佇列管理程式目錄的名稱。

程序

1. 識別您的主要節點和備份節點。
2. 在主要節點上執行下列程序:

a) 請確定佇列管理程式已結束。

b) 使用指令，確定您的 IASP 是 vary on

```
VRYCFG CFGOBJ([IASP NAME]) CFGTYPE(*DEV) STATUS(*ON)
```

c) 在 IASP 下建立佇列管理程式目錄。
根目錄下將會有一個具有 IASP 名稱的目錄，即:

```
QSH CMD('mkdir -p /[IASP_NAME]/QIBM/UserData/mqm/qmgrs/')
```

d) 使用下列指令，將管理程式的 IFS 物件移至您剛在 IASP 下建立的佇列管理程式目錄:

```
QSH CMD('mv /QIBM/UserData/mqm/qmgrs/[MANAGER NAME]
/[IASP NAME]/QIBM/UserData/mqm/qmgrs')
```

e) 使用下列指令，建立名為 MGRLIB 的暫時儲存檔:

```
CRTSAVF QGPL/MGRLIB
```

f) 使用下列指令，將佇列管理程式檔案庫儲存至 MGRLIB 儲存檔:

```
SAVLIB LIB([MANGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
```

g) 使用下列指令來刪除佇列管理程式檔案庫，並忽略所有查詢訊息:

```
DLTLIB [MANAGER LIBRARY]
```

h) 使用下列指令，將佇列管理程式檔案庫還原至 IASP:

```

RSTLIB SAVLIB([MANAGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
RSTASPDEV([IASP NAME])
```

i) 使用下列指令來刪除暫時儲存檔:

```
 DLTF FILE(QGPL/MGRLIB)
```

j) 使用下列指令，建立 IASP 下佇列管理程式 IFS 物件的符號鏈結:

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

k) 使用下列指令連接至 IASP:

```
SETASPGRP [IASP NAME]
```

l) 使用下列指令來啟動佇列管理程式:

```
STRMQM [MANAGER NAME]
```

3. 在一或多個備份節點上執行下列程序:

a) 使用下列指令來建立暫時佇列管理程式目錄:

```
QSH CMD('mkdir -p /[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

b) 使用下列指令，建立佇列管理程式暫存目錄的符號鏈結:

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

c) 使用下列指令來刪除暫存目錄:

```
QSH CMD('rm -r /[IASP NAME]')
```

d) 在檔案 /QIBM/UserData/mqm/mqs.ini 結尾新增下列:

```
QueueManager:
Name=[MANAGER NAME]
Prefix=/QIBM/UserData/mqm
Library=[MANAGER LIBRARY]
Directory=[MANAGER DIRECTORY]
```

4. 若要從 IASP 移除佇列管理程式，請發出下列指令:
   a) VRYCFG CFGOBJ ([IASP NAME]) CFGTYPE (*DEV) STATUS (*ON)
   b) SETASPGRP [IASP 名稱]
   c) ENDMQM [MANAGER 名稱]
   d) DLTMQM [MANAGER 名稱]

IBM i *IBM i 上 ASP 的鏡映異動日誌配置*

使用鏡映日誌登載之間的同步抄寫來配置健全的多重實例佇列管理程式。

鏡映佇列管理程式配置使用在基本或獨立輔助儲存區 (ASP) 中建立的異動日誌。

在 IBM i 上，佇列管理程式資料會寫入日誌登載及檔案系統。 日誌登載包含佇列管理程式資料的正本。 在系統之間使用同步或非同步日誌登載抄寫來共用日誌登載。 需要混合本端及遠端異動日誌，才能重新啟動佇列管理程式實例。 佇列管理程式重新啟動會從伺服器上本端及遠端異動日誌的混合，以及共用網路檔案系統上的佇列管理程式資料中讀取日誌登載記錄。 檔案系統中的資料可加速重新啟動佇列管理程式。 檢查點儲存在檔案系統中，標示檔案系統與日誌登載之間的同步點。 一般佇列管理程式重新啟動時，不需要在檢查點之前儲存的日誌登載記錄。 不過，檔案系統中的資料可能不是最新的，且會使用檢查點之後的日誌登載記錄來完成佇列管理程式重新啟動。 連接至實例的日誌登載中的資料會保持最新，以便重新啟動可以順利完成。

但如果正在非同步抄寫待命伺服器上的遠端日誌，且在同步化之前發生失敗，則即使日誌登載記錄也可能不是最新的。 如果您決定使用未同步的遠端日誌登載來重新啟動佇列管理程式，待命佇列管理程式實例可能會重新處理在作用中實例失敗之前已刪除的訊息，或不處理在作用中實例失敗之前收到的訊息。

另一種罕見的可能性是檔案系統包含最新的檢查點記錄，而待命資料庫上未同步的遠端異動日誌則不包含。 在此情況下，佇列管理程式不會自動重新啟動。 您可以選擇等待遠端異動日誌同步化，或從檔案系統冷啟動待命佇列管理程式。 即使在此情況下，檔案系統包含比遠端異動日誌更新的佇列管理程式資料檢查點，它可能不會包含在作用中實例失敗之前處理的所有訊息。 在與日誌登載不同步的冷重新啟動之後，部分訊息可能重新處理，部分未處理。

使用多重實例佇列管理程式時，也會使用檔案系統來控制佇列管理程式的哪些實例處於作用中狀態，以及哪些實例處於待命狀態。 作用中實例會獲得佇列管理程式資料的鎖定。 待命資料庫會等待獲得鎖定，當它獲得鎖定時，它會變成作用中實例。 如果作用中實例正常結束，則會釋放鎖定。 如果檔案系統偵測到作用中實例失敗，或無法存取檔案系統，則檔案系統會釋放鎖定。 檔案系統必須符合偵測失敗的需求; 請參閱 共用檔案系統的需求。

IBM i 上多重實例佇列管理程式的架構提供在伺服器或佇列管理程式失敗之後自動重新啟動。 它也支援在儲存佇列管理程式資料的檔案系統失敗之後還原佇列管理程式資料。

在 第 332 頁的圖 27 中，如果 ALPHA 失敗，您可以使用鏡映日誌登載在 BETA 上手動重新啟動 QM1 。 透過將多重實例佇列管理程式功能新增至 QM1，如果 ALPHA 上的作用中實例失敗，則 QM1 的待命實例會自動在測試版上回復。 如果伺服器 ALPHA 失敗，則 QM1 也可以自動回復，而不只是 QM1 的作用中實例。 一旦測試版變成作用中佇列管理程式實例的主機，即可在 ALPHA 上啟動待命實例。

第 332 頁的圖 27 顯示使用 NetServer 來鏡映佇列管理程式兩個實例之間的日誌登載，以儲存佇列管理程式資料的配置。 您可以展開型樣以包括更多日誌登載，從而包括更多實例。 遵循 第 315 頁的『IBM i 上的佇列管理程式日誌』主題中說明的日誌登載命名規則。 目前佇列管理程式的執行中實例數限制為兩個，一個在作用中，另一個在待命中。

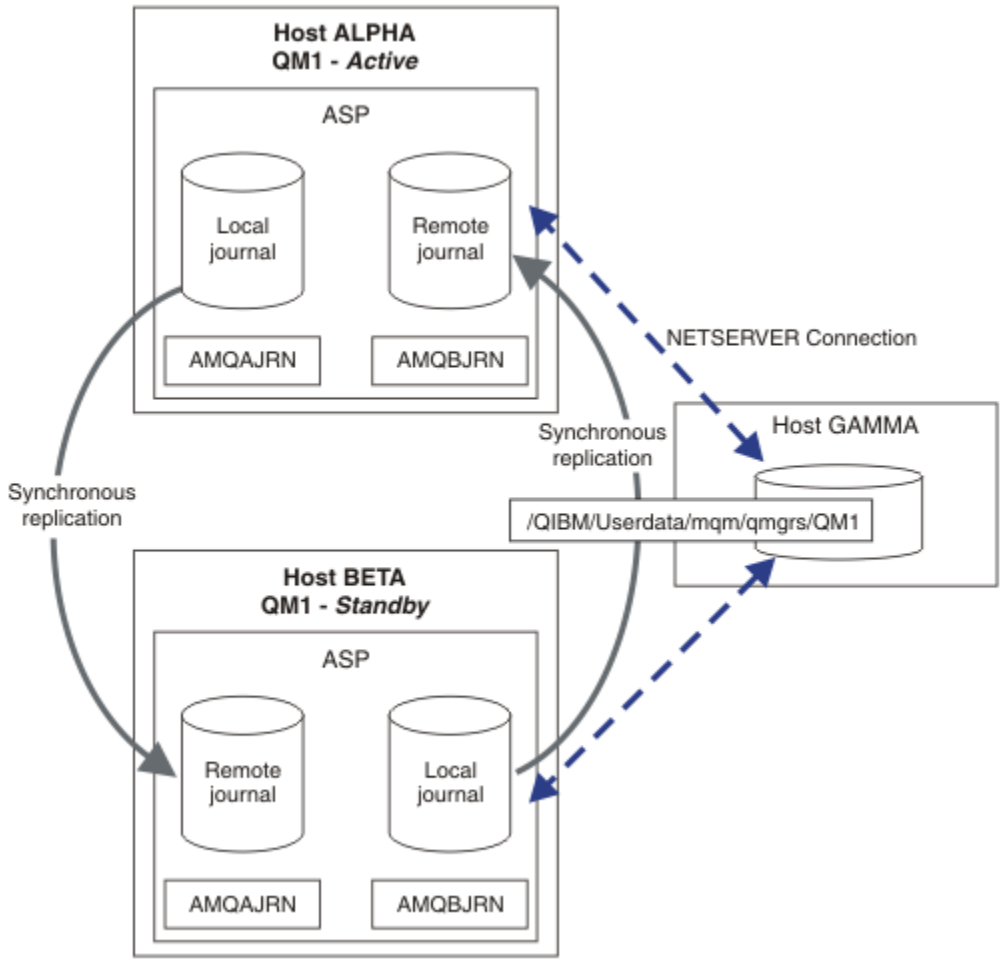


圖 *27:* 鏡映佇列管理程式日誌登載

主機 ALPHA 上 QM1 的本端日誌稱為 AMQAJRN (或更完整地，稱為 QMQM1/AMQAJRN)，在測試版上，日誌登載為 QMQM1/AMQBJRN。 每一個本端日誌都會抄寫至佇列管理程式所有其他實例上的遠端日誌登載。 如果佇列管理程式已配置兩個實例，則會將本端日誌登載抄寫至一個遠端日誌登載。

*SYNC 或 *ASYNC 遠端異動日誌抄寫

使用同步 (*SYNC) 來鏡映 IBM i 異動日誌 或非同步 (*ASYNC) 日誌登載; 請參閱 遠端異動日誌管理。

第 332 頁的圖 27 中的抄寫模式是 *SYNC，而不是 *ASYNC。 *ASYNC 更快，但如果遠端異動日誌狀態為 *ASYNCPEND 時發生失敗，則本端與遠端異動日誌不一致。 遠端異動日誌必須趕上本端異動日誌。 如果您選擇 *SYNC，則本端系統會等待遠端異動日誌，然後再從需要完成寫入的呼叫返回。 本端及遠端異動日誌通常會彼此保持一致。 僅當 *SYNC 作業花費的時間超過指定的時間時 [1]，且遠端日誌登載已取消啟動，請執行異動日誌不同步。 錯誤會記載至日誌登載訊息佇列及 QSYSOPR。 佇列管理程式會偵測到此訊息，將錯誤寫入佇列管理程式錯誤日誌，並取消啟動佇列管理程式日誌登載的遠端抄寫。 作用中佇列管理程式實例會回復，而不會遠端登載至此日誌登載。 當遠端伺服器再次可用時，您必須手動重新啟動同步遠端異動日誌抄寫。 然後會重新同步化日誌。

第 332 頁的圖 27 中說明的 *SYNC / *SYNC 配置的問題是 BETA 上待命佇列管理程式實例如何控制。 只要 BETA 上的佇列管理程式實例寫入其第一個持續訊息，它就會嘗試更新 ALPHA 上的遠端日誌登載。 如果控制從 ALPHA 傳遞至 BETA 的原因是 ALPHA 失敗，且 ALPHA 仍關閉，則 ALPHA 的遠端日誌登載會失敗。 BETA 會等待 ALPHA 回應，然後取消啟動遠端日誌登載，並僅使用本端日誌登載回復處理訊息。 BETA 必須等待一段時間，才能偵測到 ALPHA 已關閉，導致閒置一段時間。

[1] 在 IBM i 5 上指定的時間是 60 秒，在 IBM i 6.1 之後的範圍是 1-3600 秒。

選擇將遠端日誌登載設定為 *SYNC 或 *ASYNC 是取捨。 第 333 頁的表 24 彙總在一對佇列管理程式之間使用 *SYNC 與 *ASYNC 日誌登載之間的取捨:

表 *24:* 遠端日誌登載選項

作用中	待用	***SYNC**	***ASYNC**
***SYNC**		1. 一致切換及失效接手 2. 失效接手之後，待命實例不會立即回復。 3. 遠端日誌登載必須隨時可用 4. 佇列管理程式效能取決於遠端日誌登載	1. 一致切換及失效接手 2. 當待命伺服器可用時，遠端日誌登載必須切換至 *SYNC 3. 遠端日誌登載在重新啟動之後必須保持可用 4. 佇列管理程式效能取決於遠端日誌登載
***ASYNC**		1. 不是明智的組合	1. 在失效接手或切換之後，部分訊息可能遺失或重複 2. 待命實例不需要一直可用，作用中實例才能繼續而不延遲。 3. 效能與遠端日誌登載無關

***SYNC / *SYNC**

作用中佇列管理程式實例會使用 *SYNC 日誌登載，當待命佇列管理程式實例啟動時，它會立即嘗試使用 *SYNC 日誌登載。

1. 遠端異動日誌在交易上與作用中佇列管理程式的本端異動日誌一致。 如果佇列管理程式切換至待命實例，則可以立即回復。 待命實例通常會回復，而不會遺失或複製任何訊息。 只有在自前次檢查點以來遠端日誌登載失敗，且無法重新啟動先前作用中的佇列管理程式時，才會遺失或複製訊息。
2. 如果佇列管理程式失效接手至待命實例，則可能無法立即啟動。 使用 *SYNC 日誌登載來啟動待命佇列管理程式實例。 失效接手的原因可能阻止遠端日誌登載至管理待命實例的伺服器。 在處理任何持續訊息之前，佇列管理程式會等待直到偵測到問題為止。 錯誤會記載至日誌登載訊息佇列及 QSYSOPR。 佇列管理程式會偵測到此訊息，將錯誤寫入佇列管理程式錯誤日誌，並取消啟動佇列管理程式日誌登載的遠端抄寫。 作用中佇列管理程式實例會回復，而不會遠端登載至此日誌登載。 當遠端伺服器再次可用時，您必須手動重新啟動同步遠端異動日誌抄寫。 然後會重新同步化日誌。
3. 抄寫遠端異動日誌的目標伺服器必須一律可用，才能維護遠端異動日誌。 遠端日誌通常會抄寫至管理待命佇列管理程式的相同伺服器。 伺服器可能變成無法使用。 錯誤會記載至日誌登載訊息佇列及 QSYSOPR。 佇列管理程式會偵測到此訊息，將錯誤寫入佇列管理程式錯誤日誌，並取消啟動佇列管理程式日誌登載的遠端抄寫。 作用中佇列管理程式實例會回復，而不會遠端登載至此日誌登載。 當遠端伺服器再次可用時，您必須手動重新啟動同步遠端異動日誌抄寫。 然後會重新同步化日誌。
4. 遠端日誌登載比本端日誌登載更慢，而且如果伺服器相距很遠，則會更慢。 佇列管理程式必須等待遠端日誌登載，這會降低佇列管理程式效能。

一對伺服器之間的 *SYNC / *SYNC 配置在失效接手之後回復待命實例時具有延遲的缺點。 *SYNC / *ASYNC 配置沒有這個問題。

*SYNC / *SYNC 會保證在切換或失效接手之後，只要遠端異動日誌可用，就不會遺失任何訊息。 如果您想要減少失效接手或切換之後訊息遺失的風險，您有兩個選擇。 如果遠端異動日誌變成非作用中，請停止作用中實例，或在多部伺服器上建立遠端異動日誌。

***SYNC / *ASYNC**

作用中佇列管理程式實例使用 *SYNC 日誌登載，當待命佇列管理程式實例啟動時，它會使用 *ASYNC 日誌登載。 在管理新待命實例的伺服器變成可用之後不久，系統操作員必須將作用中實例上的遠端異動日誌切換至 *SYNC。 當操作員將遠端日誌登載從 *ASYNC 切換至 *SYNC 時，如果遠端日誌登載的狀態為 *ASYNCPEND，則作用中實例會暫停。 作用中佇列管理程式實例會等到剩餘的異動日誌項目傳送至遠端異動日誌。 當遠端異動日誌已與本端異動日誌同步時，新的待命資料庫會再次與新的作用中實例交易一致。 從多重實例佇列管理程式管理的角度來看，在 *SYNC / *ASYNC 配置中， IBM i 系統操作員還有一

項額外作業。 除了重新啟動失敗的佇列管理程式實例之外，操作員還必須將遠端日誌登載切換至 *SYNC 。

1. 遠端異動日誌在交易上與作用中佇列管理程式的本端異動日誌一致。 如果作用中佇列管理程式實例已切換或失效接手至待命實例，則待命實例可以立即回復。 待命實例通常會回復，而不會遺失或複製任何訊息。 只有在自前次檢查點以來遠端日誌登載失敗，且無法重新啟動先前作用中的佇列管理程式時，才會遺失或複製訊息。
2. 在管理作用中實例的系統重新變成可用之後，系統操作員必須立即將遠端異動日誌從 *ASYNC 切換至 *SYNC 。 在將遠端異動日誌切換至 *SYNC 之前，操作員可能等待遠端異動日誌擷取並回應。 或者，操作員可以立即將遠端實例切換至 *SYNC ，並強制作用中實例等待直到待命實例日誌登載已捕捉到為止。 當遠端日誌登載設為 *SYNC 時，待命實例通常與作用中實例在交易上一致。 只有在自前次檢查點以來遠端日誌登載失敗，且無法重新啟動先前作用中的佇列管理程式時，才會遺失或複製訊息。
3. 從切換或失效接手還原配置時，管理遠端異動日誌的伺服器必須隨時可用。

當您想要在失效接手之後快速回復待命佇列管理程式時，請選擇 *SYNC / *ASYNC 。 您必須手動將新作用中實例上的遠端異動日誌設定還原為 *SYNC 。 *SYNC / *ASYNC 配置符合管理一對多重實例佇列管理程式的一般型樣。 在一個實例失敗之後，有一段時間會重新啟動待命實例，在此期間作用中實例無法失效接手。

***ASYNC / *ASYNC**
同時管理作用中及待命佇列管理程式的伺服器都配置為使用 *ASYNC 遠端日誌登載。

1. 發生切換或失效接手時，佇列管理程式會繼續使用新伺服器上的日誌登載。 發生切換或失效接手時，日誌登載可能未同步。 因此，訊息可能遺失或重複。
2. 即使管理待命佇列管理程式的伺服器無法使用，作用中實例仍會執行。 當待命伺服器可用時，會以非同步方式抄寫本端日誌登載。
3. 遠端日誌登載不會影響本端佇列管理程式的效能。

如果效能是您的主體需求，且您準備在失效接手或切換之後遺失或複製部分訊息，請選擇 *ASYNC / *ASYNC 。

***ASYNC / *SYNC**
沒有理由使用此選項組合。

從遠端異動日誌啟動佇列管理程式

日誌登載會同步或非同步抄寫。 遠端異動日誌可能不在作用中，或它可能正在與本端異動日誌擷取並更新。 遠端異動日誌可能正在迎頭趕上，即使它是同步抄寫的，因為它最近可能已啟動。 佇列管理程式在啟動期間套用至其使用之遠端異動日誌狀態的規則如下。

1. 如果待命啟動必須從待命上的遠端異動日誌重播，且日誌狀態為 *FAILED 或 *INACTPEND，則待命啟動會失敗。
2. 當開始啟動待命資料庫時，待命資料庫上的遠端異動日誌狀態必須是 *ACTIVE 或 *INACTIVE。 如果狀態為 *INACTIVE，則如果尚未抄寫所有日誌資料，則啟動可能會失敗。

   如果網路檔案系統上佇列管理程式資料的檢查點記錄比遠端異動日誌中的檢查點記錄還新，則會發生失敗。 只要遠端異動日誌在檢查點之間的預設 30 分鐘間隔上限內順利啟動，就不可能發生失敗。 如果待命佇列管理程式從檔案系統讀取最近的檢查點記錄，則不會啟動。

   您可以選擇: 等到可以還原作用中伺服器上的本端日誌登載，或冷啟動待命佇列管理程式。 如果您選擇冷啟動，則佇列管理程式會在沒有日誌登載資料的情況下啟動，並依賴於檔案系統中佇列管理程式資料的一致性及完整性。

   註: 如果您冷啟動佇列管理程式，則會有在前次檢查點之後遺失或複製訊息的風險。 訊息交易已寫入日誌登載，但部分交易可能尚未寫入檔案系統中的佇列管理程式資料。 當您冷啟動佇列管理程式時，會啟動全新日誌登載，且會遺失未寫入檔案系統中佇列管理程式資料的交易。
3. 待命佇列管理程式啟動會等待待命佇列上的遠端異動日誌狀態從 *ASYNCPEND 或 *SYNCPEND 變更為 *ASYNC 或 *SYNC。 訊息會定期寫入執行控制器的工作日誌。

   註: 在此情況下，啟動會在正在啟動之待命佇列管理程式的本端遠端異動日誌上等待。 在沒有遠端異動日誌的情況下繼續之前，佇列管理程式也會等待一段時間。 當它嘗試同步寫入遠端異動日誌 (或異動日誌) 且異動日誌無法使用時，它會等待。

4. 如果異動日誌狀態變更為 *FAILED 或 *INACTPEND，則啟動會停止。

要在啟動中使用的本端及遠端異動日誌的名稱及狀態會寫入佇列管理程式錯誤日誌。

IBM i 在 *IBM i* 上使用日誌登載鏡映及 *NetServer* 建立多重實例佇列管理程式

建立要在兩部 IBM i 伺服器上執行的多重實例佇列管理程式。 佇列管理程式資料會使用 NetServer 儲存在第三部 IBM i 伺服器上。 使用遠端日誌登載在兩部伺服器之間鏡映佇列管理程式異動日誌。 **ADDMQMJRN** 指令可用來簡化建立遠端異動日誌。

開始之前

1. 此作業需要三部 IBM i 伺服器。 在其中兩個上安裝 IBM MQ : 範例中的 ALPHA 和 BETA。 產品必須至少為 IBM WebSphere MQ 7.0.1 Fix Pack 1。
2. 第三部伺服器是 IBM i 伺服器，由 NetServer 連接至 ALPHA 及 BETA。 它用來共用佇列管理程式資料。 它不需要有 IBM MQ 安裝架構。 作為暫時步驟，在伺服器上安裝 IBM MQ 有助於設定佇列管理程式目錄及許可權。
3. 請確定 QMQM 使用者設定檔在這三部伺服器上都有相同的密碼。
4. 安裝 IBM i NetServer;請參閱 i5/OS NetServer 。

關於這項作業

請執行下列步驟來建立 第 337 頁的圖 28 中所示的配置。 佇列管理程式資料是使用 IBM i NetServer 來連接。

- 在要儲存佇列管理程式資料的 GAMMA 上建立從 ALPHA 及 BETA 到目錄共用的連線。 此作業也會設定必要的許可權、使用者設定檔及密碼。
- 將「關聯式資料庫項目 (RDBE)」新增至即將執行佇列管理程式實例的 IBM i 系統。 RDBE 項目是用來連接用於遠端日誌登載的 IBM i 系統。
- 在 IBM i 伺服器 Alpha 上建立佇列管理程式 QM1 。
- 在其他 IBM i 伺服器 BETA 上新增 QM1 的佇列管理程式控制資訊。
- 在兩個佇列管理程式實例的兩個 IBM i 伺服器上建立遠端日誌登載。 每一個佇列管理程式都會寫入本端日誌登載。 本端異動日誌會複製到遠端異動日誌。 **ADDMQMJRN** 指令可簡化新增日誌登載及連線。
- 啟動佇列管理程式，允許待命實例。

程序

1. 執行作業 第 324 頁的『在 IBM i 上使用 NetServer 建立佇列管理程式資料的網路共用』。

   因此， ALPHA 和 BETA 具有一個共用 /QNTC/GAMMA/WMQ，指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 。 使用者設定檔 QMQM 和 QMQMADM 具有必要的許可權，且 QMQM 在所有三個系統上都具有相符的密碼。
2. 將「關聯式資料庫項目 (RDBE)」新增至將要管理佇列管理程式實例的 IBM i 系統。

   a) 在 ALPHA 上建立 BETA 的連線。

   ```
   ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
   ```

   b) 在測試版上，建立與 ALPHA 的連線。

   ```
   ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
   ```
3. 在 ALPHA 上建立佇列管理程式 QM1 ， 並將佇列管理程式資料儲存在 GAMMA 上。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   路徑會使用 NetServer ， 在中建立佇列管理程式資料。

4. 在 ALPHA 上執行。 此指令會在的 BETA 上新增遠端異動日誌。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
```

當的作用中實例位於 ALPHA 上時，會在 ALPHA 的本端異動日誌中建立異動日誌項目。 Alpha 上的本端異動日誌會複製到 BETA 上的遠端異動日誌。

5. 使用指令來檢查 Alpha 上針對所建立的 IBM MQ 配置資料。

下一步需要此資訊。

在此範例中，在的 ALPHA 上建立下列配置:

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QMQM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

6. 使用指令在測試版上建立 QM1 的佇列管理程式實例。 在 BETA 上執行下列指令，以在 BETA 上修改佇列管理程式控制資訊。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

提示: 複製並貼上配置資訊。 Alpha 和 BETA 上的佇列管理程式段落相同。

7. 在測試版上執行。 此指令會在 BETA 上新增本端異動日誌，並在 ALPHA 上新增遠端異動日誌。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
```

當的作用中實例在 BETA 上時，會在其在 BETA 上的本端異動日誌中建立異動日誌項目。 BETA 上的本端異動日誌會複製到 ALPHA 上的遠端異動日誌。

註: 作為替代方案，您可能想要使用非同步日誌登載來設定從 BETA 到 ALPHA 的遠端日誌登載。

使用此指令來設定從 BETA 到 ALPHA 的非同步日誌登載，而不是步驟 第 336 頁的『7』中的指令。

```
ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
```

如果 ALPHA 上的伺服器或日誌登載是失敗的來源，則 BETA 會在不等待新的異動日誌項目抄寫至 ALPHA 的情況下啟動。

當 ALPHA 再次連線時，使用指令將抄寫模式切換至 *SYNC。

使用 第 331 頁的『IBM i 上 ASP 的鏡映異動日誌配置』 中的資訊來決定是同步、非同步或兩者的混合鏡映日誌。 預設值是同步抄寫，等待來自遠端異動日誌的回應 60 秒。

8. 請驗證 ALPHA 及 BETA 上的異動日誌已啟用，且遠端異動日誌抄寫的狀態為。

a) 在 ALPHA 上:

```
WRKMQMJRN MQMNAME(QM1)
```

b) 在測試版上:

```
WRKMQMJRN MQMNAME(QM1)
```

9. 在 ALPHA 和 BETA 上啟動佇列管理程式實例。

a) 在 ALPHA 上啟動第一個實例，使它成為作用中實例。 啟用切換至待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

b) 在 BETA 上啟動第二個實例，使它成為待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

結果

用來檢查佇列管理程式狀態:

1. Alpha 上佇列管理程式實例的狀態應該為。
2. BETA 上佇列管理程式實例的狀態應該為。

範例

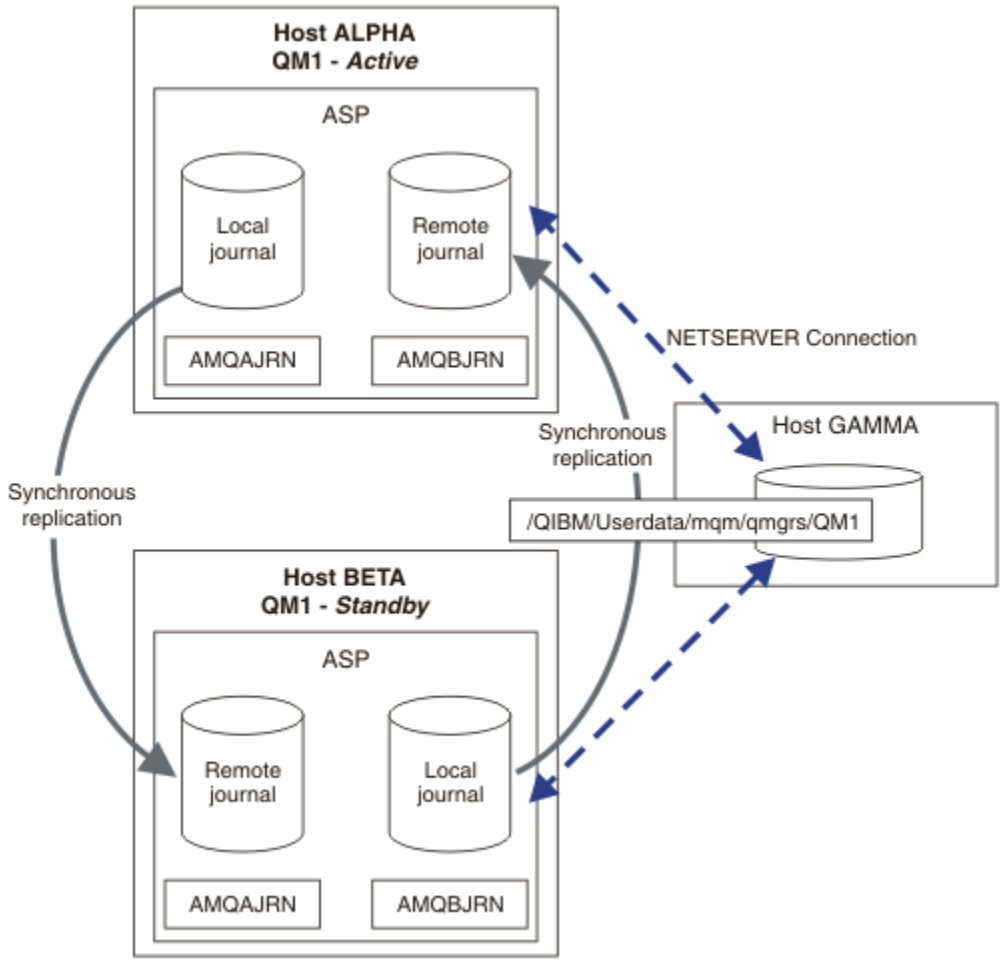


圖 *28:* 鏡映異動日誌配置

下一步

- 驗證作用中及待命實例自動切換。 您可以執行範例高可用性範例程式來測試切換; 請參閱 高可用性範例程式。 範例程式是 ' C' 用戶端。 您可以從 Windows 或 Unix 平台執行它們。
  1. 啟動高可用性範例程式。
  2. 在 ALPHA 上，結束要求切換的佇列管理程式:

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. 請檢查 BETA 上的實例是否處於作用中。
  4. 在 ALPHA 上重新啟動

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 查看替代高可用性配置:
  1. 使用 NetServer，將佇列管理程式資料放置在 Windows 伺服器上。
  2. 不使用遠端日誌登載來鏡映佇列管理程式異動日誌，而是將異動日誌儲存在獨立 ASP 上。 使用 IBM i 叢集作業，將獨立 ASP 從 ALPHA 傳送至 BETA。

IBM i 在 *IBM i* 上使用 *NetServer* 及日誌登載鏡映將單一實例佇列管理程式轉換為多重實例佇列管理程式

將單一實例佇列管理程式轉換為多重實例佇列管理程式。 將佇列管理程式資料移至 NetServer 所連接的網路共用。 使用遠端日誌登載，將佇列管理程式日誌登載鏡映至第二部 IBM i 伺服器。

開始之前

1. 此作業需要三部 IBM i 伺服器。 範例中伺服器 ALPHA 上的現有 IBM MQ 安裝必須至少為 IBM WebSphere MQ 7.0.1 Fix Pack 1。 ALPHA 正在執行範例中稱為 QM1 的佇列管理程式。
2. 在第二部 IBM i 伺服器上安裝 IBM MQ，範例中為 BETA。
3. 第三部伺服器是 IBM i 伺服器，由 NetServer 連接至 ALPHA 及 BETA。 它用來共用佇列管理程式資料。 它不需要有 IBM MQ 安裝架構。 作為暫時步驟，在伺服器上安裝 IBM MQ 有助於設定佇列管理程式目錄及許可權。
4. 請確定 QMQM 使用者設定檔在這三部伺服器上都有相同的密碼。
5. 安裝 IBM i NetServer;請參閱 i5/OS NetServer 。

關於這項作業

執行下列步驟，將單一實例佇列管理程式轉換為 第 341 頁的圖 29 中顯示的多重實例佇列管理程式。 單一實例佇列管理程式會在作業中刪除，然後重建，並將佇列管理程式資料儲存在 NetServer 所連接的網路共用上。 此程序比使用 **CPY** 指令將佇列管理程式目錄及檔案移至網路共用更為可靠。

- 在要儲存佇列管理程式資料的 GAMMA 上建立從 ALPHA 及 BETA 到目錄共用的連線。 此作業也會設定必要的許可權、使用者設定檔及密碼。
- 將「關聯式資料庫項目 (RDBE)」新增至即將執行佇列管理程式實例的 IBM i 系統。 RDBE 項目是用來連接用於遠端日誌登載的 IBM i 系統。
- 儲存佇列管理程式日誌和定義，停止佇列管理程式，然後刪除它。
- 重建佇列管理程式，並在 GAMMA 上儲存網路共用上的佇列管理程式資料。
- 將佇列管理程式的第二個實例新增至另一部伺服器。
- 在兩個佇列管理程式實例的兩個 IBM i 伺服器上建立遠端日誌登載。 每一個佇列管理程式都會寫入本端日誌登載。 本端異動日誌會複製到遠端異動日誌。 **ADDMQMJRN** 指令可簡化新增日誌登載及連線。
- 啟動佇列管理程式，允許待命實例。

註:

在作業的步驟 第 339 頁的『4』 中，您刪除單一實例佇列管理程式 QM1。 刪除佇列管理程式會刪除佇列上的所有持續訊息。 因此，在轉換佇列管理程式之前，請先完成處理佇列管理程式所儲存的所有訊息。 如果無法處理所有訊息，請在步驟 第 339 頁的『4』之前備份佇列管理程式庫。 在步驟 第 339 頁的『5』之後還原佇列管理程式庫。

註:

在作業的步驟 第 339 頁的『5』 中，您重建 QM1。 雖然佇列管理程式具有相同的名稱，但它具有不同的佇列管理程式 ID。 佇列管理程式叢集作業會使用佇列管理程式 ID。 若要刪除並重建叢集中的佇列管理程式，您必須先從叢集中移除佇列管理程式; 請參閱 從叢集中移除佇列管理程式: 替代方法 或 從叢集中移除佇列管理程式。 當您重建佇列管理程式時，請將它新增至叢集。 雖然它的名稱與之前相同，但對叢集中其他佇列管理程式而言，它似乎是新的佇列管理程式。

程序

1. 執行作業 第 324 頁的『在 IBM i 上使用 NetServer 建立佇列管理程式資料的網路共用』。

因此， ALPHA 和 BETA 具有一個共用 /QNTC/GAMMA/WMQ，指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 。 使用者設定檔 QMQM 和 QMQMADM 具有必要的許可權，且 QMQM 在所有三個系統上都具有相符的密碼。

2. 將「關聯式資料庫項目 (RDBE)」新增至將要管理佇列管理程式實例的 IBM i 系統。
   a) 在 ALPHA 上建立 BETA 的連線。

   ```
   ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
   ```

   b) 在測試版上，建立與 ALPHA 的連線。

   ```
   ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
   ```

3. 建立 Script 以重建佇列管理程式物件。

   ```
   QSAVEQMGR LCLQMGRNAM(QM1) FILENAME('*CURLIB/QMQSC(QM1)')
   OUTPUT(*REPLACE) MAKEAUTH(*YES) AUTHFN('*CURLIB/QMAUT(QM1)')
   ```

4. 停止佇列管理程式並刪除它。

   ```
   ENDMQM MQMNAME(QM1) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*YES) TIMEOUT(15)
   DLTMQM MQMNAME(QM1)
   ```

5. 在 ALPHA 上建立佇列管理程式 QM1 ， 並將佇列管理程式資料儲存在 GAMMA 上。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   路徑會使用 NetServer ， 在中建立佇列管理程式資料。

6. 從儲存的定義中重建 QM1 的佇列管理程式物件。

   ```
   STRMQMMQSC SRCMBR(QM1) SRCFILE(*CURLIB/QMQSC) MQMNAME(QM1)
   ```

7. 從儲存的資訊套用授權。
   a) 編譯已儲存的授權程式。

   ```
   CRTCLPGM PGM(*CURLIB/QM1) SRCFILE(*CURLIB/QMAUT)
   SRCMBR(QM1) REPLACE(*YES)
   ```

   b) 執行程式以套用授權。

   ```
   CALL PGM(*CURLIB/QM1)
   ```

   c) 重新整理 QM1 的安全資訊。

   ```
   RFRMQMAUT MQMNAME(QM1)
   ```

8. 在 ALPHA 上執行。 此指令會在的 BETA 上新增遠端異動日誌。

   ```
   ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
   ```

   當的作用中實例位於 ALPHA 上時， 會在 ALPHA 的本端異動日誌中建立異動日誌項目。 Alpha 上的本端異動日誌會複製到 BETA 上的遠端異動日誌。

9. 使用指令來檢查 Alpha 上針對所建立的 IBM MQ 配置資料。

   下一步需要此資訊。

   在此範例中， 在的 ALPHA 上建立下列配置:

   ```
   Name=QM1
   Prefix=/QIBM/UserData/mqm
   Library=QMQM1
   Directory=QM1
   DataPath= /QNTC/GAMMA/WMQ /QM1
   ```

10. 使用指令在測試版上建立 QM1 的佇列管理程式實例。 在 BETA 上執行下列指令，以在 BETA 上修改佇列管理程式控制資訊。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

提示: 複製並貼上配置資訊。 Alpha 和 BETA 上的佇列管理程式段落相同。

11. 在測試版上執行。 此指令會在 BETA 上新增本端異動日誌，並在 ALPHA 上新增遠端異動日誌。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
```

當的作用中實例在 BETA 上時，會在其在 BETA 上的本端異動日誌中建立異動日誌項目。 BETA 上的本端異動日誌會複製到 ALPHA 上的遠端異動日誌。

註: 作為替代方案，您可能想要使用非同步日誌登載來設定從 BETA 到 ALPHA 的遠端日誌登載。

使用此指令來設定從 BETA 到 ALPHA 的非同步日誌登載，而不是步驟 第 336 頁的『7』中的指令。

```
ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
```

如果 ALPHA 上的伺服器或日誌登載是失敗的來源，則 BETA 會在不等待新的異動日誌項目抄寫至 ALPHA 的情況下啟動。

當 ALPHA 再次連線時，使用指令將抄寫模式切換至 *SYNC。

使用 第 331 頁的『IBM i 上 ASP 的鏡映異動日誌配置』中的資訊來決定是同步、非同步或兩者的混合鏡映日誌。 預設值是同步抄寫，等待來自遠端異動日誌的回應 60 秒。

12. 請驗證 ALPHA 及 BETA 上的異動日誌已啟用，且遠端異動日誌抄寫的狀態為。

    a) 在 ALPHA 上:

    ```
    WRKMQMJRN MQMNAME(QM1)
    ```

    b) 在測試版上:

    ```
    WRKMQMJRN MQMNAME(QM1)
    ```

13. 在 ALPHA 和 BETA 上啟動佇列管理程式實例。

    a) 在 ALPHA 上啟動第一個實例，使它成為作用中實例。 啟用切換至待命實例。

    ```
    STRMQM MQMNAME(QM1) STANDBY(*YES)
    ```

    b) 在 BETA 上啟動第二個實例，使它成為待命實例。

    ```
    STRMQM MQMNAME(QM1) STANDBY(*YES)
    ```

結果

用來檢查佇列管理程式狀態:

1. Alpha 上佇列管理程式實例的狀態應該為。
2. BETA 上佇列管理程式實例的狀態應該為。

範例

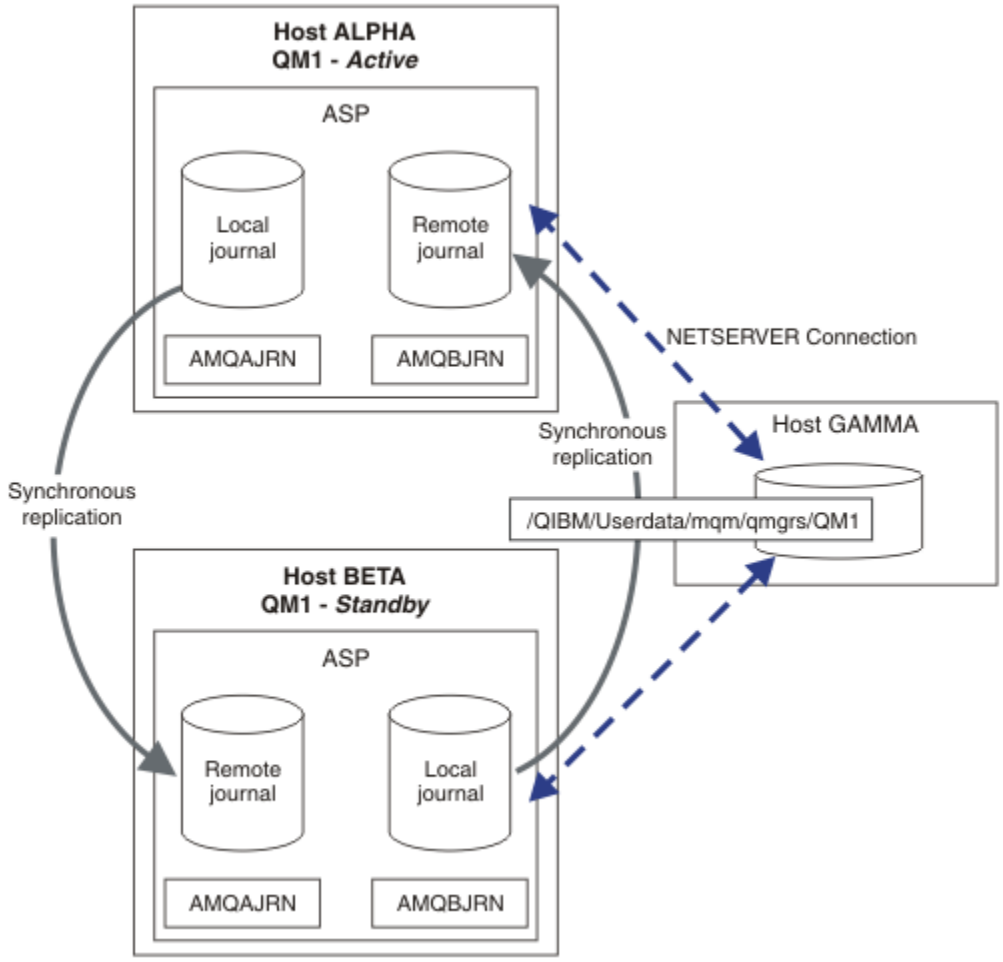


圖 *29:* 鏡映異動日誌配置

下一步

- 驗證作用中及待命實例自動切換。 您可以執行範例高可用性範例程式來測試切換; 請參閱 高可用性範例程式。 範例程式是 ' C' 用戶端。 您可以從 Windows 或 Unix 平台執行它們。
  1. 啟動高可用性範例程式。
  2. 在 ALPHA 上，結束要求切換的佇列管理程式:

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. 請檢查 BETA 上的實例是否處於作用中。
  4. 在 ALPHA 上重新啟動

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 查看替代高可用性配置:
  1. 使用 NetServer ， 將佇列管理程式資料放置在 Windows 伺服器上。
  2. 不使用遠端日誌登載來鏡映佇列管理程式異動日誌，而是將異動日誌儲存在獨立 ASP 上。 使用 IBM i 叢集作業，將獨立 ASP 從 ALPHA 傳送至 BETA。

IBM i *IBM i 上的交換式獨立 ASP 異動日誌配置*

您不需要抄寫獨立 ASP 日誌登載，即可建立多重實例佇列管理程式配置。 您確實需要自動化方法，將獨立 ASP 從作用中佇列管理程式傳送至待命佇列管理程式。 有替代高可用性解決方案可以使用獨立 ASP，但並非所有解決方案都需要使用多重實例佇列管理程式。

使用獨立 ASP 時，您不需要鏡映佇列管理程式異動日誌。 如果您已安裝叢集管理，且管理佇列管理程式實例的伺服器位於相同的叢集資源群組中，則在執行作用中實例的主機失敗時，佇列管理程式日誌登載可以在作用中伺服器的短距離內自動傳送至另一部伺服器。 您也可以在計劃的交換器中手動傳送異動日誌，也可以撰寫指令程序以程式化方式傳送獨立 ASP。

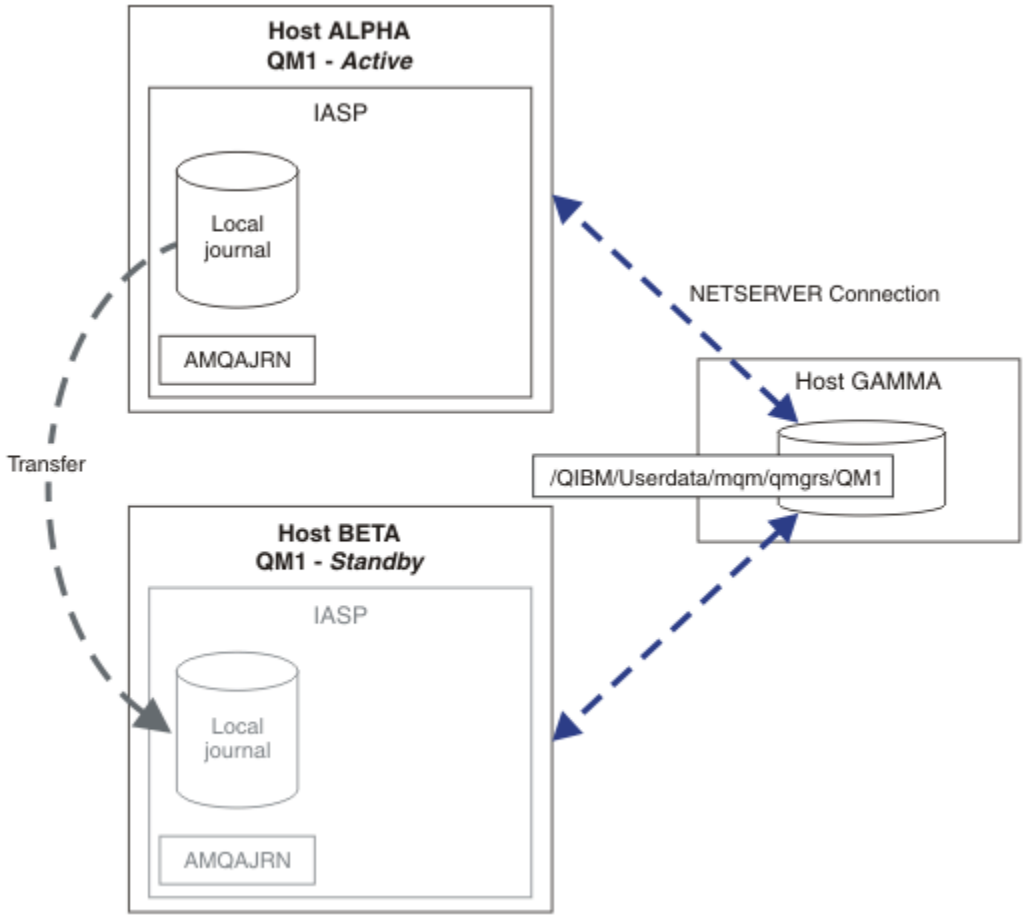


圖 *30:* 使用獨立 *ASP* 傳送佇列管理程式日誌

對於多重實例佇列管理程式作業，佇列管理程式資料必須儲存在共用檔案系統上。檔案系統可以在各種不同的平台上管理。 您無法在 ASP 或獨立 ASP 上儲存多重實例佇列管理程式資料。

共用檔案系統會在配置中執行兩個角色: 在佇列管理程式的所有實例之間共用相同的佇列管理程式資料。 檔案系統必須具有健全的鎖定通訊協定，以確保在啟動佇列管理程式資料之後，只有一個佇列管理程式實例具有佇列管理程式資料的存取權。 如果佇列管理程式失敗，或與檔案伺服器的通訊中斷，則檔案系統必須解除鎖定不再與檔案系統通訊的作用中實例所保留的佇列管理程式資料。 然後，待命佇列管理程式實例可以取得佇列管理程式資料的讀寫存取權。 檔案系統通訊協定必須符合一組規則，才能正確地使用多重實例佇列管理程式; 請參閱 第 324 頁的『IBM i 上高可用性解決方案的元件』。

鎖定機制會序列化啟動佇列管理程式指令，並控制佇列管理程式的作用中實例。 一旦佇列管理程式變成作用中，它會從您或 HA 叢集已傳送至待命伺服器的本端日誌登載重新建置其佇列。 等待重新連線至相同佇列管理程式的可重新連線用戶端會重新連線，且會取消任何進行中的交易。 已啟動配置為作為佇列管理程式服務啟動的應用程式。

您需要透過配置叢集資源管理程式或手動傳送獨立 ASP，確保將獨立 ASP 上失敗作用中佇列管理程式實例的本端日誌傳送至管理新啟動待命佇列管理程式實例的伺服器。 如果您決定使用獨立 ASP 進行備份及災難

回復，並對多重實例佇列管理程式配置使用遠端異動日誌鏡映，則使用獨立 ASP 並不會排除配置遠端異動日誌及鏡映。

如果您已選擇使用獨立 ASP，則可以考慮使用替代的高可用性配置。 第 345 頁的『獨立 ASP 及高可用性』中說明這些解決方案的背景。

1. 不使用多重實例佇列管理程式，而是完全在獨立 ASP 上安裝並配置單一實例佇列管理程式，並使用 IBM i 高可用性服務讓佇列管理程式失效接手。 您可能需要使用佇列管理程式監視器來擴增解決方案，以偵測佇列管理程式是否獨立於伺服器而失敗。 這是 *Supportpac MC41:* 配置 *IBM MQ for iSeries for High Availability*。
2. 使用獨立 ASP 及跨站台鏡映 (XSM) 來鏡映獨立 ASP，而不是在本端匯流排上切換獨立 ASP。 這會將獨立 ASP 解決方案的地理範圍延伸到容許長時間寫入日誌記錄所花費的時間。

IBM i 在 *IBM i* 上使用獨立 *ASP* 及 *NetServer* 建立多重實例佇列管理程式

建立要在兩部 IBM i 伺服器上執行的多重實例佇列管理程式。 佇列管理程式資料會使用 NetServer 儲存在 IBM i 伺服器中。 佇列管理程式異動日誌儲存在獨立 ASP 上。 使用 IBM i 叢集作業或手動程序，將包含佇列管理程式日誌登載的獨立 ASP 傳送至其他 IBM i 伺服器。

開始之前

1. 此作業需要三部 IBM i 伺服器。 在其中兩個上安裝 IBM MQ : 範例中的 ALPHA 和 BETA。 產品必須至少為 IBM WebSphere MQ 7.0.1 Fix Pack 1。
2. 第三部伺服器是 IBM i 伺服器，由 NetServer 連接至 ALPHA 及 BETA。 它用來共用佇列管理程式資料。 它不需要有 IBM MQ 安裝架構。 作為暫時步驟，在伺服器上安裝 IBM MQ 有助於設定佇列管理程式目錄及許可權。
3. 請確定 QMQM 使用者設定檔在這三部伺服器上都有相同的密碼。
4. 安裝 IBM i NetServer;請參閱 i5/OS NetServer 。
5. 建立程序以將獨立 ASP 從失敗的佇列管理程式傳送至接管的待命資料庫。 您可能會發現 *SupportPac MC41:* 配置 *IBM MQ for iSeries for High Availability* 有助於設計您的獨立 ASP 傳送程序。

關於這項作業

請執行下列步驟來建立 第 345 頁的圖 31 中所示的配置。 佇列管理程式資料是使用 IBM i NetServer 來連接。

- 在要儲存佇列管理程式資料的 GAMMA 上建立從 ALPHA 及 BETA 到目錄共用的連線。 此作業也會設定必要的許可權、使用者設定檔及密碼。
- 在 IBM i 伺服器 Alpha 上建立佇列管理程式 QM1 。
- 在其他 IBM i 伺服器 BETA 上新增 QM1 的佇列管理程式控制資訊。
- 啟動佇列管理程式，允許待命實例。

程序

1. 執行作業 第 324 頁的『在 IBM i 上使用 NetServer 建立佇列管理程式資料的網路共用』。

   因此， ALPHA 和 BETA 具有一個共用 /QNTC/GAMMA/WMQ，指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 。 使用者設定檔 QMQM 和 QMQMADM 具有必要的許可權，且 QMQM 在所有三個系統上都具有相符的密碼。
2. 在 ALPHA 上建立佇列管理程式 QM1 ，並將佇列管理程式資料儲存在 GAMMA 上。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   路徑會使用 NetServer ，在中建立佇列管理程式資料。
3. 使用指令來檢查 Alpha 上針對所建立的 IBM MQ 配置資料。

   下一步需要此資訊。

在此範例中，在的 ALPHA 上建立下列配置:

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QMQM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

4. 使用指令在測試版上建立 QM1 的佇列管理程式實例。 在 BETA 上執行下列指令，以在 BETA 上修改佇列管理程式控制資訊。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

提示: 複製並貼上配置資訊。 Alpha 和 BETA 上的佇列管理程式段落相同。

5. 在 ALPHA 和 BETA 上啟動佇列管理程式實例。
   a) 在 ALPHA 上啟動第一個實例，使它成為作用中實例。 啟用切換至待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

   b) 在 BETA 上啟動第二個實例，使它成為待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

結果

用來檢查佇列管理程式狀態:

1. Alpha 上佇列管理程式實例的狀態應該為。
2. BETA 上佇列管理程式實例的狀態應該為。

範例

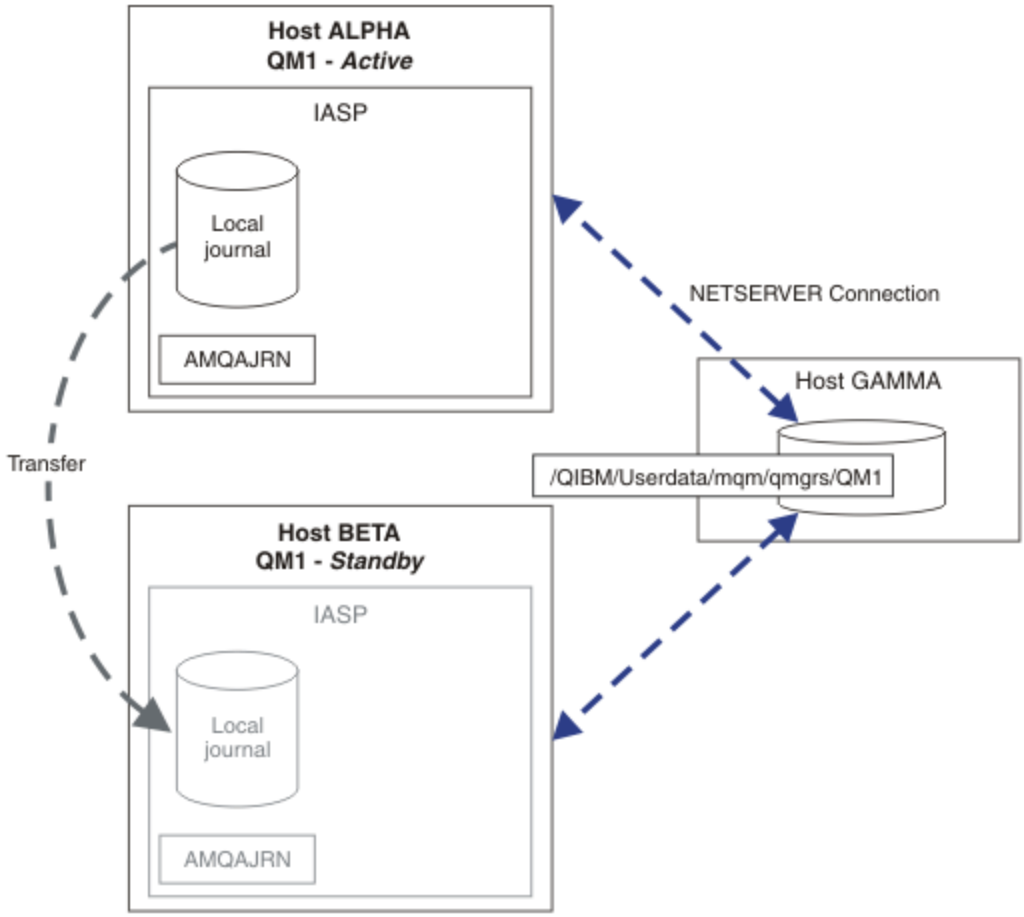


圖 *31:* 使用獨立 *ASP* 傳送佇列管理程式日誌

下一步

- 驗證作用中及待命實例自動切換。 您可以執行範例高可用性範例程式來測試切換; 請參閱 高可用性範例程式。 範例程式是 ' C' 用戶端。 您可以從 Windows 或 Unix 平台執行它們。
  1. 啟動高可用性範例程式。
  2. 在 ALPHA 上，結束要求切換的佇列管理程式:

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. 請檢查 BETA 上的實例是否處於作用中。
  4. 在 ALPHA 上重新啟動

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 查看替代高可用性配置:
  1. 使用 NetServer ， 將佇列管理程式資料放置在 IBM i 伺服器上。
  2. 使用遠端日誌登載將日誌登載鏡映至待命伺服器，而不是使用獨立 ASP 將佇列管理程式日誌登載傳送至待命伺服器。

IBM i 獨立 *ASP* 及高可用性

獨立 ASP 可讓應用程式及資料在伺服器之間移動。 獨立 ASP 的彈性表示它們是部分 IBM i 高可用性解決方案的基礎。 在考量對佇列管理程式異動日誌使用 ASP 或獨立 ASP 時，您應該考量根據獨立 ASP 的其他高可用性配置。

輔助儲存區 (ASP) 是 IBM i 架構的建置區塊。 硬碟機會分組在一起，以形成單一 ASP。 透過將物件放置在不同的 ASP 中，您可以保護一個 ASP 中的資料不受另一個 ASP 中磁碟故障的影響。

每個 IBM i 伺服器至少有一個 基本 ASP，稱為系統 ASP。 它指定為 ASP1，有時稱為 *SYSBAS。 您最多可以配置 31 個其他基本 使用者 ASP，因為它們共用相同的名稱空間，無法從應用程式的觀點來區分系統 ASP。 透過使用多個基本 ASP 來將應用程式配送至多個磁碟，您可以增進效能並減少回復時間。 使用多個基本 ASP 也可以針對磁碟故障提供某種程度的隔離，但它不會改善整體可靠性。

獨立 ASP 是特殊類型的 ASP。 它們通常稱為獨立磁碟儲存區。 獨立磁碟儲存區是 IBM i 高可用性的主要元件。 您可以儲存在獨立磁碟儲存裝置上視為與它們所連接的現行系統無關的資料及應用程式。 您可以配置可切換或不可切換的獨立 ASP。 從可用性角度來看，您通常只關心可切換的獨立 ASP，它們可以自動從伺服器傳送至伺服器。 因此，您可以將獨立 ASP 上的應用程式及資料從伺服器移至伺服器。

與基本使用者 ASP 不同，獨立 ASP 不會與系統 ASP 共用相同的名稱空間。 使用使用者 ASP 的應用程式需要變更才能使用獨立 ASP。 您需要驗證您的軟體以及您使用的協力廠商軟體在獨立 ASP 環境中運作。

當獨立 ASP 連接至不同的伺服器時，獨立 ASP 的名稱空間必須與系統 ASP 的名稱空間結合。 此處理程序稱為 轉接 獨立 ASP。 您可以在不執行伺服器 IPL 的情況下轉接獨立 ASP。 需要叢集支援，才能自動將獨立 ASP 從一部伺服器傳送至另一部伺服器。

使用獨立 ASP 來建置可靠的解決方案

日誌登載至獨立 ASP，而不是日誌登載至 ASP 並使用日誌登載抄寫，提供替代方法來為待命佇列管理程式提供失敗佇列管理程式實例的本端日誌副本。 若要自動將獨立 ASP 傳送至另一部伺服器，您必須已安裝並配置叢集作業支援。 根據叢集支援及低階磁碟鏡映，有許多獨立 ASP 的高可用性解決方案，您可以使用多重實例佇列管理程式來結合或替代。

下列清單說明根據獨立 ASP 建置可靠解決方案所需的元件。

日誌登載
佇列管理程式及其他應用程式會使用本端日誌登載來安全地將持續資料寫入磁碟，以防止因伺服器故障而遺失記憶體中的資料。 這有時稱為時間點一致性。 它不保證在一段時間內發生的多個更新項目的一致性。

確定控制
透過使用廣域交易，您可以協調訊息及資料庫的更新，以便寫入日誌登載的資料一致。 它使用兩段式確定通訊協定來提供一段時間內的一致性。

交換式磁碟

交換式磁碟由 HA 叢集中的裝置叢集資源群組 (CRG) 管理。 在非計劃性中斷的情況下， CRG 會自動將獨立 ASP 切換至新的伺服器。 CRG 在地理上受限於本端 IO 匯流排的範圍。

透過在切換式獨立 ASP 上配置本端日誌，您可以將日誌傳送至不同的伺服器，並回復處理訊息。 除非獨立 ASP 失敗，否則不會遺失沒有同步點控制或以同步點控制確定的持續訊息變更。

如果您在切換式獨立 ASP 上同時使用日誌登載及確定控制，則可以將資料庫異動日誌及佇列管理程式異動日誌傳送至不同的伺服器，並回復處理異動，而不會失去一致性或已確定異動。

跨站台鏡映 (XSM)
XSM 會透過 TCP/IP 網路將主要獨立 ASP 鏡映至地理上遠端次要獨立 ASP，並在失敗時自動傳送控制。 您可以選擇配置同步或非同步鏡映。 同步鏡映會降低佇列管理程式的效能，因為在正式作業系統上的寫入作業完成之前會鏡映資料，但它會保證次要獨立 ASP 是最新的。 而如果您使用非同步鏡映，則無法保證次要獨立 ASP 是最新的。 非同步鏡映會維護次要獨立 ASP 的一致性。

有三種 XSM 技術。

地理鏡映
地理鏡映是叢集作業的延伸，可讓您在廣域範圍內切換獨立 ASP。 它同時具有同步及非同步模式。 您只能在同步模式下保證高可用性，但分隔獨立 ASP 可能會對效能造成太大影響。 您可以結合地理鏡映與交換式磁碟，以提供本端高可用性及遠端災難回復。

Metro Mirror
Metro Mirror 是一種裝置層次服務，提供比本端匯流排更長距離的快速本端同步鏡映。 您可以將它與多重實例佇列管理程式結合，以提供佇列管理程式的高可用性，並透過具有兩個獨立 ASP 副本，提供佇列管理程式日誌的高可用性。

Global Mirror
Global Mirror 是提供非同步鏡映的裝置層次服務，適合在較長的距離上進行備份及災難回復，但並非高可用性的正常選擇，因為它只會維護時間點的一致性，而不是貨幣。

你應該考慮的關鍵決策點是，

ASP 或獨立 ASP?
您不需要執行 IBM i HA 叢集，即可使用多重實例佇列管理程式。 如果您已使用獨立 ASP，或您具有其他需要獨立 ASP 之應用程式的可用性需求，則可以選擇獨立 ASP。 它可能值得結合獨立 ASP 與多重實例佇列管理程式，以取代佇列管理程式監視，作為偵測佇列管理程式失敗的方法。

可用性?
回復時間目標 (RTO) 是什麼? 如果您需要出現近乎不中斷的行為，則哪個解決方案具有最快的回復時間?

日誌登載可用性?
如何將日誌刪除為單一失敗點。 您可以採用硬體解決方案，使用 RAID 1 裝置或更高版本，或者您可以使用抄本日誌或磁碟鏡映來結合或使用軟體解決方案。

距離?
作用中及待命佇列管理程式實例之間的距離。 您的使用者是否可以容忍在大於大約 250 公尺的距離上同步抄寫的效能降低?

技能?
需要執行一些工作，以將維護及定期執行解決方案所涉及的管理作業自動化。 根據 ASP 及獨立 ASP 的解決方案，執行自動化所需的技能有所不同。

IBM i 在 *IBM i* 上刪除多重實例佇列管理程式

在刪除多重實例佇列管理程式之前，請停止遠端日誌登載，並移除佇列管理程式實例。

開始之前

1. 在此範例中，在伺服器 ALPHA 和 BETA 上定義兩個 QM1 佇列管理程式實例。 A 是作用中實例， BETA 是待命實例。 與佇列管理程式 QM1 相關聯的佇列管理程式資料，會使用 NetServer 儲存在 IBM i 伺服器 GAMMA 上。 請參閱第 335 頁的『在 IBM i 上使用日誌登載鏡映及 NetServer 建立多重實例佇列管理程式』。
2. 必須連接 ALPHA 及 BETA， IBM MQ 才能刪除任何已定義的遠端異動日誌。
3. 驗證可以使用系統指令 **EDTF** 或 **WRKLNK** 來存取 /QNTC 目錄及伺服器目錄檔案共用

關於這項作業

在使用 **DLTMQM** 指令從伺服器刪除多重實例佇列管理程式之前，請使用 **RMVMQMINF** 指令移除其他伺服器上的任何佇列管理程式實例。

當您使用 **RMVMQMINF** 指令移除佇列管理程式實例時，會刪除字首為 AMQ 且與實例相關聯的本端及遠端異動日誌。 也會刪除伺服器本端佇列管理程式實例的相關配置資訊。

請勿在保留其餘佇列管理程式實例的伺服器上執行 **RMVMQMINF** 指令。 這樣做會導致 **DLTMQM** 無法正確運作。

使用 **DLTMQM** 指令刪除佇列管理程式。 佇列管理程式資料會從網路共用中移除。 會刪除字首為 AMQ 且與實例相關聯的本端及遠端異動日誌。 **DLTMQM** 也會刪除伺服器本端佇列管理程式實例的相關配置資訊。

在範例中，只有兩個佇列管理程式實例。 IBM MQ 支援具有一個作用中佇列管理程式實例及一個待命實例的執行中多重實例配置。 如果您已建立要在執行中配置中使用的其他佇列管理程式實例，請先使用 **RMVMQMINF** 指令移除它們，然後再刪除其餘實例。

程序

1. 在每一部伺服器上執行 **CHGMQMJRN RMTJRNSTS** (*INACTIVE) 指令，以讓佇列管理程式實例之間的遠端日誌登載成為非作用中。

   a) 在 ALPHA 上:

```
CHGMQMJRN MQMNAME('QM1')
RMTJRNRDB('BETA') RMTJRNSTS(*INACTIVE)
```

b) 在測試版上:

```
CHGMQMJRN MQMNAME('QM1')
RMTJRNRDB('ALPHA') RMTJRNSTS(*INACTIVE)
```

2. 在 ALPHA (作用中佇列管理程式實例) 上執行 **ENDMQM** 指令，以停止 QM1 的兩個實例。

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) INSTANCE(*ALL) ENDCCTJOB(*YES)
```

3. 在 ALPHA 上執行 **RMVMQMINF** 指令，以從 ALPHA 及 BETA 移除實例的佇列管理程式資源。

```
RMVMQMINF MQMNAME(QM1)
```

RMVMQMINF 會從 ALPHA 移除 QM1 的佇列管理程式配置資訊。 如果日誌名稱以 AMQ 作為字首，則會從 ALPHA 中刪除與 QM1 相關聯的本端日誌。 如果異動日誌名稱以 AMQ 為字首，且已建立遠端異動日誌，則它也會從 BETA 移除遠端異動日誌。

4. 在測試版上執行 **DLTMQM** 指令，以刪除 QM1。

```
DLTMQM MQMNAME(QM1)
```

DLTMQM 會從 GAMMA 上的網路共用中刪除佇列管理程式資料。 它會從測試版中移除 QM1 的佇列管理程式配置資訊。 如果日誌登載名稱以 AMQ 作為字首，則會從測試版中刪除與 QM1 相關聯的本端日誌登載。 如果異動日誌名稱以 AMQ 為字首，且已建立遠端異動日誌，則它也會從 ALPHA 中移除遠端異動日誌。

結果

DLTMQM 和 **RMVMQMINF** 刪除 **CRTMQM** 和 **ADDMQJRN** 所建立的本端和遠端異動日誌。 這些指令也會刪除異動日誌接收器。 異動日誌及異動日誌接收器必須遵循名稱以 AMQ 開頭的命名慣例。 **DLTMQM** 及 **RMVMQMINF** 從 `mqs.ini` 中移除佇列管理程式物件、佇列管理程式資料及佇列管理程式配置資訊。

下一步

替代方法是在步驟 第 347 頁的『1』 中取消啟動日誌登載之後，以及在結束佇列管理程式實例之前，發出下列指令。 或者，如果您未遵循命名慣例，則必須依名稱刪除異動日誌及異動日誌接收器。

1. 在 ALPHA 上:

```
RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('BETA')
```

2. 在測試版上:

```
RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('ALPHA')
```

刪除日誌之後，請繼續執行其餘步驟。

IBM i 在 *IBM i* 上備份多重實例佇列管理程式

此程序顯示如何備份本端伺服器上的佇列管理程式物件，以及網路檔案伺服器上的佇列管理程式資料。 調整範例以備份其他佇列管理程式的資料。

開始之前

在此範例中，與佇列管理程式 QM1 相關聯的佇列管理程式資料使用 NetServer 儲存在稱為 GAMMA 的 IBM i 伺服器上。 請參閱第 335 頁的『在 IBM i 上使用日誌登載鏡映及 NetServer 建立多重實例佇列管理程式』。 IBM MQ 安裝在伺服器上， Alpha 和 BETA。 佇列管理程式 QM1 在 ALPHA 和 BETA 上配置。

關於這項作業

IBM i 不支援從遠端目錄儲存資料。 使用檔案系統伺服器本端的備份程序，將佇列管理程式資料儲存在遠端檔案系統上。 在此作業中，網路檔案系統位於 IBM i 伺服器 GAMMA 上。 佇列管理程式資料會備份在 GAMMA 上的儲存檔中。

如果網路檔案系統位於 Windows 或 Linux 上，您可以將佇列管理程式資料儲存在壓縮檔中，然後儲存它。 如果您有備份系統 (例如 Tivoli Storage Manager)，請使用它來備份佇列管理程式資料。

程序

1. 在 ALPHA 上為與 QM1 相關聯的佇列管理程式檔案庫建立儲存檔。

   使用佇列管理程式檔案庫名稱來命名儲存檔。

   ```
   CRTSAVF FILE(QGPL/QMQM1)
   ```

2. 將佇列管理程式檔案庫儲存在 ALPHA 上的儲存檔中。

   ```
   SAVLIB LIB(QMQM1) DEV(*SAVF) SAVF(QGPL/QMQM1)
   ```

3. 在 GAMMA 上建立佇列管理程式資料目錄的儲存檔。

   使用佇列管理程式名稱來命名儲存檔。

   ```
   CRTSAVF FILE(QGPL/QMDQM1)
   ```

4. 儲存 GAMMA 上本端目錄中佇列管理程式資料的副本。

   ```
   SAV DEV('/QSYS.LIB/QGPL.LIB/QMDQM1.FILE') OBJ('/QIBM/Userdata/mqm/qmgrs/QM1')
   ```

IBM i 設定多重實例佇列管理程式的指令

IBM MQ 具有指令可簡化配置日誌登載抄寫、新增佇列管理程式實例，以及配置佇列管理程式以使用獨立 ASP。

建立及管理本端及遠端異動日誌的異動日誌指令為:

ADDMQMJRN
使用此指令，您可以建立佇列管理程式實例的具名本端及遠端日誌登載，並配置抄寫是同步還是非同步、同步逾時為何，以及是否要立即啟動遠端日誌登載。

CHGMQMJRN
此指令會修改影響抄本日誌的逾時、狀態及遞送參數。

RMVMQMJRN
從佇列管理程式實例移除指名的 遠端 日誌登載。

WRKMQMJRN
列出本端佇列管理程式實例的本端及遠端日誌登載狀態。

使用下列指令來新增及管理其他佇列管理程式實例，這些指令會修改 `mqs.ini` 檔案。

ADDMQMINF
此指令會使用您使用 DSPMQMINF 指令從 `mqs.ini` 檔案擷取的資訊，在不同的 IBM i 伺服器上新增佇列管理程式實例。

RMVMQMINF
移除佇列管理程式實例。 請使用這個指令來移除現有佇列管理程式的實例，或移除已從不同伺服器刪除之佇列管理程式的配置資訊。

CRTMQM 指令有三個參數可協助配置多重實例佇列管理程式,

MQMDIRP (*DFT | 目錄字首)
使用此參數來選取對映至網路儲存體上佇列管理程式資料的裝載點。

ASP (*SYSTEM|*ASPDEV| *auxiliary-storage-pool-number*)
指定 *SYSTEM 或 *auxiliary-storage-pool-number* ，以將佇列管理程式異動日誌放置在系統或基本使用者 ASP 上。 選取 *ASPDEV 選項，並使用 **ASPDEV** 參數設定裝置名稱，以將佇列管理程式日誌登載放置在獨立 ASP 上。

ASPDEV (*ASP|*device-name*)
指定主要或次要獨立 ASP 裝置的 *device-name* 。 選取 *ASP 的結果與指定 **ASP** (*SYSTEM) 的結果相同。

IBM i IBM i 上的效能及磁碟失效接手考量

使用不同的輔助儲存區來增進效能和可靠性。

如果您在應用程式中使用大量持續訊息或大量訊息，則將這些訊息寫入磁碟所花費的時間會成為系統效能的重要因素。

請確定您有足夠的磁碟啟動來處理此可能性，或考量在其中保留佇列管理程式異動日誌接收器的個別「輔助儲存區 (ASP)」。

當您使用 **CRTMQM** 的 ASP 參數建立佇列管理程式時，您可以指定儲存佇列管理程式檔案庫及日誌登載的 ASP。 依預設，佇列管理程式檔案庫和異動日誌及 IFS 資料會儲存在系統 ASP 中。

ASP 容許隔離一或多個特定硬碟機上的物件。 這也可以減少由於磁碟媒體故障而造成的資料流失。 在大部分情況下，只會遺失受影響 ASP 中硬碟機上儲存的資料。

建議您將佇列管理程式檔案庫及日誌登載資料儲存在個別使用者 ASP 與根 IFS 檔案系統的 ASP 中，以提供失效接手並減少磁碟競用。

如需相關資訊，請參閱 IBM i 說明文件中的 備份及回復 。

IBM i 使用 SAVLIB 在 IBM i 上儲存 IBM MQ 檔案庫

您無法使用 SAVLIB LIB(*ALLUSR) 來儲存 IBM MQ 檔案庫，因為這些檔案庫的名稱以 Q 開頭。

您可以使用 SAVLIB LIB(QM*) 來儲存所有佇列管理程式檔案庫，但前提是您是使用 *SAVF 以外的儲存裝置。 對於 DEV(*SAVF)，您必須對系統上每一個佇列管理程式檔案庫使用 SAVLIB 指令。

IBM i 靜止 IBM MQ for IBM i

本節說明如何靜止 (循序結束) IBM MQ for IBM i。

若要靜止 IBM MQ for IBM i:

1. 登入新的互動式 IBM MQ for IBM i 階段作業，確定您沒有存取任何物件。
2. 確保您已執行下列作業:
   - *ALLOBJ 權限，或 QMQM 檔案庫的物件管理權限
   - 有足夠權限使用 ENDSBS 指令
3. 通知所有使用者您將停止 IBM MQ for IBM i。
4. 然後如何繼續取決於您是否要關閉 (靜止) 單一佇列管理程式 (其他佇列管理程式可能存在的位置) (請參閱 第 351 頁的『關閉 IBM MQ for IBM i 的單一佇列管理程式』) 或所有佇列管理程式 (請參閱 第 352 頁的『關閉 IBM MQ for IBM i 的所有佇列管理程式』)。
5. 在 qshell 中輸入下列指令，以關閉 mqweb 伺服器:

```
/QIBM/ProdData/mqm/bin/endmqweb
```

ENDMQM 參數 ENDCCTJOB (*YES)

與舊版相比， ENDMQM 參數 ENDCCTJOB (*YES) 在 IBM MQ for IBM i V6.0 以及更新版本中的運作方式不同。

在舊版上，當您指定 ENDCCTJOB (*YES) 時， MQ 會強制終止您的應用程式。

在 IBM MQ for IBM i V6.0 或更新版本上，當您指定 ENDCCTJOB (*YES) 時，您的應用程式不會終止，而是與佇列管理程式切斷連線。

如果您指定 ENDCCTJOB (*YES) ，且您有應用程式未寫入以偵測佇列管理程式是否正在結束，則下次發出新的 MQI 呼叫時，該呼叫將會傳回 MQRC_CONNECTION_BROKEN (2009) 錯誤。

作為使用 ENDCCTJOB (*YES) 的替代方案，請使用參數 ENDCCTJOB (*NO) 並使用 WRKMQM 選項 22 (使用工作) 來手動結束將阻止佇列管理程式重新啟動的任何應用程式工作。

IBM i 關閉 IBM MQ for IBM i 的單一佇列管理程式

使用此資訊來瞭解三種關機類型。

在接下來的程序中，我們使用範例佇列管理程式名稱 QMgr1 及範例子系統名稱 SUBX。 必要的話，請將這些名稱取代為您自己的值。

計劃的關閉

計劃在 IBM i 上關閉佇列管理程式

1. 關閉之前，請執行:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

2. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

   如果 QMgr1 未結束，則通道或應用程式可能忙碌中。
3. 如果您必須立即關閉 QMgr1 ，請執行下列指令:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

意外關閉

1. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

   如果 QMgr1 未結束，則通道或應用程式可能忙碌中。
2. 如果您需要立即關閉 QMgr1 ，請執行下列指令:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

在異常狀況下關閉

1. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

   如果 QMgr1 未結束，請在下列情況下繼續步驟 3:

   - QMgr1 位於其自己的子系統中，或
   - 您可以結束與 QMgr1 共用相同子系統的所有佇列管理程式。 對所有這類佇列管理程式使用非計劃的關閉程序。
2. 當您針對共用子系統的所有佇列管理程式採取程序中的所有步驟 (在我們的範例中，是 SUBX) 時，請執行:

```
ENDSBS SUBX *IMMED
```

如果此指令無法完成，請使用非計劃性關閉程序關閉所有佇列管理程式，並在機器上執行 IPL。

警告: 對於因 ENDJOB 或 ENDSBS 而無法結束的 IBM MQ 工作，請不要使用 ENDJOBABN，除非您準備在之後立即在機器上執行 IPL。

3. 執行下列指令來啟動子系統:

```
STRSBS SUBX
```

4. 執行下列指令，立即關閉佇列管理程式:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(10)
```

5. 執行下列指令來重新啟動佇列管理程式:

```
STRMQM MQMNAME(QMgr1)
```

如果失敗，且您:

- 執行 IPL 來重新啟動您的機器，或
- 只有單一佇列管理程式

執行下列指令來清理 IBM MQ 共用記憶體:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

在重複步驟 5 之前。

如果佇列管理程式重新啟動花費數秒以上的時間， IBM MQ 會間歇性將狀態訊息新增至工作日誌，以詳述啟動進度。

如果您在重新啟動佇列管理程式時仍有問題，請聯絡 IBM 支援中心。 您可能採取的任何進一步動作可能會損壞佇列管理程式，導致 IBM MQ 無法回復。

IBM i 關閉 IBM MQ for IBM i 的所有佇列管理程式

使用此資訊來瞭解三種關機類型。

這些程序幾乎與單一佇列管理程式相同，但可能的話，會使用 *ALL 來取代佇列管理程式名稱，否則會輪流重複使用指令來使用每一個佇列管理程式名稱。 在整個程序中，我們使用範例佇列管理程式名稱 QMgr1 及範例子系統名稱 SUBX。 將這些取代為您自己的。

計劃的關閉

1. 關機前一小時，執行:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

針對您要關閉的每一個佇列管理程式，重複此步驟。

2. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

對您要關閉的每一個佇列管理程式重複此步驟; 個別指令可以平行執行。

如果任何佇列管理程式未在合理時間 (例如 10 分鐘) 內結束，請繼續步驟 3。

3. 若要立即關閉所有佇列管理程式，請執行下列動作:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

意外關閉

1. 若要關閉佇列管理程式，請執行:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ```

   對您要關閉的每一個佇列管理程式重複此步驟; 個別指令可以平行執行。

   如果佇列管理程式未結束，則通道或應用程式可能忙碌。
2. 如果您需要立即關閉佇列管理程式，請執行下列動作:

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```

在異常狀況下關閉

1. 若要關閉佇列管理程式，請執行:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ```

   對您要關閉的每一個佇列管理程式重複此步驟; 個別指令可以平行執行。
2. 透過執行下列指令來結束子系統 (在我們的範例中為 SUBX):

   ```
   ENDSBS SUBX *IMMED
   ```

   針對您要關閉的每一個子系統重複此步驟; 個別指令可以平行執行。

   如果此指令無法完成，請在系統上執行 IPL。

   警告: 對於因 ENDJOB 或 ENDSBS 而無法結束的工作，請勿使用 ENDJOBABN ，除非您準備在之後立即在系統上執行 IPL。
3. 執行下列指令來啟動子系統:

   ```
   STRSBS SUBX
   ```

   針對您要啟動的每個子系統重複此步驟。
4. 執行下列指令，立即關閉佇列管理程式:

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```
5. 執行下列指令來重新啟動佇列管理程式:

   ```
   STRMQM MQMNAME(QMgr1)
   ```

   針對您要啟動的每一個佇列管理程式，重複此步驟。

   如果任何佇列管理程式重新啟動需要幾秒鐘以上的時間， IBM MQ 會顯示間歇性詳細說明啟動進度的狀態訊息。

   如果您在重新啟動任何佇列管理程式時仍有問題，請聯絡 IBM 支援中心。 您可能採取的任何進一步動作可能會損壞佇列管理程式，導致 MQSeries 或 IBM MQ 無法回復。

z/OS 管理 IBM MQ for z/OS

管理佇列管理程式及相關聯資源包括您在啟動及管理這些資源時經常執行的作業。 選擇您偏好用來管理佇列管理程式及相關聯資源的方法。

IBM MQ for z/OS 可以由產品隨附的一組公用程式來控制及管理。 您可以使用 IBM MQ Script (MQSC) 指令或「可程式指令格式 (PCF)」來管理 IBM MQ for z/OS。 如需對 IBM MQ for z/OS 使用指令的相關資訊，請參閱 第 354 頁的『對 IBM MQ for z/OS 發出指令』。

IBM MQ for z/OS 也提供一組公用程式來協助您進行系統管理。 如需不同公用程式及其使用方式的相關資訊，請參閱 第 361 頁的『IBM MQ for z/OS 公用程式』。

如需如何管理 IBM MQ for z/OS 以及您可能必須執行的不同管理作業的詳細資料，請參閱下列鏈結:

相關概念

IBM MQ for z/OS 概念

相關工作

第 108 頁的『使用本端 IBM MQ 物件』
您可以管理本端 IBM MQ 物件，以支援使用「訊息佇列介面 (MQI)」的應用程式。

第 163 頁的『使用遠端 IBM MQ 物件』
您可以使用 MQSC 指令、PCF 指令或 administrative REST API，來管理遠端佇列管理程式上的 IBM MQ 物件。 您必須先定義本端佇列管理程式與遠端佇列管理程式之間的傳輸佇列及通道，以便指令可以傳送至遠端佇列管理程式，以及本端佇列管理程式所接收的回應，然後才能使用上述任何方法。 或者，您可以配置佇列管理程式叢集，然後使用相同的遠端管理方法。

第 7 頁的『管理 IBM MQ』
若要管理 IBM MQ 佇列管理程式及相關聯的資源，請從一組可用來啟動及管理這些資源的作業中選擇您偏好的方法。

規劃

在 z/OS 上規劃 IBM MQ 環境

配置

正在配置 z/OS

使用 IBM MQ for z/OS 公用程式

相關參考

第 18 頁的『您可以在 z/OS 上從中發出 MQSC 指令的來源』
MQSC 指令可以從各種來源發出，視指令而定。

可程式化指令格式參照

z/OS 對 IBM MQ for z/OS 發出指令

您可以在批次或互動模式下使用 IBM MQ Script 指令 (MQSC) 來控制佇列管理程式。

IBM MQ for z/OS 支援 MQSC 指令，這些指令可以從下列來源發出:

- z/OS 主控台或對等項目 (例如 SDSF/TSO)。

  使用 z/OS 主控台時，您需要將 `/cpf` 新增至指令的開頭，其中 `cpf` 是佇列管理程式子系統的指令字首。
- 起始設定輸入資料集。
- 提供的批次公用程式 CSQUTIL 會處理循序資料集中的指令清單。

  使用起始設定輸入資料集或所提供的批次公用程式時，您不需要將 `/cpf` 新增至指令的開頭。
- 適當授權的應用程式，將指令當作訊息傳送至指令輸入佇列。 應用程式可以是下列任何一項:
  - 批次區域程式
  - CICS 應用程式
  - IMS 應用程式
  - TSO 應用程式
  - 另一個 IBM MQ 系統上的應用程式或公用程式

第 357 頁的表 26 彙總 MQSC 指令及可從中發出指令的來源。

這些指令的大部分功能都可以從 IBM MQ for z/OS 作業及控制台中方便地取得。

在重新啟動 IBM MQ 子系統之後，會保留使用指令 (直接或間接) 對佇列管理程式的資源定義所做的變更。

IBM MQ for z/OS 也支援「可程式化指令格式 (PCF)」指令。 這些可簡化建立應用程式以管理 IBM MQ。 MQSC 指令是人類可讀的文字形式，而 PCF 可讓應用程式建立要求並讀取回覆，而無需剖析字串。 與

MQSC 指令一樣，應用程式也會將 PCF 指令當作訊息傳送至指令輸入佇列，以發出 PCF 指令。 如需使用 PCF 指令的相關資訊，以及指令的詳細資料，請參閱 可編程指令格式參考 說明文件。

z/OS IBM MQ for z/OS 上的專用及廣域定義

當您在 IBM MQ for z/OS 上定義物件時，可以選擇是要與其他佇列管理程式共用該定義 (廣域 定義) ，還是只由一個佇列管理程式使用該物件定義 (專用 定義)。 這稱為物件 處置。

廣域定義

如果您的佇列管理程式屬於佇列共用群組，您可以選擇與群組其他成員共用您所建立的任何物件定義。 這表示您必須只定義物件一次，以減少整個系統所需的定義總數。

廣域物件定義保留在 共用儲存庫 (Db2 共用資料庫) 中，可供佇列共用群組中的所有佇列管理程式使用。 這些物件的處置方式為 GROUP。

專用定義

如果您只想建立一個佇列管理程式所需的物件定義，或如果您的佇列管理程式不是佇列共用群組的成員，您可以建立未與佇列共用群組其他成員共用的物件定義。

專用物件定義保留在定義佇列管理程式的頁集零上。 這些物件具有 QMGR 處置。

您可以為所有類型的 IBM MQ 物件 (CF 結構除外 (亦即，通道、名稱清單、程序定義、佇列、佇列管理程式、儲存類別定義及鑑別資訊物件)) 建立專用定義，以及為所有類型的物件 (佇列管理程式除外) 建立廣域定義。

IBM MQ 會自動將群組物件的定義複製到使用它的每一個佇列管理程式的頁集零。 您可以視需要暫時變更定義的副本，而 IBM MQ 可讓您在必要時重新整理儲存庫副本中的頁面集副本。

IBM MQ 一律會在啟動時嘗試重新整理儲存庫副本中的頁集副本 (對於通道指令，這是在通道起始程式重新啟動時執行) ，或如果群組物件變更的話。

註: 只有在您建立定義副本之後群組的定義已變更時，才會從群組的定義重新整理定義副本。

這可確保頁面集副本反映儲存庫上的版本，包括佇列管理程式非作用中時所做的任何變更。 會產生 DEFINE REPLACE 指令來重新整理副本，因此在某些情況下不會執行重新整理，例如:

- 如果開啟佇列的副本，則變更佇列使用情形的重新整理會失敗。
- 如果佇列副本中有訊息，則刪除該佇列的重新整理會失敗。
- 如果佇列副本需要 ALTER 搭配 FORCE 來變更它。

在這些情況下，不會對該副本執行重新整理，而是對所有其他佇列管理程式上的副本執行重新整理。

如果佇列管理程式關閉，然後獨立式重新啟動，則會刪除任何物件的本端副本，除非例如，佇列具有相關聯的訊息。

第三個物件處置僅適用於本端佇列。 這可讓您建立共用佇列。 共用佇列的定義會保留在共用儲存庫上，且可供佇列共用群組中的所有佇列管理程式使用。 此外，共用佇列上的訊息也可供佇列共用群組中的所有佇列管理程式使用。 這在 共用佇列及佇列共用群組中有說明。 共用佇列具有物件處置 SHARED。

下表彙總佇列管理程式獨立式啟動且作為佇列共用群組成員的物件處置選項的效果。

處置	獨立式佇列管理程式	佇列共用群組的成員
QMGR	保留在頁集零上的物件定義。	保留在頁集零上的物件定義。
GROUP	不容許。	保留在共用儲存庫中的物件定義。 在群組中每一個佇列管理程式的頁集零上保留的本端副本。
共用	不容許。	保留在共用儲存庫中的佇列定義。 可供群組中任何佇列管理程式使用的訊息。

操作廣域定義

如果您要變更共用儲存庫中所保留物件的定義，則需要指定是要變更儲存庫上的版本，還是頁集零上的本端副本。 請使用物件處置作為指令的一部分來執行此動作。

z/OS 將指令導向至 z/OS 上的不同佇列管理程式

您可以使用 指令範圍 來控制指令執行所在的佇列管理程式。

您可以選擇在輸入指令的佇列管理程式上執行指令，或在佇列共用群組中的不同佇列管理程式上執行指令。 您也可以選擇在佇列共用群組中的所有佇列管理程式上平行發出特定指令。 這適用於 MQSC 指令及 PCF 指令。

這是由 指令範圍所決定。 指令範圍與物件處置一起使用，以判斷您要使用的物件版本。

例如，您可能想要變更物件的部分屬性，其定義保留在共用儲存庫中。

- 您可能只想變更一個佇列管理程式上的版本，而不要對儲存庫上的版本或其他佇列管理程式正在使用的版本進行任何變更。
- 您可能想要為未來使用者變更共用儲存庫中的版本，但保留現有副本不變。
- 您可能想要變更共用儲存庫中的版本，但也希望您所做的變更立即反映在佇列共用群組中的所有佇列管理程式上，這些佇列管理程式在其頁集零上保留物件的副本。

使用指令範圍來指定指令是在此佇列管理程式、另一個佇列管理程式或所有佇列管理程式上執行。 使用物件處置來指定您正在操作的物件是在共用儲存庫中 (群組物件)，還是在頁集零上的本端副本 (佇列管理程式物件)。

您不需要指定指令範圍和物件處置來使用共用佇列，因為佇列共用群組中的每一個佇列管理程式會將共用佇列當作單一佇列來處理。

z/OS IBM MQ for z/OS 的指令摘要

請使用本主題作為主要 MQSC 及 PCF 指令的參照。

第 356 頁的表 25 彙總 IBM MQ for z/OS 上可用來變更、定義、刪除及顯示 IBM MQ 物件的 MQSC 及 PCF 指令。

表 *25:* 依物件類型列出的主要 *MQSC* 及 *PCF* 指令摘要

MQSC 指令	**ALTER**	**定義**	**顯示畫面**	**DELETE**
PCF 指令	**變更**	**建立/複製**	**查詢**	**刪除**
AUTHINFO	X	X	X	X
CFSTATUS			X	
CFSTRUCT	X	X	X	X
CHANNEL	X	X	X	X
CHSTATUS			X	
名稱清單	X	X	X	X
PROCESS	X	X	X	X
QALIAS	M	M	M	M
QCLUSTER			M	
QLOCAL	M	M	M	M
QMGR	X		X	

表 *25:* 依物件類型列出的主要 *MQSC* 及 *PCF* 指令摘要 *(繼續)*				
MQSC 指令	**ALTER**	**定義**	**顯示畫面**	**DELETE**
QMODEL	M	M	M	M
QREMOTE	M	M	M	M
佇列	P	P	X	P
QSTATUS			X	
STGCLASS	X	X	X	X

表格符號的索引鍵:

- M = 僅限 MQSC
- P = 僅限 PCF
- X = 兩者

有許多其他 MQSC 及 PCF 指令可讓您管理其他 IBM MQ 資源，以及除了 第 356 頁的表 25 中彙總的那些動作之外，還會執行其他動作。

第 357 頁的表 26 顯示每一個 MQSC 指令，以及可從中發出每一個指令的位置:

- CSQINP1 起始設定輸入資料集
- CSQINP2 起始設定輸入資料集
- z/OS 主控台 (或對等項目)
- SYSTEM.COMMAND.INPUT 佇列及指令伺服器 (來自應用程式、CSQUTIL 或 CSQINPX 起始設定輸入資料集)

表 *26:* 從中執行 *MQSC* 指令的來源				
指令	**CSQINP1**	**CSQINP2**	**z/OS 主控台**	**指令輸入佇列及伺服器**
ALTER AUTHINFO		X	X	X
ALTER BUFFPOOL		X	X	X
ALTER CFSTRUCT		X	X	X
ALTER CHANNEL		X	X	X
ALTER NAMELIST		X	X	X
ALTER PSID			X	X
ALTER PROCESS		X	X	X
ALTER QALIAS		X	X	X
ALTER QLOCAL		X	X	X
ALTER QMGR		X	X	X
ALTER QMODEL		X	X	X
ALTER QREMOTE		X	X	X
ALTER SECURITY	X	X	X	X
ALTER STGCLASS		X	X	X
ALTER SUB		X	X	X

表 *26:* 從中執行 *MQSC* 指令的來源 *(*繼續*)*

指令	CSQINP1	CSQINP2	z/OS 主控台	指令輸入佇列及伺服器
ALTER TOPIC		X	X	X
ALTER TRACE	X	X	X	X
保存日誌	X	X	X	X
備份 CFSTRUCT			X	X
CLEAR QLOCAL		X	X	X
DEFINE AUTHINFO		X	X	X
DEFINE BUFFPOOL	X	X		
DEFINE CFSTRUCT		X	X	X
定義通道		X	X	X
DEFINE LOG			X	X
DEFINE NAMELIST		X	X	X
DEFINE PROCESS		X	X	X
DEFINE PSID	X		X	X
DEFINE QALIAS		X	X	X
DEFINE QLOCAL		X	X	X
DEFINE QMODEL		X	X	X
DEFINE QREMOTE		X	X	X
DEFINE STGCLASS		X	X	X
DEFINE SUB			X	X
DEFINE TOPIC		X	X	X
DELETE AUTHINFO		X	X	X
DELETE BUFFPOOL			X	X
DELETE CFSTRUCT		X	X	X
刪除通道			X	X
刪除名單		X	X	X
刪除處理程序		X	X	X
DELETE PSID			X	X
DELETE QALIAS		X	X	X
DELETE QLOCAL		X	X	X
DELETE QMODEL		X	X	X
DELETE QREMOTE		X	X	X
DELETE STGCLASS		X	X	X
DELETE SUB		X	X	X
刪除主題		X	X	X

表 *26:* 從中執行 *MQSC* 指令的來源 *(*繼續*)*

指令	CSQINP1	CSQINP2	z/OS 主控台	指令輸入佇列及伺服器
顯示保存檔	X	X	X	X
DISPLAY AUTHINFO		X	X	X
DISPLAY CFSTATUS			X	X
DISPLAY CFSTRUCT		X	X	X
顯示通道		X	X	X
DISPLAY CHSTATUS			X	X
DISPLAY CLUSQMGR			X	X
DISPLAY CMDSERV	X	X	X	X
DISPLAY CONN		X	X	X
顯示 CHINIT		X	X	X
顯示群組		X	X	X
DISPLAY LOG	X	X	X	X
顯示名單		X	X	X
DISPLAY PROCESS		X	X	X
DISPLAY QALIAS		X	X	X
DISPLAY QCLUSTER		X	X	X
DISPLAY QLOCAL		X	X	X
DISPLAY QMGR		X	X	X
DISPLAY QMODEL		X	X	X
DISPLAY QREMOCE		X	X	X
DISPLAY QSTATUS		X	X	X
顯示佇列		X	X	X
DISPLAY SECURITY			X	X
DISPLAY STGCLASS		X	X	X
DISPLAY SUB		X	X	X
顯示主題		X	X	X
DISPLAY SYSTEM	X	X	X	X
顯示執行緒		X	X	X
顯示追蹤	X	X	X	X
顯示使用情形		X	X	X
MOVE QLOCAL		X	X	X
Ping 通道			X	X
回復 BSDS	X	X	X	X
回復 CFSTRUCT			X	X

表 *26:* 從中執行 *MQSC* 指令的來源 *(*繼續*)*				
指令	**CSQINP1**	**CSQINP2**	**z/OS 主控台**	**指令輸入佇列及伺服器**
重新整理叢集		X	X	X
重新整理佇列管理程式		X	X	X
REFRESH SECURITY		X	X	X
重設通道			X	X
重設叢集		X	X	X
重設 QSTATS		X	X	X
重設 Tpipe			X	X
解析通道			X	X
解析不確定		X	X	X
回復佇列管理程式			X	X
RVerify 安全		X	X	X
設定保存	X	X	X	X
設定日誌	X	X	X	X
設定系統	X	X	X	X
啟動通道			X	X
開始 CHINIT		X	X	X
START CMDSERV	X	X	X	
啟動接聽器			X	X
開始佇列管理程式			X	
啟動追蹤	X	X	X	X
停止通道			X	X
停止 CHINIT			X	X
STOP CMDSERV	X	X	X	
停止接聽器			X	X
停止佇列管理程式			X	X
停止追蹤	X	X	X	X
SUSPEND 佇列管理程式			X	X

在 MQSC 指令中，每一個指令說明都會識別可從中執行該指令的來源。

z/OS IBM MQ for z/OS 的起始設定指令

起始設定指令可用來控制佇列管理程式啟動。

在佇列管理程式啟動時起始設定 IBM MQ 時，會處理起始設定輸入資料集中的指令。 可以從起始設定輸入資料集發出三種類型的指令:

- 這些指令用來定義無法在其他位置定義的 IBM MQ 實體，例如 DEFINE BUFFPOOL。

這些指令必須位於 DD 名稱 CSQINP1 所識別的資料集中。 在起始設定的重新啟動階段之前，會先處理它們。 它們無法透過主控台、作業及控制台或應用程式發出。 這些指令的回應會寫入已啟動作業程序的 CSQOUT1 陳述式中所參照的循序資料集。

- 這些指令用來定義在重新啟動之後可回復的 IBM MQ 物件。 這些定義必須在 DD 名稱 CSQINP2 所識別的資料集中指定。 它們儲存在頁集零中。 在起始設定的重新啟動階段之後，會處理 CSQINP2 。 這些指令的回應會寫入已啟動作業程序的 CSQOUT2 陳述式中所參照的循序資料集。
- 用來操作 IBM MQ 物件的指令。 這些指令也必須在 DD 名稱 CSQINP2 所識別的資料集中指定。 例如，IBM MQ 提供的範例包含 ALTER QMGR 指令，以指定子系統的無法傳送郵件的佇列。 這些指令的回應會寫入 CSQOUT2 輸出資料集。

註: 如果 IBM MQ 物件定義在 CSQINP2 中， IBM MQ 會在每次啟動佇列管理程式時嘗試重新定義它們。 如果物件已存在，則嘗試定義它們會失敗。 如果您需要在 CSQINP2 中定義物件，您可以使用 DEFINE 指令的 REPLACE 參數來避免此問題，不過，這會置換前次執行佇列管理程式期間所做的任何變更。

IBM MQ for z/OS 隨附範例起始設定資料集成員。 它們在 IBM MQ 提供的範例定義中說明。

分散式佇列作業的起始設定指令

您也可以使用 START CHINIT 指令的 CSQINP2 起始設定資料集。 如果您需要一系列其他指令來定義分散式佇列環境 (例如，啟動接聽器)，則 IBM MQ 會提供稱為 CSQINPX 的第三個起始設定輸入資料集，該資料集在通道起始程式啟動作業程序中處理。

資料集中包含的 MQSC 指令會在通道起始程式起始設定結束時執行，並將輸出寫入 CSQOUTX DD 陳述式指定的資料集。 例如，您可以使用 CSQINPX 起始設定資料集來啟動接聽器。

IBM MQ for z/OS 提供了通道起始程式起始設定資料集成員範例。 它在 IBM MQ 提供的範例定義中說明。

發佈/訂閱的起始設定指令

如果您需要一系列指令來定義發佈/訂閱環境 (例如，定義訂閱時)， IBM MQ 會提供第四個起始設定輸入資料集，稱為 CSQINPT。

資料集中包含的 MQSC 指令會在發佈/訂閱起始設定結束時執行，並將輸出寫入 CSQOUTT DD 陳述式指定的資料集。 例如，您可以使用 CSQINPT 起始設定資料集來定義訂閱。

IBM MQ for z/OS 隨附了發佈/訂閱起始設定資料集成員範例。 它在 IBM MQ 提供的範例定義中說明。

z/OS IBM MQ for z/OS 公用程式

IBM MQ for z/OS 提供一組公用程式，您可以用來協助進行系統管理。

IBM MQ for z/OS 提供一組公用程式來協助您執行各種管理作業，包括下列各項:

- 管理訊息安全原則。
- 執行備份、還原及重組作業。
- 發出指令並處理物件定義。
- 產生資料轉換結束程式。
- 修改引導資料集。
- 列出日誌的相關資訊。
- 列印日誌。
- 設定 Db2 表格及其他 Db2 公用程式。
- 處理無法傳送郵件的佇列上的訊息。

訊息安全原則公用程式

訊息安全原則公用程式 (CSQ0UTIL) 會以獨立式公用程式來執行，以管理訊息安全原則。 如需相關資訊，請參閱 訊息安全原則公用程式 (CSQ0UTIL) 。

CSQUTIL 公用程式

這是提供來協助您執行備份、還原及重組作業的公用程式。 如需相關資訊，請參閱 CSQUTIL 公用程式 。

資料轉換結束程式公用程式

IBM MQ for z/OS 資料轉換結束公用程式 (**CSQUCVX**) 作為獨立式公用程式來執行，以建立資料轉換結束常式。

變更日誌庫存公用程式

IBM MQ for z/OS 變更日誌庫存公用程式 (**CSQJU003**) 作為獨立式公用程式來執行，以變更引導資料集 (BSDS)。 您可以使用此公用程式來執行下列功能:

- 新增或刪除作用中或保存日誌資料集。
- 提供保存日誌的密碼。

列印日誌對映公用程式

IBM MQ for z/OS 列印日誌對映公用程式 (**CSQJU004**) 會以獨立式公用程式來執行，以列出下列資訊:

- 所有作用中及保存日誌資料集的兩個副本的日誌資料集名稱及日誌 RBA 關聯。 如果雙重記載不在作用中，則只有一個資料集副本。
- 可用於新日誌資料的作用中日誌資料集。
- 引導資料集 (BSDS) 中檢查點記錄佇列的內容。
- 保存日誌指令歷程記錄的內容。
- 系統及公用程式時間戳記。

日誌列印公用程式

日誌列印公用程式 (**CSQ1LOGP**) 作為獨立式公用程式執行。 您可以指定下列指令來執行公用程式:

- 引導資料集 (BSDS)
- 作用中日誌 (不含 BSDS)
- 保存日誌 (不含 BSDS)

佇列共用群組公用程式

佇列共用群組公用程式 (**CSQ5PQSG**) 作為獨立式公用程式來執行，以設定 Db2 表格，並執行佇列共用群組所需的其他 Db2 作業。

作用中日誌預先格式化公用程式

作用中日誌預先格式化公用程式 (**CSQJUFMT**) 會先格式化作用中日誌資料集，然後再由佇列管理程式使用它們。 如果公用程式已預先格式化作用中日誌資料集，則佇列管理程式第一次通過作用中日誌的日誌寫入效能會有所改善。

無法傳送郵件的佇列處理程式公用程式

無法傳送郵件的佇列處理程式公用程式 (**CSQUDLQH**) 作為獨立式公用程式執行。 它會檢查無法傳送郵件的佇列上的訊息，並根據您提供給公用程式的一組規則來處理它們。

qload 公用程式

從 IBM MQ 8.0 開始， IBM MQ Supportpac MO03 中隨附的 **qload** 公用程式已作為 **dmpmqmsg** 公用程式整合至 IBM MQ 。

在 z/OS 上，公用程式可作為可執行模組 (SCSQLOAD 檔案庫中的 CSQUDMSG) 使用別名 QLOAD ，以取得相容性。 在 SCSQPROC 中也提供範例 JCL 作為成員 CSQ4QLOD 。

z/OS IBM MQ for z/OS 的 CSQUTIL 公用程式

IBM MQ for z/OS 隨附了 CSQUTIL 公用程式，可協助您執行備份、還原及重組作業，以及發出指令和處理物件定義。

如需 CSQUTIL 公用程式的相關資訊，請參閱 IBM MQ 公用程式 (CSQUTIL)。 透過使用此公用程式，您可以呼叫下列函數:

指令
發出 MQSC 指令、記錄物件定義，以及建立用戶端通道定義檔。

COPY
讀取具名 IBM MQ for z/OS 訊息佇列的內容或具名頁集的所有佇列的內容，並將它們放入循序檔並保留原始佇列。

COPYPAGE
將整個頁面集複製到較大的頁面集。

空
若要刪除具名 IBM MQ for z/OS 訊息佇列的內容或具名頁集之所有佇列的內容，請保留佇列的定義。

格式
格式化 IBM MQ for z/OS 頁集。

LOAD
從 COPY 函數建立的循序檔還原具名 IBM MQ for z/OS 訊息佇列的內容或具名頁集的所有佇列的內容。

PAGEINFO
從一或多個頁集擷取頁集資訊。

重設頁面
將整個頁集複製到其他頁集資料集，並重設副本中的日誌資訊。

SCOPY
在佇列管理程式離線時，將佇列內容複製到資料集。

SDEFS
當佇列管理程式離線時，為物件產生一組定義指令。

SLOAD
從先前 COPY 或 SPROTICT 作業的目的地資料集還原訊息。 SLOAD 會處理單一佇列。

switch
切換或查詢與叢集傳送端通道相關聯的傳輸佇列。

XPARM
將通道起始程式參數載入模組轉換為佇列管理程式屬性 (用於移轉)。

z/OS 操作 IBM MQ for z/OS

使用這些基本程序來操作 IBM MQ for z/OS。

您也可以使用 IBM MQ Explorer(與 IBM MQ for Windows 及 IBM MQ for Linux (x86 及 x86-64 平台)一起配送) 來執行本節中所說明的作業。 獨立式 IBM MQ Explorer 可從 Fix Central 下載。 如需相關資訊，請參閱第 101 頁的『使用 IBM MQ Explorer 進行管理』。

z/OS 在 z/OS 上發出佇列管理程式指令

您可以從 z/OS 主控台或使用公用程式 CSQUTIL 來發出 IBM MQ 控制指令。 指令可以使用指令字首字串 (CPF) 來指出哪個 IBM MQ 子系統處理指令。

您可以使用 IBM MQ 指令來控制 IBM MQ 的大部分作業環境。 IBM MQ for z/OS 同時支援這些指令的 MQSC 及 PCF 類型。 本主題說明如何使用 MQSC 指令來指定屬性，因此它會使用 MQSC 指令名稱而非 PCF 名稱來參照那些指令及屬性。 如需 MQSC 指令語法的詳細資料，請參閱 MQSC 指令。 如需 PCF 指令語法的詳細資料，請參閱 第 22 頁的『使用 IBM MQ 可程式指令格式』。 如果您是適當授權的使用者，則可以從下列發出 IBM MQ 指令:

- 起始設定輸入資料集 (如 第 360 頁的『IBM MQ for z/OS 的起始設定指令』 所述)。
- z/OS 主控台或對等項目，例如 SDSF
- z/OS 主要 get 指令常式 MGCRE (SVC 34)
- IBM MQ 公用程式 CSQUTIL (在 IBM MQ 公用程式中說明。)
- 使用者應用程式，可以是:
  - CICS 程式
  - TSO 程式
  - z/OS 批次程式
  - IMS 程式

  如需相關資訊，請參閱 第 381 頁的『撰寫程式以管理 IBM MQ for z/OS』。

這些命令的大部分功能都是透過操作和控制面板以方便的方式提供的，可從 TSO 和 ISPF 訪問，並在第 369 頁的『IBM MQ for z/OS 的作業及控制面板』中進行了描述。

如需進一步資訊，請參閱

- 第 364 頁的『從 z/OS 主控台或其對等項目發出指令』
  - 指令字首字串
  - 使用 z/OS 主控台發出指令
  - 指令回應
- 從公用程式 CSQUTIL 發出指令

從 z/OS 主控台或其對等項目發出指令

您可以從 z/OS 主控台或其對等項目發出所有 IBM MQ 指令。 您也可以從可以發出 z/OS 指令 (例如 SDSF) 的任何地方發出 IBM MQ 指令，或透過使用 MGCRE 巨集的程式來發出。

在主控台鍵入指令後可顯示的資料數量上限為 32 KB。

註:

1. 您無法從 IMS 終端機使用 IMS/SSR 指令格式來發出 IBM MQ 指令。 IMS 配接器不支援此功能。
2. SDSF 提供的輸入欄位對於部分指令 (特別是通道的那些指令) 可能不夠長。

指令字首字串

每一個 IBM MQ 指令都必須以指令字首字串 (CPF) 作為字首，如 第 365 頁的圖 32 所示。

因為多個 IBM MQ 子系統可以在 z/OS 下執行，所以 CPF 用來指出哪個 IBM MQ 子系統處理指令。 例如，針對稱為 CSQ1 的子系統啟動佇列管理程式，其中 CPF 是 ' +CSQ1 ' ，您從操作員主控台發出指令 +CSQ1 START QMGR。 此 CPF 必須定義在子系統名稱表格中 (針對子系統 CSQ1)。 這在 定義指令字首字串 (CPF)中有說明。 在範例中，字串 ' +CSQ1 ' 作為指令字首。

使用 z/OS 主控台來發出指令

您可以從 z/OS 主控台鍵入簡式指令，例如 第 365 頁的圖 32 中的 DISPLAY 指令。 不過，對於複式指令或您經常發出的指令集，發出指令的其他方法會比較好。

```
+CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLOCAL)
```

圖 *32:* 從 *z/OS* 主控台發出 *DISPLAY* 指令

指令回應

指令的直接回應會傳送至發出指令的主控台。 IBM MQ 支援 z/OS 中提供的 延伸主控台支援 (EMCS) 功能，因此可以使用具有 4 位元組 ID 的主控台。 此外，當指令由使用 MGCRE 巨集的程式發出時，除了 START QMGR 及 STOP QMGR 以外的所有指令都支援使用「指令及回應記號 (CART)」。

從公用程式 CSQUTIL 發出指令

您可以使用公用程式 CSQUTIL 的 COMMAND 函數，從循序資料集發出指令。 此公用程式會將指令以訊息形式傳送至 系統指令輸入佇列 ，並等待回應，該回應與 SYSPRINT 中的原始指令一起列印。 如需詳細資料，請參閱 IBM MQ 公用程式。

z/OS 在 *z/OS* 上啟動及停止佇列管理程式

請使用本主題作為停止及啟動佇列管理程式的簡介。

本節說明如何啟動和停止佇列管理程式。 它包含下列主題的相關資訊:

- 第 365 頁的『開始之前 IBM MQ』
- 第 365 頁的『啟動佇列管理程式』
- 第 367 頁的『停止佇列管理程式』

啟動和停止佇列管理程式相對直接明確。 當佇列管理程式在正常狀況下停止時，其最後一個動作是採取終止檢查點。 此檢查點及日誌會提供佇列管理程式重新啟動所需的資訊。

本節包含 START 及 STOP 指令的相關資訊，並包含發生異常終止之後啟動的簡要概觀。

開始之前 IBM MQ

安裝 IBM MQ 之後，它會定義為正式 z/OS 子系統。 在 z/OS 的任何起始程式載入 (IPL) 期間會出現此訊息:

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm INITIALIZATION COMPLETE
```

其中 *ssnm* 是 IBM MQ 子系統名稱。

從現在開始，您可以從 已獲授權發出系統控制指令的任何 *z/OS* 主控台 ，即 z/OS SYS 指令群組，來啟動該子系統的佇列管理程式。 您必須從授權主控台發出 START 指令，無法透過 JES 或 TSO 來發出。

如果您使用佇列共用群組，則必須先啟動 RRS ，然後再啟動 Db2，然後再啟動佇列管理程式。

啟動佇列管理程式

您可以發出 START QMGR 指令來啟動佇列管理程式。 不過，除非您具有適當的權限，否則無法順利使用 START 指令。 如需 IBM MQ 安全的相關資訊，請參閱 在 z/OS 上設定安全 。 第 366 頁的圖 33 顯示 START 指令的範例。 (請記住，您必須在 IBM MQ 指令前面加上指令字首字串 (CPF)。)

```
+CSQ1  START QMGR
+CSQ1  START QMGR PARM(NEWLOG)
```

圖 *33:* 從 *z/OS* 主控台啟動佇列管理程式

如需 START QMGR 指令語法的相關資訊，請參閱 START QMGR 。

您無法以批次工作形式執行佇列管理程式，或使用 z/OS 指令 START 來啟動它。 這些方法可能會啟動 IBM MQ 的位址空間，然後異常結束。 您也無法從 CSQUTIL 公用程式或類似使用者應用程式啟動佇列管理程式。

不過，您可以透過將 START QMGR 指令傳遞至 z/OS MGCRE (SVC 34) 服務，從 APF 授權程式啟動佇列管理程式。

如果您使用佇列共用群組，當您啟動佇列管理程式時，相關聯的 Db2 系統和 RRS 必須在作用中。

啟動選項

當您啟動佇列管理程式時，會載入系統參數模組。 您可以使用下列兩種方式之一來指定系統參數模組的名稱:

- 使用 /cpf START QMGR 指令的 PARM 參數，例如

```
/cpf START QMGR PARM(CSQ1ZPRM)
```

- 使用啟動程序中的參數，例如，將 JCL EXEC 陳述式撰寫為

```
//MQM  EXEC PGM=CSQYASCP,PARM='ZPARM(CSQ1ZPRM)'
```

系統參數模組提供自訂佇列管理程式時指定的資訊。

從 IBM MQ 9.1.0 開始，您可以使用 **QMGRPROD** 選項來指定要針對其記錄佇列管理程式使用情形的產品，並使用 **AMSPROD** 選項來指定 AMS 的對等項目 (如果使用的話)。 如需允許值的詳細資料，請參閱 MQSC START QMGR 指令。

JCL EXEC 陳述式範例如下:

```
//MQM EXEC  PGM=CSQYASCP,PARM='QMGRPROD(MQ)'
```

如需產品用法錄製的相關資訊，請參閱 z/OS MVS 產品管理。

您也可以使用 ENVPARM 選項，將 JCL 程序中的一或多個參數替換為佇列管理程式。

例如，您可以更新佇列管理程式啟動程序，讓 DDname CSQINP2 成為變數。 這表示您可以在不變更啟動程序的情況下變更 CSQINP2 DDname。 這適用於實作變更，為操作員提供取消，以及佇列管理程式作業。

假設佇列管理程式 CSQ1 的啟動程序看起來像 第 367 頁的圖 34。

```
//CSQ1MSTR PROC INP2=NORM
//MQMESA  EXEC PGM=CSQYASCP
//STEPLIB DD  DISP=SHR,DSN=thlqual.SCSQANLE
//     DD  DISP=SHR,DSN=thlqual.SCSQAUTH
//     DD  DISP=SHR,DSN=db2qual.SDSNLOAD
//BSDS1  DD  DISP=SHR,DSN=myqual.BSDS01
//BSDS2  DD  DISP=SHR,DSN=myqual.BSDS02
//CSQP0000 DD  DISP=SHR,DSN=myqual.PSID00
//CSQP0001 DD  DISP=SHR,DSN=myqual.PSID01
//CSQP0002 DD  DISP=SHR,DSN=myqual.PSID02
//CSQP0003 DD  DISP=SHR,DSN=myqual.PSID03
//CSQINP1 DD  DISP=SHR,DSN=myqual.CSQINP(CSQ1INP1)
//CSQINP2 DD  DISP=SHR,DSN=myqual.CSQINP(CSQ1&INP2.)
//CSQOUT1 DD  SYSOUT=*
//CSQOUT2 DD  SYSOUT=*
```

圖 *34:* 啟動程序範例

如果您接著使用下列指令來啟動佇列管理程式:

```
+CSQ1  START QMGR
```

使用的 CSQINP2 是稱為 CSQ1NORM 的成員。

不過，假設您要將新的程式套組放入正式作業，以便下次啟動佇列管理程式 CSQ1 時，會從成員 CSQ1NEW 取得 CSQINP2 定義。 若要這樣做，您可以使用下列指令來啟動佇列管理程式:

```
+CSQ1  START QMGR ENVPARM('INP2=NEW')
```

且會使用 CSQ1NEW 來取代 CSQ1NORM。 附註: z/OS 會將符號參數的 KEYWORD=value 規格 (如 INP2=NEW) 限制為 255 個字元。

在異常終止之後啟動

IBM MQ 會自動偵測在正常關機或異常終止之後重新啟動。

在異常結束之後啟動佇列管理程式不同於在發出 STOP QMGR 指令之後啟動佇列管理程式。 在 STOP QMGR 之後，系統會依序完成其工作，並在停止之前取得終止檢查點。 當您重新啟動佇列管理程式時，它會使用來自系統檢查點及回復日誌的資訊來判定關機時的系統狀態。

不過，如果佇列管理程式異常結束，則它會終止，而無法完成其工作或取得終止檢查點。 當您在異常終止之後重新啟動佇列管理程式時，它會使用日誌中的資訊來重新整理其在終止時狀態的知識，並通知您各種作業的狀態。 通常，重新啟動程序會解決所有不一致狀態。 但是，在某些情況下，您必須採取特定步驟來解決不一致。

啟動時的使用者訊息

當您順利啟動佇列管理程式時，佇列管理程式會產生一組啟動訊息。

停止佇列管理程式

在停止佇列管理程式之前，所有 IBM MQ 相關的寫給操作員並回覆 (WTOR) 訊息都必須接收回覆，例如，取得日誌要求。 第 368 頁的圖 35 中的每一個指令都會終止執行中的佇列管理程式。

```
+CSQ1  STOP QMGR
+CSQ1  STOP QMGR MODE(QUIESCE)
+CSQ1  STOP QMGR MODE(FORCE)

+CSQ1  STOP QMGR MODE(RESTART)
```

圖 *35:* 停止佇列管理程式

指令 STOP QMGR 預設為 STOP QMGR MODE (QUIESCE)。

在 QUIESCE 模式中， IBM MQ 不容許建立任何新的連線執行緒，但容許現有的執行緒繼續; 只有在所有執行緒都已結束時，它才會終止。 應用程式可以要求在佇列管理程式靜止時收到通知。 因此，請儘可能使用 QUIESCE 模式，讓已要求通知的應用程式有機會中斷連線。 如需詳細資料，請參閱 終止期間發生的狀況 。

如果佇列管理程式未在合理時間內終止以回應 STOP QMGR MODE (QUIESCE) 指令，請使用 DISPLAY CONN 指令來判斷是否存在任何連線執行緒，並採取必要步驟來終止相關聯的應用程式。 如果沒有執行緒，請發出 STOP QMGR MODE (FORCE) 指令。

STOP QMGR MODE (QUIESCE E) 及 STOP QMGR MODE (FORCE) 指令會從 MVS 自動重新啟動管理程式 (ARM) 取消登錄 IBM MQ ，防止 ARM 自動重新啟動佇列管理程式。 STOP QMGR MODE (RESTART) 指令的運作方式與 STOP QMGR MODE (FORCE) 指令相同，但它不會從 ARM 取消登錄 IBM MQ 。 這表示佇列管理程式可以立即自動重新啟動。

如果 IBM MQ 子系統未向 ARM 登錄，則會拒絕 STOP QMGR MODE (RESTART) 指令，並將下列訊息傳送至 z/OS 主控台:

```
CSQY205I ARM element arm-element is not registered
```

如果未發出此訊息，則會自動重新啟動佇列管理程式。 如需 ARM 的相關資訊，請參閱 第 431 頁的『使用 z/OS Automatic Restart Manager (ARM)』。

只有在 STOP QMGR MODE (FORCE) 未終止佇列管理程式時，才會取消佇列管理程式位址空間。

如果透過取消位址空間或使用指令 STOP QMGR MODE (FORCE) 來停止佇列管理程式，則會維護與已連接 CICS 或 IMS 系統的一致性。 資源的重新同步會在佇列管理程式重新啟動時啟動，並在建立與 CICS 或 IMS 系統的連線時完成。

註: 當您停止佇列管理程式時，可能會發現發出 IEF352I 訊息。 如果 z/OS 偵測到未將位址空間標示為無法使用會導致完整性暴露，則會發出此訊息。 您可以忽略這個訊息。

停止訊息

發出 STOP QMGR 指令之後，您會取得訊息 CSQY009I 及 CSQY002I，例如:

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM
USER(userid), STOP MODE(FORCE)
CSQY002I +CSQ1 QUEUE MANAGER STOPPING
```

其中 `userid` 是發出 STOP QMGR 指令的使用者 ID， MODE 參數取決於指令中指定的使用者 ID。

當 STOP 指令順利完成時， z/OS 主控台上會顯示下列訊息:

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

如果您使用 ARM ，且未指定 MODE (RESTART) ，則也會顯示下列訊息:

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type
arm-element-type successful
```

在顯示下列訊息之前，您無法重新啟動佇列管理程式:

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```

z/OS IBM MQ for z/OS 的作業及控制面板

您可以使用 IBM MQ 作業及控制台，對 IBM MQ 物件執行管理作業。 請使用本主題作為指令及控制台的簡介。

您可以使用這些畫面來定義、顯示、變更或刪除 IBM MQ 物件。 使用這些畫面來進行日常管理，以及對物件進行小型變更。 如果您要設定或變更許多物件，請使用 CSQUTIL 公用程式的 COMMAND 函數。

作業及控制台支援通道起始程式 (例如，啟動通道或 TCP/IP 接聽器)、叢集作業及安全的控制項。 它們也可讓您顯示執行緒及頁集使用情形的相關資訊。

這些畫面的運作方式是透過系統指令輸入佇列，將 MQSC 類型 IBM MQ 指令傳送至佇列管理程式。

註:

1. IBM MQ for z/OS 作業及控制台 (CSQOREXX) 可能不支援從第 7 版開始新增的所有新功能及參數。 例如，主題物件或訂閱沒有直接操作的畫面。

   使用下列其中一個支援的機制，可讓您管理發佈/訂閱定義及其他無法直接從其他畫面使用的其他系統控制項:

   a. IBM MQ 瀏覽者

   b. z/OS 主控台

   c. 可程式指令格式 (PCF) 訊息

   d. CSQUTIL 的 COMMAND 函數

   e. IBM MQ Web 主控台

   請注意， CSQOREXX 畫面中的一般 **Command** 動作可讓您發出任何有效的 MQSC 指令，包括 SMDS 相關指令。 您可以使用 CSQUTIL 的 COMMAND 函數所發出的所有指令。
2. 您無法直接從畫面中的指令行發出 IBM MQ 指令。
3. 如果要使用作業和控制台，您必須具備正確的安全授權; 這在 指令安全和指令資源安全的使用者 ID 中有說明。
4. 您無法使用 CSQUTIL 或 CSQOREXX 畫面來提供使用者 ID 和密碼。 相反地，如果您的使用者 ID 對 MQCONN 中的 BATCH 設定檔具有 UPDATE 權限，則可以略過 **CHCKLOCL**(*REQUIRED* 設定。 如需相關資訊，請參閱 在本端連結的應用程式上使用 **CHCKLOCL** 。

z/OS 作業及控制台的呼叫及規則

您可以透過 ISPF 面板控制 IBM MQ 並發出控制命令。

如何存取 IBM MQ 作業及控制台

如果 IBM MQ 的 ISPF /PDF 主選項功能表已更新，您可以從該功能表存取 IBM MQ 操作和控制台。 如需更新功能表的詳細資料，請參閱 作業 20: 設定作業及控制台。

您可以從 TSO 命令處理器面板（通常是 ISPF /PDF 主選項選單上的選項 6）存取 IBM MQ 操作和控制面板。 您執行以執行此動作的執行程式名稱是 CSQOREXX。 它有兩個參數； `thlqual` 是要使用的 IBM MQ 函式庫的高階限定符， `langletter` 是標識要使用的國家語言庫的字母（例如，E 代表 U.S . 英文)。 如果 IBM MQ 庫永久安裝在您的 ISPF 設定中，則可以省略這些參數。 或者，您可以從 TSO 指令行發出 CSQOREXX。

這些面板設計為操作員及管理者使用，並提供最少的正式訓練。 在畫面執行的情況下閱讀這些指示，並試用建議的不同作業。

註: 使用畫面時，暫時動態佇列的名稱格式為 SYSTEM.CSQOREXX.* 已建立。

作業及控制面板的規則

請參閱 IBM MQ 物件的命名規則，以瞭解 IBM MQ 字串和名稱的一般規則。 不過，有些規則只適用於作業和控制台:

- 請勿以單引號或雙引號括住字串，例如說明。
- 如果您在文字欄位中併入單引號或引號，則不需要重複它或新增跳出字元。 字元的儲存方式與您鍵入的字元完全相同; 例如:

  ```
  This is Maria's queue
  ```

  畫面處理器會將它們加倍，讓您將它們傳遞至 IBM MQ。 不過，如果它必須截斷您的資料才能執行此動作，則會執行此動作。
- 您可以在大部分欄位中使用大寫或小寫字元，當您按 Enter 鍵時，它們會變成大寫字元。 異常狀況如下:
  - 儲存類別名稱和連結機能結構名稱，必須以大寫 A 到 Z 開頭，後面接著大寫 A 到 Z 或數值字元。
  - 未翻譯的特定欄位。 這些包括:
    - 應用程式 ID
    - 說明
    - 環境資料
    - 物件名稱 (但如果您使用小寫物件名稱，則可能無法在 z/OS 主控台輸入它)
    - 遠端系統名稱
    - 觸發資料
    - 使用者資料
- 在名稱中，會忽略前導空白和前導底線。 因此，物件名稱不能以空白或底線開頭。
- 底線用來顯示空白欄位的範圍。 當您按 Enter 鍵時，尾端底線會取代為空白。
- 許多說明和文字欄位會呈現在多個組件中，每一個組件由 IBM MQ 獨立處理。 這表示會保留尾端空白，且文字不連續。

空白欄位

當您指定 IBM MQ 物件的 **定義** 動作時，定義畫面上的每一個欄位都包含一個值。 如需 IBM MQ 取得值之位置的相關資訊，請參閱顯示畫面的一般說明 (延伸說明)。 如果您鍵入含有空白的欄位，且不容許空白，則 IBM MQ 會將安裝預設值放置在欄位中，或提示您輸入必要值。

當您為 IBM MQ 物件指定 **變更** 動作時，變更畫面上的每一個欄位都會包含該欄位的現行值。 如果您鍵入含有空白的欄位，且不容許空白，則該欄位的值不會變更。

z/OS *z/OS 上的物件及動作*

作業及控制台提供您許多不同類型的物件，以及您可以對它們執行的許多動作。

這些動作會列在起始畫面上，可讓您操作物件並顯示它們的相關資訊。 這些物件包括所有 IBM MQ 物件，以及一些額外的物件。 物件屬於下列種類。

- 佇列、處理程序、鑑別資訊物件、名稱清單、儲存類別及 CF 結構
- 通道
- 叢集物件

- 佇列管理程式及安全
- 連線
- 系統

請參閱 動作，以取得可對 IBM MQ 物件採取之動作的交互參照表格。

佇列、處理程序、鑑別資訊物件、名稱清單、儲存類別及 CF 結構

這些是基本 IBM MQ 物件。 每一種類型可以有許多種。 可以使用 LIST 或 DISPLAY、LIST 及 FILTER、DEFINE LIKE、MANAGE 和 ALTER 動作來列出、列出過濾器、定義及刪除它們，並具有可顯示及變更的屬性。 (使用 MANAGE 動作刪除物件。)

此種類由下列物件組成:

QLOCAL	本端佇列
QREMOTE	遠端佇列
QALIAS	佇列間接參照的別名佇列
QMODEL	動態定義佇列的模型佇列
佇列	任何類型的佇列
QSTATUS	本端佇列的狀態
PROCESS	發生觸發事件時要啟動之應用程式的相關資訊
AUTHINFO	鑑別資訊: 使用 LDAP 伺服器執行「憑證撤銷清冊 (CRL)」檢查所需的定義
名稱清單	名稱清單，例如佇列或叢集
STGCLASS	儲存體類別
CFSTRUCT	連結機能 (CF) 結構
CFSTATUS	CF 結構的狀態

通道

通道用於分散式佇列作業。 每一種類型可以有許多種，它們可以列出、列出，以及過濾器、已定義、已刪除、已顯示及已變更。 它們也可以使用 START、STOP 和 PERFORM 動作來使用其他功能。 PERFORM 提供重設、連線測試及解析通道功能。

此種類由下列物件組成:

CHANNEL	任何類型的通道
傳送者	傳送端通道
SERVER	伺服器通道
接收者	接收端通道
要求端	要求端通道
CLUSRCVR	叢集接收端通道
CLUSSDR	叢集傳送端通道
SVRCONN	伺服器連線通道
CLNTCONN	用戶端連線通道
CHSTATUS	通道連線的狀態

叢集物件

叢集物件是針對屬於叢集的佇列及通道自動建立的。 基本佇列及通道定義可以位於另一個佇列管理程式上。 每一種類型可以有許多種，且名稱可以重複。 它們可以列出、與過濾器一起列出，以及顯示。 PERFORM、START 及 STOP 也可以透過 LIST 動作來使用。

此種類由下列物件組成:

CLUSQ	叢集佇列，針對屬於叢集的佇列所建立
CLUSCHL	叢集通道，針對屬於叢集的通道建立
CLUSQMGR	叢集佇列管理程式，與叢集通道相同，但由其佇列管理程式名稱識別

叢集通道和叢集佇列管理程式確實有 PERFORM、START 和 STOP 動作，但只能透過 DISPLAY 動作間接執行。

佇列管理程式及安全

佇列管理程式及安全物件具有單一實例。 它們可以列出，並具有可顯示及變更的屬性 (使用 LIST 或 DISPLAY ，以及 ALTER 動作) ，以及具有可使用 PERFORM 動作的其他功能。

此種類由下列物件組成:

管理員	佇列管理程式 :PERFORM 動作提供暫停及回復叢集功能
安全	安全功能 :PERFORM 動作提供重新整理及重新驗證功能

連線

可以列出連線，並列出過濾器及顯示連線。

此種類僅包含連線物件 CONNECT。

系統

其他函數的集合。 此種類由下列物件組成:

系統	系統功能
CONTROL	SYSTEM 的同義字

可用的功能如下:

LIST 或 DISPLAY	顯示佇列共用群組、分散式佇列、頁集或資料集使用情形資訊。
執行	重新整理或重設叢集作業
開始	啟動通道起始程式或接聽器
停止	停止通道起始程式或接聽器

動作

下表顯示您可以對每一種物件類型執行的動作:

表 *27: IBM MQ* 物件的有效作業及控制面板動作

物件	變更	定義相似	管理 (1)	清單或顯示畫面	含有過濾器的清單	執行	開始	停止(S)
AUTHINFO	X	X	X	X	X			
CFSTATUS				X				
CFSTRUCT	X	X	X	X	X			
CHANNEL	X	X	X	X	X	X	X	X
CHSTATUS				X	X			
CLNTCONN	X	X	X	X	X			
CLUSCHL				X	X	X (2)	X (2)	X (2)

表 27: *IBM MQ* 物件的有效作業及控制面板動作 (繼續)								
物件	**變更**	**定義相似**	**管理 (1)**	**清單或顯示畫面**	**含有過濾器的清單**	**執行**	**開始**	**停止(S)**
CLUSQ				X	X			
CLUSQMGR				X	X	X (2)	X (2)	X (2)
CLUSRCVR	X	X	X	X	X	X	X	X
CLUSSDR	X	X	X	X	X	X	X	X
CONNECT				X	X			
CONTROL				X		X	X	X
管理員	X			X		X		
名稱清單	X	X	X	X	X			
PROCESS	X	X	X	X	X			
QALIAS	X	X	X	X	X			
QLOCAL	X	X	X	X	X			
QMODEL	X	X	X	X	X			
QREMOTE	X	X	X	X	X			
QSTATUS				X	X			
佇列	X	X	X	X	X			
接收者	X	X	X	X	X	X	X	X
要求端	X	X	X	X	X	X	X	X
安全	X			X		X		
傳送者	X	X	X	X	X	X	X	X
SERVER	X	X	X	X	X	X	X	X
SVRCONN	X	X	X	X	X		X	X
STGCLASS	X	X	X	X	X			
系統				X		X	X	X

附註:

1. 提供 **刪除** 及其他功能
2. 使用 **清單或顯示** 動作

z/OS *z/OS* 上的物件處置

您可以指定需要使用之物件的 處置 。 disposition 表示物件 **定義** 的保留位置，以及物件的行為方式。

只有在您使用下列任何物件類型時，此處置才有意義:

- 佇列
- 通道
- 程序
- 名單
- 儲存類別

- 鑑別資訊物件

如果您使用其他物件類型，則不會處理處置。

允許的值如下：

Q
QMGR。 物件定義位於佇列管理程式的頁集上，且只能由佇列管理程式存取。

C
收到 物件定義位於佇列管理程式的頁集上，且只能由佇列管理程式存取。 它們是定義為具有 GROUP 處置的物件本端副本。

P
專用。 物件定義位於佇列管理程式的頁集上，且只能由佇列管理程式存取。 物件已定義為具有 QMGR 或 COPY 處置。

G
GROUP。 物件定義位於共用儲存庫中，且可供佇列共用群組中的所有佇列管理程式存取。

S
共用。 此處置僅適用於本端佇列。 佇列定義位於共用儲存庫中，且可供佇列共用群組中的所有佇列管理程式存取。

A
好吧 如果動作佇列管理程式是目標佇列管理程式或 * ，則會併入 **所有** 處置的物件; 否則，只會併入 QMGR 及 COPY 處置的物件。 這是預設值。

z/OS 在 *z/OS* 上使用 *ISPF* 控制台來選取佇列管理程式、預設值及層次

您可以在 ISPF 中使用 CSQOREXX 執行程式來控制佇列管理程式。

當您檢視起始畫面時，未連接任何佇列管理程式。 不過，只要按 Enter 鍵，就會連接至佇列管理程式，或連接至 **連接名稱** 欄位中所指名之佇列共用群組中的佇列管理程式。 您可以將此欄位保留空白; 這表示您正在使用批次應用程式的預設佇列管理程式。 這在 CSQBDEFV 中定義 (如需相關資訊，請參閱 作業 19: 設定批次、TSO 和 RRS 配接器)。

使用 **目標佇列管理程式** 欄位來指定要執行您所要求動作的佇列管理程式。 如果將此欄位保留空白，則會預設為 **連接名稱** 欄位中指定的佇列管理程式。 您可以指定不是您所連接的目標佇列管理程式。 在此情況下，您通常會指定遠端佇列管理程式物件的名稱，該物件提供佇列管理程式別名定義 (開啟指令輸入佇列時，該名稱會用作 *ObjectQMgr* 名稱)。 如果要這麼做，您必須設定適當的佇列和通道來存取遠端佇列管理程式。

動作佇列管理程式 欄位可讓您指定佇列管理程式，該佇列管理程式與 **目標佇列管理程式** 欄位中指定的佇列管理程式位於相同的佇列共用群組中，成為要執行您要求之動作的佇列管理程式。 如果您在這個欄位中指定 * ，則會對佇列共用群組中的所有佇列管理程式執行您要求的動作。 如果將此欄位保留空白，則會預設為 **目標佇列管理程式** 欄位中指定的值。 **動作佇列管理程式** 欄位對應於使用 MQSC 指令中說明的 CMDSCOPE 指令修飾元。

佇列管理程式預設值

如果您將任何佇列管理程式欄位留為空白，或選擇連接至佇列共用群組，則當您按 **Enter** 鍵時，會開啟次要視窗。 這個視窗會確認您將使用的佇列管理程式名稱。 按 **Enter** 鍵繼續。 當您在提出一些要求之後回到起始畫面時，您會找到以實際名稱完成的欄位。

佇列管理程式層次

只有在 IBM WebSphere MQ 710 或更新版本的 z/OS 上執行佇列管理程式時，「作業」和「控制」畫面才能令人滿意地運作。

如果不符合這些條件，則動作可能只能局部、不正確或完全無法運作，且無法辨識來自佇列管理程式的回覆。

如果動作佇列管理程式不在 IBM MQ 8.0.0 或以上，則不會顯示部分欄位，且無法輸入部分值。 不允許一些物件和動作。 在這種情況下，會開啟次要視窗，要求您確認要繼續進行。

z/OS 搭配使用功能鍵和指令行與 *z/OS* 上的 *ISPF* 控制台

如果要使用畫面，您必須使用功能鍵，或在 ISPF 控制面板指令區中輸入對等指令。

- 功能鍵
  - 處理您的動作
  - 第 375 頁的『顯示 IBM MQ 使用者訊息』
  - 取消動作
  - 取得說明
- 使用指令行

功能鍵

功能鍵具有 IBM MQ 的特殊設定。 (這表示您無法對功能鍵使用 ISPF 預設值; 如果您先前在任何位置使用 KEYLIST OFF ISPF 指令，則必須在任何作業及控制台的指令區域中鍵入 KEYLIST ON，然後按 Enter 鍵以啟用 IBM MQ 設定。)

這些功能鍵設定可以顯示在畫面上，如 第 376 頁的圖 36 所示。 如果未顯示設定，請在任何作業及控制台的指令區域中鍵入 PFSHOW，然後按 **Enter** 鍵。 若要移除設定的顯示，請使用指令 PFSHOW OFF。

作業及控制面板中的功能鍵設定符合 CUA 標準。 雖然您可以透過一般 ISPF 程序 (例如 **KEYLIST** 公用程式) 來變更金鑰設定，但不建議您這麼做。

註: 使用 **PFSHOW** 和 **KEYLIST** 指令會影響您擁有的任何其他邏輯 ISPF 畫面，當您離開作業和控制面板時，它們的設定仍會保留。

處理您的動作

按 **Enter** 鍵以在畫面上執行所要求的動作。 畫面中的資訊會傳送至佇列管理程式進行處理。

每次在畫面中按 **Enter** 鍵時， IBM MQ 會產生一或多個操作員訊息。 如果作業成功，您會收到確認訊息 CSQ9022I，否則會收到一些錯誤訊息。

顯示 IBM MQ 使用者訊息

在任何畫面中按功能鍵 F10，以查看 IBM MQ 使用者訊息。

取消動作

在起始畫面上， F3 和 F12 會結束作業和控制面板，並讓您回到 ISPF。 不會將任何資訊傳送至佇列管理程式。

在任何其他畫面上，按功能鍵 F3 或 F12，以離開現行畫面 **忽略自前次按 Enter 鍵以來所鍵入的任何資料**。 同樣地，不會將任何資訊傳送至佇列管理程式。

- F3 會直接回到起始畫面。
- F12 帶您回到前一個畫面。

取得說明

每一個畫面都有相關聯的說明畫面。 說明畫面使用 ISPF 通訊協定:

- 在任何畫面上按功能鍵 F1，以查看作業的一般說明 (延伸說明)。
- 按功能鍵 F1，游標停在任何欄位上，以查看該欄位的特定說明。
- 從任何欄位說明畫面中按功能鍵 F5，以取得一般說明。
- 按功能鍵 F3，回到基本畫面，亦即您按下功能鍵 F1 的畫面。
- 從任何說明畫面中按功能鍵 F6，以取得功能鍵的說明。

如果說明資訊進入第二個或後續頁面，畫面右上方會顯示 **其他** 指示器。 請使用下列功能鍵來導覽說明頁面:

- F11，以進入下一個說明頁面 (如果有的話)。
- F10，以回到前一個說明頁面 (如果有的話)。

使用指令行

您絕不需要使用指令行來發出作業及控制面板所使用的指令，因為它們可從功能鍵取得。 所提供的指令行可讓您輸入一般 ISPF 指令 (例如 **PFSHOW**)。

ISPF 指令 PANELID ON 會顯示現行 CSQOREXX 畫面的名稱。

不論您有哪些 ISPF 設定，指令行一開始都會顯示在畫面底端的預設位置。 您可以從任何作業及控制面板使用 SETTINGS ISPF 指令來變更指令行的位置。 這些設定會針對作業及控制台的後續階段作業而被記住。

z/OS 在 z/OS 上使用作業及控制面板

請利用這個主題來調查從 CSQOREXX 顯示的起始控制面板

第 376 頁的圖 36 顯示當您啟動畫面階段作業時所顯示的畫面。

```
                    IBM MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . . 1     0. List with filter   4. Manage
                                    1. List or Display    5. Perform
                                    2. Define like        6. Start
                                    3. Alter              7. Stop
                                    8. Command
Object type . . . . . . . . . CHANNEL       +
Name  . . . . . . . . . . . . *
Disposition . . . . . . . . . A  Q=Qmgr, C=Copy, P=Private, G=Group,
                                 S=Shared, A=All

Connect name  . . . . . . . . MQ1C  - local queue manager or group
Target queue manager  . . . MQ1C
          - connected or remote queue manager for command input
Action queue manager  . . . MQ1C  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993, 2025. All rights reserved.

Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

圖 *36: IBM MQ* 作業及控制起始畫面

從這個畫面中，您可以執行下列動作:

- 選擇您想要的本端佇列管理程式，以及您是要在該佇列管理程式、遠端佇列管理程式，還是在與本端佇列管理程式相同佇列共用群組中的另一個佇列管理程式上發出指令。 如果您需要變更佇列管理程式名稱，請改寫它。
- 在 **動作** 欄位中鍵入適當的數字，以選取您要執行的動作。
- 指定您要使用的物件類型。 如果您不確定物件類型為何，請按功能鍵 F1 以取得物件類型的說明。
- 指定您要使用之物件類型的處置方式。
- 顯示指定類型的物件清單。 在 **名稱** 欄位中鍵入星號 (*)，然後按 **Enter** 鍵，以顯示已在動作佇列管理程式上定義的物件 (指定類型) 清單。 然後，您可以依序選取一或多個要使用的物件。 所有動作都可以從清單中取得。

註: 建議您選擇會導致顯示物件清單的選項，然後從該清單中執行。 請使用 **顯示** 動作，因為所有物件類型都容許這樣做。

z/OS 在 z/OS 上使用指令機能

使用編輯器來輸入或修正要傳遞至佇列管理程式的 MQSC 指令。

從主要畫面 CSQOPRIA 中，選取選項 **8 指令**，以啟動「指令機能」。

您會看到循序檔 *prefix*.CSQUTIL.COMMANDS，用作 CSQUTIL COMMAND 函數的輸入; 請參閱 將指令發出至 IBM MQ。

您不需要以指令字首字串 (CPF) 作為指令的字首。

您可以透過以接續字元 **+** 或 **-**終止現行行，在後續行上繼續 MQSC 指令。 或者，使用行編輯模式來提供長 MQSC 指令或指令內長屬性值的值。

行編輯

若要使用行編輯，請將游標移至編輯畫面中的適當行，並使用 **F4** 在可捲動畫面中顯示單一行。 單行最多可以有 32 760 個位元組的資料。

若要保留行編輯，請執行下列動作:

- **F3 結束程式** 會儲存對行和結束程式所做的變更
- **F12 取消** 會回到編輯畫面，捨棄對行所做的變更。

若要捨棄在編輯階段作業中所做的變更，請使用 **F12 取消** 來終止編輯階段作業，讓檔案的內容保持不變。 未執行指令。

執行指令

當您完成輸入 MQSC 指令時，請使用 **F3 結束** 來終止編輯階段作業，以儲存檔案的內容，並呼叫 CSQUTIL 以將指令傳遞至佇列管理程式。 指令處理的輸出保留在檔案 *prefix*.CSQUTIL.OUTPUT。 編輯階段作業會自動開啟此檔案，讓您可以檢視回應。 按 **F3 結束** 以結束此階段作業，並回到主功能表。

z/OS 在 z/OS 上使用 IBM MQ 物件

本文件中說明的許多作業都涉及操作 IBM MQ 物件。 物件類型是佇列管理程式、佇列、程序定義、名稱清單、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件。

- 定義簡式佇列物件
- 定義其他類型的物件
- 使用物件定義
- 使用名稱清單

定義簡式佇列物件

若要定義新物件，請使用現有定義作為其基礎。 您可以透過下列三種方式之一來執行此動作:

- 透過選取物件，該物件是在起始畫面上所選取選項的結果所顯示清單的成員。 然後輸入動作類型 2 (**定義相似**) 在所選取物件旁的動作欄位中。 您的新物件具有所選物件的屬性，但處置除外。 然後，您可以視需要變更新物件中的任何屬性。
- 在起始畫面上，選取 **定義相似** 動作類型，在 **物件類型** 欄位中輸入您要定義的物件類型，然後在 **名稱** 欄位中輸入特定現有物件的名稱。 新物件的屬性與您在 **名稱** 欄位中指定的物件相同，但處置除外。 然後，您可以視需要變更新物件定義中的任何屬性。
- 選取 **定義相似** 動作類型，指定物件類型，然後將 **名稱** 欄位留白。 然後，您可以定義新物件，且它會為您的安裝定義預設屬性。 然後，您可以視需要變更新物件定義中的任何屬性。

註: 您不是在起始畫面上輸入您要定義之物件的名稱，而是在「 **定義** 」畫面上呈現您所定義的物件。

下列範例示範如何使用現有佇列作為範本來定義本端佇列。

定義本端佇列

如果要從作業和控制台定義本端佇列物件，請使用現有的佇列定義作為新定義的基礎。 有數個畫面要完成。 當您完成所有畫面且滿意屬性正確時，請按 Enter 鍵將您的定義傳送至佇列管理程式，然後該佇列管理程式會建立實際佇列。

在起始畫面上，或針對清單中的物件項目使用 **定義相似** 動作，該清單顯示為起始畫面上所選取選項的結果。

例如，從起始畫面開始，完成下列欄位:

動作	2 (定義相似)
物件類型	QLOCAL
名稱	QUEUE.YOU.LIKE. 這是為新佇列提供屬性的佇列名稱。

按 Enter 鍵以顯示 **定義本端佇列** 畫面。 佇列名稱欄位為空白，因此您可以提供新佇列的名稱。 說明是您以此新定義為基礎之佇列的說明。 以您自己的新佇列說明來輸入此欄位。

其他欄位中的值是您以此新佇列為基礎之佇列的值，但處置除外。 您可以視需要來重新輸入這些欄位。 例如，如果適當授權的應用程式可以將訊息放置在此佇列上，請在 **啟用放置** 欄位中鍵入 Y (如果它還不是 Y)。

將游標移至欄位並按功能鍵 F1，即可取得欄位說明。 欄位說明提供可用於每一個屬性的值相關資訊。

當您完成第一個畫面時，請按功能鍵 F8 以顯示第二個畫面。

提示:

1. 請勿在此階段按 Enter 鍵，否則會在您有機會完成其餘欄位之前建立佇列。 (如果您過早按 Enter 鍵，請不要擔心; 您隨時可以稍後變更您的定義。)
2. 請勿按功能鍵 F3 或 F12，否則您鍵入的資料將會遺失。

重複按功能鍵 F8 ，以查看並完成其餘畫面，包括觸發程式定義、事件控制及取消報告畫面。

當本端佇列定義完成時

當定義完成時，請按 Enter 鍵將資訊傳送至佇列管理程式以進行處理。 佇列管理程式會根據您提供的定義來建立佇列。 如果您不想要建立佇列，請按功能鍵 F3 以結束並取消定義。

定義其他類型的物件

若要定義其他類型的物件，請使用現有定義作為新定義的基礎，如 定義本端佇列中所述。

在起始畫面上，或針對清單中的物件項目使用 **定義相似** 動作，該清單顯示為起始畫面上所選取選項的結果。

例如，從起始畫面開始，完成下列欄位:

動作	2 (定義相似)
物件類型	QALIAS、NAMELIST、PROCESS、CHANNEL 及其他資源物件。
名稱	保留空白或輸入相同類型的現有物件名稱。

按 Enter 鍵以顯示對應的 DEFINE 畫面。 視需要完成欄位，然後再次按 Enter 鍵，將資訊傳送至佇列管理程式。

就像定義本端佇列一樣，定義另一種類型的物件通常需要完成數個畫面。 定義名單需要一些其他工作，如第 379 頁的『使用名稱清單』中所述。

使用物件定義

已定義物件時，您可以在 **動作** 欄位中指定動作，以變更、顯示或管理它。

在每一種情況下，您可以:

- 從清單中選取您要使用的物件，該清單會因為在起始畫面上選取的選項而顯示。 例如，在 **動作** 欄位中輸入 1 以顯示物件，在 **物件類型** 欄位中輸入 Queue ，在 **名稱** 欄位中輸入 * ，即會呈現系統中定義的所有佇列清單。 然後，您可以從此清單中選取您需要使用的佇列。
- 從起始畫面開始，您可以透過完成 **物件類型** 及 **名稱** 欄位來指定您正在使用的物件。

變更物件定義
若要變更物件定義，請指定動作 3 ，然後按 Enter 鍵以查看 ALTER 畫面。 這些面板與 DEFINE 面板非常類似。 您可以變更您想要的值。 當變更完成時，請按 Enter 鍵將資訊傳送至佇列管理程式。

顯示物件定義
如果您想要查看物件的詳細資料，但無法變更它們，請指定動作 1 ，然後按 Enter 鍵以查看 DISPLAY 畫面。 同樣地，這些畫面與 DEFINE 畫面類似，但您無法變更任何欄位。 請變更物件名稱，以顯示另一個物件的詳細資料。

刪除物件

若要刪除物件，請指定動作 4 (管理)，並且 **刪除** 動作是產生的功能表上呈現的其中一個動作。 選取 **刪除** 動作。

系統會要求您確認您的要求。 如果您按功能鍵 F3 或 F12，則會取消要求。 如果您按 Enter 鍵，則會確認要求並傳遞至佇列管理程式。 然後會刪除您指定的物件。

註: 除非已啟動通道起始程式，否則無法刪除大部分類型的通道物件。

使用名稱清單

使用名稱清單時，請繼續進行其他物件的作業。

對於動作 DEFINE LIKE 或 ALTER，請按功能鍵 F11，將名稱新增至清單或變更清單中的名稱。 這涉及使用 ISPF 編輯器，且所有一般 ISPF 編輯指令都可供使用。 在個別行上輸入名稱清單中的每一個名稱。

當您以此方式使用 ISPF 編輯器時，功能鍵設定是一般 ISPF 設定， **不是** 其他作業和控制台所使用的設定。

如果您需要在清單中指定小寫名稱，請在編輯器畫面指令行上指定 CAPS (OFF)。 當您執行此動作時，在您指定 CAPS (ON) 之前，您未來編輯的所有名稱清單都是小寫。

當您完成編輯名單時，請按功能鍵 F3，以結束 ISPF 編輯階段作業。 然後按 Enter 鍵，將變更傳送至佇列管理程式。

請注意: 如果您在此階段未按 Enter 鍵，但改為按功能鍵 F3，則會遺失您鍵入的任何更新項目。

z/OS 使用多個叢集傳輸佇列來實作系統

如果在單一叢集或重疊叢集中使用通道，則不會有任何不同。 當選取並啟動通道時，通道會根據定義來選取傳輸佇列。

程序

- 如果您使用 DEFCLXQ 選項，請參閱 第 379 頁的『使用佇列及切換的自動定義』。
- 如果您使用暫置方法，請參閱 第 379 頁的『使用階段式方法變更叢集傳送端通道』。

z/OS 使用佇列及切換的自動定義

如果您計劃使用 DEFCLXQ 選項，請使用此選項。 將為每個通道及每個新通道建立一個佇列。

程序

1. 檢閱 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 並視需要變更屬性。
   此佇列定義在成員 SCSQPROC(csq4insx)中。
2. 建立 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 模型佇列。
3. 套用此模型佇列及 SYSTEM.CLUSTER.TRANSMIT. * * 佇列。
   對於 z/OS，通道起始程式已啟動作業使用者 ID 需要:
   - 控制對 CLASS (MQADMIN) for 的存取權

     ```
     ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channelname
     ```

   - 更新的 CLASS (MQQUEUE) 存取權

     ```
     ssid.SYSTEM.CLUSTER.TRANSMIT.channelname
     ```

z/OS 使用階段式方法變更叢集傳送端通道

如果您計劃使用暫置方法，請使用此選項。 此處理程序可讓您在不同時間移至新的叢集傳送端通道，以符合您企業的需求。

開始之前

- 識別您的商業應用程式，以及使用哪些通道。
- 對於您使用的佇列，顯示它們在其中的叢集。
- 顯示通道以顯示連線名稱、遠端佇列管理程式的名稱，以及通道支援的叢集。

關於這項作業

- 建立傳輸佇列。 在 z/OS 上，您可能想要考量用於佇列的頁集。
- 設定佇列的安全原則。
- 請變更任何佇列監視，以併入此佇列名稱。
- 決定要使用此傳輸佇列的通道。 通道應該具有類似名稱，因此一般字元 '*' 在 CLCHNAME 中識別通道。
- 當您準備好使用新功能時，請變更傳輸佇列，以指定使用此傳輸佇列的通道名稱。 例如，CLUSTER1.TOPARIS 或 CLUSTER1.* 或 *.TOPARIS
- 啟動通道

程序

1. 使用 DIS CLUSQMGR(xxxx) XMITQ 指令來顯示叢集中定義的叢集傳送端通道，其中 *xxxx* 是遠端佇列管理程式的名稱。
2. 設定傳輸佇列的安全設定檔，並提供通道起始程式的佇列存取權。
3. 定義要使用的傳輸佇列，並指定 USAGE (XMITQ) INDXTYPE (CORRELID) SHARE 及 CLCHNAME (*value*)

   通道起始程式啟動作業使用者 ID 需要下列存取權:

   ```
   alter class(MQADMIN) ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channel
   update class(MQQUEUE ssid.SYSTEM.CLUSTER.TRANSMIT.channel
   ```

   且使用 SWITCH 指令的使用者 ID 需要下列存取權:

   ```
   alter cl(MQADMIN) ssid.QUEUE.queuename
   ```

4. 停止並重新啟動通道。

   當通道使用 MQSC 指令啟動時，或您使用 CSQUTIL 時，會發生通道變更。 您可以使用 CSQUTIL 的 SWITCH CHANNEL(*)STATUS 來識別需要重新啟動哪些通道

   如果您在啟動通道時發生問題，請停止通道，解決問題，然後重新啟動通道。

   請注意，您可以根據需要經常變更 CLCHNAME 屬性。

   所使用的 CLCHNAME 值是通道啟動時的值，因此您可以在通道從啟動時開始繼續使用定義時變更 CLCHNAME 定義。 通道會在重新啟動時使用新定義。

z/OS 在 z/OS 上復原變更

如果變更結果不是您預期的結果，則您需要有一個處理程序來取消變更。

什麼會出問題?

如果新的傳輸佇列不是您預期的:

1. 請檢查 CLCHNAME 是否如您預期
2. 請檢閱工作日誌，以檢查交換器處理程序是否已完成。 如果沒有，請稍後等待並檢查通道的新傳輸佇列。

如果您使用多個叢集傳輸佇列，請務必明確設計傳輸佇列定義，並避免複雜的重疊配置。 如此一來，您可以確定如果有問題，您可以回到原始佇列和配置。

如果您在移至使用不同傳輸佇列期間遇到問題，則必須先解決所有問題，才能繼續進行變更。

必須先完成現有的變更要求，才能提出新的變更要求。 例如，您:

1. 定義新的傳輸佇列，其深度上限為 1 ，且有 10 則訊息等待傳送。
2. 請變更傳輸佇列，以在 CLCHNAME 參數中指定通道名稱。
3. 停止並重新啟動通道。 嘗試移動訊息失敗並報告問題。
4. 將傳輸佇列上的 CLCHNAME 參數變更為空白。
5. 停止並重新啟動通道。 通道會繼續嘗試並完成原始要求，因此通道會繼續使用新的傳輸佇列。
6. 需要解決問題並重新啟動通道，以便移動訊息順利完成。

下次重新啟動通道時，它會挑選任何變更，因此如果您將 CLCHNAME 設為空白，通道將不會使用指定的傳輸佇列。

在此範例中，將傳輸佇列上的 CLCHNAME 變更為空白，不一定表示通道使用 SYSTEM.CLUSTER.TRANSMIT 佇列，因為可能有其他傳輸佇列的 CLCHNAME 參數符合通道名稱。 例如，一般名稱或佇列管理程式屬性 DEFCLXQ 可能設為通道，因此通道會使用動態佇列而非 SYSTEM.CLUSTER.TRANSMIT 佇列。

z/OS 撰寫程式以管理 IBM MQ for z/OS

您可以撰寫自己的應用程式來管理佇列管理程式。 請利用這個主題來瞭解撰寫您自己的管理程式的需求。

啟動一般用途程式設計介面資訊

這組主題包含提示和指引，可讓您從 IBM MQ 應用程式發出 IBM MQ 指令。

註: 在本主題中，使用 C 語言表示法來說明 MQI 呼叫。 如需 COBOL、PL/I 及組譯語言中呼叫的一般呼叫，請參閱 函數呼叫 手冊。

瞭解一切如何運作

在大綱中，從應用程式發出指令的程序如下:

1. 將 IBM MQ 指令建置成 IBM MQ 訊息類型，稱為 要求訊息。 指令可以是 MQSC 或 PCF 格式。
2. 傳送 (使用 MQPUT) 將此訊息傳送至稱為系統指令輸入佇列的特殊佇列。 IBM MQ 指令處理器會執行指令。
3. 擷取 (使用 MQGET) 在回覆目的地佇列上作為 回覆訊息 的指令結果。 這些訊息包含使用者訊息，您需要這些訊息來判斷指令是否成功，如果成功，結果為何。

然後由您的應用程式來處理結果。

這組主題包含:

z/OS 準備管理程式的佇列

管理程式需要一些預先定義的佇列，才能進行系統指令輸入及接收回應。

本節適用於 MQSC 格式的指令。 如需 PCF 中的對等項目，請參閱 第 22 頁的『使用 IBM MQ 可程式指令格式』。

您必須先定義要使用的佇列，然後開啟，才能發出任何 MQPUT 或 MQGET 呼叫。

定義系統指令輸入佇列

系統指令輸入佇列是稱為 SYSTEM.COMMAND.INPUT。 提供的 CSQINP2 起始設定資料集 thlqual.SCSQPROC(CSQ4INSG) 包含系統指令輸入佇列的預設定義。 為了與其他平台上的 IBM MQ 相容，這個佇列的別名稱為 SYSTEM.ADMIN.COMMAND.QUEUE 。 如需相關資訊，請參閱 IBM MQ 隨附的範例定義 。

定義回覆目的地佇列

您必須定義回覆目的地佇列，才能接收來自 IBM MQ 指令處理器的回覆訊息。 它可以是任何具有屬性的佇列，容許在其上放置回覆訊息。 不過，對於一般作業，請指定下列屬性:

- USAGE (NORMAL)
- NOTRIGGER (除非您的應用程式使用觸發程式)

避免對指令使用持續訊息，但如果您選擇這樣做，則回覆目的地佇列不能是暫時動態佇列。

所提供的 CSQINP2 起始設定資料集 thlqual.SCSQPROC(CSQ4INSG) 包含稱為 SYSTEM.COMMAND.REPLY.MODEL。 您可以使用此模型來建立動態回覆目的地佇列。

註: 指令處理器所產生的回覆長度最多為 15 000 個位元組。

如果您使用永久動態佇列作為回覆目的地佇列，則您的應用程式應該容許在嘗試刪除佇列之前完成所有 PUT 及 GET 作業的時間，否則會傳回 MQRC2055 (MQRC_Q_NOT_EMPTY)。 如果發生此情況，請在幾秒後重試佇列刪除。

開啟系統指令輸入佇列

您的應用程式必須先連接至佇列管理程式，才能開啟系統指令輸入佇列。 請使用 MQI 呼叫 MQCONN 或 MQCONNX 來執行此動作。

然後使用 MQI 呼叫 MQOPEN 來開啟系統指令輸入佇列。 若要使用此呼叫，請執行下列動作:

1. 將 **Options** 參數設為 MQOO_OUTPUT
2. 設定 MQOD 物件描述子欄位，如下所示:

   ObjectType
   MQOT_Q (物件是佇列)

   ObjectName
   SYSTEM.COMMAND.INPUT

   ObjectQMgrName
   如果您要將要求訊息傳送至本端佇列管理程式，請將此欄位保留空白。 這表示會在本端處理您的指令。

   如果您要在遠端佇列管理程式上處理 IBM MQ 指令，請在這裡放置其名稱。 您也必須設定正確的佇列及鏈結，如 分散式佇列及叢集中所述。

開啟回覆目的地佇列

若要從 IBM MQ 指令擷取回覆，您必須開啟回覆目的地佇列。 作法之一是在 MQOPEN 呼叫中指定模型佇列 SYSTEM.COMMAND.REPLY.MODEL ，以建立永久動態佇列作為回覆目的地佇列。 若要使用此呼叫，請執行下列動作:

1. 將 **Options** 參數設為 MQOO_INPUT_SHARED
2. 設定 MQOD 物件描述子欄位，如下所示:

   ObjectType
   MQOT_Q (物件是佇列)

   ObjectName
   回覆目的地佇列的名稱。 如果您指定的佇列名稱是模型佇列物件的名稱，則佇列管理程式會建立動態佇列。

   ObjectQMgrName
   若要在本端佇列管理程式上接收回覆，請將此欄位保留空白。

   DynamicQName
   指定要建立的動態佇列名稱。

z/OS 使用指令伺服器

指令伺服器是與指令處理器元件搭配使用的 IBM MQ 元件。 您可以將格式化訊息傳送至指令伺服器，以解譯訊息、執行管理要求，以及將回應傳回管理應用程式。

指令伺服器會從系統指令輸入佇列讀取要求訊息，驗證它們，並將有效的訊息當作指令傳遞至指令處理器。 指令處理器會處理指令，並將任何回覆作為回覆訊息放置在您指定的回覆目的地佇列上。 第一個回覆訊息包含使用者訊息 CSQN205I。 如需相關資訊，請參閱 第 386 頁的『解譯來自指令伺服器的回覆訊息』。 不論從何處發出通道起始程式及佇列共用群組指令，指令伺服器也會處理這些指令。

識別處理指令的佇列管理程式

處理您從管理程式發出的指令的佇列管理程式是擁有系統指令輸入佇列的佇列管理程式。

啟動指令伺服器

通常，當佇列管理程式啟動時，指令伺服器會自動啟動。 只要從 START QMGR 指令傳回訊息 CSQ9022I 'START QMGR' NORMAL COMPLETION 即可使用。 在系統終止階段期間，當所有已連接的作業都已斷線時，指令伺服器即會停止。

您可以使用 START CMDSERV 及 STOP CMDSERV 指令自行控制指令伺服器。 若要防止指令伺服器在 IBM MQ 重新啟動時自動啟動，您可以將 STOP CMDSERV 指令新增至 CSQINP1 或 CSQINP2 起始設定資料集。 不過，不建議這樣做，因為它會防止處理任何通道起始程式或佇列共用群組指令。

STOP CMDSERV 指令會在完成處理現行訊息之後立即停止指令伺服器，如果未處理任何訊息，則會立即停止指令伺服器。

如果指令伺服器已由程式中的 STOP CMDSERV 指令停止，則無法處理程式中的其他指令。 若要重新啟動指令伺服器，您必須從 z/OS 主控台發出 START CMDSERV 指令。

如果您在佇列管理程式執行時停止並重新啟動指令伺服器，則在指令伺服器停止時，系統指令輸入佇列上的所有訊息都會在指令伺服器重新啟動時處理。 不過，如果您在指令伺服器停止之後停止並重新啟動佇列管理程式，則在指令伺服器重新啟動時，只會處理系統指令輸入佇列上的持續訊息。 系統指令輸入佇列上的所有非持續訊息都會遺失。

將指令傳送至指令伺服器

對於每一個指令，您可以建置包含指令的訊息，然後將它放在系統指令輸入佇列中。

建置包含 IBM MQ 指令的訊息

您可以建置包含必要指令的要求訊息，以在應用程式中納入 IBM MQ 指令。 對於每一個這類指令，您可以執行下列動作:

1. 建立緩衝區，其中包含代表指令的字串。
2. 在呼叫的 **buffer** 參數中指定緩衝區名稱，以發出 MQPUT 呼叫。

在 C 中，最簡單的作法是使用 'char' 來定義緩衝區。 例如:

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

當您建置指令時，請使用以空值結尾的字串。 請勿在以這種方式定義的指令開頭指定指令字首字串 (CPF)。 這表示如果您想要在另一個佇列管理程式上執行指令 Script，則不需要變更它們。 不過，您必須將 CPF 併入任何放入回覆目的地佇列的回應訊息中。

指令伺服器會將所有小寫字元全部轉換成大寫，除非它們是在引號內。

指令可以是任何長度，最多 32 762 個字元。

將訊息放置在系統指令輸入佇列上

使用 MQPUT 呼叫來將包含指令的要求訊息放置在系統指令輸入佇列上。 在此呼叫中，您指定已開啟的回覆目的地佇列名稱。

若要使用 MQPUT 呼叫，請執行下列動作:

1. 設定下列 MQPUT 參數:

Hconn
MQCONN 或 MQCONNX 呼叫傳回的連線控點。

Hobj
系統指令輸入佇列的 MQOPEN 呼叫所傳回的物件控點。

BufferLength
格式化指令的長度。

Buffer
包含指令的緩衝區名稱。

2. 設定下列 MQMD 欄位:

MsgType
MQMT_REQUEST

Format
MQFMT_STRING 或 MQFMT_NONE

如果您未使用與佇列管理程式相同的字碼頁，請適當地設定 *CodedCharSetId* 並設定 MQFMT_STRING，以便指令伺服器可以轉換訊息。 請勿設定 MQFMT_ADMIN，因為這會導致您的指令解譯為 PCF。

ReplyToQ
回覆目的地佇列的名稱。

ReplyToQMgr
如果您要將回覆傳送至本端佇列管理程式，請將此欄位留白。 如果您要將 IBM MQ 指令傳送至遠端佇列管理程式，請將其名稱放在這裡。 您也必須設定正確的佇列及鏈結，如 分散式佇列及叢集 中所述。

3. 視需要設定任何其他 MQMD 欄位。 您通常應該對指令使用非持續訊息。
4. 視需要設定任何 *PutMsgOpts* 選項。

如果您指定 MQPMO_SYNCPOINT (預設值)，則必須使用同步點呼叫來遵循 MQPUT 呼叫。

使用 MQPUT1 及系統指令輸入佇列
如果您只想在系統指令輸入佇列中放置一則訊息，您可以使用 **MQPUT1** 呼叫。 此呼叫會結合 **MQOPEN** 的功能，後面接著一個訊息的 **MQPUT**，後面接著一個 **MQCLOSE**，全部在一個呼叫中。 如果您使用此呼叫，請相應地修改參數。 如需詳細資料，請參閱 使用 MQPUT1 呼叫將一則訊息放到佇列中 。

z/OS 擷取指令的回覆

指令伺服器會針對它所接收的每一則要求訊息，將回應傳送至回覆佇列。 任何管理應用程式都必須接收並處理回覆訊息。

當指令處理器處理您的指令時，任何回覆訊息都會放入 MQPUT 呼叫中指定的回覆佇列。 指令伺服器會傳送回覆訊息，其持續性與收到的指令訊息相同。

等待回覆

使用 MQGET 呼叫來擷取要求訊息中的回覆。 一個要求訊息可以產生數個回覆訊息。 如需詳細資料，請參閱 第 386 頁的『解譯來自指令伺服器的回覆訊息』。

您可以指定 MQGET 呼叫等待產生回覆訊息的時間間隔。 如果您未取得回覆，請使用主題 第 386 頁的『如果您沒有收到回覆』中開始的核對清單。

若要使用 MQGET 呼叫，請執行下列動作:

1. 設定下列參數:

Hconn
MQCONN 或 MQCONNX 呼叫傳回的連線控點。

Hobj
由 MQOPEN 呼叫回覆目的地佇列傳回的物件控點。

Buffer
接收回覆的區域名稱。

BufferLength
接收回覆的緩衝區長度。 至少必須有 80 個位元組。

2. 若要確保僅從您發出的指令取得回應，您必須指定適當的 *MsgId* 及 *CorrelId* 欄位。 這些取決於您在 MQPUT 呼叫中指定的報告選項 MQMD_REPORT:

MQRO_NONE
二進位零， '00 ... 00' (24 個空值)。

MQRO_NEW_MSG_ID
二進位零， '00 ... 00' (24 個空值)。

如果未指定任何這些選項，則這是預設值。

MQRO_PASS_MSG_ID
MQPUT 中的 *MsgId* 。

MQRO_NONE
來自 MQPUT 呼叫的 *MsgId* 。

MQRO_COPY_MSG_ID_TO_CORREL_ID
來自 MQPUT 呼叫的 *MsgId* 。

如果未指定任何這些選項，則這是預設值。

MQRO_PASS_CORREL_ID
來自 MQPUT 呼叫的 *CorrelId* 。

如需報告選項的詳細資料，請參閱 報告選項及訊息旗標。

3. 設定下列 *GetMsgOpts* 欄位:

Options
MQGMO_WAIT

如果您未使用與佇列管理程式相同的字碼頁，請在 MQMD 中適當設定 MQGMO_CONVERT ，並設定 *CodedCharSetId* 。

WaitInterval
對於來自本端佇列管理程式的回覆，請嘗試 5 秒。 以毫秒來編碼，這會變成 5000。 對於來自遠端佇列管理程式的回覆，以及通道控制和狀態指令，請嘗試 30 秒。 以毫秒來編碼，這會變成 30 000。

捨棄的訊息

如果指令伺服器發現要求訊息無效，則會捨棄此訊息，並將訊息 CSQN205I 寫入指名的回覆目的地佇列。 如果沒有回覆目的地佇列，則會將 CSQN205I 訊息放入無法傳送郵件的佇列中。 此訊息中的回覆碼顯示原始要求訊息無效的原因:

00D5020F 它不是 MQMT_REQUEST 類型。

00D50210 它的長度為零。

00D50212 它超過 32 762 個位元組。

00D50211 它包含所有空白。

00D5483E 需要轉換，但 *Format* 不是 MQFMT_STRING。

其他 請參閱 指令伺服器代碼

指令伺服器回覆訊息描述子

對於任何回覆訊息，會設定下列 MQMD 訊息描述子欄位:

MsgType	MQMT_REPLY
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
Priority	至於您發出的訊息中的 MQMD。
Persistence	至於您發出的訊息中的 MQMD。
CorrelId	取決於 MQPUT 報告選項。
ReplyToQ	無。

指令伺服器會將 MQPMO 結構的 *Options* 欄位設為 MQPMO_NO_SYNCPOINT。 這表示您可以在建立回覆時擷取回覆，而不是在下一個同步點以群組方式擷取回覆。

z/OS 解譯來自指令伺服器的回覆訊息

IBM MQ 正確處理的每一個要求訊息至少會產生兩個回覆訊息。 每一個回覆訊息都包含單一 IBM MQ 使用者訊息。

回覆的長度視發出的指令而定。 您可以從 DISPLAY NAMELIST 取得的最長回覆，長度最多可達 15000 個位元組。

第一個使用者訊息 CSQN205I 一律包含:

- 回覆計數 (以十進位表示)，您可以用作迴圈中的計數器，以取得其餘回覆。 計數包括這則第一則訊息。
- 來自指令前置處理器的回覆碼。
- 原因碼，這是來自指令處理器的原因碼。

此訊息不包含 CPF。

例如:

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
```

COUNT 欄位長度為 8 個位元組，且向右對齊。 它一律從位置 18 開始，即緊接在 COUNT=之後。 RETURN 欄位長度為 8 個位元組，十六進位字元，且緊接在位置 35 的 RETURN= 之後。 「原因」欄位以字元十六進位表示，長度為 8 個位元組，緊接在位置 52 的 REASON= 之後。

如果 RETURN= 值為 00000000 且 REASON= 值為 00000004，則回覆訊息集不完整。 擷取 CSQN205I 訊息指出的回覆之後，請發出進一步 MQGET 呼叫以等待進一步的回覆集。 下一組回覆中的第一個訊息再次是 CSQN205I，指出有多少回覆，以及是否還有更多回覆。

如需個別訊息的詳細資料，請參閱 IBM MQ for z/OS 訊息、完成和原因代碼 文件。

如果您使用非英文語言特性，則回覆的文字和版面與這裡顯示的不同。 不過，訊息 CSQN205I 中計數與回覆碼的大小與位置相同。

z/OS 如果您沒有收到回覆

如果您未收到對指令伺服器的要求的回應，則可以採取一系列步驟。

如果您未收到要求訊息的回覆，請執行下列核對清單:

- 指令伺服器是否在執行中?
- *WaitInterval* 夠長嗎?
- 是否正確定義系統指令輸入及回覆目的地佇列?
- 對這些佇列的 MQOPEN 呼叫是否成功?
- 是否同時針對 MQPUT 及 MQGET 呼叫啟用系統指令輸入及回覆目的地佇列?
- 您是否考慮增加佇列的 MAXDEPTH 及 MAXMSGL 屬性?

- 您是否正確使用 *CorrelId* 和 *MsgId* 欄位?
- 佇列管理程式是否仍在執行中?
- 指令建置正確嗎?
- 您的所有遠端鏈結是否都已定義且正常運作?
- 是否正確定義 MQPUT 呼叫?
- 是否已將回覆目的地佇列定義為暫時動態佇列而非永久動態佇列? (如果要求訊息持續存在，您必須使用永久動態佇列來進行回覆。)

當指令伺服器產生回覆但無法將它們寫入您指定的回覆目的地佇列時，它會將它們寫入無法傳送郵件的佇列。

z/OS 使用 MGCRE 傳遞指令

在適當的授權下，應用程式可以使用 z/OS 服務常式向多個佇列管理程式提出要求。

如果您具有正確的授權，則可以透過 MGCRE (SVC 34) z/OS 服務將 IBM MQ 指令從程式傳遞至多個佇列管理程式。 CPF 的值可識別指令所導向的特定佇列管理程式。 如需 CPF 的相關資訊，請參閱 指令安全及指令資源安全的使用者 ID 及 第 363 頁的『在 z/OS 上發出佇列管理程式指令』。

如果您使用 MGCRE，則可以使用「指令及回應記號 (CART)」來取得指令的直接回應。

z/OS 指令及其回覆的範例

請使用這個主題作為指令伺服器的一系列指令範例，以及指令伺服器的回應。

以下是一些可以內建在 IBM MQ 訊息中的指令範例，以及作為回覆的使用者訊息。 除非另有說明，否則回覆的每一行都是個別訊息。

- 來自 DEFINE 指令的訊息
- DELETE 指令的訊息
- DISPLAY 指令的訊息
- 來自具有 CMDSCOPE 之指令的訊息
- 來自使用 CMDSCOPE 產生指令之指令的訊息

來自 DEFINE 指令的訊息

the following command:

```
DEFINE QLOCAL(Q1)
```

會產生下列訊息:

```
CSQN205I    COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

這些回覆訊息會在正常完成時產生。

DELETE 指令的訊息

the following command:

```
DELETE QLOCAL(Q2)
```

會產生下列訊息:

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
CSQM125I +CSQ1 CSQMUQLC QLOCAL (Q2) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E +CSQ1 CSQMUQLC FAILURE REASON CODE X'00D44002'
CSQ9023E +CSQ1 CSQMUQLC ' DELETE QLOCAL' ABNORMAL COMPLETION
```

這些訊息指出稱為 Q2 的本端佇列不存在。

DISPLAY 指令的訊息

下列範例顯示部分 DISPLAY 指令的回覆。

找出無法傳送郵件的佇列名稱

如果您想要找出佇列管理程式的無法傳送郵件的佇列名稱，請從應用程式發出下列指令:

```
DISPLAY QMGR DEADQ
```

會傳回下列三則使用者訊息，您可以從中擷取必要的名稱:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE          )
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

來自 DISPLAY QUEUE 指令的訊息

下列範例顯示指令的結果如何取決於該指令中指定的屬性。

範例 1

您可以使用下列指令來定義本端佇列:

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

如果您從應用程式發出下列指令:

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

會傳回下列三則使用者訊息:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(Q1                       ) TYPE(
QLOCAL ) QSGDISP(QMGR   )
DESCR(A sample queue
) SHARE  GET(ENABLED  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

註: 這裡顯示第二個訊息 CSQM401I，佔用四行。

範例 2

兩個佇列的名稱以字母 A 開頭:

- A1 是本端佇列，其 PUT 屬性設為 DISABLED。
- A2 是遠端佇列，其 PUT 屬性設為 ENABLED。

如果您從應用程式發出下列指令:

```
DISPLAY QUEUE(A*) PUT
```

會傳回下列四則使用者訊息:

```
CSQN205I    COUNT=    4, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(A1                       ) TYPE(
QLOCAL ) QSGDISP(QMGR   )
PUT(DISABLED )
CSQM406I +CSQ1 QUEUE(A2                       ) TYPE(
QREMOTE ) PUT(ENABLED  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

註: 這裡顯示第二個及第三個訊息 (CSQM401I 及 CSQM406I) , 佔用三行及兩行。

來自 DISPLAY NAMELIST 指令的訊息

您可以使用下列指令來定義名稱清單:

```
DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)
```

如果您從應用程式發出下列指令:

```
DISPLAY NAMELIST(N1) NAMES NAMCOUNT
```

會傳回下列三個使用者訊息:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM407I +CSQ1 NAMELIST(N1                       ) QS
GDISP(QMGR   ) NAMCOUNT(     2) NAMES(Q1
,SAMPLE_QUEUE                 )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY NAMELIST' NORMAL COMPLETION
```

註: 這裡顯示第二則訊息 CSQM407I, 佔用三行。

來自具有 CMDSCOPE 之指令的訊息

下列範例顯示已使用 CMDSCOPE 屬性輸入之指令的回覆。

來自 ALTER PROCESS 指令的訊息

the following command:

```
ALT PRO(V4) CMDSCOPE(*)
```

會產生下列訊息:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'ALT PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ26
CSQM125I !MQ26 CSQMMSGP PROCESS(V4) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E !MQ26 CSQMMSGP FAILURE REASON CODE X'00D44002'
CSQ9023E !MQ26 CSQMMSGP ' ALT PRO' ABNORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' ALT PRO' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=0000000C, REASON=00000008
CSQN123E !MQ25 'ALT PRO' command for CMDSCOPE(*) abnormal completion
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 指令在 MQ25 上順利完成，但程序定義在 MQ26 上不存在，因此指令在該佇列管理程式上失敗。

DISPLAY PROCESS 指令的訊息

the following command:

```
DIS PRO(V*) CMDSCOPE(*)
```

會產生下列訊息:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ26
CSQM408I !MQ26 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ26 PROCESS(V3) QSGDISP(QMGR)
CSQ9022I !MQ26 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    7, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ25
CSQM408I !MQ25 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ25 PROCESS(V2) QSGDISP(GROUP)
CSQM408I !MQ25 PROCESS(V3) QSGDISP(QMGR)
CSQM408I !MQ25 PROCESS(V4) QSGDISP(QMGR)
CSQ9022I !MQ25 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS PRO' command for CMDSCOPE(*) normal completion
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 會顯示每一個佇列管理程式上名稱以字母 V 開頭的所有處理程序的相關資訊。

DISPLAY CHSTATUS 指令的訊息

the following command:

```
DIS CHS(VT) CMDSCOPE(*)
```

會產生下列訊息:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS CHS' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ25
CSQM422I !MQ25 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ25 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=    4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ26
CSQM422I !MQ26 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ26 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS CHS' command for CMDSCOPE(*) normal completion
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 會顯示每一個佇列管理程式上通道狀態的相關資訊。

STOP CHANNEL 指令的訊息

the following command:

```
STOP CHL(VT) CMDSCOPE(*)
```

會產生下列訊息:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'STOP CHL' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQM134I !MQ25 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
SQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQM134I !MQ26 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQ9022I !MQ26 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQ9022I !MQ25 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'STOP CHL' command for CMDSCOPE(*) normal completion
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 已在每一個佇列管理程式上停止通道 VT。

來自使用 CMDSCOPE 產生指令之指令的訊息

the following command:

```
DEF PRO(V2) QSGDISP(GROUP)
```

會產生下列訊息:

```
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQM122I !MQ25 CSQMMSGP ' DEF PRO' COMPLETED FOR QSGDISP(GROUP)
CSQN138I !MQ25 'DEFINE PRO' command generated for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ26
CSQ9022I !MQ26 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DEFINE PRO' command for CMDSCOPE(*) normal completion
```

這些訊息告訴您已在佇列管理程式 MQ25 上輸入指令。 在共用儲存庫上建立物件時，會產生另一個指令並傳送至佇列共用群組 (MQ25 及 MQ26) 中的所有作用中佇列管理程式。

z/OS 在 z/OS 上管理 IBM MQ 資源

使用本主題中的鏈結，以瞭解如何管理 IBM MQ for z/OS 所使用的資源，例如，管理日誌檔、資料集、頁集、緩衝池及連結機能結構。

請使用下列鏈結，以取得您在使用 IBM MQ for z/OS 時可能必須完成的不同管理作業的詳細資料:

- 第 392 頁的『管理日誌』
- 第 399 頁的『管理引導資料集 (BSDS)』
- 第 406 頁的『管理頁面集』
- 第 411 頁的『如何備份及回復頁集』
- 第 414 頁的『如何使用 CSQUTIL 備份及還原佇列』
- 第 414 頁的『管理緩衝池』
- 第 415 頁的『在 z/OS 上管理佇列共用群組及共用佇列』

相關概念

IBM MQ for z/OS 概念

第 353 頁的『管理 IBM MQ for z/OS』
管理佇列管理程式及相關聯資源包括您在啟動及管理這些資源時經常執行的作業。 選擇您偏好用來管理佇列管理程式及相關聯資源的方法。

第 354 頁的『對 IBM MQ for z/OS 發出指令』
您可以在批次或互動模式下使用 IBM MQ Script 指令 (MQSC) 來控制佇列管理程式。

第 423 頁的『在 z/OS 上回復並重新啟動』
請利用這個主題來瞭解 IBM MQ 所使用的回復和重新啟動機制。

相關工作

在 z/OS 上規劃 IBM MQ 環境

在 z/OS 上配置佇列管理程式

使用 IBM MQ for z/OS 公用程式

相關參考

第 18 頁的『您可以在 z/OS 上從中發出 MQSC 指令的來源』
MQSC 指令可以從各種來源發出，視指令而定。

第 361 頁的『IBM MQ for z/OS 公用程式』
IBM MQ for z/OS 提供一組公用程式，您可以用來協助進行系統管理。

可程式化指令格式參照

z/OS 管理日誌

請利用這個主題來瞭解如何管理 IBM MQ 日誌檔，包括日誌保存程序、使用日誌記錄壓縮、日誌記錄回復及列印日誌記錄。

本主題說明管理 IBM MQ 日誌所涉及的作業。 它包含下列區段:

z/OS 使用 *ARCHIVE LOG* 指令保存日誌

每當需要使用 **ARCHIVE LOG** 指令時，授權操作員可以保存現行 IBM MQ 作用中日誌資料集。

當您發出 ARCHIVE LOG 指令時， IBM MQ 會截斷現行作用中日誌資料集，然後執行非同步卸載處理程序，並以卸載處理程序的記錄來更新 BSDS。

ARCHIVE LOG 指令具有 **MODE(QUIESCE)** 選項。 使用此選項， IBM MQ 工作及使用者會在確定點之後靜止，且在卸載之前，會在現行作用中日誌中擷取產生的一致性點。

在規劃離站回復的備份策略時，請考慮使用 **MODE(QUIESCE)** 選項。 它會建立全系統一致性點，當在回復期間將保存日誌與最新備份頁集副本搭配使用時，會將資料不一致的數目減至最少。 例如:

```
ARCHIVE LOG MODE(QUIESCE)
```

如果您發出 **ARCHIVE LOG** 指令但未指定 **TIME** 參數，則靜止時段預設為 CSQ6ARVP 巨集的 **QUIESCE** 參數值。 如果完成 ARCHIVE LOG MODE(QUIESCE) 所需的時間小於指定的時間，則指令會順利完成; 否則，指令會在時段到期時失敗。 您可以使用 **TIME** 選項來明確指定時段，例如:

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

此指令指定在進行 **ARCHIVE LOG** 處理之前最多 60 秒的靜止期間。

請注意: 當時間很重要時使用 **TIME** 選項，可能會大幅中斷所有使用 IBM MQ 資源之工作及使用者的 IBM MQ 可用性。

依預設，會從您提交指令時開始非同步處理指令。 (若要與其他 IBM MQ 指令同步處理指令，請搭配使用 WAIT (YES) 選項與 **QUIESCE**，但請注意，在整個 **QUIESCE** 期間內， z/OS 主控台已從 IBM MQ 指令輸入鎖定。)

在靜止期間:

- 佇列管理程式上的工作及使用者可以進行確定處理，但如果在確定之後嘗試更新任何 IBM MQ 資源，則會暫停。
- 只讀取資料的工作及使用者可能會受到影響，因為他們可能正在等待已暫停的工作或使用者所保留的鎖定。
- 新作業可以啟動，但無法更新資料。

DISPLAY LOG 指令的輸出使用 CSQV400I 訊息指出靜止生效。

V 9.2.0 例如，從 IBM MQ 9.1.4:

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     400
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
MAXCNOFF    0
WRTHRSH     20
DEALLCT     0
COMPLOG     NONE
ZHYWRITE    NO
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full zHyperWrite Encrypted DSName
 1     68  NO          NO        VICY.CSQ1.LOGCOPY1.DS01
 2     68  NO          NO        VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2019-08-15 09:49:30 using RBA=000000000891B000
Latest RBA=000000000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
```

```
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
```

例如，對於 Long Term Support 及 IBM MQ 9.1.2 之前的 Continuous Delivery :

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     400
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
MAXCNOFF    0
WRTHRSH     20
DEALLCT     0
COMPLOG     NONE
ZHYWRITE    NO                     YES
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full PPRC DSName
 1     68  NO   VICY.CSQ1.LOGCOPY1.DS01
 2     68  NO   VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2014-04-15 09:49:30 using RBA=000000000891B000
Latest RBA=000000000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
```

當所有更新項目都靜止時， BSDS 中的靜止歷程記錄會更新為截斷作用中日誌資料集的日期和時間，以及現行作用中日誌資料集中前次寫入 RBA 的日期和時間。 IBM MQ 會截斷現行作用中日誌資料集，切換至下一個可用的作用中日誌資料集，並發出訊息 CSQJ311I，指出卸載處理程序已啟動。

如果在靜止期間到期之前無法靜止更新項目， IBM MQ 會發出訊息 CSQJ317I，且 **ARCHIVE LOG** 處理程序會終止。 現行作用中日誌資料集不會截斷，也不會切換至下一個可用的日誌資料集，且卸載處理程序不會啟動。

不論靜止是否成功，都會回復所有已暫停的使用者及工作，且 IBM MQ 會發出訊息 CSQJ312I，指出靜止已結束並回復更新活動。

如果在現行作用中日誌是最後一個可用的作用中日誌資料集時發出 **ARCHIVE LOG**，則不會處理指令，且 IBM MQ 會發出下列訊息:

```
CSQJ319I -csect-name CURRENT ACTIVE LOG DATA SET 是 LAST
可用的作用中日誌資料集。 保存日誌處理
將終止
```

如果在另一個 **ARCHIVE LOG** 指令已在進行中時發出 **ARCHIVE LOG**，則不會處理新指令，且 IBM MQ 會發出下列訊息:

```
CSQJ318I -保存日誌指令已在進行中
```

如需在保存期間發出的訊息的相關資訊，請參閱 IBM MQ for z/OS 的訊息。

在失敗之後重新啟動日誌保存處理程序

如果在日誌保存處理程序期間發生問題 (例如，配置或磁帶裝載的問題)，則可能會暫停保存作用中日誌。 您可以使用下列指令取消保存程序並重新啟動它:

```
ARCHIVE LOG CANCEL OFFLOAD
```

這個指令會取消目前進行中的任何卸載處理程序，並重新啟動保存處理程序。 它會從尚未保存的最舊日誌資料集開始，並繼續進行所有需要卸載的作用中日誌資料集。 已暫停的任何日誌保存作業都會重新啟動。

只有在您確定現行日誌保存作業不再運作，或您想要重新啟動先前失敗的嘗試時，才使用這個指令。 這是因為指令可能會導致卸載作業異常終止，這可能會導致傾出。

z/OS 控制保存及記載

您可以使用 CSQ6LOGP、 CSQ6ARVP 及 CSQ6SYSP 巨集來控制壓縮、列印、保存、回復及記載。 請注意，專用物件的變更只會記載在 IBM MQ 日誌中。 也會記載 GROUP 物件 (例如共用入埠通道) 的變更，因為定義會在群組周圍傳播並在本端保留。

當自訂佇列管理程式時，保存及記載的許多層面是由使用系統參數模組的 CSQ6LOGP、 CSQ6ARVP 及 CSQ6SYSP 巨集所設定的參數所控制。 如需這些巨集的詳細資料，請參閱 自訂系統參數模組 。

當佇列管理程式使用 IBM MQ MQSC SET LOG、SET SYSTEM 及 SET ARCHIVE 指令執行時，可以變更其中部分參數。 它們顯示在 第 395 頁的表 28 中:

表 *28:* 在佇列管理程式執行時可以變更的保存及記載參數

設定指令	**參數**
日誌	WRTHRSH、MAXARCH、DEALLCT、MAXRTU、COMPLOG
保存式	全部
系統	LOGLOAD

您可以使用 MQSC DISPLAY LOG、 DISPLAY ARCHIVE 及 DISPLAY SYSTEM 指令來顯示所有參數的設定。 這些指令也會顯示保存及記載的相關狀態資訊。

控制日誌壓縮

您可以使用下列其中一項來啟用及停用日誌記錄的壓縮:

- MQSC 中的 SET 和 DISPLAY LOG 指令; 請參閱 MQSC 指令
- 正在呼叫 PCF 介面。 請參閱 第 21 頁的『IBM MQ 可程式指令格式簡介』
- 在系統參數模組中使用 CSQ6LOGP 巨集; 請參閱 使用 CSQ6LOGP

列印日誌記錄

您可以使用 CSQ1LOGP 公用程式來擷取及列印日誌記錄。 如需指示，請參閱 日誌列印公用程式。

回復日誌

通常，您不需要備份及還原 IBM MQ 日誌，尤其是當您使用雙重記載時。 不過，在極少數情況下 (例如日誌上的 I/O 錯誤) ，您可能需要回復日誌。 使用「存取方法服務」來刪除並重新定義資料集，然後將對應的雙重日誌複製到其中。

z/OS 捨棄保存日誌資料集

您可以捨棄保存日誌資料集，並選擇自動或手動捨棄日誌。

您必須保留足夠的日誌資料，才能執行工作單元回復、頁集媒體回復 (如果頁集遺失) 或 CF 結構媒體回復 (如果 CF 結構遺失)。 請勿捨棄回復可能需要的保存日誌資料集; 如果您捨棄這些保存日誌資料集，您可能無法執行必要的回復作業。

如果您已確認可以捨棄保存日誌資料集，則可以使用下列其中一種方式來執行此動作:

- 自動刪除保存日誌資料集
- 手動刪除保存日誌資料集

自動刪除保存日誌資料集

您可以使用 DASD 或磁帶管理系統來自動刪除保存日誌資料集。 IBM MQ 保存日誌資料集的保留期間由 CSQ6ARVP 安裝巨集中的保留期間欄位 ARCRETN 指定 (如需相關資訊，請參閱 使用 CSQ6ARVP)。

保留期的預設值指定將保存日誌保留 9999 天 (上限)。

重要: 您可以變更保留期間，但必須確保您可以容納已規劃的備份週期數。

.

建立保存日誌資料集時， IBM MQ 會使用保留期間值作為 JCL 參數 RETPD 的值。

此 IBM MQ 參數可以置換 MVS/DFP 儲存體管理子系統 (SMS) 所設定的保留期間。 通常，保留期間會設為 IBM MQ 或 SMS 所指定的較小值。 儲存體管理者和 IBM MQ 管理者必須同意適用於 IBM MQ 的保留期間值。

註: IBM MQ 沒有自動化方法可從 BSDS 中刪除保存日誌資料集的相關資訊，因為部分磁帶管理系統提供保留期間的外部手動置換。 因此，保存日誌資料集的相關資訊在資料集保留期已過期且磁帶管理系統已刮回資料集之後很長一段時間仍可以在 BSDS 中。 相反地，可能已超出保存日誌資料集數目上限，且在資料集達到其到期日之前，可能已捨棄 BSDS 中的資料。

如果自動刪除保存日誌資料集，請記住作業不會更新 BSDS 中的保存日誌清單。 您可以使用變更日誌庫存公用程式來更新 BSDS，如 第 401 頁的『變更 BSDS』中所述。 更新並不重要。 記錄舊的保存日誌會浪費 BSDS 中的空間，但不會造成其他傷害。

手動刪除保存日誌資料集

您必須將所有日誌記錄保留到訊息 CSQI024I 及 CSQI025I 中所識別的最低 RBA 為止。 此 RBA 是使用您在使用 方法 1: 完整備份建立回復點時發出的 DISPLAY USAGE 指令來取得。

請先閱讀 建立非共用資源的回復點，再捨棄任何日誌。

尋找並捨棄保存日誌資料集

建立回復所需的日誌 RBA 下限之後，您可以執行下列程序來尋找只包含較早日誌記錄的保存日誌資料集:

1. 使用列印日誌對映公用程式來列印 BSDS 的內容。 如需輸出的範例，請參閱 列印日誌對映公用程式。
2. 尋找標題為 ARCHIVE LOG COPY n DATA SETS 之輸出的區段。 如果您使用雙重記載，則有兩個區段。 標籤為 STARTRBA 和 ENDRBA 的直欄會顯示每一個磁區中包含的 RBA 範圍。 尋找其範圍包括您找到之訊息 CSQI024I 及 CSQI025I 的 RBA 下限的磁區。 這些是您需要保留的最早磁區。 如果您使用雙重記載，則有兩個這類磁區。

   如果沒有磁區具有適當的範圍，則適用下列其中一種情況:

   - 尚未保存最小 RBA，您可以捨棄所有保存日誌磁區。
   - 當磁區數超出 CSQ6LOGP 巨集的 MAXARCH 參數所容許的數目時， BSDS 中折返的保存日誌磁區清單。 如果 BSDS 未登錄保存日誌磁區，則該磁區無法用於回復。 因此，請考量將現有磁區的相關資訊新增至 BSDS。 如需指示，請參閱第 403 頁的『保存日誌的變更』。

     也請考量增加 MAXARCH 的值。 如需相關資訊，請參閱 使用 CSQ6LOGP。
3. 刪除 ENDRBA 值小於您要保留之最早磁區的 STARTRBA 值的任何保存日誌資料集或磁區。 如果您使用雙重記載，請刪除這兩個副本。

   因為 BSDS 項目會折返，所以 BSDS 保存日誌區段中的前幾個項目可能比結尾的項目更新。 查看日期和時間的組合，並比較其年齡。 請勿假設您可以捨棄包含最低 LOGRBA 之保存日誌項目之前的所有項目。

   刪除資料集。 如果保存檔是在磁帶上，請消除磁帶。 如果它們位於 DASD 上，請執行 z/OS 公用程式來刪除每一個資料集。 然後，如果您想要 BSDS 只列出現有的保存磁區，請使用變更日誌庫存公用程式 (CSQJU003) 來刪除捨棄磁區的項目。 如需範例，請參閱 第 403 頁的『保存日誌的變更』。

z/OS 日誌分流的影響

長時間執行交易可能會導致工作日誌記錄的單元跨越日誌資料集。 IBM MQ 會使用日誌分流來處理此實務範例，這項技術會移動日誌記錄，以最佳化所保留日誌資料的數量，以及佇列管理程式重新啟動時間。

當工作單元被視為較長時，每一個日誌記錄的表示法會進一步寫下日誌。 這稱為 日誌分流。 它在 日誌檔中有更完整的說明。

在失敗之後，佇列管理程式會使用這些延遲的日誌記錄，而非原始記錄，以確保工作單元完整性。 這有兩個好處:

- 減少工作單元協調必須保留的日誌資料量
- 必須在佇列管理程式重新啟動時遍訪較少日誌資料，因此佇列管理程式會更快速重新啟動

延遲的日誌記錄未包含足夠的媒體回復作業資訊。

日誌中保留的資料用於兩個不同的用途: 媒體回復及工作單元協調。 如果發生會影響 CF 結構或頁集的媒體故障，佇列管理程式可以透過還原先前的副本並使用日誌中包含的資料更新此副本，將媒體回復至故障點。 在工作單元中執行的持續性活動會記錄在日誌上，以便在發生失敗時，可以取消或在變更的資源上回復鎖定。 這兩個元素會影響您為了啟用佇列管理程式回復而需要保留的日誌資料數量。

對於媒體回復，您必須保留足夠的日誌資料，才能至少從最新的媒體副本執行媒體回復，並且能夠回復。 (您的網站可能會規定從較舊的備份回復的能力。) 針對工作單元完整性，您必須保留最舊的進行中或不確定工作單元的日誌資料。

為了協助您管理系統，佇列管理程式會在每一個日誌保存檔中偵測舊的工作單元，並在訊息 CSQJ160 及 CSQJ161 中報告它們。 內部作業會讀取這些舊工作單元的工作日誌資訊，並以更簡潔的形式將它重新寫入日誌中的現行位置。 訊息 CSQR026 指出何時發生。 MQSC 指令 DISPLAY USAGE TYPE (DATASET) 也可以協助您管理日誌資料的保留。 指令會報告下列三項回復資訊:

1. 工作單元回復必須保留多少日誌。
2. 頁集的媒體回復必須保留多少日誌。
3. 對於佇列共用群組中的佇列管理程式，必須保留多少日誌才能進行 CF 結構的媒體回復。

對於其中每一個資訊片段，會嘗試將所需的最舊日誌資料對映至資料集。 當新的工作單元開始和停止時，預期 (1) 會移至日誌中較新的位置。 如果未移動，長時間執行的 UOW 訊息會警告您發生問題。 (2) 如果要立即關閉並重新啟動佇列管理程式，則與頁集媒體回復相關。 它不知道您前次備份頁面集的時間，或在頁面集失敗時可能必須使用的備份。 它通常會在檢查點處理程序期間移至日誌中較新的位置，因為緩衝池中保留的變更會寫入頁集。 在 (3) 中，佇列管理程式確實知道在此佇列管理程式或佇列共用群組中其他佇列管理程式上所執行的 CF 結構備份。 不過， CF 結構回復需要合併佇列共用群組中自前次備份以來與 CF 結構互動的所有佇列管理程式的日誌資料。 這表示日誌資料由日誌記錄序號 (或 LRSN) 識別，這是時間戳記型，因此適用於整個佇列共用群組，而不是在佇列共用群組中不同佇列管理程式上不同的 RBA。 它通常會移至日誌中較新的位置，因為 BACKUP CFSTRUCT 指令是在佇列共用群組中的這個或其他佇列管理程式上執行。

z/OS 重設佇列管理程式的日誌

請利用這個主題來瞭解如何重設佇列管理程式的日誌。

您不得容許佇列管理程式日誌 RBA 從日誌 RBA 範圍結尾折返至 0 ，因為這會導致佇列管理程式中斷，且所有持續資料將變成無法回復。 日誌 RBA 的結尾是 FFFFFFFFFFFFFFFFFFFF 值 (如果使用 6 個位元組的 RBA) 或 FFFFFFFFFFFFFFFFFFFFFFFF (如果使用 8 個位元組的 RBA)。

佇列管理程式會發出訊息 CSQI045I、 CSQI046E、 CSQI047E、 CSQJ031D 及 CSQJ032E ，以指出使用的日誌範圍很重要，且您應該計劃採取動作以避免意外中斷執行。

當 RBA 值達到 FFF800000000 (如果使用 6 位元組日誌 RBA) 或 FFFFFFC000000000 (如果使用 8 位元組日誌 RBA) 時，佇列管理程式會終止，原因碼為 00D10257 。

如果 6 位元組日誌 RBA 在使用中，請遵循 第 398 頁的『實作較大的日誌相對位元組位址』中說明的處理程序，考量將佇列管理程式轉換為使用 8 位元組日誌 RBA ，而不是重設佇列管理程式的日誌。 將佇列管理程式轉換為使用 8 位元組日誌 RBA 所需的中斷時間比重設日誌所需的中斷時間更短，並增加您必須重設日誌之前的時段。

在佇列管理程式起始設定期間發出的訊息 CSQJ034I 指出已配置佇列管理程式的日誌 RBA 範圍結束，可用來判斷 6 位元組或 8 位元組日誌 RBA 是否在使用中。

重設佇列管理程式日誌所遵循的程序如下:

1. 解決任何未解決的工作單元。 在佇列管理程式啟動時，未解決的工作單元數目會顯示在訊息 CSQR005I 中，作為 INDOUBT 計數。 在每一個檢查點，以及在佇列管理程式關閉時，佇列管理程式會自動發出指令

   DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED)，提供未解決工作單元的相關資訊。

   如需解決回復單元的相關資訊，請參閱 如何解決不確定的回復單元 。 最終手段是使用 **RESOLVE INDOUBT** MQSC 指令來手動解決不確定的回復單元。

2. 完全關閉佇列管理程式。

   您可以使用 **STOP QMGR** 或 **STOP QMGR MODE(FORCE)**，因為這兩個指令都會將任何已變更的頁面從緩衝池清除至頁集。

3. 如果佇列管理程式是佇列共用群組的一部分，請針對佇列共用群組中的所有結構，在其他佇列管理程式上執行 CFSTRUCT 備份。 這可確保最近的備份不在此佇列管理程式的日誌中，且 CFSTRUCT 回復不需要此佇列管理程式的日誌。

4. 使用 CSQJU003 定義新的日誌和 BSDS (如需使用變更日誌庫存公用程式的相關資訊，請參閱 變更日誌庫存公用程式)。

5. 針對此佇列管理程式的所有頁集執行 **CSQUTIL RESETPAGE** (如需使用此功能的相關資訊，請參閱 複製頁面並重設日誌)。 請注意，頁集 RBA 可以獨立重設，因此可以提交多個並行工作 (例如，每個頁集一個)，以減少此步驟的經歷時間。

6. 重新啟動佇列管理程式

相關概念

第 398 頁的『實作較大的日誌相對位元組位址』

在 IBM MQ for z/OS 8.0 之前， IBM MQ for z/OS 使用 6 個位元組的日誌 RBA 來識別日誌內資料的位置。 從 IBM MQ for z/OS 8.0 開始，日誌 RBA 可以有 8 個位元組的長度，這會增加您必須重設日誌之前的時段。

z/OS 實作較大的日誌相對位元組位址

在 IBM MQ for z/OS 8.0 之前， IBM MQ for z/OS 使用 6 個位元組的日誌 RBA 來識別日誌內資料的位置。 從 IBM MQ for z/OS 8.0 開始，日誌 RBA 可以有 8 個位元組的長度，這會增加您必須重設日誌之前的時段。

對於 IBM MQ 9.2.0 長期支援佇列管理程式，以及 IBM MQ 9.2.4 之前建立的 Continuous Delivery 佇列管理程式，需要明確啟用此特性。

V 9.2.5 對於在 IBM MQ 9.2.5 以及更新版本建立的佇列管理程式，已啟用此特性。

如需規劃啟用 8 個位元組日誌 RBA 時的考量，請參閱 規劃增加可定址日誌範圍上限 。

重要: V 9.2.0 如果您的佇列管理程式不是佇列共用群組的一部分，且您隨後移轉回 IBM MQ for z/OS 9.0.0(已啟用 8 個位元組的日誌 RBA)，請確保在那些版次上使用 **OPMODE**=*NEWFUNC*， *900*，否則佇列管理程式將無法啟動。

依照顯示的順序執行下列指示，以在單一 IBM MQ for z/OS 佇列管理程式上啟用 8 個位元組的日誌 RBA:

1. V 9.2.0 如果佇列管理程式位於佇列共用群組中，在移至步驟 第 398 頁的『2』之前，請確定佇列共用群組中位於 IBM MQ for z/OS 9.0.0 的所有佇列管理程式都與 **OPMODE**= (*NEWFUNC,900*) 一起執行。

   您不需要採取佇列共用群組中斷總計即可執行此動作。 您可以依序停止每一個位於 IBM MQ for z/OS 9.0.0 的佇列管理程式，將它變更為 **OPMODE**= (*NEWFUNC,900*) 並重新啟動它。

   一旦佇列共用群組中的所有 IBM MQ for z/OS 9.0.0 佇列管理程式都與 **OPMODE**= (*NEWFUNC,900*) 一起執行，請針對佇列共用群組中的每一個佇列管理程式執行下列步驟，直到所有佇列管理程式都與新的 BSDS 一起執行為止。

2. 將具有類似屬性的新 BSDS 資料集配置給現行 BSDS。 您可以自訂範例 CSQ4BSDS 並刪除任何不相關的陳述式，也可以使用現有的 JCL，但將 BSDS 名稱變更為類似 ++HLQ++.NEW.BSDS01 的名稱。

   附註:

a. 在提交工作以配置新的 BSDS 之前，請先檢查新 BSDS 的屬性。 唯一可能變更的屬性是 BSDS 的大小。

b. 新的 BSDS 包含現行 BSDS 的更多資料，因此您必須確保為新資料集配置足夠的可用空間。 thlqual.SCSQPROC(CSQ4BSDS) 中的範例 JCL 在定義新的 BSDS 時包含建議值。

3. 完全關閉佇列管理程式。
4. 執行 BSDS 轉換公用程式 (CSQJUCNV)，將現有的 BSDS 轉換成新的 BSDS 資料集。 這通常需要幾秒鐘才能執行。

   在此處理程序期間將不會變更現有的 BSDS，而且在轉換失敗的情況下，您可以使用該 BSDS 來起始設定佇列管理程式。
5. 將現行 BSDS 重新命名為舊的 BSDS，並將新的 BSDS 重新命名為現行 BSDS，以便在您下次重新啟動佇列管理程式時使用新的資料集。 您可以使用 DFSMS Access Method Services ALTER 指令，例如:

   ```
   ALTER '++HLQ++.BSDS01' NEWNAME('++HLQ++.OLD.BSDS01')
   ALTER '++HLQ++.NEW.BSDS01' NEWNAME('++HLQ++.BSDS01')
   ```

   請確定您也發出指令來重新命名 VSAM 叢集的資料和索引部分。
6. 重新啟動佇列管理程式。 它應該在使用 6 個位元組日誌 RBA 時所執行的相同時間量啟動。

   如果由於無法存取已轉換的 BSDS 而無法順利重新啟動佇列管理程式，請嘗試識別失敗的原因，解決問題並重試作業。 必要的話，請聯絡 IBM 支援中心，以取得協助。

   必要的話，可以透過下列方式在此時取消變更:

   a. 將現行 BSDS 重新命名為新的 BSDS。

   b. 將舊 BSDS 重新命名為現行 BSDS。

   c. 正在重新啟動佇列管理程式。

   使用已轉換的 BSDS 順利重新啟動佇列管理程式之後，請勿嘗試使用舊的 BSDS 來啟動佇列管理程式。
7. 在佇列管理程式起始設定期間發出訊息 CSQJ034I，以指出所配置佇列管理程式的日誌 RBA 結束。 確認顯示的日誌 RBA 範圍結束為 FFFFFFFFFFFFFFFFFF。 這指出正在使用 8 個位元組的日誌 RBA。

註: 若要在新佇列管理程式上啟用 8 位元組日誌 RBA，在第一次啟動之前，您必須先建立空的第 1 版格式 BSDS，並使用該格式作為 BSDS 轉換公用程式的輸入，以產生第 2 版格式 BSDS。 如需如何執行此程序的相關資訊，請參閱 建立引導及日誌資料集。

相關概念

較大的日誌相對位元組位址

相關工作

規劃增加可定址日誌範圍上限

相關參考

BSDS 轉換公用程式 (CSQJUCNV)

z/OS 管理引導資料集 (BSDS)

引導資料集 (BSDS) 用來參照日誌資料集及日誌記錄。 請利用這個主題來瞭解如何檢查、變更及回復 BSDS。

如需相關資訊，請參閱 引導資料集。

這個主題說明管理引導資料集所涉及的作業。 它包含下列區段:

- 第 399 頁的『瞭解 BSDS 包含的內容』
- 第 401 頁的『變更 BSDS』
- 第 404 頁的『回復 BSDS』

z/OS 瞭解 *BSDS* 包含的內容

您可以使用列印日誌對映公用程式 (CSQJU004) 來檢查 BSDS 的內容。

列印日誌對映公用程式 (CSQJU004) 是一個批次公用程式，列出儲存在 BSDS 中的資訊。 如需執行它的指示，請參閱 列印日誌對映公用程式。

BSDS 包含:

- 時間戳記
- 作用中日誌資料集狀態

BSDS 中的時間戳記

列印日誌對映公用程式的輸出會顯示時間戳記，用來記錄儲存在 BSDS 中的各種系統事件的日期和時間。

下列時間戳記包含在報告的標頭區段中:

系統時間戳記
反映前次更新 BSDS 的日期和時間。 在下列情況下，可以更新 BSDS 時間戳記:

- 即會啟動佇列管理程式。
- 在日誌寫入活動期間達到寫入臨界值。 視您指定的輸出緩衝區數目及系統活動速率而定， BSDS 可能每秒更新數次，或數秒、分鐘甚至數小時都不會更新。 如需寫入臨界值的詳細資料，請參閱使用 CSQ6LOGP 中 CSQ6LOGP 巨集的 WRTHRSH 參數。
- 由於發生錯誤， IBM MQ 從其正常雙重 BSDS 模式進入單一 BSDS 模式。 當取得、插入、指向、更新或刪除 BSDS 記錄的要求不成功時，可能會發生這種情況。 發生此錯誤時， IBM MQ 會更新剩餘 BSDS 中的時間戳記，以強制時間戳記與已停用 BSDS 不符。

公用程式時間戳記
變更日誌庫存公用程式變更 BSDS 內容的日期和時間 (CSQJU003)。

報告的作用中和保存日誌資料集部分包含下列時間戳記:

作用中日誌日期
在 BSDS 中建立作用中日誌項目的日期，即完成 CSQJU003 NEWLOG 時。

作用中日誌時間
在 BSDS 中建立作用中日誌項目的時間，即完成 CSQJU003 NEWLOG 時。

保存日誌日期
在 BSDS 中建立保存日誌項目的日期，亦即當完成 CSQJU003 NEWLOG 或完成保存本身時。

保存日誌時間
在 BSDS 中建立保存日誌項目的時間，即完成 CSQJU003 NEWLOG 或完成保存本身的時間。

作用中日誌資料集狀態

BSDS 會將作用中日誌資料集的狀態記錄為下列其中一項:

新建
資料集已定義但從未被 IBM MQ 使用，或日誌已截斷至第一次使用資料集之前的點。 在任一情況下，資料集啟動及結束 RBA 值都會重設為零。

可重複使用
已定義資料集，但從未由 IBM MQ 使用，或已卸載資料集。 在列印日誌對映輸出中，最後一個可重複使用資料集的開始 RBA 值等於最後一個保存日誌資料集的開始 RBA 值。

不可重複使用
資料集包含尚未卸載的記錄。

STOPPED
卸載處理器在讀取記錄時發生錯誤，無法從作用中日誌的另一個副本取得該記錄。

已截斷
您可以:

- 發生 I/O 錯誤， IBM MQ 已停止寫入此資料集。 會卸載作用中日誌資料集，從啟動 RBA 開始，並繼續到截斷的作用中日誌資料集中最後一個有效記錄區段。 最後一個有效記錄區段的 RBA 低於作用中日誌資料集的結束 RBA。 記載會切換至下一個可用的作用中日誌資料集，並繼續不中斷。

or

- 已呼叫 ARCHIVE LOG 函數，已截斷作用中日誌。

狀態會出現在列印日誌對映公用程式的輸出中。

z/OS 變更 *BSDS*

您不需要採取特殊步驟，即可使用記載事件的記錄來更新 BSDS，因為 IBM MQ 會自動這麼做。

不過，如果您執行下列任何動作，則可能想要變更 BSDS:

- 新增更多作用中日誌資料集。
- 例如，提供較大的作用中日誌配置時，將作用中日誌資料集複製到新配置的資料集。
- 將日誌資料集移至其他裝置。
- 回復損壞的 BSDS。
- 捨棄過期的保存日誌資料集。

您可以執行變更日誌庫存公用程式 (CSQJU003) 來變更 BSDS。 只有在佇列管理程式非作用中，或您可能得到不一致的結果時，才執行此公用程式。 公用程式的動作是由 SYSIN 資料集中的陳述式所控制。 本節顯示數個範例。 如需完整指示，請參閱 變更日誌庫存公用程式。

只有在佇列管理程式非作用中時，您才能複製作用中日誌資料集，因為在佇列管理程式啟動時， IBM MQ 會將作用中日誌資料集配置成專用 (DISP = OLD)。

z/OS 作用中日誌的變更

請利用這個主題來瞭解如何使用 BSDS 來變更作用中日誌。

您可以使用變更日誌公用程式，在 BSDS 中新增、刪除及記錄作用中日誌的項目。 這裡只顯示範例; 將顯示的資料集名稱取代為您要使用的資料集名稱。 如需公用程式的詳細資料，請參閱 變更日誌庫存公用程式。

如需相關資訊，請參閱下列各節:

- 將記錄項目新增至 BSDS
- 從 BSDS 刪除作用中日誌資料集的相關資訊
- 記錄 BSDS 中日誌資料集的相關資訊
- 增加作用中日誌的大小
- 使用 CSQJUFMT

將記錄項目新增至 BSDS

如果作用中日誌已標示為「已停止」，則不會重複用於記載; 不過，它會繼續用於讀取。 使用存取方法服務來定義新的作用中日誌資料集，然後使用變更日誌庫存公用程式在 BSDS 中登錄新的資料集。 例如，使用:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

如果您要將舊作用中日誌資料集的內容複製到新的內容集，您也可以提供 RBA 範圍，以及 NEWLOG 函數的開始和結束時間戳記。

從 BSDS 刪除作用中日誌資料集的相關資訊

若要從 BSDS 刪除作用中日誌資料集的相關資訊，您可以使用:

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

記錄 BSDS 中日誌資料集的相關資訊

若要記錄 BSDS 中現有作用中日誌資料集的相關資訊，請使用:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

您可能需要在 BSDS 中插入包含此類型資訊的記錄，因為:

- 已刪除資料集的項目，但再次需要。
- 您正在將一個作用中日誌資料集的內容複製到另一個資料集。
- 您正在從備份副本回復 BSDS。

增加作用中日誌的大小

實現這一程序有兩種方法。

1. 當佇列管理程式處於作用中狀態時:
   a. 使用 JCL 來定義新的較大日誌資料集。
   b. 使用 MQSC DEFINE LOG 指令，將新的日誌資料集新增至作用中佇列管理程式。
   c. 請使用 MQSC ARCHIVE LOG 指令來移動現行作用中日誌，使其成為新的較大日誌。
   d. 等待較小作用中日誌資料集的保存完成。
   e. 使用 CSQJU003 公用程式移除舊的小型作用中日誌，以關閉佇列管理程式。
   f. 重新啟動佇列管理程式。
2. 當佇列管理程式處於非作用中狀態時:
   a. 停止佇列管理程式。 此步驟是必要的，因為 IBM MQ 會配置所有作用中日誌資料集，以供其在作用中時專用。
   b. 搭配使用 Access Method Services ALTER 與 NEWNAME 選項，以重新命名作用中日誌資料集。
   c. 使用 Access Method Services DEFINE 來定義較大的作用中日誌資料集。

      透過重複使用舊資料集名稱，您無需執行變更日誌庫存公用程式，即可在 BSDS 中建立新名稱。舊資料集名稱及正確的 RBA 範圍已在 BSDS 中。
   d. 使用 Access Method Services REPRO ，將舊 (重新命名) 資料集複製到其適當的新資料集。

      註: 此步驟可能需要很長時間，因此您的企業在此期間可能無法運作。
   e. 啟動佇列管理程式。

如果您的所有日誌資料集都是相同的大小，則您的系統作業會更一致且更有效率。 如果日誌資料集的大小不同，則更難以追蹤系統的日誌，因此可能會浪費空間。

使用 CSQJUFMT

增加作用中日誌的大小時，請勿執行 CSQJUFMT 格式。

如果您執行 CSQJUFMT (以便在佇列管理程式第一次寫入新的作用中日誌時提供效能優勢) ，您會收到下列訊息:

```
IEC070I 203-204,XS95GTLX,REPRO02,OUTPUT,B857,SPMG02, 358
IEC070I MG.W.MG4E.LOGCOPY1.DS02,MG.W.MG4E.LOGCOPY1.DS02.DATA,
IDC3302I ACTION ERROR ON MG.W.MG4E.LOGCOPY1.DS02
IDC3351I ** VSAM I/O RETURN CODE IS 28 - RPLFDBWD = X'2908001C'
IDC31467I MAXIMUM ERROR LIMIT REACHED.

IDC0005I NUMBER OF RECORDS PROCESSED WAS 0
```

此外，如果您使用 Access Method Services REPRO ，請確保定義新的空日誌。

如果您使用 REPRO 將舊 (重新命名) 資料集複製到其個別的新資料集，則預設值為 NOREPLACE。

這表示 REPRO 不會取代已在指定資料集上的記錄。 對資料集執行格式化時，會重設 RBA 值。 淨結果是格式化之後不是空的資料集。

z/OS 保存日誌的變更

請利用這個主題來瞭解如何變更保存日誌。

您可以在保存日誌的 BSDS 中新增、刪除及變更項目的密碼。 這裡只顯示範例; 將顯示的資料集名稱取代為您要使用的資料集名稱。 如需公用程式的詳細資料，請參閱 變更日誌庫存公用程式。

- 新增保存日誌
- 刪除保存日誌
- 變更保存日誌的密碼

新增保存日誌

當物件的回復取決於讀取現有的保存日誌資料集時， BSDS 必須包含該資料集的相關資訊，讓 IBM MQ 可以找到它。 若要登錄 BSDS 中現有保存日誌資料集的相關資訊，請使用:

```
NEWLOG DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,
UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

刪除保存日誌

若要刪除一或多個磁區上的整個保存日誌資料集，請使用:

```
DELETE DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

變更保存日誌的密碼

如果您變更現有保存日誌資料集的密碼，則也必須變更 BSDS 中的資訊。

1. 使用列印日誌對映公用程式列出 BSDS。
2. 使用 CSQJU003 公用程式的 DELETE 功能，刪除具有已變更密碼之保存日誌資料集的項目 (請參閱 變更日誌庫存公用程式 主題)。
3. 將資料集命名為新的保存日誌資料集。 使用 CSQJU003 公用程式的 NEWLOG 函數 (請參閱主題 變更日誌庫存公用程式)， 並提供新密碼、開始及結束 RBA， 以及磁區序號 (可在列印日誌對映公用程式輸出中找到，請參閱 列印日誌對映公用程式)。

若要變更新保存日誌資料集的密碼，請使用:

```
ARCHIVE PASSWORD= password
```

若要停止將密碼放置在新的保存日誌資料集上，請使用:

```
ARCHIVE NOPASSWD
```

註: 只有在您沒有外部安全管理程式時，才使用 ARCHIVE 公用程式功能。

z/OS 變更日誌和 *BSDS* 的高階限定元 *(HLQ)*

請利用這個主題來瞭解變更高階限定元 (HLQ) 所需的程序。

開始之前

在將任何日誌或資料集複製到新的資料集之前，您必須正常結束佇列管理程式。 這是為了確保資料一致，且在重新啟動期間不需要任何回復。

關於這項作業

此作業提供如何變更日誌及 BSDS 之 HLQ 的相關資訊。 若要執行此動作，請遵循下列步驟:

程序

1. 執行日誌列印公用程式 CSQJU004 ，以記錄日誌資料集資訊。 稍後需要此資訊。
2. 您可以:
   a) 在要重新命名的日誌和 BSDS 資料集上執行 DSS 備份及還原，或
   b) 使用 AMS DEFINE 和 REPRO 來建立 HLQ 資料集，並從舊資料集複製資料。
3. 修改 MSTR 和 CHIN 程序以指向新的資料集。
4. 使用 CSQJU003 刪除 BSDS 新副本中的舊日誌資訊。
5. 使用 CSQJU003 的 NEWLOG 函數，將新的日誌資料集定義至新的 BSDS。
   除了 HLQ 之外，每一個日誌的所有相關資訊都保持相同。
6. 新的 BSDS 應該反映針對舊 BSDS 中舊日誌所記錄的相同資訊。
   HLQ 應該是唯一改變的東西。

下一步

在啟動佇列管理程式之前，請先比較新舊 BSDS 的 CSQJU004 輸出，以確定它們看起來完全相同 (HLQ 除外)。

註: 執行這些作業時必須小心。 不正確的動作可能會導致無法復原的狀況。 請檢查 PRINT LOG MAP UTILITY 輸出，並確定已包含回復或重新啟動所需的所有資訊。

z/OS 回復 *BSDS*

如果 IBM MQ 以雙重 BSDS 模式運作，且有一個 BSDS 變成損壞，則強制 IBM MQ 進入單一 BSDS 模式，IBM MQ 會繼續運作而沒有問題 (直到下次重新啟動為止)。

若要讓環境回到雙重 BSDS 模式，請執行下列動作:

1. 使用「存取方法服務」來重新命名或刪除損壞的 BSDS ，以及定義與損壞 BSDS 同名的新 BSDS。 在 thlqual.SCSQPROC 的工作 CSQ4BREC 中可以找到控制陳述式範例。
2. 發出 IBM MQ 指令 RECOVER BSDS ，以在新配置的資料集中建立有效 BSDS 的副本，並恢復雙重 BSDS 模式。

如果 IBM MQ 以單一 BSDS 模式運作且 BSDS 已損壞，或如果 IBM MQ 以雙重 BSDS 模式運作且兩個 BSDS 都已損壞，則佇列管理程式會停止並在 BSDS 資料集修復之前不會重新啟動。 在此情況下:

1. 尋找與最新保存日誌資料集相關聯的 BSDS。 最近保存日誌的資料集名稱會出現在工作日誌最後一次出現的訊息 CSQJ003I 中，指出卸載處理已順利完成。 在準備此程序的其餘部分時，最好保留該訊息所指出的所有成功保存檔的日誌:
   - 如果保存日誌位於 DASD 上，則會在任何可用的 DASD 上配置 BSDS。 BSDS 名稱類似於對應的保存日誌資料集名稱; 只將最後一個限定元的第一個字母從 A 變更為 B ，如下列範例所示:

     保存日誌名稱
     CSQ.ARCHLOG1. ***A*** 0000001

     BSDS 副本名稱
     CSQ.ARCHLOG1. ***B*** 0000001
   - 如果保存日誌位於磁帶上，則 BSDS 是第一個保存日誌磁區的第一個資料集。 在後續磁區上不會重複 BSDS。

2. 如果最新保存日誌資料集沒有 BSDS 的副本 (例如，因為卸載時發生錯誤)，請從先前的卸載處理中尋找先前的 BSDS 副本。
3. 使用 Access Method Services ALTER 指令搭配 NEWNAME 選項，重新命名 損壞的 BSDS。 如果您要刪除損壞的 BSDS，請使用「存取方法服務 DELETE」指令。 對於每一個損壞的 BSDS，請使用「存取方法服務」來定義新的 BSDS 作為取代資料集。 thlqual.SCSQPROC 中的工作 CSQ4BREC 包含存取方法服務控制陳述式，可定義新的 BSDS。
4. 使用 Access Method Services REPRO 指令，將保存日誌中的 BSDS 複製到您在步驟 第 405 頁的『3』中定義的其中一個取代 BSDS。 請勿將任何資料複製到第二個取代 BSDS，您可以在步驟 第 406 頁的『5』中這樣做。

   a. 列印取代 BSDS 的內容。

      使用列印日誌對映公用程式 (CSQJU004) 來列印取代 BSDS 的內容。 這可讓您在繼續進行回復工作之前檢閱取代 BSDS 的內容。

   b. 更新取代 BSDS 中的保存日誌資料集庫存。

      請檢查列印日誌對映公用程式的輸出，並檢查取代 BSDS 是否不包含從中複製 BSDS 的保存日誌記錄。 如果取代 BSDS 是舊副本，則其庫存可能不會包含最近建立的所有保存日誌資料集。 必須更新保存日誌資料集的 BSDS 庫存，以反映現行子系統庫存。

      使用變更日誌庫存公用程式 (CSQJU003) NEWLOG 陳述式來更新取代 BSDS，並新增從中複製 BSDS 的保存日誌記錄。 如果保存日誌資料集受密碼保護，請使用 NEWLOG 函數的 PASSWORD 選項。 此外，如果已編目保存日誌資料集，請確定 NEWLOG 函數的 CATALOG 選項已適當地設為 CATALOG=YES。 使用 NEWLOG 陳述式來新增任何比 BSDS 副本晚建立的其他保存日誌資料集。

   c. 更新取代 BSDS 中的密碼。

      BSDS 包含保存日誌資料集及作用中日誌資料集的密碼。 若要確保替代 BSDS 中的密碼反映您安裝所使用的現行密碼，請搭配使用變更日誌庫存 ARCHIVE 公用程式功能與 PASSWORD 選項。

   d. 更新取代 BSDS 中的作用中日誌資料集庫存。

      在特殊情況下，自複製 BSDS 之後，您的安裝可能已新增、刪除或重新命名作用中日誌資料集。 在此情況下，取代 BSDS 不會反映您安裝目前使用中的作用中日誌資料集實際數目或名稱。

      如果您需要從取代 BSDS 日誌庫存中刪除作用中日誌資料集，請使用變更日誌庫存公用程式 DELETE 功能。

      如果您需要將作用中日誌資料集新增至取代 BSDS 日誌庫存，請使用變更日誌庫存公用程式 NEWLOG 功能。 請確定已在 NEWLOG 函數上正確地指定 RBA 範圍。 如果作用中日誌資料集受密碼保護，請使用 PASSWORD 選項。

      如果您需要重新命名取代 BSDS 日誌庫存中的作用中日誌資料集，請使用變更日誌庫存公用程式 DELETE 函數，後面接著 NEWLOG 函數。 請確定已在 NEWLOG 函數上正確地指定 RBA 範圍。 如果作用中日誌資料集受密碼保護，請使用 PASSWORD 選項。

   e. 更新取代 BSDS 中的作用中日誌 RBA 範圍。

      稍後，當佇列管理程式重新啟動時，它會比較 BSDS 中列出的作用中日誌資料集的 RBA 與實際作用中日誌資料集中找到的 RBA。 如果 RBA 不同意，則佇列管理程式不會重新啟動。 使用 BSDS 的舊副本時，問題會放大。 若要解決此問題，請使用變更日誌庫存公用程式 (CSQJU003)，使用實際作用中日誌資料集中的 RBA 來調整 BSDS 中找到的 RBA。 您可以透過下列方式執行此動作:

      - 使用列印日誌記錄公用程式 (CSQ1LOGP) 來列印作用中日誌資料集的摘要報告。 這會顯示開始及結束 RBA。
      - 已知所有作用中日誌資料集的 RBA 時，比較實際 RBA 範圍與您剛列印的 RBA 範圍。

        如果所有作用中日誌資料集的 RBA 範圍相等，則您可以繼續進行下一個回復步驟，而不需要任何其他工作。

        如果 RBA 範圍不相等，請調整 BSDS 中的值以反映實際值。 對於需要調整 RBA 範圍的每一個作用中日誌資料集，請使用變更日誌庫存公用程式 DELETE 功能，從取代 BSDS 中的庫存刪除作用中日誌資料集。 然後使用 NEWLOG 函數，將作用中日誌資料集重新定義為 BSDS。 如果作用中日誌資料集受密碼保護，請使用 NEWLOG 函數的 PASSWORD 選項。

f. 如果每一個作用中日誌副本只指定兩個作用中日誌資料集，則在重新啟動佇列管理程式期間， IBM MQ 會有困難。 當其中一個作用中日誌資料集已滿且尚未卸載，而第二個作用中日誌資料集即將填滿時，可能會發生此問題。 在此情況下，請為作用中日誌的每一個副本新增作用中日誌資料集，並在取代 BSDS 日誌庫存中定義每一個新的作用中日誌資料集。

   使用「存取方法服務 DEFINE」指令為作用中日誌的每一個副本定義新的作用中日誌資料集，並使用變更日誌庫存公用程式 NEWLOG 功能在取代 BSDS 中定義新的作用中日誌資料集。 您不需要在 NEWLOG 陳述式上指定 RBA 範圍。 不過，如果作用中日誌資料集受到密碼保護，請使用 NEWLOG 函數的 PASSWORD 選項。 可以在 thlqual.SCSQPROC 中的工作 CSQ4LREC 中找到完成此作業的控制陳述式範例。

5. 將更新的 BSDS 複製到第二個新的 BSDS 資料集。 現在 BSDS 是相同的。

   此時使用列印日誌對映公用程式 (CSQJU004) 來列印第二個取代 BSDS 的內容。
6. 如需您遺失現行作用中日誌資料集時要執行之動作的相關資訊，請參閱 作用中日誌問題 。
7. 使用新建構的 BSDS 來重新啟動佇列管理程式。 IBM MQ 決定現行 RBA 以及需要保存哪些作用中日誌。

z/OS 管理頁面集

請利用這個主題來瞭解如何管理與佇列管理程式相關聯的頁面集。

本主題說明如何新增、複製及一般管理與佇列管理程式相關聯的頁面集。 它包含下列區段:

- 第 406 頁的『如何變更頁面集的高階限定元 (HLQ)』
- 第 406 頁的『如何將頁面集新增至佇列管理程式』
- 第 407 頁的『當其中一個頁面集已滿時要執行的動作』
- 第 407 頁的『如何平衡頁面集上的負載』
- 如何增加頁集的大小
- 第 410 頁的『如何減少頁面集』
- 第 411 頁的『如何重新建立頁集』
- 第 411 頁的『如何備份及回復頁集』
- 第 414 頁的『如何刪除頁面集』
- 第 414 頁的『如何使用 CSQUTIL 備份及還原佇列』

如需頁集、儲存類別、緩衝區及緩衝池的說明，以及一些適用的效能考量，請參閱 頁集 。

如何變更頁面集的高階限定元 (HLQ)

此作業提供如何變更頁面集之 HLQ 的相關資訊。 若要執行此作業，請執行下列動作:

1. 定義新的 HLQ 頁面集。
2. 如果大小配置與舊頁集相同，請使用 REPRO 將現有頁集複製到空的新 HLQ 頁集。
3. 如果您要增加頁集的大小，請使用 CSQUTIL 的 FORMAT 函數來格式化目的地頁面，然後使用 CSQUTIL 的 COPYPAGE 函數將來源頁集的所有訊息複製到目的地頁集。

   如需相關資訊，請參閱 格式化頁集 (FORMAT)及 展開頁集 (COPYPAGE)。
4. 變更佇列管理程式程序中的 CSQP00xx DD 陳述式，以指向新的 HLQ 頁面集。

重新啟動佇列管理程式，並驗證對頁集所做的變更。

如何將頁面集新增至佇列管理程式

此說明假設您具有已在執行中的佇列管理程式。 例如，如果您的佇列管理程式必須使用新的佇列來處理新的應用程式，則您可能需要新增頁集。

若要新增頁集，請使用下列程序:

1. 定義並格式化新頁集。 您可以使用 thlqual.SCSQPROC(CSQ4PAGE) 中的範例 JCL 作為基準。 如需相關資訊，請參閱 格式化頁集 (FORMAT)。

   請小心不要格式化任何使用中的頁集，除非這是您想要的。 若是如此，請使用 FORMAT 公用程式函數的 FORCE 選項。
2. 搭配使用 DEFINE PSID 指令與 DSN 選項，以建立頁集與緩衝池的關聯。
3. 透過發出 DEFINE STGCLASS 指令，為您的頁集新增適當的儲存類別定義。
4. 選擇性地記載佇列管理程式的配置方式:
   a. 將新頁集新增至佇列管理程式的已啟動作業程序。
   b. 將新頁集的定義新增至 CSQINP1 起始設定資料集。
   c. 將新儲存類別的定義新增至 CSQ4INYR 起始設定資料集成員。

如需 DEFINE PSID 及 DEFINE STGCLASS 指令的詳細資料，請參閱 DEFINE PSID 及 DEFINE STGCLASS。

當其中一個頁面集已滿時要執行的動作

您可以使用 IBM MQ 指令 DISPLAY USAGE ，以瞭解頁集的使用率。 例如，指令:

```
DISPLAY USAGE PSID(03)
```

顯示頁集 03 的現行狀態。 這會告訴您此頁集有多少可用頁面。

如果您已定義頁面集的次要延伸範圍，則每次填滿時都會動態展開這些延伸範圍。 最終會使用所有次要延伸範圍，或沒有進一步可用的磁碟空間。 如果發生這種情況，應用程式會收到回覆碼 MQRC_STORAGE_MEDIUM_FULL。

如果應用程式收到來自 MQI 呼叫的回覆碼 MQRC_STORAGE_MEDIUM_FULL ，則表示頁集上沒有足夠的剩餘空間。 如果問題持續或可能再次發生，您必須執行一些動作來解決它。

您可以透過下列數種方式來處理此問題:

- 將佇列從一個頁集移至另一個頁集，以平衡頁集之間的負載。
- 展開頁面集。 請參閱第 409 頁的『如何增加頁集的大小』以取得相關指示。
- 請重新定義頁集，讓它可以擴充到超過 4 GB 的大小上限 (64 GB)。 如需指示，請參閱 定義大於 4 GB 的頁集 。

如何平衡頁面集上的負載

頁集上的負載平衡表示將與一個以上佇列相關聯的訊息從一個頁集移至另一個頁集 (較少使用)。 如果展開頁集並不實際，請使用此技術。

若要識別哪些佇列正在使用頁集，請使用適當的 IBM MQ 指令。 例如，若要找出哪些佇列對映至頁集 02 ，請先使用下列指令找出哪些儲存類別對映至頁集 02:

```
DISPLAY STGCLASS(*) PSID(02)
```

然後使用下列指令來找出哪些佇列使用哪個儲存類別:

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

移動非共用佇列

若要將佇列及其訊息從一個頁集移至另一個頁集，請使用 MQSC MOVE QLOCAL 指令 (在 MOVE QLOCAL 中說明)。 當您已識別要移至新頁集的一或多個佇列時，請針對下列每一個佇列遵循下列程序:

1. 請確定您要移動的佇列未被任何應用程式使用 (亦即， DISPLAY QSTATUS 指令中的 IPPROCS 及 OPPROCS 值為零)，且沒有未確定的訊息 (DISPLAY QSTATUS 指令中的 UNCOM 值為 NO)。

   註: 確保此狀態繼續的唯一方法是暫時變更佇列的安全授權。 如需相關資訊，請參閱 佇列安全的設定檔 。

   如果您無法執行此動作，則在應用程式開始使用佇列時，即使採取諸如設定 PUT (DISABLED) 之類的預防措施步驟，此程序中的後續階段也可能會失敗。 不過，此程序絕不會遺失訊息。
2. 透過變更佇列定義以停用 MQPUT ，防止應用程式將訊息放置在要移動的佇列上。 將佇列定義變更為 PUT (DISABLED)。
3. 使用下列指令，定義與要移動的佇列具有相同屬性的暫時佇列:

   ```
   DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
   ```

   註: 如果前次執行時已存在此暫時佇列，請先刪除它，然後再執行定義。
4. 使用下列指令，將訊息移至暫時佇列:

   ```
   MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
   ```
5. 使用下列指令來刪除您要移動的佇列:

   ```
   DELETE QLOCAL(QUEUE_TO_MOVE)
   ```
6. 定義對映至所需頁集的新儲存類別，例如:

   ```
   DEFINE STGCLASS(NEW) PSID(nn)
   ```

   將新的儲存類別定義新增至 CSQINP2 資料集，備妥可進行下一次佇列管理程式重新啟動。
7. 透過變更儲存類別屬性，重新定義您要移動的佇列:

   ```
   DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
   ```

   當重新定義佇列時，它是以步驟 第 408 頁的『3』中所建立的暫時佇列為基礎。
8. 使用下列指令，將訊息移回新佇列:

   ```
   MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
   ```
9. 不再需要在步驟 第 408 頁的『3』 中建立的佇列。 請使用下列指令來刪除它:

   ```
   DELETE QL(TEMP_QUEUE)
   ```
10. 如果要移動的佇列定義在 CSQINP2 資料集中，請變更 CSQINP2 資料集中適當 DEFINE QLOCAL 指令的 STGCLASS 屬性。 新增 REPLACE 關鍵字，以取代現有的佇列定義。

第 409 頁的圖 37 顯示從負載平衡工作擷取。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE STGCLASS(NEW) PSID(2)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

圖 *37:* 從頁面集的負載平衡工作擷取

如何增加頁集的大小

您可以起始配置大於 4 GB 的頁集，請參閱 將頁集定義為大於 4 GB

您可以指定 EXPAND (SYSTEM) 或 EXPAND (USER)，將頁集定義成在已滿時自動展開。 如果您的頁集是以 EXPAND (NONE) 來定義，您可以用下列兩種方式之一來展開它:

- 變更其定義以容許自動擴充。 請參閱 變更頁集以容許自動擴充
- 建立新的較大頁集，並將訊息從舊頁集複製到新頁集。 請參閱 將訊息移至新的較大頁集

將頁集定義為大於 4 GB

IBM MQ 可以使用大小最多為 64 GB 的頁面集，前提是資料集定義為 VSAM 的「延伸定址能力」。 延伸定址能力是 SMS 資料類別所提供的屬性。

註: 頁集和作用中日誌資料集可以位於延伸位址磁區 (EAV) 的延伸定址空間 (EAS) 中，從 z/OS V1.12 開始，保存日誌資料集也可以位於 EAS 中。

在下列範例 JCL 中顯示的範例中，管理類別 'EXTENDED' 定義給具有 'Extended addressability' 的 SMS。 如果您的現有頁集目前未定義為具有延伸定址能力，請使用下列方法來移轉至延伸定址能力格式資料集。

1. 停止佇列管理程式。
2. 使用「存取方法服務」來重新命名現有的頁面集。
3. 定義目的地頁集，其大小與現有頁集相同，但具有 DATACLas (EXTENDED)。

   註: 延伸格式資料集必須由 SMS 管理。 以下是 VSAM 資料集的要求延伸格式的機制:

   - 使用 DSNTYPE 值為 EXT 且子參數 R 或 P 的資料類別來指出必要或偏好。
   - 在 DD 陳述式上編碼 DSNTYPE=EXTREQ (需要延伸格式) 或 DSNTYPE=EXTPREF (偏好延伸格式)。
   - 在 DD 陳述式上撰寫 LIKE= 參數的程式碼，以參照現有的延伸格式資料集。

   如需相關資訊，請參閱 定義延伸格式資料集的限制。
4. 使用 CSQUTIL 的 COPYPAGE 函數，將來源頁集的所有訊息複製到目的地頁集。 如需詳細資料，請參閱 展開頁面集 (COPYPAGE) 。
5. 重新啟動佇列管理程式。
6. 變更頁集以使用系統擴充，以容許它在現行配置之外繼續成長。

下列 JCL 顯示「存取方法服務」指令範例:

```
//S1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
ALTER 'VICY.CSQ1.PAGE01' -
NEWNAME('VICY.CSQ1.PAGE01.OLD')
ALTER 'VICY.CSQ1.PAGE01.DATA' -
NEWNAME('VICY.CSQ1.PAGE01.DATA.OLD')
DEFINE CLUSTER (NAME('VICY.CSQ1.PAGE01') -
MODEL('VICY.CSQ1.PAGE01.OLD') -
DATACLAS(EXTENDED))
/*
```

變更頁集以容許自動擴充

搭配使用 ALTER PSID 指令與 EXPAND (USER) 或 EXPAND (SYSTEM) 選項。 如需展開頁集的一般資訊，請參閱 ALTER PSID 及 展開頁集 (COPYPAGE) 。

將訊息移至新的較大頁集

這項技術涉及停止及重新啟動佇列管理程式。 這會刪除重新啟動時不在共用佇列上的任何非持續訊息。 如果您有不想刪除的非持續訊息，請改用負載平衡。 如需詳細資料，請參閱第 407 頁的『如何平衡頁面集上的負載』。 在此說明中，您要展開的頁集稱為 來源 頁集; 新的較大頁集稱為 目的地 頁集。

請遵循下列步驟:

1. 停止佇列管理程式。
2. 定義目的地頁集，以確保它大於來源頁集，並具有較大的次要延伸範圍值。
3. 使用 CSQUTIL 的 FORMAT 函數來格式化目的地頁集。 如需詳細資料，請參閱 格式化頁集 (FORMAT) 。
4. 使用 CSQUTIL 的 COPYPAGE 函數，將來源頁集的所有訊息複製到目的地頁集。 如需詳細資料，請參閱 展開頁面集 (COPYPAGE) 。
5. 執行下列其中一項，以使用目的地頁集來重新啟動佇列管理程式:
   - 請變更佇列管理程式啟動型作業程序，以參照目的地頁集。
   - 使用「存取方法服務」來刪除來源頁集，然後重新命名目的地頁集，使其名稱與來源頁集的名稱相同。

請注意:

在刪除任何 IBM MQ 頁集之前，請確定您已建立必要的備份副本。

如何減少頁面集

防止 IBM MQ 管理者以外的所有使用者使用佇列管理程式。 例如，透過變更存取安全設定。

如果您的大型頁集大部分是空的 (如 DISPLAY USAGE 指令所示) ，您可能想要減少其大小。 執行此動作的程序涉及使用 CSQUTIL 的 COPY、FORMAT 及 LOAD 函數 (請參閱 IBM MQ 公用程式)。 此程序不適用於頁集零 (0) ，因為減少此頁集的大小並不實際; 唯一的方法是重新起始設定佇列管理程式 (請參閱 第 430 頁的『重新起始設定佇列管理程式』)。 此程序的必要條件是嘗試從系統中移除所有使用者，以便所有 UOW 都完成且頁面集一致。

1. 搭配使用 STOP QMGR 指令與 QUIESCE 或 FORCE 屬性，以停止佇列管理程式。
2. 使用 PSID 選項執行 CSQUTIL 的 SCOPY 函數，以複製大型頁面集中的所有訊息資料，並將它們儲存在循序資料集中。
3. 定義新的較小頁面集資料集，以取代大型頁面集。
4. 針對您在步驟 第 410 頁的『3』中建立的頁集，執行 CSQUTIL 的 FORMAT TYPE (NEW) 函數。
5. 使用在步驟 第 410 頁的『3』中建立的頁集來重新啟動佇列管理程式。
6. 執行 CSQUTIL 的 LOAD 函數，以重新載入在步驟 第 410 頁的『2』期間儲存的所有訊息。
7. 容許所有使用者存取佇列管理程式。

8. 刪除舊的大型頁面集。

如何重新建立頁集

在某些情況下，將舊頁集重新連線至佇列管理程式會很有用。 除非採取特定動作，否則當舊頁集上線時，佇列管理程式會辨識出頁集本身及檢查點記錄中所儲存的頁集回復 RBA 是舊的，因此會自動啟動頁集的媒體回復，使其保持最新。

此類媒體回復只能在佇列管理程式重新啟動時執行，而且可能需要相當長的時間，尤其是必須讀取保留在磁帶上的保存日誌時。 不過，通常在此情況下，頁集已在中間期間離線，因此日誌不包含與頁集回復相關的資訊。

可用的選項有下列三個:

容許執行完整媒體回復。

1. 停止佇列管理程式。
2. 請確定佇列管理程式的已啟動作業程序及 CSQINP1 起始設定資料集中的頁集都有可用的定義。
3. 重新啟動佇列管理程式。

容許毀損頁集上的任何訊息。

如果頁集已離線很長時間 (例如幾個月)，且現在已決定將它重複用於不同的用途，則此選項非常有用。

1. 使用具有 TYPE (NEW) 選項之 CSQUTIL 的 FORMAT 函數來格式化頁集。
2. 將頁集的定義新增至佇列管理程式及 CSQINP1 起始設定資料集的已啟動作業程序。
3. 重新啟動佇列管理程式。

使用 TYPE (NEW) 選項進行格式化會清除頁集的現行內容，並告知佇列管理程式忽略頁集相關檢查點中的任何歷程資訊。

讓頁集上線，避免媒體回復處理程序。

只有在您確定佇列管理程式完全關閉之後頁集已離線時，才使用此技術。 此選項最適合頁面集已離線一段時間的情況，通常是由於作業問題 (例如啟動佇列管理程式時執行備份)。

1. 使用具有 TYPE (REPLACE) 選項之 CSQUTIL 的 FORMAT 函數來格式化頁集。
2. 請使用 DEFINE PSID 指令搭配 DSN 選項，以動態方式將頁集新增回佇列管理程式，或容許在佇列管理程式重新啟動時新增該頁集。

使用 TYPE (REPLACE) 選項來格式化佇列管理程式已完全關閉頁集的檢查，並將它標示為不會執行媒體回復。 不會對頁面集的內容進行其他變更。

z/OS 如何備份及回復頁集

有不同的機制可用於備份及回復。 請利用這個主題來瞭解這些機制。

本節說明下列主題:

- 第 411 頁的『建立非共用資源的回復點』
- 第 412 頁的『備份頁面集』
- 第 413 頁的『回復頁集』
- 如何刪除頁面集

如需如何為共用資源建立回復點的相關資訊，請參閱 第 418 頁的『回復共用佇列』。

建立非共用資源的回復點

在下列情況下， IBM MQ 可以將物件及非共用持續訊息回復至其現行狀態:

1. 先前點的頁集副本已存在。
2. 所有 IBM MQ 日誌都可用來從該點執行回復。

這些代表非共用資源的回復點。

物件和訊息都保留在頁集上。 相同頁集中可以存在來自不同佇列的多個物件和訊息。 基於回復目的，物件及訊息無法單獨備份，因此必須整體備份頁集，以確保正確回復資料。

IBM MQ 回復日誌包含所有持續訊息及對物件所做變更的記錄。 如果 IBM MQ 失敗 (例如，由於頁集上的 I/O 錯誤)，您可以透過還原備份副本並重新啟動佇列管理程式來回復頁集。 IBM MQ 會將日誌變更從備份副本點套用至頁集。

有兩種方法可以建立回復點:

完整備份
停止佇列管理程式，這會強制對頁集進行所有更新。

這可讓您從回復點重新啟動，只使用已備份的頁面集資料集，以及從該點開始的日誌。

模糊備份
在不停止佇列管理程式的情況下取得頁面集的 模糊 備份副本。

如果您使用此方法，且稍後相關聯的日誌已損壞或遺失，則無法使用模糊頁集備份副本進行回復。 這是因為模糊頁集備份副本包含不一致的佇列管理程式狀態視圖，且取決於可用的日誌。 如果日誌無法使用，您需要回到子系統非作用中時所取得的最後一組備份頁集副本 (方法 1) 並接受從那時開始的資料流失。

方法 1: 完整備份

此方法涉及關閉佇列管理程式。 這會強制對頁面集進行所有更新，以便頁面集處於一致狀態。

1. 停止所有正在使用佇列管理程式的 IBM MQ 應用程式 (讓它們先完成)。 例如，可以透過變更存取安全或佇列設定來完成此動作。
2. 當所有活動都已完成時，顯示並解決任何不確定的回復單元。 (請依照 DISPLAY CONN 和 RESOLVE INDOUBT 中的說明，使用 DISPLAY CONN 和 RESOLVE INDOUBT 指令。)

   這會使頁面集進入一致狀態; 如果您不這麼做，則頁面集可能不一致，且您實際上正在執行模糊備份。
3. 發出 ARCHIVE LOG 指令，以確保將最新的日誌資料寫出至日誌資料集。
4. 發出 STOP QMGR MODE (QUIESCE) 指令。 記錄 CSQI024I 或 CSQI025I 訊息中的最低 RBA 值 (如需相關資訊，請參閱 CSQI024I 及 CSQI025I)。 您應該將日誌資料集從 RBA 值所指示的日誌資料集，保留到現行日誌資料集為止。
5. 取得所有佇列管理程式頁集的備份副本 (請參閱 第 412 頁的『備份頁面集』)。

方法 2: 模糊備份

此方法不涉及關閉佇列管理程式。 因此，在備份處理程序期間，虛擬儲存體緩衝區中可能有更新項目。 這表示頁集不是處於一致狀態，只能用於日誌的回復。

1. 發出 DISPLAY USAGE TYPE (ALL) 指令，並在 CSQI024I 或 CSQI025I 訊息中記錄 RBA 值 (如需相關資訊，請參閱 CSQI024I 及 CSQI025I)。
2. 取得頁面集的備份副本 (請參閱 第 412 頁的『備份頁面集』)。
3. 發出 ARCHIVE LOG 指令，以確保將最新日誌資料寫出至日誌資料集。 若要從回復點重新啟動，您必須將日誌資料集從 RBA 值所指示的日誌資料集啟動至現行日誌資料集。

備份頁面集

若要回復頁集， IBM MQ 需要知道日誌中要回溯多久。 IBM MQ 會維護每個頁集第 0 頁中的日誌 RBA 號碼，稱為 回復日誌序號 (LSN)。 此數字是日誌中的起始 RBA， IBM MQ 可以從中回復頁集。 當您備份頁集時，也會複製此數字。

如果稍後使用副本來回復頁集，則 IBM MQ 必須具有從此 RBA 值到現行 RBA 的所有日誌記錄的存取權。 這表示您必須保留足夠的日誌記錄，才能讓 IBM MQ 從您想要保留之頁集的最舊備份副本回復。

使用 ADRDSSU COPY 功能來複製頁集。

如需相關資訊，請參閱 COPY DATASET 邏輯資料集的指令語法 文件。

例如：

```
//STEP2 EXEC PGM=ADRDSSU,REGION=6M
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
 COPY -
 DATASET(INCLUDE(SCENDATA.MQPA.PAGESET.*)) -
 RENAMEU(SCENDATA.MQPA.PAGESET.**,SCENDATA.MQPA.BACKUP1.**) -
 SPHERE -
 REPUNC -
 FASTREPLICATION(PREF )-
 CANCELERROR -
 TOL(ENQF)
/*
//
```

如果您在佇列管理程式執行時複製頁集，則必須使用先複製頁集零頁的複製公用程式。 如果您不這麼做，可能會毀損頁集中的資料。

如果動態擴充頁集的處理程序被岔斷 (例如，因為失去系統電源) ，您仍然可以使用 ADRDSSU 來備份頁集。

如果您執行 Access Method Services IDCAMS `LISTCAT ENT('page set data set name') ALLOC`，您會看到 `HI-ALLOC-RBA` 高於 `HI-USED-RBA`。

下次此頁面設定填滿時，它會重新延伸 (如果可能的話) ，並使用高使用 RBA 與最高配置 RBA 之間的頁面，以及另一個新的延伸範圍。

備份物件定義

您也應該備份物件定義的副本。 如果要這麼做，請使用 CSQUTIL COMMAND 函數的 MAKEDEF 特性 (如 發出指令至 IBM MQ (COMMAND) 中所說明)。

每當您取得佇列管理程式的備份副本時，請備份物件定義，並保留最新版本。

回復頁集

如果佇列管理程式因失敗而終止，則通常可以重新啟動佇列管理程式，並在重新啟動期間執行所有回復。 不過，如果沒有任何頁集或日誌資料集可用，則無法進行這類回復。 您現在可以回復的範圍取決於頁集及日誌資料集備份副本的可用性。

若要從回復點重新啟動，您必須具有：

- 要回復之頁集的備份副本。
- 如果您已使用 第 412 頁的『方法 2: 模糊備份』中說明的 "模糊" 備份處理程序，則包含記錄 RBA 值的日誌資料集、ARCHIVE LOG 指令所建立的日誌資料集，以及這些日誌資料集之間的所有日誌資料集。
- 如果您已使用完整備份，但沒有遵循 ARCHIVE LOG 指令所建立的日誌資料集，則 **不** 需要對所有頁集執行 CSQUTIL 公用程式的 FORMAT TYPE (REPLACE) 函數。

若要將頁集回復至其現行狀態，您還必須具有自 ARCHIVE LOG 指令以來的所有日誌資料集及記錄。

有兩種方法可回復頁集。 若要使用任一方法，必須停止佇列管理程式。

簡式回復

這是較簡單的方法，適用於大部分回復狀況。

1. 刪除您要從備份還原的頁集。
2. 請使用 ADRDSSU COPY 功能，從備份副本回復頁集。

   或者，您可以將備份副本重新命名為原始名稱，或變更佇列管理程式程序中的 CSQP00xx DD 陳述式，以指向備份頁集。 不過，如果您隨後遺失或毀損頁面集，則將不再具有要從中還原的備份副本。
3. 重新啟動佇列管理程式。
4. 當佇列管理程式順利重新啟動時，您可以重新啟動應用程式
5. 恢復已還原頁面的一般備份程序。

進階回復

如果您要回復大型頁集，或自前次備份以來頁集上有許多活動，則此方法會提供效能優點。 不過，它需要比簡式方法更多的人為介入，這可能會增加錯誤的風險，以及執行回復所花費的時間。

1. 刪除並重新定義您要從備份還原的頁集。
2. 使用 ADRDSSU ，將頁集的備份副本複製到新的頁集。 使用次要範圍值來定義新的頁面集，以便可以動態展開它。

   或者，您可以將備份副本重新命名為原始名稱，或變更佇列管理程式程序中的 CSQP00xx DD 陳述式，以指向備份頁集。 不過，如果您隨後遺失或毀損頁面集，則將不再具有要從中還原的備份副本。
3. 變更佇列管理程式的 CSQINP1 定義，使與要回復之頁集相關聯的緩衝池儘可能大。 藉由使緩衝池變得很大，您可以將所有變更的頁面保留在緩衝池中，並減少頁集的 I/O 量。
4. 重新啟動佇列管理程式。
5. 當佇列管理程式順利重新啟動時，請停止它 (使用靜止) ，然後使用該頁集的一般緩衝池定義來重新啟動它。 在第二次重新啟動順利完成之後，您可以重新啟動應用程式
6. 恢復已還原頁面的一般備份程序。

重新啟動佇列管理程式時發生的情況

當佇列管理程式重新啟動時，它會將所有變更套用至日誌中登錄的頁集，從頁集的重新啟動點開始。 IBM MQ 可以用這種方式回復多個頁集。 必要的話，會在媒體回復期間動態展開頁集。

在重新啟動期間， IBM MQ 會採用下列最低值來決定要從中啟動的日誌 RBA:

- 從每一個頁集的檢查點日誌記錄回復 LSN。
- 從每個頁集中的第零頁回復 LSN。
- 取得備份時系統中最舊的不完整回復單元的 RBA。

所有物件定義都儲存在頁集零上。 訊息可以儲存在任何可用的頁集中。

註: 如果頁集零無法使用，則佇列管理程式無法重新啟動。

如何刪除頁面集

您可以使用 DELETE PSID 指令來刪除頁集; 如需此指令的詳細資料，請參閱 DELETE PSID 。

您無法刪除仍由任何儲存類別參照的頁集。 請使用 DISPLAY STGCLASS 來找出哪些儲存類別參照頁集。

資料集已從 IBM MQ 取消配置，但未刪除。 它仍可供未來使用，或者可以使用 z/OS 機能來刪除。

從佇列管理程式的已啟動作業程序中移除頁集。

從 CSQINP1 起始設定資料集中移除頁集的定義。

z/OS 如何使用 CSQUTIL 備份及還原佇列

如需使用 CSQUTIL 進行備份及還原的進一步相關資訊，請使用本主題作為參照。

您可以使用 CSQUTIL 公用程式函數來備份及還原佇列。 若要備份佇列，請使用 COPY 或 SCOPY 函數將佇列中的訊息複製到資料集。 若要還原佇列，請使用補充函數 LOAD 或 SLOAD。 如需相關資訊，請參閱 IBM MQ 公用程式。

z/OS 管理緩衝池

如果您要變更或刪除緩衝池，請使用本主題。

本主題說明如何變更及刪除緩衝池。 它包含下列區段:

- 第 415 頁的『如何變更緩衝池中的緩衝區數目』
- 第 415 頁的『如何刪除緩衝池』

在佇列管理程式起始設定期間，會使用從起始設定輸入資料集 CSQINP1 發出的 DEFINE BUFFPOOL 指令來定義緩衝池。 在佇列管理程式執行時，可以使用本主題中詳述的程序來變更其屬性，以回應商業需求。 佇列管理程式會在檢查點日誌記錄中記錄現行緩衝池屬性。 除非 CSQINP1 中的緩衝池定義包括 REPLACE 屬性，否則這些會在後續佇列管理程式重新啟動時自動還原。

使用 DISPLAY USAGE 指令來顯示現行緩衝區屬性。

您也可以搭配使用 DEFINE PSID 指令與 DSN 選項，來動態定義緩衝池。

如果您動態變更緩衝池，則也應該在起始設定資料集 CSQINP1 中更新其定義。

如需頁集、儲存類別、緩衝區及緩衝池的說明，以及適用的部分效能考量，請參閱 在 z/OS 上規劃 。

註: 緩衝池使用大量儲存體。 當您增加緩衝池大小或定義新的緩衝池時，請確保有足夠的儲存體可用。 如需相關資訊，請參閱 位址空間儲存體。

如何變更緩衝池中的緩衝區數目

如果緩衝池太小，此狀況可能會導致主控台上出現訊息 CSQP020E ，您可以使用 ALTER BUFFPOOL 指令配置更多緩衝區給它，如下所示:

1. 查看日誌中的 CSQY220I 訊息，以判斷新緩衝區可用的空間量。 以 MB 為單位報告可用空間。 因為緩衝區大小為 4 KB ，所以每 MB 可用空間可讓您配置 256 個緩衝區。 請勿將所有可用空間配置給緩衝區，因為其他作業需要一些可用空間。

   如果緩衝池使用固定 4 KB 頁面 (亦即，其 PAGECLAS 屬性為 FIXED4KB) ，請確定 LPAR 上有足夠的實際儲存體可用。
2. 如果報告的可用空間不足，請使用指令從另一個緩衝池釋放部分緩衝區

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   其中 *buf-pool-id* 是您要從中收回空間的緩衝池，而 *integer* 是要配置給此緩衝池的新緩衝區數目，它必須小於配置給它的原始緩衝區數目。
3. 使用指令將緩衝區新增至您要擴充的緩衝池

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   其中 *buf-pool-id* 是要擴充的緩衝池，而 *integer* 是要配置給此緩衝池的新緩衝區數目，它必須大於配置給它的原始緩衝區數目。

如何刪除緩衝池

當任何頁集不再使用緩衝池時，請刪除它，以釋放配置給它的虛擬儲存體。

您可以使用 DELETE BUFFPOOL 指令來刪除緩衝池。 如果任何頁集正在使用此緩衝池，則指令會失敗。

如需如何刪除頁集的相關資訊，請參閱 第 414 頁的『如何刪除頁面集』 。

z/OS 在 z/OS 上管理佇列共用群組及共用佇列

IBM MQ 可以使用不同類型的共用資源，例如佇列共用群組、共用佇列及連結機能。 請利用這個主題來檢閱管理這些共用資源所需的程序。

本節包含下列主題的相關資訊:

- 第 416 頁的『管理佇列共用群組』
- 第 418 頁的『管理共用佇列』
- 第 422 頁的『管理群組物件』
- 第 422 頁的『管理連結機能』

z/OS 管理佇列共用群組

您可以在佇列共用群組 (QSG) 中新增或移除佇列管理程式，以及管理相關聯的 Db2 表格。

本主題包含下列作業的相關章節:

- 第 416 頁的『設定佇列共用群組』
- 第 416 頁的『將佇列管理程式新增至佇列共用群組』
- 第 417 頁的『從佇列共用群組中移除佇列管理程式』
- 第 418 頁的『從 Db2 表格中移除佇列共用群組』
- 第 418 頁的『驗證 Db2 定義的一致性』

設定佇列共用群組

每一個佇列共用群組都有最多四個字元的名稱。 該名稱在您的網路中必須是唯一的，且必須不同於任何佇列管理程式名稱。

請遵循下列步驟來設定佇列共用群組:

1. 如果這是第一個使用 Db2 資料共用群組的佇列共用群組，請 設定 Db2 環境。
2. 設定連結機能。
3. 將佇列共用群組新增至 Db2 表格。 使用佇列共用群組公用程式 (CSQ5PQSG) 的 ADD QSG 功能。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45AQS) 中提供了範例。
4. 遵循 第 416 頁的『將佇列管理程式新增至佇列共用群組』 中的步驟，將佇列管理程式新增至佇列共用群組
5. 遵循 第 423 頁的『新增連結機能結構』中的步驟，將應用程式結構定義至 IBM MQ 。
6. 必要的話， 將非共用佇列移轉至共用佇列。
7. 為了可用性，請建立進出佇列共用群組的共用通道。
   - 對於佇列共用群組的連線:
     - 設定 VIPA Socket 或硬體路由器，以在 QSG 中可用的佇列管理程式之間配送工作量。
     - 使用 QSGDISP (GROUP) 定義接收端通道，以確保通道定義可在 QSG 中的所有佇列管理程式上使用。
     - 在每一個佇列管理程式上，啟動具有 INDISP (GROUP) 的接聽器，以進行 QSG 的 MCA 通道連線。與 QSG 的用戶端連線仍應該連接至以 INDISP (QMGR) 啟動的接聽器。
     - 變更應用程式以使用 QSG 名稱而非特定佇列管理程式名稱來連接。
     - 請確定 QSG 中所有佇列管理程式上的通道鑑別規則都相同，以容許應用程式連接至 QSG 中的任何佇列管理程式。
   - 對於佇列共用群組之外的連線:
     - 定義共用傳輸佇列。
     - 使用 QSGDISP (GROUP) 及 DEFCDISP (SHARED) 來定義出埠通道。

   如果您將現有通道轉換為共用通道，則可能需要在啟動通道之前發出 RESET CHANNEL 指令，因為通道使用的同步化佇列將會變更。

將佇列管理程式新增至佇列共用群組

佇列管理程式可以新增至現有的佇列共用群組。

請注意:

- 佇列共用群組必須存在，您才能將佇列管理程式新增至其中。
- 佇列管理程式只能是一個佇列共用群組的成員。

請遵循下列步驟，將佇列管理程式新增至佇列共用群組:

1. 執行 實作佇列共用群組的 ESM 安全控制項 中的作業，以授與對佇列管理程式及通道起始程式使用者 ID 的適當存取權。
2. 如果佇列共用群組具有配置為將資料卸載至 SMDS 的 CF 結構，請在 設定 SMDS 環境中執行作業。
3. 停止佇列管理程式。
4. 使用佇列共用群組公用程式 (CSQ5PQSG) 的 ADD QMGR 功能。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45AQM) 中提供範例。
5. 變更系統參數模組 ，以新增佇列共用群組資料:
   a. 修改 CSQ6SYSP 以指定 QSGDATA 參數。 如需相關資訊，請參閱 使用 CSQ6SYSP 。
   b. 組合並鏈結系統參數模組。 您可能想要對載入模組使用不同的名稱。
   c. 將啟動程序變更為使用新模組。
6. 複製並修改範例成員 thlqual.SCSQPROC(CSQ4INSS) ，其定義必要的 CF 結構及 SYSTEM 佇列。 將自訂成員新增至佇列管理程式啟動 JCL 中的 CSQINP2 DD。
7. 使用佇列共用群組系統參數模組重新啟動佇列管理程式。
8. 選擇性地移轉至以佇列共用群組名稱作為字首的安全設定檔，而不是佇列管理程式名稱。
9. 如果共用通道用於與 QSG 的連線，請建立通道鑑別規則來鏡映 QSG 中其他佇列管理程式上的那些通道鑑別規則，以容許應用程式連接至 QSG 中的任何佇列管理程式。
10. 10. 選擇性地執行下列任一項，以容許連接至 QSG 中佇列管理程式的應用程式將訊息放置至 QSG 中其他佇列管理程式所管理的佇列:
    - 發出指令 ALTER QMGR IGQ (ENABLED) ，以開啟 內部群組佇列作業 。
    - 定義傳輸佇列及通道至 QSG 中的其他佇列管理程式。 定義與目標佇列管理程式同名的傳輸佇列，可避免需要定義遠端佇列及佇列管理程式別名。

註: 若要將佇列管理程式新增至包含執行舊版 IBM MQ 之佇列管理程式的現有佇列共用群組，您必須先將群組中 IBM MQ 最高版本的共存性 PTF 套用至群組中每一個舊版佇列管理程式。

從佇列共用群組中移除佇列管理程式

如果另一個處理程序不需要佇列管理程式的日誌，且佇列管理程式所擁有的所有 SMDS 都是空的，則您只能從佇列共用群組中移除佇列管理程式。

如需相關資訊，請參閱 刪除共用訊息資料集 及 DELETE CFSTRUCT 。

如果日誌包含下列項目，則需要這些日誌:

- 佇列共用群組所使用的其中一個連結機能 (CF) 應用程式結構的最新備份
- 未來還原處理程序所需的資料，亦即，自前次備份排除間隔值所說明的時間以來，佇列管理程式已使用可回復的結構。

如果其中一個或兩個點適用，或佇列管理程式所擁有的 SMDS 包含訊息，則無法移除佇列管理程式。 若要判斷未來還原處理程序所需的佇列管理程式日誌，請搭配使用 MQSC DISPLAY CFSTATUS 指令與 TYPE (BACKUP) 選項 (如需此指令的詳細資料，請參閱 DISPLAY CFSTATUS)。

請使用下列步驟，從佇列共用群組中移除佇列管理程式:

1. 停止連接至將訊息放置到共用佇列之佇列管理程式的任何應用程式。
2. 解決涉及此佇列管理程式的任何不確定工作單元。
3. 發出指令 DISPLAY USAGE TYPE (SMDS) ，以判斷佇列管理程式擁有的任何 SMDS 中是否有任何訊息。
4. 如果有任何應用程式結構的卸載訊息，請等待從佇列中擷取那些訊息。 在繼續之前， DISPLAY USAGE TYPE (SMDS) 所報告的卸載訊息數應該為零。
5. 使用 STOP QMGR MODE (QUIESCE) 完全關閉佇列管理程式。
6. 等待間隔至少等於您將在下一個步驟中於 BACKUP CFSTRUCT 指令中指定的 EXCLINT 參數值。

7. 在另一個佇列管理程式上，使用 MQSC BACKUP CFSTRUCT 指令並指定前一個步驟中所需的 EXCLINT 值，對每一個可回復 CF 結構執行 CF 結構備份。
8. 請檢查 DISPLAY CFSTATUS (*) TYPE (BACKUP) 指令的輸出，以確認不需要佇列管理程式的日誌來還原任何 CF 結構。
9. 使用 CSQ5PQSG 公用程式的 REMOVE QMGR 功能，可以從佇列共用群組中移除佇列管理程式。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45RQM) 中提供範例。
10. 在重新啟動佇列管理程式之前，請將 QSGDATA 系統參數重設為其預設值，並重建系統參數模組。 如需如何自訂系統參數的相關資訊，請參閱 使用 CSQ6SYSP 。

請注意，當移除佇列共用群組中的最後一個佇列管理程式時，您必須使用 FORCE 選項，而不是 REMOVE。 這會從佇列共用群組中移除佇列管理程式，但不會執行回復所需的佇列管理程式日誌一致性檢查。 只有在刪除佇列共用群組時，才應該執行這項作業。

從 Db2 表格中移除佇列共用群組

若要從 Db2 表格中移除佇列共用群組，請使用佇列共用群組公用程式 (CSQ5PQSG) 的 REMOVE QSG 功能。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45RQS) 中提供範例。

從佇列共用群組中移除所有佇列管理程式之後，您只能從一般 Db2 資料共用群組表格中移除佇列共用群組 (如 第 417 頁的『從佇列共用群組中移除佇列管理程式』 中所述)。

從佇列共用群組管理表格中刪除佇列共用群組記錄時，與該佇列共用群組相關的所有物件及管理資訊都會從其他 IBM MQ Db2 表格中刪除。 這包括共用佇列及群組物件資訊。

驗證 Db2 定義的一致性

如果 Db2 物件定義因任何原因而不一致，則佇列共用群組內的共用佇列可能會發生問題。

若要驗證佇列管理程式、CF 結構及共用佇列的 Db2 物件定義一致性，請使用佇列共用群組公用程式 (CSQ5PQSG) 的 Verify QSG 函數。 此程式在 佇列共用群組公用程式中有說明。

z/OS 管理共用佇列

請利用這個主題來瞭解如何回復、移動及移轉共用佇列。

本節說明下列作業:

- 第 418 頁的『回復共用佇列』
- 第 419 頁的『移動共用佇列』
- 第 421 頁的『將非共用佇列移轉至共用佇列』
- 暫停 Db2 連線

回復共用佇列

在下列情況下， IBM MQ 可以回復共用佇列上的持續訊息:

- 已執行包含訊息之 CF 結構的備份。
- 佇列共用群組中所有佇列管理程式的所有日誌都可用，以從執行備份時開始執行回復。
- Db2 可用，且結構備份表格比最新 CF 結構備份還要新。

共用佇列上的訊息儲存在連結機能 (CF) 結構中。 持續訊息可以放置在共用佇列上，如同非共用佇列上的持續訊息，它們會複製到佇列管理程式日誌。 提供 MQSC 備份 CFSTRUCT 及 回復 CFSTRUCT 指令，以容許在不太可能的連結機能失敗事件中回復 CF 結構。 在這種情況下，儲存在受影響結構中的任何非持續訊息都會遺失，但持續訊息可以回復。 在回復結構之前，會防止使用該結構的任何進一步應用程式活動。

若要啟用回復，您必須經常使用 MQSC BACKUP CFSTRUCT 指令來備份連結機能清單結構。 CF 結構中的訊息會寫入建立備份之佇列管理程式的作用中日誌資料集。 它會將備份的記錄寫入 Db2: 所備份 CF 結構的名

稱、執行備份的佇列管理程式名稱、該佇列管理程式日誌上此備份的 RBA 範圍，以及備份時間。 備份 CF 清單結構，即使您不主動使用共用佇列 (例如，如果您已設定要在未來使用它的佇列共用群組)。

您可以對可執行回復的佇列管理程式發出 MQSC RECOVER CFSTRUCT 指令來回復 CF 結構; 您可以使用佇列共用群組中的任何佇列管理程式。 您可以指定要回復的單一 CF 結構，也可以同時回復數個 CF 結構。

如前所述，請務必經常備份 CF 清單結構，否則回復 CF 結構可能需要很長的時間。 此外，無法取消回復處理程序。

共用佇列的定義會保留在 Db2 資料庫中，因此可以在必要時使用標準 Db2 資料庫程序來回復。 如需相關資訊，請參閱 共用佇列及佇列共用群組 。

移動共用佇列

本節說明如何透過將共用佇列從一個連結機能結構移至另一個連結機能結構來執行負載平衡。 它也說明如何將非共用佇列移至共用佇列，以及如何將共用佇列移至非共用佇列。

當您移動佇列時，需要在程序中定義暫時佇列。 這是因為每一個佇列都必須有唯一名稱，因此您不能有兩個同名的佇列，即使佇列有不同的佇列處置。 IBM MQ 容許有兩個同名 (在步驟 第 419 頁的『2』 中) 的佇列，但您無法使用這些佇列。

- 將佇列從一個連結機能結構移至另一個連結機能結構
- 將非共用佇列移至共用佇列
- 將共用佇列移至非共用佇列

將佇列從一個連結機能結構移至另一個連結機能結構

若要將佇列及其訊息從一個 CF 結構移至另一個 CF 結構，請使用 MQSC MOVE QLOCAL 指令。 當您已識別要移至新 CF 結構的一或多個佇列時，請使用下列程序來移動每一個佇列:

1. 請確定您要移動的佇列未被任何應用程式使用，亦即，佇列共用群組中所有佇列管理程式上的佇列屬性 IPPROCS 及 OPPROCS 都是零。
2. 透過變更佇列定義以停用 MQPUT ，防止應用程式將訊息放置在要移動的佇列上。 將佇列定義變更為 PUT (DISABLED)。
3. 使用下列指令，定義與正在移動的佇列具有相同屬性的暫時佇列:

   ```
   DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
   ```

   註: 如果此暫時佇列存在先前的執行，請在執行定義之前先刪除它。
4. 使用下列指令，將訊息移至暫時佇列:

   ```
   MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
   ```
5. 使用下列指令來刪除您要移動的佇列:

   ```
   DELETE QLOCAL(QUEUE_TO_MOVE)
   ```
6. 使用下列指令，重新定義要移動的佇列，變更 CFSTRUCT 屬性:

   ```
   DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
   ```

   當重新定義佇列時，它是以步驟 第 419 頁的『3』 中所建立的暫時佇列為基礎。
7. 使用下列指令將訊息移回新佇列:

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

8. 不再需要在步驟 第 419 頁的『3』 中建立的佇列。 請使用下列指令來刪除它:

```
DELETE QL(TEMP_QUEUE)
```

9. 如果要移動的佇列定義在 CSQINP2 資料集中，請在 CSQINP2 資料集中變更適當 DEFINE QLOCAL 指令的 CFSTRUCT 屬性。 新增 REPLACE 關鍵字，以取代現有的佇列定義。

第 420 頁的圖 38 顯示將佇列從一個 CF 結構移至另一個 CF 結構的範例工作。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

圖 *38:* 將佇列從一個 *CF* 結構移至另一個 *CF* 結構的範例工作

將非共用佇列移至共用佇列

將非共用佇列移至共用佇列的程序與將佇列從一個 CF 結構移至另一個 CF 結構的程序類似 (請參閱 第 419 頁的『將佇列從一個連結機能結構移至另一個連結機能結構』)。 第 420 頁的圖 39 提供範例工作來執行此動作。

註: 請記住，共用佇列上的訊息會受到訊息大小上限、訊息持續性及佇列索引類型的特定限制，因此您可能無法將部分非共用佇列移至共用佇列。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

圖 *39:* 將非共用佇列移至共用佇列的範例工作

將共用佇列移至非共用佇列

將共用佇列移至非共用佇列的程序與將佇列從一個 CF 結構移至另一個 CF 結構的程序類似 (請參閱 第 419 頁的『將佇列從一個連結機能結構移至另一個連結機能結構』)。

第 421 頁的圖 40 提供範例工作來執行此動作。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW) QSGDISP(QMGR)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

圖 *40:* 將共用佇列移至非共用佇列的範例工作

將非共用佇列移轉至共用佇列

將非共用佇列移轉至共用佇列有兩個階段:

- 移轉佇列共用群組中的第一個 (或唯一) 佇列管理程式
- 移轉佇列共用群組中的任何其他佇列管理程式

移轉佇列共用群組中的第一個 (或唯一) 佇列管理程式

第 420 頁的圖 39 顯示將非共用佇列移至共用佇列的範例工作。 針對每一個需要移轉的佇列執行此動作。

註:

1. 共用佇列上的訊息受限於訊息大小上限、訊息持續性及佇列索引類型的特定限制，因此您可能無法將部分非共用佇列移至共用佇列。
2. 您必須對共用佇列使用正確的索引類型。 如果您將傳輸佇列移轉為共用佇列，則索引類型必須是 MSGID。

如果佇列是空的，或您不需要保留其中的訊息，則移轉佇列會更簡單。 第 421 頁的圖 41 顯示在這些情況下要使用的範例工作。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
DELETE QL(TEMP_QUEUE)
/*
```

圖 *41:* 將無訊息的非共用佇列移至共用佇列的範例工作

移轉佇列共用群組中的任何其他佇列管理程式

1. 對於與現有共用佇列沒有相同名稱的每一個佇列，請依照 第 420 頁的圖 39 或 第 421 頁的圖 41 中的說明來移動佇列。
2. 對於與現有共用佇列同名的佇列，請使用 第 422 頁的圖 42 中所示的指令將訊息移至共用佇列。

```
MOVE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR)
```

圖 *42:* 將訊息從非共用佇列移至現有的共用佇列

暫停與 Db2 的連線

如果您想要將維護或服務套用至與共用佇列相關的 Db2 表格或套件，而不停止佇列管理程式，則必須暫時中斷資料共用群組 (DSG) 中佇列管理程式與 Db2 的連線。

若要執行此作業：

1. 使用 MQSC 指令 SUSPEND QMGR FACILITY (Db2)。
2. 進行連結。
3. 使用 MQSC 指令 RESUME QMGR FACILITY (Db2) 重新連接 Db2

請注意，使用這些指令有一些限制。

小心: 當 Db2 連線暫停時，下列作業將無法使用。 因此，在企業最不忙碌的時期，您需要執行這項工作。

- 存取共用佇列物件以進行管理 (定義、刪除、變更)
- 啟動共用通道
- 將訊息儲存在 Db2 中
- 備份或回復 CFSTRUCT

z/OS 管理群組物件

請利用這個主題來瞭解如何使用群組物件。

IBM MQ 會自動將群組物件的定義複製到使用它的每一個佇列管理程式的頁集零。 您可以暫時變更定義的副本， IBM MQ 可讓您重新整理儲存庫副本中的頁面集副本。 IBM MQ 一律在啟動時嘗試從儲存庫副本重新整理頁面集副本 (對於通道物件，這是在通道起始程式重新啟動時完成)。 這可確保頁面集副本反映儲存庫上的版本，包括佇列管理程式非作用中時所做的任何變更。

在某些情況下，不會執行重新整理，例如:

- 如果開啟佇列副本，則會變更佇列使用情形的重新整理會失敗。
- 如果佇列副本中有訊息，則會刪除該佇列的重新整理會失敗。

在這些情況下，不會對該副本上執行重新整理，而是對所有其他佇列管理程式上的副本上執行重新整理。 在新增、變更或刪除群組物件之後，以及在佇列管理程式或通道起始程式重新啟動時，檢查並更正副本物件的任何問題。

z/OS 管理連結機能

請利用這個主題來瞭解如何新增或移除連結機能 (CF) 結構。

本節說明下列作業:

- 第 423 頁的『新增連結機能結構』
- 第 423 頁的『移除連結機能結構』

新增連結機能結構

若要新增連結機能結構，請使用下列程序:

1. 在 CFRM 原則資料集中定義 CF 結構。 設定連結機能 中設定連結機能的相關資訊說明命名連結機能結構的規則，以及如何在 CFRM 原則資料集中定義結構。
2. 如果您要配置結構以將訊息資料卸載至 SMDS，請配置並預先格式化資料集。 如需詳細資料，請參閱 建立共用訊息資料集 。
3. 使用 DEFINE CFSTRUCT 指令定義 IBM MQ 的結構。

移除連結機能結構

若要移除連結機能結構，請使用下列程序:

1. 使用下列指令，以取得使用您要刪除之連結機能結構的所有佇列清單:

```
DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)
```

2. 刪除所有使用該結構的佇列。
3. 使用 DELETE CFSTRUCT 指令從 IBM MQ 刪除 CF 結構。
4. 如果結構已配置為將訊息資料卸載至 SMDS，請刪除 SMDS。
5. 從 CFRM 原則資料集中移除結構定義，然後執行 IXCMIAPU 公用程式。 (這與設定連結機能的自訂作業相反，如 設定連結機能中所述。)

z/OS 調整連結機能清單監視

請利用這個主題來瞭解連結機能清單監視

連結機能 (CF) 清單監視是用來監視包含 IBM MQ 共用佇列之清單結構的狀態。 當訊息新增至共用佇列，且佇列的深度從零轉移至非零時， CF 會通知佇列共用群組中的所有佇列管理程式。 當收到通知時，佇列管理程式可能會執行一些動作，包括通知使用 TRIGGER (FIRST) 的觸發監視器，或執行 get-wait 的應用程式。

依預設， CF 會同時通知佇列共用群組中的所有佇列管理程式。 在某些配置中，這可能會造成問題，例如:

- 扭曲工作量配送，其中大量訊息會送往佇列共用群組中的特定佇列管理程式，通常是在最快 LPAR 上執行的佇列管理程式，或最接近 CF 的佇列管理程式，或
- 大量失敗的取得數，導致浪費 CPU 時間。

z/OS V2R3 引進稱為 **KEYRNOTIFYDELAY** 的新連結機能資源管理程式 (CFRM) 屬性，可與包含共用佇列 (亦即，應用程式結構，而非管理結構) 的清單結構搭配使用，並可針對特定工作量，將工作量偏斜及空白 MQGET 呼叫或空白 MQGET 呼叫的影響降至最低。

KEYRNOTIFYDELAY 只能在 CF 中以 CFLEVEL 22 或更高版本執行的結構上設定。

其值必須是 1 到 7 個十進位數，範圍從 0 到 1,000,000 微秒。 如果設為非零值，且佇列深度從零轉移至非零，則 CF 會從佇列共用群組中選取單一佇列管理程式，並在群組中所有其他佇列管理程式之前通知該佇列管理程式。

以循環式方式選取佇列管理程式。 如果選取的佇列管理程式未在 **KEYRNOTIFYDELAY** 所說明的時間間隔內處理訊息，也會通知佇列共用群組中的所有其他佇列管理程式。

如需 **KEYRNOTIFYDELAY** 的相關資訊，請參閱: 瞭解金鑰範圍監視通知延遲。

請注意，有兩個類似的 CFRM 屬性稱為 **LISTNOTIFYDELAY** 和 **SUBNOTIFYDELAY**。 這兩者都對 IBM MQ 工作量沒有任何可測量的影響。

z/OS 在 z/OS 上回復並重新啟動

請利用這個主題來瞭解 IBM MQ 所使用的回復和重新啟動機制。

z/OS 正在重新啟動 IBM MQ

在佇列管理程式終止之後，視佇列管理程式終止的方式而定，需要不同的重新啟動程序。 請利用這個主題來瞭解您可以使用的不同重新啟動程序。

本主題包含如何在下列情況下重新啟動佇列管理程式的相關資訊:

- 第 424 頁的『在正常關機之後重新啟動』
- 第 424 頁的『在異常終止之後重新啟動』
- 第 424 頁的『如果您遺失頁集，請重新啟動』
- 第 424 頁的『如果您遺失日誌資料集，則會重新啟動』
- 如果您遺失 CF 結構，則重新啟動

在正常關機之後重新啟動

如果已使用 STOP QMGR 指令停止佇列管理程式，則系統會依序完成其工作，並在停止之前取得終止檢查點。 當您重新啟動佇列管理程式時，它會使用來自系統檢查點及回復日誌的資訊來判定關機時的系統狀態。

若要重新啟動佇列管理程式，請發出 START QMGR 指令，如 第 365 頁的『在 z/OS 上啟動及停止佇列管理程式』中所述。

在異常終止之後重新啟動

IBM MQ 會自動偵測在正常關機或異常終止之後重新啟動。

在異常終止之後啟動佇列管理程式不同於在發出 STOP QMGR 指令之後啟動佇列管理程式。 如果佇列管理程式異常終止，則會在無法完成其工作或取得終止檢查點的情況下終止。

若要重新啟動佇列管理程式，請發出 START QMGR 指令，如 第 365 頁的『在 z/OS 上啟動及停止佇列管理程式』中所述。 當您在異常終止之後重新啟動佇列管理程式時，它會使用日誌中的資訊重新整理其在終止時狀態的知識，並通知您各種作業的狀態。

通常，重新啟動程序會解決所有不一致狀態。 但是，在某些情況下，您必須採取特定步驟來解決不一致。 這說明於第 435 頁的『手動回復工作單元』。

如果您遺失頁集，請重新啟動

如果您遺失頁集，則需要先從備份副本還原它們，然後才能重新啟動佇列管理程式。 這說明於第 411 頁的『如何備份及回復頁集』。

在這些情況下，由於媒體回復所需的時間長度，佇列管理程式可能需要很長的時間來重新啟動。

如果您遺失日誌資料集，則會重新啟動

在停止佇列管理程式 (使用 STOP QMGR 指令) 之後，如果日誌的兩個副本都遺失或損壞，您可以重新啟動佇列管理程式，但前提是您有一組一致的頁集 (使用 方法 1: 完整備份 產生)。

請遵循下列程序:

1. 定義新的頁面集，以對應佇列管理程式中的每一個現有頁面集。 如需頁集定義的相關資訊，請參閱 作業 15: 定義頁集 。

   請確定每一個新頁集都大於對應的來源頁集。
2. 使用 CSQUTIL 的 FORMAT 函數來格式化目的地頁集。 如需詳細資料，請參閱 格式化頁面集 。
3. 使用 CSQUTIL 的 RESETPAGE 函數來複製現有頁集或就地重設頁集，並重設每一頁中的日誌 RBA。 如需此功能的相關資訊，請參閱 複製頁集並重設日誌 。

4. 使用 CSQJU003 重新定義佇列管理程式日誌資料集及 BSDS (請參閱 變更日誌庫存公用程式)。
5. 使用新的頁集來重新啟動佇列管理程式。 若要執行此動作，請執行下列其中一項:
   - 變更佇列管理程式啟動型作業程序，以參照新的頁集。 如需相關資訊，請參閱 作業 6: IBM MQ 佇列管理程式的建立程序 。
   - 使用「存取方法服務」來刪除舊頁集，然後重新命名新頁集，讓它們與舊頁集同名。

請注意: 在刪除任何 IBM MQ 頁集之前，請確定您已建立必要的備份副本。

如果佇列管理程式是佇列共用群組的成員，則遺失或損壞的日誌通常不會影響 GROUP 及 SHARED 物件定義。 不過，如果遺失或損壞日誌所涵蓋的工作單元中涉及任何共用佇列訊息，則無法預期對這類未確定的訊息的影響。

註: 如果日誌已損壞，且佇列管理程式是佇列共用群組的成員，則回復共用持續訊息的能力可能會遺失。 針對具有 RECOVER (YES) 屬性的所有 CF 結構，在佇列共用群組中的另一個作用中佇列管理程式上立即發出 BACKUP CFSTRUCT 指令。

如果您已遺失 CF 結構，則會重新啟動

如果您遺失 CF 結構，則不需要重新啟動，因為佇列管理程式不會終止。

z/OS *z/OS* 上的替代站台回復

您可以回復單一佇列管理程式或佇列共用群組，或考量磁碟鏡映。

如需詳細資料，請參閱下列各節:

- 在替代站台回復單一佇列管理程式
- 回復佇列共用群組。
  - CF 結構媒體回復
  - 備份主要站台上的佇列共用群組
  - 在替代站台回復佇列共用群組
- 使用磁碟鏡映

回復替代站台上的單一佇列管理程式

如果完全失去 IBM MQ 運算中心，您可以在回復站台的另一個佇列管理程式或佇列共用群組上回復。 (如需佇列共用群組的替代站台回復程序，請參閱 第 428 頁的『在替代站台回復佇列共用群組』 。)

若要在回復站台的另一個佇列管理程式上回復，您必須定期備份頁集及日誌。 與所有資料回復作業一樣，災難回復的目標是盡可能減少資料、工作量處理 (更新) 及時間。

在回復站台:

- 回復佇列管理程式 **必須** 具有與遺失佇列管理程式相同的名稱。
- 每個回復佇列管理程式上使用的系統參數模組 (例如， CSQZPARM) 必須包含與對應的遺失佇列管理程式相同的參數。

當您完成此動作時，請重新建立所有佇列管理程式，如下列程序中所述。 這可用來在單一佇列管理程式的回復站台執行災難回復。 它假設所有可用的項目如下:

- 在主要網站上正常執行所建立的保存日誌及 BSDS 副本 (作用中日誌將與主要網站上的佇列管理程式一起遺失)。
- 來自主要網站之佇列管理程式的頁面集副本，其經歷時間與可用的最新保存日誌副本相同或更舊。

您可以對作用中及保存日誌使用雙重記載，在此情況下，您需要將 BSDS 更新項目套用至這兩個副本:

1. 定義新的頁面集資料集，並從主要網站將資料與頁面集副本中的資料一起載入。
2. 定義新的作用中日誌資料集。
3. 定義新的 BSDS 資料集，並使用 Access Method Services REPRO 將最新保存的 BSDS 複製到其中。

4. 使用列印日誌對映公用程式 CSQJU004 來列印來自這個最新 BSDS 的資訊。 在保存此 BSDS 時，您所擁有的最新保存日誌會被截斷為作用中日誌，而不會顯示為保存日誌。 記錄此日誌的 STARTRBA 和 ENDRBA。
5. 使用變更日誌庫存公用程式 CSQJU003，利用步驟 第 426 頁的『4』中記錄的 STARTRBA 和 ENDRBA，在您剛還原的 BSDS 中登錄這個最新的保存日誌資料集。
6. 使用 CSQJU003 的 DELETE 選項，從 BSDS 中移除所有作用中日誌資訊。
7. 使用 CSQJU003 的 NEWLOG 選項將作用中日誌新增至 BSDS，不要指定 STARTRBA 或 ENDRBA。
8. 使用 CSQJU003 將重新啟動控制記錄新增至 BSDS。 指定 `CRESTART CREATE,ENDRBA=highrba`，其中 `highrba` 是可用的最新保存日誌 (在步驟 第 426 頁的『4』 中找到) 的高 RBA，加上 1。

   BSDS 現在說明所有作用中日誌都是空的、您有可用的所有保存日誌，且日誌結尾之後沒有檢查點。
9. 使用 START QMGR 指令重新啟動佇列管理程式。 在起始設定期間，會發出類似下列的操作員回覆訊息:

   ```
   CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
   REPLY Y TO CONTINUE, N TO CANCEL
   ```

   輸入 Y 以啟動佇列管理程式。 佇列管理程式會啟動，並將資料回復至 CRESTART 陳述式中指定的 ENDRBA。

如需使用 CSQJU003 及 CSQJU004 的相關資訊，請參閱 使用 IBM MQ 公用程式 。

下列範例顯示步驟 6、7 及 8 的 CSQJU003 輸入陳述式範例:

```
* Step 6
DELETE DSNAME=MQM2.LOGCOPY1.DS01
DELETE DSNAME=MQM2.LOGCOPY1.DS02
DELETE DSNAME=MQM2.LOGCOPY1.DS03
DELETE DSNAME=MQM2.LOGCOPY1.DS04
DELETE DSNAME=MQM2.LOGCOPY2.DS01
DELETE DSNAME=MQM2.LOGCOPY2.DS02
DELETE DSNAME=MQM2.LOGCOPY2.DS03
DELETE DSNAME=MQM2.LOGCOPY2.DS04

* Step 7
NEWLOG DSNAME=MQM2.LOGCOPY1.DS01,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS02,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS03,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS04,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY2.DS01,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS02,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS03,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS04,COPY2

* Step 8
CRESTART CREATE,ENDRBA=063000
```

在回復網站上重新啟動通道起始程式時需要考量的事項，與使用 ARM 在不同 z/OS 映像檔上重新啟動通道起始程式時所面對的事項類似。 如需相關資訊，請參閱 第 433 頁的『在 IBM MQ 網路中使用 ARM』。 您的回復策略也應該涵蓋回復 IBM MQ 產品程式庫，以及使用 IBM MQ (例如 CICS) 的應用程式設計環境。

變更日誌庫存公用程式 (CSQJU003) 的其他功能也可以在災難回復實務範例中使用。 HIGHRBA 函數容許更新引導資料集內最高 RBA 寫入值及最高 RBA 卸載值。 CHECKPT 功能容許在 BSDS 中新增新的檢查點佇列記錄或刪除現有的檢查點佇列記錄。

請注意: 這些函數可能會影響 IBM MQ 資料的完整性。 僅在 IBM 服務人員的指引下，在災難回復實務範例中使用它們。

快速複製技術

如果在佇列管理程式凍結時建立所有頁面集和日誌的副本，則副本將是一個一致集，可用來在替代站台重新啟動佇列管理程式。 它們通常會啟用更快速的佇列管理程式重新啟動，因為幾乎沒有要執行的媒體回復。

請使用 SUSPEND QMGR LOG 指令來凍結佇列管理程式。 此指令會將緩衝池清除至頁集，取得檢查點，並停止任何進一步的日誌寫入活動。 一旦暫停日誌寫入活動，佇列管理程式實際上會凍結，直到您發出 RESUME QMGR LOG 指令為止。 當佇列管理程式凍結時，可以複製頁面集和日誌。

透過使用複製工具 (例如 FLASHCOPY 或 SNAPSHOT) 來快速複製頁集和日誌，可以將佇列管理程式凍結的時間縮短至最短。

不過，在佇列共用群組內， SUSPEND QMGR LOG 指令可能不是很好的解決方案。 若要生效，日誌副本必須全部包含相同的回復時間點，這表示必須同時在佇列共用群組內的所有佇列管理程式上發出 SUSPEND QMGR LOG 指令，因此整個佇列共用群組將凍結一段時間。

回復佇列共用群組

如果發生主要站台災難，您可以使用主要站台中的備份資料集，在遠端站台上重新啟動佇列共用群組。 若要回復佇列共用群組，您需要協調佇列共用群組中所有佇列管理程式的回復，並與其他資源 (主要是 Db2) 協調。 本節詳細說明這些作業。

- CF 結構媒體回復
- 備份主要站台上的佇列共用群組
- 在替代站台回復佇列共用群組

CF 結構媒體回復

用來在共用佇列上保留持續訊息之 CF 結構的媒體回復，取決於是否有媒體的備份，可透過套用已記載的更新項目來回復。 使用 MQSC BACKUP CFSTRUCT 指令定期備份 CF 結構。 共用佇列 (MQGET 和 MQPUT) 的所有更新項目都會寫入執行更新之佇列管理程式的日誌中。 若要執行 CF 結構的媒體回復，您必須從已使用該 CF 結構之所有佇列管理程式的日誌中，將已記載的更新項目套用至該備份。 當您使用 MQSC RECOVER CFSTRUCT 指令時， IBM MQ 會自動合併相關佇列管理程式中的日誌，並將更新項目套用至最新備份。

CF 結構備份會寫入處理 BACKUP CFSTRUCT 指令之佇列管理程式的日誌中，因此沒有其他要收集並傳輸至替代站台的資料集。

備份主要站台上的佇列共用群組

在主要站台上，您需要定期建立一組一致的備份，可在發生災難時用來在替代站台上重建佇列共用群組。 對於單一佇列管理程式，回復可以是任意時間點，通常是遠端站台可用的日誌結尾。 不過，如果持續訊息已儲存在共用佇列中，則必須合併佇列共用群組中所有佇列管理程式的日誌，才能回復共用佇列，因為佇列共用群組中的任何佇列管理程式可能已在佇列上執行更新 (MQPUT 或 MQGET)。

若要回復佇列共用群組，您需要建立一個在所有佇列管理程式的日誌資料日誌範圍內的復原點。 不過，因為您只能從日誌 **轉遞** 回復媒體，所以這個時間點必須在發出 BACKUP CFSTRUCT 指令之後，以及執行任何頁集備份之後。 (一般而言，回復的時間點可能對應於工作日或週的結束。)

下圖顯示佇列共用群組中兩個佇列管理程式的時間表。 對於每一個佇列管理程式，會建立頁集的模糊備份 (請參閱 方法 2: 模糊備份)。 在佇列管理程式 A 上，發出 BACKUP CFSTRUCT 指令。 隨後，會在每一個佇列管理程式上發出 ARCHIVE LOG 指令來截斷作用中日誌，並從佇列管理程式離線將它複製到媒體，它可以傳輸至替代站台。 日誌結尾會識別發出 ARCHIVE LOG 指令的時間，因此會標示替代站台通常可用的日誌資料範圍。 回復的時間點必須介於任何頁集或 CF 結構備份的結尾，以及替代位置可用的日誌的最早結尾之間。

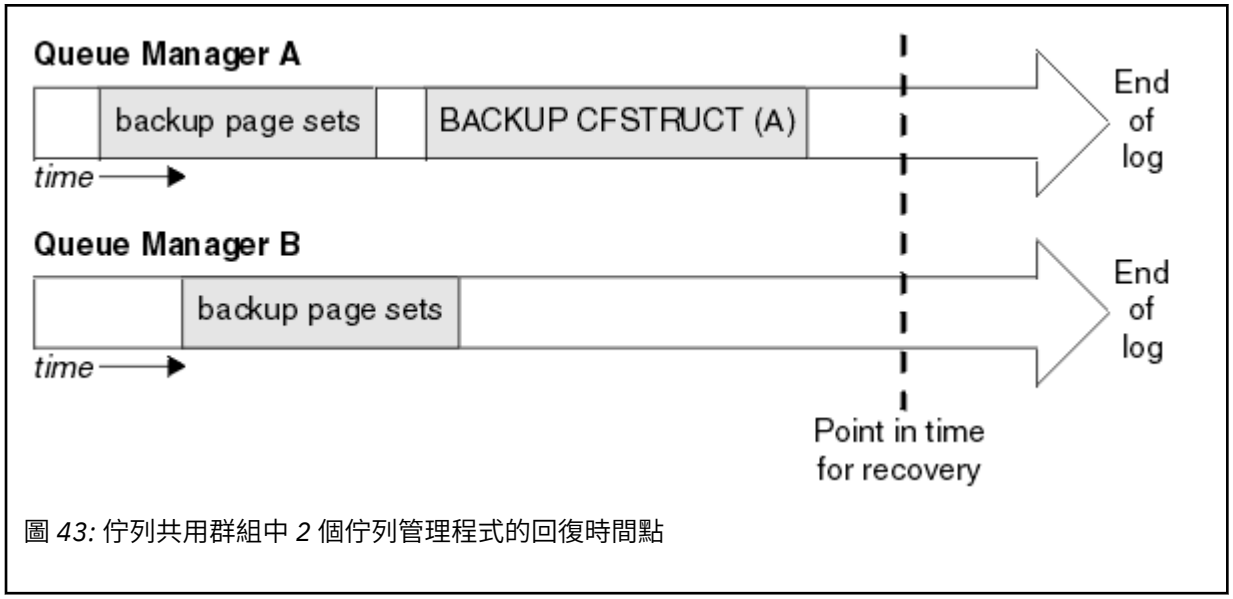


圖 *43:* 佇列共用群組中 *2* 個佇列管理程式的回復時間點

IBM MQ 會在 Db2 的表格中記錄與 CF 結構備份相關聯的資訊。 視您的需求而定，您可能想要將 IBM MQ 回復的復原點與 Db2 的復原點進行協調，或者取得 IBM MQ CSQ.ADMIN_B_STRBACKUP 表格。

若要準備回復，請執行下列動作:

1. 為佇列共用群組中的每一個佇列管理程式建立頁面集備份。
2. 對具有 RECOVER (YES) 屬性的每一個 CF 結構發出 BACKUP CFSTRUCT 指令。 您可以從單一佇列管理程式或佇列共用群組內的不同佇列管理程式發出這些指令，以平衡工作量。
3. 完成所有備份之後，請發出 ARCHIVE LOG 指令來切換作用中日誌，並建立佇列共用群組中每一個佇列管理程式的日誌和 BSDS 副本。
4. 傳輸頁面集備份、保存日誌、佇列共用群組中所有佇列管理程式的保存 BSDS ， 以及您選擇的離站 Db2 備份資訊。

在替代站台回復佇列共用群組

您需要先準備環境，才能回復佇列共用群組:

1. 當您安裝佇列共用群組時，如果您在連結機能中有來自實際啟動的舊資訊，則需要先清除下列項目:

   註: 如果您在連結機能中沒有舊資訊，則可以省略此步驟。

   a. 輸入下列 z/OS 指令，以顯示此佇列共用群組的 CF 結構:

   ```
   D XCF,STRUCTURE,STRNAME= qsgname
   ```

   b. 對於以佇列共用群組名稱開頭的所有結構，請使用 z/OS 指令 SETXCF FORCE CONNECTION 來強制關閉這些結構的連線:

   ```
   SETXCF FORCE,CONNECTION,STRNAME= strname,CONNAME=ALL
   ```

   c. 針對每一個結構使用下列指令，刪除所有 CF 結構:

   ```
   SETXCF FORCE,STRUCTURE,STRNAME= strname
   ```

2. 還原 Db2 系統及資料共用群組。
3. 回復 CSQ.ADMIN_B_STRBACKUP 表格，以便它包含在主要網站上取得的最新結構備份的相關資訊。

   註: STRBACKUP 表格必須包含最新的結構備份資訊。 較舊的結構備份資訊可能需要您因最近的 DISPLAY USAGE TYPE (DATASET) 指令所提供的資訊而捨棄的資料集，這表示已回復的 CF 結構將不會包含正確的資訊。

4. 針對佇列共用群組中的每個佇列管理程式，執行 CSQ5PQSG 公用程式的 ADD QMGR 指令。 這會還原每一個佇列管理程式的 XCF 群組項目。

   當您在此實務範例中執行公用程式時，下列訊息是正常的:

```
CSQU566I Unable to get attributes for admin structure, CF not found
or not allocated
CSQU546E Unable to add QMGR queue_manager_name entry,
already exists in DB2 table CSQ.ADMIN_B_QMGR
CSQU148I CSQ5PQSG Utility completed, return code=4
```

如果要回復佇列共用群組中的佇列管理程式，請執行下列動作:

1. 定義新的頁面集資料集，並從主要網站將資料與頁面集副本中的資料一起載入。
2. 定義新的作用中日誌資料集。
3. 定義新的 BSDS 資料集，並使用 Access Method Services REPRO 將最新保存的 BSDS 複製到其中。
4. 使用列印日誌對映公用程式 CSQJU004 來列印來自這個最新 BSDS 的資訊。 在保存此 BSDS 時，您所擁有的最新保存日誌會被截斷為作用中日誌，而不會顯示為保存日誌。 記錄此日誌的 STARTRBA、STARTLRSN、ENDRBA 及 ENDRSN 值。
5. 使用變更日誌庫存公用程式 CSQJU003，利用步驟 第 429 頁的『4』中所記錄的值，在您剛還原的 BSDS 中登錄這個最新的保存日誌資料集。
6. 使用 CSQJU003 的 DELETE 選項，從 BSDS 中移除所有作用中日誌資訊。
7. 使用 CSQJU003 的 NEWLOG 選項將作用中日誌新增至 BSDS ，不要指定 STARTRBA 或 ENDRBA。
8. 計算佇列共用群組的 *recoverylrsn* 。 *recoverylrsn* 是佇列共用群組中所有佇列管理程式的最低 ENTLRSNs (如步驟 第 429 頁的『4』中所記錄)，減 1。 比方說，如果佇列共用群組中有兩個佇列管理程式，其中一個是 B713 3C72 22C5，另一個是 B713 3D45 2123 ，則 *recoverylrsn* 是 B713 3C72 22C4。
9. 使用 CSQJU003 將重新啟動控制記錄新增至 BSDS。 指定:

   ```
   CRESTART CREATE,ENDLRSN= recoverylrsn
   ```

   其中 *recoverylrsn* 是您在步驟 第 429 頁的『8』中記錄的值。

   BSDS 現在說明所有作用中日誌都是空的、您有可用的所有保存日誌，且日誌結尾之後沒有檢查點。

   您必須針對佇列共用群組內的每一個佇列管理程式，將 CRESTART 記錄新增至 BSDS。
10. 使用 START QMGR 指令重新啟動佇列共用群組中的每一個佇列管理程式。 在起始設定期間，會發出類似下列的操作員回覆訊息:

    ```
    CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
    REPLY Y TO CONTINUE, N TO CANCEL
    ```

    回覆 Y 以啟動佇列管理程式。 佇列管理程式會啟動，並將資料回復至 CRESTART 陳述式中指定的 ENDRBA。

    對於 IBM WebSphere MQ 7.0.1 以及更新版本，第一個啟動的佇列管理程式可以為佇列共用群組的其他成員及其自己的成員重建管理結構分割區，而且在此階段不再需要重新啟動佇列共用群組中的每一個佇列管理程式。
11. 當已重建所有佇列管理程式的管理結構資料時，請針對每一個 CF 應用程式結構發出 RECOVER CFSTRUCT 指令。

    如果您針對單一佇列管理程式上的所有結構發出 RECOVER CFSTRUCT 指令，則日誌合併處理程序只會執行一次，因此比針對每一個 CF 結構在不同佇列管理程式上發出指令來得快，其中每一個佇列管理程式都必須執行日誌合併步驟。

在佇列共用群組中使用條件式重新啟動處理程序時， IBM WebSphere MQ 7.0.1 以及更新版本的佇列管理程式會執行對等節點管理重建，請檢查對等節點 BSDS 是否包含與其自己相同的 CRESTART LRSN。 這是為了確保重建的管理結構的完整性。 因此，在群組的任何成員下一次無條件重新啟動之前，重新啟動 QSG 中的其他對等節點是很重要的，因此它們可以處理自己的 CRESTART 資訊。

使用磁碟鏡映

現在許多安裝都使用磁碟鏡映技術 (例如 IBM Metro Mirror (舊稱為 PPRC))，在替代站台上建立資料集的同步副本。 在這種情況下，由於替代網站上的 IBM MQ 頁面集和日誌實際上與主要網站上的那些頁面集和日誌完全相同，因此許多詳細的步驟變得不需要。 在使用這類技術的情況下，在替代網站上重新啟動佇列共用群組的步驟可以彙總為:

- 清除替代位置上的 IBM MQ CF 結構。 (這些通常包含先前任何災難回復作業的剩餘資訊)。
- 還原 Db2 系統及 IBM MQ 佇列共用群組所使用資料庫中的所有表格。
- 重新啟動佇列管理程式。 在 IBM WebSphere MQ 7.0.1 之前，必須重新啟動佇列共用群組中定義的每一個佇列管理程式，因為在佇列管理程式重新啟動期間，每一個佇列管理程式都會回復其自己的管理結構分割區。 重新啟動每一個佇列管理程式之後，可以再次關閉不在其起始 LPAR 上的那些佇列管理程式。 對於 IBM WebSphere MQ 7.0.1 以及更新版本，第一個已啟動的佇列管理程式會為佇列共用群組的其他成員及其自己的成員重新建置管理結構分割區，而且不再需要重新啟動佇列共用群組中的每一個佇列管理程式。
- 重建管理結構之後，請回復應用程式結構。

V 9.2.0 IBM MQ 9.1.2 及更高版本支援在寫入使用 Metro Mirror 鏡像的活動日誌時使用 zHyperWrite 。 zHyperWrite 可以幫助減少使用 Metro Mirror 對效能的影響；有關詳細信息，請參閱將 Metro Mirror 與 IBM MQ 結合使用。

z/OS 重新起始設定佇列管理程式

如果佇列管理程式已異常終止，您可能無法重新啟動它。 這可能是因為您的頁面集或日誌已遺失、截斷或毀損。 如果發生這種情況，您可能必須重新起始設定佇列管理程式 (執行冷啟動)。

請注意

只有在無法以任何其他方式重新啟動佇列管理程式時，才會執行冷啟動。 執行冷啟動可讓您回復佇列管理程式及物件定義; 您 **無法** 回復訊息資料。 在執行此動作之前，請檢查本主題中說明的其他重新啟動實務範例是否都不適用於您。

當您重新啟動時，所有 IBM MQ 物件都已定義且可供使用，但沒有訊息資料。

註: 當佇列管理程式是叢集的一部分時，請勿重新起始設定它。 您必須先從叢集中移除佇列管理程式 (在叢集中的其他佇列管理程式上使用 RESET CLUSTER 指令)，然後重新起始設定它，最後將它重新引進叢集作為新的佇列管理程式。

這是因為在重新起始設定期間，佇列管理程式 ID (QMID) 會變更，因此必須從叢集中移除任何具有舊佇列管理程式 ID 的叢集物件。

如需進一步資訊，請參閱下列各節:

- 重新起始設定不在佇列共用群組中的佇列管理程式
- 重新起始設定佇列共用群組中的佇列管理程式

重新起始設定不在佇列共用群組中的佇列管理程式

若要重新起始設定佇列管理程式，請遵循下列程序:

1. 準備要在重新啟動佇列管理程式時使用的物件定義陳述式。 若要執行此動作，請執行下列任一動作:
   - 如果頁集零可用，請使用 CSQUTIL SDEFS 函數 (請參閱 產生 IBM MQ 定義指令清單)。 您必須取得所有物件類型 (鑑別資訊物件、CF 結構、通道、名稱清單、處理程序、佇列及儲存類別) 的定義。
   - 如果無法使用頁集零，請使用前次備份物件定義時的定義。
2. 重新定義佇列管理程式資料集 (在完成步驟 第 430 頁的『1』 之前，請不要這麼做)。

   如需相關資訊，請參閱 建立引導及日誌資料集 及 定義頁面集 。
3. 使用新定義且已起始設定的日誌資料集、BSDS 及頁面集，重新啟動佇列管理程式。 使用您在步驟 第 430 頁的『1』 中建立的物件定義輸入陳述式，作為 CSQINP2 起始設定輸入資料集的輸入。

重新起始設定佇列共用群組中的佇列管理程式

在佇列共用群組中，重新起始設定佇列管理程式更為複雜。 由於頁集或日誌問題，可能需要重新起始設定一或多個佇列管理程式，但 Db2 或要處理的連結機能也可能有問題。 因此，有許多替代方案:

冷啟動 (cold start)
重新起始設定整個佇列共用群組包括強制所有連結機能結構、從 Db2 清除佇列共用群組的所有物件定義、刪除或重新定義日誌和 BSDS ， 以及格式化佇列共用群組中所有佇列管理程式的頁集。

保留的共用定義
刪除或重新定義日誌和 BSDS、佇列共用群組中所有佇列管理程式的格式頁集，以及強制所有連結機能結構。 重新啟動時，將會刪除所有訊息。 佇列管理程式會重建 COPY 物件，這些物件對應於仍然存在於 Db2 資料庫中的 GROUP 物件。 任何共用佇列仍然存在且可以使用。

已重新起始設定單一佇列管理程式
刪除或重新定義日誌和 BSDS ， 以及單一佇列管理程式的格式頁集 (這會刪除其所有專用物件和訊息)。 重新啟動時，佇列管理程式會重建 COPY 物件，這些物件對應於仍然存在於 Db2 資料庫中的 GROUP 物件。 任何共用佇列仍然存在，就像它們上的訊息一樣，可以使用。

佇列共用群組的復原點回復
這是替代站台災難回復實務範例。

共用物件會回復至 Db2 回復 (在 A Db2 系統失效 中說明) 所達到的復原點。 每一個佇列管理程式都可以從替代站台可用的備份副本回復至某個復原點。

持續訊息可以在佇列共用群組中使用，並且可以使用 MQSC RECOVER CFSTRUCT 指令來回復。 請注意，這個指令會回復到失敗的時間。 不過，不會回復非持續性共用佇列訊息; 除非您使用 CSQUTIL 公用程式的 COPY 功能獨立製作備份副本，否則它們會遺失。

不需要嘗試將每一個佇列管理程式還原至相同的復原點，因為不同佇列管理程式上的本端物件 (實際上正在回復) 之間沒有交互相依關係，且重新啟動時佇列管理程式與 Db2 重新同步會根據佇列管理程式在佇列管理程式上的需要建立或刪除 COPY 物件。

z/OS 使用 z/OS Automatic Restart Manager (ARM)

請利用這個主題來瞭解如何使用 ARM 來自動重新啟動佇列管理程式。

本節包含下列主題的相關資訊:

- 第 431 頁的『什麼是 ARM?』
- 第 432 頁的『ARM 原則』
- 第 433 頁的『在 IBM MQ 網路中使用 ARM』

什麼是 ARM?

z/OS Automatic Restart Manager (ARM) 是一種 z/OS 回復功能，可改善佇列管理程式的可用性。 當工作或作業失敗，或執行它的系統失敗時， ARM 可以重新啟動工作或作業，不需要操作員介入。

如果佇列管理程式或通道起始程式失敗， ARM 會在相同的 z/OS 映像檔上重新啟動它。 如果 z/OS 以及整個相關子系統和應用程式群組都失敗， ARM 可以在 Sysplex 內的另一個 z/OS 映像檔上以預先定義的順序自動重新啟動所有失敗的系統。 這稱為 跨系統重新啟動。

只有在異常情況下，才由 ARM 重新啟動通道起始程式。 如果 ARM 重新啟動佇列管理程式，請從 CSQINP2 起始設定資料集重新啟動通道起始程式 (請參閱 第 433 頁的『在 IBM MQ 網路中使用 ARM』)。

如果 z/OS 失敗，您可以使用 ARM ， 在 Sysplex 內的不同 z/OS 映像檔上重新啟動佇列管理程式。 第 433 頁的『在 IBM MQ 網路中使用 ARM』 中說明不同 z/OS 映像檔上 IBM MQ ARM 重新啟動的網路含意。

若要啟用自動重新啟動，請執行下列動作:

- 設定 ARM 連結資料集。
- 在 *ARM* 原則中定義您要 z/OS 執行的自動重新啟動動作。

- 啟動 ARM 原則。

此外， IBM MQ 必須在啟動時向 ARM 登錄 (這會自動發生)。

註: 如果您想要自動重新啟動不同 z/OS 映像檔中的佇列管理程式，您必須在每一個 z/OS 映像檔中，將每一個佇列管理程式定義為可能重新啟動該佇列管理程式的子系統，並使用 Sysplex 層面唯一的四個字元子系統名稱。

ARM 連結資料集

在啟動您想要 ARM 支援的任何佇列管理程式之前，請確定您已定義 ARM 所需的連結資料集，且它們在線上且在作用中。 如果在佇列管理程式啟動時無法使用這兩個資料集，則 IBM MQ 自動 ARM 登錄會失敗。 在這種情況下， IBM MQ 會假設沒有連結資料集，表示您不想要 ARM 支援，且會繼續起始設定。

如需 ARM 連結資料集的相關資訊，請參閱 *z/OS MVS Setting up a Sysplex* 。

z/OS *ARM* 原則

「自動重新啟動管理程式」原則是使用者定義的規則，可控制 ARM 函數，可控制佇列管理程式的任何重新啟動。

ARM 函數由使用者定義的 *ARM* 原則控制。 每一個執行由 ARM 重新啟動之佇列管理程式實例的 z/OS 映像檔，都必須連接至具有作用中 ARM 原則的 ARM 連結資料集。

IBM 提供預設 ARM 原則。 您可以使用 z/OS 提供的 管理資料公用程式 (IXCMIAPU) 來定義新的原則，或置換原則預設值。 *z/OS MVS* 設定 *Sysplex* 說明此公用程式，並包含如何定義 ARM 原則的完整詳細資料。

第 432 頁的圖 44 顯示 ARM 原則的範例。 如果佇列管理程式失敗或整個系統失敗，此範例原則會重新啟動 Sysplex 內的任何佇列管理程式。

```
//IXCMIAPU EXEC PGM=IXCMIAPU,REGION=2M
//SYSPRINT DD SYSOUT=*
//SYSIN  DD *
DATA TYPE(ARM)
DEFINE POLICY NAME(ARMPOL1) REPLACE(YES)
RESTART_GROUP(DEFAULT)
ELEMENT(*)
RESTART_ATTEMPTS(0)  /* Jobs not to be restarted by ARM  */
RESTART_GROUP(GROUP1)
ELEMENT(SYSMQMGRMQ*)  /* These jobs to be restarted by ARM */
/*
```

圖 *44: ARM* 原則範例

如需相關資訊，請參閱:

- 定義 ARM 原則
- 啟動 ARM 原則
- 向 ARM 登錄

定義 ARM 原則

設定 ARM 原則，如下所示:

- 為每一個佇列管理程式實例定義 RESTART_GROUPs ，這些佇列管理程式實例也包含連接至該佇列管理程式實例的任何 CICS 或 IMS 子系統。 如果您使用子系統命名慣例，則可能可以使用 '?' 以及元素名稱中的 '*' 萬用字元，以最少的定義工作量來定義 RESTART_GROUPS。
- 指定通道起始程式的 TERMTYPE (ELEMTERM) ， 以指出只有在通道起始程式失敗且 z/OS 映像檔未失敗時，才會重新啟動它們。
- 指定佇列管理程式的 TERMTYPE (ALLTERM) ， 以指出如果佇列管理程式失敗或 z/OS 映像檔失敗，將會重新啟動它們。

- 同時為佇列管理程式及通道起始程式指定 RESTART_METHOD (BOTH , PERSIST)。 這會告知 ARM 在前次啟動期間使用它所儲存的 JCL 來重新啟動 (在系統符號解析之後)。 不論個別元素是否失敗，或 z/OS 映像檔是否失敗，它都會告訴 ARM 來執行這個動作。
- 接受所有其他 ARM 原則選項的預設值。

啟動 ARM 原則

若要啟動自動重新啟動管理原則，請發出下列 z/OS 指令:

```
SETXCF START,POLICY,TYPE=ARM,POLNAME= mypol
```

啟動原則時，所有連接至 ARM 聯結資料集的系統都會使用相同的作用中原則。

請使用 SETXCF STOP 指令來停用自動重新啟動。

向 ARM 登錄

在佇列管理程式啟動期間， IBM MQ 會自動登錄為 *ARM* 元素 (取決於 ARM 可用性)。 除非未要求，否則它會在其關閉階段期間取消登錄。

啟動時，佇列管理程式會判斷 ARM 是否可用。 如果是， IBM MQ 會使用名稱 SYSMQMGR *ssid* 進行登錄，其中 *ssid* 是四個字元的佇列管理程式名稱，而 SYSMQMGR 是元素類型。

STOP QMGR MODE (QUIESCE) 及 STOP QMGR MODE (FORCE) 指令會從 ARM 取消登錄佇列管理程式 (如果在啟動時已向 ARM 登錄)。 這會防止 ARM 重新啟動此佇列管理程式。 STOP QMGR MODE (RESTART) 指令不會從 ARM 取消登錄佇列管理程式，因此它適合立即自動重新啟動。

每一個通道起始程式位址空間都會決定 ARM 是否可用，如果可用，則會以元素名稱 SYSMQCH *ssid* 登錄，其中 *ssid* 是佇列管理程式名稱，而 SYSMQCH 是元素類型。

通道起始程式在正常停止時一律會從 ARM 取消登錄，且只有在異常結束時才會維持登錄。 如果佇列管理程式失敗，通道起始程式一律會取消登錄。

z/OS 在 *IBM MQ* 網路中使用 *ARM*

您可以設定佇列管理程式，以便在重新啟動佇列管理程式時自動啟動通道起始程式及相關聯的接聽器。

若要確保在 LU 6.2 及 TCP/IP 通訊協定的相同 z/OS 映像檔上完全自動重新啟動佇列管理程式，請執行下列動作:

- 將適當的 START LISTENER 指令新增至 CSQINPX 資料集，以自動啟動接聽器。
- 將適當的 START CHINIT 指令新增至 CSQINP2 資料集，以自動啟動通道起始程式。

若要使用 TCP/IP 或 LU6.2 重新啟動佇列管理程式，請參閱

- 第 433 頁的『使用 TCP/IP 在不同的 z/OS 映像檔上重新啟動』
- 第 435 頁的『在具有 LU 6.2 的不同 z/OS 映像檔上重新啟動』

如需 CSQINP2 及 CSQINPX 資料集的相關資訊，請參閱 作業 13: 自訂起始設定輸入資料集 。

使用 TCP/IP 在不同的 z/OS 映像檔上重新啟動

如果您使用 TCP/IP 作為通訊協定，並且使用虛擬 IP 位址，則可以將這些配置為在其他 z/OS 映像檔上回復，容許連接至該佇列管理程式的通道重新連接，而無需進行任何變更。 否則，只有在您使用叢集或使用「WLM 動態網域名稱系統 (DNS)」邏輯群組名稱連接至佇列共用群組時，才能在將佇列管理程式移至不同的 z/OS 映像檔之後重新配置 TCP/IP 位址。

- 使用叢集作業時
- 連接至佇列共用群組時

使用叢集作業時

z/OS ARM 會在相同 Sysplex 的不同 z/OS 映像檔上重新啟動佇列管理程式，以回應系統故障; 這個系統與原始 z/OS 映像檔有不同的 TCP/IP 位址。 下列說明在 ARM 重新啟動將佇列管理程式的 TCP/IP 位址移至不同的 z/OS 映像檔之後，如何使用 IBM MQ 叢集來重新指派它。

當用戶端佇列管理程式偵測到佇列管理程式失敗 (作為通道失敗) 時，它會回應將其叢集傳輸佇列上的適當訊息重新配置給管理目標叢集佇列不同實例的不同伺服器佇列管理程式。 不過，它無法重新配置因親緣性限制而連結至原始伺服器的訊息，或因為伺服器佇列管理程式在批次結束處理期間失敗而不確定的訊息。 若要處理這些訊息，請執行下列動作:

1. 配置不同的叢集接收端通道名稱及不同的 TCP/IP 埠給每一個 z/OS 佇列管理程式。 每一個佇列管理程式都需要不同的埠，以便兩個系統可以在 z/OS 映像檔上共用單一 TCP/IP 堆疊。 其中一個是最初在該 z/OS 映像檔上執行的佇列管理程式，另一個是 ARM 在系統失效之後將在該 z/OS 映像檔上重新啟動的佇列管理程式。 在每一個 z/OS 映像檔上配置每一個埠，以便 ARM 可以在任何 z/OS 映像檔上重新啟動任何佇列管理程式。
2. 針對要在通道起始程式啟動期間參照的每一個佇列管理程式及 z/OS 映像檔組合，建立不同的通道起始程式指令輸入檔 (CSQINPX)。

   每一個 CSQINPX 檔案都必須包括該佇列管理程式特有的 START LISTENER PORT (port) 指令，以及該佇列管理程式及 z/OS 映像檔組合特有之叢集接收端通道的 ALTER CHANNEL 指令。 ALTER CHANNEL 指令需要將連線名稱設為重新啟動所在之 z/OS 映像檔的 TCP/IP 名稱。 它必須包含重新啟動的佇列管理程式特有的埠號，作為連線名稱的一部分。

   每一個佇列管理程式的啟動 JCL 都可以具有此 CSQINPX 檔案的固定資料集名稱，且每一個 z/OS 映像檔在非共用 DASD 磁區上必須具有每一個 CSQINPX 檔案的不同版本。

如果發生 ARM 重新啟動， IBM MQ 會將已變更的通道定義通告至叢集儲存庫，然後將它發佈至表示對伺服器佇列管理程式感興趣的所有用戶端佇列管理程式。

用戶端佇列管理程式會將伺服器佇列管理程式失敗視為通道失敗，並嘗試重新啟動失敗的通道。 當用戶端佇列管理程式得知新的伺服器連線名稱時，通道重新啟動會將用戶端佇列管理程式重新連接至重新啟動的伺服器佇列管理程式。 然後，用戶端佇列管理程式可以重新同步化其訊息，解決用戶端佇列管理程式的傳輸佇列上任何不確定的訊息，並可繼續正常處理。

連接至佇列共用群組時

透過 TCP/IP 動態網域名稱系統 (DNS) 邏輯群組名稱連接至佇列共用群組時，通道定義中的連線名稱會指定佇列共用群組的邏輯群組名稱，而不是實體機器的主機名稱或 IP 位址。 當此通道啟動時，它會連接至動態 DNS，然後連接至佇列共用群組中的其中一個佇列管理程式。 使用佇列共用群組來設定 IBM MQ for z/OS 的通訊中說明了此程序。

在映像檔失敗的不太可能事件中，會發生下列其中一項:

- 從 Sysplex 上執行的動態 DNS 取消登錄失敗映像檔上的佇列管理程式。 通道會進入 RETRYING 狀態以回應連線失敗，然後連接至在 Sysplex 上執行的動態 DNS。 動態 DNS 會將入埠要求配置給仍在剩餘映像檔上執行的佇列共用群組的其中一個剩餘成員。
- 如果佇列共用群組中沒有其他佇列管理程式在作用中，且 ARM 會在不同映像檔上重新啟動佇列管理程式和通道起始程式，則群組接聽器會從這個新映像檔向動態 DNS 登錄。 這表示邏輯群組名稱 (從通道的連線名稱欄位) 連接至動態 DNS，然後連接至現在在不同映像檔上執行的相同佇列管理程式。 不需要變更通道定義。

若要進行這種類型的回復，必須注意下列要點:

- 在 z/OS 上，動態 DNS 會在 Sysplex 的其中一個 z/OS 映像檔上執行。 如果此映像檔失敗，則需要配置動態 DNS，以便在 Sysplex 中有作用中的次要名稱伺服器，作為主要名稱伺服器的替代方案。 如需主要及次要動態 DNS 伺服器的相關資訊，請參閱 *OS/390 SecureWay CS IP* 配置 手冊。
- TCP/IP 群組接聽器可能已在此 z/OS 映像檔上無法使用的特定 IP 位址上啟動。 如果是這樣，則可能需要在新映像檔的不同 IP 位址上啟動接聽器。 如果您使用虛擬 IP 位址，則可以將這些配置為在其他 z/OS 映像檔上回復，以便不需要變更 START LISTENER 指令。

在具有 LU 6.2 的不同 z/OS 映像檔上重新啟動

如果您只使用 LU 6.2 通訊協定，請執行下列程序，在 Sysplex 內不同 z/OS 映像檔上自動重新啟動佇列管理程式之後啟用網路重新連接:

- 請使用唯一子系統名稱來定義 Sysplex 內的每一個佇列管理程式。
- 使用唯一 LUNAME 來定義 Sysplex 內的每一個通道起始程式。 這同時在佇列管理程式屬性及 START LISTENER 指令中指定。

  註: LUNAME 會命名 APPC 端表格中的登錄，然後將此對映至實際 LUNAME。
- 設定共用 APPC 端表格，供 Sysplex 內每一個 z/OS 映像檔參照。 這應該包含每一個通道起始程式的 LUNAME 的項目。 如需相關資訊，請參閱 *z/OS MVS* 規劃 *:APPC/MVS* 管理 。
- 針對 Sysplex 內的每一個通道起始程式設定 SYS1.PARMLIB 的 APPCPM *xx* 成員，以包含 LUADD 來啟動該通道起始程式的 APPC 端表格項目。 這些成員應該由每一個 z/OS 映像檔共用。 適當的 SYS1.PARMLIB 成員由 z/OS 指令 SET APPC= *xx* 啟動，在不同 z/OS 映像檔上 ARM 重新啟動佇列管理程式 (及其通道起始程式) 期間會自動發出此指令，如下列文字中所述。
- 使用 LU62ARM 佇列管理程式屬性，為每一個通道起始程式指定此 SYS1.PARMLIB 成員的 *xx* 字尾。 這會導致通道起始程式發出必要的 z/OS 指令 SET APPC= *xx* ，以啟動其 LUNAME。

定義 ARM 原則，以便只有在通道起始程式的 z/OS 映像檔保持啟動時失敗時，它才會重新啟動; 與 XCFAS 位址空間相關聯的使用者 ID 必須獲得授權，才能發出 IBM MQ 指令 START CHINIT。 如果通道起始程式的 z/OS 映像檔也失敗，請不要自動重新啟動通道起始程式，請改用 CSQINP2 及 CSQINPX 資料集中的指令來啟動通道起始程式及接聽器。

z/OS 手動回復工作單元

您可以手動回復佇列共用群組中的工作單元 CICS、 IMS、RRS 或其他佇列管理程式。 您可以使用佇列管理程式指令來顯示與佇列管理程式的每一個連線相關聯的工作單元狀態。

本主題包含下列主題的相關資訊:

- 第 435 頁的『顯示連線和執行緒』
- 第 436 頁的『手動回復 CICS 回復單元』
- 第 439 頁的『手動回復 IMS 回復單元』
- 第 440 頁的『手動回復 RRS 回復單元』
- 第 441 頁的『在佇列共用群組中的另一個佇列管理程式上回復回復單元』

顯示連線和執行緒

您可以使用 DISPLAY CONN 指令來取得佇列管理程式及其相關聯工作單元的連線相關資訊。 您可以顯示作用中工作單元，以查看目前發生的情況，或查看需要終止以容許佇列管理程式關閉的項目，並且您可以顯示未解決的工作單元，以協助進行回復。

作用中工作單元

若只要顯示作用中的工作單元，請使用

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ ACTIVE)
```

尚未解決的工作單元

未解析的工作單元 (也稱為「不確定的執行緒」) 是在兩段式確定作業的第二次傳遞中的工作單元。 資源代表其保留在 IBM MQ 中。 如果要顯示尚未解析的工作單元，請使用

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

需要外部介入來解決未解決工作單元的狀態。 這可能只涉及啟動回復協調程式 (CICS、 IMS 或 RRS) , 也可能涉及更多，如下列各節所述。

z/OS 手動回復 *CICS* 回復單元

使用本主題來瞭解當 CICS 配接器重新啟動時所發生的情況，然後說明如何處理任何產生的未解決回復單元。

當 CICS 配接卡重新啟動時發生的情況

每當連線中斷時，配接器必須在 重新連接處理程序期間經歷 重新啟動階段 。 重新啟動階段會重新同步化資源。 CICS 與 IBM MQ 之間的重新同步可識別並解決不確定的工作單元。

重新同步的原因可能是:

- 來自分散式佇列元件的明確要求
- 對 IBM MQ 建立連線時的隱含要求

如果連接至 IBM MQ 導致重新同步，則事件順序為:

1. 連線程序會從 IBM MQ 擷取不確定的工作單元 (UOW) ID 清單。
2. UOW ID 會顯示在主控台的 CSQC313I 訊息中。
3. UOW ID 會傳遞至 CICS。
4. CICS 會針對每一個不確定的 UOW ID 起始重新同步作業 (CRSY)。
5. 主控台上會顯示每一個不確定 UOW 的作業結果。

您需要檢查在連接程序期間顯示的訊息:

CSQC313I
顯示 UOW 不確定。

CSQC400I
識別 UOW , 後面接著下列其中一則訊息:

- CSQC402I 或 CSQC403I 顯示已順利解析 UOW (已確定或已取消)。
- CSQC404E、 CSQC405E、 CSQC406E 或 CSQC407E 顯示未解析 UOW。

CSQC409I
顯示已順利解析所有 UOW。

CSQC408I
顯示並非所有 UOW 都已順利解決。

CSQC314I
警告不會自動解析以 * 強調顯示的 UOW ID。 這些 UOW 必須在重新啟動時由分散式佇列元件明確解析。

第 437 頁的圖 45 顯示 z/OS 主控台上顯示的一組重新啟動訊息範例。

```
CSQ9022I +CSQ1 CSQYASCP ' START QMGR' NORMAL COMPLETION
+CSQC323I VICIC1 CSQCQCON CONNECT received from TERMID=PB62 TRANID=CKCN
+CSQC303I VICIC1 CSQCCON CSQCSERV loaded. Entry point is 850E8918
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F0E2178D25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F055B2AC25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6EFFD60D425 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F07AB56D22 is in doubt
+CSQC307I VICIC1 CSQCCON Successful connection to subsystem VC2
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAD18) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAA10) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BA708) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAE88) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAB80) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA878) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA570) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA268) connect
successful
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F0E2178D25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F055B2AC25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F07AB56D22
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6EFFD60D425
+CSQC409I VICIC1 CSQCTRUE Resynchronization completed successfully
```

圖 *45:* 重新啟動訊息範例

CSQC313I 訊息總數應該等於 CSQC402I 加上 CSQC403I 訊息總數。 如果總計不相等，則有連線處理程序無法解析的 UOW。 無法解決的那些 UOW 是由 CICS (例如，冷開機) 或 IBM MQ 的問題或配送佇列作業所造成。 當這些問題已修正時，您可以中斷連線，然後重新連接，以起始另一個重新同步。

或者，您可以使用 RESOLVE INDOUBT 指令及訊息 CSQC400I 中顯示的 UOW ID ，自行解決每一個未完成的 UOW。 然後，您必須起始斷線及連接，以清除 CICS 中的 回復單元描述子 。 您需要知道 UOW 的正確結果，才能手動解析 UOW。

IBM MQ 會鎖定所有與未解析 UOW 相關聯的訊息，且沒有任何「批次」、TSO 或 CICS 作業可以存取它們。

如果 CICS 失敗，且需要緊急重新啟動， 請勿 改變 CICS 系統的 GENERIC APPLID。 如果您這樣做，然後重新連接至 IBM MQ，則無法保證 IBM MQ 的資料完整性。 這是因為 IBM MQ 會將新的 CICS 實例視為不同的 CICS (因為 APPLID 不同)。 因此，不確定的解決方案是根據錯誤的 CICS 日誌。

如何手動解析 CICS 回復單元

如果配接卡異常結束，則 CICS 及 IBM MQ 建置不確定清單會動態或在重新啟動期間，視導致異常終止的子系統而定。

註: 如果您使用 DFH$INDB 範例程式來顯示工作單元，則可能會發現它並非一律正確地顯示 IBM MQ UOW。

當 CICS 連接至 IBM MQ 時，可能有一個以上尚未解析的回復單元。

下列其中一則訊息會傳送至主控台:

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E
- CSQC408I

有關這些訊息含義的詳細信息，請參閱 CICS 適配器訊息和 Bridge 訊息。

CICS 會保留連線啟動期間未解析的回復單元詳細資料。 當項目不再出現在 IBM MQ 所呈現的清單上時，即會清除該項目。

任何 CICS 無法解析的回復單元都必須使用 IBM MQ 指令來手動解析。 此手動程序很少在安裝內使用，因為只有在作業錯誤或軟體問題導致無法自動解決時才需要。 必須調查在不確定解決期間發現的任何不一致。

若要解析回復單元，請執行下列動作:

1. 使用下列指令，從 IBM MQ 取得回復單元的清單:

```
+CSQ1  DISPLAY CONN( * ) WHERE(UOWSTATE EQ UNRESOLVED)
```

   您會收到下列訊息:

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC85772CBE3E0001)
EXTCONN(C3E2D8C3C7D9F0F94040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2005-02-04)
UOWLOGTI(10.17.44)
UOWSTDA(2005-02-04)
UOWSTTI(10.17.44)
UOWSTATE(UNRESOLVED)
NID(IYRCSQ1 .BC8571519B60222D)
EXTURID(BC8571519B60222D)
QMURID(0000002BDA50)
URTYPE(CICS)
USERID(MQTEST)
APPLTAG(IYRCSQ1)
ASID(0000)
APPLTYPE(CICS)
TRANSID(GP02)
TASKNO(0000096)
END CONN DETAILS
```

   對於 CICS 連線， NID 由 CICS 應用程式 ID 和 CICS 在寫入同步點日誌項目時提供的唯一號碼組成。 這個唯一數字儲存在同步點處理時間同時寫入 CICS 系統日誌和 IBM MQ 日誌的記錄中。 在 CICS 中，此值稱為 回復記號。

2. 掃描 CICS 日誌，以找出與特定回復單元相關的項目。

   尋找作業相關安裝的 PREPARE 記錄，其中回復記號欄位 (JCSRMTKN) 等於從網路 ID 取得的值。 網路 ID 由 IBM MQ 在 DISPLAY CONN 指令輸出中提供。

   回復單元的 CICS 日誌中的 PREPARE 記錄提供 CICS 作業號碼。 使用此數字可以找到此 CICS 作業日誌上的所有其他項目。

   掃描日誌時，您可以使用 CICS 日誌登載列印公用程式 DFHJUP。 如需使用此程式的詳細資料，請參閱 *CICS* 作業和公用程式手冊。

3. 掃描 IBM MQ 日誌，以找出具有與特定回復單元相關之 NID 的記錄。 然後使用此記錄中的 URID 來取得此回復單元的其餘日誌記錄。

   掃描 IBM MQ 日誌時，請注意 IBM MQ 啟動訊息 CSQJ001I 會提供此階段作業的啟動 RBA。

   列印日誌記錄程式 (CSQ1LOGP) 可用於該目的。

4. 如果您需要，請在 IBM MQ 中執行不確定的解決方案。

   可以使用 IBM MQ RESOLVE INDOUBT 指令，引導 IBM MQ 採取回復單元的回復動作。

   若要回復與特定 *connection-name* 相關聯的所有執行緒，請使用 NID (*) 選項。

   該指令會產生下列其中一則訊息，指出執行緒是已確定還是已取消:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

執行不確定的解決方案時， CICS 和配接器不知道 IBM MQ 用來確定或取消回復單元的指令，因為只會影響 IBM MQ 資源。 不過， CICS 會保留 IBM MQ 無法解決之不確定執行緒的詳細資料。 當呈現的清單是空的，或當清單不包含 CICS 具有詳細資料的回復單元時，即會清除此資訊。

z/OS 手動回復 *IMS* 回復單元

使用本主題來瞭解當 IMS 配接器重新啟動時所發生的情況，然後說明如何處理任何產生的未解決回復單元。

當 IMS 配接卡重新啟動時發生的情況

每當重新啟動與 IBM MQ 的連線時，在佇列管理程式重新啟動或 IMS /START SUBSYS 指令之後， IMS 會起始下列重新同步處理程序:

1. IMS 會以「確定」或「取消」的解析參數來呈現它認為對 IBM MQ IMS 配接卡不確定的工作單元 (UOW) ID 清單 (一次一個)。
2. IMS 配接器將解析要求傳遞至 IBM MQ ， 並將結果回報給 IMS。
3. 在處理所有 IMS 解析要求之後， IMS 配接器會從 IBM MQ 取得 IBM MQ 仍不確定且由 IMS 系統起始的所有 UOW 清單。 這些會以訊息 CSQQ008I 向 IMS 主要終端機報告。

註: 當 UOW 不確定時， IBM MQ 會鎖定任何相關聯的 IBM MQ 訊息，且無法供任何應用程式使用。

如何手動解析 IMS 回復單元

當 IMS 連接至 IBM MQ 時， IBM MQ 可能有一個以上不確定的回復單元尚未解決。

如果 IBM MQ 具有 IMS 未解析的不確定回復單元，則會在 IMS 主要終端機上發出下列訊息:

```
CSQQ008I nn units of recovery are still in doubt in queue manager qmgr-name
```

如果發出此訊息，則表示 IMS 已冷啟動或以不完整的日誌磁帶啟動。 如果 IBM MQ 或 IMS 因軟體錯誤或其他子系統失敗而異常終止，也會發出此訊息。

收到 CSQQ008I 訊息之後:

- 連線會保持作用中。
- IMS 應用程式仍然可以存取 IBM MQ 資源。
- 部分 IBM MQ 資源仍被鎖定。

如果未解析不確定的執行緒， IMS 訊息佇列可以開始建置。 如果 IMS 佇列填滿容量，則 IMS 會終止。 您必須瞭解這個潛在的困難，且必須監視 IMS ， 直到完全解決不確定的回復單元為止。

回復程序

使用下列程序來回復 IMS 工作單元:

1. 使用 /SWI OLDS 強制關閉 IMS 日誌，然後保存 IMS 日誌。 使用 DFSERA10 公用程式來列印前一個 IMS 日誌磁帶中的記錄。 類型 X'3730' 日誌記錄指出 phase-2 確定要求，類型 X'38' 日誌記錄指出中斷要求。 記錄每一個相依區域中最後一個交易所要求的動作。
2. 執行 DL/I 批次工作，以取消涉及未達到確定點的每一個 PSB。 處理程序可能需要一些時間，因為交易仍在處理中。 它也可能會鎖定一些記錄，這可能會影響其餘的處理及其餘的訊息佇列。
3. 使用下列指令，從 IBM MQ 產生不確定的回復單元清單:

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

您會收到下列訊息:

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3E2C5C3F24040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2005-02-15)
UOWLOGTI(16.39.43)
UOWSTDA(2005-02-15)
UOWSTTI(16.39.43)
UOWSTATE(UNRESOLVED)
NID(IM8F  .BC45A794D3810344)
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0000)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

對於 IMS， NID 由 IMS 連線名稱和 IMS 提供的唯一號碼組成。 在 IMS 中，此值稱為 回復記號。 如需相關資訊，請參閱 *IMS* 自訂作業手冊。

4. 比較 DISPLAY THREAD 訊息中顯示的 NID (IMSID 加上十六進位的 OASN) 與 DFSERA10 輸出中顯示的 OASN (4 位元組十進位)。 決定要確定還是取消。
5. 使用 RESOLVE INDOUBT 指令在 IBM MQ 中執行不確定的解析，如下所示:

```
RESOLVE INDOUBT( connection-name )
ACTION(COMMIT|BACKOUT)
NID( network-id )
```

若要回復與 *connection-name* 相關聯的所有執行緒，請使用 NID (*) 選項。 此指令會產生下列其中一則訊息，指出執行緒是否已確定或取消:

```
CSQV414I  THREAD network-id COMMIT SCHEDULED
CSQV415I  THREAD network-id BACKOUT SCHEDULED
```

執行不確定的解析時， IMS 及配接器不知道 IBM MQ 的指令，無法確定或取消不確定的回復單元，因為只會影響 IBM MQ 資源。

z/OS 手動回復 *RRS* 回復單元

請利用這個主題來瞭解如何判斷是否有不確定的 RRS 回復單元，以及如何手動解決這些回復單元。

當 RRS 連接至 IBM MQ 時， IBM MQ 可能有一個以上不確定的回復單元尚未解決。 如果 IBM MQ 具有 RRS 未解析的不確定回復單元，則會在 z/OS 主控台發出下列其中一則訊息:

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

IBM MQ 和 RRS 都提供工具來顯示不確定回復單元的相關資訊，以及手動解決它們的技術。

在 IBM MQ 中，使用 DISPLAY CONN 指令來顯示不確定 IBM MQ 執行緒的相關資訊。 指令的輸出包括 RRS 作為協調程式之 IBM MQ 執行緒的 RRS 回復單元 ID。 這可用來判斷回復單元的結果。

請使用 RESOLVE INDOUBT 指令來手動解決 IBM MQ 不確定執行緒。 在您決定正確的決策之後，此指令可用來確定或取消回復單元。

z/OS 在佇列共用群組中的另一個佇列管理程式上回復回復單元

請利用這個主題來識別及手動回復佇列共用群組中其他佇列管理程式的回復單元。

如果屬於佇列共用群組成員的佇列管理程式失敗且無法重新啟動，則群組中的其他佇列管理程式可以執行同層級回復，並從中接管。 不過，佇列管理程式可能有無法由同層級回復解決的不確定回復單元，因為只有失敗的佇列管理程式才知道該回復單元的最終處置。 當佇列管理程式最終重新啟動時，會解析這些回復單元，但在此之前，它們仍無法確定。

這表示某些資源 (例如訊息) 可能已鎖定，使群組中的其他佇列管理程式無法使用它們。 在此狀況下，您可以使用 DISPLAY THREAD 指令在非作用中佇列管理程式上顯示這些工作單元。 如果您要手動解析這些回復單元，使訊息可供群組中的其他佇列管理程式使用，您可以使用 RESOLVE INDOUBT 指令。

當您發出 DISPLAY THREAD 指令以顯示不確定的回復單元時，您可以使用 QMNAME 關鍵字來指定非作用中佇列管理程式的名稱。 例如，如果您發出下列指令:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

您會收到下列訊息:

```
CSQV436I +CSQ1 INDOUBT THREADS FOR QM01 -
NAME   THREAD-XREF    URID NID
USER1  00000000000000000000000000 CSQ:0001.0
USER2  00000000000000000000000000 CSQ:0002.0
DISPLAY THREAD REPORT COMPLETE
```

如果指定的佇列管理程式處於作用中，則 IBM MQ 不會傳回不確定執行緒的相關資訊，但會發出下列訊息:

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

使用 IBM MQ 指令 RESOLVE INDOUBT 來手動解決不確定的執行緒。 使用 QMNAME 關鍵字，在指令中指定非作用中佇列管理程式的名稱。

此指令可用來確定或取消回復單元。 此指令只會解析回復單元的共用部分; 任何本端訊息都不會受到影響，且會保持鎖定狀態，直到佇列管理程式重新啟動或重新連接至 CICS、 IMS 或 RRS 批次為止。

z/OS IBM MQ 及 IMS

IBM MQ 提供兩個元件來與 IMS、 IBM MQ - IMS 配接器及 IBM MQ - IMS 橋接器連接。 這些元件通常稱為 IMS 配接器，以及 IMS 橋接器。

z/OS 操作 IMS 配接器

請利用這個主題來瞭解如何操作 IMS 配接器，以將 IBM MQ 連接至 IMS 系統。

註: IMS 配接器不會納入任何作業和控制面板。

本主題包含下列各節:

- 第 442 頁的『控制 IMS 連線』
- 第 442 頁的『從 IMS 控制區域連接』
- 第 444 頁的『顯示不確定的回復單元』
- 第 445 頁的『控制 IMS 相依區域連線』
- 第 447 頁的『中斷連線 IMS』
- 第 448 頁的『控制 IMS 觸發監視器』

z/OS 控制 *IMS* 連線

請利用這個主題來瞭解控制及監視 IBM MQ 連線的 IMS 操作員指令。

IMS 提供下列操作員指令，以控制及監視與 IBM MQ 的連線:

/CHANGE SUBSYS
從 IMS 刪除不確定的回復單元。

/DISPLAY OASN SUBSYS
顯示未執行的回復元素。

/DISPLAY SUBSYS
顯示連線狀態及執行緒活動。

/START SUBSYS
將 IMS 控制區域連接至佇列管理程式。

/STOP SUBSYS
中斷 IMS 與佇列管理程式的連線。

/TRACE
控制 IMS 追蹤。

如需這些指令的相關資訊，請參閱您正在使用之 IMS 層次的 *IMS/ESA Operator 's Reference* 手冊。

IMS 指令回應會傳送至從中發出指令的終端機。 發出 IMS 指令的授權基於 IMS 安全。

z/OS 從 *IMS* 控制區域連接

請利用這個主題來瞭解可從 IMS 連接至 IBM MQ 的機制。

IMS 會從其控制區域建立一個連線，以連接至每一個使用 IMS 的佇列管理程式。 必須啟用 IMS，才能以下列其中一種方式建立連線:

- 在下列任一期間自動執行:
  - 冷啟動起始設定。
  - 如果在 IMS 關閉時 IBM MQ 連線處於作用中狀態，則 IMS 會暖啟動。
- 為了回應 IMS 指令:

```
/START SUBSYS sysid
```

其中 *sysid* 是佇列管理程式名稱。

不論佇列管理程式是否在作用中，都可以發出指令。

在對佇列管理程式進行第一次 MQ API 呼叫之前，不會建立連線。 在該時間之前， IMS 指令 /DIS SUBSYS 會將狀態顯示為 'NOT CONN'。

您啟動 IMS 及佇列管理程式的順序並不重要。

IMS cannot re-enable the connection to the queue manager automatically if the queue manager is stopped with a STOP QMGR command, the IMS command /STOP SUBSYS, or an abnormal end. 因此， 您必須使用 IMS 指令 /START SUBSYS 來建立連線。

如果在佇列管理程式主控台日誌中看到類似如下的 IMS 指令:

```
MODIFY IMS*,SS*
```

檢查 IMS 主日誌並確保 IBM MQ 具有 RACF 權限來發出 IMS 適配器 MODIFY 命令。

起始設定配接器並連接至佇列管理程式

配接器是一組模組，使用 IMS 外部子系統連接機能載入至 IMS 控制和相依區域。

此程序會起始設定配接器並連接至佇列管理程式:

1. 從 IMS 讀取子系統成員 (SSM)。PROCLIB。 選擇的 SSM 是 IMS EXEC 參數。 對於「 IMS 」可以連接的每一個佇列管理程式，成員中有一個項目。 每一個項目都包含 IBM MQ 配接卡的相關控制資訊。
2. 載入 IMS 配接器。

   註: IMS 會為 SSM 成員中定義的每一個 IBM MQ 實例載入一個配接器模組副本。
3. 連接 IBM MQ 的外部子系統作業。
4. 以 CTL EXEC 參數 (IMSID) 作為連線名稱來執行配接器。

無論連線是起始設定的一部分，還是 IMS 指令 /START SUBSYS 的結果，處理程序都是相同的。

當 IMS 嘗試建立連線時，如果佇列管理程式處於作用中，則會傳送下列訊息:

- 至 z/OS 主控台:

```
DFS3613I ESS TCB INITIALIZATION COMPLETE
```

- 至 IMS 主要終端機:

```
CSQQ000I IMS/TM imsid connected to queue manager ssnm
```

當 IMS 嘗試建立連線且 佇列管理程式不在作用中時，每次應用程式發出 MQI 呼叫時，下列訊息會傳送至 IMS 主要終端機:

```
CSQQ001I IMS/TM imsid not connected to queue manager ssnm.
Notify message accepted
DFS3607I MQM1   SUBSYSTEM ID EXIT FAILURE, FC = 0286, RC = 08,
JOBNAME = IMSEMPR1
```

如果您在啟動與 IMS 的連線時或在系統啟動時收到 DFS3607I 訊息，則表示佇列管理程式無法使用。 若要防止產生大量訊息，您必須執行下列其中一項:

1. 啟動相關佇列管理程式。
2. 發出 IMS 指令:

   ```
   /STOP SUBSYS
   ```

   因此 IMS 不會預期連接至佇列管理程式。

如果您都不這麼做，則每次在區域中排定工作時，以及每次應用程式對佇列管理程式提出連線要求時，都會發出 DFS3607I 訊息及相關聯的 CSQQ001I 訊息。

執行緒附件

在 MPP 或 IFP 區域中，當第一個應用程式排定到該區域時，即使該應用程式未發出 IBM MQ 呼叫， IMS 也會建立執行緒連線。 在 BMP 區域中，當應用程式進行其第一個 IBM MQ 呼叫 (MQCONN 或 MQCONNX) 時，會建立執行緒連線。 此執行緒會在區域期間保留，或直到連線停止為止。

對於訊息驅動及非訊息驅動區域，與執行緒相關聯的回復執行緒交互參照 ID *Thread-xref* 為:

```
PSTid + PSBname
```

其中:

PSTid
分割區規格表格區域 ID

PSBname
指定程式區塊名稱

您可以在 IBM MQ 指令中使用連線 ID 作為唯一 ID，在此情況下，IBM MQ 會自動將這些 ID 插入它所產生的任何操作員訊息中。

z/OS 顯示不確定的回復單元

您可以顯示不確定的回復單元，並嘗試回復它們。

本主題中用來列出及回復不確定回復單元的作業步驟僅適用於相對簡單的情況。如果佇列管理程式在連接至 IMS 時異常結束，則 IMS 可能會確定或取消工作，而不會 IBM MQ 知道它。當佇列管理程式重新啟動時，該工作稱為 不確定。必須針對工作狀態做出決策。

若要顯示不確定的回復單元清單，請發出下列指令:

```
+CSQ1  DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

IBM MQ 會回應如下的訊息:

```
CSQM201I +CSQ1 CSQMDRTC DIS CONN DETAILS
CONN(BC0F6125F5A30001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-11-02)
UOWLOGTI(12.27.58)
UOWSTDA(2004-11-02)
UOWSTTI(12.27.58)
UOWSTATE(UNRESOLVED)
NID(CSQ1CHIN.BC0F5F1C86FC0766)
EXTURID(000000000000001F000000007472616E5F6964547565204E6F762020...)
QMURID(000000026232)
URTYPE(XA)
USERID( )
APPLTAG(CSQ1CHIN)
ASID(0000)
APPLTYPE(CHINIT)
CHANNEL( )
CONNAME( )
END CONN DETAILS
```

如需此訊息中屬性的說明，請參閱 DISPLAY CONN 指令的說明。

回復不確定的回復單元

若要回復不確定的回復單元，請發出下列指令:

```
+CSQ1  RESOLVE INDOUBT( connection-name ) ACTION(COMMIT|BACKOUT)
NID( net-node.number )
```

其中:

connection-name
IMS 系統 ID。

動作
指出要確定 (COMMIT) 或取消 (BACKOUT) 此回復單元。

net-node.number
相關聯的 net-node.number。

當您發出 RESOLVE INDOUBT 指令時，會顯示下列其中一則訊息:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED

CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

解析剩餘回復項目

在給定時間， IMS 會建置殘留回復項目 (RRE) 的清單。 RRE 是 IBM MQ 可能不確定的回復單元。 它們在幾種情況下產生:

- 如果佇列管理程式不在作用中，則 IMS 具有在佇列管理程式處於作用中之前無法解析的 RRE。 這些 RRE 不是問題。
- 如果佇列管理程式處於作用中並連接至 IMS，且 IMS 取消 IBM MQ 已確定的工作，則 IMS 配接器會發出訊息 CSQQ010E。 如果兩個系統中的資料必須一致，則會有問題。 如需解決此問題的相關資訊，請參閱第 439 頁的『手動回復 IMS 回復單元』。
- 如果佇列管理程式處於作用中並連接至 IMS，則即使沒有訊息通知您此問題，仍可能有 RRE。 建立與 IMS 的 IBM MQ 連線之後，您可以發出下列 IMS 指令，以找出是否有問題:

  ```
  /DISPLAY OASN SUBSYS sysid
  ```

若要清除 RRE ，請發出下列其中一個 IMS 指令:

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

其中 *nnnn* 是為了回應 +CSQ1 DISPLAY 指令而列出的原始應用程式序號。 這是程式實例的排程號碼，提供它自前次 IMS 冷啟動以來在該程式呼叫順序中的位置。 IMS 不能有兩個不確定的回復單元具有相同的排程號碼。

這些指令會重設 IMS 的狀態; 它們不會導致與 IBM MQ 進行任何通訊。

z/OS 控制 *IMS* 相依區域連線

您可以控制、監視及在必要時終止 IMS 與 IBM MQ 之間的連線。

控制 IMS 相依區域連線涉及下列活動:

- 從相依區域連接
- 區域錯誤選項
- 監視連線上的活動
- 切斷與相依區域的連線

從相依區域連接

控制區域中使用的 IMS 配接器也會載入至相依區域。 會建立從每一個相依區域到 IBM MQ 的連線。 此連線用於協調 IBM MQ 和 IMS 工作的確定。 若要起始設定並建立連線， IMS 會執行下列動作:

1. 從 IMS 讀取子系統成員 (SSM)。 PROCLIB。

   可以在相依區域 EXEC 參數上指定子系統成員。 如果未指定，則會使用控制區域 SSM。 如果區域永不可能連接至 IBM MQ，為了避免載入配接器，請指定不含任何項目的成員。

2. 載入 IBM MQ 配接器。

   對於批次訊息程式，除非應用程式發出其第一個傳訊指令，否則不會完成載入。 此時， IMS 會嘗試建立連線。

   對於訊息處理程式區域或 IMS 捷徑區域，會在起始設定區域時進行嘗試。

區域錯誤選項

如果佇列管理程式不在作用中，或從應用程式傳送第一個傳訊指令時無法使用資源，則所採取的動作取決於 SSM 項目上指定的錯誤選項。 選項有:

R
適當的回覆碼會傳送至應用程式。

Q
應用程式異常結束，異常終止碼為 U3051。 輸入訊息已重新排入佇列。

A
應用程式異常結束，異常終止碼為 U3047。 已捨棄輸入訊息。

監視連線上的活動

當應用程式提出其第一個成功 IBM MQ 要求時，會從相依區域建立執行緒。 您可以從 IBM MQ 發出下列指令，以顯示連線及目前使用它們的應用程式的相關資訊:

```
+CSQ1 DISPLAY CONN(*) ALL
```

指令會產生如下所示的訊息:

```
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-12-15)
UOWLOGTI(16.39.43)
UOWSTDA(2004-12-15)
UOWSTTI(16.39.43)
UOWSTATE(ACTIVE)
NID( )
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0049)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

對於控制區域， *thread-xref* 是特殊值 CONTROL。 對於相依區域，它是與 PSBname 連結的 PSTid。 *auth-id* 是工作卡中的使用者欄位，或 z/OS 啟動程序表格中的 ID。

如需所顯示清單的說明，請參閱 IBM MQ for z/OS 訊息、完成和原因代碼 文件中訊息 CSQV402I 的說明。

IMS 提供顯示指令來監視與 IBM MQ 的連線。 它會顯示每一個相依區域連線上的作用中程式、LTERM 使用者名稱及控制區域連線狀態。 指令為:

```
/DISPLAY SUBSYS name
```

IMS 與 IBM MQ 之間的連線狀態顯示為下列其中一項:

```
CONNECTED
NOT CONNECTED
CONNECT IN PROGRESS
STOPPED
STOP IN PROGRESS
INVALID SUBSYSTEM NAME= name
SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING
```

每一個相依區域的執行緒狀態是下列其中一項:

```
CONN
CONN, ACTIVE (includes LTERM of user)
```

中斷與相依區域的連線

變更 IMS 的 SSM 成員中的值。PROCLIB ， 您中斷相依區域的連線。 若要執行此動作，您必須:

1. 發出 IMS 指令:

   ```
   /STOP REGION
   ```

2. 更新 SSM 成員。
3. 發出 IMS 指令:

   ```
   /START REGION
   ```

z/OS 中斷連線 *IMS*

當 IMS 或佇列管理程式終止時，會結束連線。 或者， IMS 主要終端機操作員可以明確中斷連線。

若要終止 IMS 與 IBM MQ 之間的連線，請使用下列 IMS 指令:

```
/STOP SUBSYS sysid
```

該指令會將下列訊息傳送至發出該訊息的終端機，通常是主要終端機操作員 (MTO):

```
DFS058I STOP COMMAND IN PROGRESS
```

IMS 指令:

```
/START SUBSYS sysid
```

需要重新建立連線。

註: 如果 IMS 觸發監視器正在執行中，則 IMS 指令 /STOP SUBSYS 未完成。

z/OS 控制 *IMS* 觸發監視器

您可以使用 CSQQTRMN 交易來停止並啟動 IMS 觸發監視器。

設定 IMS 觸發監視器中說明 IMS 觸發監視器 (CSQQTRMN 交易)。

若要控制 IMS 觸發監視器，請參閱:

- 啟動 CSQQTRMN
- 停止 CSQQTRMN

啟動 CSQQTRMN

1. 針對您要監視的每一個起始佇列，啟動執行程式 CSQQTRMN 的批次導向 BMP。
2. 修改批次 JCL 以新增 DDname CSQQUT1 ，其指向包含下列資訊的資料集:

```
QMGRNAME=q_manager_name    Comment: queue manager name
INITQUEUENAME=init_q_name   Comment: initiation queue name
LTERM=lterm           Comment: LTERM to remove error messages
CONSOLEMESSAGES=YES      Comment: Send error messages to console
```

   其中:

q_manager_name	佇列管理程式的名稱 (如果這是空白，則會採用 CSQQDEFV 中指定的預設值)
init_q_name	要監視的起始佇列名稱
lterm	錯誤訊息目的地的 IMS LTERM 名稱 (如果此為空白，則預設值為 MASTER)。
CONSOLEMESSAGES=YES	傳送至指定 IMS LTERM 的訊息也會傳送至 z/OS 主控台的要求。 如果省略或拼錯此參數，則預設值為「不」將訊息傳送至主控台。

3. 如果您想要 CSQQUT1 輸入處理程序的列印報告，請新增 DD 名稱 CSQQUT2 。

註:

1. 使用 LRECL=80 定義資料集 CSQQUT1 。 其他 DCB 資訊取自資料集。 資料集 CSQQUT2 的 DCB 是 RECFM=VBA 和 LRECL=125。
2. 您只能在每一筆記錄上放置一個關鍵字。 關鍵字值以關鍵字後面的第一個空白區隔; 這表示您可以併入註解。 直欄 1 中的星號表示整個輸入記錄是註解。
3. 如果您拼錯 QMGRNAME 或 LTERM 關鍵字，則 CSQQTRMN 會使用該關鍵字的預設值。
4. 提交觸發監視器 BMP 工作之前，請確定已在 IMS 中啟動子系統 (由 /START SUBSYS 指令啟動)。 如果未啟動，則觸發監視器工作會終止，異常終止碼為 U3042。

正在停止 CSQQTRMN

啟動之後， CSQQTRMN 會一直執行，直到 IBM MQ 與 IMS 之間的連線因下列其中一個事件而中斷為止:

- 佇列管理程式結束中

- IMS 結束

或輸入 z/OS STOP **jobname** 指令。

z/OS 控制 IMS 橋接器

請利用這個主題來瞭解您可以用來控制 IMS 橋接器的 IMS 指令。

沒有 IBM MQ 指令可控制 IBM MQ-IMS 橋接器。 不過，您可以使用下列方式來停止將訊息遞送至 IMS：

- 若為非共用佇列，請針對所有橋接器佇列使用 ALTER QLOCAL (xxx) GET (DISABLED) 指令。
- 若為叢集佇列，請使用 SUSPEND QMGR CLUSTER (xxx) 指令。 這只有在另一個佇列管理程式也管理叢集橋接器佇列時才有效。
- 對於叢集佇列，請使用 SUSPEND QMGR FACILITY (IMSBRIDGE) 指令。 不會將進一步訊息傳送至 IMS，但會從 IMS 收到任何未完成交易的回應。

  若要重新開始傳送訊息至 IMS，請發出 RESUME QMGR FACILITY (IMSBRIDGE) 指令。

您也可以使用 MQSC 指令 DISPLAY SYSTEM 來顯示橋接器是否已暫停。

如需這些指令的詳細資料，請參閱 MQSC 指令。

如需進一步資訊，請參閱：

- 第 449 頁的『啟動及停止 IMS 橋接器』
- 第 449 頁的『控制 IMS 連線』
- 控制橋接器佇列
- 第 450 頁的『重新同步化 IMS 橋接器』
- 使用 tpipe 名稱
- 從 IMS 刪除訊息
- 刪除 tp 管道
- 第 452 頁的『IMS 交易到期』

啟動及停止 IMS 橋接器

透過啟動 OTMA 來啟動 IBM MQ 橋接器。 請使用 IMS 指令：

```
/START OTMA
```

或在 IMS 系統參數中指定 OTMA=YES 來自動啟動它。 如果 OTMA 已啟動，當佇列管理程式啟動完成時，橋接器會自動啟動。 啟動 OTMA 時會產生 IBM MQ 事件訊息。

使用 IMS 指令：

```
/STOP OTMA
```

來停止 OTMA 通訊。 當發出這個指令時，會產生 IBM MQ 事件訊息。

控制 IMS 連線

IMS 提供下列操作員指令，以控制及監視與 IBM MQ 的連線：

/DEQUEUE TMEMBER *tmember* TPIPE *tpipe*
從 Tpipe 移除訊息。 指定 PURGE1 以移除所有訊息，或指定 PURGE1 以僅移除第一個訊息。

/DISPLAY OTMA
顯示 OTMA 伺服器和用戶端的摘要資訊，以及用戶端狀態。

/DISPLAY TMEMBER *name*
顯示 OTMA 用戶端的相關資訊。

/DISPLAY TRACE TMEMBER *name*
顯示正在追蹤之內容的相關資訊。

/SECURE OTMA
設定安全選項。

/START OTMA
透過 OTMA 啟用通訊。

/START TMEMBER *tmember* TPIPE *tpipe*
啟動具名 Tpipe。

/STOP OTMA
停止透過 OTMA 的通訊。

/STOP TMEMBER *tmember* TPIPE *tpipe*
停止具名 Tpipe。

/TRACE
控制 IMS 追蹤。

如需這些指令的相關資訊，請參閱您正在使用之 IMS 層次的 *IMS/ESA Operators Reference* 手冊。

IMS 指令回應會傳送至從中發出指令的終端機。 發出 IMS 指令的授權基於 IMS 安全。

控制橋接器佇列

若要透過橋接器停止與具有 XCF 成員名稱 *tmember* 的佇列管理程式通訊，請發出下列 IMS 指令:

```
/STOP TMEMBER tmember TPIPE ALL
```

若要回復通訊，請發出下列 IMS 指令:

```
/START TMEMBER tmember TPIPE ALL
```

可以使用 MQ DISPLAY QUEUE 指令來顯示佇列的 Tp 管道。

若要在單一 Tpipe 上停止與佇列管理程式的通訊，請發出下列 IMS 指令:

```
/STOP TMEMBER tmember TPIPE tpipe
```

會為每一個作用中橋接器佇列建立一或兩個 Tp 管道，因此發出此指令會停止與 IBM MQ 佇列的通訊。 若要回復通訊，請使用下列 IMS 指令:

```
/START TMEMBER tmember TPIPE tpipe
```

或者，您可以變更 IBM MQ 佇列的屬性，讓它被禁止。

重新同步化 IMS 橋接器

每當佇列管理程式、 IMS 或 OTMA 重新啟動時，都會自動重新啟動 IMS 橋接器。

IMS 橋接器所承擔的第一個作業是與 IMS 重新同步化。 這涉及 IBM MQ 和 IMS 檢查每個同步化 Tpipe 上的序號。 使用確定模式零 (commit-then-send) 將持續訊息從 IBM MQ - IMS 橋接器佇列傳送至 IMS 時，會使用 synchronized Tpipe。

如果橋接器無法與 IMS 重新同步化，則會在訊息 CSQ2023E 中傳回 IMS 感應碼，且會停止與 OTMA 的連線。 如果橋接器無法與個別 IMS Tpipe 重新同步化，則會在訊息 CSQ2025E 中傳回 IMS 感應碼，且 Tpipe 已停止。 如果已冷啟動 Tpipe ，則可回復的序號會自動重設為 1。

如果橋接器在與 Tpipe 重新同步化時探索到不符的序號，則會發出 CSQ2020E 訊息。 使用 IBM MQ 指令 RESET TPIE 來起始與 IMS Tpipe 的重新同步。 您需要提供 XCF 群組和成員名稱，以及 Tpipe 的名稱; 此資訊由訊息提供。

您也可以指定:

- 要在 Tpipe 中設定 IBM MQ 所傳送訊息的新可回復序號，並設為友機的接收序號。 如果您未指定此項，則夥伴的接收序號會設為現行 IBM MQ 傳送序號。
- 要在 Tpipe 中針對 IBM MQ 接收的訊息設定新的可回復序號，並設為夥伴的傳送序號。 如果您未指定此項，則夥伴的傳送序號會設為現行 IBM MQ 接收序號。

如果有未解析的回復單元與 Tpipe 相關聯，則也會在訊息中通知此回復單元。 請使用 IBM MQ 指令 RESET TPIPE 來指定是要確定回復單元，還是取消回復單元。 如果您確定回復單元，則訊息批次已傳送至 IMS，且已從橋接器佇列中刪除。 如果您支援回復單元輸出，則訊息會傳回橋接器佇列，稍後會傳送至 IMS。

確定模式 1 (send-then-commit) Tp 管道未同步。

確定模式 1 交易的考量

在 IMS 中，確定模式 1 (CM1) 交易會在同步點之前傳送其輸出回覆。

CM1 交易可能無法傳送其回覆，例如:

- 要傳送回覆的 Tpipe 已停止
- OTMA 已停止
- OTMA 用戶端 (即佇列管理程式) 已消失
- 無法使用回覆目的地佇列及無法傳送郵件的佇列

基於這些原因， IMS 應用程式傳送訊息虛擬異常終止，程式碼為 U0119。 在此情況下，不會停止 IMS 交易及程式。

這些原因通常會阻止將訊息傳送至 IMS，以及阻止從 IMS 遞送回覆。 在下列情況下，可能會發生 U0119 異常終止:

- 當訊息位於 IMS 時， Tpipe、OTMA 或佇列管理程式會停止
- IMS 在不同的 Tpipe 上回覆送入訊息，且 Tpipe 已停止
- IMS 會回覆不同的 OTMA 用戶端，且該用戶端無法使用。

每當發生 U0119 異常終止時， IMS 的送入訊息和 IBM MQ 的回覆訊息都會遺失。 如果 CM0 交易的輸出因上述任何原因而無法遞送，則會在 IMS 內的 Tpipe 上排入佇列。

使用 tpipe 名稱

許多用來控制 IBM MQ - IMS 橋接器的指令都需要 *tpipe* 名稱。 請利用這個主題來瞭解如何尋找 tpipe 名稱的進一步詳細資料。

許多控制 IBM MQ - IMS 橋接器的指令都需要 *tpipe* 名稱。 您可以從 DISPLAY QUEUE 指令取得 tpipe 名稱，並注意下列要點:

- 當定義本端佇列時，會指派 tpipe 名稱
- 本端佇列有兩個 tpipe 名稱，一個用於同步，另一個用於非同步
- 在 IMS 與 IBM MQ 之間特定於該特定本端佇列的某些通訊發生之後， IMS 才會知道 tpipe 名稱
- 若要讓 tpipe 可供 IBM MQ - IMS 橋接器使用，必須將其相關聯佇列指派給已完成正確 XCF 群組及成員名稱欄位的「儲存類別」

刪除其中的訊息 IMS

如果 Tmember/Tpipe 已停止，則可以刪除透過 IMS 橋接器以 IBM MQ 為目的地的訊息。 若要針對具有 XCF 成員名稱 *tmember* 的佇列管理程式刪除一則訊息，請發出下列 IMS 指令:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

若要刪除 Tpipe 上的所有訊息，請發出下列 IMS 指令:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

正在刪除 tp 管道

您無法自行刪除 IMS tp 管道。 IMS 會在下列時間刪除它們:

- 當 IMS 冷啟動時，會刪除已同步的 tp 管道。
- 當 IMS 重新啟動時，會刪除未同步的 tp 管道。

IMS 交易到期

有效期限與交易相關聯; 任何 IBM MQ 訊息都可以有相關聯的有效期限。 使用 MQMD.Expiry 欄位，將有效期限間隔從應用程式傳遞至 IBM MQ。 時間是訊息到期之前的持續時間，以十分之一秒的值表示。 嘗試執行訊息的 MQGET (晚於它已過期) 會導致從佇列中移除訊息，並執行到期處理程序。 當訊息在 IBM MQ 網路上的佇列管理程式之間流動時，有效期限會減少。 當 IMS 訊息透過 IMS 橋接器傳遞至 OTMA 時，剩餘的訊息到期時間會作為交易到期時間傳遞至 OTMA。

如果交易已指定有效期限，則 OTMA 會在 IMS 中的三個不同位置使輸入交易到期:

- 從 XCF 接收輸入訊息
- 輸入訊息移入佇列時間
- 應用程式 GU 時間

在 GU 時間之後不會執行任何到期。

交易 EXPRTIME 可以由下列提供:

- IMS 交易定義 (transaction definition)
- IMS OTMA 訊息標頭
- IMS DFSINSX0 使用者結束程式
- IMS CREATE 或 UPDATE TRAN 指令

IMS 指出它已過期交易，方法為異常終止交易 0243 ， 並發出訊息。 發出的訊息為非共用佇列環境中的 DFS555I ，或共用佇列環境中的 DFS2224I 。

z/OS 在 z/OS 上操作 Advanced Message Security

Advanced Message Security 位址空間接受使用 z/OS MODIFY 指令的指令。

若要輸入 Advanced Message Security (AMS) 位址空間的指令，請使用 z/OS MODIFY 指令。

例如,

```
F qmgrAMSM, cmd
```

其中 *qmgr* 是已啟動作業名稱的字首。

第 453 頁的表 29 說明接受的 MODIFY 指令:

表 29: *Advanced Message Security* 位址空間 *MODIFY* 指令		
指令	**選項**	**說明**
顯示		顯示版本資訊
重新整理	keyRing 原則 ALL	重新整理金鑰環憑證及/或安全原則。
SMFAUDIT	成功 失敗 ALL	設定當 AMS 順利保護或解除保護訊息時，以及當 AMS 無法保護或解除保護訊息時，是否需要 SMF 審核。
SMFTYPE	0 - 255	設定當 AMS 保護或取消保護訊息時要產生的 SMF 記錄類型。 如果要停用 SMF 審核，請指定記錄類型 0。

註: 若要指定選項，必須以逗點區隔。 例如:

```
F qmgrAMSM,REFRESH KEYRING
F qmgrAMSM,SMFAUDIT ALL
F qmgrAMSM,SMFTYPE 180
```

refresh 指令

透過發出 **REFRESH** 指令而生效的變更會套用至在 **REFRESH** 指令完成之後發出 MQOPEN 的應用程式。 已開啟佇列的現有應用程式會繼續使用應用程式開啟佇列時的選項。 若要使用新值，應用程式必須關閉並重新開啟佇列。

啟動和停止 AMS

您不需要輸入指令來啟動或停止 Advanced Message Security 位址空間。 如果已使用 CSQ6SYSP 的 **SPLCAP** 參數啟用 AMS，則會在啟動佇列管理程式時自動啟動 AMS 位址空間，並在佇列管理程式停止時停止。

管理 IBM MQ Internet Pass-Thru

本節說明如何管理 IBM MQ Internet Pass-Thru (MQIPT)。

透過變更 `mqipt.conf` 配置檔來配置 MQIPT，如 配置 IBM MQ Internet Pass-Thru 中所述。 若要管理 MQIPT(包括重新整理 MQIPT 以讓配置變更生效而不重新啟動 MQIPT)，請使用 **mqiptAdmin** 指令。 如需使用 **mqiptAdmin** 指令來管理 MQIPT 的相關資訊，請參閱 第 455 頁的『使用指令行來管理 MQIPT』。

啟動和停止 MQIPT

您可以從指令行啟動 MQIPT，或讓它在系統啟動時自動啟動。 您可以使用 **mqiptAdmin** 指令來停止 MQIPT。

從指令行啟動 MQIPT

MQIPT 已安裝至安裝目錄，例如:

- Windows Windows 系統上的 `C:\MQIPT`，在 `C:\MQIPT\bin` 中具有執行檔 Script
- Linux AIX AIX and Linux 系統上的 `/opt/mqipt`，在 `/opt/mqipt/bin` 中具有執行檔 Script

MQIPT 也會使用起始目錄，其中包含配置檔 `mqipt.conf`，以及 MQIPT 執行時所輸出的任何檔案。 第一次呼叫 MQIPT 時，會自動建立 MQIPT 起始目錄的下列子目錄:

- 在其中寫入任何 First Failure Support Technology (FFST) 和追蹤檔的 `errors` 目錄
- 在其中保留連線日誌的 `logs` 目錄

用來執行 MQIPT 的使用者 ID 必須具有建立這些目錄的許可權，或者目錄必須已存在，且使用者 ID 必須具有在其中建立、讀取及寫入檔案的許可權。 此外，如果您使用 Java security manager 原則，則安全原則必須授與這些目錄的必要許可權。 如需 Security Manager 原則設定的相關資訊，請參閱 Java security manager。

您可以使用安裝目錄作為起始目錄。 如果您使用此目錄，則必須確保執行 MQIPT 所使用的使用者 ID 具有適當的許可權，且任何安全管理程式原則都已正確配置。

若要啟動 MQIPT，請使用 **mqipt** 指令，該指令位於 MQIPT 安裝目錄的 `bin` 目錄中。 例如，下列指令會啟動使用 `C:\mqiptHome` 目錄作為起始目錄的 MQIPT 實例:

```
mqipt C:\mqiptHome
```

如需 **mqipt** 指令的相關資訊，請參閱 mqipt (start MQIPT)。

V 9.2.0 V 9.2.0 您可以使用 **mqipt** 指令來指定要提供給所啟動 MQIPT 實例的名稱。 MQIPT 實例的名稱是用來使用 **mqiptAdmin** 指令來管理 MQIPT 的本端實例，而不需要使用指令埠。 如果未指定此參數，則會使用 MQIPT 起始目錄的名稱作為 MQIPT 實例的名稱。

主控台訊息顯示 MQIPT 的狀態。 如果發生錯誤，請參閱 疑難排解 IBM MQ Internet Pass-Thru。 下列訊息是 MQIPT 順利啟動時的輸出範例:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2025. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.2.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is C:\mqiptHome
MQCPI021 Password checking has been enabled on the command port
MQCPI144 MQ Advanced capabilities not enabled
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1414 is starting and will forward messages to :
MQCPI034 ....examplehost(1414)
MQCPI035 ....using MQ protocols
MQCPI057 ....trace level 5 enabled
MQCPI078 Route 1414 ready for connection requests
```

自動啟動 MQIPT

您可以將 MQIPT 安裝為系統服務，並在系統啟動時自動啟動。 使用 **mqiptService** 指令來安裝及解除安裝 MQIPT 服務。

- Windows 在 Windows 系統上， **mqiptService** 指令會將 MQIPT 安裝為 Windows 服務。
- Linux AIX 在 AIX and Linux 系統上， **mqiptService** 指令會將 MQIPT 安裝為系統開機時啟動的 System V init 服務。 在不支援 System V init 的 Linux 系統上，請使用另一種方法 (例如 `systemd`) ，將 MQIPT 當作服務來管理。

當 MQIPT 服務啟動時，所有作用中 MQIPT 路徑都會啟動。 當服務停止時，所有路徑都會立即關閉。

即使系統上有多個 MQIPT 安裝，也只能在系統上安裝一個 MQIPT 服務。

如需 **mqiptService** 指令的相關資訊，請參閱 mqiptService (管理 MQIPT 服務)。

停止中 MQIPT

您可以搭配使用 **mqiptAdmin** 指令與 **-stop** 參數，來停止 MQIPT 。

V 9.2.0 V 9.2.0 例如，下列指令會停止名稱為 `mqipt1` 的 MQIPT 實例，該實例在本端以與 **mqiptAdmin** 指令相同的使用者 ID 執行:

```
mqiptAdmin -stop -n ipt1
```

mqiptAdmin 指令會連接至 MQIPT 的作用中實例，以使用下列其中一種方法來管理:

- V 9.2.0 V 9.2.0 透過連接至 MQIPT 的本端實例，而不使用指令埠。
- 建立與指令埠的網路連線。

必須透過將 **RemoteShutDown** 內容設為 `true` 來啟用遠端關閉，然後才能使用 **mqiptAdmin** 指令將指令傳送至指令埠來停止 MQIPT 。

如需使用 **mqiptAdmin** 指令來管理 MQIPT 的相關資訊，請參閱 第 455 頁的『使用指令行來管理 MQIPT』。

V 9.2.0 指定密碼加密金鑰

從 IBM MQ 9.1.5 開始，如果 MQIPT 配置包含使用預設金鑰以外的加密金鑰加密的密碼，則您必須在 MQIPT 啟動時可以讀取的檔案中提供密碼加密金鑰。

密碼加密金鑰檔

可以使用您提供的加密金鑰來加密要由 MQIPT 儲存及使用的密碼。 如果您未提供加密金鑰，則會使用預設加密金鑰。 您不需要指定密碼加密金鑰，但這樣做更安全。 如果您未指定自己的加密金鑰，則會使用預設加密金鑰。

如果您提供密碼加密金鑰，它必須儲存在 **mqiptPW** 指令可用來加密密碼和 MQIPT 的檔案中。 檔案內容的唯一限制是它必須至少包含一個字元，且只有一行文字。

註: 您必須確保在密碼加密金鑰檔上設定適當的檔案許可權，以防止任何未獲授權的使用者讀取加密金鑰。 只有執行 **mqiptPW** 指令的使用者及執行 MQIPT 的使用者才需要讀取密碼加密金鑰的權限。

使用相同的密碼加密金鑰來加密及解密 MQIPT 實例的所有已儲存密碼。 因此，每一個 MQIPT 安裝架構只需要單一密碼加密金鑰檔。

如果變更 MQIPT 安裝的密碼加密金鑰，則必須使用新的加密金鑰來重新加密所有已加密密碼。

啟動中 MQIPT

密碼加密金鑰檔的預設名稱是 *MQIPT_HOME_DIR*/mqipt_cred.key，其中 *MQIPT_HOME_DIR* 是儲存 `mqipt.conf` 配置檔的目錄。 如果您計劃以自動啟動的服務形式執行 MQIPT ，則必須建立具有預設名稱的密碼加密金鑰檔。

如果使用預設名稱以外的名稱來建立密碼加密金鑰檔，則在啟動時必須將檔案名稱提供給 MQIPT 。 您可以依照喜好設定，使用下列任何方法來指定密碼加密金鑰檔的名稱:

1. **mqipt** 指令上用來啟動 MQIPT 的 **-sf** 參數。
2. `MQS_MQIPTCRED_KEYFILE` 環境變數。
3. `com.ibm.mq.ipt.cred.keyfile` Java 內容。

如果未提供密碼加密金鑰檔名，則會使用預設檔名 (如果檔案存在的話)。 如果預設密碼加密金鑰檔不存在，則會使用預設密碼加密金鑰。

使用指令行來管理 MQIPT

您可以在指令行上使用 **mqiptAdmin** 指令來管理 MQIPT。

您可以使用 **mqiptAdmin** 指令來執行下列管理功能:

- 列出 MQIPT 的作用中本端實例。
- 在變更配置檔之後，請重新整理 MQIPT 的實例。
- 停止 MQIPT 實例。

`mqiptAdmin` 指令位於 MQIPT 安裝目錄的 `bin` 子目錄中。

mqiptAdmin 指令會連接至 MQIPT 的作用中實例，以使用下列其中一種方法來管理:

- 建立與指令埠的網路連線。

- V 9.2.0 V 9.2.0 透過連接至 MQIPT 的本端實例，而不使用指令埠。

mqiptAdmin 指令與舊版 MQIPT 相容，但您無法使用該指令來管理版本比 **mqiptAdmin** 指令版本更高的 MQIPT 版本。 在包含不同版本 MQIPT 的環境中，您必須使用最新版本的 **mqiptAdmin** 指令。

如需 **mqiptAdmin** 指令語法的相關資訊，請參閱 mqiptAdmin (管理 MQIPT)。

沒有指令埠的本端管理

V 9.2.0 V 9.2.0

從 IBM MQ 9.2.0 開始，無需使用指令埠即可管理 MQIPT 的本端實例。 只有在與您要管理的 MQIPT 實例相同的系統上執行時，本端管理才可讓您使用 **mqiptAdmin** 指令來管理 MQIPT 。

為了讓 **mqiptAdmin** 有權在不使用指令埠的情況下管理 MQIPT 的本端實例， MQIPT 實例必須在與 **mqiptAdmin** 相同的系統上執行，且使用相同的使用者 ID。 或者，在 AIX and Linux 上， **mqiptAdmin** 可以作為 `root` 執行。

依預設會啟用本端管理。 若要停用本端管理，請使用 **LocalAdmin** 配置內容。 如需 **LocalAdmin** 內容的相關資訊，請參閱 LocalAdmin。

若要管理 MQIPT 的本端實例，您必須為每一個實例提供一個名稱。 您可以在使用 **mqipt** 指令啟動 MQIPT 時使用 **-n** 參數，將名稱指派給 MQIPT 的實例。 如果您在啟動 MQIPT 時未指定名稱，則會使用起始目錄的名稱作為 MQIPT 實例的名稱。 例如，下列指令會啟動 MQIPT ，並將名稱 `ipt1` 指派給實例:

```
mqipt /opt/mqipt1 -n ipt1
```

實例具有名稱之後，您可以透過在 **mqiptAdmin** 指令中使用 **-n** 參數指定名稱來管理該實例。 例如，下列指令會停止名稱為 `ipt1` 的 MQIPT 本端實例:

```
mqiptAdmin -stop -n ipt1
```

您可以透過搭配使用 **mqiptAdmin** 指令與 **-list** 參數，來列出 **mqiptAdmin** 指令獲授權管理的所有 MQIPT 本端作用中實例，而無需使用指令埠。 例如，下列指令會列出啟動 **mqiptAdmin** 指令之使用者獲授權管理的所有 MQIPT 本端作用中實例:

```
mqiptAdmin -list
```

使用指令埠進行管理

V 9.2.0 V 9.2.0

從 IBM MQ 9.2.0 開始，您可以使用一個未受保護的指令埠及一個使用 TLS 保護的指令埠來配置 MQIPT 。 您可以使用這些指令埠，以您要管理之 MQIPT 實例所在系統上的任何使用者身分，或從遠端系統來管理 MQIPT 。

舊版 MQIPT 只接受對未受保護的指令埠發出的管理指令。

註: 未加密與未受保護指令埠的連線，因此透過網路傳送至未受保護指令埠的資料 (包括 MQIPT 存取密碼) 可以對網路上的其他使用者可見。

為了讓 MQIPT 能夠在指令埠上接聽 **mqiptAdmin** 指令所發出的指令，必須在 `mqipt.conf` 配置檔的廣域區段中指定 **CommandPort** 或 **SSLCommandPort** 內容的值。

在啟用任一 MQIPT 指令埠之前，請先檢閱 其他安全考量 中的安全考量。 請考慮對指令埠所接收的指令啟用鑑別。 如需指令埠鑑別的相關資訊，請參閱 第 459 頁的『指令埠鑑別』。

若要使用指令埠來管理 MQIPT 實例，請將 MQIPT 執行所在主機的網址及指令埠號指定為 **mqiptAdmin** 指令的參數。 例如，若要重新整理在 `mqipt.example.com` 上執行的 MQIPT 實例，並將未受保護的指令埠配置為在埠 1890 上接聽，請發出下列指令:

```
mqiptAdmin -refresh -r mqipt.example.com:1890
```

如果您未指定主機名稱及埠號， **mqiptAdmin** 會嘗試連接至 `localhost` 埠 1881。

如需使用 TLS 指令埠來管理 MQIPT 的相關資訊，請參閱 第 457 頁的『使用 TLS 指令埠管理 MQIPT』。

V 9.2.0 V 9.2.0 使用 TLS 指令埠管理 MQIPT

從 IBM MQ 9.2.0 開始， MQIPT 可以配置為使用 TLS 指令埠來接聽 **mqiptAdmin** 指令發出的管理指令。 使用 TLS 指令埠可保護機密資料，例如 **mqiptAdmin** 與 MQIPT 之間網路上的 MQIPT 存取密碼。 使用此程序來配置 TLS 指令埠，並使用 TLS 指令埠來管理 MQIPT 。

關於這項作業

必須使用儲存在 PKCS #12 金鑰環中或支援「PKCS #11 加密記號介面」的加密硬體中的伺服器憑證來配置 TLS 指令埠。 在 TLS 信號交換期間，指令埠伺服器憑證會傳送至 **mqiptAdmin** 指令。 這項作業假設您向授信憑證管理中心 (CA) 要求新的伺服器憑證，且憑證會以檔案形式傳回給您。 **mqiptAdmin** 指令會使用簽署伺服器憑證之 CA 的 CA 憑證來驗證指令埠憑證。 CA 憑證必須儲存在可由 **mqiptAdmin** 指令存取的 PKCS #12 金鑰環中。

TLS 指令埠不支援用戶端憑證鑑別。 若要對發出至指令埠的管理指令啟用鑑別，請參閱 第 459 頁的『指令埠鑑別』。

此程序說明如何使用 **mqiptKeycmd** (iKeyman) 指令行介面 (CLI) 來管理使用 TLS 指令埠所需的金鑰環及數位憑證。 您可以使用 **mqiptKeycmd** 指令來使用 CLI。 如需可用來管理金鑰環和數位憑證之其他指令的相關資訊，請參閱 **mqiptKeyman** 和 **mqiptKeycmd**。

程序

1. 遵循下列步驟，以配置 MQIPT 實例的 TLS 指令埠。
   a) 建立要由 TLS 指令埠使用的 PKCS #12 金鑰環檔。 金鑰環用來儲存 TLS 指令埠伺服器憑證。
   若要使用 CLI 建立金鑰環檔案，請輸入下列指令:

   ```
   mqiptKeycmd -keydb -create -db filename -pw password -type pkcs12
   ```

   其中 *filename* 是要建立的金鑰環檔案的名稱，而 *password* 是金鑰環密碼。
   b) 為 CA 簽署的 TLS 指令埠伺服器憑證建立憑證申請。
   若要使用 iKeyman CLI 來建立憑證申請，請輸入下列指令:

   ```
   mqiptKeycmd -certreq -create -db filename -pw password
               -label label -size key_size -sig_alg algorithm
               -dn distinguished_name -file certreq_filename -type pkcs12
   ```

   其中:

   -db 檔名
   指定金鑰環檔名。

   -pw *password*
   指定金鑰環密碼。

   -label *label*
   指定憑證標籤。

   -size *key_size*
   指定金鑰大小。

   -sig_alg *algorithm*
   指定用於建立項目金鑰組的非對稱簽章演算法。

   -dn *distinguished_name*
   指定以雙引號括住的 X.500 識別名稱。

   -file *certreq_filename*
   指定憑證申請的檔名。

   c) 將步驟 第 457 頁的『1.b』 中建立的憑證申請檔案傳送至 CA 進行簽署。
   d) 在 CA 傳送已簽章的憑證給您之後，請將已簽章的憑證接收到金鑰環檔案中。

若要使用 CLI 將已簽章的憑證接收至金鑰環，請輸入下列指令:

```
mqiptKeycmd -cert -receive -file cert_filename -db filename
            -pw password -type pkcs12
```

其中 *cert_filename* 是包含憑證的檔案名稱， *filename* 是金鑰環檔案的名稱，而 *password* 是金鑰環密碼。

e) 使用 **mqiptPW** 指令來加密金鑰環密碼。

執行下列指令:

```
mqiptPW -sf encryption_key_file
```

其中 *encryption_key_file* 是包含 MQIPT 安裝之密碼加密金鑰的檔案名稱。 如果您的 MQIPT 安裝使用預設密碼加密金鑰，則不需要指定 **-sf** 參數。 在提示時鍵入要加密的金鑰環密碼。

如需 **mqiptPW** 指令的相關資訊，請參閱 加密金鑰環密碼。

f) 編輯 `mqipt.conf` 配置檔，並指定下列內容來配置 TLS 指令埠:

i) 將 **SSLCommandPort** 內容的值設為 TLS 指令埠號。

ii) 將 **SSLCommandPortKeyRing** 內容的值設定為步驟 第 457 頁的『1.a』中所建立金鑰環的檔名。

iii) 在步驟 第 458 頁的『1.e』中，將 **SSLCommandPortKeyRingPW** 的值設為 **mqiptPW** 指令所輸出的字串。

iv) 將 **SSLCommandPortSiteLabel** 內容的值設為 TLS 指令埠憑證的標籤名稱，在步驟 第 457 頁的『1.b』中建立憑證申請時指定。

v) 如果您要將 TLS 指令埠的入埠連線限制為來自特定網路介面的連線，請將 **SSLCommandPortListenerAddress** 內容的值設為屬於 MQIPT 執行所在系統上其中一個網路介面的網址。 例如，若要將 TLS 指令埠的入埠連線限制為僅來自本端機器的連線，請將 **SSLCommandPortListenerAddress** 內容的值設為 `localhost`。

g) 啟動或重新整理 MQIPT ， 以啟用 TLS 指令埠。

MQIPT 會發出主控台訊息，例如下列訊息，以顯示有效的 TLS 指令埠配置:

```
MQCPI155 Listening for control commands on port 1882 on local address * using TLS
MQCPI139 ......secure socket protocols <NULL>
MQCPI031 ......cipher suites <NULL>
MQCPI032 ......key ring file c:\\iptHome\\ssl\\commandport.p12
MQCPI072 ......and certificate label mqiptadmin
```

2. 在使用 **mqiptAdmin** 指令來管理 MQIPT 的系統上，請遵循下列步驟，讓 **mqiptAdmin** 能夠連接至 TLS 指令埠。

a) 建立 PKCS #12 金鑰環，以作為 **mqiptAdmin** 指令的信任儲存庫。

若要使用 CLI 建立金鑰環檔案，請輸入下列指令:

```
mqiptKeycmd -keydb -create -db filename -pw password -type pkcs12
```

其中 *filename* 是要建立的金鑰環檔案的名稱，而 *password* 是金鑰環密碼。

b) 將已簽署 TLS 指令埠憑證之 CA 的 CA 憑證匯入步驟 第 458 頁的『2.a』中所建立的金鑰環。

若要使用 iKeyman CLI 匯入 CA 憑證，請輸入下列指令:

```
mqiptKeycmd -cert -add -db filename -pw password -type pkcs12
            -label certlabel -file cert_filename
```

其中:

檔名
指定金鑰環檔名

密碼
指定金鑰環密碼

certlabel
指定要提供給 CA 憑證的標籤

cert_filename
指定包含 CA 憑證的檔案名稱

c) 使用 **mqiptPW** 指令來加密金鑰環密碼。

執行下列指令：

```
mqiptPW -sf encryption_key_file
```

其中 *encryption_key_file* 是包含密碼加密金鑰的檔案名稱。 密碼加密金鑰檔可以不同於 MQIPT 配置中用來加密密碼的檔案。 如果您未使用 **-sf** 參數指定加密金鑰檔，則會使用預設密碼加密金鑰。 在提示時鍵入要加密的金鑰環密碼。

如需 **mqiptPW** 指令的相關資訊，請參閱 加密金鑰環密碼。

d) 建立要由 **mqiptAdmin** 指令使用的內容檔，並指定下列內容：

```
SSLClientCAKeyRing=key_ring_file_name
SSLClientCAKeyRingPW=key_ring_password
PasswordProtectionKeyFile=encryption_key_file
```

其中：

key_ring_file_name
是在步驟 第 458 頁的『2.a』中建立的金鑰環檔案名稱。

key_ring_password
是步驟 第 459 頁的『2.c』中 **mqiptPW** 指令所輸出的已加密密碼。

ENCRYPTION_KEY_FILE
是包含密碼加密金鑰的檔案名稱。 只有在步驟 第 459 頁的『2.c』中使用加密金鑰檔來加密金鑰環密碼時，才需要指定 **PasswordProtectionKeyFile** 內容。

e) 發出 **mqiptAdmin** 指令以管理 MQIPT，指定 **-s** 參數以指出需要 TLS 連線，並指定 **-p** 參數以指定在步驟 第 459 頁的『2.d』中建立的內容檔名稱。

例如，輸入下列指令，將 refresh 指令傳送至 TLS 指令埠，以重新整理 MQIPT 的實例：

```
mqiptAdmin -refresh -r hostname:port -s -p properties_file
```

mqiptAdmin 指令會發出如下的訊息，以確認 MQIPT 的連線受到 TLS 保護：

```
MQCAI109 The connection to MQIPT is secured with TLSv1.2.
```

下一步

若要對 TLS 指令埠所接收的指令啟用鑑別，請遵循 第 459 頁的『指令埠鑑別』中的步驟。

V 9.2.0 V 9.2.0 指令埠鑑別

MQIPT 可以配置為使用密碼來鑑別未受保護的指令埠及 TLS 指令埠所接收的指令。 使用此程序來啟用指令埠鑑別。

關於這項作業

當指令連接至已啟用指令埠鑑別之 MQIPT 實例的指令埠時， **mqiptAdmin** 指令會提示使用者輸入密碼。 MQIPT 會根據 MQIPT 配置中指定的存取密碼來驗證在 **mqiptAdmin** 指令中輸入的密碼。

您為指令埠鑑別設定的內容同時適用於 TLS 指令埠及未受保護的指令埠。

程序

1. 使用 **mqiptPW** 指令加密 MQIPT 存取密碼。

   執行下列指令：

   ```
   mqiptPW -sf encryption_key_file
   ```

其中 *encryption_key_file* 是包含 MQIPT 安裝之密碼加密金鑰的檔案名稱。 如果您的 MQIPT 安裝使用預設密碼加密金鑰，則不需要指定 **-sf** 參數。 在提示時鍵入要加密的存取密碼。

如需在 MQIPT 配置中加密密碼的相關資訊，請參閱 加密儲存的密碼。

2. 編輯 `mqipt.conf` 配置檔，並指定下列內容:

```
AccessPW=encrypted_password
RemoteCommandAuthentication=auth_setting
```

其中:

encrypted_password
是步驟 第 459 頁的『1』中 **mqiptPW** 指令所輸出的已加密密碼。

auth_setting
是鑑別需求。 如果此內容設為下列其中一個值，則會啟用指令埠鑑別:

選用
不需要密碼，但如果提供密碼，則密碼必須有效。 例如，在移轉期間，此選項可能很有用。

必要
必須隨指令埠所接收的每一個指令一起提供有效密碼。

如需這些內容的相關資訊，請參閱 MQIPT 廣域內容。

3. 啟動或重新整理 MQIPT，使變更生效。

MQIPT 會發出訊息，指出是否已啟用指令埠鑑別。 例如，如果 MQIPT 配置為每次執行 **mqiptAdmin** 指令時都需要輸入有效的密碼，則會發出下列訊息:

```
MQCPI021 Password checking has been enabled on the command port
```

製作備份

在一般備份程序中，您應該備份一些 MQIPT 檔案。

定期備份下列檔案:

- 配置檔 `mqipt.conf`
- 由 `mqipt.conf` 中的下列內容指定的 SSL/TLS 金鑰環檔:
  - **SSLClientKeyRing**
  - **SSLClientCAKeyRing**
  - **SSLServerKeyRing**
  - **SSLServerCAKeyRing**
  - V 9.2.0 V 9.2.0 **SSLCommandPortKeyRing**
- `mqipt.conf` 中由下列內容指定的 SSL/TLS 金鑰環密碼檔:
  - **SSLClientKeyRingPW**
  - **SSLClientCAKeyRingPW**
  - **SSLServerKeyRingPW**
  - **SSLServerCAKeyRingPW**
- V 9.2.0 密碼加密金鑰檔 (如果 MQIPT 配置包含使用預設金鑰以外的加密金鑰加密的密碼)。
- **SecurityManagerPolicy** 指定的原則檔 (如果已設定該內容)。
- `mqipt.conf` 中由下列內容指定的安全結束程式檔案及憑證結束程式檔案:
  - **SecurityExitName**
  - **SSLExitName**
- MQIPT 起始目錄的 `log` 子目錄中的連線日誌檔 (如果需要這些連線日誌檔來進行審核)。

效能調整

您可以使用執行緒儲存區與閒置逾時規格的組合，來調整每一個 MQIPT 路徑的相對效能。

連線執行緒

每一個 MQIPT 路徑都會指派一個工作儲存區，其中包含可處理送入通訊要求的並行執行中執行緒。 起始設定時，會建立執行緒儲存區 (大小為路徑的 `MinConnectionThreads` 屬性中指定的大小) ，並指派執行緒來處理第一個送入要求。 當此要求到達時，會指派另一個執行緒，備妥下一個送入要求。 當所有執行緒都指派給工作時，會建立新的執行緒、新增至工作儲存區，以及指派給工作。

以此方式，儲存區會一直成長，直到達到執行緒數目上限 (在 **MaxConnectionThreads** 中指定) 為止。 當交談結束或已經歷指定的閒置逾時期間時，執行緒會釋放回儲存區。 當達到工作執行緒數目上限時，下一個送入要求會等待執行緒釋放回工作儲存區。

您可以增加可用的執行緒數目，以減少要求可能必須等待的時間。 不過，您必須在此增加與可用的系統資源之間取得平衡。

閒置逾時值

依預設，工作中執行緒不會因閒置而終止。 當執行緒已指派給交談時，它會持續指派給該交談，直到正常關閉、取消啟動路徑或 MQIPT 關閉為止。 您可以選擇性地在 **IdleTimeout** 內容中指定閒置逾時間隔 (分鐘) ，以便重新啟動在指定時段內處於非作用中狀態的執行緒。 將執行緒放回工作儲存區，即可回收使用這些執行緒。

如果 IBM MQ 活動是間歇性，請將其活動訊號間隔設為小於 MQIPT 逾時的值，以便不會持續回收執行緒。

注意事項

本資訊係針對 IBM 在美國所提供之產品與服務所開發。

在其他國家中，IBM 可能不會提供本書中所提的各項產品、服務或功能。 請洽當地 IBM 業務代表，以取得當地目前提供的產品和服務之相關資訊。 這份文件在提及 IBM 的產品、程式或服務時，不表示或暗示只能使用 IBM 的產品、程式或服務。 只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。 不過，任何非 IBM 的產品、程式或服務，使用者必須自行負責作業的評估和驗證責任。

本文件所說明之主題內容，IBM 可能擁有其專利或專利申請案。 提供本文件不代表提供這些專利的授權。 您可以書面提出授權查詢，來函請寄到：

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

如果是有關雙位元組 (DBCS) 資訊的授權查詢，請洽詢所在國的 IBM 智慧財產部門，或書面提出授權查詢，來函請寄到：

智慧財產權授權
法務部與智慧財產權法律
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

下列段落不適用於英國，若與任何其他國家之法律條款抵觸，亦不適用於該國： International Business Machines Corporation 只依 "現況" 提供本出版品，不提供任何明示或默示之保證，其中包括且不限於不侵權、可商用性或特定目的之適用性的隱含保證。 有些地區在特定交易上，不允許排除明示或暗示的保證，因此，這項聲明不一定適合您。

這項資訊中可能會有技術上或排版印刷上的訛誤。 因此，IBM 會定期修訂；並將修訂後的內容納入新版中。 IBM 隨時會改進及/或變更本出版品所提及的產品及/或程式，不另行通知。

本資訊中任何對非 IBM 網站的敘述僅供參考，IBM 對該網站並不提供任何保證。 這些網站所提供的資料不是 IBM 本產品的資料內容，如果要使用這些網站的資料，您必須自行承擔風險。

IBM 得以各種適當的方式使用或散布由您提供的任何資訊，無需對您負責。

如果本程式的獲授權人為了 (i) 在個別建立的程式和其他程式（包括本程式）之間交換資訊，以及 (ii) 相互使用所交換的資訊，因而需要相關的資訊，請洽詢：

IBM Corporation
軟體交互作業能力協調程式，部門 49XA
3605 公路 52 N
Rochester， MN 55901
U.S.A.

在適當條款與條件之下，包括某些情況下（支付費用），或可使用此類資訊。

IBM 基於雙方之 IBM 客戶合約、IBM 國際程式授權合約或任何同等合約之條款，提供本資訊所提及的授權程式與其所有適用的授權資料。

本文件中所含的任何效能資料都是在受管制的環境下判定。 因此不同作業環境之下所得的結果，可能會有很大的差異。 有些測定已在開發階段系統上做過，不過這並不保證在一般系統上會出現相同結果。 甚至有部分的測量，是利用插補法而得的估計值， 實際結果可能有所不同。 本書的使用者應依自己的特定環境，查證適用的資料。

© Copyright IBM Corp. 2007, 2025

本文件所提及之非 IBM 產品資訊，取自產品的供應商，或其發佈的聲明或其他公開管道。 IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性或任何對產品的其他主張是否完全無誤。 有關非 IBM 產品的性能問題應直接洽詢該產品供應商。

有關 IBM 未來方針或目的之所有聲明，僅代表 IBM 的目標與主旨，隨時可能變更或撤銷，不必另行通知。

這份資訊含有日常商業運作所用的資料和報告範例。 為了要使它們儘可能完整，範例包括個人、公司、 品牌和產品的名稱。 這些名稱全屬虛構，如與實際公司的名稱和住址雷同，純屬巧合。

著作權授權:

本資訊含有原始語言之範例應用程式，用以說明各作業平台中之程式設計技術。 您可以基於研發、使用、銷售或散布符合作業平台（撰寫範例程式的作業平台）之應用程式介面的應用程式等目的，以任何形式複製、修改及散布這些範例程式，而不必向 IBM 付費。 這些範例並未在所 有情況下完整測試。 因此，IBM 不保證或暗示這些程式的可靠性、有用性或功能。

若 貴客戶正在閱讀本項資訊的電子檔，可能不會有照片和彩色說明。

程式設計介面資訊

程式設計介面資訊 (如果有提供的話) 旨在協助您建立與此程式搭配使用的應用軟體。

本書包含預期程式設計介面的相關資訊，可讓客戶撰寫程式以取得 WebSphere MQ 的服務。

不過，本資訊也可能包含診斷、修正和調整資訊。 提供診斷、修正和調整資訊，是要協助您進行應用軟體的除錯。

重要: 請勿使用此診斷、修改及調整資訊作為程式設計介面，因為它可能會變更。

商標

IBM、 IBM 標誌 ibm.com 是 IBM Corporation 在全球許多適用範圍的商標。 IBM 商標的最新清單可在 Web 的 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml 中找到。 其他產品和服務名稱，可能是 IBM 或其他公司的商標。

Microsoft 及 Windows 是 Microsoft Corporation 在美國及/或其他國家或地區的商標。

UNIX 是 The Open Group 在美國及/或其他國家/地區的註冊商標。

Linux 是 Linus Torvalds 在美國及/或其他國家或地區的註冊商標。

本產品包含 Eclipse Project (https://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: