9.3

IBM MQ 技術概觀

附註

使用本資訊及其支援的產品之前，請先閱讀第 255 頁的『注意事項』中的資訊。

目錄

IBM MQ 技術概觀

使用 IBM MQ 來連接您的應用程式，並管理跨組織的資訊配送。

IBM MQ 可讓程式使用一致的應用程式設計介面，透過不同元件 (處理器、作業系統、子系統及通訊協定) 的網路彼此通訊。 使用此介面設計及撰寫的應用程式稱為訊息佇列應用程式。

請使用下列子主題，以瞭解 IBM MQ 所提供的訊息佇列作業及其他特性。

相關概念

IBM MQ 簡介

在何處尋找產品需求及支援資訊

相關工作

規劃 IBM MQ 架構

相關參考

第 6 頁的『訊息佇列作業的主要特性及好處』

此資訊強調顯示訊息佇列作業的部分特性及好處。 它說明訊息佇列作業的安全及資料完整性等特性。

訊息佇列作業簡介

IBM MQ 產品可讓程式使用一致的應用程式設計介面，透過不同元件 (處理器、作業系統、子系統及通訊協定) 的網路彼此通訊。

使用此介面設計及撰寫的應用程式稱為 訊息佇列作業 應用程式，因為它們使用 傳訊 及 佇列作業 樣式:

- 傳訊表示程式透過在訊息中彼此傳送資料來進行通訊，而不是直接彼此呼叫。
- 佇列作業表示將訊息放在儲存體中的佇列上，可讓程式以不同的速度和時間，在不同的位置，彼此之間沒有邏輯連線，彼此獨立執行。

訊息佇列作業已在資料處理中使用多年。 目前最常用在電子郵件中。 在沒有排隊的情況下，長距離傳送電子訊息需要路徑上的每個節點都可以轉遞訊息，而且收件人會登入並意識到您嘗試傳送訊息給他們的事實。 在佇列作業系統中，訊息會儲存在中間節點，直到系統準備好轉遞它們為止。 在其最終目的地，它們會儲存在電子郵件信箱中，直到收件人準備好閱讀它們為止。

即便如此，今天仍有許多複雜的商業交易在沒有排隊的情況下處理。 在大型網路中，系統可能以現成可用的狀態維護數千條連線。 如果系統的某個部分發生問題，則系統的許多部分會變成無法使用。

您可以將訊息佇列作業視為程式的電子郵件。 在訊息佇列作業環境中，組成應用程式套組一部分的每一個程式都會執行完整定義且自行包含的函數，以回應特定要求。 若要與另一個程式通訊，程式必須將訊息放置在預先定義的佇列上。 另一個程式會從佇列中擷取訊息，並處理訊息中包含的要求及資訊。 因此訊息佇列作業是一種程式對程式通訊的樣式。

佇列作業是在應用程式準備好處理訊息之前用來保留訊息的機制。 佇列作業可讓您:

- 在程式之間進行通訊 (每個程式可能在不同環境中執行) ，而不需要撰寫通訊程式碼。
- 選取程式處理訊息的順序。
- 當訊息數超出臨界值時，安排多個程式來處理佇列，以平衡系統上的負載。
- 如果主要系統無法使用，請安排替代系統來處理佇列，以增加應用程式的可用性。

何謂訊息佇列?

訊息佇列 (簡單稱為佇列) 是訊息可以傳送至其中的具名目的地。 訊息會累積在佇列上，直到服務那些佇列的程式擷取這些訊息為止。

佇列位於佇列管理程式中，並由佇列管理程式管理 (請參閱 第 8 頁的『訊息佇列作業術語』)。 佇列的實體本質取決於執行佇列管理程式的作業系統。 佇列可以是電腦記憶體中的暫時緩衝區，或是永久儲存裝置 (例如磁碟) 上的資料集。 佇列的實體管理是佇列管理程式的責任，對於參與的應用程式並不明顯。

程式只會透過佇列管理程式的外部服務來存取佇列。 他們可以開啟佇列、在佇列上放置訊息、從中取得訊息，以及關閉佇列。 他們也可以設定及查詢佇列的屬性。

訊息佇列作業的不同樣式

點對點 (point-to-point)

一個訊息會放置在佇列上，且一個應用程式會接收該訊息。

在點對點傳訊中，傳送端應用程式必須知道接收端應用程式的相關資訊，才能將訊息傳送至該應用程式。 例如，傳送端應用程式可能需要知道要將資訊傳送至其中的佇列名稱，也可能指定佇列管理程式名稱。

發佈/訂閱

發佈應用程式發佈的每一個訊息的副本會遞送至每一個感興趣的應用程式。 可能有許多、一個或沒有感興趣的應用程式。 在發佈/訂閱中，有興趣的應用程式稱為訂閱者，且訊息會排入訂閱所識別的佇列中。

發佈/訂閱傳訊可讓您將資訊提供者與該資訊的消費者取消連結。 傳送端應用程式和接收端應用程式不需要彼此瞭解多少，即可傳送和接收資訊。 如需相關資訊，請參閱第 51 頁的『發佈/訂閱傳訊』。

應用程式設計者和開發人員的訊息佇列作業好處

IBM MQ 容許應用程式使用 訊息佇列作業 來參與訊息驅動處理。 應用程式可以使用適當的訊息佇列軟體產品，在不同平台之間進行通訊。 例如， z/OS 應用程式可以透過 IBM MQ for z/OS 進行通訊。 應用程式與基礎通訊的機制隔離。 訊息佇列作業的部分其他好處如下:

- 您可以使用可在許多應用程式之間共用的小型程式來設計應用程式。
- 您可以重複使用這些建置區塊來快速建置新的應用程式。
- 寫入以使用訊息佇列作業技術的應用程式，不會受到佇列管理程式運作方式的變更影響。
- 您不需要使用任何通訊協定。 佇列管理程式會為您處理通訊的所有層面。
- 接收訊息的程式在傳送訊息時不需要執行。 訊息會保留在佇列上。

設計人員可以降低其應用程式的成本，因為開發速度更快，需要的開發人員較少，且對程式設計技能的需求低於不使用訊息佇列作業的應用程式。

每當應用程式執行時， IBM MQ 都會實作一個稱為 訊息佇列介面 (或 MQI) 的一般應用程式設計介面。 這可讓您更容易將應用程式從一個平台移植到另一個平台。

如需 MQI 的詳細資料，請參閱 訊息佇列介面概觀。

訊息佇列作業的主要特性及好處

此資訊強調顯示訊息佇列作業的部分特性及好處。 它說明訊息佇列作業的安全及資料完整性等特性。

使用訊息佇列作業技術之應用程式的主要特性如下:

- 第 7 頁的『程式之間沒有直接連線』
- 第 7 頁的『與時間無關的通訊』
- 第 7 頁的『小型程式』
- 第 7 頁的『訊息驅動處理』
- 第 7 頁的『事件驅動處理』
- 第 8 頁的『訊息優先順序』
- 第 8 頁的『安全』
- 第 8 頁的『資料完整性』
- 第 8 頁的『回復支援』

註: 在考量 IBM MQ 用戶端及伺服器時，您不需要變更伺服器應用程式，即可在新平台上支援其他 IBM MQ MQI clients 。 同樣地， IBM MQ MQI client 可以在沒有變更的情況下使用其他類型的伺服器來運作。

程式之間沒有直接連線

訊息佇列作業是一種間接程式對程式通訊的技術。 它可以在程式彼此通訊的任何應用程式內使用。 通訊是透過一個程式將訊息放入佇列 (由佇列管理程式所擁有)，以及另一個程式從佇列取得訊息來進行。

程式可以取得由其他程式放入佇列的訊息。 其他程式可以連接至與接收端程式相同的佇列管理程式，或另一個佇列管理程式。 這個其他佇列管理程式可能位於另一個系統、不同的電腦系統，甚至在不同的商業或企業內。

使用訊息佇列進行通訊的程式之間沒有實體連線。 程式會將訊息傳送至佇列管理程式所擁有的佇列，而另一個程式會從佇列中擷取訊息 (請參閱 第 7 頁的圖 1)。

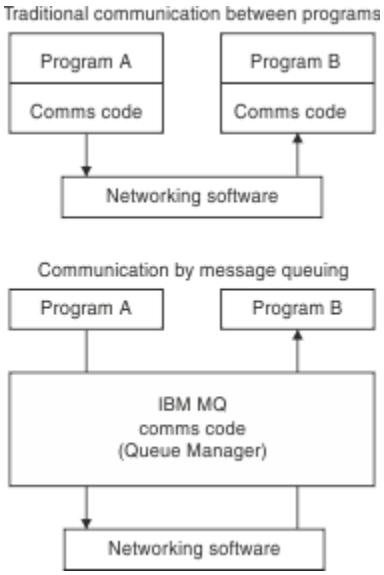


圖 *1:* 與傳統通訊相比的訊息佇列作業

如同電子郵件，屬於交易一部分的個別訊息會透過儲存及轉寄上的網路傳送。基礎。 如果節點之間的鏈結失敗，則會保留訊息，直到還原鏈結或操作員或程式重新導向訊息為止。

在程式中隱藏訊息從佇列移至佇列的機制。 因此程式更簡單。

與時間無關的通訊

要求他人執行工作的程式不必等待要求的回覆。 他們可以執行其他工作，並在回覆到達時或稍後處理回覆。撰寫傳訊應用程式時，您不需要知道 (或關注) 程式何時傳送訊息，或目標何時能夠接收訊息。 訊息不會遺失; 它會由佇列管理程式保留，直到目標準備好處理它為止。 訊息會一直留在佇列中，直到程式移除為止。這表示傳送和接收應用程式已取消連結; 傳送端可以繼續處理，而不等待接收端確認接收訊息。 當傳送訊息時，目標應用程式甚至不必在執行中。 它可以在啟動之後擷取訊息。

小型程式

訊息佇列作業可讓您使用小型自行包含程式的優點。 您可以將工作分散在數個較小且獨立的程式中，而不是單一大型程式循序執行工作的所有部分。 要求程式會傳送訊息至每一個個別程式，要求它們執行其功能; 當每一個程式完成時，結果會以一或多個訊息傳回。

訊息驅動處理

當訊息抵達佇列時，它們可以使用 觸發來自動啟動應用程式。 必要的話，可以在處理訊息時停止應用程式。

事件驅動處理

您可以根據佇列的狀態來控制程式。 例如，您可以安排在訊息到達佇列時立即啟動程式，或者您可以指定在佇列上具有高於特定優先順序的 10 則訊息或佇列上具有任何優先順序的 10 則訊息之前不會啟動程式。

訊息優先順序

當程式將訊息放入佇列時，可以指派訊息的優先順序。 這會決定在佇列中新增訊息的位置。

程式可以依照訊息在佇列中的順序，或取得特定訊息，從佇列中取得訊息。 (如果程式正在尋找先前傳送之要求的回覆，則可能想要取得特定訊息。)

安全

提供安全機能，包括應用程式使用佇列管理程式時的鑑別、使用資源 (例如佇列管理程式上的佇列) 時的授權檢查，以及訊息資料在透過網路傳送及位於佇列上時的加密。 如需安全的相關資訊，請參閱 安全概觀。

資料完整性

資料完整性由工作單元提供。 在每一個 MQGET 或 MQPUT 上，完全支援工作單元開始及結束的同步化，以容許確定或回復工作單元的結果。 同步點支援在內部或外部對 IBM MQ 運作，視針對應用程式選取的同步點協調形式而定。

回復支援

為了能夠進行回復，會記載所有持續性 IBM MQ 更新項目。 如果需要回復，則會還原所有持續訊息，回復所有進行中交易，並以控制同步點管理程式的正常方式處理任何同步點確定及取消。 如需持續訊息的相關資訊，請參閱 訊息持續性。

訊息佇列作業術語

此資訊可讓您深入瞭解訊息佇列作業中使用的部分術語。

其中包括:

- 通道
- 叢集
- IBM MQ MQI client
- z/OS 內部群組佇列
- 訊息
- 訊息通道代理程式
- 訊息描述子
- 點對點 (point-to-point)
- 發佈/訂閱 (Publish/Subscrib)
- 佇列
- 佇列管理程式
- z/OS 佇列共用群組
- z/OS 共用佇列
- 訂閱
- 主題

通道

通道用來將訊息從一個佇列管理程式移至另一個佇列管理程式，它們會保護應用程式免受基礎通訊協定的影響。 佇列管理程式可能存在於相同系統上，或相同平台上或不同平台上的不同系統上。 傳送的訊息可以來自許多地方:

- 使用者撰寫的應用程式，可將資料從一個節點傳送至另一個節點。
- 使用 PCF 指令或 MQAI 的使用者撰寫管理應用程式。

- IBM MQ Explorer。
- 將檢測事件訊息傳送至另一個佇列管理程式的佇列管理程式。
- 將遠端管理指令傳送至另一個佇列管理程式的佇列管理程式。 例如，使用 MQSC 指令或 administrative REST API。

如需通道的相關資訊，請參閱 第 26 頁的『通道定義』。

叢集

叢集 是邏輯上以某種方式關聯的佇列管理程式網路。

在使用不含叢集作業的分散式佇列的 IBM MQ 網路中，每個佇列管理程式都是獨立的。 如果一個佇列管理程式需要將訊息傳送至另一個佇列管理程式，它必須已定義傳輸佇列及遠端佇列管理程式的通道。

使用叢集有兩個不同的原因: 減少系統管理，以及改善可用性和工作量平衡。

只要您建立甚至最小的叢集，就可以從簡化的系統管理中獲益。 屬於叢集的佇列管理程式需要較少的定義，因此會減少在定義中產生錯誤的風險。

如需叢集作業的相關資訊，請參閱 叢集。

IBM MQ MQI client

IBM MQ MQI 用戶端 是 IBM MQ 的可獨立安裝元件。 MQI 用戶端可讓您以通訊協定執行 IBM MQ 應用程式，與其他平台上的一個以上「訊息佇列介面 (MQI)」伺服器互動，並連接至其佇列管理程式。

如需如何安裝及使用 IBM MQ MQI client 元件的完整資料，請參閱下列主題:

- AIX 在 AIX 上安裝 IBM MQ 用戶端
- Linux 在 Linux® 上安裝 IBM MQ 用戶端
- Windows 在 Windows 上安裝 IBM MQ 用戶端
- IBM i 在 IBM i 上安裝 IBM MQ 用戶端

及 配置伺服器與用戶端之間的連線。

內部群組佇列

z/OS

佇列共用群組中的佇列管理程式可以使用一般通道進行通訊，或者您可以使用稱為 群組內佇列作業 (IGQ) 的技術，它可讓您執行快速訊息傳送，而無需定義通道。 這僅適用於 IBM MQ for z/OS。

如需內部群組佇列作業的相關資訊，請參閱 第 176 頁的『內部群組佇列』。

訊息

在訊息佇列作業中，訊息是由一個程式所傳送且預期用於另一個程式的資料集合。 請參閱 IBM MQ 訊息。

如需訊息類型的相關資訊，請參閱 訊息類型。

訊息通道代理程式

訊息通道代理程式是通道的一端。 一對訊息通道代理程式 (一個傳送和一個接收) 組成通道，並將訊息從一個佇列管理程式移至另一個佇列管理程式。

如需如何使用訊息通道代理程式的相關資訊，請參閱 分散式佇列管理簡介。

訊息描述子

IBM MQ 訊息包含控制資訊和應用程式資料。

控制資訊定義在訊息描述子結構 (MQMD) 中，並包含如下內容:

- 訊息類型
- 訊息的 ID
- 遞送訊息的優先順序

應用程式資料的結構及內容由參與程式決定，而不是由 IBM MQ 決定。

如需相關資訊，請參閱 MQMD。

點對點傳訊

在點對點傳訊中，每一個訊息會從一個生產端應用程式傳送至一個消費端應用程式。 訊息會透過生產端應用程式傳送，並將訊息放到佇列中，而消費端應用程式會從該佇列中取得訊息。

發佈/訂閱傳訊

在發佈/訂閱傳訊中，發佈應用程式所發佈的每一個訊息的副本會遞送給每一個感興趣的應用程式。 可能有許多、一個或沒有感興趣的應用程式。 在發佈/訂閱中，有興趣的應用程式稱為訂閱者，且訊息會排入訂閱所識別的佇列中。

如需相關資訊，請參閱 第 51 頁的『發佈/訂閱傳訊』。

佇列

可將訊息傳送至其中的具名目的地。 訊息會累積在佇列上，直到服務那些佇列的程式擷取這些訊息為止。

如需相關資訊，請參閱 第 16 頁的『佇列』。

佇列管理程式

佇列管理程式 是向應用程式提供佇列作業服務的系統程式。

它提供應用程式設計介面，讓程式可以將訊息放置在佇列上，以及從佇列取得訊息。 佇列管理程式提供其他功能，讓管理者可以建立新的佇列、變更現有佇列的內容，以及控制佇列管理程式的作業。

若要在系統上提供 IBM MQ 訊息佇列作業服務，必須有佇列管理程式在執行中。 您可以在單一系統上執行多個佇列管理程式 (例如，將測試系統與 即時 系統分開)。 對於應用程式，每一個佇列管理程式都由 連線控點 (*Hconn*) 來識別。

許多不同的應用程式可以同時使用佇列管理程式的服務，這些應用程式可以完全不相關。 若要讓程式使用佇列管理程式的服務，它必須建立與該佇列管理程式的連線。

若要讓應用程式將訊息傳送至連接至其他佇列管理程式的應用程式，佇列管理程式必須能夠彼此通訊。 IBM MQ 實作 儲存及轉遞 通訊協定，以確保在這類應用程式之間安全遞送訊息。

如需相關資訊，請參閱 第 23 頁的『佇列管理程式』。

佇列共用群組

z/OS

可以存取同一組共用佇列的佇列管理程式會形成稱為 佇列共用群組 (QSG) 的群組。 它們透過儲存共用佇列的連結機能 (CF) 彼此通訊。 這僅適用於 IBM MQ for z/OS。

如需相關資訊，請參閱 第 139 頁的『共用佇列及佇列共用群組』。

共用佇列

z/OS

共用佇列 是具有訊息的本端佇列類型，可由 Sysplex 中的一或多個佇列管理程式存取。 這與使用相同佇列管理程式的多個應用程式所共用的佇列不同。 這僅適用於 IBM MQ for z/OS。

訂閱

發佈/訂閱應用程式可以在特定主題的相關訊息中登錄興趣。 當應用程式執行此動作時，稱為訂閱者，且術語訂閱定義如何將相符訊息排入佇列以進行處理。

訂閱包含訂閱者身分及要放置發佈資訊之目的地佇列身分的相關資訊。 它也包含如何將發佈資訊放置在目的地佇列上的相關資訊。

如需相關資訊，請參閱 第 54 頁的『訂閱者和訂閱』。

主題

主題是說明發佈/訂閱訊息中已發佈之資訊主旨的字串。

主題是在發佈/訂閱系統中順利遞送訊息的關鍵。 發佈者不在每一個訊息中包含特定的目的地位址，而是指派主題給每一個訊息。 佇列管理程式使主題符合已訂閱至該主題的訂閱者清單，並將訊息遞送至其中每一個訂閱者。

如需相關資訊，請參閱 第 56 頁的『主題』。

訊息和佇列

訊息和佇列是訊息佇列系統的基本元件。

何謂訊息?

訊息 是對使用它的應用程式有意義的位元組字串。 訊息是用來將資訊從一個應用程式傳送至另一個應用程式 (或在相同應用程式的不同部分之間)。 應用程式可以在相同平台上執行，也可以在不同平台上執行。

IBM MQ 訊息包含:

- 應用程式資料。應用程式資料的內容和結構由使用它的應用程式定義。
- 訊息描述子。訊息描述子會識別訊息並包含其他控制資訊，例如訊息類型及傳送應用程式指派給訊息的優先順序。

  訊息描述子的格式由 IBM MQ 定義。 如需訊息描述子的完整說明，請參閱 MQMD-訊息描述子。
- 訊息內容。 訊息的相關 meta 資料。 訊息內容的內容由使用它們的應用程式定義。 如需相關資訊，請參閱訊息內容。

訊息長度

預設訊息長度上限是 4 MB，不過您可以將它增加至長度上限 100 MB (其中 1 MB 等於 1 048 576 個位元組)。 實際上，訊息長度可能受到下列限制:

- 定義給接收佇列的訊息長度上限
- 定義給佇列管理程式的訊息長度上限
- 佇列定義的訊息長度上限
- 傳送或接收應用程式所定義的訊息長度上限
- 訊息可用的儲存體數量

可能需要數則訊息才能傳送應用程式所需的所有資訊。

應用程式如何傳送及接收訊息?

應用程式會使用 **MQI 呼叫**來傳送及接收訊息。

例如，若要將訊息放入佇列，應用程式:

1. 發出 MQI MQOPEN 呼叫來開啟必要的佇列
2. 發出 MQI MQPUT 呼叫，將訊息放入佇列中

另一個應用程式可以透過發出 MQI MQGET 呼叫，從相同佇列擷取訊息。

如需 MQI 呼叫的相關資訊，請參閱 MQI 呼叫。

何謂佇列?

佇列 是用來儲存訊息的資料結構。

每一個佇列都是由 佇列管理程式所擁有。 佇列管理程式負責維護所擁有的佇列，並將其接收的所有訊息儲存至適當的佇列中。 應用程式或佇列管理程式可能會將訊息放置在佇列中，作為其正常作業的一部分。

預先定義的佇列及動態佇列

佇列可以用其建立方式來描述:

- **預先定義佇列** 由管理者使用適當的 MQSC 或 PCF 指令建立。預先定義的佇列是永久的; 它們獨立於使用它們的應用程式而存在，且在 IBM MQ 重新啟動之後仍然存在。
- 當應用程式建立時，會建立 **動態佇列** 發出指定 模型佇列名稱的 MQOPEN 要求。 所建立的佇列是根據 範本佇列定義(稱為模型佇列)。您可以使用 MQSC 指令 DEFINE QMODEL 來建立模型佇列。 模型佇列的屬性(例如，可以儲存在它上的訊息數上限) 由從它建立的任何動態佇列繼承。

  模型佇列具有一個屬性，可指定動態佇列是永久還是暫時。 永久佇列會在應用程式及佇列管理程式重新啟動後繼續存在; 暫時佇列會在重新啟動時遺失。

從佇列擷取訊息

適當授權的應用程式可以根據下列擷取演算法，從佇列中擷取訊息:

- 先進先出 (FIFO)。
- 訊息優先順序，如訊息描述子中所定義。 以 FIFO 為基準擷取具有相同優先順序的訊息。
- 特定訊息的程式要求。

來自應用程式的 MQGET 要求會決定使用的方法。

IBM MQ 物件

佇列管理程式會定義 IBM MQ 物件的內容。 這些內容的值會影響 IBM MQ 處理這些物件的方式。 您可以使用 IBM MQ 指令及介面來建立及管理物件。 從應用程式中，您可以使用「訊息佇列介面 (MQI)」來控制物件。 從程式定址時，由 IBM MQ 物件描述子 (MQOD) 識別物件。

物件管理

物件管理包括下列作業:

- 啟動和停止佇列管理程式。
- 建立應用程式的物件，特別是佇列。
- 顯示或變更物件的屬性。
- 正在刪除物件。
- 使用通道來建立通往其他 (遠端) 系統上佇列管理程式的通訊路徑。
- 建立佇列管理程式的 叢集 ， 以簡化整體管理程序及平衡工作量。

除了動態佇列之外，必須先在佇列管理程式中定義物件，然後才能使用它們。

當您使用 IBM MQ 指令來執行物件管理作業時，佇列管理程式會檢查您是否具有執行作業所需的權限層次。同樣地，當應用程式使用 MQOPEN 呼叫來開啟物件時，佇列管理程式會先檢查應用程式是否具有必要的權限層次，然後才容許存取該物件。 會對正在開啟的物件名稱進行檢查。

您可以使用下列方法來定義及管理物件:

- 可程式化指令格式參照 及 自動化管理作業 中說明的 PCF 指令
- MQSC 指令 中說明的 MQSC 指令

- z/OS IBM MQ for z/OS 作業及控制台，如 管理 IBM MQ for z/OS 中所述
- Windows Linux IBM MQ Explorer (僅限 Intel 系統的 Windows 和 Linux)。 如需相關資訊，請參閱 MQ 探險家簡介。

您也可以使用下列方法來管理物件:

- 控制指令，從鍵盤輸入。 請參閱 使用控制指令管理 IBM MQ for Multiplatforms。
- 程式中的 IBM MQ 管理介面 (MQAI) 呼叫。 請參閱 IBM MQ 管理介面 (MQAI)。

ALW 對於 AIX, Linux, and Windows 上 IBM MQ 指令的順序，您可以使用 MQSC 機能來執行檔案中保留的一系列指令。 如需相關資訊，請參閱 使用 MQSC 指令管理 IBM MQ。

IBM i 對於您定期使用的 IBM MQ for IBM i 指令序列，您可以撰寫 CL 程式。 如需相關資訊，請參閱 使用 CL 指令來管理 IBM MQ for IBM i。

z/OS 對於您定期使用的 IBM MQ for z/OS 指令序列，您可以撰寫管理程式來建立包含指令的訊息，並將這些訊息放置在系統指令輸入佇列上。 佇列管理程式處理此佇列上的訊息的方式，與它處理從指令行或從作業及控制台輸入之指令的方式相同。 此技術在 撰寫程式以管理 IBM MQ 中有說明，並在 IBM MQ for z/OS 隨附的「郵件管理程式」範例應用程式中示範。 如需此範例的說明，請參閱 IBM MQ for z/OS 的程式範例 。

物件屬性

物件的內容由其屬性定義。 有些您可以指定，有些您只能檢視。

例如，佇列可容納的訊息長度上限由其 **MaxMsgLength** 屬性定義; 您可以在建立佇列時指定此屬性。 **DefinitionType** 屬性指定如何建立佇列; 您只能顯示此屬性。

在 IBM MQ 中，有兩種參照屬性的方式:

- 使用其 PCF 名稱，例如 **MaxMsgLength**。
- 使用其 MQSC 指令名稱，例如 MAXMSGL。

佇列共用群組

z/OS

可以存取同一組共用佇列的佇列管理程式會形成稱為 佇列共用群組 (QSG) 的群組，並使用儲存共用佇列的連結機能 (CF) 彼此通訊。 請注意， QSG 不是嚴格的物件。

共用佇列是具有訊息的本端佇列類型，可由佇列共用群組中的一或多個佇列管理程式存取。 這與使用相同佇列管理程式由多個應用程式共用的佇列不同。

佇列共用群組有一個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。

重要: 只有在 IBM MQ for z/OS 上才支援共用佇列及佇列共用群組。

如需相關資訊，請參閱 第 139 頁的『共用佇列及佇列共用群組』 。

系統預設物件

系統預設物件 是每當建立佇列管理程式時自動建立的一組物件定義。 您可以複製並修改任何這些物件定義，以在安裝時用於應用程式中。

預設物件名稱具有詞幹 SYSTEM; 例如，預設本端佇列為 SYSTEM.DEFAULT.LOCAL.QUEUE，且預設接收端通道是 SYSTEM.DEF.RECEIVER。 您無法重新命名這些物件; 需要這些名稱的預設物件。

當您定義物件時，會從適當的預設物件複製您未明確指定的任何屬性。 例如，如果您定義本端佇列，則您未指定的那些屬性會從預設佇列 SYSTEM.DEFAULT.LOCAL.QUEUE。

如需相關資訊，請參閱 系統及預設物件 。

物件類型

許多管理作業涉及操作各種類型的 IBM MQ 物件。

如需命名 IBM MQ 物件的相關資訊，請參閱 第 29 頁的『命名 IBM MQ 物件』。

如需在佇列管理程式上建立之預設物件的相關資訊，請參閱 第 13 頁的『系統預設物件』。

如需不同類型 IBM MQ 物件的相關資訊，請參閱下列各項:

鑑別資訊物件

鑑別資訊物件提供執行憑證撤銷檢查所需的定義。

佇列管理程式鑑別資訊物件構成「傳輸層安全 (TLS)」的 IBM MQ 支援的一部分。 它提供檢查已撤銷憑證所需的定義。 憑證管理中心會撤銷無法再信任的憑證。

您可以使用 MQSC 指令 **DEFINE AUTHINFO** 來定義鑑別資訊物件。 如需鑑別資訊物件的屬性相關資訊，請參閱 **DEFINE AUTHINFO**。

您可以搭配使用下列 IBM MQ 控制指令與鑑別資訊物件:

- **setmqaut** (授與或撤銷權限)
- **dspmqaut** (顯示物件授權)
- **dmpmqaut** (傾出授權)
- **rcrmqobj** (重建物件)
- **rcdmqimg** (記錄媒體影像)
- **dspmqfls** (顯示檔名)

如需 TLS 及使用鑑別資訊物件的概觀，請參閱 傳輸層安全 (TLS) 概念 及 IBM MQ 中的 TLS 安全通訊協定。

通道

通道是提供從一個佇列管理程式到另一個佇列管理程式之通訊路徑的物件。

如需相關資訊，請參閱 第 24 頁的『通道』。

通訊資訊物件

IBM MQ 多重播送提供低延遲、高扇出、可靠的多重播送傳訊功能。 需要通訊資訊 (COMMINFO) 物件，才能使用「多重播送」傳輸。

如需相關資訊，請參閱 第 87 頁的『IBM MQ 多重播送』。

COMMINFO 物件是包含與多重播送傳輸相關聯之屬性的 IBM MQ 物件。 如需這些屬性的相關資訊，請參閱 DEFINE COMMINFO。 如需建立 COMMINFO 物件的相關資訊，請參閱 開始使用多重播送。

接聽器

接聽器 是接受來自其他佇列管理程式或用戶端應用程式的網路要求，並啟動相關聯通道的處理程序。

可以使用 **runmqlsr** 控制指令來啟動 接聽器處理程序 。

接聽器物件 是 IBM MQ 物件，可讓您從佇列管理程式範圍內管理接聽器處理程序的啟動和停止。 透過定義接聽器物件的屬性，您可以執行下列動作:

- 配置接聽器程序。
- 指定當佇列管理程式啟動和停止時，接聽器處理程序是否自動啟動和停止。

重要: z/OS IBM MQ for z/OS 不支援接聽器物件。 如需 IBM MQ for z/OS 如何使用通道起始程式來實作接聽的相關資訊，請參閱 第 135 頁的『z/OS 上的通道起始程式』。

名單

名單 是一個 IBM MQ 物件，包含叢集名稱、佇列名稱或鑑別資訊物件名稱的清單。 在叢集中，它可以用來識別佇列管理程式為其保留儲存庫的叢集清單。

名稱清單是包含其他 IBM MQ 物件清單的 IBM MQ 物件。 通常名稱清單是提供給諸如觸發監視器之類的應用程式，以用來識別佇列的群組。 使用名單的優點是獨立於應用程式進行維護; 它可以在不停止任何使用它的應用程式的情況下進行更新。 此外，如果一個應用程式失敗，則名單不受影響，其他應用程式可以繼續使用它。

名稱清單也會與佇列管理程式叢集搭配使用，以維護多個 IBM MQ 物件所參照的叢集清單。

您可以使用 MQSC 指令 DEFINE NAMELIST 及 ALTER NAMELIST 來定義及修改名稱清單。

註: z/OS 或者，在 z/OS 上，您可以使用 IBM MQ for z/OS 作業及控制面板

程式可以使用 MQI 來找出這些名稱清單中包含的佇列。 名稱清單的組織是應用程式設計者和系統管理者的責任。

如需可用的名稱清單屬性清單，請參閱 名稱清單的屬性。

程序定義

程序定義物件可讓應用程式在不需要操作員介入的情況下啟動，方法是定義應用程式的屬性以供佇列管理程式使用。

程序定義物件定義應用程式，該應用程式啟動以回應 IBM MQ 佇列管理程式上的觸發事件。 程序定義屬性包括應用程式 ID、應用程式類型，以及應用程式特定的資料。 如需相關資訊，請參閱 第 22 頁的『IBM MQ 用於特定目的的佇列』中的 起始佇列 。

若要容許在不需要操作員介入的情況下啟動應用程式 (如 使用觸發程式啟動 IBM MQ 應用程式中所述) ，佇列管理程式必須知道應用程式的屬性。 這些屬性定義在 程序定義物件中。

建立物件時， **ProcessName** 屬性是固定的。 不過，您可以使用 IBM MQ 指令來變更其他屬性。

註: z/OS 或者，在 z/OS 上，您可以使用 IBM MQ for z/OS 作業及控制台。

您可以使用 MQINQ-查詢物件屬性來查詢所有屬性的值。

如需可用的程序定義屬性清單，請參閱 程序定義的屬性。

佇列

IBM MQ 佇列 是應用程式可以放置訊息的具名物件，以及應用程式可以從中取得訊息的具名物件。

如需相關資訊，請參閱 第 16 頁的『佇列』。

佇列管理程式

IBM MQ 佇列管理程式向應用程式提供佇列作業服務，並管理屬於它們的佇列。

如需相關資訊，請參閱 第 23 頁的『佇列管理程式』。

服務

服務 物件是定義當佇列管理程式啟動或停止時要執行之程式的一種方式。

程式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將執行的程式定義。 伺服器處理程序只能同時執行一個實例。 執行時，可以使用 MQSC 指令 DISPLAY SVSTATUS 來監視伺服器處理程序的狀態。 一般而言，伺服器服務物件是程式的定義，例如無法傳送郵件的處理程式或觸發監視器，不過，可執行的程式不限於 IBM MQ 所提供的程式。 此外，可以定義伺服器服務物件，以包括在關閉指定的佇列管理程式以結束程式時將執行的指令。

指令

指令 是將參數 SERVTYPE 指定為 COMMAND 的服務物件。 指令服務物件是將在啟動或停止指定的佇列管理程式時執行之程式的定義。 一個指令處理程序的多個實例可以同時執行。 指令服務物件不同於伺服器服務物件，因為一旦執行程式，佇列管理程式就不會監視程式。 通常指令服務物件是短暫存活且將執行特定作業 (例如啟動一或多個其他作業) 的程式定義。

重要: z/OS IBM MQ for z/OS 不支援服務物件。

如需相關資訊，請參閱 使用服務。

儲存類別

z/OS

儲存類別將一或多個佇列對映至某頁集。

這表示該佇列的訊息會儲存在該頁集上 (受限於緩衝)。

儲存類別僅在 IBM MQ for z/OS 上受支援。

如需儲存類別的進一步相關資訊，請參閱 在 z/OS 上規劃 IBM MQ 環境。

主題物件

主題物件 是一個 IBM MQ 物件，可讓您將特定非預設屬性指派給主題。

主題 是由發佈或訂閱特定 主題字串的應用程式所定義。 主題字串可以用正斜線字元 (/) 來區隔主題，以指定主題的階層。 這可以透過 主題樹狀結構來視覺化。 例如，如果應用程式發佈至主題字串 `/Sport/American Football` 和 `/Sport/Soccer`，則會建立主題樹狀結構，其母節點 `Sport` 具有兩個子節點 `American Football` 和 `Soccer`。

主題會從在其主題樹狀結構中找到的第一個母項管理節點繼承其屬性。 如果特定主題樹狀結構中沒有管理主題節點，則所有主題都會從基本主題物件 SYSTEM.BASE.TOPIC。

您可以在主題樹狀結構的任何節點上建立主題物件，方法是在主題物件的 TOPICSTR 屬性中指定該節點的主題字串。 您也可以定義管理主題節點的其他屬性。 如需這些屬性的相關資訊，請參閱 MQSC 指令或 使用 PCF 指令自動化管理。 依預設，每一個主題物件會從其最接近的上層管理主題節點繼承其屬性。

主題物件也可以用來向應用程式開發者隱藏完整主題樹狀結構。 如果針對主題 `/Sport/American Football` 建立名為 `FOOTBALL.US` 的主題物件，則應用程式可以發佈或訂閱名為 `FOOTBALL.US` 的物件，而不是具有相同結果的字串 `/Sport/American Football` 。

如果在主題物件的主題字串內輸入 #、+、/或 * 字元，則會將該字元視為字串內的一般字元，並視為與主題物件相關聯之主題字串的一部分。

如需主題物件的相關資訊，請參閱 第 51 頁的『發佈/訂閱傳訊』。

相關概念

第 5 頁的『訊息佇列作業簡介』
IBM MQ 產品可讓程式使用一致的應用程式設計介面，透過不同元件 (處理器、作業系統、子系統及通訊協定) 的網路彼此通訊。

相關參考

MQSC 指令

佇列

IBM MQ 佇列及佇列屬性簡介。

訊息儲存在佇列中，因此如果放置應用程式預期會回覆其訊息，則可以在等待該回覆時執行其他工作。 應用程式會使用「訊息佇列介面 (MQI)」來存取佇列，如 「訊息佇列介面」概觀中所述。

必須已建立佇列，才能將訊息放置在佇列上。 佇列是由佇列管理程式所擁有，且該佇列管理程式可以擁有許多佇列。 不過，每一個佇列的名稱在該佇列管理程式中必須是唯一的。

佇列是透過佇列管理程式來維護。 在大部分情況下，每一個佇列都由其佇列管理程式實際管理，但應用程式並不清楚這一點。 IBM MQ for z/OS 共用佇列可以由佇列共用群組中的任何佇列管理程式來管理。

若要建立佇列，您可以使用 IBM MQ 指令 (MQSC)、PCF 指令或平台專用介面。 例如， IBM MQ for z/OS 作業及控制面板是平台專用的。

您可以從應用程式 動態 建立暫時工作的本端佇列。 例如，您可以建立 *reply-to* 佇列 (在應用程式結束之後不需要)。 如需相關資訊，請參閱 第 21 頁的『動態和模型佇列』。

在使用佇列之前，您必須先開啟佇列，並指定您要如何處理它。 例如，您可以開啟下列項目的佇列:

- 僅瀏覽訊息 (不擷取訊息)
- 擷取訊息 (並與其他程式共用存取權，或具有獨佔性存取權)
- 將訊息放置在佇列上
- 查詢佇列的屬性
- 設定佇列的屬性

如需開啟佇列時可以指定之選項的完整清單，請參閱 MQOPEN-開啟物件。

佇列的屬性

當定義佇列時，會指定佇列的部分屬性，且之後無法變更 (例如，佇列類型)。 佇列的其他屬性可以分組成可變更的屬性:

- 由佇列管理程式在處理佇列期間 (例如，佇列的現行深度)
- 僅透過指令 (例如，佇列的文字說明)
- 由應用程式使用 MQSET 呼叫 (例如，是否容許在佇列上放置作業)

您可以使用 MQINQ 呼叫來尋找所有屬性的值。

多種佇列類型共用的屬性如下:

完整名稱
佇列的名稱。

QTYPE
佇列的類型。

QDesc
佇列的文字說明。

InhibitGet
是否容許程式從佇列取得訊息。 不過，您永遠無法從遠端佇列取得訊息。

InhibitPut
是否容許程式將訊息放置在佇列上。

DefPriority
放置在佇列上之訊息的預設優先順序。

DefPersistence
放置在佇列上之訊息的預設持續性

範圍
控制此佇列的項目是否也存在於名稱服務中。

z/OS z/OS 不支援 **Scope** 屬性

如需這些屬性的完整說明，請參閱 佇列的屬性。

定義佇列的方法

您可以使用 MQSC DEFINE 指令或 PCF 建立佇列 指令，對 IBM MQ 定義佇列。 指令指定佇列類型及其屬性。 例如，本端佇列物件具有屬性可指定當應用程式在 MQI 呼叫中參照該佇列時所發生的情況。 屬性範例如下:

- 應用程式是否可以從佇列擷取訊息 (啟用 GET)
- 應用程式是否可以將訊息放置在佇列上 (啟用 PUT)
- 佇列的存取權是一個應用程式專用，還是在應用程式之間共用
- 可同時儲存在佇列上的訊息數上限 (佇列深度上限)
- 可以放置在佇列上的訊息長度上限

您也可以使用各種平台專用介面來定義佇列。

相關概念

第 48 頁的『叢集佇列數』
叢集佇列是由叢集佇列管理程式所管理的佇列，並可提供給叢集的其他佇列管理程式使用。

第 41 頁的『無法傳送郵件的佇列』
無法傳送郵件的佇列 (或無法遞送的訊息佇列) 是訊息無法遞送至正確目的地時傳送至其中的佇列。 每一個佇列管理程式通常都有無法傳送郵件的佇列。

使用 PCF 指令自動化管理

在 IBM MQ Console 中使用佇列

相關工作

使用 MQSC 指令管理 IBM MQ

使用「 MQ 探險家」建立及配置佇列管理程式和物件

IBM i 使用 CL 指令管理 IBM MQ for IBM i

z/OS 您可以從中在 IBM MQ for z/OS 上發出 MQSC 及 PCF 指令的來源

相關參考

第 48 頁的『共用佇列與叢集佇列之間的比較』
此資訊旨在協助您比較共用佇列和叢集佇列，並決定哪些可能更適合您的系統。

相關資訊

第 139 頁的『何謂共用佇列?』

本端佇列

傳輸、起始、無法傳送的郵件、指令、預設、通道及事件佇列都是本端佇列的類型。

如果佇列是由程式連接到的佇列管理程式所擁有，則該佇列對程式而言是本端佇列。 您可以從本端佇列中取得訊息，以及將訊息放置在本端佇列上。

佇列定義物件會保留佇列的定義資訊，以及放置在該佇列上的實體訊息。

每一個佇列管理程式都可以有一些本端佇列，用於特殊用途:

傳輸佇列

當應用程式將訊息傳送至遠端佇列時，本端佇列管理程式會將訊息儲存在特殊本端佇列中，稱為 傳輸佇列。 應用程式可以將訊息直接放置在傳輸佇列上，或透過遠端佇列定義間接放置。

當佇列管理程式將訊息傳送至遠端佇列管理程式時，它會使用下列順序來識別傳輸佇列:

1. 在遠端佇列本端定義的 XMITQ 屬性上命名的傳輸佇列。
2. 與遠端佇列管理程式同名的傳輸佇列。 此值是遠端佇列本端定義的 XMITQ 預設值。
3. 在本端佇列管理程式的 DEFXMITQ 屬性上指定的傳輸佇列。

訊息通道代理程式 是與傳輸佇列相關聯的通道程式，它會將訊息遞送至下一個目的地。 下一個目的地是訊息通道所連接的佇列管理程式。 它不一定是與訊息最終目的地相同的佇列管理程式。 當訊息遞送至其下一個目的地時，會從傳輸佇列中刪除它。 訊息可能必須在其前往其最終目的地的旅程中通過許多佇列管理程式。 您必須在路徑上的每一個佇列管理程式中定義傳輸佇列，每一個佇列管理程式都會保留等待傳輸至下一個目的地的訊息。 雖然訊息可能有不同的最終目的地，但一般傳輸佇列會保留下一個目的地的訊息。 叢集傳輸佇列會保留多個目的地的訊息。 每則訊息的 `correlID` 會識別放置訊息的通道，以將它傳送至其下一個目的地。

您可以在佇列管理程式中定義數個傳輸佇列。 您可以為相同的目的地定義數個傳輸佇列，每個傳輸佇列都用於不同的服務類別。 例如，您可能想要為小訊息及進入相同目的地的大訊息建立不同的傳輸佇列。 然後，您可以使用不同的訊息通道來傳送訊息，以便大型訊息不會保留較小的訊息。 依預設，叢集佇列或叢集主題的所有訊息都會放在單一叢集傳輸佇列 SYSTEM.CLUSTER.TRANSMIT.QUEUE 上。 作為選項，您可以變更預設值，並將進入不同叢集佇列管理程式的訊息資料流量分隔到不同的叢集傳輸佇列。 如果您將佇列管理程式屬性 DEFCLXQ 設為 CHANNEL，則每一個叢集傳送端通道會建立個別叢集傳輸佇列。 作為替代方案，您可以手動定義叢集傳送端通道要使用的叢集傳輸佇列。

傳輸佇列可以觸發訊息通道代理程式向前傳送訊息; 請參閱 使用觸發程式啟動 IBM MQ 應用程式。

z/OS 在 IBM MQ for z/OS 上，如果您是使用內部群組佇列作業，則 內部群組佇列作業代理程式會處理傳輸佇列。 在 IBM MQ for z/OS 上使用內部群組佇列作業時，會使用共用傳輸佇列。

起始佇列

起始佇列 是一種本端佇列，當應用程式佇列上發生觸發事件時，佇列管理程式會在該佇列上放置觸發訊息。

觸發事件是要讓程式開始處理佇列的事件。 例如，事件可能超過 10 則訊息送達。 如需觸發如何運作的相關資訊，請參閱 使用觸發程式啟動 IBM MQ 應用程式。

無法傳送郵件 (未遞送訊息) 佇列

無法傳送的郵件 (未遞送的訊息) 佇列 是佇列管理程式放置無法遞送之訊息的本端佇列。

當佇列管理程式將訊息放入無法傳送郵件的佇列時，它會將標頭新增至訊息。 標頭資訊包括佇列管理程式將訊息放置在無法傳送郵件的佇列上的原因。 它也包含原始訊息的目的地、日期，以及佇列管理程式將訊息放入無法傳送郵件的佇列的時間。

應用程式也可以使用無法遞送之訊息的佇列。 如需相關資訊，請參閱 使用無法傳送的郵件 (未遞送訊息) 佇列。

系統指令佇列

系統指令佇列 是適當授權的應用程式可以將 IBM MQ 指令傳送至其中的佇列。 這些佇列會接收平台上支援的 PCF、MQSC 及 CL 指令，以準備好讓佇列管理程式對它們執行動作。

z/OS 在 IBM MQ for z/OS 上，佇列稱為 SYSTEM.COMMAND.INPUT ; 在其他平台上，它稱為 SYSTEM.ADMIN.COMMAND.QUEUE。 所接受的指令因平台而異。 如需詳細資料，請參閱可程式化指令格式參照。

系統預設佇列

系統預設佇列 包含系統的佇列起始定義。 當您建立佇列定義時，佇列管理程式會從適當的系統預設佇列複製定義。 建立佇列定義不同於建立動態佇列。 動態佇列的定義基於您選擇作為動態佇列範本的模型佇列。

事件佇列

事件佇列 會保留事件訊息。 這些訊息由佇列管理程式或通道報告。

遠端佇列

對程式而言，如果佇列是由與程式所連接之佇列管理程式不同的佇列管理程式所擁有，則該佇列是 遠端 佇列。

在已建立通訊鏈結的情況下，程式可以將訊息傳送至遠端佇列。 程式永遠無法從遠端佇列取得訊息。

當您定義遠端佇列時所建立的佇列定義物件只會保留本端佇列管理程式所需的資訊，以尋找您要訊息前往的佇列。 此物件稱為 遠端佇列的本端定義。 遠端佇列的所有屬性都由擁有它的佇列管理程式保留，因為它是該佇列管理程式的本端佇列。

開啟遠端佇列時，若要識別佇列，您必須指定下列其中一項:

- 定義遠端佇列的本端定義名稱。 從應用程式的觀點來看，這與開啟本端佇列相同。 應用程式不需要知道佇列是本端或遠端。

  若要在 IBM i 以外的所有平台上建立遠端佇列的本端定義，請使用 DEFINE QREMOTE 指令。

  IBM i 在 IBM i 上，使用 CRTMQMQ 指令。
- 遠端佇列管理程式的名稱，以及該遠端佇列管理程式已知的佇列名稱。

除了 第 17 頁的『佇列的屬性』中說明的共同屬性之外，遠端佇列的本端定義還具有三個屬性。 這三個屬性如下:

RemoteQName
佇列擁有的佇列管理程式用來識別它的名稱。

RemoteQmgrName
擁有端佇列管理程式的名稱。

XmitQName
將訊息轉遞至其他佇列管理程式時所使用的本端傳輸佇列名稱。

如需這些屬性的相關資訊，請參閱 佇列的屬性。

如果您針對遠端佇列的本端定義使用 MQINQ 呼叫，則佇列管理程式只會傳回本端定義的屬性，即遠端佇列名稱、遠端佇列管理程式名稱及傳輸佇列名稱，而不是遠端系統中相符本端佇列的屬性。

另請參閱 傳輸佇列。

別名佇列

別名佇列 是一個 IBM MQ 物件，可用來存取另一個佇列或主題。 這表示多個程式可以使用相同的佇列，並使用不同的名稱來存取該佇列。

解析別名 (稱為基本佇列) 所產生的佇列， 可以是下列任何類型的佇列，如平台所支援:

- 本端佇列
- 遠端佇列的本端定義。
- z/OS 共用佇列，是一種本端佇列類型，只能在 IBM MQ for z/OS 上使用。
- 預先定義的佇列
- 動態佇列

別名也可以解析成主題。 如果應用程式目前將訊息放入佇列，則可以讓佇列名稱成為主題的別名，以發佈至主題。 不需要變更應用程式碼。

註: 別名無法直接解析為相同佇列管理程式上的另一個別名。

別名佇列的使用範例是讓系統管理者對基本佇列名稱 (亦即，別名所解析的佇列) 及別名佇列名稱提供不同的存取權。 這表示程式或使用者可以獲授權使用別名佇列，但不能使用基本佇列。

或者，授權可以設定為禁止對別名執行 put 作業，但允許對基本佇列執行這些作業。

在某些應用程式中，使用別名佇列表示系統管理者可以輕鬆變更別名佇列物件的定義，而不需要變更應用程式。

當程式嘗試使用別名時， IBM MQ 會對別名進行授權檢查。 它不會檢查程式是否有權存取別名所解析的名稱。 因此，程式可以獲授權存取別名佇列名稱，但不能存取已解析的佇列名稱。

除了 第 16 頁的『佇列』中說明的一般佇列屬性之外，別名佇列還具有 **BaseQName** 屬性。 這是別名所解析成的基本佇列名稱。 如需此屬性的完整說明，請參閱 BaseQName (MQCHAR48)。

InhibitGet 和 **InhibitPut** 屬性 (請參閱 第 16 頁的『佇列』) 別名佇列的屬於別名。 例如，如果別名佇列名稱 ALIAS1 解析為基本佇列名稱 BASE ，則 ALIAS1 只會影響 ALIAS1 ，且不禁止 BASE。 不過， BASE 上的禁止也會影響 ALIAS1。

DefPriority 和 **DefPersistence** 屬性也屬於別名。 因此，例如，您可以將不同的預設優先順序指派給相同基本佇列的不同別名。 此外，您也可以變更這些優先順序，而不需要變更使用別名的應用程式。

動態和模型佇列

此資訊可讓您深入瞭解動態佇列、暫時及永久動態佇列的內容、動態佇列的使用、使用動態佇列時的一些考量，以及模型佇列。

當應用程式發出 MQOPEN 呼叫來開啟模型佇列時，佇列管理程式會動態建立與模型佇列具有相同屬性的本端佇列實例。 視模型佇列的 *DefinitionType* 欄位值而定，佇列管理程式會建立暫時或永久動態佇列 (請參閱 建立動態佇列)。

暫時動態佇列的內容

暫時動態佇列 具有下列內容:

- z/OS 它們不能是共用佇列，可從佇列共用群組中的佇列管理程式存取。

  請注意，佇列共用群組只能在 IBM MQ for z/OS 上使用。
- 它們只會保留非持續訊息。
- 它們是無法復原的。
- 當佇列管理程式啟動時，會刪除它們。
- 當發出 MQOPEN 呼叫 (建立佇列) 的應用程式關閉或終止佇列時，會刪除它們。
  - 如果佇列上有任何已確定訊息，則會刪除這些訊息。
  - 如果此時有任何未確定的 MQGET、MQPUT 或 MQPUT1 呼叫未針對佇列執行，則該佇列會標示為邏輯刪除，且在關閉處理期間或應用程式終止時，只會實際刪除 (在確定這些呼叫之後)。
  - 如果目前正在使用佇列 (由建立中或另一個應用程式使用) ，則該佇列會標示為邏輯刪除中，且只有在最後一個使用該佇列的應用程式關閉時才會實際刪除。
  - 嘗試存取邏輯刪除的佇列 (除了關閉它之外) 失敗，原因碼為 MQRC_Q_DELETED。
  - 當 MQCLOSE 呼叫針對建立佇列的對應 MQOPEN 呼叫指定時， MQCO_NONE、MQCO_DELETE 及 MQCO_DELETE_PURGE 都被視為 MQCO_NONE。

永久動態佇列的內容

永久動態佇列 具有下列內容:

- 它們會保留持續或非持續訊息。
- 它們可以在系統失效時回復。
- 當應用程式 (不一定是發出建立佇列的 MQOPEN 呼叫的應用程式) 使用 MQCO_DELETE 或 MQCO_DELETE_PURGE 選項順利關閉佇列時，會刪除它們。
  - 如果佇列上仍有任何訊息 (已確定或未確定) ，則具有 MQCO_DELETE 選項的關閉要求會失敗。 具有 MQCO_DELETE_PURGE 選項的關閉要求會成功，即使佇列上有已確定訊息 (作為關閉的一部分而被刪除的訊息) ，但如果佇列有任何未確定的 MQGET、MQPUT 或 MQPUT1 呼叫未完成，則會失敗。
  - 如果刪除要求成功，但佇列剛好在使用中 (由建立中或另一個應用程式) ，則佇列會標示為邏輯刪除中，且只有在由使用該佇列的最後一個應用程式關閉時才會實際刪除。
- 如果由未獲授權刪除佇列的應用程式關閉，則不會刪除它們，除非關閉的應用程式發出建立佇列的 MQOPEN 呼叫。 會針對用來驗證對應 MQOPEN 呼叫的使用者 ID (或替代使用者 ID，如果已指定 MQOO_ALTERNATE_USER_AUTHORITY) 執行授權檢查。
- 可以使用與一般佇列相同的方式來刪除它們。

使用動態佇列

您可以將動態佇列用於:

- 在應用程式終止之後，不需要保留佇列的應用程式。
- 需要另一個應用程式處理訊息之回覆的應用程式。 此類應用程式可以透過開啟模型佇列來動態建立回覆目的地佇列。 例如，用戶端應用程式可以:
  1. 建立動態佇列。
  2. 請在要求訊息之訊息描述子結構的 **ReplyToQ** 欄位中提供其名稱。
  3. 將要求放在伺服器正在處理的佇列上。

然後伺服器可以將回覆訊息放置在回覆目的地佇列上。 最後，用戶端可以處理回覆，並使用刪除選項來關閉回覆目的地佇列。

使用動態佇列時的考量

使用動態佇列時，請考量下列要點:

- 在主從架構模型中，每一個用戶端必須建立並使用自己的動態回覆目的地佇列。 如果在多個用戶端之間共用動態回覆目的地佇列，則刪除回覆目的地佇列可能會延遲，因為沒有針對佇列的未確定活動，或因為另一個用戶端正在使用該佇列。 此外，佇列可能標示為邏輯上已刪除，且後續 API 要求 (非 MQCLOSE) 無法存取。
- 如果您的應用程式環境要求必須在應用程式之間共用動態佇列，請確保只有在已確定佇列的所有活動時，才會關閉佇列 (使用刪除選項)。 這應該由最後一位使用者來執行。 這可確保佇列的刪除不會延遲，並將佇列因已標示為邏輯刪除而無法存取的期間縮至最小。

模型佇列

模型佇列 是您在建立動態佇列時使用的佇列定義範本。

您可以從 IBM MQ 程式動態建立本端佇列，並命名您要用作佇列屬性範本的模型佇列。 此時，您可以變更新佇列的部分屬性。 不過，您無法變更 **DefinitionType**。 例如，如果您需要永久佇列，請選取定義類型設為永久的模型佇列。 部分交談式應用程式可以使用動態佇列來保留其查詢的回覆，因為它們在處理回覆之後可能不需要維護這些佇列。

您可以在 MQOPEN 呼叫的 物件描述子 (MQOD) 中指定模型佇列的名稱。 使用模型佇列的屬性，佇列管理程式會為您動態建立本端佇列。

您可以指定動態佇列的名稱 (完整) 或名稱的詞幹 (例如 ABC) ，並讓佇列管理程式將唯一組件新增至此，也可以讓佇列管理程式為您指派完整的唯一名稱。 如果佇列管理程式指派名稱，則會將它放入 MQOD 結構中。

您無法直接對模型佇列發出 MQPUT1 呼叫，但可以對透過開啟模型佇列建立的動態佇列發出 MQPUT1 。

無法針對模型佇列發出 MQSET 和 MQINQ。 使用 MQOO_INQUIRE 或 MQOO_SET 開啟模型佇列會導致對動態建立的佇列進行後續 MQINQ 及 MQSET 呼叫。

模型佇列的屬性是本端佇列的子集。 如需更完整的說明，請參閱 佇列的屬性。

IBM MQ 用於特定目的的佇列

IBM MQ 會將某些本端佇列用於與其作業相關的特定用途。

您必須先定義這些佇列，然後 IBM MQ 才能使用它們。

起始佇列

起始佇列是在觸發時使用的佇列。 發生觸發事件時，佇列管理程式會將觸發訊息放置在起始佇列上。 觸發事件是佇列管理程式偵測到的條件邏輯組合。 例如，當佇列上的訊息數達到預先定義的深度時，可能會產生觸發事件。 此事件會導致佇列管理程式將觸發訊息放置在指定的起始佇列上。 此觸發訊息由 觸發監視器擷取，觸發監視器是監視起始佇列的特殊應用程式。 然後，觸發監視器會啟動觸發訊息中指定的應用程式。

如果佇列管理程式要使用觸發，則必須為該佇列管理程式至少定義一個起始佇列。 請參閱 管理物件以使用觸發程式來觸發、 runmqtrm 及 啟動 IBM MQ 應用程式

傳輸佇列

傳輸佇列是暫時儲存以遠端佇列管理程式為目的地之訊息的佇列。 您必須為本端佇列管理程式要直接傳送訊息的每一個遠端佇列管理程式至少定義一個傳輸佇列。 這些佇列也用於遠端管理; 請參閱 從本端佇列管理程式進行遠端管理。 如需在分散式佇列中使用傳輸佇列的相關資訊，請參閱 IBM MQ 分散式佇列技術。

每一個佇列管理程式都可以有預設傳輸佇列。 如果不屬於叢集的佇列管理程式將訊息放入遠端佇列，則預設動作是使用預設傳輸佇列。 如果有傳輸佇列與目的地佇列管理程式同名，則會將訊息放置在該傳輸佇列上。 如果有佇列管理程式別名定義，其中 **RQMNAME** 參數符合目的地佇列管理程式，且指定 **XMITQ** 參數，則訊息會放置在 **XMITQ** 所命名的傳輸佇列上。 如果沒有 **XMITQ** 參數，則訊息會放在訊息中指名的本端佇列上。

叢集傳輸佇列

叢集內的每一個佇列管理程式都有一個稱為 SYSTEM.CLUSTER.TRANSMIT.QUEUE 的叢集傳輸佇列，以及一個模型叢集傳輸佇列 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE。 依預設，當您定義佇列管理程式時，會建立這些佇列的定義。 如果佇列管理程式屬性 **DEFCLXQ** 設為 CHANNEL，則會自動為所建立的每一個叢集傳送端通道建立永久動態叢集傳輸佇列。 佇列稱為 SYSTEM.CLUSTER.TRANSMIT.*ChannelName*。 您也可以手動定義叢集傳輸佇列。

屬於叢集的佇列管理程式會將其中一個佇列上的訊息傳送至相同叢集中的其他佇列管理程式。

在名稱解析期間，叢集傳輸佇列優先於預設傳輸佇列，而特定叢集傳輸佇列優先於 SYSTEM.CLUSTER.TRANSMIT.QUEUE。

無法傳送郵件的佇列

無法傳送郵件的 (無法遞送的訊息) 佇列是儲存無法遞送至其正確目的地之訊息的佇列。 例如，當目的地佇列已滿時，無法遞送訊息。 提供的無法傳送郵件的佇列稱為 SYSTEM.DEAD.LETTER.QUEUE。

對於分散式佇列，請在涉及的每一個佇列管理程式上定義無法傳送郵件的佇列。

指令佇列

指令佇列 SYSTEM.ADMIN.COMMAND.QUEUE 是本端佇列，適當授權的應用程式可以將 MQSC 指令傳送至其中進行處理。 然後，稱為指令伺服器的 IBM MQ 元件會擷取這些指令。 指令伺服器會驗證指令，傳遞有效的指令以供佇列管理程式處理，並將任何回應傳回適當的回覆目的地佇列。

建立每一個佇列管理程式時，會自動建立該佇列管理程式的指令佇列。

回覆目的地佇列

當應用程式傳送要求訊息時，接收訊息的應用程式可以傳回回覆訊息給傳送應用程式。 此訊息放置在佇列上，稱為回覆目的地佇列，通常是傳送端應用程式的本端佇列。 回覆目的地佇列的名稱由傳送端應用程式指定為訊息描述子的一部分。

事件佇列

檢測事件可用來獨立監視佇列管理程式，與 MQI 應用程式無關。

發生檢測事件時，佇列管理程式會將事件訊息放置在事件佇列上。 然後，監視應用程式可以讀取此訊息，如果事件指出問題，則可能會通知管理者或起始一些補救動作。

註: 觸發程式事件不同於檢測事件。 觸發事件不是由相同的條件所造成，且不會產生事件訊息。

如需設備測試事件的相關資訊，請參閱 設備測試事件。

佇列管理程式

佇列管理程式 及其提供給應用程式之佇列服務的簡介。

程式必須具有與佇列管理程式的連線，才能使用該佇列管理程式的服務。 程式可以明確地建立此連線 (使用 MQCONN 或 MQCONNX 呼叫)，也可以隱含地建立連線 (這取決於執行程式的平台及環境)。

IBM MQ 佇列管理程式可確保執行下列動作:

- 根據接收的指令變更物件屬性。
- 當符合適當的條件時，會產生特殊事件 (例如觸發事件或設備測試事件)。
- 訊息會依照發出 MQPUT 呼叫的應用程式所要求，放置在正確的佇列中。 如果無法執行此動作，則會通知應用程式，並提供適當的原因碼。

每一個佇列都屬於單一佇列管理程式，且被認為是該佇列管理程式的 本端佇列 。 應用程式所連接的佇列管理程式被認為是該應用程式的 本端佇列管理程式 。 對於應用程式，屬於其本端佇列管理程式的佇列是本端佇列。

遠端佇列 是屬於另一個佇列管理程式的佇列。 遠端佇列管理程式 是本端佇列管理程式以外的任何佇列管理程式。 遠端佇列管理程式可以存在於網路上的遠端機器上，也可以存在於與本端佇列管理程式相同的機器上。 IBM MQ 支援相同機器上的多個佇列管理程式。

佇列管理程式物件可以在部分 MQI 呼叫中使用。 例如，您可以使用 MQI 呼叫 MQINQ 來查詢佇列管理程式物件的屬性。

佇列管理程式的屬性

與每一個佇列管理程式相關聯的是一組定義其性質的屬性 (或內容)。佇列管理程式的部分屬性在建立時是固定的; 您可以使用 IBM MQ 指令來變更其他屬性。 您可以使用 MQINQ 呼叫來查詢所有屬性的值，但那些用於「傳輸層安全 (TLS)」加密的屬性除外。

固定屬性包括:

- 佇列管理程式的名稱
- 佇列管理程式執行所在的平台 (例如， Windows)
- 佇列管理程式支援的系統控制指令層次
- 您可以指派給佇列管理程式所處理訊息的優先順序上限
- 程式可以將 IBM MQ 指令傳送至其中的佇列名稱
- 佇列管理程式可以處理的訊息長度上限 z/OS (僅在 IBM MQ for z/OS 中固定)
- 當程式放置及取得訊息時，佇列管理程式是否支援同步點

可變更 屬性包括:

- 佇列管理程式的文字說明
- 佇列管理程式在處理 MQI 呼叫時用於字串的字集 ID
- 佇列管理程式用來限制觸發訊息數目的時間間隔
- z/OS 佇列管理程式用來決定在佇列中掃描過期訊息的頻率的時間間隔 (僅限 IBM MQ for z/OS)
- 佇列管理程式的無法傳送郵件 (未遞送訊息) 佇列名稱
- 佇列管理程式的預設傳輸佇列名稱
- 任何一個連線的開啟控點數目上限
- 啟用及停用事件報告的各種種類
- 工作單元內未確定的訊息數上限

佇列管理程式及工作量管理

您可以設定佇列管理程式的叢集，該佇列管理程式具有相同佇列的多個定義 (例如，叢集中的佇列管理程式可以是彼此的複本)。 管理佇列實例的任何佇列管理程式都可以處理特定佇列的訊息。 工作量管理演算法會決定哪個佇列管理程式處理訊息，因此會在佇列管理程式之間分散工作量; 如需進一步資訊，請參閱 叢集工作量管理演算法 。

通道

通道 是邏輯通訊鏈結，由分散式佇列管理程式、 IBM MQ MQI client 與 IBM MQ 伺服器之間或兩個 IBM MQ 伺服器之間使用。

通道用來將訊息從一個佇列管理程式移至另一個佇列管理程式，它們會保護應用程式免受基礎通訊協定的影響。 佇列管理程式可能存在於相同系統上，或相同平台上或不同平台上的不同系統上。 傳送的訊息可以來自許多地方:

- 使用者撰寫的應用程式，可將資料從一個節點傳送至另一個節點。
- 使用 PCF 指令或 MQAI 的使用者撰寫管理應用程式。
- IBM MQ Explorer。
- 將檢測事件訊息傳送至另一個佇列管理程式的佇列管理程式。
- 將遠端管理指令傳送至另一個佇列管理程式的佇列管理程式。 例如，使用 MQSC 指令或 administrative REST API。

通道有兩個定義: 連線兩端各一個。 若要讓佇列管理程式彼此通訊，您必須在佇列管理程式中定義一個通道物件來傳送訊息，並在佇列管理程式中定義另一個互補的通道物件來接收訊息。 必須在連線的每一端使用相同的 通道名稱 ，且使用的 通道類型 必須相容。

IBM MQ 中有三種通道種類，在這些種類內具有不同的通道類型:

- 訊息通道是單向的，並將訊息從一個佇列管理程式傳送至另一個佇列管理程式。
- MQI 通道是雙向的，可將 MQI 呼叫從 IBM MQ MQI client 傳送至佇列管理程式，並將回應從佇列管理程式傳送至 IBM MQ 用戶端。
- AMQP 通道，其為雙向，並將 AMQP 用戶端連接至伺服器機器上的佇列管理程式。 IBM MQ 使用 AMQP 通道在 AMQP 應用程式與佇列管理程式之間傳送 AMQP 呼叫及回應

訊息通道

訊息通道的目的是將訊息從一個佇列管理程式傳送至另一個佇列管理程式。 用戶端伺服器環境不需要訊息通道。

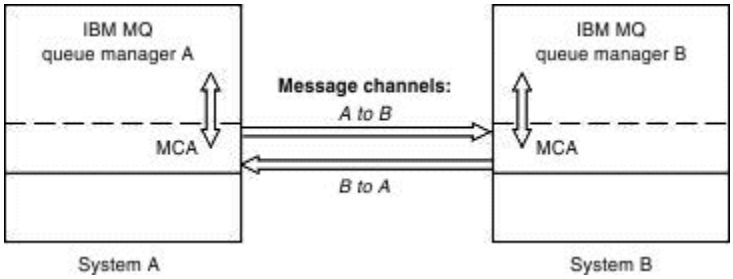


圖 *2:* 兩個佇列管理程式之間的訊息通道

訊息通道是單向鏈結。 如果您想要遠端佇列管理程式回應本端佇列管理程式所傳送的訊息，您必須設定第二個通道，將回應傳回本端佇列管理程式。

訊息通道會使用 訊息通道代理程式 (MCA) 來連接兩個佇列管理程式。 通道的每一端都有一個訊息通道代理程式。 您可以容許 MCA 使用多個執行緒來傳送訊息。 此處理程序稱為 管線化。 管線化可讓 MCA 更有效率地傳送訊息，改善通道效能。 如需管線化的相關資訊，請參閱 通道屬性。

如需通道的相關資訊，請參閱 通道結束程式呼叫和資料結構及 第 38 頁的『分散式佇列元件』。

MQI 通道

「訊息佇列介面 (MQI)」通道會將 IBM MQ MQI client 連接至伺服器機器上的佇列管理程式，而且會在您從 IBM MQ MQI client 應用程式發出 MQCONN 或 MQCONNX 呼叫時建立。

它是雙向鏈結，僅用於傳送 MQI 呼叫及回應，包括包含訊息資料的 MQPUT 呼叫及導致傳回訊息資料的 MQGET 呼叫。 您可以使用不同的方式來建立及使用通道定義 (請參閱 定義 MQI 通道)。

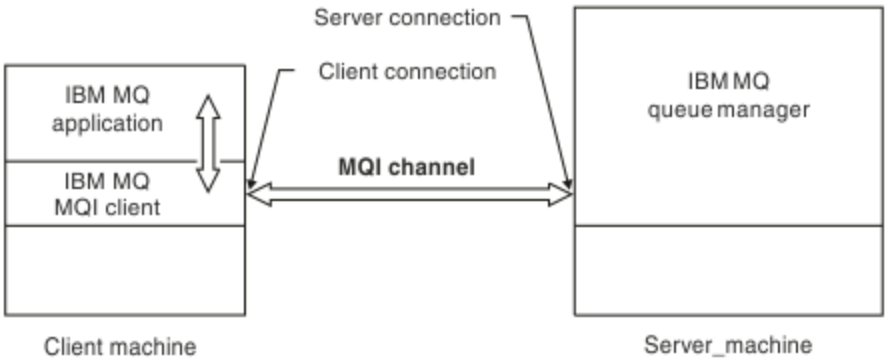


圖 *3: MQI* 通道上的用戶端連線及伺服器連線

z/OS MQI 通道可用來將用戶端連接至單一佇列管理程式，或連接至屬於佇列共用群組的佇列管理程式 (請參閱 將用戶端連接至佇列共用群組)。

MQI 通道定義有兩種通道類型。 它們定義雙向 MQI 通道。

用戶端連線通道
此類型適用於 IBM MQ MQI client。

伺服器連線通道
此類型適用於執行佇列管理程式的伺服器，在 IBM MQ MQI client 環境中執行的 IBM MQ 應用程式將與該伺服器進行通訊。

AMQP 通道

只有一種類型的 AMQP 通道。

您可以使用此通道將 AMQP 傳訊應用程式與佇列管理程式連接，從而允許此應用程式與 IBM MQ 應用程式交換訊息。 AMQP 通道可讓您使用 MQ Light 開發應用程式，然後將它部署為企業應用程式，並利用 IBM MQ 所提供的企業層次機能。

用戶端連線通道

用戶端連線通道 是提供從 IBM MQ MQI client 到佇列管理程式之通訊路徑的物件。

在分散式佇列中使用用戶端連線通道，以在佇列管理程式與用戶端之間移動訊息。 它們會從基礎通訊協定保護應用程式。 用戶端可能存在於與佇列管理程式相同的或不同的平台上。

通道定義

如需每一種通道類型的說明，請參閱 第 26 頁的『通道定義』 。

相關概念
第 35 頁的『分散式佇列及叢集』
分散式佇列作業表示將訊息從一個佇列管理程式傳送至另一個佇列管理程式。 接收端佇列管理程式可以位於相同或另一部機器上; 附近或世界另一端。 它可以在與本端佇列管理程式相同的平台上執行，也可以在 IBM MQ 支援的任何平台上執行。 您可以在分散式佇列環境中手動定義所有連線，也可以建立叢集，並讓 IBM MQ 為您定義許多連線詳細資料。

訊息佇列介面概觀

相關工作
管理遠端 IBM MQ 物件
停止 MQI 通道
配置伺服器與用戶端之間的連線

相關參考
通道結束程式呼叫和資料結構
第 29 頁的『傳播業』
IBM MQ MQI clients 使用 MQI 通道來與伺服器進行通訊。

通道定義

下表說明 IBM MQ 使用的不同類型的訊息通道及 MQI 通道。

談到訊息通道時，通道一詞通常用來作為通道定義的同義字。 通常從上下文會很清楚所談的是完整的通道（具備兩端），或者是通道定義（只有一端）。

訊息通道

訊息通道定義可以是下列其中一種類型:

訊息通道定義類型	說明
傳送端	傳送端通道是佇列管理程式用來傳送訊息到另一個佇列管理程式的訊息通道。 若要使用傳送端通道來傳送訊息，您也必須在另一個佇列管理程式上，建立一個和傳送端通道同名的接收端通道。 如果您正在實作「回呼」機制，也可以使用有要求端通道的傳送端通道。
伺服器	伺服器通道是佇列管理程式用來傳送訊息到另一個佇列管理程式的訊息通道。 若要使用伺服器通道來傳送訊息，您也必須在另一個佇列管理程式上，建立一個和伺服器通道同名的接收端通道。 您也可以將伺服器通道與要求端通道一起使用。 在此情況下，通道另一端的要求端通道定義會要求伺服器通道定義啟動。 伺服器會傳送訊息給要求端。 只要伺服器知道友機通道的連線名稱，伺服器也可以起始通訊。
接收端	接收端通道是佇列管理程式用來從另一個佇列管理程式接收訊息的訊息通道。 若要使用接收端通道來接收訊息，您也必須在另一個佇列管理程式上，建立一個和接收端通道同名的傳送端或伺服器通道。
要求端	「要求端」通道是佇列管理程式用來接收其他佇列管理程式訊息的訊息通道。 要求端通道可以要求在遠端結束時定義的夥伴通道啟動。 如果友機通道是伺服器通道，伺服器通道會接受啟動要求，並從「伺服器」通道定義中識別的傳輸佇列，開始傳送訊息至要求端通道。 如果友機通道是傳送端通道，則傳送端通道會接受啟動要求，但會關閉與要求端之間的連線。 然後，傳送端通道會啟動，並與友機要求端通道協議階段作業，並開始從傳送端通道定義中識別的傳輸佇列傳送訊息。 後一種情況主要針對要求端通道要求傳送端通道進行呼叫支援的呼叫支援機制。
叢集傳送端	叢集傳送端 (CLUSSDR) 通道定義會定義通道傳送端，叢集佇列管理程式可以在其中傳送叢集資訊給其中一個完整儲存庫。 叢集傳送端通道會用來通知儲存庫關於佇列管理程式狀態的任何變更，例如：新增或移除佇列。 它也可以用來傳輸訊息。 完整儲存庫佇列管理程式本身有指向彼此的叢集傳送端通道。 它們會使用這些通道來互相溝通叢集狀態變更。 佇列管理程式的 CLUSSDR 通道定義指向哪個完整儲存庫有點重要。 在初次聯絡之後，會依需求自動定義進一步的叢集佇列管理程式物件，使得佇列管理程式可以傳送叢集資訊給每一個完整儲存庫，並將訊息傳給每一個佇列管理程式。
叢集接收端	叢集接收端 (CLUSRCVR) 通道定義會定義通道接收端，叢集佇列管理程式可以在其中接收來自叢集中其他佇列管理程式的訊息。 叢集接收端通道也可以附帶叢集的相關資訊（指定給儲存庫的資訊）。 佇列管理程式藉由定義叢集接收端通道，向另一個叢集佇列管理程式表示它可用來接收訊息。 每一個叢集佇列管理程式至少需要一個叢集接收端通道。

您必須定義每一個通道的兩端，以便通道每一端都具有通道定義。 通道兩端的類型必須相容。

您可以有下列通道定義組合：

- 傳送端-接收端
- 伺服器-接收端

- 要求端-伺服器
- 要求端-傳送端（回呼）
- 叢集-傳送端-叢集-接收端

訊息通道代理程式

您建立的每一個通道定義都屬於特定的佇列管理程式。 一個佇列管理程式可以有數個同類型或不同類型的通道。 通道的每一端是一個程式，也就是訊息通道代理程式 (MCA)。 在通道的一端，呼叫端 MCA 會從傳輸佇列取得訊息，再透過通道來傳送。 在通道的另一端，回應端 MCA 會接收訊息，再將它們遞送到遠端佇列管理程式。

呼叫端 MCA 可以與傳送端、伺服器或要求端通道關聯。 回應端 MCA 可以與任何類型的訊息通道關聯。

IBM MQ 支援在連線兩端的下列通道類型組合：

呼叫端		**訊息流方向**	**回應端**	
通道類型	**需要接聽器?**		**需要接聽器?**	**通道類型**
傳送端	否	呼叫端到回應端	是	接收端
伺服器	否	呼叫端到回應端	是	接收端
伺服器	否	呼叫端到回應端	是	要求端
要求端	否	回應端到呼叫端	是	伺服器
要求端	是	回應端到呼叫端	是	傳送端

MQI 通道

MQI 通道可以是下列其中一種類型：

MQI 通道類型	**說明**
伺服器連線	伺服器連線通道是一種雙向 MQI 通道，用來將 IBM MQ 用戶端連接至 IBM MQ 伺服器。 伺服器連線通道是指通道的伺服器端。
用戶端連線	用戶端連線通道是一種雙向 MQI 通道，用來將 IBM MQ 用戶端連接至 IBM MQ 伺服器。 「IBM MQ Explorer」也使用用戶端連線，連接到遠端佇列管理程式。 用戶端連線通道是指通道的用戶端。 當您建立用戶端連線通道時，控管佇列管理程式的電腦上會建立一個檔案。 然後，您必須將用戶端連線檔案複製到 IBM MQ 用戶端電腦。

Multi 多執行緒支援-管線化

您可以選擇性地容許訊息通道代理程式 (MCA) 使用多個執行緒來傳送訊息。 此處理程序稱為 管線化，可讓 MCA 以較少等待狀態更有效率地傳送訊息，從而增進通道效能。 每一個 MCA 最多只能有兩個執行緒。

您可以使用 qm.ini 檔案中的 *PipeLineLength* 參數來控制管線化。 此參數會新增至 通道段落。

註: 管線只對 TCP/IP 通道有效。

當您使用管線化時，通道兩端的佇列管理程式必須配置為 *PipeLine* 長度 大於 1。

通道結束程式考量

管線化可能會導致部分結束程式失敗，因為：

- 可能不會循序呼叫結束程式。

- 可以從不同的執行緒輪流呼叫結束程式。

在使用管線化之前，請先檢查結束程式的設計:

- 結束程式必須在其執行的所有階段重新進入。
- 當您使用 MQI 呼叫時，請記住，從不同執行緒呼叫結束程式時，無法使用相同的 MQI 控點。

假設訊息結束程式會開啟佇列，並在所有後續呼叫結束程式時使用其控點來進行 MQPUT 呼叫。 這在管線化模式中失敗，因為從不同的執行緒呼叫結束程式。 若要避免此失敗，請保留每一個執行緒的佇列控點，並在每次呼叫結束程式時檢查執行緒 ID。

傳播業

IBM MQ MQI clients 使用 MQI 通道來與伺服器進行通訊。

必須在連線的 IBM MQ MQI client 及伺服器兩端建立通道定義。 定義 MQI 通道中說明如何建立通道定義。

下表顯示可能的傳輸通訊協定:

表 *1: MQI* 通道的傳輸通訊協定				
用戶端平台	**LU 6.2**	**TCP/IP**	**NetBIOS**	**SPX**
IBM i IBM i		是		
Linux AIX AIX and Linux 系統	是 [1]	是		
Windows Windows	是	是	是	是

註:

1. Linux 下列平台不支援 LU6.2 :
   - Linux (POWER 平台)
   - Linux (x86-64 平台)
   - Linux (zSeries s390x 平台)

傳輸通訊協定- IBM MQ MQI client 與伺服器平台的組合 顯示使用這些傳輸通訊協定的 IBM MQ MQI client 與伺服器平台的可能組合。

IBM MQ MQI client 上的 IBM MQ 應用程式可以使用所有 MQI 呼叫，其方式與佇列管理程式在本端時相同。 **MQCONN** 或 **MQCONNX** 會建立 IBM MQ 應用程式與所選佇列管理程式的關聯，並建立 連線控點。 然後，連接的佇列管理程式會處理使用該連線控點的其他呼叫。 IBM MQ MQI client 通訊需要用戶端與伺服器之間的作用中連線，與佇列管理程式之間的通訊相反，後者與連線無關且與時間無關。

傳輸通訊協定是使用通道定義來指定，不會影響應用程式。 例如， Windows 應用程式可以透過 TCP/IP 連接至一個佇列管理程式，並透過 NetBIOS 連接至另一個佇列管理程式。

效能考量

您使用的傳輸通訊協定可能會影響 IBM MQ 用戶端及伺服器系統的效能。 在傳輸速度緩慢的特定狀況下，您可以使用 IBM MQ 通道壓縮。

命名 IBM MQ 物件

IBM MQ 物件採用的命名慣例視物件而定。 與 IBM MQ 搭配使用的機器名稱及使用者 ID 也會受到一些命名限制。

佇列管理程式的每一個實例都以其名稱來識別。 此名稱在交互連接的佇列管理程式網路內必須是唯一的，因此一個佇列管理程式可以明確地識別任何給定訊息傳送至其中的目標佇列管理程式。

對於其他類型的物件，每一個物件都有一個相關聯的名稱，且可以用該名稱來參照。 這些名稱在一個佇列管理程式及物件類型中必須是唯一的。 例如，您可以具有同名的佇列及處理程序，但不能具有同名的兩個佇列。

在 IBM MQ 中，名稱最多可以有 48 個字元，但 *channels* (最多 20 個字元) 除外。 如需命名 IBM MQ 物件的相關資訊，請參閱 第 30 頁的『IBM MQ 物件的命名規則』。

與 IBM MQ 搭配使用的機器名稱及使用者 ID 也會受到一些命名限制:

- 請確定機器名稱不包含任何空格。 IBM MQ 不支援包含空格的機器名稱。 如果您在這類機器上安裝 IBM MQ，則無法建立任何佇列管理程式。
- 對於 IBM MQ 授權，使用者 ID 和群組的名稱不得超過 20 個字元 (不接受空格)。
- Windows 如果用戶端以包含 @ 字元 (例如， abc@d.) 的使用者 ID 執行，則 IBM MQ for Windows 伺服器不支援 IBM MQ MQI client 的連線。

相關概念

第 33 頁的『IBM MQ 檔案名稱』
每一個 IBM MQ 佇列管理程式、佇列、程序定義、名單、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件都由一個檔案代表。 因為物件名稱不一定是有效的檔名，所以佇列管理程式會在必要時將物件名稱轉換成有效的檔名。

相關參考

第 30 頁的『IBM MQ 物件的命名規則』
IBM MQ 物件名稱具有長度上限，且區分大小寫。 並非每個物件類型都支援所有字元，而且許多物件都有關於名稱唯一性的規則。

IBM MQ 物件的命名規則

IBM MQ 物件名稱具有長度上限，且區分大小寫。 並非每個物件類型都支援所有字元，而且許多物件都有關於名稱唯一性的規則。

IBM MQ 物件有許多不同類型，而且每一種類型的物件都可以具有相同的名稱，因為它們存在於個別物件名稱空間中: 例如，本端佇列和傳送端通道都可以具有相同的名稱。 但物件名稱不能與相同名稱空間中的另一個物件名稱相同：例如，本端佇列的名稱不能與模型佇列的名稱相同，且傳送端通道的名稱不能與接收端通道的名稱相同。

下列 IBM MQ 物件存在於個別物件名稱空間中:

- 鑑別資訊
- 通道
- 用戶端通道
- 接聽器
- 名稱清單
- 處理程序
- 佇列
- 服務
- 儲存體類別
- 訂閱
- 主題

物件名稱的字元長度

一般而言， IBM MQ 物件名稱最長可為 48 個字元。 此規則適用於下列物件:

- 鑑別資訊
- 叢集

- 接聽器
- 名稱清單
- 程序定義
- 佇列
- 佇列管理程式
- 服務
- 訂閱
- 主題

有一些限制:

1. z/OS 在 z/OS 系統上，佇列管理程式最多只能有 4 個字元，且必須是大寫字元及數值字元。
2. 通道物件名稱及用戶端連線通道名稱的長度上限為 20 個字元。 如需通道的相關資訊，請參閱 定義通道 。
3. 主題字串最多可以有 10240 個位元組。 所有 IBM MQ 物件名稱都區分大小寫。
4. 訂閱名稱最多可以有 10240 個位元組，且可以包含空格。
5. 儲存類別名稱的長度上限為 8 個字元。
6. CF 結構名稱的長度上限為 12 個字元。

物件名稱中的字元

IBM MQ 物件名稱的有效字元為:

個字元後	限制
大寫 A-Z	• 無
小寫 a-z	• 在 MQSC Script 中，必須以單引號括住具有小寫字元的名稱。 這可防止小寫字元轉換成大寫。 • 使用 EBCDIC Katakana 的系統無法在物件名稱中使用小寫 a-z 字元。 • z/OS 在 z/OS 系統上使用小寫字元時可能有一些限制，例如，佇列管理程式名稱不能包含小寫字元。 • IBM i 在 IBM i 系統上使用 CL 指令時，必須以單引號括住具有小寫字元的名稱。 這可防止小寫字元轉換成大寫。
數字 0-9	• 無
句點 (.)	• 無
底線 (_)	• Multi 無 • z/OS 請避免使用含前導或尾端底線的名稱，因為 IBM MQ for z/OS 作業和控制面板無法處理它們。

個字元後	限制
正斜線 (/)	• Windows 在 Windows 系統上，佇列管理程式名稱的第一個字元不能是正斜線。 • IBM i 在 IBM i 系統上使用 CL 指令時，包含正斜線的名稱必須以單引號括住。 • z/OS 無
百分比符號 (%)	• ALW 無 • z/OS 如果您使用 RACF 作為 IBM MQ for z/OS 的外部安全管理程式，請勿在物件名稱中使用%，因為當使用 RACF 通用設定檔時，這些名稱不會包含在安全檢查中。 • IBM i 在 IBM i 系統上使用 CL 指令時，包含百分比符號的名稱必須以單引號括住。

還有一些關於物件名稱字元的一般規則:

1. 不容許前導或內嵌空白。
2. 不接受國家語言字元。
3. 任何小於完整欄位長度的名稱都可以用空白填補右邊。 佇列管理程式所傳回的所有簡稱一律以空白填補在右側。

佇列名稱

佇列名稱有兩個部分:

- 佇列管理程式的名稱
- 佇列管理程式已知的佇列本端名稱

佇列名稱的每一個部分長度為 48 個字元。

若要參照本端佇列，您可以省略佇列管理程式的名稱 (方法是將它取代為空白字元或使用前導空值字元)。 不過， IBM MQ 傳回給程式的所有佇列名稱都包含佇列管理程式的名稱。

z/OS 共用佇列 (可供其佇列共用群組中的任何佇列管理程式存取) 不能與相同佇列共用群組中的任何非共用本端佇列同名。 此限制可避免應用程式在想要開啟本端佇列時誤開啟共用佇列的可能性，反之亦然。 共用佇列和佇列共用群組只能在 IBM MQ for z/OS 上使用。

若要參照遠端佇列，程式必須在完整佇列名稱中包含佇列管理程式的名稱，或必須有遠端佇列的本端定義。

當應用程式使用佇列名稱時，該名稱可以是本端佇列的名稱 (或別名) 或遠端佇列的本端定義名稱，但應用程式不需要知道該名稱，除非它需要從佇列取得訊息 (當佇列必須是本端佇列時)。 當應用程式開啟佇列物件時， MQOPEN 呼叫會執行名稱解析函數，以決定要對哪個佇列執行後續作業。 其重要性在於應用程式對於在佇列管理程式網路中的特定位置所定義的特定佇列沒有內建相依關係。 因此，如果系統管理者重新定位網路中的佇列，並變更其定義，則不需要變更使用那些佇列的應用程式。

保留物件名稱

以 SYSTEM. 開頭的物件名稱會保留給佇列管理程式所定義的物件。 您可以使用 **Alter**、 **Define** 及 **Replace** 指令來變更這些物件定義，以符合您的安裝。 定義給 IBM MQ 的名稱會完整列出在 佇列名稱中。

z/OS 在 IBM MQ for z/OS 上， 會保留連結機能應用程式結構名稱 CSQSYSAPPL。

相關概念

AIX, Linux, and Windows 上的安裝名稱

IBM MQ 檔案名稱

每一個 IBM MQ 佇列管理程式、佇列、程序定義、名單、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件都由一個檔案代表。 因為物件名稱不一定是有效的檔名，所以佇列管理程式會在必要時將物件名稱轉換成有效的檔名。

佇列管理程式目錄的預設路徑如下:

- 在 IBM MQ 配置資訊中定義的字首:
  - Linux AIX 在 AIX and Linux 上，預設字首為 `/var/mqm`。 這會配置在 mqs.ini 配置檔的 `DefaultPrefix` 段落中。
  - Windows 在 Windows 32 位元系統上，預設字首為 `C:\Program Files (x86)\IBM\WebSphere MQ`。 在 Windows 64 位元系統上，預設字首為 `C:\Program Files\IBM\MQ`。 對於 32 位元及 64 位元安裝，資料目錄會安裝至 `C:\ProgramData\IBM\MQ`。 這會配置在 mqs.ini 配置檔的 `DefaultPrefix` 段落中。

  如果可用的話，可以使用「IBM MQ 檔案總管」中的 IBM MQ 內容頁面來變更字首，否則手動編輯 `mqs.ini` 配置檔。
- 佇列管理程式名稱會轉換成有效的目錄名稱。 例如，佇列管理程式:

```
queue.manager
```

  將表示為:

```
queue!manager
```

  此程序稱為 名稱轉換。

在 IBM MQ 中，您可以為佇列管理程式提供最多包含 48 個字元的名稱。

例如，您可以命名佇列管理程式:

```
QUEUE.MANAGER.ACCOUNTING.SERVICES
```

不過，每一個佇列管理程式都由一個檔案代表，且檔名的長度上限以及名稱中可以使用的字元有一些限制。 因此，會自動轉換代表物件的檔案名稱，以符合檔案系統的需求。

控管佇列管理程式名稱轉換的規則如下:

1. 轉換個別字元:
   - 從。 到!
   - From / to &
2. 如果名稱仍然無效:
   a. 將它截斷為 8 個字元
   b. 附加三個字元的數值字尾

例如，假設預設字首及名稱為 `queue.manager` 的佇列管理程式:

- Windows 在具有 NTFS 或 FAT32 的 Windows 上，佇列管理程式名稱會變成:

```
C:\Program Files\IBM\MQ\qmgrs\queue!manager
```

- Windows 在具有 FAT 的 Windows 上，佇列管理程式名稱會變成:

```
C:\Program Files\IBM\MQ\qmgrs\queue!ma
```

- Linux AIX 在 AIX and Linux 上，佇列管理程式名稱會變成:

```
/var/mqm/qmgrs/queue!manager
```

在不區分大小寫的檔案系統上，轉換演算法也會區分只有大小寫不同的名稱。

物件名稱轉換

物件名稱不一定是有效的檔案系統名稱。 您可能需要轉換物件名稱。 所使用的方法不同於佇列管理程式名稱的方法，因為雖然每一部機器上只有少數佇列管理程式名稱，但每一部佇列管理程式可能會有大量其他物件。 檔案系統中代表佇列、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。

當轉換程序產生新名稱時，與原始物件名稱沒有簡式關係。 您可以使用 **dspmqfls** 指令，在實際和轉換後的物件名稱之間進行轉換。

相關參考
dspmqfls (顯示檔名)
相關資訊
mqs.ini 檔案的 AllQueue 管理程式段落

IBM i IBM i 上的物件名稱

佇列管理程式具有唯一名稱的相關聯佇列管理程式檔案庫。 佇列管理程式名稱及物件名稱可能需要轉換，以符合 IBM i Integrated File System (IFS) 的需求。

當建立佇列管理程式時， IBM MQ 會使佇列管理程式庫與它產生關聯。 根據使用者定義的佇列管理程式名稱，此佇列管理程式檔案庫會獲得唯一名稱，長度不超過 10 個字元。 佇列管理程式及佇列管理程式檔案庫都會放在目錄中，該目錄也會以字首為 /QIBM/UserData/mqm 的佇列管理程式名稱為基礎。 佇列管理程式、佇列管理程式檔案庫及目錄的範例如下:

佇列管理程式名稱	橙色
佇列管理程式檔案庫名稱	QMORANGE
目錄	/QIBM/UserData/mqm/ORANGE

所有佇列管理程式名稱及佇列管理程式檔案庫名稱都會寫入檔案 /QIBM/UserData/mqm/mqs.ini 中的段落。

IBM MQ IFS 目錄和檔案

IBM MQ 廣泛使用 IBM i Integrated File System (IFS) 來儲存資料。 如需 IFS 的相關資訊，請參閱 *Integrated File System* 簡介。

每一個 IBM MQ 物件 (例如，通道或佇列管理程式) 都由一個檔案代表。 因為物件名稱不一定是有效的檔名，所以佇列管理程式會在必要時將物件名稱轉換成有效的檔名。

佇列管理程式目錄的路徑由下列項目組成:

- 在佇列管理程式配置檔 qm.ini 中定義的字首。 預設字首為 /QIBM/UserData/mqm。
- 文字 qmgrs。
- 編碼佇列管理程式名稱，這是轉換成有效目錄名稱的佇列管理程式名稱。 例如，佇列管理程式 queue/manager 由 queue&manager 代表。

此程序稱為名稱轉換。

IFS 佇列管理程式名稱轉換

在 IBM MQ 中，您可以為佇列管理程式提供最多包含 48 個字元的名稱。

例如，您可以將佇列管理程式命名為 QUEUE/MANAGER/ACCOUNTING/SERVICES。 就像為每一個佇列管理程式建立程式庫一樣，每一個佇列管理程式也會以檔案來代表。 由於 EBCDIC 中的變式字碼點，名稱中可以使用的字元有一些限制。 因此，會自動轉換代表物件的 IFS 檔案名稱，以符合檔案系統的需求。

使用名稱為 queue/manager 的佇列管理程式範例，將字元 / 轉換為 &，並假設預設字首，則 IBM MQ for IBM i 中的佇列管理程式名稱會變成 /QIBM/UserData/mqm/qmgrs/queue&manager。

物件名稱轉換

物件名稱不一定是有效的檔案系統名稱，因此可能需要轉換物件名稱。 所使用的方法不同於佇列管理程式名稱的方法，因為雖然每一部機器只有少數佇列管理程式名稱，但每一部佇列管理程式可能有大量其他物件。 檔案系統中只代表程序定義、佇列和名稱清單; 通道不受這些考量影響。

當轉換程序產生新名稱時，與原始物件名稱沒有簡式關係。 您可以使用 DSPMQMOBJN 指令來檢視 IBM MQ 物件的轉換名稱。

分散式佇列及叢集

分散式佇列作業表示將訊息從一個佇列管理程式傳送至另一個佇列管理程式。 接收端佇列管理程式可以位於相同或另一部機器上; 附近或世界另一端。 它可以在與本端佇列管理程式相同的平台上執行，也可以在 IBM MQ 支援的任何平台上執行。 您可以在分散式佇列環境中手動定義所有連線，也可以建立叢集，並讓 IBM MQ 為您定義許多連線詳細資料。

分散式佇列 (distributed queuing)

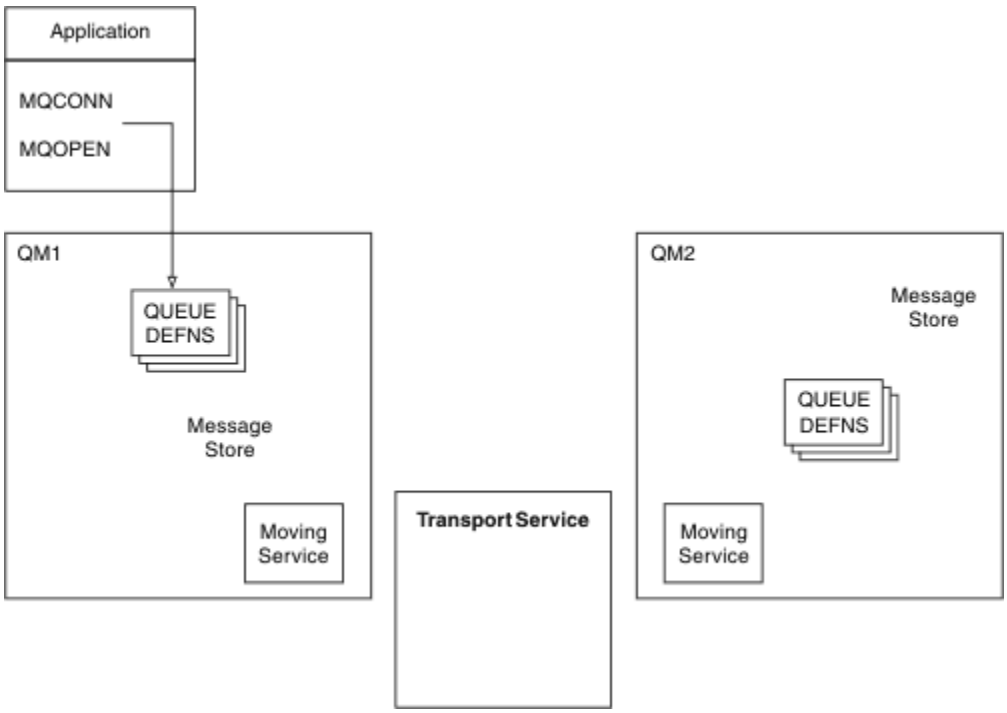


圖 *4:* 分散式佇列的元件概觀

在上圖中:

- 應用程式使用 MQCONN 呼叫來連接至佇列管理程式。 然後，應用程式會使用 MQOPEN 呼叫來開啟佇列，以便它可以將訊息放置在佇列上。
- 每一個佇列管理程式都有其每一個佇列的定義。 它可以具有 本端佇列 (由這個佇列管理程式所管理) 的定義，以及 遠端佇列 (由其他佇列管理程式所管理) 的定義。

- 如果訊息以遠端佇列為目的地，則本端佇列管理程式會將訊息保留在 傳輸佇列上，這會將訊息持續保存在訊息儲存庫中，直到它們可以轉遞至遠端佇列管理程式為止。
- 每一個佇列管理程式都包含佇列管理程式用來與其他佇列管理程式通訊的通訊軟體 (稱為 移動服務)。
- 傳輸服務 與佇列管理程式無關，可以是下列其中一項 (視平台而定):
  - 系統網路架構進階程式對程式通訊 (SNA APPC)
  - 傳輸控制通訊協定/網際網路通訊協定 (Transmission Control Protocol/Internet Protocol, (TCP/IP)
  - 網路基本輸入/輸出系統 (NetBIOS)
  - 循序封包交換 (SPX)

傳送訊息所需的元件

如果要將訊息傳送至遠端佇列管理程式，則本端佇列管理程式需要 傳輸佇列 及 通道的定義。 通道是兩個佇列管理程式之間的單向通訊鏈結。 它可以在遠端佇列管理程式中傳送指定給任意數目佇列的訊息。

通道的每一端都有個別的定義，例如，將它定義為傳送端或接收端。 簡式通道包含本端佇列管理程式中的傳送端 通道定義，以及遠端佇列管理程式中的 接收端 通道定義。 這兩個定義必須具有相同的名稱，且它們一起構成一個通道。

處理訊息傳送及接收的軟體稱為 訊息通道代理程式 (MCA)。 通道的每一端都有一個 訊息通道代理程式 (MCA)。

每一個佇列管理程式都應該有一個 無法傳送郵件的佇列 (也稱為 未遞送訊息佇列)。 如果訊息無法遞送至其目的地，則會將訊息放置在此佇列上。

下圖顯示佇列管理程式、傳輸佇列、通道及 MCA 之間的關係:

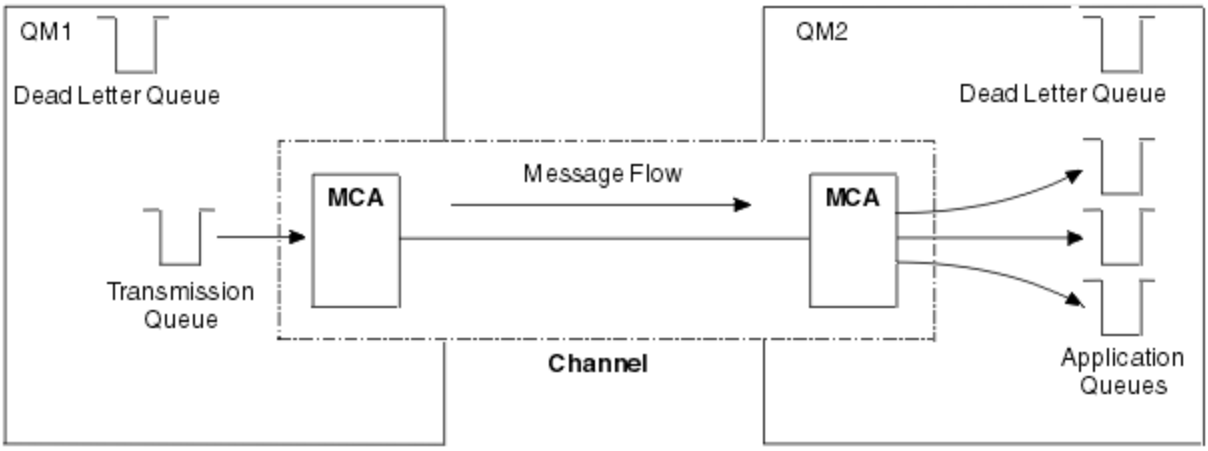


圖 *5:* 傳送訊息

傳回訊息所需的元件

如果您的應用程式需要從遠端佇列管理程式傳回訊息，您需要定義另一個通道，以便在佇列管理程式之間以相反方向執行，如下圖所示:

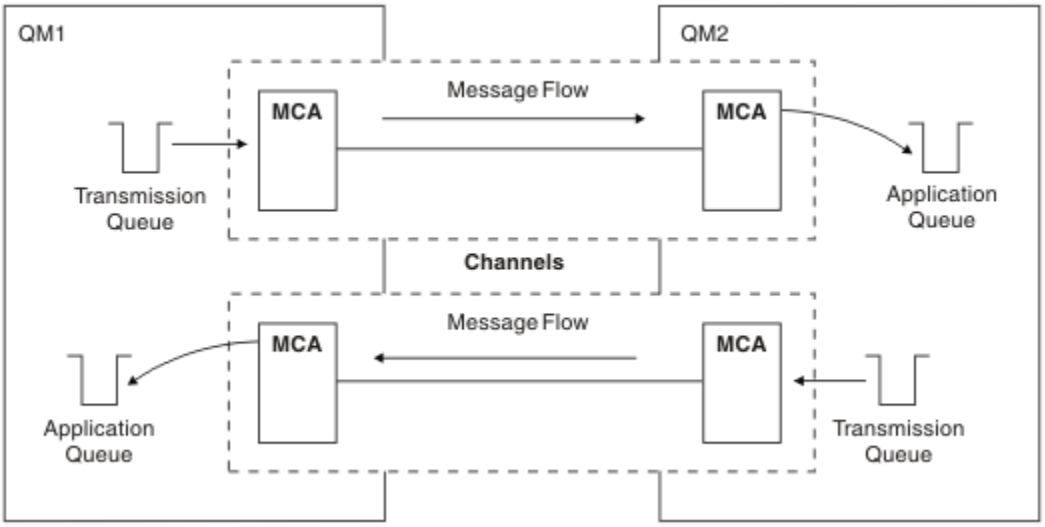


圖 *6:* 雙向傳送訊息

叢集

您可以將叢集中的一組佇列管理程式分組，而不是手動定義分散式佇列環境中的所有連線。 當您這麼做時，佇列管理程式可以讓它們所管理的佇列可供叢集中的其他佇列管理程式使用，而不需要明確通道定義、遠端佇列定義或每一個目的地的傳輸佇列。 叢集中的每個佇列管理程式都有單一傳輸佇列，可將訊息傳輸至叢集中的任何其他佇列管理程式。 對於每一個佇列管理程式，您只需要定義一個叢集接收端通道及一個叢集傳送端通道; 任何其他通道都由叢集自動管理。

IBM MQ 用戶端可以連接至屬於叢集的佇列管理程式，就像它可以連接至任何其他佇列管理程式一樣。 如同使用手動配置的分散式佇列作業，您可以使用 MQPUT 呼叫，將訊息放入任何佇列管理程式中的佇列。 您可以使用 MQGET 呼叫，從本端佇列擷取訊息。

支援叢集的平台上的佇列管理程式不必成為叢集的一部分。 您可以繼續手動配置分散式佇列作業，也可以使用叢集來代替。

使用叢集的優點

叢集作業提供兩個主要好處:

- 叢集可簡化 IBM MQ 網路的管理，這通常需要配置通道、傳輸佇列及遠端佇列的許多物件定義。 在許多佇列管理程式需要交互連接的大型、潛在變更網路中，尤其如此。 此架構特別難以配置及主動維護。
- 叢集可用來在叢集中的佇列及佇列管理程式之間配送訊息資料流量的工作量。 這類配送可讓單一佇列的訊息工作量分散在位於多個佇列管理程式上該佇列的對等實例之間。 此工作量分佈可用來提高系統故障的復原力，並增進系統中特別作用中訊息流程的調整效能。 在這種環境中，分散式佇列的每一個實例都有消費端應用程式處理訊息。 如需相關資訊，請參閱 使用叢集進行工作量管理。

如何在叢集中遞送訊息

您可以將叢集視為由依良心系統管理者維護的佇列管理程式網路。 每當您定義叢集佇列時，系統管理者會視需要在其他佇列管理程式上自動建立對應的遠端佇列定義。

您不需要建立傳輸佇列定義，因為 IBM MQ 會在叢集中的每一個佇列管理程式上提供傳輸佇列。 此單一傳輸佇列可用來將訊息傳送至叢集中的任何其他佇列管理程式。 您不限於使用單一傳輸佇列。 佇列管理程式可以使用多個傳輸佇列來區隔進入叢集中每一個佇列管理程式的訊息。 一般而言，佇列管理程式會使用單一叢集傳輸佇列。 您可以變更佇列管理程式屬性 DEFCLXQ，讓佇列管理程式對叢集中的每一個佇列管理程式使用不同的叢集傳輸佇列。 您也可以手動定義叢集傳輸佇列。

加入叢集的所有佇列管理程式都同意以這種方式運作。 它們會送出其本身及其所管理佇列的相關資訊，並接收叢集其他成員的相關資訊。

為了確保在佇列管理程式變成無法使用時不會遺失任何資訊，您在叢集中指定兩個佇列管理程式作為 完整儲存庫。 這些佇列管理程式會儲存叢集中所有佇列管理程式及佇列的完整相關資訊集。 叢集中的所有其他佇

列管理程式只會儲存與其交換訊息之佇列管理程式及佇列的相關資訊。 這些佇列管理程式稱為 局部儲存庫。 如需相關資訊，請參閱 第 47 頁的『叢集儲存庫』。

為了成為叢集的一部分，佇列管理程式必須有兩個通道: 叢集傳送端通道和叢集接收端通道:

- 叢集傳送端通道是與傳送端通道類似的通訊通道。 您必須在佇列管理程式上手動建立一個叢集傳送端通道，才能將它連接至已是叢集成員的完整儲存庫。
- 叢集接收端通道是類似於接收端通道的通訊通道。 您必須手動建立一個叢集接收端通道。 通道充當佇列管理程式接收叢集通訊的機制。

然後會自動建立此佇列管理程式與叢集其他成員之間通訊所需的所有其他通道。

下圖顯示稱為 CLUSTER 之叢集的元件:

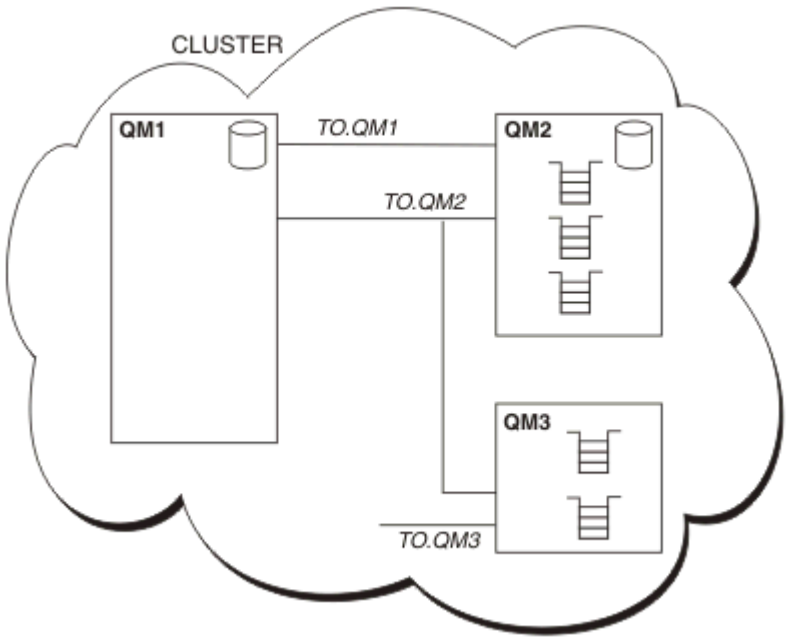


圖 7: 佇列管理程式的叢集

- CLUSTER 包含三個佇列管理程式: QM1、 QM2 及 QM3。
- QM1 及 QM2 會管理叢集中佇列管理程式及佇列相關資訊的完整儲存庫。
- QM2 及 QM3 會管理部分叢集佇列，即叢集中任何其他佇列管理程式可存取的佇列。
- 每一個佇列管理程式都有一個稱為 TO.qmgr 的叢集接收端通道，它可以在該通道上接收訊息。
- 每一個佇列管理程式也都有一個叢集傳送端通道，可將資訊傳送至其中一個儲存庫佇列管理程式。
- QM1 和 QM3 會傳送至 QM2 的儲存庫，而 QM2 會傳送至 QM1 的儲存庫。

分散式佇列元件

分散式佇列的元件包括訊息通道、訊息通道代理程式、傳輸佇列、通道起始程式及接聽器，以及通道結束程式。 訊息通道每一端的定義可以是數種類型之一。

訊息通道是將訊息從一個佇列管理程式傳送至另一個佇列管理程式的通道。 請勿將訊息通道與 MQI 通道混淆。 MQI 通道有兩種類型: 伺服器連線 (SVRCONN) 和用戶端連線 (CLNTCONN)。 如需相關資訊，請參閱 通道。

訊息通道每一端的定義可以是下列其中一種類型:

- 傳送端 (SDR)
- 接收端 (RCVR)
- 伺服器 (SVR)

- 要求者 (RQSTR)
- 叢集傳送端 (CLUSSDR)
- 叢集接收端 (CLUSRCVR)

訊息通道是使用定義在一端的其中一種類型，以及定義在另一端的相容類型來定義。 可能的組合為:

- 傳送端-接收端
- 要求端-伺服器
- 要求端-傳送端（回呼）
- 伺服器-接收端
- 叢集傳送端-叢集接收端

定義通道中包含建立傳送端-接收端通道的詳細指示。 如需設定傳送端-接收端通道所需的參數範例，請參閱適用於您平台的 配置資訊範例 。 如需定義任何類型通道所需的參數，請參閱 DEFINE CHANNEL。

傳送端-接收端通道

一個系統中的傳送端會啟動通道，以便它可以將訊息傳送至另一個系統。 傳送端會要求通道另一端的接收端啟動。 傳送端會將訊息從其傳輸佇列傳送至接收端。 接收端將訊息放置在目的地佇列上。 第 39 頁的圖 8 說明此情況。

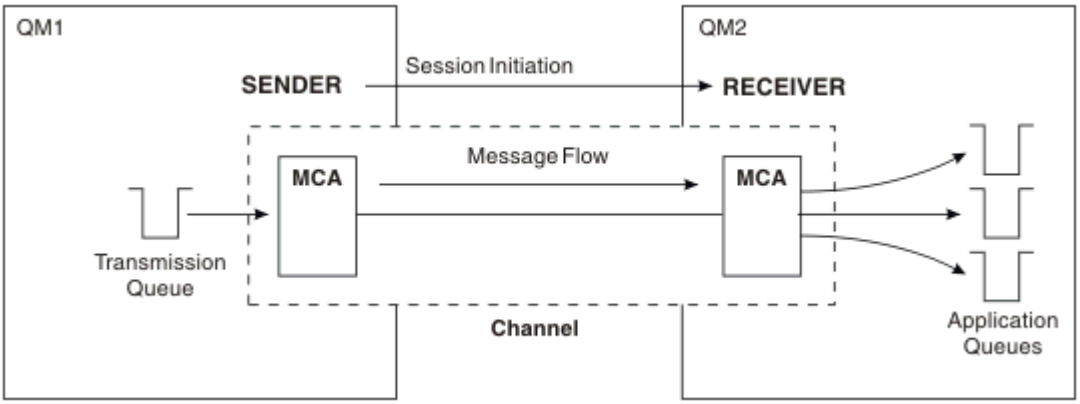


圖 *8:* 傳送端-接收端通道

要求端-伺服器通道

一個系統中的要求端會啟動通道，以便它可以從另一個系統接收訊息。 要求端要求通道另一端的伺服器啟動。 伺服器會從其通道定義中所定義的傳輸佇列，將訊息傳送給要求端。

伺服器通道也可以起始通訊，並將訊息傳送給要求端。 這僅適用於 完整 伺服器，即具有通道定義中所指定友機連線名稱的伺服器通道。 完整伺服器可以由要求者啟動，也可以起始與要求者的通訊。

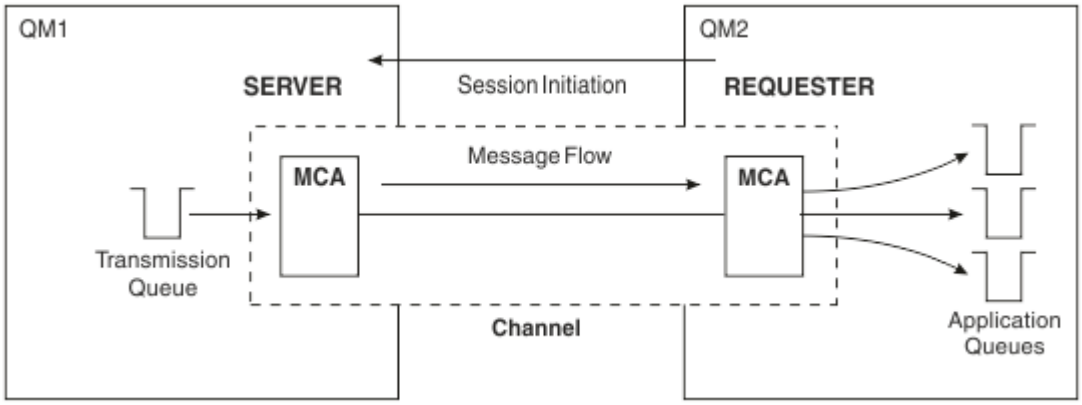


圖 *9:* 要求端-伺服器通道

要求端 - 傳送端通道

要求端會啟動通道，且傳送端會終止呼叫。 然後，傳送端會根據其通道定義中的資訊 (稱為 回呼) 來重新啟動通訊。 它會將訊息從傳輸佇列傳送至要求端。

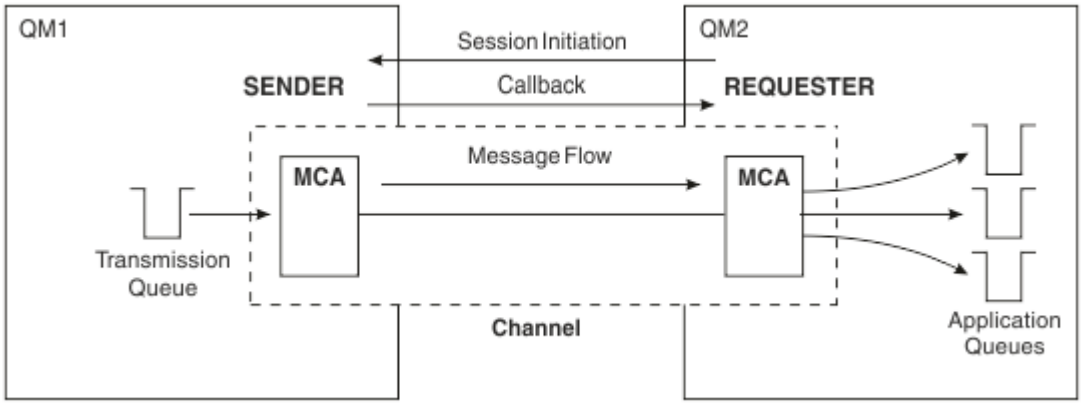


圖 *10:* 要求端-傳送端通道

伺服器-接收端通道

這類似於傳送端-接收端，但僅適用於 完整 伺服器，即具有通道定義中所指定友機連線名稱的伺服器通道。 通道啟動必須在鏈結的伺服器端起始。 此圖的圖解類似於 第 39 頁的圖 8 中的圖解。

叢集傳送端通道

在叢集中，每一個佇列管理程式都有一個叢集傳送端通道，可將叢集資訊傳送至其中一個完整儲存庫佇列管理程式。 佇列管理程式也可以將訊息傳送至叢集傳送端通道上的其他佇列管理程式。

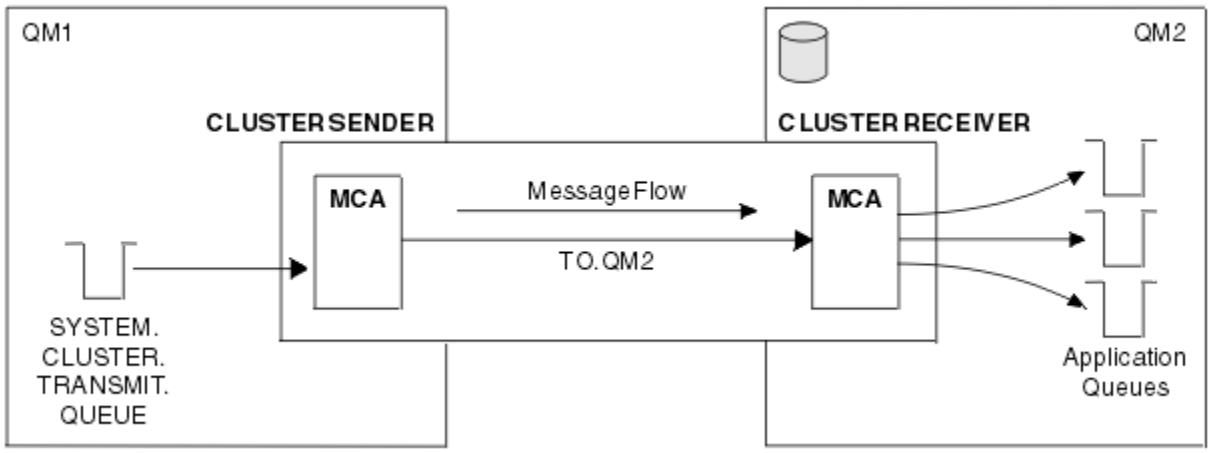


圖 *11:* 叢集傳送端通道

叢集接收端通道

在叢集中，每一個佇列管理程式都有一個叢集接收端通道，它可以在其中接收訊息及叢集的相關資訊。 此圖的圖解類似於 第 41 頁的圖 11 中的圖解。

無法傳送郵件的佇列

無法傳送郵件的佇列 (或無法遞送的訊息佇列) 是訊息無法遞送至正確目的地時傳送至其中的佇列。 每一個佇列管理程式通常都有無法傳送郵件的佇列。

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 (例如，因為佇列不存在或已滿) 之訊息的保留佇列。 通道傳送端也會使用無法傳送郵件的佇列，用於資料轉換錯誤。 網路中的每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起來，以供稍後擷取。

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。 MQDLH 結構的 原因 欄位包含原因碼，可識別訊息在 DLQ 上的原因。

您通常應該為每一個佇列管理程式定義無法傳送郵件的佇列。 如果您沒有，且 MCA 無法放置訊息，則它會留在傳輸佇列中，且通道會停止。 此外，如果快速、非持續訊息 (請參閱 快速、非持續訊息) 無法遞送，且目標系統上沒有無法傳送郵件的佇列，將捨棄這些訊息。

不過，使用無法傳送郵件的佇列可能會影響訊息遞送的順序，因此您可以選擇不使用它們。

相關工作

使用無法傳送郵件的佇列

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)

遠端佇列定義

遠端佇列定義是另一個佇列管理程式所擁有之佇列的定義。

雖然應用程式只能從本端佇列擷取訊息，但可以將訊息放置在本端佇列或遠端佇列上。 因此，除了其每一個本端佇列的定義外，佇列管理程式也可以具有 遠端佇列定義。 遠端佇列定義的優點是它們可讓應用程式將訊息放入遠端佇列，而不需要指定遠端佇列或遠端佇列管理程式的名稱，或傳輸佇列的名稱。 遠端佇列定義提供您位置獨立性。

遠端佇列定義還有其他用途，稍後會說明這些用途。

如何取得遠端佇列管理程式

您可能不會一律在每一個來源與目標佇列管理程式之間有一個通道。 兩者之間還有許多其他連結方式，包括多跳、共用通道、使用不同通道及叢集。

多跳

如果來源佇列管理程式與目標佇列管理程式之間沒有直接通訊鏈結，則在前往目標佇列管理程式的途中，可以通過一或多個 中繼佇列管理程式 。 這稱為 多重中繼站。

您需要定義所有佇列管理程式之間的通道，以及中間佇列管理程式上的傳輸佇列。 這會顯示在 第 42 頁的圖 12 中。

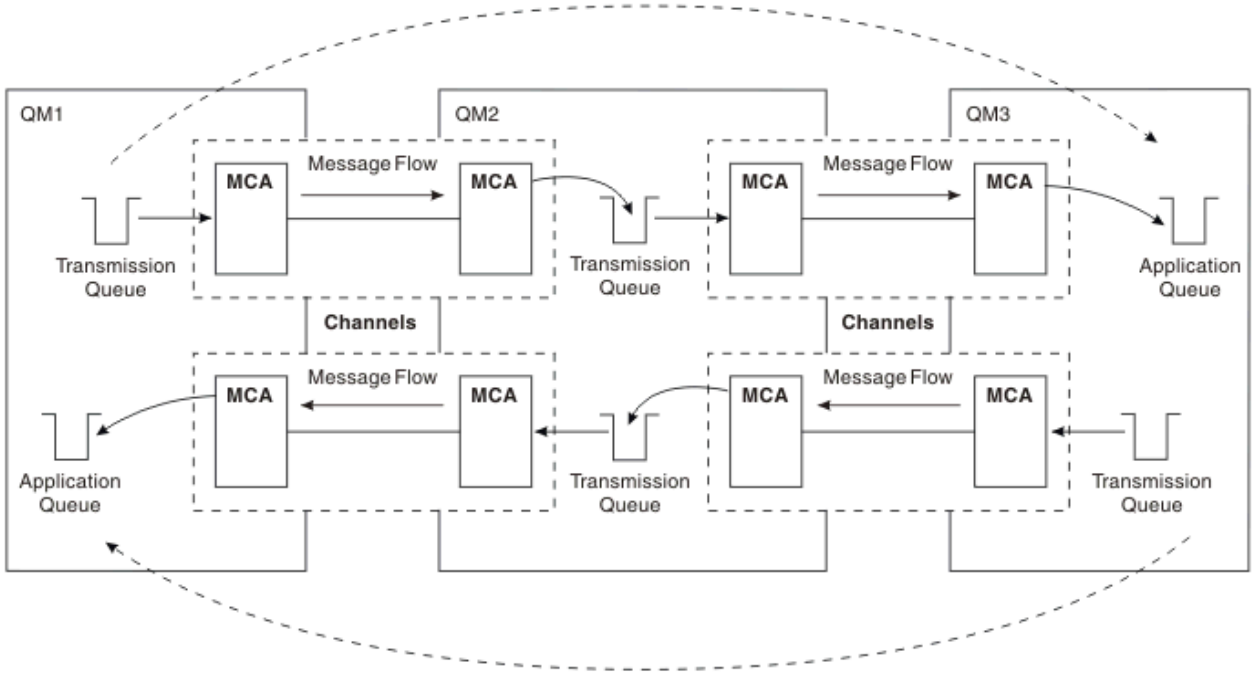


圖 *12:* 通過中繼佇列管理程式

共用通道

作為應用程式設計程式，您可以選擇強制應用程式指定遠端佇列管理程式名稱及佇列名稱，或為每一個遠端佇列建立 遠端佇列定義 。 此定義會保留遠端佇列管理程式名稱、佇列名稱及傳輸佇列的名稱。 無論如何，所有來自相同遠端位置之所有應用程式定址佇列的所有訊息，都會透過相同的傳輸佇列傳送其訊息。 這會顯示在 第 42 頁的圖 13 中。

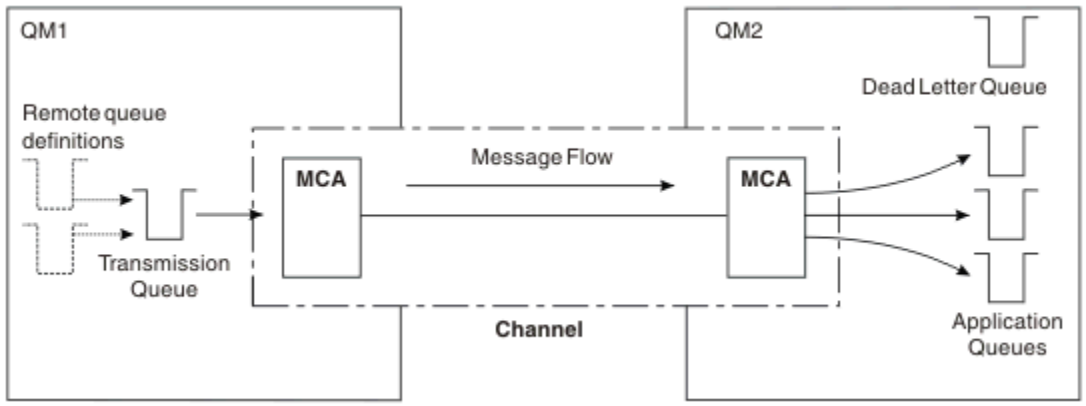


圖 *13:* 共用傳輸佇列

第 42 頁的圖 13 說明從多個應用程式到多個遠端佇列的訊息可以使用相同的通道。

使用不同的通道

如果您有不同類型的訊息要在兩個佇列管理程式之間傳送，則可以在兩個佇列管理程式之間定義多個通道。有時您需要替代通道 (可能為了安全起見)，或與龐大的訊息資料流量交換遞送速度。

若要設定第二個通道，您需要定義另一個通道及另一個傳輸佇列，並建立遠端佇列定義，以指定位置及傳輸佇列名稱。 然後您的應用程式可以使用任一通道，但訊息仍會遞送至相同的目標佇列。 這會顯示在 第 43 頁的圖 14 中。

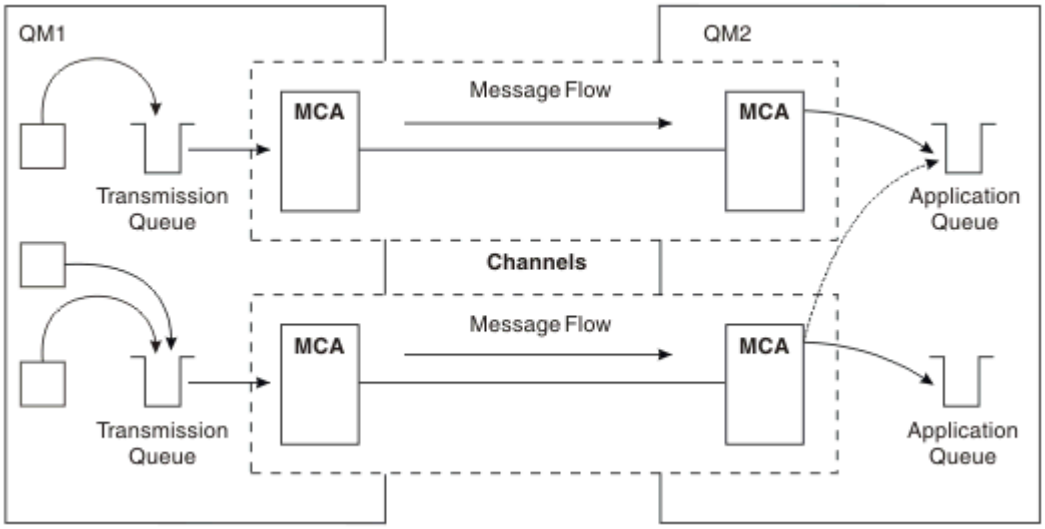


圖 *14:* 使用多個通道

當您使用遠端佇列定義來指定傳輸佇列時，您的應用程式 **不得** 自行指定位置 (即目的地佇列管理程式)。 如果有的話，佇列管理程式不會使用遠端佇列定義。 遠端佇列定義提供您位置獨立性。 應用程式可以在不知道佇列位置的情況下將訊息放置到 邏輯 佇列中，並且您可以變更 實體 佇列，而不需要變更應用程式。

使用叢集作業

叢集內的每個佇列管理程式都會定義叢集接收端通道。 當另一個佇列管理程式想要傳送訊息至該佇列管理程式時，它會自動定義對應的叢集傳送端通道。 例如，如果叢集中有一個佇列的多個實例，則叢集傳送端通道可以定義給管理該佇列的任何佇列管理程式。 IBM MQ 使用工作量管理演算法，該演算法使用循環式常式來選取要將訊息遞送至其中的可用佇列管理程式。 如需相關資訊，請參閱 叢集。

定址資訊

當應用程式放置以遠端佇列管理程式為目的地的訊息時，本端佇列管理程式會將傳輸標頭新增至它們，然後再將它們放置在傳輸佇列上。 此標頭包含目的地佇列及佇列管理程式的名稱，即 定址資訊。

在單一佇列管理程式環境中，當應用程式開啟佇列來放置訊息時，會建立目的地佇列的位址。 因為目的地佇列位於相同的佇列管理程式上，所以不需要任何定址資訊。

在分散式佇列環境中，佇列管理程式不僅需要知道目的地佇列名稱，還需要知道該佇列的位置 (即佇列管理程式名稱)，以及該遠端位置的路徑 (即傳輸佇列)。 此定址資訊包含在傳輸標頭中。 接收通道會移除傳輸標頭，並使用其中的資訊來尋找目的地佇列。

如果您使用遠端佇列定義，則可以避免應用程式需要指定目的地佇列管理程式的名稱。 此定義指定遠端佇列的名稱、訊息目的地的遠端佇列管理程式名稱，以及用來傳輸訊息的傳輸佇列名稱。

何謂別名?

別名用來提供訊息的服務品質。 佇列管理程式別名可讓系統管理者變更目標佇列管理程式的名稱，而不會導致您必須變更應用程式。 它還可讓系統管理者變更目的地佇列管理程式的路徑，或設定涉及通過許多其他佇列管理程式 (多跳) 的路徑。 回覆目的地佇列別名提供回覆的服務品質。

佇列管理程式別名及回覆目的地佇列別名是使用具有空白 RNAME 的遠端佇列定義來建立。 這些定義不會定義實際佇列; 佇列管理程式會使用它們來解析實體佇列名稱、佇列管理程式名稱及傳輸佇列。

別名定義的特徵是具有空白 RNAME。

佇列名稱解析

每次開啟佇列時，每個佇列管理程式都會進行佇列名稱解析。 其目的是識別目標佇列、目標佇列管理程式 (可能是本端) ，以及該佇列管理程式的路徑 (可能是空值)。 已解析的名稱有三個部分: 佇列管理程式名稱、佇列名稱，以及傳輸佇列 (如果佇列管理程式在遠端)。

當遠端佇列定義存在時，不會參照別名定義。 應用程式提供的佇列名稱會解析為遠端佇列定義中指定的目的地佇列、遠端佇列管理程式及傳輸佇列的名稱。 如需佇列名稱解析的詳細資訊，請參閱 佇列名稱解析。

如果沒有遠端佇列定義，且已指定佇列管理程式名稱，或由名稱服務解析，則佇列管理程式會查看是否有佇列管理程式別名定義符合所提供的佇列管理程式名稱。 如果有，則會使用其中的資訊將佇列管理程式名稱解析為目的地佇列管理程式的名稱。 佇列管理程式別名定義也可以用來決定傳送至目的地佇列管理程式的傳輸佇列。

如果解析的佇列名稱不是本端佇列，則佇列管理程式名稱及佇列名稱都會併入應用程式放置至傳輸佇列之每一則訊息的傳輸標頭中。

除非遠端佇列定義或佇列管理程式別名定義變更，否則所使用的傳輸佇列通常與已解析的佇列管理程式同名。 如果您尚未定義此類傳輸佇列，但已定義預設傳輸佇列，則會使用此傳輸佇列。

z/OS 在 z/OS 上執行的佇列管理程式名稱限制為四個字元。

佇列管理程式別名定義

當應用程式開啟佇列以放置訊息、指定佇列名稱 **及** 佇列管理程式名稱時，會套用佇列管理程式別名定義。

佇列管理程式別名定義有三種用法:

- 傳送訊息時，重新對映佇列管理程式名稱
- 傳送訊息時，變更或指定傳輸佇列
- 接收訊息時，判斷本端佇列管理程式是否為那些訊息的預期目的地

出埠訊息-重新對映佇列管理程式名稱

佇列管理程式別名定義可用來重新對映 MQOPEN 呼叫中指定的佇列管理程式名稱。 例如， MQOPEN 呼叫指定佇列名稱 THISQ 和佇列管理程式名稱 YOURQM。 在本端佇列管理程式中，有一個類似下列範例的佇列管理程式別名定義:

```
DEFINE QREMOTE (YOURQM) RQMNAME(REALQM)
```

這會顯示當應用程式將訊息放入佇列管理程式 YOURQM 時，要使用的實際佇列管理程式是 REALQM。 如果本端佇列管理程式為 REALQM，則會將訊息放入佇列 THISQ(本端佇列)。 如果本端佇列管理程式不是稱為 REALQM，它會將訊息遞送至稱為 REALQM 的傳輸佇列。 佇列管理程式會將傳輸標頭變更為 REALQM ，而非 YOURQM。

出埠訊息-變更或指定傳輸佇列

第 45 頁的圖 15 顯示訊息到達佇列管理程式 QM1 的實務範例，其中傳輸標頭顯示佇列管理程式中的佇列名稱 QM3。 在此實務範例中，透過 QM2 可透過多個跳躍來呼叫到 QM3 。

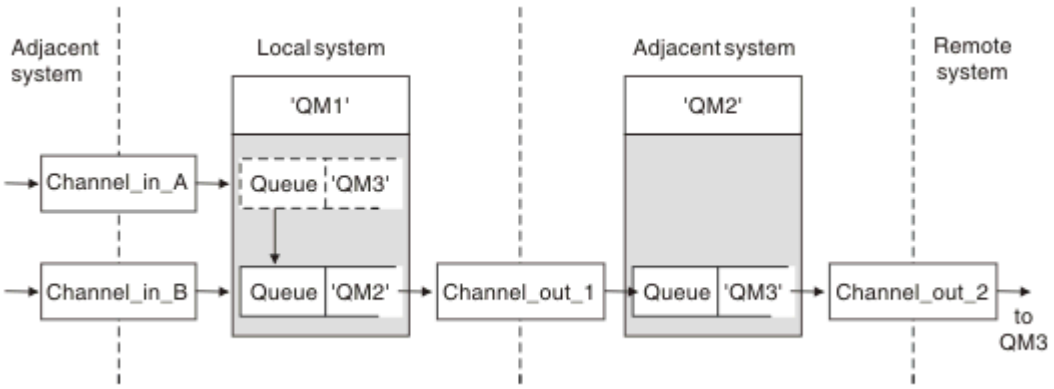


圖 *15:* 佇列管理程式別名

QM3 的所有訊息都會以佇列管理程式別名在 QM1 中擷取。 佇列管理程式別名名為 QM3 ， 且包含 QM3 透過傳輸佇列 QM2 的定義。 定義類似下列範例:

```
DEFINE QREMOTE (QM3) RNAME(' ') RQMNAME(QM3) XMITQ(QM2)
```

佇列管理程式會將訊息放入傳輸佇列 QM2 ， 但不會變更傳輸佇列標頭，因為目的地佇列管理程式 QM3 的名稱不會變更。

所有到達 QM1 並顯示包含佇列名稱 (位於 QM2) 的傳輸標頭的訊息，也會放在 QM2 傳輸佇列上。 以此方式，將具有不同目的地的訊息收集到共用傳輸佇列至適當的相鄰系統，以向前傳輸至其目的地。

入埠訊息-決定目的地

接收 MCA 會開啟傳輸標頭中所參照的佇列。 如果存在與所參照的佇列管理程式同名的佇列管理程式別名定義，則會以該定義中的 RQMNAME 取代傳輸標頭中所收到的佇列管理程式名稱。

此處理程序有兩個用途:

- 將訊息導向至另一個佇列管理程式
- 將佇列管理程式名稱變更為與本端佇列管理程式相同

回覆目的地佇列別名定義

回覆目的地佇列別名定義會在訊息描述子中指定回覆資訊的替代名稱。 其優點是您可以變更佇列或佇列管理程式的名稱，而不需要變更應用程式。

佇列名稱解析

當應用程式回覆訊息時，它會使用所收到訊息的 訊息描述子 中的資料來找出要回覆的佇列名稱。 傳送端應用程式會指出回覆傳送至何處，並將此資訊附加至其訊息。 此概念必須作為應用程式設計的一部分進行協調。

在將訊息放入佇列之前，會在應用程式傳送端進行佇列名稱解析。 因此，在與訊息傳送至的遠端應用程式互動之前，會先進行佇列名稱解析。 這是唯一在未開啟佇列時進行名稱解析的狀況。

使用佇列管理程式別名解析佇列名稱

通常應用程式會指定回覆目的地佇列，並將回覆目的地佇列管理程式名稱留白。 佇列管理程式會在放置時間完成它自己的名稱。 除非您想要將替代通道用於回覆 (例如，使用傳輸佇列 QM1_relief 的通道，而不是使用傳輸佇列 QM1 的預設傳回通道) ， 否則此方法運作良好。 在此狀況下，傳輸佇列標頭中指定的佇列管理程式名稱不符合「實際」佇列管理程式名稱，但會使用佇列管理程式別名定義重新指定。 為了沿著替代路徑傳回回覆，也必須使用回覆目的地佇列別名定義來對映回覆目的地佇列資料。

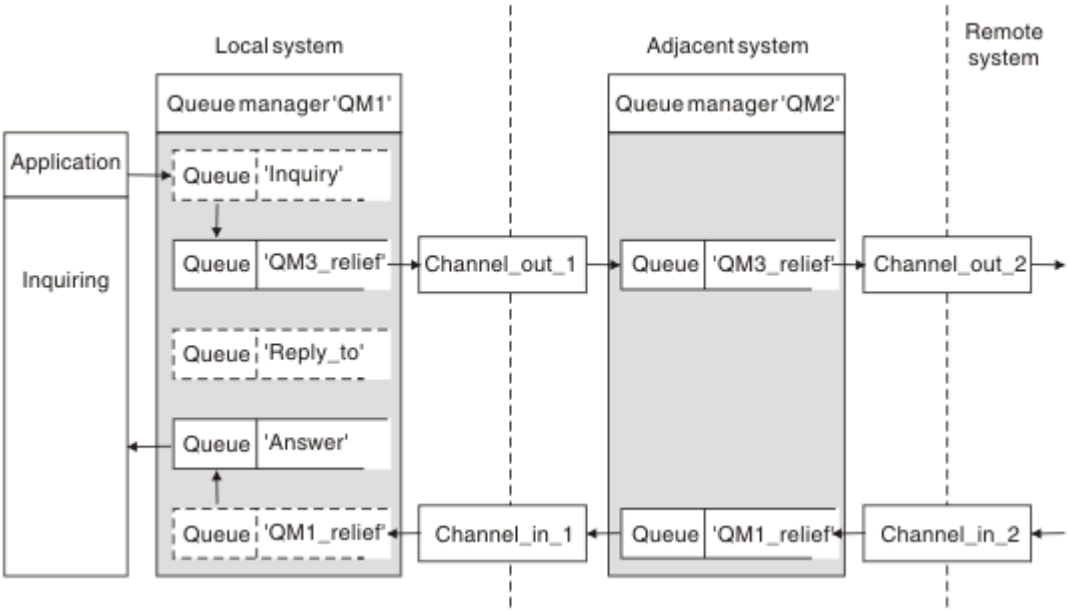


圖 *16:* 用於變更回覆位置的回覆目的地佇列別名

在 第 46 頁的圖 16 的範例中:

1. 應用程式會使用 MQPUT 呼叫並在訊息描述子中指定下列資訊來放置訊息:

   ```
   ReplyToQ='Reply_to'
   ReplyToQMgr=' '
   ```

   ReplyTo 佇列管理程式必須空白，才能使用回覆目的地佇列別名。
2. 您可以建立名為 `Reply_to` 的回覆目的地佇列別名定義，其中包含名稱 `Answer` 及佇列管理程式名稱 `QM1_relief`。

   ```
   DEFINE QREMOTE ('Reply_to') RNAME ('Answer')
   RQMNAME ('QM1_relief')
   ```
3. 訊息會隨顯示 `ReplyToQ='Answer'` 和 `ReplyToQMgr='QM1_relief'`的訊息描述子一起傳送。
4. 應用程式規格必須包含要在佇列 `Answer` 而非 `Reply_to` 中找到回覆的資訊。

若要準備回覆，您必須建立平行傳回通道，定義:

- 在 QM2，名為 `QM1_relief` 的傳輸佇列

  ```
  DEFINE QLOCAL ('QM1_relief') USAGE(XMITQ)
  ```
- 在 QM1，佇列管理程式別名 `QM1_relief`

  ```
  DEFINE QREMOTE ('QM1_relief') RNAME() RQMNAME(QM1)
  ```

  此佇列管理程式別名會終止平行傳回通道的鏈結，並擷取 QM1 的訊息。

如果您認為您可能想要在未來某個時間執行此動作，請確保應用程式從一開始就使用別名。 目前這是回覆目的地佇列的一般佇列別名，但稍後可以變更為佇列管理程式別名。

回覆目的地佇列名稱

命名回覆目的地佇列時需要小心。 應用程式在訊息中放置回覆目的地佇列名稱的原因是它可以指定其回覆傳送至的佇列。 當您建立具有此名稱的回覆目的地佇列別名定義時，您無法具有具有相同名稱的實際回覆目的地佇列 (即本端佇列定義)。 因此，回覆目的地佇列別名定義必須包含新的佇列名稱及佇列管理程式名稱，且應用程式規格必須包含在此其他佇列中找到其回覆的資訊。

現在，當應用程式放置原始訊息時，它們必須從不同佇列中擷取訊息，該佇列不同於它們命名為回覆目的地佇列的佇列。

叢集元件

叢集由佇列管理程式、叢集儲存庫、叢集通道及叢集佇列組成。

如需每一個叢集元件的相關資訊，請參閱下列子主題:

相關概念

叢集作業與分散式佇列作業的比較

相關工作

配置佇列管理程式叢集

設定新的叢集

叢集儲存庫

儲存庫是屬於叢集成員之佇列管理程式的相關資訊集合。

儲存庫資訊包括佇列管理程式名稱、其位置、其通道、其管理的佇列，以及其他資訊。 資訊以訊息形式儲存在稱為 SYSTEM.CLUSTER.REPOSITORY.QUEUE 的佇列中。 此佇列是其中一個預設物件。 Multi 在 多平台上，當您建立 IBM MQ 佇列管理程式時，會定義它。 z/OS 在 IBM MQ for z/OS 上，它定義為佇列管理程式自訂作業的一部分。

完整儲存庫和局部儲存庫

通常，叢集中的兩個佇列管理程式會保留完整儲存庫。 其餘佇列管理程式都會保留局部儲存庫。

管理叢集中每個佇列管理程式的完整資訊集的佇列管理程式具有完整儲存庫。 叢集中的其他佇列管理程式具有局部儲存庫，其中包含完整儲存庫中的部分資訊。

局部儲存庫只包含佇列管理程式需要與其交換訊息之那些佇列管理程式的相關資訊。 佇列管理程式會要求更新它們所需的資訊，因此如果它變更，完整儲存庫佇列管理程式會傳送新資訊給它們。 在大部分時間內，局部儲存庫包含佇列管理程式在叢集內執行所需的所有資訊。 當佇列管理程式需要某些其他資訊時，它會查詢完整儲存庫，再更新其局部儲存庫。 佇列管理程式會使用 SYSTEM.CLUSTER.COMMAND.QUEUE 佇列來要求及接收儲存庫的更新項目。

當 移轉 屬於叢集成員的佇列管理程式時，請在局部儲存庫之前移轉完整儲存庫。 這是因為較舊的儲存庫無法儲存較新版本中引進的較新屬性。 它容忍它們，但不儲存它們。

叢集佇列管理程式

叢集佇列管理程式是屬於叢集成員的佇列管理程式。

佇列管理程式可以是多個叢集的成員。 每一個叢集佇列管理程式在其所屬的所有叢集中都必須具有唯一的名稱。

叢集佇列管理程式可以管理向叢集中其他佇列管理程式通告的佇列。 不過，它不需要這樣做。 它可以改為將訊息饋送至叢集中其他位置所管理的佇列，並只接收明確導向至它的回應。

z/OS 在 IBM MQ for z/OS 中，叢集佇列管理程式可以是佇列共用群組的成員。 在此情況下，它會與相同佇列共用群組中的其他佇列管理程式共用其佇列定義。

叢集佇列管理程式是自主的。 他們可以完全控制他們定義的佇列和通道。 其他佇列管理程式 (相同佇列共用群組中的佇列管理程式除外) 無法修改其定義。 儲存庫佇列管理程式不會控制叢集中其他佇列管理程式中的定義。 它們會保留一組完整的所有定義，以便在必要時使用。 叢集是佇列管理程式的聯合。

在叢集佇列管理程式上建立或變更定義之後，會將資訊傳送至完整儲存庫佇列管理程式。 稍後會更新叢集中的其他儲存庫。

完整儲存庫佇列管理程式

完整儲存庫佇列管理程式是一種叢集佇列管理程式，可保留叢集資源的完整呈現。 為了確保可用性，請在每一個叢集中設定兩個以上完整儲存庫佇列管理程式。 完整儲存庫佇列管理程式會接收叢集中其他佇列管理程式所傳送的資訊，並更新其儲存庫。 它們會彼此傳送訊息，以確保它們都保持最新的叢集相關新資訊。

佇列管理程式和儲存庫

每個叢集具有至少一個 (最好是兩個) 佇列管理程式，其中包含叢集中佇列管理程式、佇列及通道相關資訊的完整儲存庫。 這些儲存庫也包含來自叢集中其他佇列管理程式的要求，以更新資訊。

其他佇列管理程式每一個都保留局部儲存庫，其中包含需要與其通訊之佇列及佇列管理程式子集的相關資訊。 佇列管理程式會在第一次需要存取另一個佇列或佇列管理程式時進行查詢，以建置其局部儲存庫。 他們要求收到有關該佇列或佇列管理程式的任何新資訊通知。

每一個佇列管理程式都會將其儲存庫資訊儲存在稱為 SYSTEM.CLUSTER.REPOSITORY.QUEUE 的佇列上的訊息中。 佇列管理程式會在稱為 SYSTEM.CLUSTER.COMMAND.QUEUE 的佇列上交換訊息中的儲存庫資訊。

每一個加入叢集的佇列管理程式都會定義一個叢集傳送端 CLUSSDR 通道至其中一個儲存庫。 它會立即得知叢集中哪些其他佇列管理程式保留完整儲存庫。 從那時開始，佇列管理程式可以從任何儲存庫要求資訊。 當佇列管理程式將資訊傳送至選擇的儲存庫時，它也會將資訊傳送至另一個儲存庫 (如果有的話)。

當管理完整儲存庫的佇列管理程式從鏈結至它的其中一個佇列管理程式接收新資訊時，會更新完整儲存庫。 新資訊也會傳送至另一個儲存庫，以減少儲存庫佇列管理程式無法運作時延遲它的風險。 因為所有資訊都會傳送兩次，所以儲存庫必須捨棄重複項目。 每一個資訊項目都有一個序號，供儲存庫用來識別重複項目。 透過交換訊息，所有儲存庫彼此保持一致。

叢集佇列數

叢集佇列是由叢集佇列管理程式所管理的佇列，並可提供給叢集的其他佇列管理程式使用。

叢集佇列定義會通告至叢集中的其他佇列管理程式。 叢集中的其他佇列管理程式不需要相對應的遠端佇列定義，就可以將訊息放置在叢集佇列中。 可以使用叢集名稱清單在多個叢集中通告叢集佇列。

在通告佇列時，叢集中的任何佇列管理程式都可以將訊息置入該佇列中。 若要放置訊息，佇列管理程式必須從完整儲存庫中找出管理該佇列的位置。 然後，它會將一些遞送資訊新增到訊息中，並將訊息放置在叢集傳輸佇列上。

z/OS 在 IBM MQ for z/OS 中，叢集佇列可以是佇列共用群組的成員共用的佇列。

相關工作

定義叢集佇列

共用佇列與叢集佇列之間的比較

此資訊旨在協助您比較共用佇列和叢集佇列，並決定哪些可能更適合您的系統。

通道起始程式成本

在叢集佇列中，訊息是由通道傳送，因此除了應用程式成本之外，還容許通道起始程式成本。 網路中有一些成本，因為通道會取得及放置訊息。 共用佇列不存在這些成本，因此在佇列共用群組中的佇列管理程式之間移動訊息時，使用的處理能力會比叢集佇列少。

訊息可用性

當放入佇列時，叢集佇列會將訊息傳送至其中一個佇列管理程式，且作用中通道會連接至您的佇列管理程式。 在遠端佇列管理程式上，如果用來處理訊息的應用程式無法運作，則不會處理訊息，並等待應用程式啟動。 同樣地，如果佇列管理程式已關閉，則在佇列管理程式重新啟動之前，佇列管理程式上的任何訊息都無法使用。 這些實例顯示的訊息可用性低於使用共用佇列時。

使用共用佇列時，佇列共用群組中的任何應用程式都可以取得傳送的訊息。 如果您關閉佇列共用群組中的一個佇列管理程式，則訊息可供其他佇列管理程式使用，提供比使用叢集佇列更高的訊息可用性。

容量

連結機能比磁碟更貴; 因此，將 1,000,000 則訊息儲存在本端佇列的成本低於具有足夠容量來儲存相同訊息數的連結機能。

傳送至其他佇列管理程式

共用佇列訊息只能在佇列共用群組內使用。 如果您要使用佇列共用群組之外的佇列管理程式，則必須使用通道。 您可以使用叢集作業來平衡多個遠端分散式佇列管理程式之間的工作量。

工作量平衡

您可以使用叢集作業來提供加權，讓哪些通道及佇列管理程式取得所傳送訊息的比例。 例如，您可以將 60% 的訊息傳送至一個佇列管理程式，並將 40% 的訊息傳送至另一個佇列管理程式。 此實例不取決於遠端佇列管理程式處理工作的能力。 具有第一個佇列管理程式的系統可能超載，且具有第二個佇列管理程式的系統可能閒置，但大部分訊息仍會移至第一個佇列管理程式。

使用共用佇列，兩個 CICS 系統可以取得訊息。 如果其中一個系統超載，則另一個系統會接管大部分工作量。

叢集通道

在每個完整儲存庫上，您可以手動定義叢集接收端通道及一組叢集傳送端通道，以連接至叢集中的所有其他完整儲存庫。 當您新增局部儲存庫時，您可以手動定義叢集接收端通道，以及連接至其中一個完整儲存庫的單一叢集傳送端通道。 叢集會在需要時自動定義進一步的叢集傳送端通道。 自動定義的叢集傳送端通道會從接收端佇列管理程式上對應的叢集接收端通道定義取得其屬性。

叢集接收端通道: CLUSRCVR

CLUSRCVR 通道定義定義通道的結尾，叢集佇列管理程式可以在此通道上接收來自叢集中其他佇列管理程式的訊息。

您必須為每一個叢集佇列管理程式至少定義一個 CLUSRCVR 通道。 透過定義 CLUSRCVR 通道，佇列管理程式會顯示可用來接收訊息的其他叢集佇列管理程式。

CLUSRCVR 通道定義也可讓其他佇列管理程式自動定義對應的叢集傳送端通道定義。 請參閱本文的 第 50 頁的『自動定義叢集傳送端通道』 一節。

叢集傳送端通道: CLUSSDR

您手動定義從每個完整儲存庫佇列管理程式到叢集中所有其他完整儲存庫佇列管理程式的 CLUSSDR 通道。 完整儲存庫所交換的所有更新項目都只會在這些通道上流動。 透過手動定義這些通道，您可以明確地控制完整儲存庫的網路。

當您將局部儲存庫佇列管理程式新增至叢集時，請手動定義單一 CLUSSDR 通道，以連接至其中一個完整儲存庫。 您選擇的完整儲存庫差異不大，因為在進行起始聯絡之後，會根據需要自動定義佇列管理程式的進一步叢集佇列管理程式物件 (包括 CLUSSDR 通道)。 這可讓您的佇列管理程式將叢集資訊傳送至任何完整儲存庫，並將訊息傳送至叢集中的任何佇列管理程式。

如本文的一節所述，自動定義的傳送端通道是根據叢集接收端通道的配置。 因此，您在叢集通道上設定的任何通道內容應該在相符的 CLUSSDR 及叢集接收端通道上相同設定，或只在叢集接收端通道上設定。

基於先前說明的原因，您應該只手動定義 CLUSSDR 通道。 亦即，起始將局部儲存庫連接至完整儲存庫，或將兩個完整儲存庫連接在一起。 手動配置連接至局部儲存庫或非叢集中的佇列管理程式的 CLUSSDR 通道會導致發出錯誤訊息，例如 AMQ9427 及 AMQ9428 。 雖然有時這可能無法避免作為暫時狀況，例如在修改完整儲存庫的位置時，應該儘快刪除手動定義。

自動定義叢集傳送端通道

一般而言，當您將局部儲存庫佇列管理程式新增至叢集時，您只會在佇列管理程式上手動定義兩個叢集通道:

- 叢集傳送端 (CLUSSDR) channel to a full repository queue manager for the cluster.
- 叢集接收端 (CLUSRCVR) 通道。

您定義的 CLUSSDR 通道可讓佇列管理程式與叢集進行起始聯絡。 在進行起始聯絡之後，叢集會在需要時自動定義進一步的 CLUSSDR 通道。

自動定義的 CLUSSDR 通道會從接收端佇列管理程式上對應的 CLUSRCVR 通道定義取得其屬性。 即使有手動定義的 CLUSSDR 通道，也會使用自動定義的 CLUSSDR 通道中的屬性。 例如，假設您定義 CLUSRCVR 通道，但未在 **CONNAME** 參數中指定埠號，並手動定義 CLUSSDR 通道，且該通道指定埠號。 當自動定義的 CLUSSDR 通道取代手動定義的通道時，埠號 (取自 CLUSRCVR 通道) 會變成空白。 使用預設埠號且通道失敗。

如果手動定義的 CLUSSDR 通道與對應的 CLUSRCVR 通道定義之間有配置差異，則部分差異會立即生效 (例如，工作量平衡參數) ，而部分差異只會在通道重新啟動時生效 (例如， TLS 配置)。

為避免混淆，請盡可能遵守下列準則:

- 僅手動定義 CLUSSDR 通道以指向完整儲存庫。
- 如果您已手動定義 CLUSSDR 通道，請將它們配置成完全符合接收佇列管理程式上對應的 CLUSRCVR 通道定義。

另請參閱 使用自動定義通道。

相關概念
使用自動定義的通道
使用叢集傳輸佇列及叢集傳送端通道
相關工作
設定新的叢集
將佇列管理程式加入叢集中

叢集主題

叢集主題是已定義 **cluster** 屬性的管理主題。 叢集主題的相關資訊會推送給叢集的所有成員，並與本端主題相結合，以建立橫跨多個佇列管理程式的主題空間的某些部分。 這可讓一個佇列管理程式中的某個主題上發佈的訊息傳遞至叢集中其他佇列管理程式的訂閱。

在佇列管理程式上定義叢集主題時，叢集主題定義會傳送到完整儲存庫佇列管理程式。 然後，完整儲存庫會將叢集主題定義傳送到叢集內的所有佇列管理程式，使得相同的叢集主題可供叢集中任何佇列管理程式的發佈者和訂閱者使用。 您在其中建立叢集主題的佇列管理程式稱為叢集主題主機。 叢集主題可供叢集中的任何佇列管理程式使用，但是對叢集主題所做的任何修改都必須在定義該主題的佇列管理程式（主機）上進行，此時所做的修改會透過完整儲存庫傳送給叢集的所有成員。

如需配置叢集主題以使用 直接遞送 或 主題主機遞送的相關資訊，以及叢集主題繼承和萬用字元訂閱的相關資訊，請參閱 定義叢集主題。

如需用來顯示叢集主題的指令的相關資訊，請參閱相關資訊。

相關概念
使用管理主題
使用訂閱
相關參考
DISPLAYTOPIC
DISPLAYTPSTATUS

DISPLAYSUB

預設叢集物件

Multi 在 Multiplatforms 上，當您定義佇列管理程式時，預設叢集物件會併入自動建立的預設物件集。 z/OS 在 z/OS 上，可以在自訂作業範例中找到預設叢集物件定義。

註: 您可以執行 MQSC 或 PCF 指令，以與任何其他通道定義相同的方式變更預設通道定義。 請勿變更預設佇列定義， SYSTEM.CLUSTER.HISTORY.QUEUE 除外。

SYSTEM.CLUSTER.COMMAND.QUEUE
叢集中的每一個佇列管理程式都有一個稱為 SYSTEM.CLUSTER.COMMAND.QUEUE 的本端佇列，用來將訊息傳送至完整儲存庫。 此訊息包含任何新的或已變更的佇列管理程式相關資訊，或其他佇列管理程式相關資訊的任何要求。 SYSTEM.CLUSTER.COMMAND.QUEUE 通常是空的。

SYSTEM.CLUSTER.HISTORY.QUEUE
叢集中的每一個佇列管理程式都有一個稱為 SYSTEM.CLUSTER.HISTORY.QUEUE 的本端佇列。 SYSTEM.CLUSTER.HISTORY.QUEUE 用來儲存叢集狀態資訊的歷程，以作為服務用途。

在預設物件設定中， SYSTEM.CLUSTER.HISTORY.QUEUE 設為 PUT (ENABLED)。 若要暫停歷程收集，請將設定變更為 PUT (DISABLED)。

SYSTEM.CLUSTER.REPOSITORY.QUEUE
叢集中的每一個佇列管理程式都有一個稱為 SYSTEM.CLUSTER.REPOSITORY.QUEUE 的本端佇列。 此佇列用來儲存所有完整儲存庫資訊。 此佇列通常不是空的。

SYSTEM.CLUSTER.TRANSMIT.QUEUE
每一個佇列管理程式都有一個稱為 SYSTEM.CLUSTER.TRANSMIT.QUEUE 的本端佇列定義。 SYSTEM.CLUSTER.TRANSMIT.QUEUE 是所有訊息到叢集內所有佇列及佇列管理程式的預設傳輸佇列。 您可以變更佇列管理程式屬性 DEFCLXQ ，將每一個叢集傳送端通道的預設傳輸佇列變更為 SYSTEM.CLUSTER.TRANSMIT. *ChannelName*。 您無法刪除 SYSTEM.CLUSTER.TRANSMIT.QUEUE。 它也用來定義授權檢查所使用的預設傳輸佇列是 SYSTEM.CLUSTER.TRANSMIT.QUEUE 還是 SYSTEM.CLUSTER.TRANSMIT. *ChannelName*。

SYSTEM.DEF.CLUSRCVR
每一個叢集都有一個稱為 SYSTEM.DEF.CLUSRCVR 的預設 CLUSRCVR 通道定義。 SYSTEM.DEF.CLUSRCVR 用來為您在叢集中的佇列管理程式上建立叢集接收端通道時未指定的任何屬性提供預設值。

SYSTEM.DEF.CLUSSDR
每一個叢集都有一個稱為 SYSTEM.DEF.CLUSSDR 的預設 CLUSSDR 通道定義。 SYSTEM.DEF.CLUSSDR 用來為您在叢集中的佇列管理程式上建立叢集傳送端通道時未指定的任何屬性提供預設值。

相關概念
使用預設叢集物件

發佈/訂閱傳訊

發佈/訂閱傳訊可讓您取消資訊提供者與該資訊消費者之間的連結。 傳送應用程式及接收應用程式不必對彼此進行任何瞭解，即可傳送及接收資訊。

在點對點 IBM MQ 應用程式可以將訊息傳送至另一個應用程式之前，它需要瞭解該應用程式的相關資訊。 例如，它需要知道要將資訊傳送至其中的佇列名稱，也可能指定佇列管理程式名稱。

IBM MQ 發佈/訂閱可讓您的應用程式不需要知道目標應用程式的任何相關資訊。 傳送端應用程式只需要執行下列動作:

- 放置 一則 IBM MQ 訊息，其中包含應用程式想要的資訊。
- 將訊息指派給表示資訊主旨的主題。
- 讓 IBM MQ 處理該資訊的配送。

同樣地，目標應用程式不需要知道它所接收資訊的來源。

下圖顯示最簡單的發佈/訂閱系統。 有一個發佈者、一個佇列管理程式及一個訂閱者。 佇列管理程式上的訂閱者會進行訂閱，發佈會從發佈者傳送至佇列管理程式，然後佇列管理程式會將發佈轉遞至訂閱者。

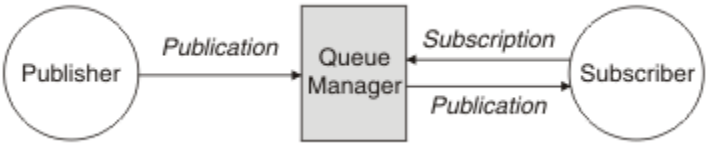


圖 *17:* 簡式發佈/訂閱配置

一般發佈/訂閱系統在許多不同的主題上有多個發佈者和多個訂閱者，且通常有多個佇列管理程式。 應用程式可以同時是發佈者和訂閱者。

發佈/訂閱傳訊與點對點之間的另一個顯著差異是傳送至點對點佇列的訊息僅由單一消費端應用程式處理。 發佈至發佈/訂閱主題的訊息，其中多個訂閱者已登錄興趣，由每個感興趣的訂閱者處理。

發佈/訂閱元件

發佈/訂閱是訂閱者可以從發佈者接收資訊 (以訊息形式) 的機制。 發佈者與訂閱者之間的互動由佇列管理程式使用標準 IBM MQ 機能來控制。

一般發佈/訂閱系統在許多不同的主題上有多個發佈者和多個訂閱者，且通常有多個佇列管理程式。 應用程式可以同時是發佈者和訂閱者。

資訊的提供者稱為發佈者。 發佈者提供主旨的相關資訊，而不需要知道對該資訊感興趣的應用程式的任何相關資訊。 發佈者以訊息形式產生此資訊，稱為 發佈 ，他們想要發佈並定義這些訊息的主題。

資訊的取用者稱為訂閱者。 訂閱者建立 訂閱 ，以說明訂閱者感興趣的主題。 因此，訂閱會決定將哪些發佈轉遞給訂閱者。 訂閱者可以執行多個訂閱，並且可以從許多不同的發佈者接收資訊。

已發佈資訊會以 IBM MQ 訊息傳送，且資訊的主旨由其 主題識別。 發佈者會指定發佈資訊時的主題，而訂閱者會指定它要接收其發佈資訊的主題。 訂閱者只會收到其訂閱之主題的相關資訊。

主題的存在可讓資訊提供者和消費者在發佈/訂閱傳訊中取消連結，因為不需要在每一個訊息中包含點對點傳訊所需要的特定目的地。

發佈者與訂閱者之間的互動全都由佇列管理程式控制。 佇列管理程式會從發佈者接收訊息，並從訂閱者接收訂閱 (主題範圍)。 佇列管理程式的工作是將已發佈的訊息遞送至已登錄感興趣訊息主題的訂閱者。

標準 IBM MQ 機能用來配送訊息，因此您的應用程式可以使用現有 IBM MQ 應用程式可用的所有特性。 這表示您可以使用持續訊息來取得僅一次的保證遞送，且您的訊息可以是交易式工作單元的一部分，以確保只有在發佈者確定訊息時才將訊息遞送至訂閱者。

發佈者和發佈

在 IBM MQ 中，發佈/訂閱發佈者是一種應用程式，以稱為發佈的標準 IBM MQ 訊息形式，提供指定主題的相關資訊給佇列管理程式使用。 發佈者可以發佈多個主題的相關資訊。

發佈者使用 MQPUT 動詞將訊息放置到先前開啟的主題，此訊息是發佈。 然後，本端佇列管理程式會將發佈遞送給具有訂閱發佈主題的任何訂閱者。 已發佈的訊息可由多個訂閱者耗用。

除了將發佈配送至具有適當訂閱的所有本端訂閱者之外，佇列管理程式也可以直接或透過具有主題訂閱者的佇列管理程式網路，將發佈配送至與其連接的任何其他佇列管理程式。

在 IBM MQ 發佈/訂閱網路中，發佈應用程式也可以是訂閱者。

同步點下的發佈

發佈者可以在同步點中發出 MQPUT 或 MQPUT1 呼叫，以將遞送給訂閱者的所有訊息併入工作單元中。 如果指定 MQPMO_RETAIN 選項或主題遞送選項 NPMSGDLV 或 PMSGDLV ，且值為 ALL 或 ALLDURR ，則佇列管理程式會在發佈者 MQPUT 或 MQPUT1 呼叫的範圍內，使用同步點中的內部 MQPUT 或 MQPUT1 呼叫。

狀態和事件資訊

發佈可以分類為狀態發佈 (例如股票的現行價格) 或事件發佈 (例如該股票的交易)。

狀態發佈

狀態出版品 包含某項目現行狀態的相關資訊，例如股票價格或足球比賽中的現行評分。 當發生某些情況（比方說，股價變動或足球分數改變）時，之前的狀態資訊就不再需要，因為它會被新的資訊所取代。

訂閱者會想要在啟動時接收狀態資訊的現行版本，並在狀態變更時傳送新資訊。

如果發佈包含狀態資訊，則通常會將它發佈為保留的發佈。 新訂閱者通常會立即想要現行狀態資訊; 訂閱者不想要等待導致重新發佈資訊的事件。 除非訂閱者使用 MQSO_PUBLICATIONS_ON_REQUEST 或 MQSO_NEW_PUBLICATIONS_ONLY 選項，否則訂閱者會在訂閱時自動接收主題的保留發佈。

事件發佈

事件出版品 包含所發生個別事件的相關資訊，例如部分股票的交易或特定目標的評分。 每一個事件與其他事件都是互相獨立的。

訂閱者將想要在事件發生時接收事件的相關資訊。

保留的發佈

依預設，在將發佈傳送至所有感興趣的訂閱者之後，會捨棄它。 不過，發佈者可以指定保留發佈的副本，以便將它傳送給未來對主題有興趣的訂閱者。

將發佈傳送至所有感興趣的訂閱者之後刪除發佈，適用於事件資訊，但並非一律適用於狀態資訊。 透過保留訊息，新訂閱者不必在接收起始狀態資訊之前等待重新發佈資訊。 例如，訂閱股票價格的訂閱者會立即收到現行價格，而不會等待股價變更 (因此重新發佈)。

佇列管理程式只能為每個主題保留一個發佈資訊，因此當新的保留發佈資訊到達佇列管理程式時，會刪除主題的現有保留發佈資訊。 不過，刪除現有發佈資訊可能不會與新保留發佈資訊的到達同步發生。 因此，在可能的情況下，最多只有一個發佈者傳送任何主題的保留發佈資訊。

訂閱者可以使用 MQSO_NEW_PUBLICATIONS_ONLY 訂閱選項，指定他們不想要接收保留的發佈。 現有訂閱者可以要求將保留發佈的副本傳送給他們。

有時您可能不想要保留發佈，即使是為了取得狀態資訊:

- 如果某個主題的所有訂閱都在對該主題進行任何發佈之前進行，且您不預期或不容許新的訂閱，則不需要保留發佈，因為它們在第一次發佈時遞送給完整的訂閱者集。
- 如果經常發生發佈，例如每秒，則新訂閱者 (或從失敗回復的訂閱者) 會在其起始訂閱之後幾乎立即收到現行狀態，因此不需要保留這些發佈。
- 如果發佈資訊很大，您可能最終需要大量儲存體空間來儲存每一個主題的保留發佈資訊。 在多個佇列管理程式環境中，網路中具有相符訂閱的所有佇列管理程式都會儲存保留的發佈。

在決定是否使用保留的發佈資訊時，請考量訂閱應用程式如何從失敗中回復。 如果發佈者未使用保留的發佈，則訂閱者應用程式可能需要在本端儲存其現行狀態。

若要確保保留發佈資訊，請使用 MQPMO_RETAIN put-message 選項。 如果使用此選項，且無法保留發佈，則不會發佈訊息，且呼叫會失敗並產生 MQRC_PUT_NOT_RETAINED。

如果訊息是保留的發佈資訊，則由 MQIsRetained 訊息內容指出。 訊息的持續性與最初發佈訊息時一樣。

相關概念

發佈/訂閱叢集中保留發佈的設計考量

同步點下的發佈

在 IBM MQ 發佈/訂閱中，同步點可由發佈者使用，或由佇列管理程式在內部使用。

當發佈者使用 MQPMO_SYNCPOINT 選項發出 MQPUT/MQPUT1 呼叫時，會使用同步點。 遞送給訂閱者的所有訊息都會計入 work.The MAXUMSGS 佇列管理程式屬性會指定此限制。 如果達到限制，發佈者會收到 2024 (07E8) (RC2024): MQRC_SYNCPOINT_LIMIT_REACHED 原因碼。

當發佈者使用 MQPMO_NO_SYNCPOINT 搭配 MQPMO_RETAIN 選項或主題遞送選項 NPMSGDLV/PMSGDLV (值為 ALL 或 ALLDURR) 發出 MQPUT/MQPUT1 呼叫時，佇列管理程式會使用內部同步點來保證訊息會依要求遞送。 如果在發佈者 MQPUT/MQPUT1 呼叫的範圍內達到限制，發佈者可以接收 2024 (07E8) (RC2024) :XX_ENCODE_CASE_ONE mqrc_syncpoint_limit_reached 原因碼。

訂閱者和訂閱

在 IBM MQ 發佈/訂閱中，訂閱者是一個應用程式，可從發佈/訂閱網路中的佇列管理程式要求特定主題的相關資訊。 訂閱者可以從多個發佈者接收關於相同或不同主題的訊息。

訂閱可以使用 MQSC 指令或由應用程式手動建立。 這些訂閱會發出至本端佇列管理程式，並包含訂閱者想要接收之發佈的相關資訊:

- 訂閱者感興趣的主題; 如果使用萬用字元，則這可以解析為多個主題。
- 要套用至已發佈訊息的選用選擇字串。
- 應該放置所選發佈的佇列控點 (稱為 訂閱者佇列) ， 以及選用的 CorrelId。

本端佇列管理程式會儲存訂閱資訊，當它收到發佈資訊時，會掃描資訊以判斷是否有訂閱符合發佈資訊的主題和選擇字串。 對於每一個相符的訂閱，佇列管理程式會將發佈導向訂閱者的訂閱者佇列。 可以使用 DIS SUB 及 DIS SBSTATUS 指令來檢視佇列管理程式儲存的訂閱相關資訊。

只有在發生下列其中一個事件時，才會刪除訂閱:

- 訂閱者使用 MQCLOSE 呼叫取消訂閱 (如果訂閱已非可延續的話)。
- 訂閱到期。
- 系統管理者會使用 DELETE SUB 指令來刪除訂閱。
- 訂閱者應用程式結束 (如果使訂閱無法持久)。
- 佇列管理程式會停止或重新啟動 (如果訂閱已非可延續的話)。

取得訊息時，請在 MQGET 呼叫上使用適當的選項。 如果您的應用程式只處理一個訂閱的訊息，則您至少應該使用 `get-by-correlid`，如 C 範例程式 `amqssbxa.c` 及 未受管理 MQ 訂閱者中所示範。 要使用的 **CorrelId** 會從 MQSD 中的 MQSUB 傳回。 **SubCorrelId** 欄位。

相關概念

複製及共用訂閱

相關參考

如何定義 sharedSubscription 內容的範例

受管理佇列及發佈/訂閱

當您建立訂閱時，可以選擇使用受管理佇列作業。 如果您使用受管理佇列作業，則會在建立訂閱時自動建立訂閱佇列。 受管理佇列會根據訂閱的延續性自動進行整理。 使用受管理佇列表示您不必擔心建立佇列來接收發佈，如果關閉不可延續訂閱連線，則會自動從訂閱者佇列中移除任何未耗用的發佈。

如果應用程式不需要使用特定佇列作為其訂閱者佇列 (它所接收發佈的目的地) ，則可以使用 MQSO_MANAGED 訂閱選項來使用 受管理訂閱 。 如果您建立受管理訂閱，佇列管理程式會針對佇列管理程式建立的訂閱者佇列，將物件控點傳回給訂閱者，其中將會接收發佈。 這是因為 受管理訂閱 是 IBM MQ 處理訂閱的訂閱。 將會傳回佇列的物件控點，可讓您瀏覽、取得或查詢佇列 (除非明確授與您暫時動態佇列的存取權，否則無法放置或設定受管理佇列的屬性)。

訂閱的延續性會決定當訂閱應用程式與佇列管理程式的連線中斷時，受管理佇列是否仍然存在。

當與不可延續訂閱一起使用時，受管理訂閱特別有用，因為當應用程式連線結束時，未耗用的訊息會無限期保留在取用者佇列中，佔用佇列管理程式中的空間。 如果您是使用受管理訂閱，則受管理佇列會是暫時動態佇列，因此當連線因下列任何原因而中斷時，會連同任何未耗用的訊息一併刪除:

- 使用具有 MQCO_REMOVE_SUB 的 MQCLOSE ， 並關閉受管理 Hobj。
- 使用不可延續訂閱 (MQSO_NON_DURABLE) 來失去與應用程式的連線。

- 已移除訂閱，因為它已過期且受管理 Hobj 已關閉。

受管理訂閱也可以與可延續訂閱搭配使用，但您可能想要將未耗用的訊息保留在訂閱者佇列上，以便在重新開啟連線時可以擷取它們。 因此，可延續訂閱的受管理佇列會採用永久動態佇列的形式，當訂閱應用程式與佇列管理程式的連線中斷時仍會保留。

如果您想要使用永久動態受管理佇列，則可以在訂閱上設定期限，這樣雖然在連線中斷之後佇列仍會存在，但不會無限期地繼續存在。

如果您刪除受管理佇列，則會收到錯誤訊息。

所建立的受管理佇列會以結束時的數字 (時間戳記) 來命名，因此每一個佇列都是唯一的。

訂閱延續性

訂閱可以配置成可延續或不可延續。 訂閱延續性會決定當訂閱應用程式與佇列管理程式中斷連線時，訂閱會發生什麼情況。

可延續訂閱

關閉訂閱應用程式與佇列管理程式的連線時，可延續訂閱會繼續存在。 如果訂閱是可延續的，當訂閱應用程式中斷連線時，訂閱會保留在原處，當訂閱應用程式使用建立訂閱時所傳回的 **SubName** 重新連接要求訂閱時，它可以使用該訂閱。

持續訂閱時，訂閱名稱 (**SubName**) 是必要的。 在佇列管理程式內，訂閱名稱必須是唯一的，才能用來識別訂閱。 如果您故意關閉訂閱的連線 (使用 MQCO_KEEP_SUB 選項) ，或已與佇列管理程式中斷連線，則在指定您要回復的訂閱時，這是必要的識別方法。 您可以使用 MQSUB 呼叫搭配 MQSO_RESUME 選項來回復現有的訂閱。 如果您搭配使用 DISPLAY SBSTATUS 指令與 SUBTYPE ALL 或 ADMIN ，也會顯示訂閱名稱。

當應用程式不再需要可延續訂閱時，可以搭配使用 MQCLOSE 函數呼叫與 MQCO_REMOVE_SUB 選項來移除它，也可以使用 MQSC 指令 DELETE SUB 來手動刪除它。

您可以使用 **DURSUB** 主題屬性來指定是否可以對主題進行可延續訂閱。

使用 MQSO_RESUME 選項從 MQSUB 呼叫傳回時，訂閱期限會設為訂閱的原始期限，而不是剩餘到期時間。

即使該訂閱者應用程式未連接，佇列管理程式仍會繼續傳送發佈以滿足可延續訂閱。 這會導致在訂閱者佇列上建立訊息。 避免此問題的最簡單方法是在適當的情況下使用不可延續訂閱。 不過，當需要使用可延續訂閱時，如果訂閱者使用 保留的發佈 選項來訂閱，則可以避免建立訊息。 然後，訂閱者可以使用 MQSUBRQ 呼叫來控制何時接收發佈。

不可延續訂閱

只有在訂閱應用程式與佇列管理程式的連線維持開啟時，不可延續訂閱才會存在。 當訂閱應用程式有意地或由於遺失連線，而中斷與佇列管理程式的連線時，會移除訂閱。 當連線關閉時，會從佇列管理程式移除訂閱的相關資訊，如果您使用 DISPLAY SBSTATUS 指令來顯示訂閱，則不再顯示訂閱的相關資訊。 不再將任何訊息放置到訂閱者佇列。

對於不可延續訂閱，訂閱者佇列上任何未耗用的發佈會發生什麼情況，如下所示。

- 如果訂閱應用程式使用 受管理目的地，則會自動移除任何尚未使用的發佈。
- 如果訂閱應用程式在訂閱時提供其專屬訂閱者佇列的控點，則不會自動移除未耗用的訊息。 如果適當的話，應用程式有責任清除佇列。 如果佇列由多個訂閱者或其他點對點應用程式共用，則可能不適合完全清除佇列。

雖然不可延續訂閱不需要，但佇列管理程式會使用訂閱名稱 (如果有提供的話)。 在佇列管理程式內，訂閱名稱必須是唯一的，才能用來識別訂閱。

相關概念

複製及共用訂閱

相關工作

使用 JMS 2.0 共用訂閱

相關參考
如何定義 sharedSubscription 內容的範例

選取字串
選取字串 是套用至發佈的表示式，用來判斷它是否符合訂閱。 選取字串可以包含萬用字元。

當您訂閱時，除了指定主題之外，您還可以指定選取字串，以根據發佈的訊息內容來選取發佈。

在修改選取字串以遞送給每一個訂閱者之前，會根據發佈者所放置的訊息來評估選取字串。 在選取字串中使用可能在發佈作業中修改的欄位時請小心。 例如， MQMD 欄位 `UserIdentifier`、 `MsgId` 及 `CorrelId`。

選取字串不應參照佇列管理程式在發佈作業中新增的任何訊息內容欄位 (請參閱 發佈/訂閱訊息內容)，但包含發佈主題字串的訊息內容 `MQTopicString` 除外。

相關概念
選取字串規則及限制

主題
主題是發佈/訂閱訊息中所發佈資訊的主旨。

點對點系統中的訊息會傳送至特定的目的地位址。 根據說明訊息內容的主旨，將主旨型發佈/訂閱系統中的訊息傳送給訂閱者。 在內容型系統中，訊息會根據訊息本身的內容來傳送給訂閱者。

IBM MQ 發佈/訂閱系統是主體型發佈/訂閱系統。 發佈者會建立訊息，並使用最適合發佈主題的主題字串來發佈訊息。 為了接收發佈資訊，訂閱者會建立訂閱，此訂閱具有用來選取發佈主題的型樣相符主題字串。 佇列管理程式會將發佈資訊，遞送至其訂閱與發佈主題相符且獲授權來接收發佈資訊的訂閱者。 第 56 頁的『主題字串』一文說明識別發佈主題之主題字串的語法。 訂閱者也會建立主題字串，以選取要接收的主題。 訂閱者建立的主題字串可以包含兩種替代萬用字元架構中的任一種，以根據發佈中的主題字串進行型樣比對。 型樣相符在 第 57 頁的『萬用架構』中說明。

在主題型發佈/訂閱中，發佈者或管理者負責將主題分類成主題。 通常主體會以階層方式組織成主題樹狀結構，並使用 '/' 字元在主題字串中建立子主題。 如需主題樹狀結構的範例，請參閱 第 62 頁的『主題樹狀結構』。 主題是主題樹狀結構中的節點。 主題可以是沒有任何子主題的葉節點，或者是具有子主題的中間節點。

與將主體組織成階層式主題樹狀結構一樣，您可以將主題與管理主題物件相關聯。 您可以將屬性指派給主題，例如主題是否配送在叢集中，方法是將它與管理主題物件相關聯。 建立關聯的方式是使用管理主題物件的 `TOPICSTR` 屬性來命名主題。 如果您未明確關聯管理主題物件與主題，該主題會繼承與管理主題物件 相關聯 之主題樹狀結構中最接近上代的屬性。 如果您根本未定義任何母項主題，它會繼承自 `SYSTEM.BASE.TOPIC`。 管理主題物件在 第 63 頁的『管理主題物件』中說明。

註: 即使您從 `SYSTEM.BASE.TOPIC`，為直接繼承自 `SYSTEM.BASE.TOPIC`。 例如，在美國各州的主題空間中， `USA/Alabama` `USA/Alaska` 等， `USA` 是根主題。 根主題的主要目的是建立離散、非重疊的主題空間，以避免發佈與錯誤訂閱相符。 這也表示您可以變更根主題的屬性，以影響整個主題空間。 例如，您可以設定 **CLUSTER** 屬性的名稱。

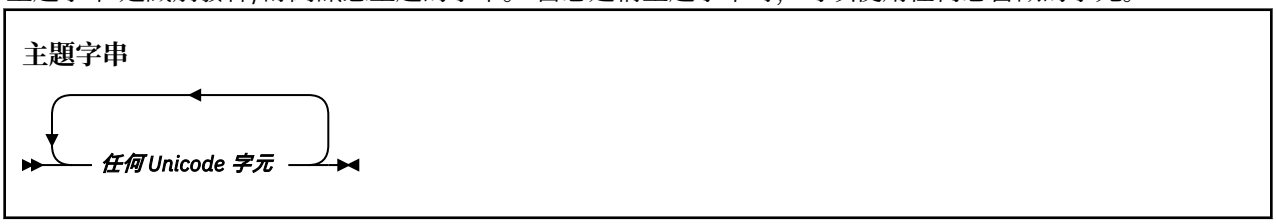
當您將主題稱為發佈者或訂閱者時，您可以選擇提供主題字串，或參照主題物件。 或者，您可以執行兩者，在此情況下，您提供的主題字串會定義主題物件的子主題。 佇列管理程式會識別主題，方法是將主題字串附加到主題物件中指定的主題字串字首，並在兩個主題字串之間插入其他 '/' ，例如 *topic string/object string*。 第 61 頁的『結合主題字串』進一步說明此情況。 產生的主題字串用來識別主題，並將它與管理主題物件相關聯。 管理主題物件不一定與對應於主要主題的主題物件相同。

在內容型發佈/訂閱中，您可以提供選項字串來搜尋每一則訊息的內容，以定義您要接收的訊息。 IBM MQ 提供中間形式的內容型發佈/訂閱，使用訊息選取器來掃描訊息內容，而不是訊息的完整內容，請參閱 選取器。 訊息選取元的原型用法是訂閱主題，然後以數值內容來限定選項。 選取元可讓您指定您只對特定範圍內的值感興趣; 您無法使用字元或主題型萬用字元來執行此動作。 如果您確實需要根據訊息的完整內容進行過濾，則需要使用 IBM Integration Bus。

主題字串
使用主題字串來標示您發佈為主題的資訊。 使用以字元為主或主題型萬用字元主題字串，來訂閱主題群組。

主題

主題字串 是識別發佈/訂閱訊息主題的字串。 當您建構主題字串時，可以使用任何您喜歡的字元。



在 IBM WebSphere MQ 7 發佈/訂閱中，有三個字元具有特殊意義。 它們可以在主題字串中的任何位置使用，但請小心使用。 特殊字元的用法在 第 58 頁的『主題型萬用架構』中說明。

正斜線 (/)

主題層次分隔字元。 使用 '/' 字元，將主題建構成主題樹狀結構。

如果可以的話，請避免空的主題層次 '//'。 這些對應於主題階層中沒有主題字串的節點。 主題字串中的前導或尾端 '/' 對應於前導或尾端空白節點，也應該避免。

雜湊符號 (#)

與 '/' 組合使用，以在訂閱中建構多層次萬用字元。 在用來命名已發佈主題的主題字串中，請小心使用 '/' 旁的 '#'。 第 57 頁的『主題字串範例』顯示可合理使用 '#'。

字串 '.../#/...'、 '#/...' 和 '.../#' 在訂閱主題字串中具有特殊意義。 字串符合主題階層中一個以上層次的所有主題。 因此，如果您建立具有其中一個序列的主題，則無法訂閱它，除非同時訂閱主題階層中多個層次的所有主題。

加號 (+)

與 '/' 組合使用，以在訂閱中建構單一層次萬用字元。 在用來命名已發佈主題的主題字串中，請小心使用 '/' 旁的 '+'。

字串 '.../+/...'、 '+/...' 和 '.../+' 在訂閱主題字串中具有特殊意義。 字串符合主題階層中一個層次的所有主題。 因此，如果您建立具有其中一個序列的主題，則無法訂閱它，除非同時訂閱主題階層中一個層次的所有主題。

主題字串範例

```
IBM/Business Area#/Results
IBM/Diversity/%African American
```

相關參考

TOPIC

萬用架構

有兩種用於訂閱多個主題的萬用架構。 架構選項是一個訂閱選項。

MQSO_WILDCARD_TOPIC
使用主題型萬用字元架構來選取要訂閱的主題。

如果未明確選取萬用字元綱目，則這是預設值。

MQSO_WILDCARD_CHAR
選取要使用文字萬用字元架構訂閱的主題。

透過在 DEFINE SUB 指令上指定 **wschema** 參數來設定任一架構。 如需相關資訊，請參閱 DEFINE SUB。

註: 在 IBM WebSphere MQ 7.0 之前建立的訂閱一律使用文字萬用字元架構。

範例

```
IBM/+/Results
#/Results
```

```
IBM/Software/Results
IBM/*ware/Results
```

主題型萬用架構
主題型萬用字元可讓訂閱者一次訂閱多個主題。

主題型萬用字元是 IBM MQ 發佈/訂閱中主題系統的強大功能。 多層次萬用字元及單層次萬用字元可用於訂閱，但訊息發佈者不可在主題中使用這些萬用字元。

主題型萬用字元架構可讓您選取依主題層次分組的發佈。 您可以針對主題階層中的每一個層次，選擇訂閱中該主題層次的字串是否必須完全符合發佈中的字串。 例如，訂閱 IBM/+/Results 會選取所有主題，

```
IBM/Software/Results
IBM/Services/Results
IBM/Hardware/Results
```

萬用字元有兩種類型。

多層次萬用字元

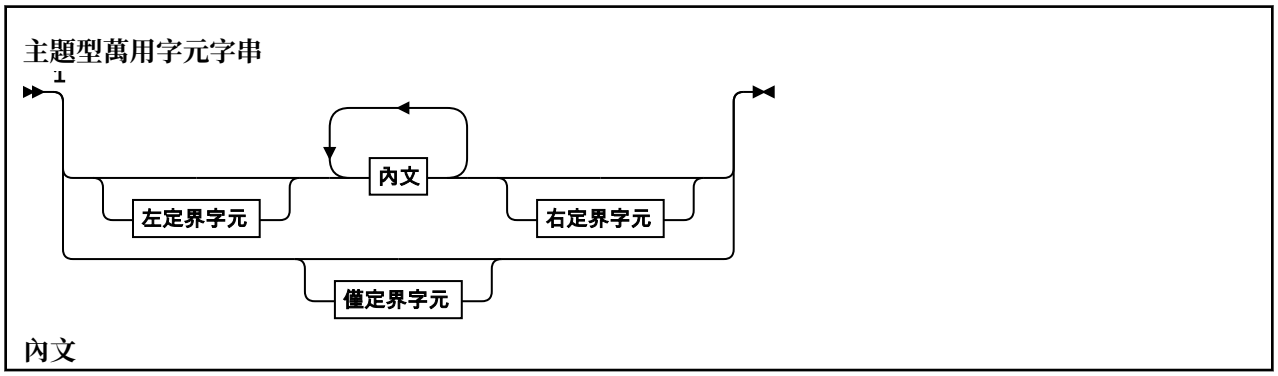
- 訂閱中使用多層次萬用字元。 在發佈中使用時，會將它視為文字。
- 多層次萬用字元 '#' 用來符合主題內任意數目的層次。 例如，使用範例主題樹狀結構，如果您訂閱 'USA/Alaska/#'，則會收到主題 'USA/Alaska' 及 'USA/Alaska/Juneau'的相關訊息。
- 多層次萬用字元可以代表零或多個層次。 因此， 'USA/#' 也可以符合單數 'USA'，其中 '#' 代表零個層次。 主題層次分隔字元在此環境定義中沒有意義，因為沒有可分隔的層次。
- 只有在自行指定或在主題層次分隔字元旁指定時，多層次萬用字元才有效。 因此， '#' 和 'USA/#' 是將 '#' 字元視為萬用字元的有效主題。 不過，雖然 'USA#' 也是有效的主題字串，但 '#' 字元不會被視為萬用字元，也沒有任何特殊意義。 如需相關資訊，請參閱第 60 頁的『當主題型萬用字元不是萬用字元時』。

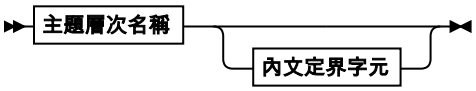
單一層次萬用字元

- 訂閱中使用單一萬用字元。 在發佈中使用時，會將它視為文字。
- 單一層次萬用字元 '+' 符合一個 (且僅符合一個) 主題層次。 例如， 'USA/+' 符合 'USA/Alabama'，但不符合 'USA/Alabama/Auburn'。 因為單一層次萬用字元只符合單一層次，所以 'USA/+' 不符合 'USA'。
- 單一層次萬用字元可以在主題樹狀結構中的任何層次使用，並與多層次萬用字元一起使用。 必須在主題層次分隔字元旁邊指定單一層次萬用字元，但自行指定時除外。 因此， '+' 和 'USA/+' 是將 '+' 字元視為萬用字元的有效主題。 不過，雖然 'USA+' 也是有效的主題字串，但 '+' 字元不會被視為萬用字元，也沒有任何特殊意義。 如需相關資訊，請參閱第 60 頁的『當主題型萬用字元不是萬用字元時』。

主題型萬用字元架構的語法沒有跳出字元。 是否將 '#' 和 '+' 視為萬用字元視其環境定義而定。 如需相關資訊，請參閱第 60 頁的『當主題型萬用字元不是萬用字元時』。

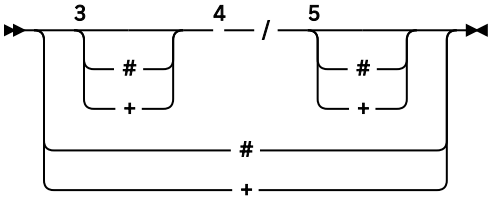
註: 主題字串的開頭和結尾會以特殊方式處理。 使用 '$' 來表示字串結尾，則 '$#/...' 是多層次萬用字元和 '$/#/..'。 是位於根目錄的空節點，後面接著多層次萬用字元。





主題層次名稱

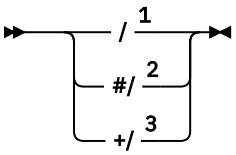
僅定界字元



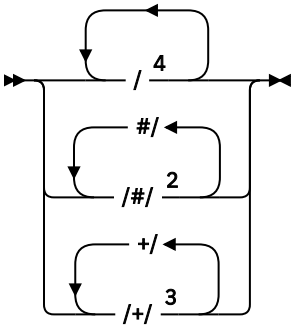
註:

[1] 空值或零長度主題字串無效
[2] 基於字元型和主題型萬用字元架構之間的相容性，建議您不要在層次名稱字串中使用任何 *, ?, % 。
[3] 這些觀察值相當於 左定界字元 型樣。
[4] 沒有萬用字元的 / 符合單一空主題。
[5] 這些觀察值相當於 右定界字元 型樣。
[6] 符合每個主題。
[7] 比對只有一個層次的每一個主題。

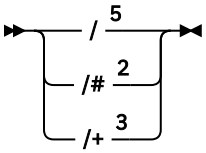
左定界字元



內文定界字元



右定界字元



註:

[1] 主題字串以空主題開頭
[2] 符合零個以上層次。 多個多層級比對字串具有與一個多層級比對字串相同的效果。

[3] 完全符合一個層次。
[4] // 是空的主題-沒有主題字串的主題物件。
[5] 主題字串以空主題結尾

當主題型萬用字元不是萬用字元時

在主題層次中與其他字元 (包括本身) 混合時，萬用字元 '+' 和 '#' 沒有特殊意義。

這表示可以發佈包含 '+' 或 '#' 以及主題層次中其他字元的主題。

例如，請考量下列兩個主題:

1. level0/level1/+/level4/#
2. level0/level1/#+/level4/level#

在第一個範例中，字元 '+' 和 '#' 被視為萬用字元，因此在要發佈至但在訂閱中有效的主題字串中無效。

在第二個範例中，字元 '+' 和 '#' 不會被視為萬用字元，因此主題字串可以同時發佈和訂閱。

範例

```
IBM/+/Results
#/Results
IBM/Software/Results
```

字元型萬用字元架構
文字萬用字元架構可讓您根據傳統字元比對來選取主題。

您可以使用字串 '*'來選取主題階層中多個層次的所有主題。 在文字型萬用字元架構中使用 '*' 相當於使用主題型萬用字元字串 '#'

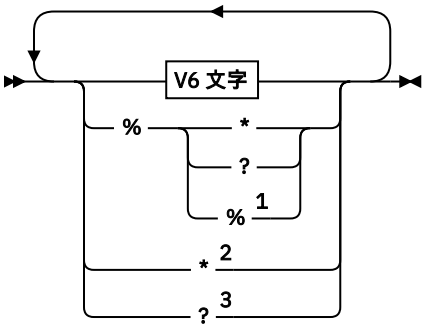
' *x*/*/*y* ' 相當於主題型架構中的 ' *x*/#/*y* '，並在層次 ' *x* 和 *y* '之間選取主題階層中的所有主題，其中 ' *x* ' 和 ' *y* ' 是不在萬用字元所傳回的層次集內的主題名稱。

主題型架構中的 '/+/' 在文字型架構中沒有確切的對等項目。 'IBM/*/Results' 也會選取 'IBM/Patents/Software/Results'。 只有在階層的每一個層次上的主題名稱集都是唯一的時，您才能一律使用產生相同相符項的兩個方法來建構查詢。

以一般方式使用，文字架構中的 '*' 及 '?' 在主題型架構中沒有對等項目。 主題型架構不會使用萬用字元來執行局部比對。 字元型萬用字元訂閱 'IBM/*ware/Results' 沒有主題型對等項目。

註: 使用萬用字元萬用字元訂閱的相符項比使用主題型訂閱的相符項慢。

字元型萬用字元字串



V6 文字

除了? 以外的任何 Unicode 字元 及 %*

註:

[1] 表示跳出下列字元，以便將它視為文字。 '%' 後面必須是 '*'、 '?' 或 '%'。 請參閱 第 57 頁的『主題字串範例』。
[2] 表示在訂閱中符合零個以上字元。
[3] 表示只符合訂閱中的一個字元。

範例

```
IBM/*/Results
IBM/*ware/Results
```

結合主題字串

建立訂閱或開啟主題以向其發佈訊息時，可以結合兩個個別的子主題字串或 "子主題"來形成主題字串。 一個子主題由應用程式或管理指令以主題字串提供，另一個是與主題物件相關聯的主題字串。 您可以單獨使用子主題作為主題字串，或結合子主題以形成新的主題名稱。

例如，當您使用 MQSC 指令 **DEFINE SUB** 定義訂閱時，指令可以將 **TOPICSTR** (主題字串) 及/或 **TOPICOBJ** (主題物件) 視為屬性。 如果只提供 **TOPICOBJ** ，則會使用與該主題物件相關聯的主題字串作為主題字串。 如果只提供 **TOPICSTR** ，則會使用它作為主題字串。 如果同時提供兩者，則它們會連結以形成 **TOPICOBJ** / **TOPICSTR** 格式的單一主題字串，其中 **TOPICOBJ** 配置的主題字串一律是第一個，且字串的兩個部分一律以 "/" 字元區隔。

同樣地，在 MQI 程式中， MQOPEN 會建立完整主題名稱。 它由兩個在發佈/訂閱 MQI 呼叫中使用的欄位組成，依列出的順序:

1. 主題物件的 **TOPICSTR** 屬性，在 **ObjectName** 欄位中指定。
2. **ObjectString** 參數定義應用程式所提供的子主題。

產生的主題字串會在 **ResObjectString** 參數中傳回。

如果每一個欄位的第一個字元不是空白或空字元，且欄位長度大於零，則這些欄位會被視為存在。 如果只存在其中一個欄位，則會使用未變更的欄位作為主題名稱。 如果任一欄位都沒有值，則呼叫會失敗，原因碼為 MQRC_UNKNOWN_OBJECT_NAME; 如果完整主題名稱無效，則呼叫會失敗 MQRC_TOPIC_STRING_ERROR 。

如果這兩個欄位都存在，則會在產生的合併主題名稱的兩個元素之間插入 "/" 字元。

下表顯示主題字串連結的範例:

表 *2:* 主題字串連結範例

主題物件的 TOPICSTR	**應用程式或 DEFINE SUB 指令提供的主題字串**	**完整主題名稱**	**註解**
足球/評分	''	足球/評分	主題物件的 TOPICSTR 是單獨使用。
''	足球/評分	足球/評分	單獨使用 ObjectString/ TOPICSTR。
football	評分	足球/評分	在連結點新增 "/" 字元。
football	/Scores	足球//Scores	在兩個字串之間產生「空節點」。 這與 "Football/ Scores"不同。
/Football	評分	/Football/Scores	主題以「空節點」開頭。 這與 "Football/Scores"不同。

"/" 字元被視為特殊字元，為第 62 頁的『主題樹狀結構』中的完整主題名稱提供結構。 "/" 字元不得用於任何其他原因，因為主題樹狀結構會受到影響。 主題 "/Football" 與主題 "Football"不同。

註: 如果您在建立訂閱時使用主題物件，則會在定義訂閱時修正主題物件主題字串的值。 對主題物件所做的任何後續變更都不會影響對其定義訂閱的主題字串。

主題字串中的萬用字元

下列萬用字元是特殊字元:

- 加號 (+)
- # 記號 (#)
- 星號 (*)
- 問號 (?)

只有在訂閱使用時，萬用字元才具有特殊意義。 這些字元在其他位置使用時不會被視為無效，不過您必須確定您瞭解如何使用它們，且在發佈或定義主題物件時，您可能不想要在主題字串中使用這些字元。

如果您在主題層次內發佈主題字串，且 # 或 + 與其他字元 (包括本身) 混合，則可以使用任一萬用字元架構來訂閱主題字串。

如果您在主題字串上發佈，且 # 或 + 是兩個 / 字元之間的唯一字元，則應用程式無法使用萬用字元架構 MQSO_WILDCARD_TOPIC 來明確訂閱主題字串。 此狀況會導致應用程式取得超出預期的發佈數。

您不應在已定義的主題物件的主題字串中使用萬用字元。 如果您這樣做，當發佈者使用物件時，該字元會被視為文字字元，當訂閱使用時，該字元會被視為萬用字元。 這可能會造成混淆。

程式碼 Snippet 範例

此程式碼 Snippet 擷取自範例程式 Example 2: Publisher to a variable topic，結合主題物件與變數主題字串:

```
MQOD   td = {MQOD_DEFAULT}; /* Object Descriptor      */
td.ObjectType = MQOT_TOPIC;  /* Object is a topic      */
td.Version = MQOD_VERSION_4;  /* Descriptor needs to be V4  */
strncpy(td.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
td.ObjectString.VSPtr = topicString;
td.ObjectString.VSLength = (MQLONG)strlen(topicString);
td.ResObjectString.VSPtr = resTopicStr;
td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
```

主題樹狀結構

您定義的每一個主題就是主題樹狀結構中的一個元素或節點。 主題樹狀結構可以是空的，以開始或包含先前使用 MQSC 或 PCF 指令定義的主題。 您可以定義新主題，方式是使用「建立主題」指令，亦或在發佈或訂閱中第一次指定該主題。

雖然您可以使用任何字串來定義主題的主題字串，但建議您選擇適合階層式樹狀結構的主題字串。 深入設計主題字串和主題樹狀結構可協助您執行下列作業:

- 訂閱多個主題。
- 建立安全原則。

雖然您可以將主題樹狀結構建構為平面線性結構，但最好以具有一個以上根主題的階層式結構來建置主題樹狀結構。 如需安全規劃和主題的相關資訊，請參閱 發佈/訂閱安全。

第 63 頁的圖 18 顯示具有一個根主題的主題樹狀結構範例。

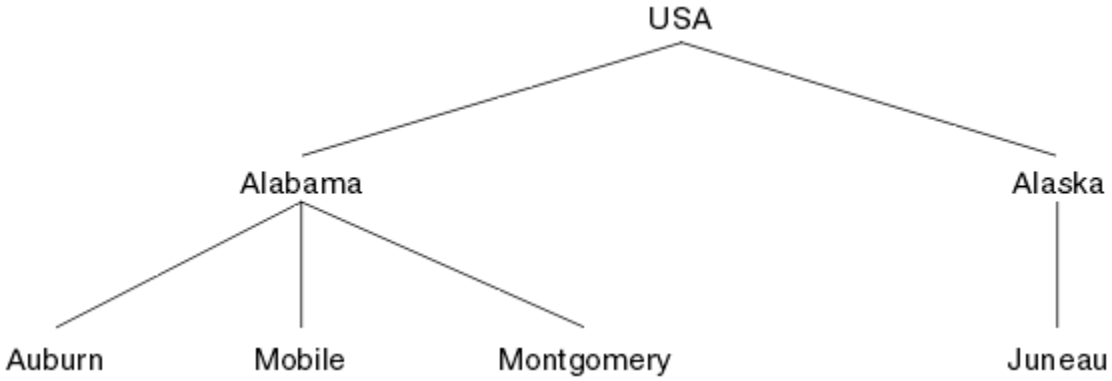


圖 *18:* 主題樹狀結構範例

圖中的每一個字串都代表主題樹狀結構中的一個節點。 透過從主題樹狀結構中的一個以上層次聚集節點來建立完整主題字串。 層次以 "/" 字元區隔。 完整指定的主題字串格式為: "root/level2/level3"。

第 63 頁的圖 18 所顯示主題樹狀結構中的有效主題如下:

"美國"
"美國/阿拉巴馬"
"美國/阿拉斯加"
"USA/Alabama/Auburn"
"USA/Alabama/Mobile"
"USA/Alabama/Montgomery"
"美國/阿拉斯加/朱諾"

當您設計主題字串和主題樹狀結構時，請記住佇列管理程式不會解譯或嘗試從主題字串本身衍生意義。 它只會使用主題字串，將選取的訊息傳送給該主題的訂閱者。

下列原則適用於主題樹狀結構的建構和內容:

- 主題樹狀結構中的層次數沒有限制。
- 主題樹狀結構中層次名稱的長度沒有限制。
- 可以有任意數目的「根」節點; 亦即，可以有任意數目的主題樹狀結構。

相關工作
減少主題樹狀結構中不想要的主題數目

管理主題物件
使用管理主題物件，您可以將特定非預設屬性指派給主題。

第 63 頁的圖 19 顯示如何將 `Sport` 的高階主題分成涵蓋不同體育項目的個別主題視覺化為主題樹狀結構:

圖 *19:* 主題樹狀結構的視覺化

第 64 頁的圖 20 顯示如何進一步分割主題樹狀結構，以區隔每一個運動的不同類型相關資訊:

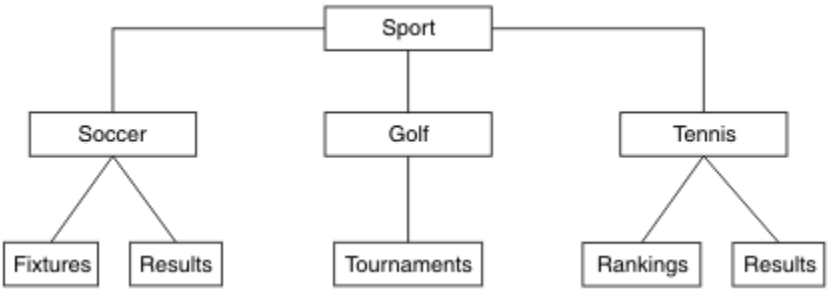


圖 *20:* 延伸主題樹狀結構

如果要建立圖解的主題樹狀結構，不需要定義任何管理主題物件。 此樹狀結構中的每一個節點都是由發佈或訂閱作業中所建立的主題字串所定義。 樹狀結構中的每一個主題都會從其母項繼承其屬性。 屬性繼承自上層主題物件，因為依預設所有屬性都設為 ASPARENT。 在此範例中，每個主題都具有與 `Sport` 主題相同的屬性。 `Sport` 主題沒有管理主題物件，並從 SYSTEM.BASE.TOPIC。

請注意，在主題樹狀結構的根節點 (即 SYSTEM.BASE.TOPIC，因為權限是繼承的，但無法限制。 因此，透過在此層次授與權限，您會將權限授與整個樹狀結構。 您應該在階層中的較低主題層次提供權限。

管理主題物件可用來定義主題樹狀結構中特定節點的特定屬性。 在下列範例中，管理主題物件定義為將足球主題的可延續訂閱內容 DURSUB 設為值 NO:

```
DEFINE TOPIC(FOOTBALL.EUROPEAN)
TOPICSTR('Sport/Soccer')
DURSUB(NO)
DESCR('Administrative topic object to disallow durable subscriptions')
```

現在可以將主題樹狀結構視覺化為:

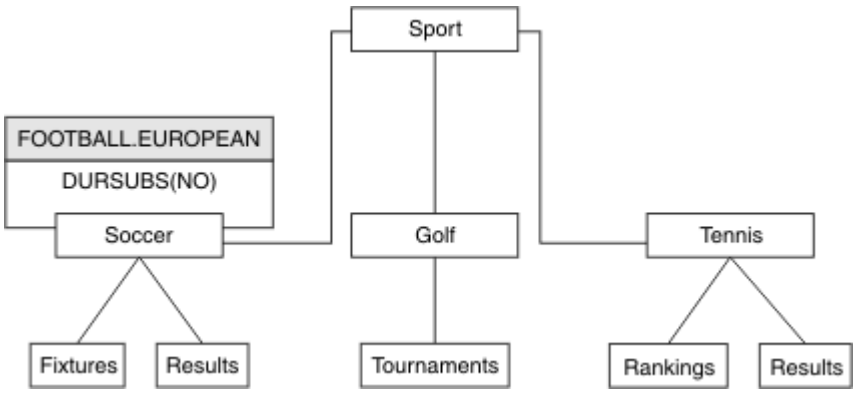


圖 *21:* 與 *Sport/Soccer* 主題相關聯之管理主題物件的視覺化

在樹狀結構中，訂閱 Soccer 之下主題的任何應用程式，在新增管理主題物件之前，仍然可以使用它們所使用的主題字串。 不過，現在可以使用物件名稱 `FOOTBALL.EUROPEAN` 而非字串 `/Sport/Soccer` 來撰寫應用程式以訂閱。 例如，若要訂閱 `/Sport/Soccer/Results`，應用程式可以將 `MQSD.ObjectName` 指定為 `FOOTBALL.EUROPEAN` ，並將 `MQSD.ObjectString` 指定為 `Results`。

使用此特性，您可以對應用程式開發人員隱藏部分主題樹狀結構。 在主題樹狀結構中的特定節點上定義管理主題物件，然後應用程式開發人員可以將他們自己的主題定義為節點的子項。 開發人員必須知道上層主題，但不能知道上層樹狀結構中的任何其他節點。

繼承屬性

如果主題樹狀結構有許多管理主題物件，依預設，每一個管理主題物件會從其最接近的上層管理主題繼承其屬性。 前一個範例已在 第 65 頁的圖 22 中延伸:

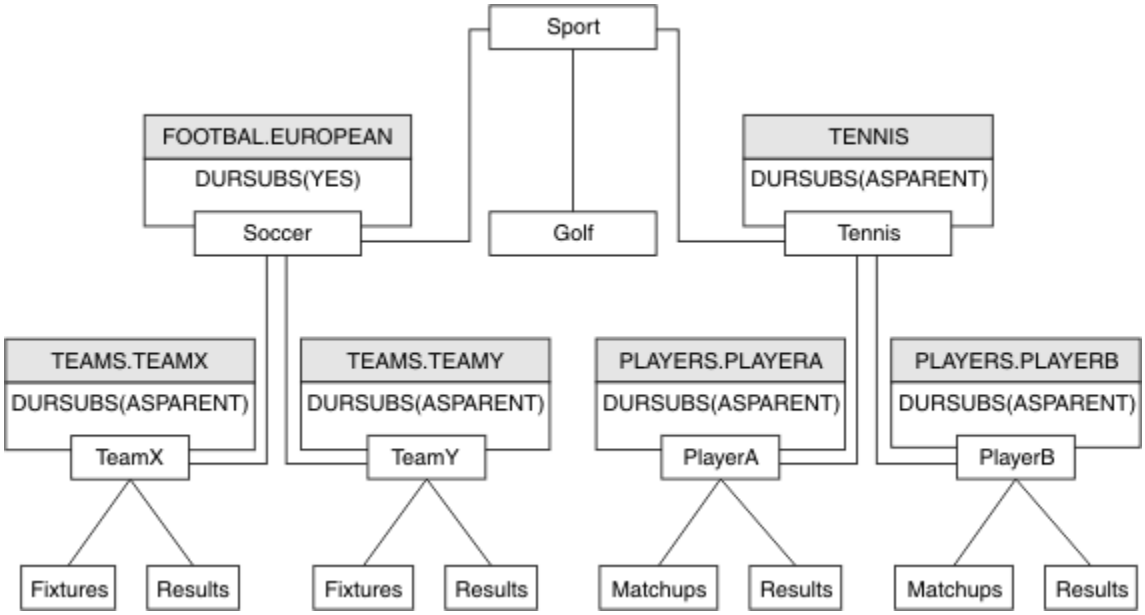


圖 *22:* 含有數個管理主題物件的主題樹狀結構

例如，使用繼承，為 /Sport/Soccer 的所有子主題提供訂閱不可延續的內容。 將 FOOTBALL.EUROPEAN 的 DURSUB 屬性變更為 NO。

可以使用下列指令來設定此屬性:

```
ALTER TOPIC(FOOTBALL.EUROPEAN) DURSUB(NO)
```

Sport/Soccer 子主題的所有管理主題物件，都會將 DURSUB 內容設為預設值 ASPARENT。 將 FOOTBALL.EUROPEAN 的 DURSUB 內容值變更為 NO 之後， Sport/Soccer 的子主題會繼承 DURSUB 內容值 NO。 Sport/Tennis 的所有子主題都會從 SYSTEM.BASE.TOPIC 物件繼承 DURSUB 的值。 SYSTEM.BASE.TOPIC 的值為 YES。

嘗試對主題 Sport/Soccer/TeamX/Results 進行可延續訂閱現在會失敗; 不過，嘗試對 Sport/Tennis/PlayerB/Results 進行可延續訂閱將會成功。

使用 萬用字元 內容控制萬用字元用法

使用 MQSC **Topic** 萬用字元 內容或對等 PCF Topic WildcardOperation 內容來控制將發佈遞送至使用萬用字元主題字串名稱的訂閱者應用程式。 WILDCARD 內容可以具有下列兩個可能值之一:

WILDCARD
關於此主題的萬用字元訂閱的行為。

PASSTHRU
對於比此主題物件的主題字串更不具體的萬用字元式主題所做的訂閱，將接收到對此主題以及比此主題更具體的主題字串所進行的發佈。

BLOCK
對於比此主題物件的主題字串更不具體的萬用字元式主題所做的訂閱，不會接收到對此主題或比此主題更具體的主題字串所進行的發佈。

在定義訂閱時將使用此屬性的值。 如果變更此屬性，則現有訂閱所涵蓋的主題集不會因為此修改而受到影響。 如果在建立或刪除主題物件時拓蹼發生變更，也適用此實務範例；將使用修改後的拓蹼來建立與 WILDCARD 屬性修改後建立的訂閱相符的主題集。 若要針對現有訂閱強制重新評估相符的主題集，則必須重新啟動佇列管理程式。

在範例 第 68 頁的『範例: 建立 Sport 發佈/訂閱叢集』中，您可以遵循步驟來建立 第 66 頁的圖 23 中所示的主題樹狀結構。

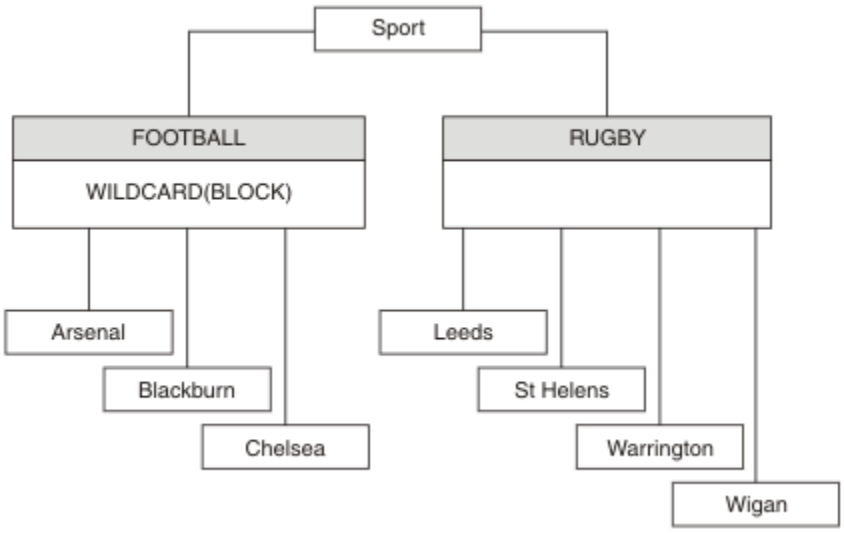


圖 *23:* 使用 萬用字元 內容 *BLOCK* 的主題樹狀結構

使用萬用字元主題字串 *#* 的訂閱者會接收 Sport 主題和 Sport/Rugby 子樹狀結構的所有發佈。 訂閱者不會收到 Sport/Football 子樹狀結構的發佈，因為 Sport/Football 主題的 **萬用字元** 內容值是 BLOCK。

PASSTHRU 是預設值。 您可以將 **萬用字元** 內容值 PASSTHRU 設為 Sport 樹狀結構中的節點。 如果節點沒有 **萬用字元** 內容值 BLOCK，則設定 PASSTHRU 不會變更訂閱者對 Sports 樹狀結構中節點所觀察到的行為。

在範例中，建立訂閱以查看萬用字元設定如何影響遞送的發佈; 請參閱 第 70 頁的圖 27。 在 第 71 頁的圖 30 中執行發佈指令，以建立部分發佈。

```
pub QMA
```

圖 *24:* 發佈至 *QMA*

結果顯示在 第 66 頁的表 3 中。 請注意如何設定 WILDCARD 內容值 BLOCK，防止使用萬用字元的訂閱接收對萬用字元範圍內主題的發佈。

表 *3:* 在 *QMA* 上收到的出版品

訂閱	**主題字串**	**收到的出版物**	**附註**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football 上的 WILDCARD(BLOCK) 封鎖所有 Football 子樹狀結構的發佈
SARSENAL	Sports/#/Arsenal	-	WILDCARD(BLOCK) on Sports/Football 阻止在 Arsenal 上進行萬用字元訂閱
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的預設 WILDCARD 不會阻止 Leeds 上的萬用字元訂閱。

註:

假設訂閱具有萬用字元，其符合具有 WILDCARD 內容值 BLOCK 的主題物件。 如果訂閱在相符萬用字元的右側也有主題字串，則訂閱永遠不會收到發佈。 未封鎖的發佈資訊集是屬於已封鎖萬用字元母項之主題的發佈

資訊。 作為主題子項且具有 BLOCK 內容值的主題的發佈會被萬用字元封鎖。 因此，如果訂閱主題字串包含萬用字元右側的主題，則永遠不會收到任何要符合的發佈。

將 WILDCARD 內容值設為 BLOCK 並不表示您無法使用包含萬用字元的主題字串來訂閱。 這類訂閱是正常的。 訂閱有一個明確主題符合主題，且主題物件具有 WILDCARD 內容值 BLOCK。 對於作為主題母項或子項且具有 **萬用字元** 內容值 BLOCK 的主題，它會使用萬用字元。 在 第 66 頁的圖 23 的範例中， Sports/Football/# 之類的訂閱可以接收發佈。

萬用字元和叢集主題

叢集主題定義會延伸到叢集中的每個佇列管理程式。 訂閱叢集中某個佇列管理程式的叢集主題會導致佇列管理程式建立 Proxy 訂閱。 在叢集中的所有其他佇列管理程式上建立 Proxy 訂閱。 使用包含萬用字元且與叢集主題結合的主題字串的訂閱，可能會提供難以預測的行為。 此行為在下列範例中說明。

在範例 第 68 頁的『範例: 建立 Sport 發佈/訂閱叢集』 的叢集設定中， QMB 具有與 QMA 相同的訂閱集，但 QMB 在發佈者發佈至 QMA 之後未收到任何發佈，請參閱 第 66 頁的圖 24。 雖然 Sports/Football 及 Sports/Rugby 主題是叢集主題，但 fullsubs.tst 中定義的訂閱不會參照叢集主題。 沒有任何 Proxy 訂閱從 QMB 延伸到 QMA。 如果沒有 Proxy 訂閱，則不會將任何對 QMA 的發佈轉遞至 QMB。

部分訂閱 (例如 Sports/#/Leeds) 似乎參照叢集主題，在此情況下為 Sports/Rugby。 Sports/#/Leeds 訂閱實際上會解析為主題物件 SYSTEM.BASE.TOPIC。

用於解析訂閱 (例如 Sports/#/Leeds) 所參照的主題物件的規則如下。 將主題字串截斷為第一個萬用字元。 掃描左側的主題字串，以尋找具有相關聯管理主題物件的第一個主題。 主題物件可以指定叢集名稱，或定義本端主題物件。 在範例 Sports/#/Leeds 中，截斷之後的主題字串是 Sports，它沒有主題物件，因此 Sports/#/Leeds 繼承自 SYSTEM.BASE.TOPIC，它是本端主題物件。

若要查看訂閱叢集主題如何變更萬用字元傳播的運作方式，請執行批次 Script upsubs.bat。 該 Script 會清除訂閱佇列，並在 fullsubs.tst 中新增叢集主題訂閱。 再次執行 puba.bat 以建立一批出版品; 請參閱 第 66 頁的圖 24。

第 67 頁的表 4 顯示將兩個新訂閱新增至發佈發佈所在之相同佇列管理程式的結果。 結果如預期，新訂閱各會收到一個發佈，而其他訂閱所收到的發佈數則維持不變。 其他叢集佇列管理程式上發生非預期的結果; 請參閱 第 68 頁的表 5。

表 *4:* 在 *QMA* 上收到的出版品

訂閱	**主題字串**	**收到的出版物**	**附註**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football 上的 WILDCARD(BLOCK) 封鎖所有 Football 子樹狀結構的發佈
SARSENAL	Sports/#/Arsenal	-	WILDCARD(BLOCK) on Sports/Football 阻止在 Arsenal 上進行萬用字元訂閱
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的預設 WILDCARD 不會阻止 Leeds 上的萬用字元訂閱。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 會接收發佈，因為訂閱沒有萬用字元。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 會在任何事件中收到發佈。

第 68 頁的表 5 顯示在 QMB 上新增兩個新訂閱，以及在 QMA 上發佈的結果。 回想一下，如果沒有這兩個新訂閱， QMB 不會收到任何發佈。 如預期，這兩個新的訂閱會接收發佈，因為 Sports/FootBall 和 Sports/Rugby 都是叢集主題。 QMB 已將 Sports/Football/Arsenal 和 Sports/Rugby/Leeds 的 Proxy 訂閱轉遞至 QMA，然後將發佈傳送至 QMB。

非預期的結果是先前未收到任何發佈的兩個訂閱 Sports/# 及 Sports/#/Leeds 現在已收到發佈。 原因是其他訂閱的 Sports/Football/Arsenal 及 Sports/Rugby/Leeds 轉遞至 QMB 的發佈現在可用於連接至 QMB 的任何訂閱者。 因此，對本端主題 Sports/# 及 Sports/#/Leeds 的訂閱會收到 Sports/Rugby/Leeds 發佈。 Sports/#/Arsenal 會繼續不接收發佈，因為「體育/足球」已將其 **萬用字元** 內容值設為 BLOCK。

表 *5:* 在 *QMB* 上收到的出版品			
訂閱	**主題字串**	**收到的出版物**	**附註**
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*WILDCARD(BLOCK)* 在 *Sports/Football* 上封鎖所有發佈至 *Football* 子樹狀結構
SARSENAL	Sports/#/Arsenal	-	WILDCARD(BLOCK) on Sports/Football 阻止在 Arsenal 上進行萬用字元訂閱
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	*Sports/Rugby* 上的預設 *WILDCARD* 不會阻止 *Leeds* 上的萬用字元訂閱。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 會接收發佈，因為訂閱沒有萬用字元。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 會在任何事件中收到發佈。

在大部分應用程式中，一個訂閱不想要影響另一個訂閱的行為。 WILDCARD 內容與值 BLOCK 的一個重要用途是讓相同主題字串 (包含萬用字元) 的訂閱統一運作。 不論訂閱位於與發佈者相同的佇列管理程式上，或位於不同的佇列管理程式上，訂閱的結果都相同。

萬用字元和串流

對於寫入至發佈/訂閱 API 的新應用程式，其效果是 * 的訂閱不會收到任何發佈。 若要接收所有「體育」發佈資訊，您必須訂閱 Sports/*或 Sports/#，以及類似的 Business 發佈資訊。

當發佈/訂閱分配管理系統移轉至更新版本的 IBM MQ 時，現有排入佇列的發佈/訂閱應用程式的行為不會變更。 **Publish**、 **Register Publisher** 或 **Subscriber** 指令中的 **StreamName** 內容對映至已移轉串流的主題名稱。

萬用字元和訂閱點

對於寫入發佈/訂閱 API 的新應用程式，移轉的效果是 * 的訂閱不會收到任何發佈。 若要接收所有「體育」發佈資訊，您必須訂閱 Sports/*或 Sports/#，以及類似的 Business 發佈資訊。

當發佈/訂閱分配管理系統移轉至更新版本的 IBM MQ 時，現有排入佇列的發佈/訂閱應用程式的行為不會變更。 **Publish**、 **Register Publisher** 或 **Subscriber** 指令中的 **SubPoint** 內容對映至已移轉訂閱的主題名稱。

範例: 建立 Sport 發佈/訂閱叢集

接下來的步驟會建立叢集 CL1，其中包含四個佇列管理程式: 兩個完整儲存庫、 CL1A 和 CL1B，以及兩個局部儲存庫 QMA 和 QMB。 完整儲存庫用來只保留叢集定義。 QMA 指定為叢集主題主機。 可延續訂閱同時定義在 QMA 和 QMB 上。

註: 此範例是針對 Windows 進行編碼。 您必須重新編碼 建立 qmgrs.bat 及 建立 pub.bat ，以在其他平台上配置及測試範例。

1. 建立 Script 檔。
   a. 建立 topics.tst

b. 建立 wildsubs.tst
c. 建立 fullsubs.tst
d. 建立 qmgrs.bat
e. 建立 pub.bat

2. 執行 Create qmgrs.bat 以建立配置。

```
qmgrs
```

在 第 66 頁的圖 23 中建立主題。 圖 5 中的 Script 會建立叢集主題 Sports/Football 和 Sports/Rugby。

註: REPLACE 選項不會取代主題的 TOPICSTR 內容。 TOPICSTR 是在範例中用來測試不同主題樹狀結構的有用內容。 若要變更主題，請先刪除主題。

```
DELETE TOPIC ('Sports')
DELETE TOPIC ('Football')
DELETE TOPIC ('Arsenal')
DELETE TOPIC ('Blackburn')
DELETE TOPIC ('Chelsea')
DELETE TOPIC ('Rugby')
DELETE TOPIC ('Leeds')
DELETE TOPIC ('Wigan')
DELETE TOPIC ('Warrington')
DELETE TOPIC ('St. Helens')

DEFINE TOPIC ('Sports')   TOPICSTR('Sports')
DEFINE TOPIC ('Football')  TOPICSTR('Sports/Football') CLUSTER(CL1) WILDCARD(BLOCK)
DEFINE TOPIC ('Arsenal')  TOPICSTR('Sports/Football/Arsenal')
DEFINE TOPIC ('Blackburn') TOPICSTR('Sports/Football/Blackburn')
DEFINE TOPIC ('Chelsea')  TOPICSTR('Sports/Football/Chelsea')
DEFINE TOPIC ('Rugby')   TOPICSTR('Sports/Rugby')  CLUSTER(CL1)
DEFINE TOPIC ('Leeds')   TOPICSTR('Sports/Rugby/Leeds')
DEFINE TOPIC ('Wigan')   TOPICSTR('Sports/Rugby/Wigan')
DEFINE TOPIC ('Warrington') TOPICSTR('Sports/Rugby/Warrington')
DEFINE TOPIC ('St. Helens') TOPICSTR('Sports/Rugby/St. Helens')
```

圖 *25:* 刪除並建立主題*: topics.tst*

註: 刪除主題，因為 REPLACE 不會取代主題字串。

使用萬用字元建立訂閱。 萬用字元會對應主題與 第 66 頁的圖 23 中的主題物件。 為每一個訂閱建立佇列。 執行或重新執行 Script 時，會清除佇列並刪除訂閱。

註: REPLACE 選項不會取代訂閱的 TOPICOBJ 或 TOPICSTR 內容。 TOPICOBJ 或 TOPICSTR 是在測試不同訂閱的範例中有效改變的內容。 若要變更它們，請先刪除訂閱。

```
DEFINE QLOCAL(QSPORTS)  REPLACE
DEFINE QLOCAL(QSARSENAL) REPLACE
DEFINE QLOCAL(QSLEEDS)  REPLACE
CLEAR QLOCAL(QSPORTS)
CLEAR QLOCAL(QSARSENAL)
CLEAR QLOCAL(QSLEEDS)

DELETE SUB  (SPORTS)
DELETE SUB  (SARSENAL)
DELETE SUB  (SLEEDS)
DEFINE SUB  (SPORTS)  TOPICSTR('Sports/#')     DEST(QSPORTS)
DEFINE SUB  (SARSENAL) TOPICSTR('Sports/+/Arsenal') DEST(QSARSENAL)
DEFINE SUB  (SLEEDS)  TOPICSTR('Sports/+/Leeds')  DEST(QSLEEDS)
```

圖 *26:* 建立萬用字元訂閱*: wildsubs.tst*

建立參照叢集主題物件的訂閱。

註:

定界字元 /會自動插入 TOPICOBJ 所參照的主題字串與 TOPICSTR 所定義的主題字串之間。

DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) 定義會建立相同的訂閱。 TOPICOBJ 是用來作為參照您已定義之主題字串的快速方式。 建立的訂閱不再參照主題物件。

```
DEFINE QLOCAL(QFARSENAL) REPLACE
DEFINE QLOCAL(QRLEEDS)  REPLACE
CLEAR QLOCAL(QFARSENAL)
CLEAR QLOCAL(QRLEEDS)

DELETE SUB  (FARSENAL)
DELETE SUB  (RLEEDS)
DEFINE SUB  (FARSENAL) TOPICOBJ('Football') TOPICSTR('Arsenal') DEST(QFARSENAL)
DEFINE SUB  (RLEEDS)  TOPICOBJ('Rugby')  TOPICSTR('Leeds')  DEST(QRLEEDS)
```

圖 *27:* 刪除並建立訂閱*: fullsubs.tst*

建立具有兩個儲存庫的叢集。 建立兩個局部儲存庫以進行發佈和訂閱。 請重新執行 Script 以刪除所有項目，然後重新啟動。 Script 也會建立主題階層及起始萬用字元訂閱。

註:

在其他平台上，撰寫類似的 Script ，或輸入所有指令。 使用 Script 可讓您快速刪除所有項目，並以相同的配置重新開始。

```
@echo off
set port.CL1B=1421
set port.CLIA=1420
for %%A in (CL1A CL1B QMA QMB) do call :createQM %%A
call :configureQM CL1A CL1B %port.CL1B% full
call :configureQM CL1B CL1A %port.CL1A% full
for %%A in (QMA QMB) do call :configureQM %%A CL1A %port.CL1A% partial
for %%A in (topics.tst wildsubs.tst) do runmqsc QMA < %%A
for %%A in (wildsubs.tst) do runmqsc QMB < %%A
goto:eof

:createQM
echo Configure Queue manager %1
endmqm -p %1
for %%B in (dlt crt str) do %%Bmqm %1
goto:eof

:configureQM
if %1==CL1A set p=1420
if %1==CL1B set p=1421
if %1==QMA set p=1422
if %1==QMB set p=1423
echo configure %1 on port %p% connected to repository %2 on port %3 as %4 repository
echo DEFINE LISTENER(LST%1) TRPTYPE(TCP) PORT(%p%) CONTROL(QMGR) REPLACE | runmqsc %1
echo START LISTENER(LST%1) | runmqsc %1
if full==%4 echo ALTER QMGR REPOS(CL1) DEADQ(SYSTEM.DEAD.LETTER.QUEUE) | runmqsc %1
echo DEFINE CHANNEL(TO.%2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('LOCALHOST(%3)') CLUSTER(CL1)
REPLACE | runmqsc %1
echo DEFINE CHANNEL(TO.%1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('LOCALHOST(%p%)')
CLUSTER(CL1) REPLACE | runmqsc %1
goto:eof
```

圖 *28:* 建立佇列管理程式*: qmgrs.bat*

透過將訂閱新增至叢集主題來更新配置。

```
@echo off
for %%A in (QMA QMB) do runmqsc %%A < wildsubs.tst
for %%A in (QMA QMB) do runmqsc %%A < upsubs.tst
```

圖 *29:* 更新訂閱*: upsubs.bat*

執行 pub.bat，並以佇列管理程式作為參數，以發佈包含發佈主題字串的訊息。 Pub.bat 使用範例程式 **amqspub**。

```
@echo off
@rem Provide queue manager name as a parameter
set S=Sports
set S=6 Sports/Football Sports/Football/Arsenal
set S=6 Sports/Rugby Sports/Rugby/Leeds
for %%B in (6) do echo %%B | amqspub %%B %1
```

圖 *30:* 發佈*: pub.bat*

串流和主題

已排入佇列的發佈/訂閱具有發佈串流的概念，該發佈串流不存在於整合的發佈/訂閱模型中。 在排入佇列的發佈/訂閱中，串流提供一種方式來區隔不同主題的資訊流程。 串流實作為最上層主題，可透過管理方式對映至不同的主題 ID。

會針對網路上的所有分配管理系統及佇列管理程式自動設定預設串流 SYSTEM.BROKER.DEFAULT.STREAM，並且不需要其他配置即可使用預設串流。 將預設串流視為未命名的預設主題空間。 發佈至預設串流的主題會立即提供給所有已連接的佇列管理程式，並啟用排入佇列的發佈/訂閱。 具名串流類似於個別的具名主題空間。 指定的串流必須定義在使用它的每一個分配管理系統上。

如果發佈者和訂閱者位於不同的佇列管理程式上，則在相同的分配管理系統階層中連接分配管理系統之後，不需要進一步配置發佈，以及在它們之間流動的訂閱。 同樣的互操作性也是相反的。

具名串流

使用排入佇列的發佈/訂閱程式設計模型的解決方案設計程式，可能會決定將所有體育發佈放入名為 Sport 的具名串流中。 如果要讓在 IBM MQ 上執行且已啟用排入佇列的發佈/訂閱的佇列管理程式可以使用串流，必須手動新增該串流。

在串流 Sport 上訂閱 Soccer/Results 的已排入佇列發佈/訂閱應用程式會運作而不會變更。 使用 MQSUB 訂閱主題 Sport，並提供主題字串 Soccer/Results 的整合發佈/訂閱應用程式也會收到相同的發佈。

新增串流主題中說明了新增串流的作業。 基於兩個原因，您可能需要手動新增串流。

1. 您可以繼續開發在更新版本佇列管理程式上執行的已排入佇列的發佈/訂閱應用程式，而不是將應用程式移轉至整合發佈/訂閱 MQI 介面。
2. 串流至主題的預設對映會導致主題空間中的「衝突」，且串流上的發佈與來自其他位置的發佈具有相同的主題字串。

權限

依預設，在主題樹狀結構的根位置有多個主題物件: SYSTEM.BASE.TOPIC, SYSTEM.BROKER.DEFAULT.STREAM 和 SYSTEM.BROKER.DEFAULT.SUBPOINT。 權限 (例如，發佈或訂閱) 由 SYSTEM.BASE.TOPIC 上的權限決定; 會忽略 SYSTEM.BROKER.DEFAULT.STREAM 或 SYSTEM.BROKER.DEFAULT.SUBPOINT 上的任何權限。 如果使用非空主題字串刪除並重建 SYSTEM.BROKER.DEFAULT.STREAM 或 SYSTEM.BROKER.DEFAULT.SUBPOINT 之一，則會以與一般主題物件相同的方式使用對那些物件定義的權限。

串流與主題之間的對映

在 IBM MQ 中，透過建立佇列並為其提供與串流相同的名稱，來模擬已排入佇列的發佈/訂閱串流。 有時佇列稱為串流佇列，因為這就是佇列發佈/訂閱應用程式的顯示方式。 將佇列新增至稱為 SYSTEM.QPUBSUB.QUEUE.NAMELIST 的特殊名稱清單，即可向發佈/訂閱引擎識別該佇列。 您可以視需要新增任意數目的串流，方法是將其他特殊佇列新增至名稱清單。 最後，您需要新增主題，其名稱與串流相同，且主題字串與串流名稱相同，因此您可以發佈及訂閱主題。

不過，在異常情況下，您可以為對應於串流的主題提供您在定義主題時選擇的任何主題字串。 主題字串的目的是提供主題空間中的唯一名稱。 通常，串流名稱會完美地達到該目的。 有時，串流名稱與現有主題名稱衝突。 如果要解決問題，請為與串流相關聯的主題選擇另一個主題字串。 選擇任何主題字串，確定它是唯一的。

主題定義中定義的主題字串會以一般方式加上發佈者及訂閱者使用 MQOPEN 或 MQSUB MQI 呼叫所提供的主題字串的字首。 使用主題物件來參照主題的應用程式不受選擇字首主題字串的影響-因此，您可以選擇任何主題字串，讓發佈資訊在主題空間中保持唯一。

將不同的串流重新對映至不同的主題，取決於主題字串所使用的字首是唯一的，以完全將一組主題與另一組主題分開。 您必須定義嚴格遵循的通用主題命名慣例，對映才能運作。

在 IBM MQ 中，您可以使用字首機制，將主題字串重新對映至主題空間中的另一個位置。

註: 當您刪除串流時，請先刪除串流上的所有訂閱。 如果任何訂閱源自分配管理系統階層中的其他分配管理系統，則此動作最重要。

訂閱點和主題

主題和主題物件會模擬具名訂閱點。

若要手動新增訂閱點，請參閱 新增訂閱點。

IBM MQ 中的訂閱點

IBM MQ 會將訂閱對映至 IBM MQ 主題樹狀結構內的不同主題空間。 指令訊息中沒有訂閱點的主題會對映至 IBM MQ 主題樹狀結構的根目錄，並繼承 SYSTEM.BASE.TOPIC 的內容。

具有訂閱點的指令訊息正在使用 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中的主題物件清單進行處理。 指令訊息中的訂閱點名稱會與清單中每一個主題物件的主題字串進行比對。 如果找到相符項，則會在主題字串前面附加訂閱點名稱作為主題節點。 主題會從 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中找到的相關聯主題物件繼承其內容。

使用訂閱點的效果是為每一個訂閱點建立個別主題空間。 主題空間根在與訂閱點同名的主題中。 每一個主題空間中的主題會從與訂閱點同名的主題物件繼承其內容。

未在相符主題物件中設定的任何內容都會以正常方式從 SYSTEM.BASE.TOPIC 繼承。

現有排入佇列的發佈/訂閱應用程式 (使用 MQRFH2 訊息標頭) 可透過在 Publish 或 Register subscriber 指令訊息中設定 **SubPoint** 內容來繼續運作。 訂閱點會與指令訊息中的主題字串結合，而產生的主題會像任何其他主題一樣處理。

IBM MQ 應用程式不受訂閱點影響。 如果應用程式使用從其中一個相符主題物件繼承資訊的主題，則該應用程式會使用相符的訂閱點與已排入佇列的應用程式交互作業。

範例

現有的 WebSphere Message Broker (現在稱為 IBM Integration Bus) 已移轉至 IBM MQ 的發佈/訂閱應用程式已建立兩個主題物件 GBP 和 USD，並具有對應的主題字串 'GBP' 和 'USD'。

主題 NYSE/IBM/SPOT 的現有發佈者已移轉至在 IBM MQ 上執行，使用訂閱點 USD 在主題上建立發佈 USD/NYSE/IBM/SPOT。 與 NYSE/IBM/SPOT 的現有訂閱者類似，使用訂閱點 USD 建立 USD/NYSE/IBM/SPOT 的訂閱。

透過呼叫 MQSUB，在 IBM MQ 發佈/訂閱程式中訂閱美元現貨價格。 使用 USD topic 物件和主題字串 'NYSE/IBM/SPOT'來建立訂閱，如 'C' 程式碼片段中所示。

```
strncpy(sd.ObjectName, "USD", MQ_TOPIC_NAME_LENGTH);
sd.ObjectString.VSPtr = "NYSE/IBM/SPOT";
sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
```

1. 設定叢集主題主機上 USD 及 GBP 主題物件的 CLUSTER 屬性。
2. 刪除叢集中其他佇列管理程式上 USD 及 GBP 主題物件的所有副本。
3. 確保在叢集中每個佇列管理程式的 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中定義 USD 和 GBP 。

單一佇列管理程式發佈/訂閱配置的範例

第 73 頁的圖 31 說明基本單一佇列管理程式發佈/訂閱配置。 此範例顯示新聞服務的配置，其中發佈者提供數個主題的相關資訊:

- 發佈者 1 使用「體育」主題來發佈體育結果的相關資訊
- 發佈者 2 使用「股票」主題來發佈股價的相關資訊
- 發佈者 3 使用影片主題來發佈電影評論的相關資訊，以及使用電視主題來發佈電視清單的相關資訊

三個訂閱者已登錄對不同主題感興趣，因此佇列管理程式會將他們感興趣的資訊傳送給他們:

- 用戶 1 收到運動成績和股票價格
- 訂閱者 2 收到影片評論
- 訂閱者 3 會接收運動結果

沒有任何使用者對電視清單有興趣，因此這些都沒有被分發。

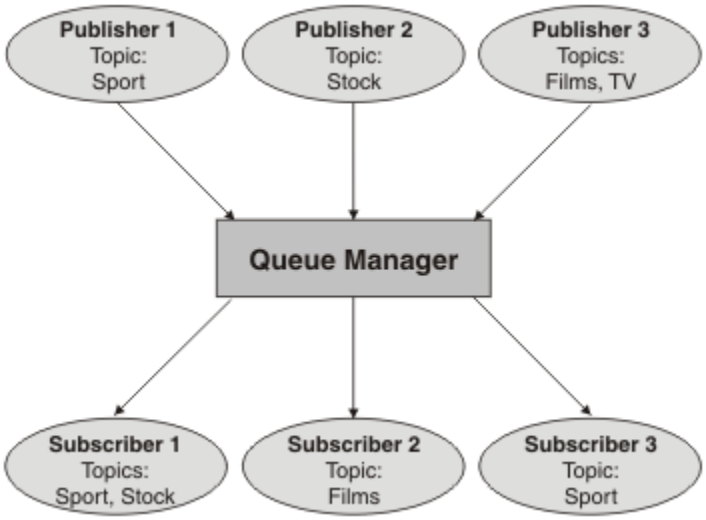


圖 *31:* 單一佇列管理程式發佈/訂閱範例

分散式發佈/訂閱網路

每一個佇列管理程式都會將發佈至主題的訊息與已訂閱該主題的本端建立訂閱進行比對。 您可以配置佇列管理程式的網路，以便將連接至一個佇列管理程式的應用程式所發佈的訊息遞送至網路中其他佇列管理程式上所建立的相符訂閱。 這需要對佇列管理程式之間的簡式通道進行額外配置。

分散式發佈/訂閱配置是一組連接在一起的佇列管理程式。 佇列管理程式可以全部位於相同的實體系統上，也可以分散在數個實體系統上。 當您將佇列管理程式連接在一起時，訂閱者可以訂閱一個佇列管理程式，並接收最初發佈至另一個佇列管理程式的訊息。 為了說明這一點，下圖將第二個佇列管理程式新增至 第 72 頁的『單一佇列管理程式發佈/訂閱配置的範例』中說明的配置。

- 「發佈者 4」使用「佇列管理程式 2」來發佈天氣預測資訊 (使用「天氣」主題)，以及主要道路上交通狀況的相關資訊 (使用「交通」主題)。
- 訂閱者 4 也會使用此佇列管理程式，並使用主題「資料流量」來訂閱資料流量狀況的相關資訊。
- 訂閱者 3 也會訂閱天氣狀況的相關資訊，即使它使用與發佈者不同的佇列管理程式也一樣。 這可能是因為佇列管理程式彼此鏈結。

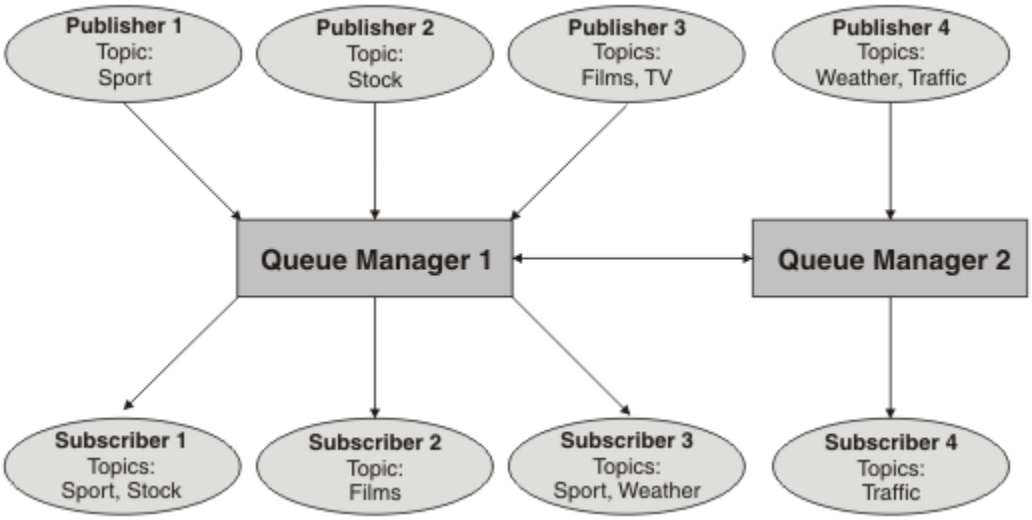


圖 *32:* 具有兩個佇列管理程式的發佈/訂閱範例

您可以手動連接母項和子項階層中的佇列管理程式，也可以建立發佈/訂閱叢集，並讓 IBM MQ 為您定義許多連線詳細資料。 您也可以結合使用這兩種拓蹼，例如在階層中將數個叢集結合在一起。

發佈/訂閱叢集概觀

發佈/訂閱叢集是將一或多個主題物件新增至叢集的標準叢集。 當您在叢集中的任何佇列管理程式上定義 管理主題物件 ，並透過指定叢集名稱來叢集化該主題物件時，該主題的發佈者和訂閱者可以連接至叢集中的任何佇列管理程式，且發佈的訊息會透過佇列管理程式之間的叢集通道遞送至訂閱者。

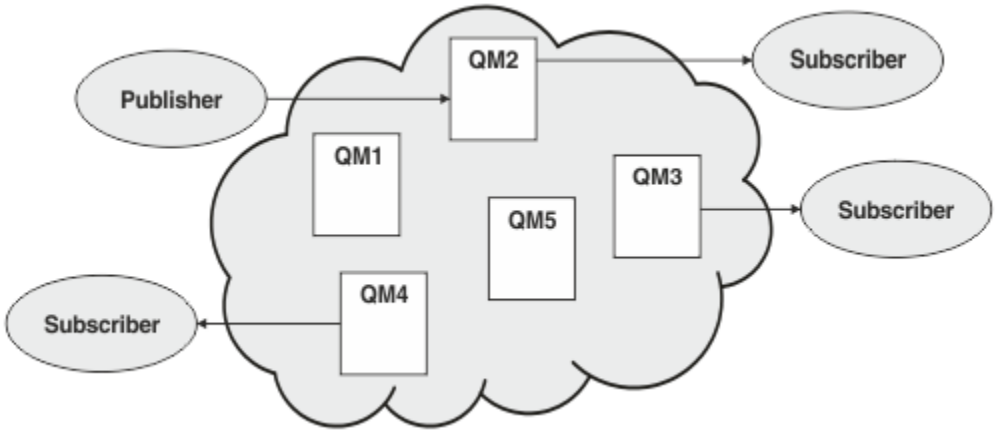


圖 *33:* 發佈/訂閱叢集 *(publish/subscribe cluster)*

有兩種方法可配置如何在叢集中遞送發佈/訂閱訊息:

- 直接遞送 (direct routing)
- 主題主機遞送 (topic host routing)

當您配置直接遞送叢集主題時，在一個佇列管理程式上發佈的訊息會直接從該佇列管理程式傳送至叢集中任何其他佇列管理程式上的每個訂閱。 這可以為發佈提供最直接的路徑，但會導致叢集中的所有佇列管理程式都知道所有其他佇列管理程式，每一個佇列管理程式都可能在它們之間建立叢集通道。

當您使用主題主機遞送時，在某個佇列管理程式上發佈的訊息會從該佇列管理程式傳送至管理受管理主題物件定義的佇列管理程式。 那個主題主機佇列管理程式會將訊息繼續遞送到叢集中任何其他佇列管理程式上的每個訂閱。 如果發佈者或訂閱者不在主題主機佇列管理程式上，這會導致發佈的路徑較長。 不過，好處是只有主題主機佇列管理程式才會知道叢集中的所有其他佇列管理程式，並可能與它們一起建立叢集通道。

如需相關資訊，請參閱第 76 頁的『發佈/訂閱叢集』。

發佈/訂閱階層概觀

發佈/訂閱階層是由通道連接至階層式結構的一組佇列管理程式。 每個佇列管理程式都會識別其 母項 佇列管理程式，如 將佇列管理程式連接至發佈/訂閱階層中所述。

主題的發佈者和訂閱者可以連接至階層中的任何佇列管理程式，並使用階層式佇列管理程式連線功能在它們之間傳送訊息。

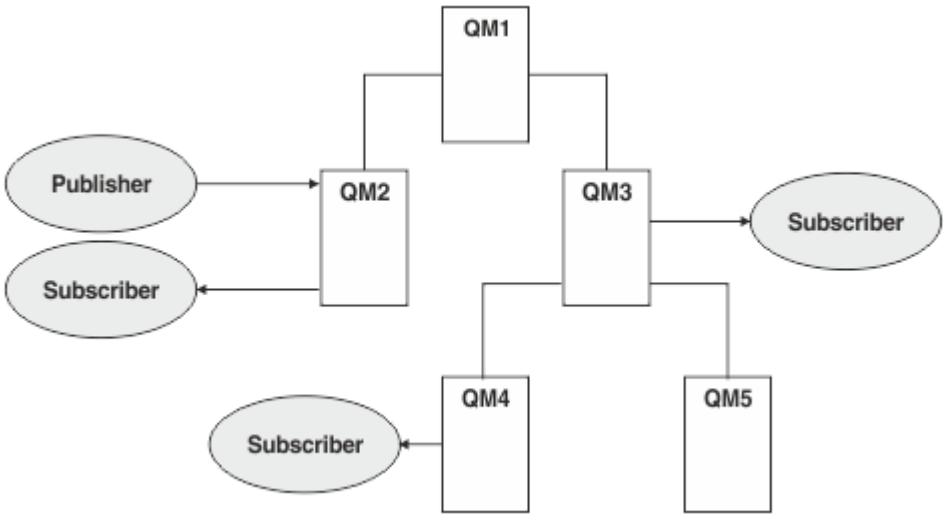


圖 *34:* 發佈/訂閱階層

在上圖中，遞送至 QM3 及 QM4 上的訂閱者的發佈已從 QM2 遞送至 QM1 ， 然後遞送至 QM3 ， 最後遞送至 QM4。

階層可讓您直接控制階層中每個佇列管理程式之間的關係。 這可讓您細部控制從發佈者到訂閱者的訊息遞送，在具有受限連線功能的佇列管理程式網路之間遞送時特別有用。 您應該仔細考量每個佇列管理程式的可用性及功能，透過這些佇列管理程式將訊息從發佈者遞送至訂閱者。

如需相關資訊，請參閱第 78 頁的『發佈/訂閱階層』。

佇列管理程式之間的發佈配送

除了遞送選項之外，還有兩種方法可透過佇列管理程式網路來配送發佈:

- 僅將發佈從一個佇列管理程式傳送至目前管理該發佈的訂閱的佇列管理程式。
- 將每一個發佈傳送至所有佇列管理程式，並讓它們符合其訂閱。

前者只會在必要時傳送發佈訊息，但需要在佇列管理程式之間共用訂閱知識層次。 後者不需要共用訂閱知識，但可能會導致在佇列管理程式之間傳送不必要的發佈訊息。

依預設， IBM MQ 會使用先前的方法，在此方法中，發佈只會傳送至具有其訂閱的佇列管理程式。 訂閱知識以 *Proxy* 訂閱形式在佇列管理程式之間延伸。 它取決於訂閱的配送和生命期限，以及發佈頻率，至於在分散式發佈/訂閱拓蹼中使用最有效率的是哪一個。 請參閱 發佈/訂閱網路中的訂閱效能。

相關概念

第 62 頁的『主題樹狀結構』
您定義的每一個主題就是主題樹狀結構中的一個元素或節點。 主題樹狀結構可以是空的，以開始或包含先前使用 MQSC 或 PCF 指令定義的主題。 您可以定義新主題，方式是使用「建立主題」指令，亦或在發佈或訂閱中第一次指定該主題。

發佈/訂閱階層實務

相關工作

設計發佈/訂閱叢集

發佈/訂閱叢集

發佈/訂閱叢集是交互連接的佇列管理程式的標準叢集，在此叢集上，發佈會自動從發佈應用程式移至存在於叢集中任何佇列管理程式上的訂閱。 要在發佈/訂閱叢集之間遞送發佈有兩個選項：直接遞送和主題主機遞送。 您選擇的遞送取決於叢集的大小及預期活動型樣。

用於發佈/訂閱傳訊的叢集與標準 IBM MQ 叢集沒有不同。 因此，發佈/訂閱叢集內的佇列管理程式可以存在於實際個別的電腦上，且每一對佇列管理程式會在必要時由叢集通道自動連接在一起。 如需相關資訊，請參閱 叢集。

若要為發佈/訂閱傳訊配置佇列管理程式的標準叢集，請在叢集中的佇列管理程式上定義一個以上受管理的主題物件。 若要使主題成為叢集主題，請使用叢集名稱來配置 **CLUSTER** 內容。 當您這樣做時，在 主題樹狀結構 中，發佈者或訂閱者在該點或以下使用的任何主題會在叢集中的所有佇列管理程式之間共用，而發佈至主題樹狀結構叢集分支的訊息會自動遞送至叢集中其他佇列管理程式上的訂閱。

不論目標佇列管理程式上訊息的訂閱者數目為何，在發佈者佇列管理程式與其他佇列管理程式之間只會傳送每則訊息的一個副本。 到達具有一個以上訂閱的佇列管理程式時，會在所有訂閱之間複製訊息。

任何加入叢集的佇列管理程式都會自動察覺叢集主題，而該佇列管理程式上的發佈者和訂閱者會自動參與叢集。

非叢集發佈/訂閱活動也可以在發佈/訂閱叢集中進行，方法是使用不屬於叢集主題物件的主題字串。

要在發佈/訂閱叢集之間遞送發佈有兩個選項：直接遞送和主題主機遞送。 若要選擇要在叢集內使用的訊息遞送，請將受管理主題物件上的 **CLROUTE** 內容設定為下列其中一個值：

- **DIRECT**
- **TOPICHOST**

依預設，主題遞送是 **DIRECT**。 這是 IBM MQ 8.0 之前的唯一選項。 當您在佇列管理程式上配置直接遞送的叢集主題時，叢集裡的所有佇列管理程式便可察覺叢集裡的所有其他佇列管理程式。 當執行發佈及訂閱作業時，每一個佇列管理程式都可以直接連接到叢集中的任何其他佇列管理程式。

從 IBM MQ 8.0 開始，您可以改為將主題遞送配置成 **TOPICHOST**。 當您使用主題主機遞送時，叢集中的所有佇列管理程式會感知到管理遞送主題定義的叢集佇列管理程式（亦即，您定義主題物件所在的佇列管理程式）。 執行發佈和訂閱作業時，叢集中的佇列管理程式只會連接到這些主題主機佇列管理程式，而不會彼此直接連接。 主題主機佇列管理程式負責從已發佈這些發佈資訊的佇列管理程式中，將發佈資訊遞送到具有相符訂閱的佇列管理程式。

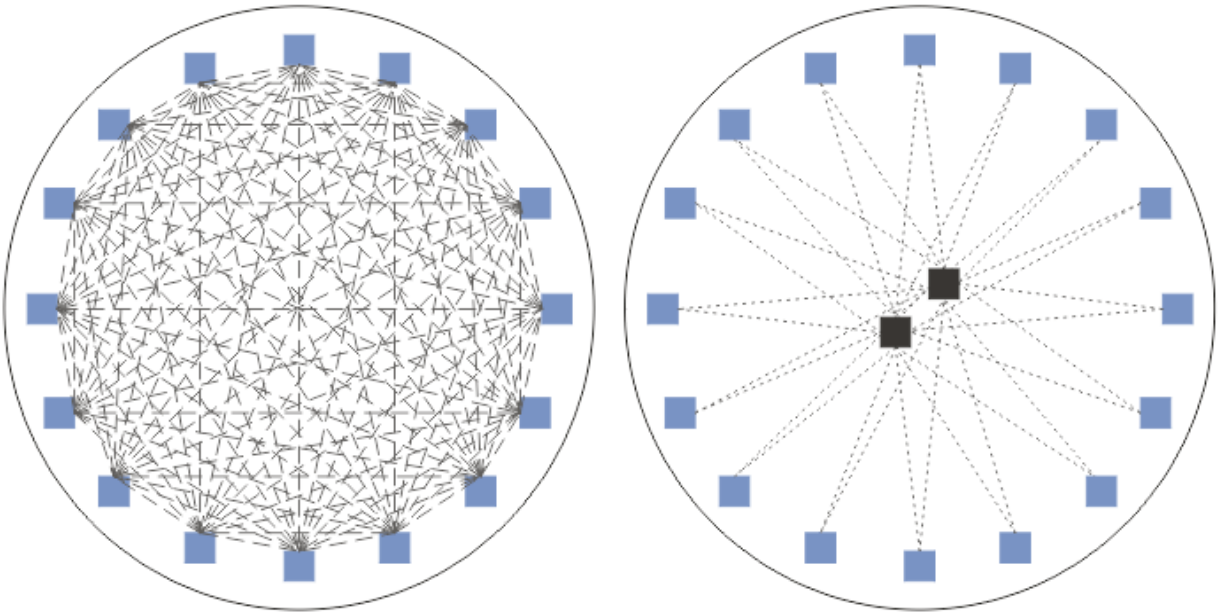


圖 *35:* 直接遞送和主題主機遞送

直接遞送的概觀

當受管理主題物件配置為直接遞送時，只需要在叢集中的其中一個佇列管理程式上定義主題物件，所有佇列管理程式就能瞭解該主題物件。 在其中定義主題的佇列管理程式選項不會影響主題的發佈/訂閱傳訊行為。

每一個訊息會直接從發佈者佇列管理程式流向叢集中其他佇列管理程式上的每一個訂閱，而不會通過任何中間佇列管理程式。

依預設，訊息只會傳送至叢集中管理一個以上訂閱的其他佇列管理程式。

- 這會根據每一個佇列管理程式，直接通知叢集中所有其他佇列管理程式目前具有一個以上訂閱的所有主題。 這會導致叢集中的所有佇列管理程式都知道所訂閱的所有主題，以及管理訂閱的任何佇列管理程式(建立與所有其他佇列管理程式的通道)。 這與每一個佇列管理程式是否有發佈者無關。
- 透過變更為將所有發佈傳送至叢集中所有佇列管理程式的模型，可以移除所有佇列管理程式上每個個別訂閱主題的知識，而不論它們是否具有訂閱。 這會減少訂閱知識資料流量，但可能會增加發佈資料流量及每個佇列管理程式建立的通道數目。 請參閱 發佈/訂閱網路中的訂閱效能。

使用直接遞送叢集主題的發佈/訂閱訊息流程可以跨越多個發佈/訂閱叢集，方法是將每一個叢集中的一個佇列管理程式新增至發佈/訂閱階層。 請參閱 結合多個叢集的主題空間。

如需更詳細的直接遞送探索，請參閱 發佈/訂閱叢集中的直接遞送。

主題主機遞送的概觀

當配置受管理主題物件以進行主題主機遞送時，來自叢集中佇列管理程式的發佈會透過已配置主題物件的佇列管理程式 (「主題主機」) 遞送，並從該佇列管理程式遞送至存在訂閱的佇列管理程式。

- 這會根據每一個佇列管理程式，通知所有主題主機目前具有一個以上訂閱的每一個主題。 任何管理訂閱的佇列管理程式都會針對與訂閱相關的主題，建立每個主題主機的通道。
- 基於發佈/訂閱的目的，不會讓非主題管理佇列管理程式知道叢集中的其他非主題管理佇列管理程式，且不會為此目的在它們之間建立通道。
- 如果發佈應用程式連接至管理主題的佇列管理程式，則發佈的訊息會直接遞送至已建立相符訂閱的佇列管理程式，而不需要額外的「中繼站」。 同樣地，如果在管理主題的唯一佇列管理程式上建立相符訂閱，則發佈至該主題的訊息會直接遞送至該佇列管理程式，而不需要額外的中繼站。
- 在與發佈者相同的佇列管理程式上滿足訂閱，而不先將發佈遞送至主題物件的主機。

對於叢集佇列，多個佇列管理程式可以配置相同的管理主題物件。 這提供較高的訊息遞送可用性，並透過工作量平衡進行水平調整。 對於主題主機遞送主題物件，當多個佇列管理程式為主題樹狀結構的相同分支配置相同的具名主題時，每個管理訂閱的佇列管理程式會讓每個主題主機瞭解訂閱的主題。

- 當發佈訊息時，會將訊息傳送至其中一個主題主機佇列管理程式，以轉遞至管理佇列管理程式的訂閱。 主題主機佇列管理程式的選項遵循與叢集點對點佇列相同的預設工作量平衡規則。
- 如果發佈佇列管理程式無法聯絡一或多個主題主機佇列管理程式，則會將訊息遞送至其餘可用的主題管理佇列管理程式。

主題樹狀結構遞送分支中主題的每個發佈都會轉遞至其中一個主題主機，即使叢集中任何位置沒有該主題的訂閱。 依預設，訊息只會從這裡傳送至叢集中管理一個以上訂閱的其他佇列管理程式。

- 這取決於每一個主題主機佇列管理程式是否收到叢集中每一個佇列管理程式上所有已訂閱主題字串的通知。
- 可以透過變更為將遞送至叢集中所有佇列管理程式之主題主機的所有發佈傳送至叢集中所有佇列管理程式的模型，來移除每個個別訂閱主題的知識，而不論它們是否具有訂閱。 這會減少訂閱知識資料流量，但可能會增加發佈資料流量，並可能增加使用每一個主題管理佇列管理程式所建立的通道數目。 請參閱 發佈/訂閱網路中的訂閱效能。

使用主題主機遞送叢集主題的發佈/訂閱訊息流程， **無法** 透過使用發佈/訂閱階層來跨越多個發佈/訂閱叢集。

如需更詳細的主題主機遞送探索，請參閱 發佈/訂閱叢集中的主題主機遞送。

發佈/訂閱階層

您可以透過使用通道將佇列管理程式鏈結在一起，然後定義佇列管理程式配對之間的子項-母項關係，來建置發佈/訂閱階層。 透過階層中的直接關係，從發佈者到訂閱的訊息流程。 請注意，這可能表示有多個 "躍點" 可到達該處。

在任何一對佇列管理程式之間只會傳送訊息的一個副本，而不考慮目標佇列管理程式上訊息的訂閱者數目。 到達具有一個以上訂閱的佇列管理程式時，會在所有訂閱之間複製訊息。

依預設，訊息只會傳送至階層中的其他佇列管理程式，這些佇列管理程式位於另一個佇列管理程式上訂閱的路徑上:

- 這會根據每一個佇列管理程式，來通知目前在此佇列管理程式或其中一個其他關係上具有一個以上訂閱之所有主題的每一個直接關係。 這會導致階層中的所有佇列管理程式都知道要訂閱的所有主題。
- 此行為可以變更為一律將發佈傳送至階層中的所有佇列管理程式，而不考慮任何現有的訂閱。 這不需要在階層中傳播訂閱資訊，但可以增加發佈資料流量。

當您建立叢集時，必須小心不要建立迴圈，導致訊息在網路內永遠循環。 在階層中無法建立此類迴圈。

每個佇列管理程式都必須具有唯一的佇列管理程式名稱。

發佈/訂閱訊息流程可以跨越多個發佈/訂閱叢集。 若要這樣做，請從每一個叢集新增一個佇列管理程式至發佈/訂閱階層。

如需更詳細的探索，請參閱 在發佈/訂閱階層中遞送。

發佈/訂閱網路中的 Proxy 訂閱

Proxy 訂閱是指一個佇列管理程式針對在另一個佇列管理程式中發佈的主題所進行的訂閱。 Proxy 訂閱會針對某個訂閱所訂閱的每個個別主題字串，在佇列管理程式之間流動。 您不會明確建立 Proxy 訂閱，佇列管理程式會代表您執行此動作。

您可以一起將佇列管理程式連接至發佈/訂閱叢集，或連接至發佈/訂閱階層。 Proxy 訂閱在連接的佇列管理程式之間流動。 Proxy 訂閱會讓連接至某個佇列管理程式的發佈者所建立的主題發佈，由連接至其他佇列管理程式的該主題訂閱者接收。 請參閱 第 73 頁的『分散式發佈/訂閱網路』。

在具有數千個個別主題字串訂閱的發佈/訂閱拓蹼中，或者如果這些訂閱的存在可能正在快速變更，則必須考量 Proxy 訂閱延伸的額外負擔。 除了本主題其餘部分中說明的自動聚集之外，您還可以進行手動配置變更，以進一步限制已連接佇列管理程式之間的 Proxy 訂閱及發佈流程，並減少等待 Proxy 訂閱延伸到所有已連接佇列管理程式的延遲。 請參閱 發佈/訂閱網路中的訂閱效能。

Proxy 訂閱不包含本端訂閱所使用的任何選取元，且可能會簡化包含萬用字元的訂閱主題字串。 這可能導致發佈與實際訂閱不相符的 Proxy 訂閱相符，導致佇列管理程式之間的額外發佈流程。 管理訂閱的佇列管理程式會過濾掉這類不相符，以便不會將其他發佈傳回給訂閱。

Proxy 訂閱聚集

使用重複排除系統來聚集 Proxy 訂閱。 對於特定的已解析主題字串，會在第一個本端訂閱或接收的 Proxy 訂閱上傳送 Proxy 訂閱。 相同主題字串的後續訂閱會使用這個現有的 Proxy 訂閱。

在取消前次本端訂閱或收到的 Proxy 訂閱之後，會取消 Proxy 訂閱。

發佈聚集

當佇列管理程式上有多個相同主題字串的訂閱時，在發佈/訂閱拓蹼中，只會從其他佇列管理程式傳送每一個符合該主題字串之發佈的單一副本。 訊息到達時，本端佇列管理程式會將訊息副本遞送至每一個相符的訂閱。

當 Proxy 訂閱包含萬用字元時，多個 Proxy 訂閱可以符合單一發佈的主題字串。 如果在佇列管理程式上發佈的訊息符合單一連接佇列管理程式所建立的兩個以上 Proxy 訂閱，則只會將發佈的一個副本轉遞至遠端佇列管理程式，以滿足多個 Proxy 訂閱。

相關概念

分散式發佈/訂閱網路中的迴圈偵測

Proxy 訂閱中的萬用字元

訂閱可以在主題字串中使用萬用字元，以符合發佈資訊中的多個主題字串。

訂閱可以使用兩個萬用字元綱目: *topic-based* 和 *characterbased*。 請參閱第 57 頁的『萬用架構』。

在 IBM MQ 中，萬用字元訂閱的所有 Proxy 訂閱都會轉換成使用主題型萬用字元。 如果找到文字型萬用字元，則會將它取代為 # 字元，回到最接近的 /。 例如， /aaa/bbb/c*d 會轉換為 /aaa/bbb/#。 轉換會導致遠端佇列管理程式傳送的發佈數比明確訂閱的發佈數略多。 將發佈遞送給本端訂閱者時，本端佇列管理程式會過濾掉其他發佈。

使用 萬用字元 內容控制萬用字元用法

使用 MQSC **Topic** 萬用字元 內容或對等 PCF Topic `WildcardOperation` 內容來控制將發佈遞送至使用萬用字元主題字串名稱的訂閱者應用程式。 WILDCARD 內容可以具有下列兩個可能值之一:

WILDCARD
關於此主題的萬用字元訂閱的行為。

PASSTHRU
對於比此主題物件的主題字串更不具體的萬用字元式主題所做的訂閱，將接收到對此主題以及比此主題更具體的主題字串所進行的發佈。

BLOCK
對於比此主題物件的主題字串更不具體的萬用字元式主題所做的訂閱，不會接收到對此主題或比此主題更具體的主題字串所進行的發佈。

在定義訂閱時將使用此屬性的值。 如果變更此屬性，則現有訂閱所涵蓋的主題集不會因為此修改而受到影響。 如果在建立或刪除主題物件時拓蹼發生變更，也適用此實務範例；將使用修改後的拓蹼來建立與 WILDCARD 屬性修改後建立的訂閱相符的主題集。 若要針對現有訂閱強制重新評估相符的主題集，則必須重新啟動佇列管理程式。

在範例 第 68 頁的『範例: 建立 Sport 發佈/訂閱叢集』中，您可以遵循步驟來建立 第 66 頁的圖 23 中所示的主題樹狀結構。

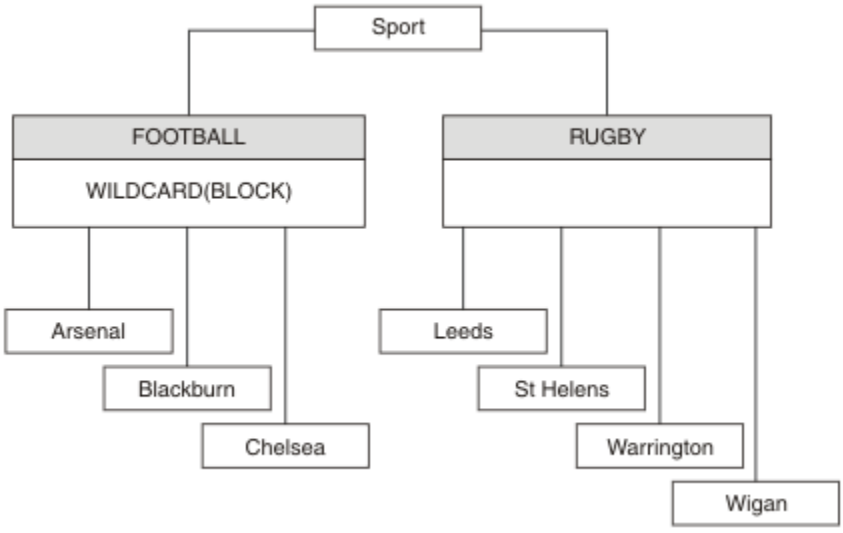


圖 *36:* 使用 萬用字元 內容 *BLOCK* 的主題樹狀結構

使用萬用字元主題字串 # 的訂閱者會接收 `Sport` 主題和 `Sport/Rugby` 子樹狀結構的所有發佈。 訂閱者不會收到 `Sport/Football` 子樹狀結構的發佈，因為 `Sport/Football` 主題的 **萬用字元** 內容值是 `BLOCK`。

PASSTHRU 是預設值。 您可以將 **萬用字元** 內容值 PASSTHRU 設為 `Sport` 樹狀結構中的節點。 如果節點沒有 **萬用字元** 內容值 `BLOCK`，則設定 PASSTHRU 不會變更訂閱者對 `Sports` 樹狀結構中節點所觀察到的行為。

在範例中，建立訂閱以查看萬用字元設定如何影響遞送的發佈; 請參閱 第 70 頁的圖 27。 在 第 71 頁的圖 30 中執行發佈指令，以建立部分發佈。

```
pub QMA
```

圖 *37:* 發佈至 *QMA*

結果顯示在 第 66 頁的表 3 中。 請注意如何設定 WILDCARD 內容值 BLOCK，防止使用萬用字元的訂閱接收對萬用字元範圍內主題的發佈。

表 *6:* 在 *QMA* 上收到的出版品

訂閱	主題字串	收到的出版物	附註
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football 上的 WILDCARD(BLOCK) 封鎖所有 Football 子樹狀結構的發佈
SARSENAL	Sports/#/Arsenal	-	WILDCARD(BLOCK) on Sports/Football 阻止在 Arsenal 上進行萬用字元訂閱
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的預設 WILDCARD 不會阻止 Leeds 上的萬用字元訂閱。

註:

假設訂閱具有萬用字元，其符合具有 WILDCARD 內容值 BLOCK 的主題物件。 如果訂閱在相符萬用字元的右側也有主題字串，則訂閱永遠不會收到發佈。 未封鎖的發佈資訊集是屬於已封鎖萬用字元母項之主題的發佈資訊。 作為主題子項且具有 BLOCK 內容值的主題的發佈會被萬用字元封鎖。 因此，如果訂閱主題字串包含萬用字元右側的主題，則永遠不會收到任何要符合的發佈。

將 WILDCARD 內容值設為 BLOCK 並不表示您無法使用包含萬用字元的主題字串來訂閱。 這類訂閱是正常的。 訂閱有一個明確主題符合主題，且主題物件具有 WILDCARD 內容值 BLOCK。 對於作為主題母項或子項且具有 **萬用字元** 內容值 BLOCK 的主題，它會使用萬用字元。 在 第 66 頁的圖 23 的範例中， Sports/Football/# 之類的訂閱可以接收發佈。

萬用字元和叢集主題

叢集主題定義會延伸到叢集中的每個佇列管理程式。 訂閱叢集中某個佇列管理程式的叢集主題會導致佇列管理程式建立 Proxy 訂閱。 在叢集中的所有其他佇列管理程式上建立 Proxy 訂閱。 使用包含萬用字元且與叢集主題結合的主題字串的訂閱，可能會提供難以預測的行為。 此行為在下列範例中說明。

在範例 第 68 頁的『範例: 建立 Sport 發佈/訂閱叢集』的叢集設定中， QMB 具有與 QMA 相同的訂閱集，但 QMB 在發佈者發佈至 QMA 之後未收到任何發佈，請參閱 第 66 頁的圖 24。 雖然 Sports/Football 及 Sports/Rugby 主題是叢集主題，但 fullsubs.tst 中定義的訂閱不會參照叢集主題。 沒有任何 Proxy 訂閱從 QMB 延伸到 QMA。 如果沒有 Proxy 訂閱，則不會將任何對 QMA 的發佈轉遞至 QMB。

部分訂閱 (例如 Sports/#/Leeds) 似乎參照叢集主題，在此情況下為 Sports/Rugby。 Sports/#/Leeds 訂閱實際上會解析為主題物件 SYSTEM.BASE.TOPIC。

用於解析訂閱 (例如 Sports/#/Leeds) 所參照的主題物件的規則如下。 將主題字串截斷為第一個萬用字元。 掃描左側的主題字串，以尋找具有相關聯管理主題物件的第一個主題。 主題物件可以指定叢集名稱，或定義本端主題物件。 在範例 Sports/#/Leeds 中，截斷之後的主題字串是 Sports，它沒有主題物件，因此 Sports/#/Leeds 繼承自 SYSTEM.BASE.TOPIC，它是本端主題物件。

若要查看訂閱叢集主題如何變更萬用字元傳播的運作方式，請執行批次 Script upsubs.bat。 該 Script 會清除訂閱佇列，並在 fullsubs.tst 中新增叢集主題訂閱。 再次執行 puba.bat 以建立一批出版品; 請參閱 第 66 頁的圖 24。

第 67 頁的表 4 顯示將兩個新訂閱新增至發佈發佈所在之相同佇列管理程式的結果。 結果如預期，新訂閱各會收到一個發佈，而其他訂閱所收到的發佈數則維持不變。 其他叢集佇列管理程式上發生非預期的結果; 請參閱 第 68 頁的表 5。

表 *7:* 在 *QMA* 上收到的出版品			
訂閱	**主題字串**	**收到的出版物**	**附註**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football 上的 WILDCARD(BLOCK) 封鎖所有 Football 子樹狀結構的發佈
SARSENAL	Sports/#/Arsenal	-	WILDCARD(BLOCK) on Sports/Football 阻止在 Arsenal 上進行萬用字元訂閱
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的預設 WILDCARD 不會阻止 Leeds 上的萬用字元訂閱。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 會接收發佈，因為訂閱沒有萬用字元。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 會在任何事件中收到發佈。

第 68 頁的表 5 顯示在 QMB 上新增兩個新訂閱，以及在 QMA 上發佈的結果。 回想一下，如果沒有這兩個新訂閱， QMB 不會收到任何發佈。 如預期，這兩個新的訂閱會接收發佈，因為 Sports/FootBall 和 Sports/Rugby 都是叢集主題。 QMB 已將 Sports/Football/Arsenal 和 Sports/Rugby/Leeds 的 Proxy 訂閱轉遞至 QMA，然後將發佈傳送至 QMB。

非預期的結果是先前未收到任何發佈的兩個訂閱 Sports/# 及 Sports/#/Leeds 現在已收到發佈。 原因是其他訂閱的 Sports/Football/Arsenal 及 Sports/Rugby/Leeds 轉遞至 QMB 的發佈現在可用於連接至 QMB 的任何訂閱者。 因此，對本端主題 Sports/# 及 Sports/#/Leeds 的訂閱會收到 Sports/Rugby/Leeds 發佈。 Sports/#/Arsenal 會繼續不接收發佈，因為「體育/足球」已將其 **萬用字元** 內容值設為 BLOCK。

表 *8:* 在 *QMB* 上收到的出版品			
訂閱	**主題字串**	**收到的出版物**	**附註**
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*WILDCARD(BLOCK)* 在 *Sports/Football* 上封鎖所有發佈至 *Football* 子樹狀結構
SARSENAL	Sports/#/Arsenal	-	WILDCARD(BLOCK) on Sports/Football 阻止在 Arsenal 上進行萬用字元訂閱
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	*Sports/Rugby* 上的預設 *WILDCARD* 不會阻止 *Leeds* 上的萬用字元訂閱。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 會接收發佈，因為訂閱沒有萬用字元。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 會在任何事件中收到發佈。

在大部分應用程式中，一個訂閱不想要影響另一個訂閱的行為。 WILDCARD 內容與值 BLOCK 的一個重要用途是讓相同主題字串 (包含萬用字元) 的訂閱統一運作。 不論訂閱位於與發佈者相同的佇列管理程式上，或位於不同的佇列管理程式上，訂閱的結果都相同。

萬用字元和串流

對於寫入至發佈/訂閱 API 的新應用程式，其效果是 * 的訂閱不會收到任何發佈。 若要接收所有「體育」發佈資訊，您必須訂閱 Sports/*或 Sports/#，以及類似的 Business 發佈資訊。

當發佈/訂閱分配管理系統移轉至更新版本的 IBM MQ 時，現有排入佇列的發佈/訂閱應用程式的行為不會變更。 **Publish**、 **Register Publisher** 或 **Subscriber** 指令中的 **StreamName** 內容對映至已移轉串流的主題名稱。

萬用字元和訂閱點

對於寫入發佈/訂閱 API 的新應用程式，移轉的效果是 * 的訂閱不會收到任何發佈。 若要接收所有「體育」發佈資訊，您必須訂閱 Sports/*或 Sports/#，以及類似的 Business 發佈資訊。

當發佈/訂閱分配管理系統移轉至更新版本的 IBM MQ 時，現有排入佇列的發佈/訂閱應用程式的行為不會變更。 **Publish**、 **Register Publisher** 或 **Subscriber** 指令中的 **SubPoint** 內容對映至已移轉訂閱的主題名稱。

範例: 建立 Sport 發佈/訂閱叢集

接下來的步驟會建立叢集 CL1，其中包含四個佇列管理程式: 兩個完整儲存庫、 CL1A 和 CL1B，以及兩個局部儲存庫 QMA 和 QMB。 完整儲存庫用來只保留叢集定義。 QMA 指定為叢集主題主機。 可延續訂閱同時定義在 QMA 和 QMB 上。

註: 此範例是針對 Windows 進行編碼。 您必須重新編碼 建立 qmgrs.bat 及 建立 pub.bat ，以在其他平台上配置及測試範例。

1. 建立 Script 檔。
   a. 建立 topics.tst
   b. 建立 wildsubs.tst
   c. 建立 fullsubs.tst
   d. 建立 qmgrs.bat
   e. 建立 pub.bat
2. 執行 Create qmgrs.bat 以建立配置。

```
qmgrs
```

在 第 66 頁的圖 23 中建立主題。 圖 5 中的 Script 會建立叢集主題 Sports/Football 和 Sports/Rugby。

註: REPLACE 選項不會取代主題的 TOPICSTR 內容。 TOPICSTR 是在範例中用來測試不同主題樹狀結構的有用內容。 若要變更主題，請先刪除主題。

```
DELETE TOPIC ('Sports')
DELETE TOPIC ('Football')
DELETE TOPIC ('Arsenal')
DELETE TOPIC ('Blackburn')
DELETE TOPIC ('Chelsea')
DELETE TOPIC ('Rugby')
DELETE TOPIC ('Leeds')
DELETE TOPIC ('Wigan')
DELETE TOPIC ('Warrington')
DELETE TOPIC ('St. Helens')

DEFINE TOPIC ('Sports')   TOPICSTR('Sports')
DEFINE TOPIC ('Football')  TOPICSTR('Sports/Football') CLUSTER(CL1) WILDCARD(BLOCK)
DEFINE TOPIC ('Arsenal')  TOPICSTR('Sports/Football/Arsenal')
DEFINE TOPIC ('Blackburn') TOPICSTR('Sports/Football/Blackburn')
DEFINE TOPIC ('Chelsea')  TOPICSTR('Sports/Football/Chelsea')
DEFINE TOPIC ('Rugby')   TOPICSTR('Sports/Rugby')  CLUSTER(CL1)
DEFINE TOPIC ('Leeds')   TOPICSTR('Sports/Rugby/Leeds')
DEFINE TOPIC ('Wigan')   TOPICSTR('Sports/Rugby/Wigan')
DEFINE TOPIC ('Warrington') TOPICSTR('Sports/Rugby/Warrington')
DEFINE TOPIC ('St. Helens') TOPICSTR('Sports/Rugby/St. Helens')
```

圖 *38:* 刪除並建立主題*: topics.tst*

註: 刪除主題，因為 REPLACE 不會取代主題字串。

使用萬用字元建立訂閱。 萬用字元會對應主題與 第 66 頁的圖 23 中的主題物件。 為每一個訂閱建立佇列。 執行或重新執行 Script 時，會清除佇列並刪除訂閱。

註: REPLACE 選項不會取代訂閱的 TOPICOBJ 或 TOPICSTR 內容。 TOPICOBJ 或 TOPICSTR 是在測試不同訂閱的範例中有效改變的內容。 若要變更它們，請先刪除訂閱。

```
DEFINE QLOCAL(QSPORTS)  REPLACE
DEFINE QLOCAL(QSARSENAL) REPLACE
DEFINE QLOCAL(QSLEEDS)  REPLACE
CLEAR QLOCAL(QSPORTS)
CLEAR QLOCAL(QSARSENAL)
CLEAR QLOCAL(QSLEEDS)

DELETE SUB  (SPORTS)
DELETE SUB  (SARSENAL)
DELETE SUB  (SLEEDS)
DEFINE SUB  (SPORTS)  TOPICSTR('Sports/#')     DEST(QSPORTS)
DEFINE SUB  (SARSENAL) TOPICSTR('Sports/+/Arsenal') DEST(QSARSENAL)
DEFINE SUB  (SLEEDS)  TOPICSTR('Sports/+/Leeds')  DEST(QSLEEDS)
```

圖 *39:* 建立萬用字元訂閱*: wildsubs.tst*

建立參照叢集主題物件的訂閱。

註:

定界字元 / 會自動插入 TOPICOBJ 所參照的主題字串與 TOPICSTR 所定義的主題字串之間。

DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) 定義會建立相同的訂閱。 TOPICOBJ 是用來作為參照您已定義之主題字串的快速方式。 建立的訂閱不再參照主題物件。

```
DEFINE QLOCAL(QFARSENAL) REPLACE
DEFINE QLOCAL(QRLEEDS)  REPLACE
CLEAR QLOCAL(QFARSENAL)
CLEAR QLOCAL(QRLEEDS)

DELETE SUB  (FARSENAL)
DELETE SUB  (RLEEDS)
DEFINE SUB  (FARSENAL) TOPICOBJ('Football') TOPICSTR('Arsenal') DEST(QFARSENAL)
DEFINE SUB  (RLEEDS)  TOPICOBJ('Rugby')  TOPICSTR('Leeds')  DEST(QRLEEDS)
```

圖 *40:* 刪除並建立訂閱*: fullsubs.tst*

建立具有兩個儲存庫的叢集。 建立兩個局部儲存庫以進行發佈和訂閱。 請重新執行 Script 以刪除所有項目，然後重新啟動。 Script 也會建立主題階層及起始萬用字元訂閱。

註:

在其他平台上，撰寫類似的 Script ，或輸入所有指令。 使用 Script 可讓您快速刪除所有項目，並以相同的配置重新開始。

```
@echo off
set port.CL1B=1421
set port.CLIA=1420
for %%A in (CL1A CL1B QMA QMB) do call :createQM %%A
call :configureQM CL1A CL1B %port.CL1B% full
call :configureQM CL1B CL1A %port.CL1A% full
for %%A in (QMA QMB) do call :configureQM %%A CL1A %port.CL1A% partial
for %%A in (topics.tst wildsubs.tst) do runmqsc QMA < %%A
for %%A in (wildsubs.tst) do runmqsc QMB < %%A
goto:eof

:createQM
echo Configure Queue manager %1
endmqm -p %1
for %%B in (dlt crt str) do %%Bmqm %1
goto:eof

:configureQM
if %1==CL1A set p=1420
if %1==CL1B set p=1421
if %1==QMA set p=1422
if %1==QMB set p=1423
echo configure %1 on port %p% connected to repository %2 on port %3 as %4 repository
echo DEFINE LISTENER(LST%1) TRPTYPE(TCP) PORT(%p%) CONTROL(QMGR) REPLACE | runmqsc %1
echo START LISTENER(LST%1) | runmqsc %1
if full==%4 echo ALTER QMGR REPOS(CL1) DEADQ(SYSTEM.DEAD.LETTER.QUEUE) | runmqsc %1
echo DEFINE CHANNEL(TO.%2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('LOCALHOST(%3)') CLUSTER(CL1)
REPLACE | runmqsc %1
echo DEFINE CHANNEL(TO.%1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('LOCALHOST(%p%)')
CLUSTER(CL1) REPLACE | runmqsc %1
goto:eof
```

圖 *41:* 建立佇列管理程式*: qmgrs.bat*

透過將訂閱新增至叢集主題來更新配置。

```
@echo off
for %%A in (QMA QMB) do runmqsc %%A < wildsubs.tst
for %%A in (QMA QMB) do runmqsc %%A < upsubs.tst
```

圖 *42:* 更新訂閱*: upsubs.bat*

執行 pub.bat，並以佇列管理程式作為參數，以發佈包含發佈主題字串的訊息。 Pub.bat 使用範例程式 **amqspub**。

```
@echo off
@rem Provide queue manager name as a parameter
set S=Sports
set S=6 Sports/Football Sports/Football/Arsenal
set S=6 Sports/Rugby Sports/Rugby/Leeds
for %%B in (6) do echo %%B | amqspub %%B %1
```

圖 *43:* 發佈*: pub.bat*

相關概念

萬用字元訂閱及保留的發佈

發佈範圍

當您配置發佈/訂閱叢集或階層時，發佈的範圍會進一步控制佇列管理程式是否將發佈轉遞至遠端佇列管理程式。 使用 **PUBSCOPE** 主題屬性來管理發佈範圍。

如果發佈未轉遞至遠端佇列管理程式，則只有本端訂閱者才會接收發佈。

當您使用發佈/訂閱叢集時，發佈範圍主要是由主題樹狀結構中特定點的叢集主題物件定義所控制。 發佈範圍必須設為容許發佈流至叢集中的其他佇列管理程式。 當您需要細部控制特定佇列管理程式上的特定主題時，應該只限制叢集主題的發佈範圍。

當您使用發佈/訂閱階層時，發佈範圍主要是由這個屬性結合 訂閱範圍 屬性來控制。

PUBSCOPE 屬性用來決定特定主題的發佈範圍。 您可以將屬性設為下列其中一個值:

QMGR
發佈只會遞送給本端訂閱者。 這些出版品稱為 本端出版品。 本端發佈不會轉遞至遠端佇列管理程式，因此連接至遠端佇列管理程式的訂閱者不會接收。

ALL
發佈會遞送給本端訂閱者，以及連接至發佈/訂閱叢集或階層中遠端佇列管理程式的訂閱者。 這些出版品稱為 廣域出版品。

如母項
使用主題樹狀結構中上層主題的 **PUBSCOPE** 設定。

發佈者也可以使用 MQPMO_SCOPE_QMGR 放置訊息選項來指定發佈是本端還是廣域。 如果使用此選項，它會置換已使用 **PUBSCOPE** topic 屬性設定的任何行為。

相關概念
第 63 頁的『管理主題物件』
使用管理主題物件，您可以將特定非預設屬性指派給主題。

相關工作
配置分散式發佈/訂閱網路

訂閱範圍

訂閱的範圍可控制某個佇列管理程式上的訂閱是否接收發佈在發佈/訂閱叢集或階層中的另一個佇列管理程式上發佈的發佈，或只接收來自本端發佈者的發佈。

將訂閱範圍限制為佇列管理程式會停止將 Proxy 訂閱轉遞至發佈/訂閱拓蹼中的其他佇列管理程式。 這會減少佇列管理程式之間的發佈/訂閱傳訊資料流量。

當您使用發佈/訂閱叢集時，訂閱範圍主要是由主題樹狀結構中特定點的叢集主題物件定義所控制。 必須設定訂閱範圍，以容許 Proxy 訂閱流向叢集中的其他佇列管理程式。 當您需要細部控制特定佇列管理程式上的特定主題時，應該只限制叢集主題的訂閱範圍。

當您使用發佈/訂閱階層時，訂閱的範圍主要是由這個屬性結合 發佈範圍 屬性來控制。

SUBSCOPE 主題屬性用來決定對特定主題所做的訂閱範圍。 您可以將屬性設為下列其中一個值:

QMGR
訂閱只會接收本端發佈， Proxy 訂閱不會延伸到遠端佇列管理程式。

ALL
Proxy 訂閱會延伸到發佈/訂閱叢集或階層中的遠端佇列管理程式，且訂閱者會接收本端和遠端發佈。

如母項
使用主題樹狀結構中上層主題的 **SUBSCOPE** 設定。

當主題的訂閱範圍設為 ALL(直接或透過 ASPARENT 解決) 時，該主題的個別訂閱可以在建立訂閱時指定 MQSO_SCOPE_QMGR ，將其範圍限制為 QMGR 。 對具有 QMGR 範圍之主題的訂閱無法將範圍擴大至 ALL。

相關概念
第 63 頁的『管理主題物件』
使用管理主題物件，您可以將特定非預設屬性指派給主題。

相關工作
配置分散式發佈/訂閱網路

主題空間

主題空間是您可以訂閱及發佈的主題集。 分散式發佈/訂閱拓蹼中的佇列管理程式有一個主題空間，可能包含已在該拓蹼中所連接佇列管理程式上訂閱及發佈至的主題。

註: 如需佇列管理程式內主題 (例如管理主題物件、主題字串及主題樹狀結構) 的概觀，請參閱第 56 頁的『主題』。 除非另有指定，否則現行文章中 主題 的進一步參照會參照 主題字串 。

一開始會以下列其中一種方式來建立主題:

- 在管理上，當您定義主題物件或可延續訂閱時。
- 當應用程式動態建立發佈或訂閱至新主題時。

主題會透過 Proxy 訂閱，以及透過建立管理叢集主題物件，傳送至其他佇列管理程式。 Proxy 訂閱會導致發佈從發佈者所連接的佇列管理程式轉遞至訂閱者的佇列管理程式。

Proxy 訂閱會在佇列管理程式階層中由上下代關係連接在一起的所有佇列管理程式之間延伸。 結果是您可以在一個佇列管理程式上訂閱階層中任何其他佇列管理程式上定義的主題。 只要佇列管理程式之間有連接路徑，就不會影響佇列管理程式的連接方式。

對於發佈/訂閱叢集中叢集主題的訂閱，也會延伸 Proxy 訂閱。 叢集主題是連接至具有 **CLUSTER** 屬性的主題物件的主題，或從其母項繼承屬性的主題。 非叢集主題的主題稱為本端主題，且不會抄寫至叢集。 沒有 Proxy 訂閱會從本端主題的訂閱延伸到叢集。

總而言之，在兩種情況下會為訂閱者建立 Proxy 訂閱。

1. 佇列管理程式是階層的成員， Proxy 訂閱會轉遞至佇列管理程式的母項及子項。
2. 佇列管理程式是叢集的成員，訂閱主題字串會解析成與叢集主題物件相關聯的主題。 當主題是 直接遞送 叢集主題時， Proxy 訂閱會轉遞至叢集的所有成員。 當主題是 主題主機遞送 叢集主題時， Proxy 訂閱只會轉遞至叢集中已定義叢集主題物件的佇列管理程式。 如需相關資訊，請參閱第 76 頁的『發佈/訂閱叢集』。

如果佇列管理程式是叢集和階層的成員，這兩種機制會傳送 Proxy 訂閱，而不會將重複的發佈遞送給訂閱者。

下列清單中說明三個發佈/訂閱拓蹼的主題空間:

- 第 86 頁的『案例 1. 發佈/訂閱叢集』.
- 第 87 頁的『案件 2. 發佈/訂閱階層』.

在個別主題中，下列配置作業說明如何結合主題空間。

- 在發佈/訂閱叢集中建立單一主題空間。
- 結合多個叢集的主題空間。
- 結合及隔離多個叢集中的主題空間。
- 發佈及訂閱多個叢集中的主題空間。

案例 1. 發佈/訂閱叢集

在範例中，假設佇列管理程式未連接至發佈/訂閱階層。

如果佇列管理程式是發佈/訂閱叢集的成員，則其主題空間是由本端主題及叢集主題組成。 本端主題與不含 **CLUSTER** 屬性的主題物件相關聯。 如果佇列管理程式具有本端主題物件定義，則其主題空間不同於叢集中的另一個佇列管理程式，該佇列管理程式也具有自己本端定義的主題物件。

在發佈/訂閱叢集中，除非您訂閱的主題解析為叢集主題物件，否則無法訂閱另一個佇列管理程式上定義的主題。

當多個佇列管理程式上需要叢集主題物件的相同具名定義時 (例如，當使用 主題主機遞送時) ，所有定義在必要時都必須相符是很重要的。 如需相關資訊，請參閱 在發佈/訂閱叢集中建立單一主題空間。

主題物件的本端定義 (不論定義是用於叢集主題或本端主題) 優先於叢集中其他位置所定義的相同主題物件。 使用本端定義的主題，即使在其他位置定義的物件較新也一樣。

叢集主題物件必須與叢集中每個位置的相同主題字串相關聯。 您無法修改與主題物件相關聯的主題字串。 若要將相同的主題物件與不同的主題字串相關聯，您必須刪除主題物件，並以新的主題字串重建它。 如果主題已叢集化，則效果是刪除儲存在叢集其他成員上的主題物件副本，然後在叢集中到處建立新主題物件的副本。 主題物件的副本都參照相同的主題字串。

在叢集中的不同佇列管理程式上，可能會意外使用不同的主題字串來建立兩個相同具名主題物件的定義。 這可能會導致混淆行為，因為根據參照主題的方式及位置，具有不同主題字串之相同主題物件的多個定義可能會產生不同的結果。 如需此重要點的相關資訊，請參閱 相同名稱的多個叢集主題定義 。

案件 2. 發佈/訂閱階層

在範例中，假設佇列管理程式不是發佈/訂閱叢集的成員。

在 IBM MQ 中，如果佇列管理程式是發佈/訂閱階層的成員，則其主題空間包含在本端及連接的佇列管理程式上定義的所有主題。 階層中所有佇列管理程式的主題空間相同。 沒有將主題劃分為本端主題和廣域主題。

將 **PUBSCOPE** 及 **SUBSCOPE** 選項任一設為 QMGR，以防止主題從發佈者傳送至階層中連接至不同佇列管理程式的訂閱者。

假設您在佇列管理程式 QMA 上使用主題字串 USA/Alabama 來定義主題物件 Alabama 。 結果如下:

1. QMA 中的主題空間現在包括主題物件 Alabama 及主題字串 USA/Alabama。
2. 應用程式或管理者可以使用主題物件名稱 Alabama 在 QMA 建立訂閱。
3. 應用程式可以在階層中的任何佇列管理程式上建立任何主題 (包括 USA/Alabama) 的訂閱。 如果尚未在本端定義 QMA ，則主題 USA/Alabama 會解析為主題物件 SYSTEM.BASE.TOPIC。

相關概念

第 84 頁的『發佈範圍』
當您配置發佈/訂閱叢集或階層時，發佈的範圍會進一步控制佇列管理程式是否將發佈轉遞至遠端佇列管理程式。 使用 **PUBSCOPE** 主題屬性來管理發佈範圍。

第 85 頁的『訂閱範圍』
訂閱的範圍可控制某個佇列管理程式上的訂閱是否接收發佈在發佈/訂閱叢集或階層中的另一個佇列管理程式上發佈的發佈，或只接收來自本端發佈者的發佈。

相關工作

配置分散式發佈/訂閱網路

IBM MQ 多重播送

IBM MQ 多重播送提供低延遲、高扇出、可靠的多重播送傳訊功能。

多重播送是發佈/訂閱傳訊的有效形式，因為它可以調整為大量訂閱者，而不會對效能造成不利影響。 為了進行可靠的「多重播送」傳訊，IBM MQ 使用確認通知、負值確認通知及序號來達成高度展開的低延遲傳訊。

IBM MQ 多重播送的公平遞送可實現近乎同時遞送，從而確保任何收件者都沒有優先權。 由於 IBM MQ 多重播送使用網路來遞送訊息，因此扇出資料不需要發佈/訂閱引擎。 將主題對映至群組位址之後，不需要佇列管理程式，因為發佈者和訂閱者可以在對等模式下運作。 這樣可以降低佇列管理程式伺服器的負載，且佇列管理程式伺服器就不會再成為可能的故障點。

起始多重播送概念

IBM MQ 使用「通訊資訊 (COMMINFO)」物件，可以輕鬆地將「多重播送」整合至現有系統及應用程式。 兩個 TOPIC 物件欄位可啟用現有 TOPIC 物件的快速配置，以支援或忽略多重播送資料流量。

多重播送所需的物件

下列資訊是 IBM MQ Multicast 所需的兩個物件的簡要概觀:

COMMINFO 物件
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。

「必須」設定的唯一 COMMINFO 欄位是 COMMINFO 物件的名稱。 然後會使用這個名稱來識別主題的 COMMINFO 物件。 必須檢查 COMMINFO 物件的 **GRPADDR** 欄位，以確保該值是有效的多重播送群組位址。

TOPIC 物件
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。 如需 TOPIC 物件參數的相關資訊，請參閱 DEFINE TOPIC。

透過變更下列 TOPIC 物件參數的值，可以將現有主題與多重播送搭配使用: **COMMINFO** 及 **MCAST**。

- **COMMINFO** 此參數指定多重播送通訊資訊物件的名稱。
- **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。 依預設， **MCAST** 會設為 ASPARENT，表示主題的多重播送屬性繼承自母項。 將 **MCAST** 設為 ENABLED 可容許此節點上的多重播送資料流量。

多重播送網路及主題

下列資訊是具有不同訂閱類型及主題定義之訂閱的概觀。 這些範例都假設 TOPIC 物件 **COMMINFO** 參數設為有效 COMMINFO 物件的名稱:

已啟用多重播送的主題集
如果主題字串 **MCAST** 參數設為 ENABLED，則容許具有多重播送功能用戶端的訂閱，並建立多重播送訂閱，除非:

- 它是來自具有多重播送功能之用戶端的可延續訂閱。
- 它是來自具有多重播送功能之用戶端的非受管理訂閱。
- 它是來自具有非多重播送功能的用戶端的訂閱。

在這些情況下，會建立非多重播送訂閱，並將 訂閱降級為正常發佈/訂閱。

已停用主題設為多重播送
如果主題字串 **MCAST** 參數設為 DISABLED，一律會建立非多重播送訂閱，且 訂閱降級為正常發佈/訂閱。

主題僅設為多重播送
如果主題字串 **MCAST** 參數設為 ONLY，則容許從具有多重播送功能的用戶端進行訂閱，並建立多重播送訂閱，除非:

- 它是可延續訂閱: 拒絕可延續訂閱，原因碼為 2436 (0984) (RC2436) :MQRC_DURABILITY_NOT_ALLOWED
- 它是非受管理訂閱: 拒絕非受管理訂閱，原因碼為 2046 (07FE) (RC2046) :MQRC_OPTIONS_ERROR
- 它是來自具有非多重播送功能之用戶端的訂閱: 這些訂閱遭到拒絕，原因碼為 2560 (0A00) (RC2560) :MQRC_MULTICAST_only
- 它是來自本端連結應用程式的訂閱: 這些訂閱遭到拒絕，原因碼為 2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY

Windows Linux AIX MQ Telemetry 概觀

MQ Telemetry 包含屬於佇列管理程式一部分的遙測 (MQXR) 服務、您可以自行撰寫或免費下載的遙測用戶端，以及指令行和瀏覽器管理介面。 遙測是指從各種遠端裝置收集資料以及管理這些裝置。 使用 MQ Telemetry，您可以將裝置的資料收集及控制與 Web 應用程式整合。

MQ Telemetry 是 IBM MQ 的元件。 這些版本的升級基本上是安裝更新版本的 IBM MQ。

範例應用程式仍可從 Eclipse Paho 和 MQTT.org 免費取得。 請參閱 IBM MQ Telemetry Transport 程式範例。

因為 MQ Telemetry 是 IBM MQ 的元件，所以 MQ Telemetry 可以與主要產品一起安裝，或在安裝主要產品之後安裝。 如需移轉資訊，請參閱 在 Windows 上移轉 MQ Telemetry 和 在 Linux 上移轉 MQ Telemetry。

MQ Telemetry 中包含下列元件:

遙測通道
使用遙測通道來管理 MQTT 用戶端與 IBM MQ 的連線。 遙測通道會使用新的 IBM MQ 物件 (例如 SYSTEM.MQTT.TRANSMIT.QUEUE) 來與 IBM MQ 互動。

遙測 (MQXR) 服務
MQTT 用戶端使用 SYSTEM.MQXR.SERVICE 遙測服務來連接至遙測通道。

IBM MQ Explorer 其支援 MQ Telemetry
MQ Telemetry 可以使用 IBM MQ Explorer 來管理。

文件
MQ Telemetry 說明文件包含在標準 IBM MQ 產品說明文件中。 Java 和 C 用戶端的 SDK 文件在產品說明文件中提供，並以 Javadoc 和 HTML 形式提供。

遙測概念

您可以從周圍的環境收集資訊，以決定要執行的動作。 身為消費者，在決定要購買哪些食物之前，您先檢查商店中的商店。 您想要知道如果您現在離開，在預訂連線之前，旅程將會花費多久時間。 在決定是否去看醫生之前你先檢查一下你的症狀 在決定是否等待之前，您先檢查巴士何時抵達。 這些決策的資訊直接來自儀表和裝置，來自紙上或螢幕上的書面文字，以及來自您的資訊。 在任何地方，當您需要時，您可以收集資訊，將它結合在一起，分析它，並對它採取行動。

如果資訊來源廣泛分散或無法存取，則收集最準確的資訊會變得困難且代價高昂。 如果您要進行許多變更，或很難進行變更，則不會進行變更，或會在變更無效時進行。

如果將具有數位技術的裝置連接至網際網路，以大幅降低從裝置收集資訊及控制廣泛分散裝置的成本，該怎麼辦? 可以使用網際網路和企業的資源來分析資訊。 您有更多機會做出明智決策並採取行動。

技術趨勢以及環境和經濟壓力正在推動這些變化的發生:

1. 由於與低成本數字處理器的標準化和連線，連線和控制感測器和致動器的成本正在降低。
2. 網際網路和網際網路技術越來越被用來連接裝置。 在一些國家，行動電話在連線網際網路應用的數量上超過個人電腦。 其他裝置肯定在追蹤。
3. 網際網路和網際網路技術讓應用程式更容易取得資料。 輕鬆存取資料會推動使用資料分析，將資料從感應器轉換成在許多其他解決方案中很有用的資訊。
4. 智慧型資源使用通常是一種更快速且更便宜的方式來減少碳排放及成本。 替代方案: 尋找新資源或開發新技術以使用現有資源，可能是長期解決方案。 短期而言，開發新技術或尋找新資源往往比改善現有解決方案風險更大、更緩慢且更昂貴。

範例

範例顯示這些趨勢如何創造新的機會，以智慧方式與環境互動。

The International Convention for the Safety of Life at Sea (SOLAS) requires Automatic Identification System (AIS) to be deployed on many ships. 它需要 300 噸以上的商船和客船。 AIS 主要是沿海航運的避免碰撞系統。 它被海洋當局用來監測和控制沿海水域。

世界各地的發燒友正在部署低成本 AIS 追蹤站，並將沿海航運資訊放到網際網路上。 其他愛好者正在撰寫將來自 AIS 的資訊與來自網際網路的其他資訊相結合的應用程式。 結果會放在網站上，並使用 Twitter 和 SMS 來發佈。

在一份申請中，來自南安普頓附近 AIS 站的資訊與船舶所有權和地理資訊相結合。 應用程式會將渡船抵達及離開的相關即時資訊提供給 Twitter。 使用南安普頓和懷特島之間渡輪的普通通勤者使用 Twitter 或 SMS 訂閱新聞資訊。 如果訊號顯示他們的渡輪晚了，乘客可以推遲他們的出發時間，並在渡輪抵達時間晚於預定的到達時間時搭船。

如需其他範例，請參閱 第 91 頁的『遙測使用案例』。

相關工作

正在安裝 MQ Telemetry

管理 MQ Telemetry

在 Windows 上移轉 MQ Telemetry

在 Linux 上移轉 MQ Telemetry

開發適用於 MQ Telemetry 的應用程式

MQ Telemetry 問題疑難排解

相關參考

MQ Telemetry 參照

Windows Linux AIX

MQ Telemetry 簡介

人們、企業和政府越來越希望使用 MQ Telemetry 與我們生活和工作的環境進行更聰明的互動。 MQ Telemetry 會將所有類型的裝置連接至網際網路及企業，並減少為智慧型裝置建置應用程式的成本。

什麼是 MQ Telemetry?

- 它是 IBM MQ 的一項特性，可將 IBM MQ 所提供的通用傳訊骨幹延伸至各種遠端感應器、掣動器及遙測裝置。 MQ Telemetry 延伸 IBM MQ ，以便它可以將智慧型企業應用程式、服務及決策者與已檢測裝置的網路交互連接。
- MQ Telemetry 的核心部分如下:

  MQ Telemetry (MQXR) 服務。
  此服務在 IBM MQ 伺服器內執行，並使用 IBM MQ Telemetry Transport (MQTT) 通訊協定與遙測裝置進行通訊。

  您撰寫的 MQTT 應用程式。
  這些應用程式控制遙測裝置與 IBM MQ 佇列管理程式之間所攜帶的資訊，以及為了回應該資訊而採取的任何動作。 為了協助建立這些應用程式，您可以使用 MQTT 用戶端程式庫。

它能為我做什麼?

- MQTT 是一種開放式傳訊傳輸，容許為各種裝置建立 MQTT 實作。
- MQTT 用戶端可以在具有有限資源的小型覆蓋區裝置上執行。
- MQTT 在頻寬較低、傳送資料成本昂貴或可能很脆弱的網路上有效運作。
- 訊息遞送已確定並與應用程式取消連結。
- 應用程式設計師不需要具備通訊程式設計知識。
- 訊息可以與其他傳訊應用程式交換。 這些可以是另一個遙測應用程式，或 MQI、 JMS 或企業傳訊應用程式。

如何使用它?

- 所提供的範例 Script 會使用範例 IBM MQ Telemetry Transport v3 用戶端應用程式 (`mqttv3app.jar`)。請參閱 IBM MQ Telemetry Transport 程式範例。
- 使用 IBM MQ Explorer 及其相關聯工具來管理 IBM MQ 的遙測特性。
- 使用用戶端程式庫來協助您建立 MQTT 應用程式，以連接至佇列管理程式，並使用發佈/訂閱傳訊。
- 將應用程式及用戶端程式庫配送至要執行應用程式的裝置。

如何運作?

- MQTT 是發佈訂閱通訊協定。 MQTT 用戶端應用程式可以將訊息發佈至 MQTT 伺服器，或訂閱連接至 MQTT 伺服器的應用程式所傳送的訊息。
- MQTT 用戶端應用程式使用實作 MQTT 訊息傳輸的用戶端程式庫。
- 基本 MQTT 用戶端應用程式的運作方式類似於標準 MQ 用戶端，但可以在更廣泛的平台和網路上執行。

- MQ Telemetry (MQXR) 服務會將 IBM MQ 佇列管理程式轉換為 MQTT 伺服器。
- 當 IBM MQ 佇列管理程式充當 MQTT 伺服器時，連接至佇列管理程式的其他應用程式可以訂閱及接收來自 MQTT 用戶端的訊息。
- 佇列管理程式充當路由器將訊息從發佈應用程式配送至訂閱應用程式。
- 訊息可以在不同類型的用戶端應用程式之間配送。 例如，在 Telemetry 用戶端與 JMS 用戶端之間。

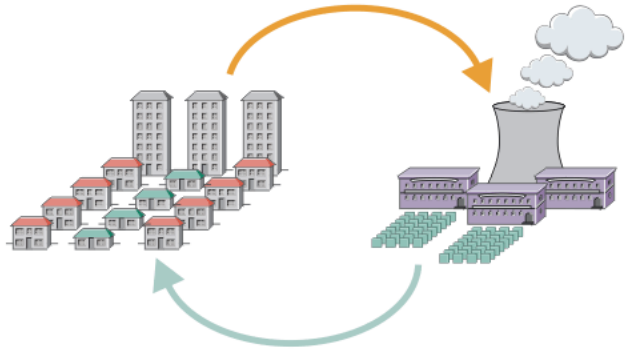
註: MQ Telemetry 取代在 WebSphere Message Broker 第 7 版中撤銷的 SCADA 節點 (現在稱為 IBM Integration Bus) 並在 Windows、 Linux 和 AIX 上執行。

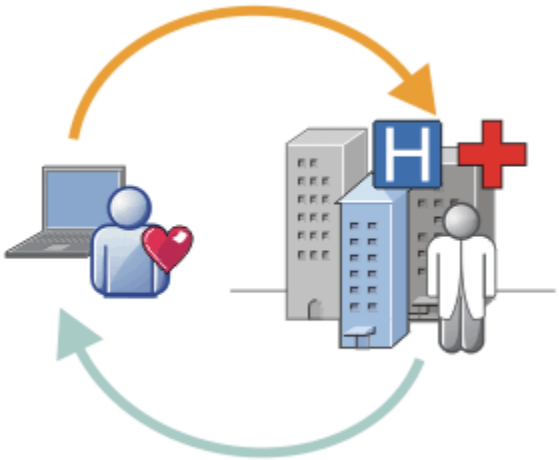
Windows Linux AIX 遙測使用案例

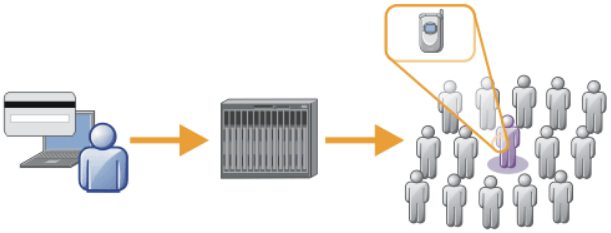
遙測可用於自動感應、測量資料及控制遠端裝置。 主要用於將資料從裝置傳輸至中央控制點。 遙測還包括將配置及控制資訊傳送至裝置。

MQ Telemetry 會使用 MQTT protocol 來連接小型裝置，並使用 IBM MQ 將裝置連接至其他應用程式。 MQ Telemetry 會橋接裝置與網際網路之間的間隙，讓您更容易建置「智慧型解決方案」。 智慧型解決方案會針對監視及控制裝置的應用程式，解除鎖定網際網路上及企業應用程式中可用的豐富資訊。

下列圖表示範 MQ Telemetry 的一些一般用法:

遙測: 智慧型電力	
	• MQTT 訊息，包含傳送至服務提供者的能源使用情形資料。 • MQ Telemetry 會根據能源使用資料的分析來傳送 CONTROL COMMANDS。 • 如需相關資訊，請參閱下列使用案例: 第 93 頁的『遙測使用案例: 家庭能源監視及控制』

遙測: 智慧型健康服務	
• MQ Telemetry 會將「健康資料」傳送至您的醫院與醫生。 • MQTT 訊息警示或意見可以根據「性能資料」的分析來傳送。 • 如需相關資訊，請參閱下列使用案例: 第 92 頁的『遙測使用案例: 家庭病患監視』	

遙測: 一個人在人群中	
	• 簡式卡片交易會傳送至銀行的伺服器。 • MQ Telemetry 可識別數千中的一個人員，並警示客戶已使用其卡。 • MQ Telemetry 可以使用最簡單的資訊輸入，並尋找該個人。

子主題中說明的使用案例取自實際範例。 它們說明使用遙測的一些方法，以及遙測技術必須解決的一些一般問題。

Windows Linux AIX 遙測使用案例: 家庭病患監視

在 IBM 與心臟病患者護理系統上的醫療保健提供者之間的協同作業中，植入的心臟過去纖顫器會與醫院進行通訊。 使用 RF 遙測將病患及植入裝置的相關資料傳送至病患家中的 MQTT 裝置。

通常是每晚傳送至位於床端的轉送器。 轉送器會透過電話系統安全地將資料傳送至醫院，並在其中分析資料。

該系統減少了病人必須向醫生就診的次數。 它會偵測病患或裝置何時需要注意，並在發生緊急狀況時警示待命醫師。

IBM 與醫療保健提供者之間的協同作業具有許多遙測使用案例常見的性質:

Invisibility

裝置不需要使用者介入，只需要提供電源、電話線，以及在一天中的部分時間接近裝置。 本實用新型結構簡單，操作可靠，使用方便。

為了消除患者設定裝置的需求，裝置供應商預先配置裝置。 病人只能把它插進去 患者消除配置簡化了裝置的操作，減少了裝置錯誤配置的機會。

MQTT 用戶端內嵌為裝置的一部分。 裝置開發人員會在裝置中內嵌 MQTT 用戶端實作，且開發人員或供應商會將 MQTT 用戶端配置為預先配置的一部分。

MQTT 用戶端以 Java SE JAR 檔的形式出貨，開發人員將這個檔案包含在其 Java 應用程式中。 對於非 Java 環境 (例如，此環境)，裝置開發人員可以使用已發佈的 MQTT 格式及通訊協定來實作不同語言的用戶端。 另外，開發人員也可以使用 Windows、 Linux 和 ARM 平台所提供的其中一個 C 用戶端作為共用程式庫。

連線不平均

去纖顫器與醫院之間的通訊具有不平均的網路特性。 採用兩種不同的網路，解決了從病人採集資料，並將資料傳送到醫院的不同問題。 在專利與 MQTT 裝置之間，使用短程低功率 RF 網路。 轉送器會透過低頻寬電話線，使用 VPN TCP/IP 連線來連接至醫院。

通常無法找到將每個裝置直接連接至 Internet Protocol 網路的方法。 使用由中心連接的兩個網路是一般解決方案。 MQTT 裝置是一個簡單的集線器，儲存來自病患的資訊，並將它轉遞至醫院。

安全

醫生必須能夠信任病患資料的真實性，且病患想要尊重其資料的隱私。

在某些情況下，使用 VPN 或 TLS 來加密連線已足夠。 在其他情況下，即使在儲存資料之後，也需要保持資料安全。

有時遙測裝置並不安全。 例如，它可能位於共用住宅中。 必須鑑別裝置的使用者，以確定資料來自正確的病患。 裝置本身可以使用 TLS 向伺服器鑑別，而伺服器則由裝置鑑別。

裝置與佇列管理程式之間的遙測通道支援 JAAS 進行使用者鑑別，以及 TLS 進行通訊加密及裝置鑑別。 發佈的存取權由 IBM MQ 中的物件權限管理程式控制。

用來鑑別使用者的 ID 可以對映至不同的 ID，例如一般病患身分。 一般 ID 可簡化在 IBM MQ 中配置發佈主題的授權。

連線功能

MQTT 裝置與醫院之間的連線使用撥號，並使用低到 300 波的頻寬。

為了以 300 Baud 有效運作，除了 TCP/IP 標頭之外， MQTT protocol 只會將幾個額外位元組新增至訊息。

MQTT protocol 提供單一傳輸 發動並忘記 傳訊，這可保持低延遲。 如果保證遞送比回應時間更重要，它也可以使用多重傳輸來保證 至少一次 和 正好一次 遞送。 為了保證遞送，訊息會儲存在裝置中，直到順利遞送為止。 如果裝置以無線方式連線，則保證遞送特別有用。

可擴充性

遙測裝置通常會大量部署，從數萬個到數百萬個。

將許多裝置連接至系統會對解決方案提出大量需求。 有一些商業需求，例如裝置及其軟體的成本，以及管理授權、裝置及使用者的管理需求。 技術需求包括網路及伺服器上的負載。

開啟連線所使用的伺服器資源多於維護開啟連線所使用的伺服器資源。 但在使用電話線的使用案例中，連線費用表示連線保持開啟的時間不會超過需要。 資料傳送基本上是批次性質。 連線可以整晚排程，以避免在睡覺時突然出現連線尖峰。

在用戶端上，用戶端的可調整性是由所需的最小用戶端配置所協助。 MQTT 用戶端內嵌在裝置中。 不需要在將裝置部署給病患時內建配置或 MQTT 用戶端授權接受步驟。

在伺服器上， MQ Telemetry 具有每個佇列管理程式 50,000 個開啟連線的起始目標。

連線是使用 IBM MQ Explorer 來管理。 IBM MQ Explorer 會將要顯示的連線過濾成可管理的數字。 透過適當選擇將 ID 配置給用戶端的方法，您可以根據地理位置或依病患名稱的字母順序來過濾連線。

Windows Linux AIX 遙測使用案例: 家庭能源監視及控制

智慧型計量會收集比傳統計量更多關於能源消耗的詳細資料。

智慧型電表通常與本端遙測網路結合使用，以監視和控制家中的個別電器用品。 一些智慧型電表也會以遠端方式連接，以便在遠方進行監視和控制。

遠端連線可以由個人、電源公用程式或中央控制點設定。 遠端控制點可以讀取電源使用情形並提供使用情形資料。 它可以提供資料來影響使用情形，例如連續定價和天氣資訊。 可以限制負載，提高整體發電效率。

智慧型計量器已開始廣泛部署。 例如，英國政府正在就 2020 年前在英國每個家庭部署智慧儀表進行磋商。

家庭計量使用案例具有許多一般性質:

Invisibility

除非使用者想要使用計量來參與節省能源，否則計量不需要使用者人為介入。 它不能降低個別電器的能源供應的可靠性。

MQTT 用戶端可以內嵌在隨計量器部署的軟體中，且不需要個別安裝或配置。

連線不平均

應用裝置與智慧型計量器之間的通訊需要與計量器與遠端連線點之間不同的連線標準。

從智慧型計量器到應用裝置的連線必須具有高可用性，且符合家庭區域網路的網路標準。

遠端網路可能使用各種實體連線。 其中有些 (例如行動電話) 具有高傳輸成本，且可能是間歇性的。 MQTT v3 規格是針對遠端連線，以及本端配接卡與智慧型計量之間的連線。

電源插座與應用程式與計量器之間的連線使用住家區域網路，例如 Zigbee。 MQTT for sensor network (MQTT-S)，設計為使用 Zigbee 及其他低頻寬網路通訊協定。 MQ Telemetry 不直接支援 MQTT-S。 它需要閘道才能將 MQTT-S 連接至 MQTT v3。

如同家庭病患監視，家庭能源監視和控制的解決方案需要多個網路，使用智慧型計量作為中心來連接。

安全

有許多與智慧型計量相關聯的安全問題。 這些問題包括交易不可否認性、已起始任何控制動作的授權，以及耗電量資料的隱私權。

為了確保隱私權，可以使用 TLS 來加密 MQTT 在計量與遠端控制點之間傳送的資料。 為了確保控制動作的授權，可以使用 TLS 來相互鑑別計量與遠端控制點之間的 MQTT 連線。

連線功能

遠端網路的實體本質可能會有很大的不同。 它可能使用現有的寬頻連線，或使用具有高通話成本及間歇性可用性的行動網路。 對於高成本、間歇性的連線， MQTT 是有效且可靠的通訊協定; 請參閱 第 92 頁的『遙測使用案例: 家庭病患監視』。

可擴充性

最終，電力公司 (或中央控制點) 計劃部署數千萬智慧電錶。 一開始，每個部署的計量數是從數十到數十萬。 此數目與每個佇列管理程式 50,000 個開放式用戶端連線的起始 MQTT 目標相當。

家用能源監視和控制架構的重要方面是使用智慧型計量器作為網路集中器。 每一個應用裝置配接卡都是個別感應器。 透過使用 MQTT 將它們連接至本端中心，該中心可以將資料流集中到具有中央控制點的單一 TCP/IP 階段作業，並在短時間內儲存訊息以克服階段作業中斷。

在家用能源使用案例中，遠端連線必須保持開啟狀態，有兩個原因。 首先，因為相對於傳送要求，開啟連線需要很長的時間。 在短間隔內開啟許多連線以傳送 "load-limitation" 要求的時間太長。 其次，若要接收來自電力公司的負載限制要求，必須先由用戶端開啟連線。 使用 MQTT，連線一律由用戶端起始，若要接收來自電力公司的負載限制要求，連線必須保持開啟。

如果開啟連線的速率是重要的，或伺服器起始時間重要要求，解決方案通常會維護許多開啟的連線。

Windows Linux AIX 遙測使用案例: 射頻識別 (RFID)

RFID 是使用內嵌式 RFID 標籤來無線識別及追蹤物件。 RFID 標籤最多可以讀取數公尺的範圍，且不在 RFID 讀取器的視線範圍內。 被動式標籤由 RFID 讀取器啟動。 作用中標籤在沒有外部啟動的情況下傳輸。 作用中標籤必須具有電源。 被動標籤可以包括電源以增加其範圍。

RFID 在許多應用程式中使用，使用案例的類型會有很大不同。 RFID 使用案例，以及家庭病患監視和家庭能源監視及控制使用案例，有一些相似性和差異。

Invisibility

在許多使用案例中， RFID 讀取器會大量部署，且必須在沒有使用者介入的情況下運作。 讀取器包括內嵌的 MQTT 用戶端，以與中央控制點進行通訊。

例如，在配送倉儲中，讀取器使用移動感應器來偵測棧板。 它會啟動棧板上項目的 RFID 標籤，並將資料及要求傳送至中央應用程式。 資料用來更新庫存的位置。 要求會控制接下來棧板的狀況，例如將它移至特定機槽。 航空公司和機場行李系統都以這種方式使用 RFID。

在部分 RFID 使用案例中，讀取器具有標準運算環境，例如 Java Platform, Micro Edition (Java ME)。 在這些情況下，在製造之後，可能會在不同的配置步驟中部署 MQTT 用戶端。

連線不平均

RFID 讀取器可能與包含 MQTT 用戶端的本端控制裝置分離，或者每一個讀取器都可能內嵌 MQTT 用戶端。 通常，地理或通訊因素會指出拓蹼的選擇。

安全

隱私和真實性是 RFID 標籤附件中的安全問題。 RFID 標籤是不明顯的，可以被祕密監視、欺騙或竄改。

RFID 安全問題解決方案增加部署新 RFID 解決方案的機會。 雖然安全暴露在 RFID 標籤和當地讀取器中，但使用中央資訊處理會建議應對不同威脅的方法。 例如，動態將庫存層次與遞送和分派產生關聯，即可偵測到標籤竄改。

連線功能

RFID 應用程式通常包含批次儲存及轉遞從 RFID 讀取器收集的資訊及立即查詢。 在配送倉儲使用案例中， RFID 讀取器一直連接。 當讀取標籤時，它會連同讀取器的相關資訊一起發佈。 倉儲應用程式會將回應發佈回讀取器。

在倉儲應用程式中，網路通常是可靠的，而立即要求可能會使用 發動並忘記 訊息，以取得低延遲效能。批次儲存及轉遞資料可能會使用 正好一次 傳訊，以將與鬆散資料相關聯的管理成本降到最低。

可擴充性

如果 RFID 應用程式需要立即回應，則在第二個或第二個的順序中， RFID 讀取器必須保持連線。

Windows Linux AIX 遙測使用案例: 環境感應

環境感應使用遙測來收集河流水位及品質、大氣汙染物及其他環境資料的相關資訊。

感應器經常位於遠端位置，無法存取有線通訊。 無線頻寬昂貴，可靠性低。 通常，小地理區域中的許多環境感應器會連接至安全位置中的本端監視裝置。 本端連線可能是有線或無線。

Invisibility

與中央監視裝置相比，感應器裝置可能較不容易存取、供電較低，且部署的數目較多。 感應器有時會「啞巴」，而本端監視裝置包括用來轉換及儲存感應器資料的配接卡。 監視裝置可能會納入支援 Java Platform, Standard Edition (Java SE) 或 Java Platform, Micro Edition (Java ME) 的通用電腦。 配置 MQTT 用戶端時，不可見性不太可能是主要需求。

連線不平均

感應器的功能，以及遠端連線的成本和頻寬，通常會導致本端監視中心連接至中央伺服器。

安全

除非在軍事或防禦性使用案例中使用解決方案，否則安全不是主要需求。

連線功能

許多用途不需要持續監視或立即提供資料。 異常狀況資料 (例如洪水層次警示) 需要立即轉遞。 在本端監視器中聚集感應器資料，以減少連線及通訊成本，然後使用排程連線進行傳送。 一旦在監視器上偵測到異常狀況資料，即會立即轉遞該異常狀況資料。

可擴充性

感應器集中在本端中心周圍，並將感應器資料聚集到根據排程傳輸的封包中。 這兩個因素都會減少使用直接連接感應器所強制的中央伺服器負載。

Windows Linux AIX 遙測使用案例: 行動式應用程式

行動式應用程式是在無線裝置上執行的應用程式。 裝置是一般應用程式平台或自訂裝置。

一般平台包括手持裝置 (如手機及個人資料助理) ，以及行動式裝置 (如筆記型電腦)。 自訂裝置使用針對特定應用程式自訂的特殊用途硬體。 記錄 "signed-for" 包裹遞送的裝置是自訂行動式裝置的範例。 自訂行動式裝置上的應用程式通常建置在一般軟體平台上。

Invisibility

自訂行動式應用程式的部署是受管理的，並且可以包括 MQTT 用戶端應用程式的配置。 配置 MQTT 用戶端時，不可見性不太可能是主要需求。

連線不平均

與前述使用案例的本端中心拓蹼不同，行動式用戶端會從遠端連接。 用戶端應用程式層會直接連接至中央中心的應用程式。

安全

在幾乎沒有實體安全的情況下，必須鑑別行動式裝置及行動式使用者。 TLS 用來確認裝置的身分，JAAS 用來鑑別使用者。

連線功能

如果移動應用依賴於無線覆蓋，它必須能夠離線操作，並且有效地處理中斷的連線。 在此環境中，目標是保持連線，但應用程式必須能夠儲存及轉遞訊息。 通常訊息是訂單或遞送確認，且具有重要商業價值。 需要可靠地儲存並轉遞它們。

可擴充性

可擴充性不是主要問題。 在自訂行動式應用程式使用案例中，應用程式用戶端數目可能不會超過數千 (或數萬)。

Windows Linux AIX 將遙測裝置連接至佇列管理程式

遙測裝置會使用 MQTT v3 用戶端連接至佇列管理程式。 MQTT v3 用戶端使用 TCP/IP 連接至稱為遙測 (MQXR) 服務的 TCP/IP 接聽器。

當您將遙測裝置連接至佇列管理程式時， MQTT 用戶端會使用 `MqttClient.connect` 方法來起始 TCP/IP 連線。 與 IBM MQ 用戶端一樣， MQTT 用戶端必須連接至佇列管理程式，才能傳送及接收訊息。 使用與 MQ Telemetry 一起安裝的 TCP/IP 接聽器 (稱為遙測 (MQXR) 服務) 在伺服器上建立連線。 每一個佇列管理程式最多執行一項遙測 (MQXR) 服務。

遙測 (MQXR) 服務會使用 `MqttClient.connect` 方法中每一個用戶端所設定的遠端 Socket 位址，來配置與遙測通道的連線。 Socket 位址是 TCP/IP 主機名稱與埠號的組合。 遙測 (MQXR) 服務會將多個使用相同遠端 Socket 位址的用戶端連接至相同的遙測通道。

如果伺服器上有多個佇列管理程式，請在佇列管理程式之間分割遙測通道。 在佇列管理程式之間配置遠端 Socket 位址。 使用唯一的遠端 Socket 位址來定義每一個遙測通道。 兩個遙測通道不得使用相同的 Socket 位址。

如果針對多個佇列管理程式上的遙測通道配置相同的遠端 Socket 位址，則第一個要連接的遙測通道會獲勝。 在相同位址上連接的後續通道失敗。

如果伺服器上有多個網路配接卡，請在遙測通道之間分割遠端 Socket 位址。 只要只在一個遙測通道上配置任何特定 Socket 位址， Socket 位址的配置就完全是任意的。

配置 IBM MQ ， 以使用 IBM MQ Explorer 的 MQ Telemetry 補充中提供的精靈來連接 MQTT 用戶端。 或者，遵循 在 Linux 及 AIX 上配置遙測的佇列管理程式 及 在 Windows 上配置遙測的佇列管理程式 中的指示， 以手動配置遙測。

相關參考

MQXR 內容

Windows Linux AIX 遙測連線通訊協定

MQ Telemetry 支援 TCP/IP IPv4 和 IPv6 以及 TLS。

Windows Linux AIX 遙測 (MQXR) 服務

遙測 (MQXR) 服務是一個 TCP/IP 接聽器，作為 IBM MQ 服務進行管理。 使用 IBM MQ Explorer 精靈或 **runmqsc** 指令來建立服務。

MQ Telemetry (MQXR) 服務稱為 SYSTEM.MQXR.SERVICE 。

IBM MQ Explorer 的 MQ Telemetry 函數中所提供的遙測範例配置精靈會建立遙測服務及遙測通道範例; 請參閱 使用 IBM MQ Explorer 來驗證 MQ Telemetry 的安裝 。

從指令行建立配置範例; 請參閱 使用指令行驗證 MQ Telemetry 的安裝。

遙測 (MQXR) 服務會隨佇列管理程式自動啟動及停止。 使用 IBM MQ Explorer 中的 services 資料夾來控制服務。 若要查看服務，您必須按一下圖示，以停止 IBM MQ Explorer 從顯示畫面中濾除 SYSTEM 物件。

如需如何手動建立服務的範例，請參閱

- Linux AIX 在 Linux 上建立 `SYSTEM.MQXR.SERVICE`。
- Windows 在 Windows 上建立 `SYSTEM.MQXR.SERVICE`。

V 9.3.0 從 IBM MQ 9.3.0 開始，會更新 在 Linux 上建立 `SYSTEM.MQXR.SERVICE` 及 在 Windows 上建立 `SYSTEM.MQXR.SERVICE` ， 以指定預設金鑰來要求加密 MQTT TLS 通道的通行詞組。 如需相關資訊，請參閱 MQTT TLS 通道的通行詞組加密。

Windows Linux AIX 遙測通道

建立遙測通道，以建立具有不同內容 (例如 Java 鑑別和授權服務 (JAAS) 或 TLS 鑑別) 的連線，或管理用戶端群組。

使用 **New Telemetry Channel** 精靈建立遙測通道，該精靈在 IBM MQ Explorer 的 MQ Telemetry 函數中提供。 使用精靈來配置通道，以接受來自特定 TCP/IP 埠上 MQTT 用戶端的連線。 從 IBM WebSphere MQ 7.1 開始，您可以使用指令行程式 **runmqsc** 來配置 MQ Telemetry 。

在不同埠上建立多個遙測通道，透過將用戶端分割成多個群組，讓大量用戶端連線更容易管理。 每一個遙測通道都有不同的名稱。

您可以使用不同的安全屬性來配置遙測通道，以建立不同類型的連線。 建立多個通道以接受不同 TCP/IP 位址上的用戶端連線。 使用 TLS 來加密訊息並鑑別遙測通道及用戶端; 請參閱 MQTT 用戶端及遙測通道的 TLS 配置。 指定使用者 ID 以簡化 IBM MQ 物件的授權存取。 指定 JAAS 配置，以利用 JAAS 來鑑別 MQTT 使用者; 請參閱 MQTT 用戶端識別、授權和鑑別。

Windows Linux AIX IBM MQ Telemetry Transport 通訊協定

IBM MQ Telemetry Transport (MQTT) v3 通訊協定設計用於在低頻寬或昂貴連線的小型裝置之間交換訊息，並可靠地傳送訊息。 它使用 TCP/IP。

MQTT protocol 已發佈; 請參閱 IBM MQ Telemetry Transport 格式及通訊協定。 第 3 版通訊協定使用發佈/訂閱，並支援三種服務品質: 發動並忘記、 至少一次及 正好一次。

較小的通訊協定標頭及位元組陣列訊息有效負載會保留較小的訊息。 標頭包含 2 個位元組的固定標頭，以及最多 12 個位元組的其他變數標頭。 通訊協定使用 12 位元組變數標頭來訂閱及連接，而大部分發佈只使用 2 位元組變數標頭。

有了三種服務品質，您可以在低延遲與可靠性之間進行交易; 請參閱 MQTT 用戶端所提供的服務品質。 隨發即忘 不會使用持續性裝置儲存體，只會使用一次傳輸來傳送或接收發佈。 至少一次，且 正好一次 需要裝置上的持續性儲存體，以維護通訊協定狀態並儲存訊息，直到確認為止。

Windows Linux AIX MQTT 用戶端

MQTT 用戶端應用程式負責從遙測裝置收集資訊、連接至伺服器，以及將資訊發佈至伺服器。 它也可以訂閱主題、接收發佈資訊，以及控制遙測裝置。

與 IBM MQ 用戶端應用程式不同， MQTT 用戶端應用程式不是 IBM MQ 應用程式。 它們未指定要連接的佇列管理程式。 它們不限於使用特定的 IBM MQ 程式設計介面。 相反地， MQTT 用戶端會實作 MQTT 3 通訊協定。 您可以撰寫自己的用戶端程式庫，以您選擇的程式設計語言及平台來連接至 MQTT protocol 。 請參閱 IBM MQ Telemetry Transport 格式及通訊協定。

若要簡化撰寫 MQTT 用戶端應用程式，請使用針對許多平台封裝 MQTT protocol 的 C、 Java 及 JavaScript 用戶端程式庫。 如果您在 MQTT 應用程式中納入這些程式庫，則功能完整的 MQTT 用戶端可以短到 15 行程式碼。 MQTT 用戶端程式庫可從 Eclipse Paho 和 MQTT.org 免費取得。 請參閱 IBM MQ Telemetry Transport 程式範例。

MQTT 用戶端應用程式一律負責起始與遙測通道的連線。 連接之後， MQTT 用戶端應用程式或 IBM MQ 應用程式可以開始交換訊息。

MQTT 用戶端應用程式及 IBM MQ 應用程式會發佈及訂閱同一組主題。 IBM MQ 應用程式也可以直接將訊息傳送至 MQTT 用戶端應用程式，而不需要用戶端應用程式先建立訂閱。 請參閱 配置分散式佇列以將訊息傳送至 MQTT 用戶端。

MQTT 用戶端應用程式使用遙測通道連接至 IBM MQ 。 遙測通道充當 MQTT 與 IBM MQ 所使用的不同訊息類型之間的橋接器。 它會代表 MQTT 用戶端應用程式在佇列管理程式中建立發佈及訂閱。 遙測通道會將符合 MQTT 用戶端應用程式訂閱的發佈從佇列管理程式傳送至 MQTT 用戶端應用程式。

Windows Linux AIX 傳送訊息至 MQTT 用戶端

IBM MQ 應用程式可以透過發佈至用戶端建立的訂閱，或直接傳送訊息，來傳送 MQTT v3 用戶端訊息。 MQTT 用戶端可以透過發佈至其他用戶端訂閱的主題，彼此傳送訊息。

MQTT 用戶端會訂閱它從 IBM MQ 接收的發佈資訊

執行作業 第 99 頁的『從 IBM MQ Explorer 發佈訊息至 MQTT 用戶端公用程式』，以將發佈從 IBM MQ 傳送至 MQTT 用戶端。

MQTT v3 用戶端接收訊息的標準方式是建立主題或主題集的訂閱。 在程式碼 Snippet 範例 第 98 頁的圖 44 中， MQTT 用戶端會使用主題字串 "MQTT Examples"來訂閱。 IBM MQ C 應用程式 第 99 頁的圖 45 會使用主題字串 "MQTT Examples"發佈至主題。 在程式碼 Snippet 第 99 頁的圖 46 中， MQTT 用戶端會在回呼方法 messageArrived 中接收發佈。

如需如何配置 IBM MQ 來傳送發佈以回應來自 MQTT 用戶端的訂閱的進一步資訊，請參閱 發佈訊息以回應 MQTT 用戶端訂閱。

IBM MQ 應用程式將訊息直接傳送至 MQTT 用戶端

執行 第 103 頁的『使用 IBM MQ Explorer 將訊息傳送至 MQTT 用戶端』 作業，將訊息直接從 IBM MQ 傳送至 MQTT 用戶端。

以此方式傳送至 MQTT 用戶端的訊息稱為自發訊息。 MQTT v3 用戶端會將自發的訊息接收為具有主題名稱集的發佈。 遙測 (MQXR) 服務會將主題名稱設為遠端佇列名稱。

如需如何配置 IBM MQ 以將訊息直接傳送至 MQTT 用戶端的進一步相關資訊，請參閱 將訊息直接傳送至用戶端。

MQTT 用戶端發佈訊息

MQTT v3 用戶端可以發佈由另一個 MQTT v3 用戶端接收的訊息，但它無法傳送自發的訊息。 程式碼 Snippet 第 99 頁的圖 47 顯示以 Java 撰寫的 MQTT v3 用戶端如何發佈訊息。

將訊息傳送至某個特定 MQTT v3 用戶端的一般型樣，是針對每一個用戶端建立其專屬 ClientIdentifier 的訂閱。 執行作業 第 104 頁的『將訊息發佈至特定的 MQTT v3 用戶端』，以使用 ClientIdentifier 作為主題字串，將訊息從某個 MQTT 用戶端發佈至另一個 MQTT 用戶端。

程式碼 Snippet 範例

第 98 頁的圖 44 中的程式碼 Snippet 顯示在 Java 中撰寫的 MQTT 用戶端如何建立訂閱。 它還需要回呼方法 messageArrived 才能接收訂閱的發佈。

```
String    clientId = String.format("%-23.23s",
                     System.getProperty("user.name") + "_" +
                     (UUID.randomUUID().toString())).trim()).replace('-', '_');
MqttClient  client = new MqttClient("localhost", clientId);
String topicString = "MQTT Examples";
int            QoS = 1;
client.subscribe(topicString, QoS);
```

圖 *44: MQTT v3* 用戶端訂閱者

第 99 頁的圖 45 中的程式碼 Snippet 顯示以 C 撰寫的 IBM MQ 應用程式如何傳送發佈。 程式碼 Snippet 擷取自作業 Create a publisher to a variable topic

```
/* Define and set variables to.defaults */
/* Omitted lines declaring variables    */
char * topicName   = ""
char * topicString = "MQTT Examples"
char * publication = "Hello world!";
do {
  MQCONN(qMgrName, &Hconn, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  td.ObjectType = MQOT_TOPIC;    /* Object is a topic             */
  td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4     */
  strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
  td.ObjectString.VSPtr = topicString;
  td.ObjectString.VSLength = (MQLONG)strlen(topicString);
  MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
  MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  MQDISC(&Hconn, &CompCode, &Reason);
} while (0);
```

圖 *45: IBM MQ* 發佈者 *(publisher)*

當發佈到達時， MQTT 用戶端會呼叫 MQTT 應用程式用戶端 `MqttCallback` 類別的 `messageArrived` 方法。

```
public class CallBack implements MqttCallback {
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
// ... Other callback methods
}
```

圖 *46: messageArrived* 方法

第 99 頁的圖 47 顯示 MQTT v3 將訊息發佈至 第 98 頁的圖 44 中所建立的訂閱。

```
      String          address = "localhost";
      String         clientId = String.format("%-23.23s",
                                System.getProperty("user.name") + "_" +
                                (UUID.randomUUID().toString())).trim()).replace('-', '_');
      MqttClient       client = new MqttClient(address, clientId);
      String      topicString = "MQTT Examples";
      MqttTopic         topic = client.getTopic(Example.topicString);
      String     publication  = "Hello world";
      MqttMessage     message = new MqttMessage(publication.getBytes());
      MqttDeliveryToken token = topic.publish(message);
```

圖 *47: MQTT v3* 用戶端發佈者

Windows Linux AIX 從 IBM MQ Explorer 發佈訊息至 MQTT 用戶端公用程式

請遵循此作業中的步驟，使用 IBM MQ Explorer 來發佈訊息，並使用 MQTT 用戶端公用程式來訂閱訊息。還有一項作業顯示如何配置佇列管理程式別名，而不是將預設傳輸佇列設為 `SYSTEM.MQTT.TRANSMIT.QUEUE`。

開始之前

這項作業假設您熟悉 IBM MQ 和 IBM MQ Explorer，且已安裝 IBM MQ 和 MQ Telemetry 特性。

為這項作業建立佇列管理程式資源的使用者必須具備足夠的權限才能這麼做。 為了示範，會假設 IBM MQ Explorer 使用者 ID 是 mqm 群組的成員。

關於這項作業

在作業中，您在 IBM MQ 中建立主題，並使用 MQTT 用戶端公用程式來訂閱主題。 當您使用 IBM MQ Explorer 發佈至主題時， MQTT 用戶端會接收發佈。

程序

執行下列其中一個作業：

- 您已安裝 MQ Telemetry，但尚未啟動它。 執行作業: 第 100 頁的『在尚未定義遙測 (MQXR) 服務的情況下啟動作業』。
- 您之前已執行 IBM MQ 遙測，但想要使用新的佇列管理程式來執行示範。 執行作業: 第 100 頁的『在尚未定義遙測 (MQXR) 服務的情況下啟動作業』。
- 您想要使用未定義遙測資源的現有佇列管理程式來執行作業。 您不想執行「**定義配置範例**」精靈。
  a. 請執行下列其中一項作業來設定遙測:
     – 在 Linux 和 AIX 上配置遙測的佇列管理程式
     – 在 Windows 上配置遙測的佇列管理程式
  b. 執行作業: 第 101 頁的『使用執行中遙測 (MQXR) 服務啟動作業』
- 如果您要使用已定義遙測資源的現有佇列管理程式來執行作業，請執行下列作業: 第 101 頁的『使用執行中遙測 (MQXR) 服務啟動作業』。

下一步

執行 第 103 頁的『使用 IBM MQ Explorer 將訊息傳送至 MQTT 用戶端』，將訊息直接傳送至用戶端公用程式。

在尚未定義遙測 *(MQXR)* 服務的情況下啟動作業

建立佇列管理程式並執行 **定義配置範例**，以定義佇列管理程式的遙測資源範例。 使用 IBM MQ Explorer 發佈訊息，並使用 MQTT 用戶端公用程式訂閱該訊息。

關於這項作業

當您使用 **定義配置範例**來設定遙測資源範例時，精靈會設定訪客使用者 ID 許可權。 請仔細考量您是否要以這種方式授權來賓使用者 ID。 Windows 上的 guest 及 Linux 上的 nobody， 會獲授與發佈及訂閱主題樹狀結構根目錄的許可權，以及將訊息放置到 SYSTEM.MQTT.TRANSMIT.QUEUE 上的許可權。

精靈也會將預設傳輸佇列設為 SYSTEM.MQTT.TRANSMIT.QUEUE，這可能會干擾在現有佇列管理程式上執行的應用程式。 可以配置遙測，但不使用預設傳輸佇列; 請執行下列作業: 第 102 頁的『使用佇列管理程式別名』。 在這項作業中，您將建立佇列管理程式，以避免干擾任何現有的預設傳輸佇列。

程序

1. 使用 IBM MQ Explorer，建立並啟動新的佇列管理程式。
   a) 用滑鼠右鍵按一下 Queue Managers 資料夾> **新建 > 佇列管理程式 ...**。 輸入佇列管理程式名稱> **完成**。

      組成佇列管理程式名稱; 例如， MQTTQMGR。
2. 建立並啟動遙測 (MQXR) 服務，並建立遙測通道範例。
   a) 開啟 Queue Managers*QmgrName*\Telemetry 資料夾。
   b) 按一下 **定義配置範例 ... > 完成**

維持勾選 **啟動 MQTT 用戶端公用程式** 勾選框。

3. 使用 MQTT 用戶端公用程式建立 MQTT Example 的訂閱。
   a) 按一下**連接**。

   用戶端歷程 會記錄 Connected 事件。

   b) 在 **訂閱 \ 主題** 欄位> **訂閱**中鍵入 MQTT Example 。

   用戶端歷程 會記錄 Subscribed 事件。
4. 在 IBM MQ 中建立 MQTTExampleTopic 。
   a) 在「 **MQ 探險家> 新建** > **主題**」中，用滑鼠右鍵按一下 Queue Managers*QmgrName*\Topics 資料夾。
   b) 輸入 MQTTExampleTopic 作為 **名稱** > **下一步**。
   c) 輸入 MQTT Example 作為 **主題字串** > **完成**。
   d) 按一下 **確定** 以關閉確認視窗。
5. 使用 IBM MQ Explorer 將 Hello World! 發佈至主題 MQTT Example 。
   a) 按一下 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 資料夾。
   b) 用滑鼠右鍵按一下 MQTTExampleTopic > **測試發佈 ...**
   c) 在 **訊息資料** 欄位中鍵入 Hello World! > **發佈訊息** > 切換至「 MQTT 用戶端公用程式」視窗。

   用戶端歷程 會記錄 Received 事件。

使用執行中遙測 *(MQXR)* 服務啟動作業

建立遙測通道及主題。 授權使用者使用主題及遙測傳輸佇列。 使用 IBM MQ Explorer 發佈訊息，並使用 MQTT 用戶端公用程式訂閱該訊息。

開始之前

在此版本的作業中，佇列管理程式 *QmgrName* 已定義且在執行中。 遙測 (MQXR) 服務已定義且在執行中。 遙測 (MQXR) 服務可能已手動建立，或透過執行「 **定義配置範例** 」精靈來建立。

關於這項作業

在這項作業中，您將配置現有的佇列管理程式，以將發佈資訊傳送至 MQTT 用戶端公用程式。

作業的步驟 第 101 頁的『1』 會將預設傳輸佇列設為 SYSTEM.MQTT.TRANSMIT.QUEUE，這可能會干擾在現有佇列管理程式上執行的應用程式。 可以配置遙測，但不使用預設傳輸佇列; 請執行下列作業: 第 102 頁的『使用佇列管理程式別名』。

程序

1. 將 SYSTEM.MQTT.TRANSMIT.QUEUE 設為預設傳輸佇列。
   a) 用滑鼠右鍵按一下 Queue Managers*QmgrName* folder > **內容 ...**
   b) 在導覽器中按一下 **通訊** 。
   c) 按一下 **選取 ...** > 選取 SYSTEM.MQTT.TRANSMIT.QUEUE > **確定** > **確定**。
2. 建立遙測通道 MQTTExampleChannel ， 以將 MQTT 用戶端公用程式連接至 IBM MQ，並啟動 MQTT 用戶端公用程式。
   a) 用滑鼠右鍵按一下「 **MQ 探險家> 新建** > **遙測通道 ...**」中的 Queue Managers*QmgrName* \Telemetry\Channels 資料夾。
   b) 在 **通道名稱** 欄位中鍵入 MQTTExampleChannel > **Next** > **Next**。
   c) 將用戶端授權畫面上的 **固定使用者 ID** 變更為要發佈及訂閱 MQTTExample > **下一步**的使用者 ID。
   d) 保持 **啟動用戶端公用程式** 已勾選> **完成**。
3. 使用 MQTT 用戶端公用程式建立 MQTT Example 的訂閱。

a) 按一下**連接**。

用戶端歷程 會記錄 Connected 事件。

b) 在 **訂閱 \ 主題** 欄位> **訂閱**中鍵入 MQTT Example 。

用戶端歷程 會記錄 Subscribed 事件。

4. 在 IBM MQ 中建立 MQTTExampleTopic 。
   a) 在「**MQ 探險家> 新建 > 主題**」中，用滑鼠右鍵按一下 Queue Managers*QmgrName*\Topics 資料夾。
   b) 輸入 MQTTExampleTopic 作為 **名稱 > 下一步**。
   c) 輸入 MQTT Example 作為 **主題字串 > 完成**。
   d) 按一下 **確定** 以關閉確認視窗。
5. 如果您想要使用者 (不在 mqm 群組中) 發佈及訂閱 MQTTExample 主題，請執行下列動作:
   a) 授權使用者發佈及訂閱主題 MQTTExampleTopic:

```
setmqaut -m qMgrName -t topic -n MQTTExampleTopic -p User ID -all +pub +sub
```

   b) 授權使用者將訊息放置到 SYSTEM.MQTT.TRANSMIT.QUEUE:

```
setmqaut -m qMgrName -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p User ID -all +put
```

6. 使用 IBM MQ Explorer 將 Hello World! 發佈至主題 MQTT Example 。
   a) 按一下 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 資料夾。
   b) 用滑鼠右鍵按一下 MQTTExampleTopic > **測試發佈 ...**
   c) 在 **訊息資料** 欄位中鍵入 Hello World! > **發佈訊息** > 切換至「MQTT 用戶端公用程式」視窗。

用戶端歷程 會記錄 Received 事件。

使用佇列管理程式別名

使用 IBM MQ Explorer 將訊息發佈至 MQTT 用戶端公用程式，而不將預設傳輸佇列設為 SYSTEM.MQTT.TRANSMIT.QUEUE。

此作業是前一項作業的延續，並使用佇列管理程式別名來避免將預設傳輸佇列設為 SYSTEM.MQTT.TRANSMIT.QUEUE。

開始之前

完成作業 第 100 頁的『在尚未定義遙測 (MQXR) 服務的情況下啟動作業』 或作業 第 101 頁的『使用執行中遙測 (MQXR) 服務啟動作業』。

關於這項作業

當 MQTT 用戶端建立訂閱時， IBM MQ 會使用 ClientIdentifier 作為遠端佇列管理程式名稱來傳送其回應。 在此作業中，它會使用 ClientIdentifier(MyClient)。

如果沒有稱為 MyClient 的傳輸佇列或佇列管理程式別名，則會將回應放置在預設傳輸佇列上。 透過將預設傳輸佇列設為 SYSTEM.MQTT.TRANSMIT.QUEUE， MQTT 用戶端會取得回應。

您可以使用佇列管理程式別名，避免將預設傳輸佇列設為 SYSTEM.MQTT.TRANSMIT.QUEUE 。 您必須為每一個 ClientIdentifier 設定佇列管理程式別名。 一般而言，用戶端太多，無法實際使用佇列管理程式別名。 通常無法預期 ClientIdentifier ，因此無法以這種方式配置遙測。

不過，在某些情況下，您可能必須將預設傳輸佇列配置成 SYSTEM.MQTT.TRANSMIT.QUEUE 以外的其他項目。 程序 中的步驟會配置佇列管理程式別名，而不是將預設傳輸佇列設為 SYSTEM.MQTT.TRANSMIT.QUEUE。

程序

1. 移除 SYSTEM.MQTT.TRANSMIT.QUEUE 作為預設傳輸佇列。
   a) 用滑鼠右鍵按一下 Queue Managers*QmgrName* folder > **內容 ...**
   b) 在導覽器中按一下 **通訊** 。
   c) 從 **預設傳輸佇列** 欄位> **確定**中移除 SYSTEM.MQTT.TRANSMIT.QUEUE 。
2. 確認您無法再使用 MQTT 用戶端公用程式來建立訂閱:
   a) 按一下**連接**。

      用戶端歷程 會記錄 Connected 事件。
   b) 在 **訂閱 \ 主題** 欄位> **訂閱**中鍵入 MQTT Example 。

   用戶端歷程 會記錄 Subscribe failed 及 Connection lost 事件。
3. 建立 ClientIdentifier 的佇列管理程式別名 MyClient。
   a) 用滑鼠右鍵按一下 Queue Managers*QmgrName*\Queues 資料夾> **新建** > **遠端佇列定義**。
   b) 將定義命名為 MyClient > **Next**。
   c) 在 **遠端佇列管理程式** 欄位中輸入 MyClient 。
   d) 在 **傳輸佇列** 欄位中鍵入 SYSTEM.MQTT.TRANSMIT.QUEUE > **完成**。
4. 重新連接 MQTT 用戶端公用程式。
   a) 檢查 **用戶端 ID** 是否設為 MyClient。
   b) **CONNECT**

   用戶端歷程 會記錄 Connected 事件。
5. 使用 MQTT 用戶端公用程式建立 MQTT Example 的訂閱。
   a) 按一下**連接**。

      用戶端歷程 會記錄 Connected 事件。
   b) 在 **訂閱 \ 主題** 欄位> **訂閱**中鍵入 MQTT Example 。

   用戶端歷程 會記錄 Subscribed 事件。
6. 使用 IBM MQ Explorer 將 Hello World! 發佈至主題 MQTT Example 。
   a) 按一下 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 資料夾。
   b) 用滑鼠右鍵按一下 MQTTExampleTopic > **測試發佈 ...**
   c) 在 **訊息資料** 欄位中鍵入 Hello World! > **發佈訊息** > 切換至「MQTT 用戶端公用程式」視窗。

   用戶端歷程 會記錄 Received 事件。

Windows Linux AIX 使用 IBM MQ Explorer 將訊息傳送至 MQTT 用戶端

使用 IBM MQ Explorer 將訊息放入 IBM MQ 佇列，以將訊息傳送至 MQTT 用戶端公用程式。 此作業顯示如何配置遠端佇列定義，以將訊息直接傳送至 MQTT 用戶端。

開始之前

執行作業 第 99 頁的『從 IBM MQ Explorer 發佈訊息至 MQTT 用戶端公用程式』。 讓 MQTT 用戶端公用程式保持連線。

關於這項作業

此作業示範使用佇列而非發佈至主題，將訊息傳送至 MQTT 用戶端。 您未在用戶端中建立訂閱。 作業的步驟 第 104 頁的『2』 會示範已刪除先前的訂閱。

程序

1. 中斷連線並重新連接 MQTT 用戶端公用程式，以捨棄任何現有的訂閱。

   除非您變更預設值，否則會捨棄訂閱，因為 MQTT 用戶端公用程式會使用全新階段作業來連接; 請參閱 全新階段作業。

   若要讓執行作業更容易，請鍵入您自己的 ClientIdentifier，而不是使用 MQTT 用戶端公用程式所建立的已產生 ClientIdentifier 。

   a) 按一下 **中斷連線** ，以中斷 MQTT 用戶端公用程式與遙測通道的連線。

   用戶端歷程 會記錄 Disconnected 事件

   b) 將 **用戶端 ID** 變更為 MyClient。

   c) 按一下**連接**。

   用戶端歷程 會記錄 Connected 事件

2. 確認 MQTT 用戶端公用程式不再收到 MQTTExampleTopic 的發佈。

   a) 按一下 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 資料夾。

   b) 用滑鼠右鍵按一下 MQTTExampleTopic > **測試發佈 ...**

   c) 在 **訊息資料** 欄位中鍵入 Hello World! > **發佈訊息** > 切換至「MQTT 用戶端公用程式」視窗。

   用戶端歷程中未記錄任何事件。

3. 建立用戶端的遠端佇列定義。

   將 ClientIdentifier MyClient 設為遠端佇列定義中的遠端佇列管理程式名稱。 使用您喜歡的任何名稱作為遠端佇列名稱。 遠端佇列名稱會傳遞至 MQTT 用戶端作為主題名稱。

   a) 用滑鼠右鍵按一下 Queue Managers*QmgrName*\Queues 資料夾> **新建** > **遠端佇列定義**。

   b) 將定義命名為 MyClientRemoteQueue > **Next**。

   c) 在 **遠端佇列** 欄位中鍵入 MQTTExampleQueue 。

   d) 在 **遠端佇列管理程式** 欄位中輸入 MyClient 。

   e) 在 **傳輸佇列** 欄位中鍵入 SYSTEM.MQTT.TRANSMIT.QUEUE > **完成**。

4. 將測試訊息放置在 MyClientRemoteQueue 上。

   a) 用滑鼠右鍵按一下 **MyClientRemoteQueue** > **放置測試訊息 ...**

   b) 在「訊息資料」欄位中鍵入 Hello queue! > **Put message** > **Close**

   用戶端歷程 會記錄 Received 事件。

5. 移除 SYSTEM.MQTT.TRANSMIT.QUEUE 作為預設傳輸佇列。

   a) 用滑鼠右鍵按一下 Queue Managers*QmgrName* folder > **內容 ...**

   b) 在導覽器中按一下 **通訊** 。

   c) 從 **預設傳輸佇列** 欄位> **確定**中移除 SYSTEM.MQTT.TRANSMIT.QUEUE 。

6. 重做步驟 第 104 頁的『4』。

   MyClientRemoteQueue 是明確命名傳輸佇列的遠端佇列定義。 您不需要定義預設傳輸佇列，即可將訊息傳送至 MyClient。

下一步

由於預設傳輸佇列不再設為 SYSTEM.MQTT.TRANSMIT.QUEUE，除非為 ClientIdentifier 定義佇列管理程式別名 MyClient，否則「MQTT 用戶端公用程式」無法建立新的訂閱。 將預設傳輸佇列還原至 SYSTEM.MQTT.TRANSMIT.QUEUE。

Windows Linux AIX 將訊息發佈至特定的 MQTT v3 用戶端

使用 ClientIdentifier 作為主題名稱，並使用 IBM MQ 作為發佈/訂閱分配管理系統，將訊息從某個 MQTT v3 用戶端發佈至另一個用戶端。

開始之前

執行作業 第 99 頁的『從 IBM MQ Explorer 發佈訊息至 MQTT 用戶端公用程式』。 讓 MQTT 用戶端公用程式保持連線。

關於這項作業

此作業示範兩件事:

1. 訂閱某個 MQTT 用戶端中的主題，並從另一個 MQTT 用戶端接收發佈。
2. 使用 `ClientIdentifier` 作為主題字串來設定「點對點」訂閱。

程序

1. 中斷連線並重新連接 MQTT 用戶端公用程式，以捨棄任何現有的訂閱。

   除非您變更預設值，否則會捨棄訂閱，因為 MQTT 用戶端公用程式會使用全新階段作業來連接; 請參閱 全新階段作業。

   若要讓執行作業更容易，請鍵入您自己的 `ClientIdentifier`，而不是使用 MQTT 用戶端公用程式所建立的已產生 `ClientIdentifier` 。

   a) 按一下 **中斷連線** ， 以中斷 MQTT 用戶端公用程式與遙測通道的連線。

   用戶端歷程 會記錄 `Disconnected` 事件

   b) 將 **用戶端 ID** 變更為 `MyClient`。

   c) 按一下**連接**。

   用戶端歷程 會記錄 `Connected` 事件
2. 建立主題 `MyClient` 的訂閱

   `MyClient` 是此用戶端的 `ClientIdentifier` 。

   a) 在 **訂閱 \ 主題** 欄位> **訂閱**中鍵入 `MyClient` 。

   用戶端歷程 會記錄 `Subscribed` 事件。
3. 啟動另一個 MQTT 用戶端公用程式。

   a) 開啟 `Queue Managers\`*`QmgrName`*`\Telemetry\channels` 資料夾。

   b) 用滑鼠右鍵按一下 **PlainText** 通道> **執行 MQTT 用戶端公用程式 ...**

   c) 按一下**連接**。

   用戶端歷程 會記錄 `Connected` 事件
4. 將 `Hello MyClient!` 發佈至主題 `MyClient`。

   a) 從使用 `ClientIdentifier(MyClient)` 執行的 MQTT 用戶端公用程式中複製訂閱主題 `MyClient`。

   b) 將 `MyClient` 貼至每一個 MQTT 用戶端公用程式實例的 **Publication \ Topic** 欄位。

   c) 在 **Publication \ message** 欄位中鍵入 `Hello MyClient!` 。

   d) 在兩個實例中按一下 **發佈** 。

結果

MQTT 用戶端公用程式中具有 `ClientIdentifier MyClient` 的 **用戶端歷程** 會記錄兩個 **已接收** 事件及一個 **已發佈** 事件。 另一個 MQTT 用戶端公用程式實例會記錄一個 **已發佈** 事件。

如果您只看到一個 **已接收** 事件，請檢查下列可能的原因:

1. 佇列管理程式的預設傳輸佇列是否設為 `SYSTEM.MQTT.TRANSMIT.QUEUE` ?
2. 您是否已在執行其他練習時建立佇列管理程式別名或參照 `MyClient` 的遠端佇列定義? 如果您有配置問題，請刪除任何參照 `MyClient` 的資源，例如佇列管理程式別名或傳輸佇列。 中斷用戶端公用程式的連線，停止並重新啟動遙測 (MQXR) 服務。

Windows Linux AIX 從 MQTT 用戶端傳送訊息至 IBM MQ 應用程式

透過訂閱主題， IBM MQ 應用程式可以從 MQTT v3 用戶端接收訊息。 MQTT 用戶端使用遙測通道連接至 IBM MQ ，並透過發佈至相同主題，將訊息傳送至 IBM MQ 應用程式。

執行作業 第 106 頁的『從 MQTT 用戶端將訊息發佈至 IBM MQ』，以瞭解如何將發佈從 MQTT 用戶端傳送至 IBM MQ 中定義的訂閱。

如果主題已叢集化，或使用發佈/訂閱階層進行配送，則訂閱可以位於與 MQTT 用戶端所連接的佇列管理程式不同的佇列管理程式上。

Windows Linux AIX 從 MQTT 用戶端將訊息發佈至 IBM MQ

使用 IBM MQ Explorer 建立主題訂閱，並使用 MQTT 用戶端公用程式發佈至主題。

開始之前

執行作業 第 99 頁的『從 IBM MQ Explorer 發佈訊息至 MQTT 用戶端公用程式』。 讓 MQTT 用戶端公用程式保持連線。

關於這項作業

此作業示範使用 MQTT 用戶端來發佈訊息，以及使用使用 IBM MQ Explorer 所建立的未受管理可延續訂閱來接收發佈。

程序

1. 建立主題字串 `MQTT Example` 的可延續訂閱。

   請執行下列步驟，以使用 IBM MQ Explorer 來建立佇列及訂閱。

   a) 用滑鼠右鍵按一下 IBM MQ Explorer> **新建** > **本端佇列 ...**中的 `Queue Managers\`*`QmgrName`*`\Queues` 資料夾。

   b) 輸入 `MQTTExampleQueue` 作為佇列名稱> **完成**。

   c) 用滑鼠右鍵按一下 IBM MQ Explorer> **新建** > **訂閱 ...**中的 `Queue Managers\`*`QmgrName`*`\Subscriptions` 資料夾。

   d) 輸入 `MQTTExampleSubscription` 作為佇列名稱> **下一步**。

   e) 按一下 **選取 ...** > `MQTTExampleTopic` > **確定**。

   您已在 第 99 頁的『從 IBM MQ Explorer 發佈訊息至 MQTT 用戶端公用程式』的 `MQTTExampleTopic` 步驟 第 101 頁的『4』 中建立主題。

   f) 輸入 `MQTTExampleQueue` 作為目的地名稱> **完成**。

2. 作為選用步驟，在沒有 `mqm` 權限的情況下，將佇列設定為供不同使用者使用。

   如果您要為權限低於 `mqm` 的使用者設定配置，則必須將 `put` 及 `get` 權限授與 `MQTTExampleQueue`。 已在 第 99 頁的『從 IBM MQ Explorer 發佈訊息至 MQTT 用戶端公用程式』中配置對主題及傳輸佇列的存取權。

   a) 授權使用者放置並進入佇列 `MQTTExampleQueue`:

   ```
   setmqaut -m qMgrName -t queue -n MQTTExampleQueue -p User ID -all +put +get
   ```

3. 使用 MQTT 用戶端公用程式將 `Hello IBM MQ!` 發佈至主題 `MQTT Example` 。

   如果您尚未連接 MQTT 用戶端公用程式，請用滑鼠右鍵按一下 **PlainText** 通道> **執行 MQTT 用戶端公用程式 ...** > **連接**。

   a) 在 **Publication \ Topic** 欄位中鍵入 `MQTT Example` 。

   b) 在 **發佈 \ 訊息** 欄位> **發佈**中鍵入 `Hello IBM MQ!` 。

4. 開啟 `Queue Managers\`*`QmgrName`*`\Queues` 資料夾並尋找 `MQTTExampleQueue`。

   現行佇列深度 欄位為 1

5. 用滑鼠右鍵按一下 MQTTExampleQueue > **瀏覽訊息 ...** 並檢查該出版品。

Windows Linux AIX MQTT 發佈/訂閱應用程式

使用主題型發佈/訂閱來撰寫 MQTT 應用程式。

當連接 MQTT 用戶端時，發佈會以任一方向在用戶端與伺服器之間流動。 在用戶端發佈資訊時，會從用戶端傳送發佈。 當訊息發佈至符合用戶端所建立之訂閱的主題時，會在用戶端接收發佈。

IBM MQ 發佈/訂閱分配管理系統會管理 MQTT 用戶端所建立的主題及訂閱。 MQTT 用戶端所建立的主題與 IBM MQ 應用程式所建立的主題共用相同的主題空間。

與 MQTT 用戶端訂閱中的主題字串相符的發佈資訊會放置在 SYSTEM.MQTT.TRANSMIT.QUEUE 上，並將遠端佇列管理程式名稱設為用戶端的 ClientIdentifier。 遙測 (MQXR) 服務會將發佈轉遞至建立訂閱的用戶端。 它使用已設為遠端佇列管理程式名稱的 ClientIdentifier 來識別用戶端。

一般而言， SYSTEM.MQTT.TRANSMIT.QUEUE 必須定義為預設傳輸佇列。 可以配置 MQTT 不使用預設傳輸佇列; 請參閱 配置分散式佇列以將訊息傳送至 MQTT 用戶端。

MQTT 用戶端可以建立持續性階段作業; 請參閱 第 109 頁的『MQTT 無狀態及有狀態階段作業』。 在持續性階段作業中建立的訂閱是可延續的。 到達具有持續性階段作業之用戶端的發佈會儲存在 SYSTEM.MQTT.TRANSMIT.QUEUE 中，並在重新連接時轉遞至用戶端。

MQTT 用戶端也可以發佈及訂閱保留的發佈; 請參閱 保留的發佈及 MQTT 用戶端。 保留發佈主題的訂閱者會接收主題的最新發佈。 當訂閱者建立訂閱時，或當它重新連接至先前的階段作業時，會收到保留的發佈。

Windows Linux AIX 遙測應用程式

使用 IBM MQ 或 IBM Integration Bus 訊息流程撰寫遙測應用程式。

使用 JMS、MQI 或其他 IBM MQ 程式設計介面，在 IBM MQ 中對遙測應用程式進行程式設計。

遙測 (MQXR) 服務會在 MQTT v3 訊息與 IBM MQ 訊息之間進行轉換。 它會代表 MQTT 用戶端建立訂閱及發佈，並將發佈轉遞至 MQTT 用戶端。 發佈是 MQTT v3 訊息的有效負載。 有效負載包含 jms-bytes 格式的訊息標頭和位元組陣列。 遙測伺服器會在 MQTT v3 訊息與 IBM MQ 訊息之間對映標頭; 請參閱 第 107 頁的『MQ Telemetry 與佇列管理程式的整合』。

使用 Publication、MQInput 和 JMSInput 節點，在 IBM Integration Bus 與 MQTT 用戶端之間傳送及接收發佈。

使用訊息流程，您可以使用 HTTP 來整合遙測與網站，以及使用 IBM MQ 和 WebSphere Adapters 來整合其他應用程式。

Windows Linux AIX MQ Telemetry 與佇列管理程式的整合

MQTT 用戶端與 IBM MQ 作為發佈/訂閱應用程式整合。 它可以發佈或訂閱 IBM MQ 中的主題、建立新主題或使用現有主題。 它會因為 MQTT 用戶端 (包括其本身，或其他發佈至其訂閱主題的 IBM MQ 應用程式) 而接收來自 IBM MQ 的發佈。 套用規則以決定發佈的屬性。

不支援與 IBM MQ 提供的主題、發佈、訂閱及訊息相關聯的許多屬性。 第 108 頁的『MQTT 用戶端至 IBM MQ 發佈/訂閱分配管理系統』 和 第 109 頁的『IBM MQ 至 MQTT 用戶端』 說明如何設定發佈的屬性。 這些設定取決於發佈是否要進入或來自 IBM MQ 發佈/訂閱分配管理系統。

在 IBM MQ 中，發佈/訂閱主題與管理主題物件相關聯。 MQTT 用戶端所建立的主題沒有不同。 當 MQTT 用戶端建立發佈的主題字串時， IBM MQ 發佈/訂閱分配管理系統會將它與管理主題物件相關聯。 分配管理系統會將發佈資訊中的主題字串對映至最接近的管理主題物件母項。 對映與 IBM MQ 應用程式的對映相同。 如果沒有使用者建立的主題，則發佈主題會對映至 SYSTEM.BASE.TOPIC。 套用至發佈資訊的屬性衍生自主題物件。

當 IBM MQ 應用程式或管理者建立訂閱時，會命名訂閱。 使用 IBM MQ Explorer 或使用 **runmqsc** 或 PCF 指令來列出訂閱。 所有 MQTT 用戶端訂閱皆已命名。 他們會取得下列格式的名稱:
ClientIdentifier:Topic name

MQTT 用戶端至 IBM MQ 發佈/訂閱分配管理系統

MQTT 用戶端已傳送發佈至 IBM MQ。 遙測 (MQXR) 服務會將發佈資訊轉換為 IBM MQ 訊息。 IBM MQ 訊息包含三個部分:

1. MQMD
2. RFH2
3. 訊息

MQMD 內容會設為其預設值，除非在 第 108 頁的表 9 中註明。

表 *9: MQMD* 內容的設定

MQMD 欄位	**類型**	**值**
Format	MQCHAR8	MQFMT_RF_HEADER_2
UserIdentifier	MQCHAR12	設為下列其中一項: MqttClient.ClientIdentifier MqttConnectOptions.UserName 由 IBM MQ 管理者為遙測通道設定的使用者 ID。
Priority	MQLONG	MQPRI_PRIORITY_AS_Q_DEF (IBM MQ 的預設值，不同於預設值為 4 的 JMS 。)
Persistence	MQLONG	QoS=0→MQPER_NOT_PERSISTENT QoS=1→MQPER_PERSISTENT QoS=2→MQPER_PERSISTENT

RFH2 標頭不包含 <msd> 資料夾來定義 JMS 訊息的類型。 遙測 (MQXR) 服務會將 IBM MQ 訊息建立為預設 JMS 訊息。 預設 JMS 訊息類型是 jms-bytes 訊息。 應用程式可以存取其他標頭資訊作為訊息內容; 請參閱訊息內容。

RFH2 值的設定如 第 108 頁的表 10 所示。 「格式」內容設定在 RFH2 固定標頭中，其他值設定在 RFH2 資料夾中。

表 *10: RFH2* 內容的設定

RFH2 內容	**類型/資料夾**	**標頭**
格式	MQCHAR8	MQFMT_NONE
ClientIdentifier	mqtt/clientId	複製長度為 1...23 位元組的 MqttClient.ClientIdentifier 。
QoS	mqtt/qos	從送入的 MQTT 訊息複製 QoS 。
訊息 ID	mqtt/msgid	如果 QoS 是 1 或 2，請從送入的 MQTT 訊息複製 **訊息 ID** 。
MQIsRetained	mqps/Ret	如果原始 MQTT 發佈資訊隨 RETAIN 內容集一起傳送，且收到訊息作為保留的發佈資訊。
MQTopicString	mqps/Top	MQTT 訊息發佈至其中的主題。

MQTT 發佈資訊中的有效負載會對映至 IBM MQ 訊息的內容:

表 *11: MQTT* 發佈中的有效負載如何對映至 *IBM MQ* 訊息內容		
訊息內容	**類型**	**IBM MQ 訊息的內容**
緩衝期間	MQBYTE *n*	來自送入 MQTT 訊息的位元組副本。 長度可以為零。

IBM MQ 至 MQTT 用戶端

用戶端已訂閱發佈主題。 IBM MQ 應用程式已發佈至主題，導致 IBM MQ 發佈/訂閱分配管理系統將發佈資訊傳送至 MQTT 訂閱者。 或者， IBM MQ 應用程式已直接將自發的訊息傳送至 MQTT 用戶端。 第 109 頁的表 12 說明如何在傳送至 MQTT 用戶端的訊息中設定固定訊息標頭。 IBM MQ 訊息標頭或任何其他標頭中的任何其他資料都會被捨棄。 IBM MQ 訊息中的訊息資料會以 MQTT 訊息中的訊息有效負載來傳送，不會有任何變更。 遙測 (MQXR) 服務會將 MQTT 訊息傳送至 MQTT 用戶端。

表 *12:* 如何在傳送至 *MQTT* 用戶端的 *IBM MQ* 訊息中設定固定訊息標頭		
MQTT 欄位	**類型**	**值**
DUP	布林值	設定是 QoS = 1 或 2，且訊息已在前一個傳輸中傳送至此用戶端，且訊息在一段時間之後仍未確認。
QoS	int	在 IBM MQ 中，從發佈/訂閱分配管理系統送出的發佈資訊中設定 QoS 值的方式取決於送入的發佈資訊。 它取決於送入的發佈資訊是從 MQTT 用戶端傳送，還是從 IBM MQ 應用程式傳送。 **MQTT** 在送入的發佈以及訂閱者所要求的 QoS 中， QoS 的較低值。 **IBM MQ** 衍生自送入發佈的 QoS 較低值: MQPER_NOT_PERSISTENT→QoS=0 MQPER_PERSISTENT→QoS=2 及訂閱者所要求的 QoS 。 如果訊息傳送至沒有訂閱的用戶端，依預設會將 QoS 設為 2。 用戶端可以使用不同的 QoS 來訂閱 DEFAULT.QoS ， 以變更此值。
RETAIN	布林值	如果送入的發佈具有保留的內容集，則設定。

第 109 頁的表 13 說明如何在傳送至 MQTT 用戶端的 MQTT 訊息中設定可變訊息標頭。

表 *13:* 如何在傳送至 *MQTT* 用戶端的 *MQTT* 訊息中設定 *MQTT* 變數標頭內容		
MQTT 欄位	**類型**	**值**
Topic name	字串	用來發佈訊息的主題字串。
Message ID	字串	將發佈資訊放在 SYSTEM.MQTT.TRANSMIT.QUEUE 中時，發佈資訊的 MQMD.MsgId 內容的最後 2 個位元組。
Payload	byte[]	將送入發佈的位元組直接複製到發佈/訂閱分配管理系統。 長度可以為零。

Windows Linux AIX MQTT 無狀態及有狀態階段作業

MQTT 用戶端可以建立與佇列管理程式的有狀態階段作業。 當有狀態 MQTT 用戶端中斷連線時，佇列管理程式會維護用戶端所建立的訂閱及進行中訊息。 用戶端重新連接時，它會解析進行中的訊息。 它會傳送排入佇列以進行遞送的所有訊息，並接收在其斷線時為其訂閱發佈的所有訊息。

當 MQTT 用戶端連接至遙測通道時，它會啟動新的階段作業，或回復舊的階段作業。 新的階段作業沒有尚未確認的未完成訊息，沒有訂閱，也沒有等待遞送的發佈。 當用戶端連接時，它會指定是使用全新階段作業啟動，還是回復現有的階段作業; 請參閱 清除階段作業。

如果用戶端回復現有的階段作業，它會繼續進行，好像連線未中斷一樣。 等待遞送的發佈會傳送至用戶端，且任何尚未確定的訊息傳送都會完成。 當持續性階段作業中的用戶端與遙測 (MQXR) 服務中斷連線時，用戶端建立的任何訂閱都會保留。 訂閱的發佈會在用戶端重新連接時傳送至用戶端。 如果它重新連接而不回復舊的階段作業，遙測 (MQXR) 服務會捨棄發佈。

階段作業狀態資訊由佇列管理程式儲存在 SYSTEM.MQTT.PERSISTENT.STATE 佇列中。

IBM MQ 管理者可以中斷連線並清除階段作業。

Windows Linux AIX 未連接 MQTT 用戶端時

當用戶端未連接時，佇列管理程式可以繼續代表它接收發佈。 當用戶端重新連接時，會將它們轉遞至用戶端。 如果用戶端非預期地中斷連線，則用戶端可以建立佇列管理程式代表用戶端發佈的 "最後留言"。

如果您想要在用戶端非預期地中斷連線時收到通知，則可以登錄最後留言發佈; 請參閱 最後留言發佈。 如果遙測 (MQXR) 服務偵測到用戶端連線中斷，而沒有用戶端要求它，則它會由遙測 (MQXR) 服務傳送。

用戶端可以隨時發佈保留的發佈資訊; 請參閱 保留的發佈資訊及 MQTT 用戶端。 主題的新訂閱可以要求傳送與主題相關聯的任何保留發佈資訊。 如果您將最後留言建立為保留發佈，則可以使用它來監視用戶端的狀態。

例如，用戶端會在連接時發佈保留的發佈資訊，以廣告其可用性。 同時，它會建立保留的最後留言發佈，以宣告其無法使用。 此外，在進行計劃斷線之前，它會將其無效性發佈為保留發佈。 若要瞭解用戶端是否可用，您可以訂閱保留發佈的主題。 您一律會收到三種出版品中的其中一種。

如果用戶端要在斷線時接收發佈的訊息，請將用戶端重新連接至其前一個階段作業; 請參閱 第 109 頁的『MQTT 無狀態及有狀態階段作業』。 其訂閱在刪除之前或用戶端建立全新階段作業之前都處於作用中。

Windows Linux AIX MQTT 用戶端與 IBM MQ 應用程式之間的鬆散耦合

MQTT 用戶端與 IBM MQ 應用程式之間的發佈流程鬆散耦合。 發佈可能源自 MQTT 用戶端或 IBM MQ 應用程式，且沒有設定順序。 發佈者和訂閱者是鬆散耦合的。 它們透過發佈和訂閱間接彼此互動。 您也可以從 IBM MQ 應用程式直接將訊息傳送至 MQTT 用戶端。

MQTT 用戶端和 IBM MQ 應用程式在兩種意義上鬆散耦合:

1. 發佈者和訂閱者可透過發佈與訂閱與主題的關聯來鬆散結合。 發佈者和訂閱者通常不知道發佈或訂閱的其他來源的位址或身分。
2. MQTT 用戶端在個別執行緒上發佈、訂閱、接收發佈及處理程序遞送確認通知。

MQTT 用戶端應用程式不會等到發佈遞送之後。 應用程式會將訊息傳遞至 MQTT 用戶端，然後應用程式會在其自己的執行緒上繼續執行。 遞送記號用來將應用程式與發佈的遞送同步化; 請參閱 遞送記號。

將訊息傳遞至 MQTT 用戶端之後，應用程式可以選擇等待遞送記號。 用戶端可以提供在發佈遞送至 IBM MQ 時所呼叫的回呼方法，而不是等待。 它也可以忽略 delivery-token。

視與訊息相關聯的服務品質而定， delivery-token 會立即傳回回呼方法，或可能在相當長的時間之後。 在用戶端中斷連線並重新連接之後，甚至可能傳回 delivery-token。 如果服務品質是 發動並忘記，則會立即傳回遞送記號。 在另外兩種情況下，只有在用戶端收到發佈已傳送給訂閱者的確認通知時，才會傳回遞送記號。

由於用戶端訂閱而傳送至 MQTT 用戶端的發佈會遞送至 messageArrived 回呼方法。 messageArrived 在與主要應用程式不同的執行緒上執行。

將訊息直接傳送至 MQTT 用戶端

您可以使用兩種方式之一，將訊息傳送至特定的 MQTT 用戶端。

1. IBM MQ 應用程式可以將訊息直接傳送至 MQTT 用戶端，而不需要訂閱; 請參閱 直接傳送訊息至用戶端。

2. 替代方法是使用 ClientIdentifier 命名慣例。 讓所有 MQTT 訂閱者使用其唯一 ClientIdentifier 作為主題來建立訂閱。 發佈至 *ClientIdentifier*。 發佈會傳送至訂閱主題 *ClientIdentifier* 的用戶端。 使用此技術，您可以將發佈傳送至特定 MQTT 訂閱者。

Windows Linux AIX MQ Telemetry 安全性

維護遙測裝置安全可能很重要，因為裝置可能是可攜式，並會在無法對其進行謹慎控制的位置使用。 您可以使用 VPN 來保護從 MQTT 裝置到遙測 (MQXR) 服務的連線安全。 MQ Telemetry 提供兩個其他安全機制: TLS 和 JAAS。

TLS 主要用於加密裝置與遙測通道之間的通訊，並鑑別裝置是否連接至正確的伺服器; 請參閱 使用 TLS 進行遙測通道鑑別。 您也可以使用 TLS 來檢查是否允許用戶端裝置連接至伺服器; 請參閱 MQTT 使用 TLS 進行用戶端鑑別。

JAAS 主要用來檢查裝置的使用者是否被允許使用伺服器應用程式; 請參閱 使用密碼的 MQTT 用戶端鑑別。 JAAS 可以與 LDAP 搭配使用，以使用單一登入目錄來檢查密碼。

TLS 和 JAAS 可以一起使用，以提供雙因素鑑別。 您可以將 TLS 使用的密碼限制為符合 FIPS 標準的密碼。

至少有數萬個使用者，提供個別安全設定檔並不總是可行。 使用設定檔來授權個別使用者存取 IBM MQ 物件也並非一律可行。 而是將使用者分組至類別，以授權發佈及訂閱主題，並將發佈傳送至用戶端。

配置每一個遙測通道，以將用戶端對映至一般用戶端使用者 ID。 針對在特定通道上連接的每個用戶端使用一般使用者 ID; 請參閱 MQTT 用戶端身分及授權。

授權使用者群組不會危及每個個人的鑑別。 每一個個別使用者都可以在用戶端或伺服器上使用其 **使用者名稱** 及 **密碼**進行鑑別，然後在伺服器上使用一般使用者 ID 進行授權。

Windows Linux AIX MQ Telemetry 全球化

MQTT v3 通訊協定中的訊息有效負載編碼為位元組陣列。 一般而言，處理文字的應用程式會在 UTF-8 中建立訊息有效負載。 遙測通道會將訊息有效負載說明為 UTF-8，但不會執行任何字碼頁轉換。 發佈主題字串必須是 UTF-8。

應用程式負責將英文字母資料轉換成正確的字碼頁，並將數值資料轉換成正確的數字編碼。

MQTT Java 用戶端具有方便的 MqttMessage.toString 方法。 此方法會將訊息有效負載視為以本端平台預設字集編碼，通常是 UTF-8。 它會將有效負載轉換為 Java 字串。 Java 具有字串方法 getBytes ，可將字串轉換為使用本端平台預設字集編碼的位元組陣列。 在具有相同預設字集的平台之間，兩個 MQTT Java 程式在訊息有效負載中交換文字，在 UTF-8 中輕鬆且有效率地執行。

如果其中一個平台的預設字集不是 UTF-8，則應用程式必須建立交換訊息的慣例。 例如，發佈者使用 getBytes("UTF8") 方法指定從字串到 UTF-8 的轉換。 為了接收訊息文字，訂閱者假設訊息是以 UTF-8 字集編碼。

遙測 (MQXR) 服務說明將來自 MQTT 用戶端訊息的所有送入發佈編碼為 UTF-8。 它會設定 MQMD.CodedCharSetId 至 UTF-8， RFH2.CodedCharSetId 至 MQCCSI_INHERIT ; 請參閱 第 107 頁的『MQ Telemetry 與佇列管理程式的整合』。 發佈的格式設為 MQFMT_NONE，因此通道或 MQGET 無法執行任何轉換。

Windows Linux AIX MQ Telemetry 的效能和可調整性

當管理大量用戶端並提高 MQ Telemetry 的可調整性時，請考量下列因素。

容量規劃

如需 MQ Telemetry 效能報告的相關資訊，請參閱 MQ 效能文件。

連線

與連線相關的成本包括

- 根據處理器用量和時間來設定連線本身的成本。

- 網路成本。
- 保持連線開啟但不使用它時使用的記憶體。

用戶端保持連線會產生額外負載。 如果連線保持開啟，則 TCP/IP 流程及 MQTT 訊息會使用網路來檢查連線是否仍然存在。 此外，伺服器中針對保持開啟的每一個用戶端連線使用記憶體。

如果您每分鐘傳送多個訊息，請保持連線開啟，以避免起始新連線的成本。 如果您每 10-15 分鐘傳送少於一個訊息，請考量捨棄連線，以避免保持開啟的成本。 您可能想要將 TLS 連線保持開啟但閒置的時間較長，因為設定的成本較高。

此外，請考量用戶端的功能。 如果用戶端上有儲存及轉遞機能，則您可以批次處理訊息，並捨棄傳送批次之間的連線。 不過，如果用戶端已斷線，則用戶端無法從伺服器接收訊息。 因此，應用程式的目的會影響決策。

如果您的系統有一個用戶端傳送許多訊息 (例如檔案傳送)，請勿等待每個訊息的伺服器回應。 相反地，請傳送所有訊息，並在結束時確認已收到所有訊息。 或者，使用 服務品質 (QoS)。

您可以依訊息來改變 QoS，使用 QoS 0 來遞送不重要的訊息，以及使用 QoS 2 來遞送重要訊息。 如果 QoS 為 0，則訊息傳輸量大約是 QoS 2 的兩倍。

命名慣例

如果您要為許多用戶端設計應用程式，請實作有效的命名慣例。 若要將每一個用戶端對映至正確的 `ClientIdentifier`，請使 `ClientIdentifier` 有意義。 良好的命名慣例可讓管理者更容易找出哪些用戶端在執行中。 命名慣例可協助管理者在 IBM MQ Explorer 中過濾長清單的用戶端，並協助判斷問題; 請參閱 用戶端 ID。

產量

主題名稱的長度會影響流經網路的位元組數。 當發佈或訂閱時，訊息中的位元組數可能很重要。 因此，請限制主題名稱中的字元數。 當 MQTT 用戶端訂閱主題 IBM MQ 時，會提供下列格式的名稱:

```
ClientIdentifier: TopicName
```

若要檢視 MQTT 用戶端的所有訂閱，您可以使用 IBM MQ MQSC **DISPLAY** 指令:

```
DISPLAY SUB(' ClientID1:*')
```

在 IBM MQ 中定義資源以供 MQTT 用戶端使用

MQTT 用戶端會連接至 IBM MQ 遠端佇列管理程式。 IBM MQ 應用程式有兩種基本方法可將訊息傳送至 MQTT 用戶端: 將預設傳輸佇列設為 `SYSTEM.MQTT.TRANSMIT.QUEUE`，或使用佇列管理程式別名。 如果有大量 MQTT 用戶端，請定義佇列管理程式的預設傳輸佇列。 使用預設傳輸佇列設定可簡化管理工作; 請參閱 配置分散式佇列以將訊息傳送至 MQTT 用戶端。

透過避免訂閱來改善可調整性。

當 MQTT V3 用戶端訂閱主題時， IBM MQ 中的遙測 (MQXR) 服務會建立訂閱。 訂閱會將用戶端的發佈遞送至 `SYSTEM.MQTT.TRANSMIT.QUEUE`。 每一個發佈的傳輸標頭中的遠端佇列管理程式名稱會設為進行訂閱之 MQTT 用戶端的 `ClientIdentifier`。 如果有許多用戶端，每一個都會建立自己的訂閱，這會導致在整個 IBM MQ 發佈/訂閱叢集或階層中維護許多 Proxy 訂閱。 如需不使用發佈/訂閱，而是改用點對點型解決方案的相關資訊，請參閱 直接傳送訊息至用戶端。

管理大量用戶端

若要支援許多同時連接的用戶端，請設定 JVM 參數 **-Xms** 及 **-Xmx**，以增加可用於遙測 (MQXR) 服務的記憶體。 請遵循下列步驟:

1. 在遙測服務配置目錄中尋找 `java.properties` 檔案; 請參閱 遙測 (MQXR) 服務配置目錄 (在 Windows 上) 或 遙測服務配置目錄 (在 Linux 上)。
2. 遵循檔案中的指示; 1 GB 的資料堆足夠 50,000 個同時連接的用戶端使用。

```
# Heap sizing options - uncomment the following lines to set the heap to 1G
#-Xmx1024m
#-Xms1024m
```

3. 在 `java.properties` 檔案中新增其他指令行引數，以傳遞至執行遙測 (MQXR) 服務的 JVM; 請參閱 將 JVM 參數傳遞至遙測 (MQXR) 服務。

若要增加 Linux 上開啟檔案描述子的數目，請將下列行新增至 `/etc/security/limits.conf/`，然後重新登入。

```
@mqm soft nofile 65000
@mqm hard nofile 65000
```

每一個 Socket 都需要一個檔案描述子。 遙測服務需要一些額外的檔案描述子，因此此數目必須大於所需的開啟 Socket 數目。

佇列管理程式會對每一個不可延續訂閱使用物件控點。 為了支援許多作用中的不可延續訂閱，會增加佇列管理程式中的作用中控點數目上限; 例如:

```
echo ALTER QMGR MAXHANDS(999999999) | runmqsc qMgrName
```

圖 *48:* 變更 *Windows* 上的控點數上限

```
echo "ALTER QMGR MAXHANDS(999999999)" | runmqsc qMgrName
```

圖 *49:* 變更 *Linux* 上的控點數上限

其他注意事項

規劃系統需求時，請考量重新啟動系統所花費的時間長度。 計劃的關閉時間可能會影響佇列中等待處理的訊息數目。 請配置系統，以便在可接受的時間內順利處理訊息。 檢閱磁碟儲存體、記憶體及處理能力。 對於部分用戶端應用程式，可能可以在用戶端重新連接時捨棄訊息。 如果要捨棄訊息，請在用戶端連線參數中設定 `CleanSession`; 請參閱 清除階段作業。 或者，在 MQTT 用戶端中使用最佳效能服務品質 0 來發佈及訂閱; 請參閱 服務品質。 從 IBM MQ 傳送訊息時，請使用 **非持續性** 訊息。 當系統或連線重新啟動時，不會回復具有這些服務品質的訊息。

Windows Linux AIX MQ Telemetry 支援的裝置

MQTT 用戶端可以在一系列裝置上執行，從感應器及掣動器，到手持裝置及車輛系統。

MQTT 用戶端較小，且在受記憶體小及處理能力低限制的裝置上執行。 MQTT protocol 可靠且具有小型標頭，適用於受低頻寬、高成本及間歇性可用性限制的網路。

MQ Telemetry 透過 MQTT 用戶端應用程式與遙測裝置進行通訊。 這些應用程式使用下列資源，所有這些資源都實作 MQTT v3 通訊協定:

- 下列用戶端程式庫:
  - *MQTT client for Java*，用於建置 (例如) Android、OS X Linux 或 Windows 裝置的原生應用程式。 使用此用戶端程式庫的應用程式可以透過 CDC (連接裝置配置) /Foundation、 J2SE (Java Platform Standard Edition) 及 J2EE (Java Platform Enterprise Edition) ， 從最小的 CLDC (連接限制裝置配置) / MIDP (行動式資訊裝置設定檔) 在 Java 的所有變異上執行。 也支援 IBM jclRM 自訂類別庫。 Java ME 平台通常用於小型裝置，例如掣動器、感應器、行動電話及其他內嵌裝置。 Java SE 平台通常安裝在高階內嵌裝置上，例如桌上型電腦和伺服器。

- *MQTT client for C*，用於建置 (例如) iOS、OS X Linux 或 Windows 裝置的原生應用程式。 此用戶端程式庫提供 C 參照實作，以及 Windows 和 Linux 系統的預先建置原生用戶端。 C 參照實作可將 MQTT 移植到廣泛的裝置和平台。 Intel 上的部分 Windows 系統 (包括 Windows 7、 RedHat、 Ubuntu)，以及 ARM 平台上的部分 Linux 系統 (例如 Eurotech Viper)，實作執行 C 用戶端的 Linux 版本，但 IBM 不提供平台的服務支援。 如果您想要聯絡 IBM 支援中心，則必須在受支援平台上重新產生用戶端的問題。
- *MQTT client for Java*，用於建置瀏覽器型 Web 應用程式。

MQTT 用戶端程式庫可從 Eclipse Paho 和 MQTT.org 免費取得。 請參閱 IBM MQ Telemetry Transport 程式範例。

中的安全 IBM MQ

在 IBM MQ 中，有數種提供安全的方法: 授權服務介面; 使用者撰寫或協力廠商通道結束程式; 使用「傳輸層安全 (TLS)」的通道安全、通道鑑別記錄 及訊息安全。

授權服務介面

使用 MQI 呼叫、指令及存取物件的授權是由 **物件權限管理程式** (OAM) 提供，依預設會啟用。 IBM MQ 實體的存取權是透過 IBM MQ 使用者群組和 OAM 來控制。 管理者可以視需要使用指令行介面來授與或撤銷授權。

如需建立授權服務元件的相關資訊，請參閱 在 AIX, Linux, and Windows 系統上設定安全。

使用者撰寫或協力廠商通道結束程式

通道可以使用使用者撰寫或協力廠商通道結束程式。 如需相關資訊，請參閱 傳訊通道的通道結束程式。

使用 TLS 的通道安全

「傳輸層安全 (TLS)」通訊協定提供業界標準通道安全，可防止竊聽、竄改及模擬。

TLS 使用公開金鑰和對稱技術來提供訊息機密性和完整性以及交互鑑別。

如需 IBM MQ 中安全的綜合性檢閱，包括 TLS 的詳細資訊，請參閱 保護。 如需 TLS 的概觀，包括本節中所說明之指令的指標，請參閱 加密安全通訊協定: TLS。

通道鑑別記錄

使用通道鑑別記錄，在通道層次對授與連接系統的存取權進行精確控制。 如需相關資訊，請參閱 通道鑑別記錄。

訊息安全

使用 Advanced Message Security(這是個別安裝並授權的 IBM MQ 元件)，為使用 IBM MQ 傳送及接收的訊息提供加密保護。 請參閱 Advanced Message Security。

相關工作

保護安全

規劃安全需求

IBM MQ.NET 受管理用戶端 TLS 支援

IBM MQ.NET 完全受管理用戶端提供以 Microsoft.NET SSLStreams 套件為基礎的「傳輸層安全 (TLS)」支援。 這與以 IBM Global Security Kit (GSKit)為基礎的其他 IBM MQ 用戶端不同。

您可以開發 IBM MQ.NET 應用程式，以在受管理模式或未受管理模式下執行。

- 在受管理模式中， .NET 應用程式在 .NET CLR (共用語言執行時期) 內運作，而不進行任何跨平台呼叫 (例如呼叫 C MQI)。

- 在未受管理模式中，會對基礎 MQI 作業呼叫 C MQI。 基本上，未受管理模式介面包含 C MQI 頂端的 .NET 封套類別。

受管理 IBM MQ.NET 用戶端使用 Microsoft.NET Framework 程式庫來實作 TLS 安全 Socket 通訊協定。 Microsoft 中的 System.NET.Security.SSLStream 類別用於在 IBM MQ.NET 中實作「安全 (TLS)」。

未受管理的 IBM MQ.NET 用戶端模式已支援基於 C MQI (及 GSKit) 的 TLS 特性。 亦即， TLS 作業由 C MQI 處理。 在此情況下， GSKit 會實作 TLS 安全 Socket 通訊協定。

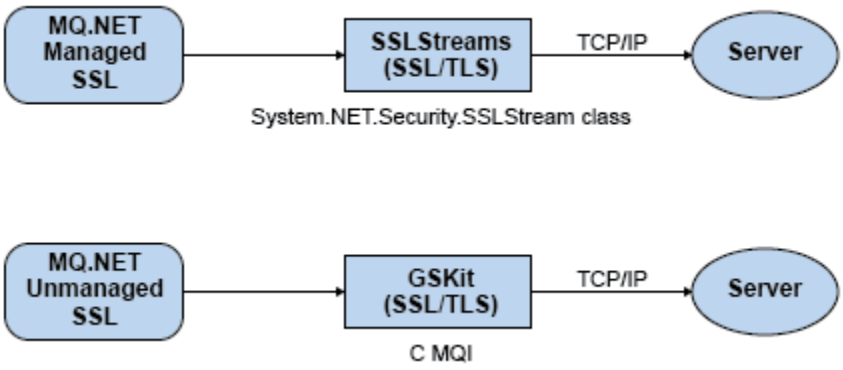


圖 *50: IBM MQ.NET* 受管理與未受管理的 *TLS* 比較

下表彙總受管理實作與未受管理實作之間的差異:

表 *14:* 受管理與未受管理實作之間的差異

Mode	**通訊協定**	**實作**	**註解**
IBM MQ.NET 受管理 SSL	TLS	系統.NET.Security.SSLStream 類別 SSLStream 類別在連接的 TCP Socket 上作為串流運作	TLS 1.0 TLS 1.2 (僅限 Microsoft.NET Framework v4.5)
IBM MQ.NET 未受管理的 SSL	TLS	GSKIT 及 C-MQI	TLS 安全 Socket 通訊協定

相關概念

.NET 的 Secure Sockets Layer (SSL) 和傳輸層安全 (TLS) 支援

IBM MQ MQI clients

IBM MQ MQI client 是 IBM MQ 產品的元件，可以安裝在沒有佇列管理程式執行的系統上。

IBM MQ MQI 用戶端 是一個元件，可讓在系統上執行的應用程式對在另一個系統上執行的佇列管理程式發出 MQI 呼叫。 呼叫的輸出會傳回給用戶端，而用戶端會將它傳回給應用程式。

使用 IBM MQ MQI client，在與用戶端相同的系統上執行的應用程式可以連接至在另一個系統上執行的佇列管理程式。 應用程式可以對該佇列管理程式發出 MQI 呼叫。 這類應用程式稱為 IBM MQ MQI client 應用程式，而佇列管理程式稱為 伺服器佇列管理程式。

IBM MQ 伺服器 是為一個以上用戶端提供佇列作業服務的佇列管理程式。 所有 IBM MQ 物件 (例如佇列) 都只存在於佇列管理程式機器 (IBM MQ 伺服器機器) 上，而不存在於用戶端上。 IBM MQ 伺服器也可以支援本端 IBM MQ 應用程式。

IBM MQ 伺服器與一般佇列管理程式之間的差異是伺服器與每一個用戶端都有專用通訊鏈結。 如需為用戶端及伺服器建立通道的相關資訊，請參閱 配置分散式佇列作業。

IBM MQ MQI client 應用程式和伺服器佇列管理程式會使用 *MQI* 通道來彼此通訊。 當用戶端應用程式發出 **MQCONN** 或 **MQCONNX** 呼叫來連接佇列管理程式時，會啟動 MQI 通道; 當用戶端應用程式發出 **MQDISC** 呼叫來切斷與佇列管理程式的連線時，會結束 MQI 通道。 MQI 呼叫流程在 MQI 通道上的一個方向的輸入參數，以及輸出參數在相反方向的輸入參數。

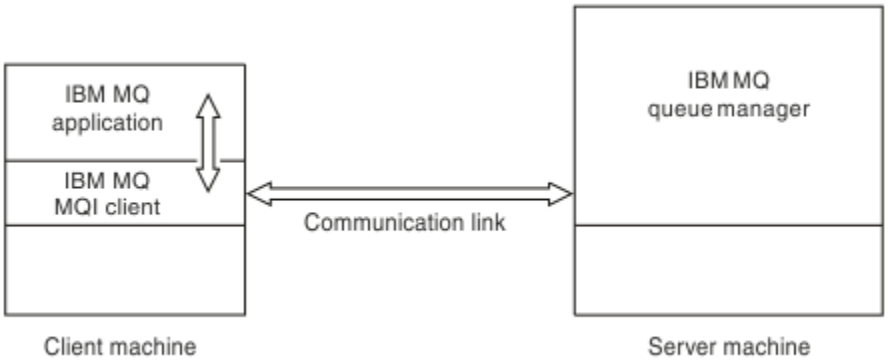


圖 *51:* 用戶端與伺服器之間的鏈結

可以使用下列平台。 這些組合視您使用的 IBM MQ 產品而定，並在 第 117 頁的『IBM MQ 用戶端的平台支援』中說明。

IBM MQ MQI client	**IBM MQ 伺服器**
AIX and Linux Windows IBM i	AIX and Linux Windows IBM i z/OS

MQI 可供在用戶端平台上執行的應用程式使用; 佇列及其他 IBM MQ 物件會保留在您已安裝在伺服器上的佇列管理程式上。

您要在 IBM MQ MQI client 環境中執行的應用程式必須先與相關用戶端程式庫鏈結。 當應用程式發出 MQI 呼叫時，「IBM MQ MQI client」會將要求導向至佇列管理程式，以處理該要求，並從中將回覆傳回 IBM MQ MQI client。

在執行時期會動態建立應用程式與 IBM MQ MQI client 之間的鏈結。

您也可以使用 IBM MQ classes for .NET、 IBM MQ classes for Java 或 IBM MQ classes for Java Message Service (JMS) 來開發用戶端應用程式。 您可以在下列平台上使用 Java 和 JMS 用戶端:

- IBM i
- AIX
- Linux
- Windows

這裡未說明 Java 和 JMS 的用法。 如需如何安裝、配置及使用 IBM MQ classes for Java 和 IBM MQ classes for JMS 的完整詳細資料，請參閱 使用 IBM MQ classes for Java 和 使用 IBM MQ classes for JMS。

主從架構環境中的 IBM MQ 應用程式

鏈結至伺服器時，用戶端 IBM MQ 應用程式可以使用與本端應用程式相同的方式發出大部分 MQI 呼叫。 用戶端應用程式發出 MQCONN 呼叫來連接至指定的佇列管理程式。 然後，此佇列管理程式會處理指定從連接要求傳回之連線控點的任何其他 MQI 呼叫。

您必須將應用程式鏈結至適當的用戶端程式庫。 請參閱 建置 IBM MQ MQI clients 的應用程式。

相關概念

第 117 頁的『為何使用 IBM MQ 用戶端?』
使用 IBM MQ 用戶端是實作 IBM MQ 傳訊及佇列作業的有效方式。

第 118 頁的『何謂延伸交易式用戶端?』
IBM MQ 延伸交易式用戶端可以在外部交易管理程式的控制下，更新另一個資源管理程式所管理的資源。

第 119 頁的『用戶端如何連接至伺服器』
用戶端使用 MQCONN 或 MQCONNX 連接至伺服器，並透過通道進行通訊。

第 120 頁的『交易管理及支援』
交易管理及 IBM MQ 如何支援交易的簡介。

第 121 頁的『延伸佇列管理程式機能』
您可以使用使用者結束程式、API 結束程式或可安裝的服務來延伸佇列管理程式機能。

相關資訊

如何設定 IBM MQ MQI client

為何使用 IBM MQ 用戶端?

使用 IBM MQ 用戶端是實作 IBM MQ 傳訊及佇列作業的有效方式。

您可以讓應用程式使用在一部機器上執行的 MQI，以及在不同機器 (實體或虛擬) 上執行的佇列管理程式。這樣做的好處如下:

- 用戶端機器上不需要完整的 IBM MQ 實作。
- 減少用戶端系統上的硬體需求。
- 減少系統管理需求。
- 在用戶端上執行的 IBM MQ 應用程式可以連接至不同系統上的多個佇列管理程式。
- 可以使用使用不同傳輸通訊協定的替代通道。

IBM MQ 用戶端的平台支援

所有受支援伺服器平台上的 IBM MQ 都接受來自許多平台上 IBM MQ MQI clients 的用戶端連線。

在所有受支援伺服器平台上安裝為 基本產品及伺服器 的 IBM MQ 可以接受來自下列平台上 IBM MQ MQI clients 的連線:

- IBM i
- AIX
- Linux
- Windows

用戶端連線會受到編碼字集 ID (CCSID) 和通訊協定的差異。

哪些應用程式在 IBM MQ MQI client 上執行?

用戶端環境中支援完整 MQI。 這可將 IBM MQ MQI client 上的應用程式鏈結至 MQIC 程式庫，而非 MQI 程式庫，將幾乎任何 IBM MQ 應用程式配置成在 IBM MQ MQI client 系統上執行。 異常狀況如下:

- 具有信號的 MQGET
- 需要與其他資源管理程式同步點協調的應用程式必須使用延伸交易式用戶端

如果啟用先讀，為了改善非持續性傳訊效能，並非所有 MQGET 選項都可用。 此表格顯示容許的選項，以及在 MQGET 呼叫之間是否可以變更這些選項。

表 15: 啟用先讀時允許 *MQGET* 選項			
值	**已啟用先讀且可以在 MQGET 呼叫之間變更時允許**	**已啟用先讀但無法在 MQGET 呼叫之間變更時允許 [1]**	**啟用先讀時不允許的 MQGET 選項 [2]**
MQGET MD 值	MsgId [3] CorrelId [3]	編碼 CodedCharSetId	
MQGET MQGMO 選項	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST [4] MQGMO_BROWSE_NEXT [4] MQGMO_BROWSE_MESSAGE_UNDER_CURSOR [4]	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQ GMO_CONVERT MQ GMO_LOGICAL_ORDER MQ GMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQ GMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQ GMO_同步點 MQGMO_MARK_SKIP_BACKOUT MQ GMO_MSG_UNDER_CURSOR [4] MQGMO_LOCK MQGMO_UNLOCK
MQGMO 值		MsgHandle	

1. 如果在 MQGET 呼叫之間變更了這些選項，則會傳回 MQRC_OPTIONS_CHANGED 原因碼。
2. 如果在第一個 MQGET 呼叫上指定這些選項，則會停用先讀功能。 如果在後續的 MQGET 呼叫上指定了這些選項，則會傳回原因碼 MQRC_OPTIONS_ERROR。
3. 用戶端應用程式需要意識到，如果 MsgId 和 CorrelId 值在 MQGET 呼叫之間變更，則具有先前值的訊息可能已經傳送至用戶端，並保留在用戶端先讀緩衝區中，直到被取用（或自動清除）為止。
4. 第一個 MQGET 呼叫決定在啟用先讀功能時，是否要從佇列中瀏覽或取得訊息。 如果應用程式嘗試同時執行瀏覽與取得動作，將傳回原因碼 MQRC_OPTIONS_CHANGED。
5. MQGMO_MSG_UNDER_CURSOR 不能與先讀功能一起使用。 在啟用先讀功能之後，可瀏覽或取得訊息，但不能同時執行這兩項動作。

在「 IBM MQ MQI client 」上執行的應用程式可以同時連接至多個佇列管理程式，或在 MQCONN 或 MQCONNX 呼叫中使用具有星號 (*) 的佇列管理程式名稱 (請參閱 將 IBM MQ MQI client 應用程式連接至佇列管理程式 中的範例)。

何謂延伸交易式用戶端?

IBM MQ 延伸交易式用戶端可以在外部交易管理程式的控制下，更新另一個資源管理程式所管理的資源。

如果您不熟悉交易管理的概念，請參閱 第 120 頁的『交易管理及支援』。

請注意， XA 交易式用戶端現在提供作為 IBM MQ 的一部分。

用戶端應用程式可以參與由它所連接的佇列管理程式所管理的工作單元。 在工作單元內，用戶端應用程式可以將訊息放置在該佇列管理程式所擁有的佇列中，以及從中取得訊息。 然後用戶端應用程式可以使用 **MQCMIT** 呼叫來確定工作單元，或使用 **MQBACK** 呼叫來取消工作單元。 不過，在相同的工作單元內，用戶端應用程式無法更新另一個資源管理程式的資源，例如 Db2 資料庫的表格。 使用 IBM MQ 延伸交易式用戶端會移除此限制。

IBM MQ 延伸交易式用戶端是具有一些額外功能的 IBM MQ MQI client 。 使用此功能，在相同的工作單元內，用戶端應用程式可以執行下列作業:

- 將訊息放置在它所連接的佇列管理程式所擁有的佇列中，並從中取得訊息
- 更新非 IBM MQ 佇列管理程式的資源管理程式資源

此工作單元必須由與用戶端應用程式在相同系統上執行的外部交易管理程式管理。 用戶端應用程式所連接的佇列管理程式無法管理工作單元。 這表示佇列管理程式只能作為資源管理程式，而不能作為交易管理程式。 它也表示用戶端應用程式只能使用外部交易管理程式所提供的應用程式設計介面 (API) 來確定或取消工作單元。 因此，用戶端應用程式無法使用 MQI 呼叫、 **MQBEGIN**、 **MQCMIT** 及 **MQBACK**。

外部交易管理程式會使用連接至佇列管理程式的用戶端應用程式所使用的相同 MQI 通道，來與作為資源管理程式的佇列管理程式進行通訊。 不過，在失敗之後的回復狀況中，當沒有應用程式正在執行時，交易管理程式可以使用專用 MQI 通道，來回復在失敗時佇列管理程式所參與的任何不完整工作單元。

在本節中，沒有延伸交易式功能的 IBM MQ MQI client 稱為 IBM MQ 基本用戶端。 因此，您可以將 IBM MQ 延伸交易式用戶端視為包含 IBM MQ 基本用戶端及延伸交易式功能。

註: IBM i IBM i 上的 IBM MQ MQI client 不支援 IBM MQ 延伸交易式功能。

延伸交易式用戶端的平台支援

Multi

延伸交易式用戶端適用於支援基本用戶端的所有 Multiplatforms。 用戶端無法用於 z/OS。

使用延伸交易式用戶端的用戶端應用程式只能連接至下列 IBM MQ 9.0 或更新版本產品的佇列管理程式:

- AIX IBM MQ for AIX
- IBM i IBM MQ for IBM i
- Linux IBM MQ for Linux
- Windows IBM MQ for Windows

z/OS 雖然沒有在 z/OS 上執行的延伸交易式用戶端，但使用延伸交易式用戶端的用戶端應用程式可以連接至在 z/OS 上執行的佇列管理程式。

對於每一個平台，延伸交易式用戶端的軟硬體需求與 IBM MQ 基本用戶端的那些需求相同。 如果 IBM MQ 基本用戶端及您使用的交易管理程式支援程式設計語言，則延伸交易式用戶端會支援該語言。

如需所有平台的外部交易管理程式的相關資訊，請參閱 IBM MQ 的系統需求。

用戶端如何連接至伺服器

用戶端使用 MQCONN 或 MQCONNX 連接至伺服器，並透過通道進行通訊。

在 IBM MQ 用戶端環境中執行的應用程式必須維護用戶端與伺服器機器之間的作用中連線。

連線是由發出 MQCONN 或 MQCONNX 呼叫的應用程式所建立。 用戶端及伺服器會透過 *MQI* 通道進行通訊，或在使用共用交談時，每一個都會共用 MQI 通道實例。 當呼叫成功時， MQI 通道實例或交談會維持連接狀態，直到應用程式發出 MQDISC 呼叫為止。 這是應用程式需要連接的每個佇列管理程式的情況。

相同機器上的用戶端和佇列管理程式

當您的機器也已安裝佇列管理程式時，您也可以在 IBM MQ MQI client 環境中執行應用程式。

在此狀況下，您可以選擇鏈結至佇列管理程式庫或用戶端庫，但請記住，如果您鏈結至用戶端庫，仍需要定義通道連線。 在應用程式的開發階段期間，這可能很有用。 您可以在自己的機器上測試您的程式，而不依賴其他程式，並確信當您將它移至獨立 IBM MQ MQI client 環境時，它仍會運作。

不同平台上的用戶端

在此範例中，伺服器機器在不同平台上與三個 IBM MQ MQI clients 進行通訊。

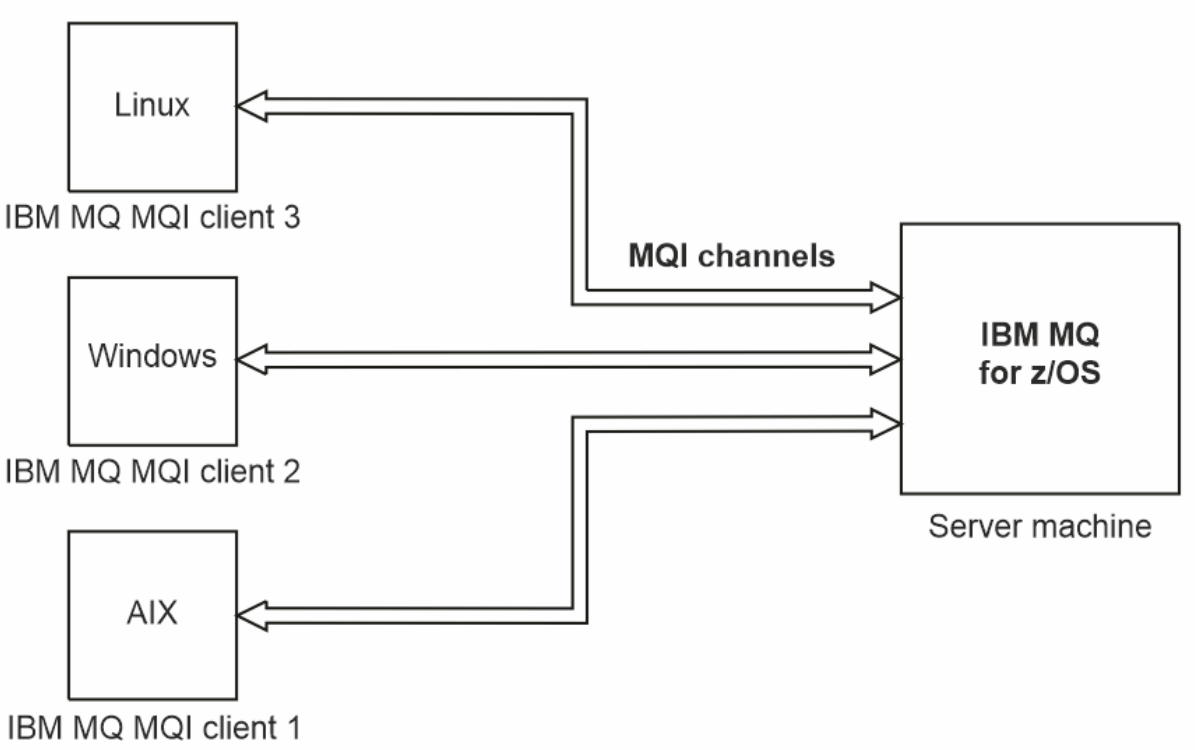


圖 *52: IBM MQ* 伺服器已連接至不同平台上的用戶端

其他更複雜的環境是可能的。 例如， IBM MQ 用戶端可以連接至多個佇列管理程式，或連接為佇列共用群組一部分的任意數目佇列管理程式。

使用不同版本的用戶端及伺服器軟體

如果您使用舊版 IBM MQ 產品，請確定伺服器支援從用戶端的 CCSID 進行程式碼轉換。

IBM MQ 用戶端可以連接至所有支援的佇列管理程式版本。 如果您要連接至舊版佇列管理程式，則無法在用戶端上的 IBM MQ 應用程式中使用較新版本產品的特性及結構。

IBM MQ 佇列管理程式可以透過向下協議至最高相互支援的通訊協定層次，與不同版本的用戶端本身進行通訊。 這表示較舊的用戶端可以與較新的佇列管理程式層次搭配使用。 建議用戶端及伺服器都是目前支援的 IBM MQ 版本，以協助問題診斷並啟用 IBM 的支援。

如需相關資訊，請參閱 開發應用程式中支援的程式設計語言。

交易管理及支援

交易管理及 IBM MQ 如何支援交易的簡介。

資源管理程式 是一個電腦子系統，它擁有並管理可由應用程式存取及更新的資源。 以下是資源管理程式的範例:

- IBM MQ 佇列管理程式，具有作為其佇列的資源
- Db2 資料庫，具有作為其表格的資源

當應用程式更新一或多個資源管理程式的資源時，可能有商業需求可確保某些更新全部以群組方式順利完成，或全部都未完成。 這種需求的原因是如果其中部分更新順利完成，但其他更新未順利完成，則商業資料會保持不一致狀態。

以這種方式管理之資源的更新項目據說會在 工作單元或 交易內發生。 應用程式可以將一組更新項目分組到工作單元。

在工作單元期間，應用程式會向資源管理程式發出要求，以更新其資源。 當應用程式發出要求來確定所有更新項目時，工作單元即會結束。 在確定更新項目之前，其他存取相同資源的應用程式都不會看到這些更新項目。 或者，如果應用程式決定由於任何原因而無法完成工作單元，則它可以發出要求，以回復到該點為止所要求的所有更新項目。 在此情況下，任何更新項目都不會對其他應用程式顯示。 這些更新項目通常在邏輯上相關，且必須全部成功，才能保留資料完整性。 如果一個更新成功，而另一個更新失敗，則資料完整性會遺失。

當工作單元順利完成時，即表示 確定。 一旦確定，在該工作單元內所做的所有更新都會變成永久且無法回復。 不過，如果工作單元失敗，則會改為 取消所有更新。 此處理程序 (其中工作單元已確定或取消且具有完整性) 稱為 同步點協調。

確定或取消工作單元內所有更新項目的時間點稱為 同步點。 工作單元內的更新在 同步點控制內進行。 如果應用程式要求 超出同步點控制的更新，即使有工作單元在進行中，資源管理程式也會立即確定更新，且稍後無法取消更新。

管理工作單元的電腦子系統稱為 交易管理程式或 點協調程式。

本端 工作單元是其中唯一更新的資源是 IBM MQ 佇列管理程式的資源。 在這裡，佇列管理程式本身會使用一段式確定處理程序來提供同步點協調。

廣域 工作單元是其中屬於其他資源管理程式 (例如符合 XA 標準的資料庫) 的資源也會更新的工作單元。 在這裡，必須使用兩階段確定程序，且工作單元可以由佇列管理程式本身或外部另一個符合 XA 標準的交易管理程式 (例如 IBM TXSeries 或 BEA Tuxedo) 來協調。

交易管理程式負責確保工作單元內資源的所有更新順利完成，或全部都不完成。 對於交易管理程式，應用程式會發出要求來確定或取消工作單元。 交易管理程式的範例有 CICS 和 WebSphere Application Server，但兩者也擁有其他功能。

部分資源管理程式提供自己的交易管理功能。 例如， IBM MQ 佇列管理程式可以管理工作單元，包括其自己資源的更新及 Db2 表格的更新。 佇列管理程式不需要個別的交易管理程式來執行此功能，但如果是使用者需求，則可以使用此功能。 如果使用個別交易管理程式，則稱為 外部交易管理程式。

如果要讓外部交易管理程式管理工作單元，交易管理程式與參與工作單元的每一個資源管理程式之間必須有標準介面。 這個介面可讓交易管理程式和資源管理程式彼此通訊。 其中一個介面是 *XA* 介面，它是一些交易管理程式和資源管理程式所支援的標準介面。 XA 介面是由「開放式群組」在 分散式交易處理*: XA* 規格中發佈。

當多個資源管理程式參與工作單元時，即使系統失效，交易管理程式也必須使用 兩段式確定 通訊協定，以確保工作單元內的所有更新都順利完成，或沒有任何更新完成。 當應用程式向交易管理程式發出確定工作單元的要求時，交易管理程式會執行下列動作:

階段 1 (準備確定)
: 交易管理程式會要求參與工作單元的每一個資源管理程式，以確保其資源之預期更新項目的所有相關資訊都處於可回復狀態。 資源管理程式通常透過將資訊寫入日誌並確保將資訊寫入硬碟來執行此動作。 當交易管理程式收到來自每一個資源管理程式的通知，指出其資源的預期更新項目相關資訊處於可回復狀態時，階段 1 即會完成。

階段 2 (確定)
: 當階段 1 完成時，交易管理程式會做出不可撤銷的決策來確定工作單元。 它會要求參與工作單元的每一個資源管理程式確定其資源的更新。 當資源管理程式收到這個要求時，它必須確定更新項目。 在此階段，它無法選擇將它們退出。 當交易管理程式收到來自每一個資源管理程式的通知，表示它已確定其資源的更新項目時，階段 2 即完成。

「XA 介面」使用兩段式確定通訊協定。

如需相關資訊，請參閱 交易式支援情境。

IBM MQ 也提供 Microsoft Transaction Server (COM +) 的支援。 使用 Microsoft Transaction Server (COM +) 提供如何設定 IBM MQ 以利用 COM + 支援的相關資訊。

延伸佇列管理程式機能

您可以使用使用者結束程式、API 結束程式或可安裝的服務來延伸佇列管理程式機能。

使用者結束程式

使用者結束程式提供一種機制，可讓您將您自己的程式碼插入佇列管理程式功能中。 支援的使用者結束程式包括:

通道結束程式
這些結束程式會變更通道運作的方式。 通道結束程式在 傳訊通道的通道結束程式中說明。

資料轉換結束程式
這些結束程式會建立可放入應用程式中的原始碼片段，以將資料從一種格式轉換成另一種格式。 資料轉換結束程式在 寫入資料轉換結束程式中有說明。

叢集工作量結束程式
這個結束程式所執行的功能是由結束程式的提供者所定義。 呼叫定義資訊在 MQ_CLUSTER_WORKLOAD_EXIT-呼叫說明中提供。

API 結束程式

API 結束程式可讓您撰寫程式碼來變更 IBM MQ API 呼叫 (例如 MQPUT 和 MQGET) 的行為，然後在那些呼叫之前或之後立即插入該程式碼。 插入是自動的; 佇列管理程式會在登錄點驅動結束碼。 如需 API 結束程式的相關資訊，請參閱 使用及撰寫 API 結束程式。

可安裝的服務

可安裝服務具有具有多個進入點的正規化介面 (API)。

可安裝服務的實作稱為 服務元件。 您可以使用 IBM MQ 隨附的元件，也可以撰寫您自己的元件來執行您需要的功能。

目前提供下列可安裝服務:

授權服務
授權服務可讓您建置自己的安全機能。

實作服務的預設服務元件是物件權限管理程式 (OAM)。 依預設， OAM 處於作用中，您不需要執行任何動作來配置它。 您可以使用授權服務介面來建立其他元件，以取代或擴增 OAM。 如需 OAM 的相關資訊，請參閱 在 AIX, Linux, and Windows 系統上設定安全。

名稱服務
名稱服務可讓應用程式透過識別遠端佇列來共用佇列，如同它們是本端佇列一樣。

您可以撰寫自己的名稱服務元件。 例如，如果您想要將名稱服務與 IBM MQ 搭配使用，則可能想要執行此動作。 若要使用名稱服務，您必須具有由使用者撰寫或由不同軟體供應商提供的元件。 依預設，名稱服務為非作用中。

相關概念
使用者結束程式、API 結束程式及 IBM MQ 可安裝服務

IBM MQ Java 語言介面

IBM MQ 提供三個在 Java 應用程式中使用的應用程式設計介面 (API): IBM MQ classes for Jakarta Messaging、 IBM MQ classes for JMS 及 IBM MQ classes for Java。

IBM 支援開放式標準，並且是開放式標準中的作用中參與者。

- 從 IBM MQ 8.0 開始，產品會實作 JMS 2.0 標準，其中引進新的簡化 API 以及共用訂閱等特性。
- V 9.3.0 V 9.3.0 從 IBM MQ 9.3.0 開始，也支援 Jakarta Messaging 3.0 。
- 此外， WebSphere Liberty 還支援 JMS 2.0 和 Jakarta Messaging 3.0 搭配 IBM MQ。

在 IBM MQ 內，有三個 API 可在 Java 應用程式中使用:

JM 3.0 V 9.3.0 V 9.3.0 **IBM MQ classes for Jakarta Messaging**

IBM MQ classes for Jakarta Messaging 是一個 Jakarta Messaging 提供者，將 IBM MQ 的 Jakarta Messaging 介面實作為傳訊系統。 Jakarta Connectors Architecture 提供將在 Jakarta EE 環境中執行的應用程式連接至「企業資訊系統 (EIS)」(例如 IBM MQ 或 Db2) 的標準方式。

JMS 2.0 **IBM MQ classes for JMS**

IBM MQ classes for JMS 是一個 JMS 提供者，將 IBM MQ 的 JMS 介面實作為傳訊系統。 Java Platform, Enterprise Edition Connector Architecture (JCA) 提供將在 Java EE 環境中執行的應用程式連接至「企業資訊系統 (EIS)」(例如 IBM MQ 或 Db2) 的標準方式。

IBM MQ classes for Java

IBM MQ classes for Java 可讓您在 Java 環境中使用 IBM MQ 。 IBM MQ classes for Java 可讓 Java 應用程式以 IBM MQ 用戶端身分連接至 IBM MQ ，或直接連接至 IBM MQ 佇列管理程式。

註:

- V 9.3.0 V 9.3.0 JMS 2.0 已由 Jakarta Messaging 取代。 IBM MQ classes for JMS 繼續支援 JMS 2.0 標準，但 Java 傳訊的未來加強功能只會出現在 Jakarta Messaging 中，因此會出現在 IBM MQ classes for Jakarta Messaging 中。 僅建議使用 IBM MQ classes for JMS 來維護及延伸現有的 JMS 2.0 應用程式。 IBM MQ classes for Jakarta Messaging 應該是新開發的偏好技術。
- Stabilized IBM MQ classes for Java 在 IBM MQ 8.0 隨附的層次提供穩定的功能。 將繼續完全支援使用 IBM MQ classes for Java 的現有應用程式，但此 API 已穩定化，因此將不會新增功能，而且加強功能要求會予以拒絕。 完全支援表示問題報告修正將與「IBM MQ 系統需求」變更而引發的任何必要變更一併進行。

JM 3.0 V 9.3.0 V 9.3.0 從 IBM MQ 9.3 開始， IBM MQ classes for Java、 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 是使用 Java 8 來建置。 必須使用這些層次或更高層次的 Java 執行時期環境，才能使用這些介面來執行應用程式。

相關概念

從 Java 存取 IBM MQ -API 的選項

V 9.3.0 V 9.3.0 為何應該將 IBM MQ 類別用於 Jakarta Messaging?

為何應該使用 IBM MQ classes for JMS?

為何應該使用 IBM MQ classes for Java?

IBM MQ classes for JMS/Jakarta Messaging

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 是 IBM MQ 隨附的傳訊提供者。 其中每一個提供者也提供兩組傳訊 API 延伸。 Java Platform, Standard Edition (Java SE) 和 Java Platform, Enterprise Edition (Java EE) 應用程式都可以使用這些傳訊提供者。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 引進 Jakarta Messaging 3.0 的支援。 仍完全支援 JMS 2.0 。

JMS 和 Jakarta Messaging 規格定義一組介面，應用程式可用來執行傳訊作業。 從 IBM MQ 8.0 開始，產品支援 JMS 2.0 版本的 JMS 標準。 此實作提供典型 API 的所有特性，但需要的介面更少，更容易使用。 如需相關資訊，請參閱 第 126 頁的『JMS 和 Jakarta Messaging 模型』 及 Java.net 中的 JMS 2.0 規格。

JM 3.0 從 IBM MQ 9.3.0 開始，也支援 Jakarta Messaging 。

jakarta.jms (Jakarta Messaging 3.0) 或 javax.jms (JMS 2.0) 套件指定傳訊介面的詳細資料，傳訊提供者會針對特定的傳訊產品實作這些介面。 例如:

- IBM MQ classes for JMS 是一個 JMS 提供者，用於實作 IBM MQ 的 JMS 介面，並為 JMS API 提供下列兩組延伸:
  - IBM MQ JMS 延伸
  - IBM JMS 延伸

- 使用 javax.dims 或 jakarta.jms 建立的 Connection Factory、佇列或主題物件，可以利用任何這些 API 來解決介面或任何一組 JMS 延伸; 也就是說，它可以強制轉型成任何介面。 若要將應用程式可攜性維護在最高層次，請使用適合您需求的最通用 API。

因為 JMS 和 Jakarta Messaging 有許多共同之處，本主題中對 JMS 的進一步參照可以視為同時參照兩者。必要時會強調顯示任何差異。

IBM MQ JMS 延伸

IBM MQ classes for JMS 還提供了 JMS API 的延伸。 IBM MQ classes for JMS 包含在 MQConnectionFactory、MQQueue 及 MQTopic 物件中實作的延伸。 這些物件具有特定於 IBM MQ 的內容及方法。 物件可以是受管理物件，或應用程式可以在執行時期動態地建立物件。 這些延伸稱為 IBM MQ JMS 延伸。 請注意，在本文件中，應用程式在執行時期動態建立的物件不會被視為受管理物件。

IBM JMS 延伸

除了 IBM MQ JMS 延伸之外， IBM MQ classes for JMS 還為 JMS API 或 Java 提供一組更通用的延伸，作為使用的程式設計語言。 這些延伸稱為 IBM JMS 延伸，具有下列廣泛目標:

- 在 IBM JMS 提供者之間提供更高層次的一致性。
- 更容易在兩個 IBM 傳訊系統之間撰寫橋接器應用程式。
- 更容易將應用程式從一個 IBM JMS 提供者移轉至另一個提供者。

這些延伸的主要焦點關注在執行時期動態地建立及配置 Connection Factory 和目的地，但這些延伸也提供與傳訊不直接相關的功能，如問題判斷的功能。

相關工作

使用 IBM MQ 類別進行 JMS/Jakarta 傳訊

配置 JMS 和 Jakarta 傳訊資源

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging: 概觀

IBM MQ 9.3.0 引進 Jakarta Messaging 的支援。 對於 Jakarta Messaging 3.0， JMS 規格的控制權已從 Oracle 移至 Java Community Process。 不過， Oracle 會保留其他 Java 技術中使用的 "javax" 名稱的控制權。 因此，雖然 Jakarta Messaging 3.0 在功能上等同於 JMS 2.0，但在命名方面仍有一些差異。 3.0 版的正式名稱是 Jakarta Messaging 而非 Java Message Service，套件和常數名稱會以 `jakarta` 而非 `javax` 作為字首。

背景

多年來， Java 平台有兩種形式- Standard Edition 和 Enterprise Edition。

Java Platform, Standard Edition (有時縮寫為 Java SE) 是核心語言及類別庫，能夠在獨立式環境定義中執行。 Java SE 中大部分 Java 套件的名稱都以 "java." 開頭。

Java Platform, Enterprise Edition (Java EE) 延伸此功能，新增傳訊、各種 Bean、交易方式等之類的功能。其中部分技術也可以在 Java SE 環境定義中使用。 Java EE 中大部分 Java 套件的名稱都以 "javax" 開頭。不過，有些交叉，因此有些 Java SE 套件會有 "javax"。 作為其名稱的字首。

Java Message Service (JMS) 是 Java Platform, Enterprise Edition 的一部分。 Java EE 7 納入 JMS 2.0。

到 Java EE 7 為止，這些技術都由 Oracle 來管理。

Java EE 技術最近已從 Oracle 的管理工作移至 Eclipse Foundation 所監督的社群程序。

作為 "標槍" 無法將名稱移至新專案，已採用新命名-所有套件及內容名稱現在都以 "jakarta" 為字首。 且 Java Platform, Enterprise Edition 在未來將稱為 "Jakarta EE"。 版本編號繼續: 第 8 版是過渡版本，基本上可以忽略，而 Jakarta EE 9 是 "jakarta" 的點。 字首生效。

適用於 IBM MQ 環境定義的主要 Jakarta EE 技術是 Jakarta Messaging 3.0 - Java Message Service (JMS) 2.0 的後置作業。 因此 Jakarta EE 9 納入 Jakarta Messaging 3.0。

IBM MQ 繼續支援 Java EE 7 和 JMS 2.0，同時引進 Jakarta EE 9 和 Jakarta Messaging 3.0 的支援。

交付內容: Java SE

對於 Java Platform, Standard Edition，除了 IBM MQ classes for JMS (使用 IBM MQ 支援 JMS 2.0 作業) IBM MQ 9.3.0 之外，還提供 IBM MQ classes for Jakarta Messaging。 這些類別提供與 IBM MQ 整合的 Jakarta Messaging 3.0 提供者，容許使用 IBM MQ 佇列管理程式來協助 Jakarta Messaging 作業。

這些作為標準 JAR 檔 `com.ibm.mq.jakarta.client.jar` 提供於 IBM MQ 安裝的 `java/lib` 子目錄中。

為了用於 OSGi 儲存器 (例如 Apache Felix 或 Eclipse Equinox)， IBM MQ 也提供一對 OSGi 軟體組:

- `com.ibm.mq.osgi.jms30.clientprereqs_`*`V.R.M.F`*`.jar`
- `com.ibm.mq.osgi.jms30.client_`*`V.R.M.F`*`.jar`

其中 *V.R.M.F* 代表 IBM MQ 的版本，例如 9.3.0.0。 您可以在 IBM MQ 安裝架構的 `java/lib/OSGi` 子目錄中找到這些組合。

交付內容: Jakarta EE 9

為了支援 Jakarta EE 9 相容應用程式伺服器中的 IBM MQ 型傳訊， IBM MQ 提供 Jakarta EE 9 相容資源配接器: `wmq.jakarta.jmsra.rar`。 這可在 IBM MQ 安裝架構的 `java/lib/jca` 子目錄中找到。

IBM MQ 繼續在 IBM MQ 安裝的 `java/lib/jca` 子目錄中提供 Java EE 7 相容資源配接器 `wmq.jmsra.rar`。

如何交付這些構件

「資源配接器」的這些 JAR 和 RAR 檔隨附於一般 IBM MQ 安裝媒體中的預先存在構件-平台專用安裝媒體 (例如 "`.rpm`" 檔案) 和可重新配送的媒體 (例如自行解壓縮的可重新配送用戶端 JAR 檔)。

JMS 2.0 與 Jakarta Messaging 3.0 之間的變更內容

Jakarta EE 9 和 Jakarta Messaging 3.0 未引進新功能。 所有的改變都是名字。 例如，在 JMS 2.0 中使用 "javax.jms.Connection"，則在 Jakarta Messaging 3.0 中使用 "jakarta.jms.Connection"。

當 Eclipse Foundation 採用 Jakarta EE 平台時，它會在這個基礎上建置，且這個命名慣例會用於未來引進的新功能。

IBM MQ classes for JMS 與 IBM MQ classes for Jakarta Messaging 之間的變更內容

摘要

提供 JMS 2.0 支援的 IBM MQ classes for JMS 仍然可用，建議主要用於維護及延伸現有應用程式。 完全支援它們。

IBM MQ classes for Jakarta Messaging 提供 Jakarta Messaging 3.0 的支援，建議用於新的開發。

在 IBM MQ 9.3.0 中，這兩個產品與服務在功能上是相等的。 只有命名不同。 不過，新的傳訊功能更可能出現在 IBM MQ classes for Jakarta Messaging 中，而不是在 IBM MQ classes for JMS 中。

這兩個供應項目可交互作業。 IBM MQ classes for Jakarta Messaging 可以耗用 IBM MQ classes for JMS 所產生的訊息，反之亦然。 但這兩個供應項目不能同時存在於單一應用程式中。

命名變更

表 *16:* 套件名稱的變更

IBM MQ classes for JMS 套件名稱	**IBM MQ classes for Jakarta Messaging 套件名稱**
com.ibm.mq.jms[.*]	com.ibm.mq.jakarta.jms[.*]
com.ibm.jms	com.ibm.jakarta.jms
com.ibm.msg.client.jms.*	com.ibm.msg.client.jakarta.jms.*

表 16: 套件名稱的變更 (繼續)	
IBM MQ classes for JMS 套件名稱	**IBM MQ classes for Jakarta Messaging 套件名稱**
com.ibm.msg.client.wmq.*	com.ibm.msg.client.jakarta.wmq.*

與共用服務 (追蹤、記載、國家語言支援等) 及 JMQI 實作 (本端及遠端) 相關的套件是 IBM MQ classes for JMS 及 IBM MQ classes for Jakarta Messaging 的共用套件，因此在這些區域中不需要任何變更。

請注意，內容名稱也已變更。 例如，在 IBM MQ classes for Jakarta Messaging 中啟用 IBM MQ 延伸的內容是 **com.ibm.mq.jakarta.jms.SupportMQExtensions**。

與 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 無關的內容名稱 (例如各種 **com.ibm.msg.client.commonservices.trace.*** 內容) 同樣適用於這兩個供應項目。

管理公用程式

crtmqenv 和 **setmqenv** 公用程式現在接受一個選項，用來指定是否應該針對 IBM MQ classes for JMS (`-j 2.0`) 或 IBM MQ classes for Jakarta Messaging (`-j 3.0`) 配置類別路徑，以及 **runjms** 公用程式是否有 IBM MQ classes for Jakarta Messaging 變式 (稱為 **runjms30** 及類似名稱)。

當要求報告 Java 元件時， **dspmqver** 公用程式會在其輸出中包括 IBM MQ classes for Jakarta Messaging 。

若要配置要透過 JNDI 擷取的 IBM MQ classes for Jakarta Messaging 物件，新的 **JMS30Admin** 公用程式相當於 IBM MQ classes for JMS 的 **JMSAdmin** 公用程式。

請注意，因為基礎物件來自不同的套件。 **JMSAdmin** 所建立的 JNDI 定義不能供 IBM MQ classes for Jakarta Messaging 使用， **JMS30Admin** 所建立的 JNDI 定義也不能供 IBM MQ classes for JMS 使用。

註: 不支援 IBM MQ Explorer 所提供的 IBM MQ classes for Jakarta Messaging 物件; 其 JNDI 整合僅適用於 IBM MQ classes for JMS 。

相關概念

為何應該將 IBM MQ 類別用於 Jakarta Messaging?

JMS 和 Jakarta Messaging 模型

JMS 和 Jakarta Messaging 模型定義一組介面，供 Java 應用程式用來執行傳訊作業。 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 是傳訊提供者，定義 Java 傳訊物件如何與 IBM MQ 概念相關。 JMS 和 Jakarta Messaging 規格預期某些傳訊物件是受管理物件。

從 IBM MQ 8.0 開始，本產品支援 JMS 標準的 JMS 2.0 版本，其中引進了簡化的 API ，同時保留 JMS 1.1 中的標準 API。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 引進 Jakarta Messaging 3.0 的支援。 仍完全支援 JMS 2.0 。 因為 JMS 和 Jakarta Messaging 有許多共同之處，本主題中對 JMS 的進一步參照可以視為同時參照兩者。 必要時會強調顯示任何差異。

簡化的 API

JMS 2.0 引進了簡化的 API ，同時保留 JMS 1.1 中的網域特定及網域無關介面。 簡化的 API 減少了傳送及接收訊息所需的物件數，且包含下列介面:

ConnectionFactory
ConnectionFactory 是 JMS 用戶端用於建立連線的受管理物件。 此介面也用於典型 API 中。

JMS 環境定義
此物件結合典型 API 的 Connection 及 Session 物件。 您可以使用重複的基礎連線，從其他 JMSContext 物件建立 JMSContext 物件。

JMS 生產者
JMSProducer 由 JMSContext 建立，且用於將訊息傳送至佇列或主題。 JMSProducer 物件會導致建立傳送訊息所需的物件。

JMS 消費者
JMSConsumer 由 JMSContext 建立，且用於接收來自主題或佇列的訊息。

簡化的 API 具有許多影響：

- JMSContext 物件會一律自動啟動基礎連線。
- 使用訊息的 getBody 方法，JMSProducers 及 JMSConsumers 現在可以直接處理訊息內文，而無需取得整個訊息物件。
- 在傳送「內文」（訊息內容）之前，使用方法鏈在 JMSProducer 物件上設定訊息內容。 JMSProducer 將處理傳送訊息所需的所有物件的建立作業。 使用 JMS 2.0，可以設定內容，且傳送的訊息如下：

```
context.createProducer().
setProperty("foo", "bar").
setTimeToLive(10000).
setDeliveryMode(NON_PERSISTENT).
setDisableMessageTimestamp(true).
send(dataQueue, body);
```

JMS 2.0 也引進了共用訂閱，其中可以在多個消費者之間共用訊息。 所有 JMS 1.1 訂閱視為非共用訂閱閱。

典型 API

下列清單彙總典型 API 的主要 JMS 介面：

目的地
目的地是應用程式傳送訊息的地方，及（或）應用程式從中接收訊息的來源。

ConnectionFactory
ConnectionFactory 物件封裝連線的一組配置內容。 應用程式使用 Connection Factory 來建立連線。

連線
Connection 物件封裝傳訊伺服器的應用程式的作用中連線。 應用程式使用連線建立階段作業。

Session
階段作業是用於傳送及接收訊息的單一執行緒環境定義。 應用程式使用階段作業建立訊息、訊息產生者及訊息消費者。 階段作業已交易或未交易。

訊息
Message 物件封裝應用程式傳送或接收的訊息。

MessageProducer
應用程式使用訊息產生者將訊息傳送至目的地。

MessageConsumer
應用程式使用訊息消費者接收已傳送至目的地的訊息。

第 128 頁的圖 53 顯示這些物件及其關係。

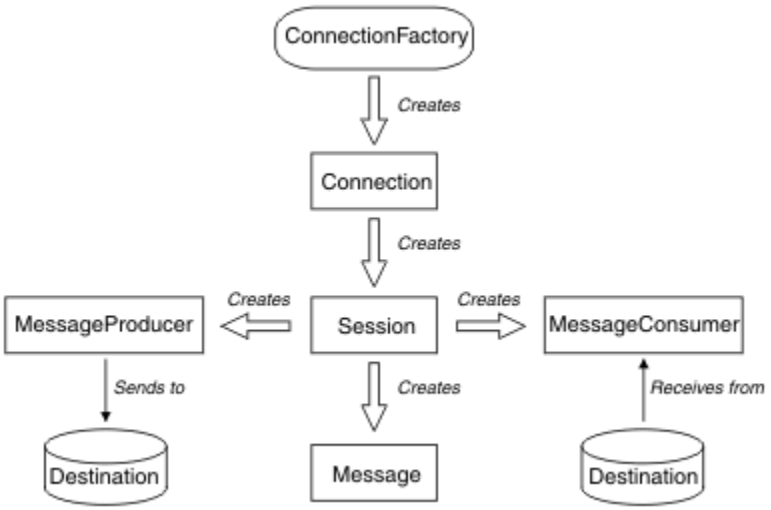


圖 *53: JMS* 物件及其關係

此圖表顯示下列主要介面：ConnectionFactory、Connection、Session、MessageProducer、MessageConsumer、Message 及 Destination。 應用程式使用 Connection Factory 建立連線，使用連線建立階段作業。 然後，應用程式可以使用階段作業建立訊息、訊息產生者及訊息消費者。 應用程式使用訊息產生者將訊息傳送至目的地，使用訊息消費者接收已傳送至目的地的訊息。

Destination、ConnectionFactory 或 Connection 物件可以同時供多執行緒應用程式的不同執行緒使用，但 Session、MessageProducer 或 MessageConsumer 物件不能同時供不同的執行緒使用。 確保不同時使用 Session、MessageProducer 或 MessageConsumer 的最簡單的方法是為每個執行緒建立個別 Session 物件。

JMS 支援以下兩種傳訊的樣式：

- 點對點傳訊
- 發佈/訂閱傳訊

這些傳訊樣式亦稱為傳訊網域，您可以將這兩種傳訊樣式結合在一個應用程式中。 在點對點網域中，目的地是佇列；在發佈/訂閱網域中，目的地是主題。

在 JMS 1.1 之前的 JMS 版本中，點對點網域的程式設計會使用一組介面和方法，而發佈/訂閱網域的程式設計則會使用另一組介面和方法。 兩組類似，但彼此獨立。 從 JMS 1.1 開始，您可以使用支援這兩種傳訊網域的一組共用介面及方法。 共用介面提供了每個傳訊網域的網域無關視圖。 第 128 頁的表 17 列出 JMS 網域無關介面及其對應的網域專用介面。

表 *17:* 與 *JMS* 網域無關及其對應的網域特定介面

網域無關介面	**點對點網域的網域專用介面**	**發佈/訂閱網域的網域專用介面**
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
連線	QueueConnection	TopicConnection
目的地	佇列	主題
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

JMS 2.0 IBM MQ classes for JMS 2.0 同時支援舊版 JMS 1.1 網域特定介面及 JMS 2.0 的簡化 API。 因此， IBM MQ classes for JMS 2.0 可用來維護現有應用程式，包括在現有應用程式中開發新功能。

JM 3.0 IBM MQ classes for Jakarta Messaging 3.0 支援相同介面的 Jakarta Messaging 版本，建議用於新的應用程式開發。

在 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 中， JMS 物件以下列方式與 IBM MQ 概念相關:

- Connection 物件包含的內容衍生自 Connection Factory 內容，用於建立連線。 這些內容控制應用程式連接至佇列管理程式的方式。 這些內容的範例是佇列管理程式的名稱，對於在用戶端模式中連接至佇列管理程式的應用程式，則是佇列管理程式正在其中執行的系統的主機名稱或 IP 位址。
- Session 物件封裝 IBM MQ 連線控點，因此定義階段作業的交易式範圍。
- MessageProducer 物件及 MessageConsumer 物件各自封裝 IBM MQ 物件控點。

使用 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 時，會套用 IBM MQ 的所有一般規則。 請特別注意，應用程式可以將訊息傳送至遠端佇列，但它只能接收來自應用程式所連接佇列管理程式擁有的佇列的訊息。

JMS 規格預期 ConnectionFactory 及 Destination 物件是受管理物件。 管理者可在中央儲存庫中建立及維護受管理物件，且 JMS 應用程式使用「Java 命名和目錄介面 (JNDI)」擷取這些物件。

在 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 中， 「目的地」介面的實作是「佇列」和「主題」的抽象超類別，因此「目的地」實例是「佇列」物件或「主題」物件。 網域無關介面將佇列或主題視為目的地。 MessageProducer 或 MessageConsumer 物件的傳訊網域由目的地是佇列還是主題來判斷。

因此，在 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 中， 下列類型的物件可以是受管理物件:

- ConnectionFactory
- QueueConnectionFactory
- TopicConnectionFactory
- 佇列
- 主題
- XAConnectionFactory
- XAQueueConnectionFactory
- XATopicConnectionFactory

IBM MQ classes for JMS/Jakarta Messaging 架構

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 具有分層架構。 程式碼最上層是任何 IBM Java 傳訊提供者都可以使用的一般層。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 引進 Jakarta Messaging 3.0 的支援。 仍完全支援 JMS 2.0 。

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 具有圖 第 130 頁的圖 54 所示的分層架構。 程式碼最上層是可供任何 IBM JMS 或 Jakarta Messaging 提供者使用的一般層。 當應用程式呼叫 JMS 或 Jakarta Messaging 方法時，一般層會執行任何非傳訊系統特有的呼叫處理程序，這也會提供一致的呼叫回應。 特定於傳訊系統的任何呼叫處理都會委派給低層。 在下圖中，IBM MQ 傳訊提供者與另兩個傳訊提供者（傳訊提供者 A 及傳訊提供者 B）一起顯示在低層。

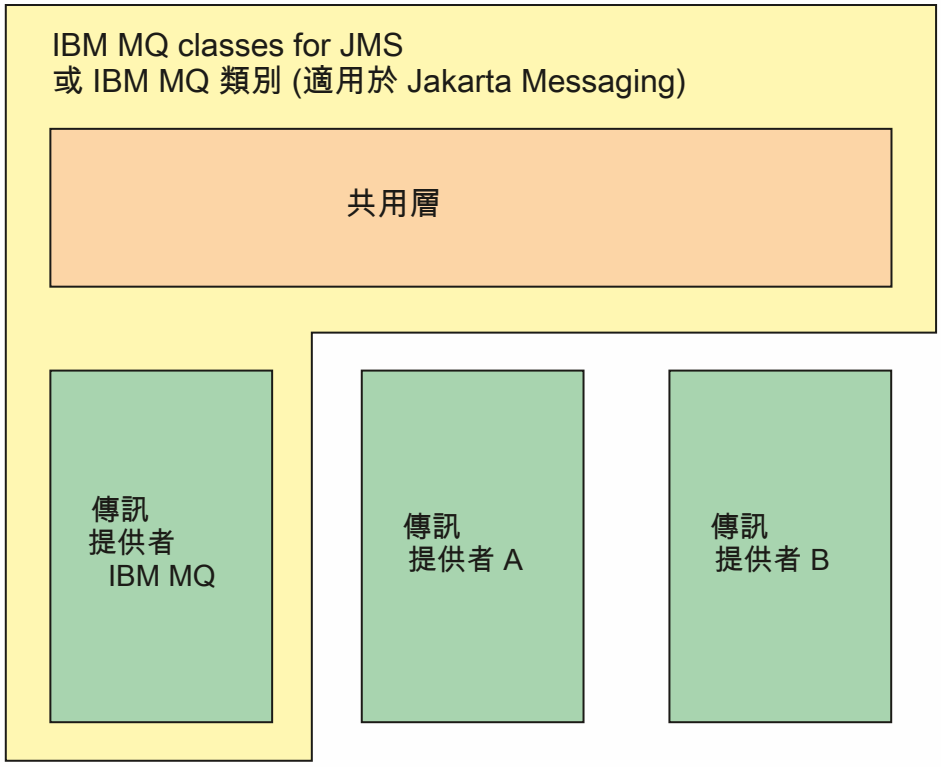


圖 *54: IBM JMS* 和 *Jakarta Messaging* 提供者的分層架構

分層式構造可實現下列目標:

- 增進各種 IBM JMS 和 Jakarta Messaging 提供者的行為一致性
- 更輕鬆地在兩個 IBM 傳訊系統之間撰寫橋接應用程式
- 更容易將應用程式從某個 IBM JMS 或 Jakarta Messaging 提供者移轉至另一個

相關工作

使用 IBM MQ 類別進行 JMS/Jakarta 傳訊

支援管理物件

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 支援使用受管理物件。

JM 3.0 V 9.3.0 V 9.3.0 從 IBM MQ 9.3.0 開始，支援 Jakarta Messaging 3.0 開發新的應用程式。 IBM MQ 9.3.0 繼續支援 JMS 2.0 現有的應用程式。 不支援在同一應用程式中同時使用 Jakarta Messaging 3.0 API 和 JMS 2.0 API。 如需相關資訊，請參閱 使用 IBM MQ 類別進行 JMS/Jakarta 傳訊。

JMS 或 IBM MQ classes for Jakarta Messaging 應用程式內的邏輯流程以 ConnectionFactory 和目的地物件開頭。 應用程式使用 ConnectionFactory 物件來建立 Connection 物件，此物件代表從應用程式到傳訊伺服器的作用中連線。 應用程式使用 Connection 物件來建立 Session 物件，這是用於產生及耗用訊息的單一執行緒環境定義。 然後，應用程式可以使用 Session 物件及 Destination 物件來建立 MessageProducer 物件，應用程式會使用該物件將訊息傳送至指定的目的地。 目的地是傳訊系統中的佇列或主題，由 Destination 物件封裝。 應用程式也可以使用 Session 物件和 Destination 物件來建立 MessageConsumer 物件，供應用程式用來接收已傳送至指定目的地的訊息。

JMS 和 Jakarta Messaging 規格預期 ConnectionFactory 和「目的地」物件是受管理物件。 管理者會在中央儲存庫中建立及維護受管理物件，而 JMS 或 Jakarta Messaging 應用程式會使用 Java Naming Directory Interface (JNDI) 來擷取這些物件。 受管理物件的儲存庫範圍可以從簡式檔案到「輕量型目錄存取通訊協定 (LDAP)」目錄。

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 支援使用受管理物件。 應用程式可以使用透過 IBM MQ 公開的 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 的所有特性，而不需要將任何 IBM MQ 特定資訊寫在應用程式本身中。 此安排為應用程式提供基本 IBM MQ 配置中的獨立性程度。

為了達到此獨立性，應用程式可以使用 JNDI 來擷取儲存為受管理物件的 Connection Factory 及目的地，並僅使用 `javax.jms` (JMS 2.0) 或 `jakarta.jms` (Jakarta Messaging 3.0) 套件中定義的介面來執行傳訊作業。

JMS 2.0 對於 JMS 2.0，管理者可以使用 IBM MQ JMS 管理工具 **JMSAdmin** 或 IBM MQ Explorer，在中央儲存庫中建立及維護受管理物件。

JM 3.0 對於 Jakarta Messaging 3.0，您無法使用 IBM MQ Explorer 來管理 JNDI。 **JMSAdmin** 的 Jakarta Messaging 3.0 變式 **JMS30Admin** 支援 JNDI 管理。

應用程式伺服器通常會提供它自己的受管理物件儲存庫，以及它自己用來建立及維護物件的工具。 因此，Java EE JM 3.0 或 Jakarta EE 應用程式可以使用 JNDI，從應用程式伺服器儲存庫或中央儲存庫擷取受管理物件。

相關工作

配置 JMS 和 Jakarta 傳訊資源

Java EE 和 Jakarta EE 平台上支援的通訊類型

在 Java EE 及 Jakarta EE 平台上， IBM MQ classes for JMS 及 IBM MQ classes for Jakarta Messaging 支援應用程式元件與 IBM MQ 佇列管理程式之間兩種類型的通訊。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 引進 Jakarta Messaging 3.0 的支援。 仍完全支援 JMS 2.0 。 因為 JMS 和 Jakarta Messaging 有許多共同之處，本主題中對 JMS 的進一步參照可以視為同時參照兩者。 必要時會強調顯示任何差異。

支援應用程式元件與 IBM MQ 佇列管理程式之間的下列兩種通訊類型：

- 出埠通訊
- 入埠通訊

出埠通訊

直接使用 JMS 或 Jakarta Messaging API，應用程式元件會建立與佇列管理程式的連線，然後傳送及接收訊息。

例如，應用程式元件可以是應用程式用戶端、Servlet、JavaServer Page (JSP)、企業 Java Bean (EJB) 或訊息驅動 Bean (MDB)。 在此通訊類型中，應用程式伺服器儲存器僅提供支援傳訊作業的低層次功能，如連線儲存區及執行緒管理。

入埠通訊

若為入埠通訊，到達目的地的訊息會遞送至 MDB，然後處理訊息。

Java EE JM 3.0 和 Jakarta EE 應用程式使用 MDB 非同步地處理訊息。 MDB 用作 JMS 訊息接聽器，且由 onMessage() 方法實作，從而定義處理訊息的方式。 MDB 部署在應用程式伺服器的 EJB 儲存器中。 配置 MDB 的精確方式視您使用的應用程式伺服器而定，但配置資訊必須指定要連接的佇列管理程式、連接至佇列管理程式的方式、監視是否有訊息的目的地，以及 MDB 的交易式行為。 然後，EJB 儲存器會使用此資訊。 當符合 MDB 選取準則的訊息到達指定的目的地時， EJB 儲存器會使用 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 從佇列管理程式擷取訊息，然後呼叫其 onMessage() 方法，將訊息遞送至 MDB。

與 IBM MQ classes for Java 的關係

IBM MQ classes for Java、 IBM MQ classes for Jakarta Messaging 及 IBM MQ classes for JMS 是使用 MQI 共用 Java 介面的對等節點。

第 132 頁的圖 55 顯示 IBM MQ classes for JMS、 IBM MQ classes for Jakarta Messaging 與 IBM MQ classes for Java 之間的關係。

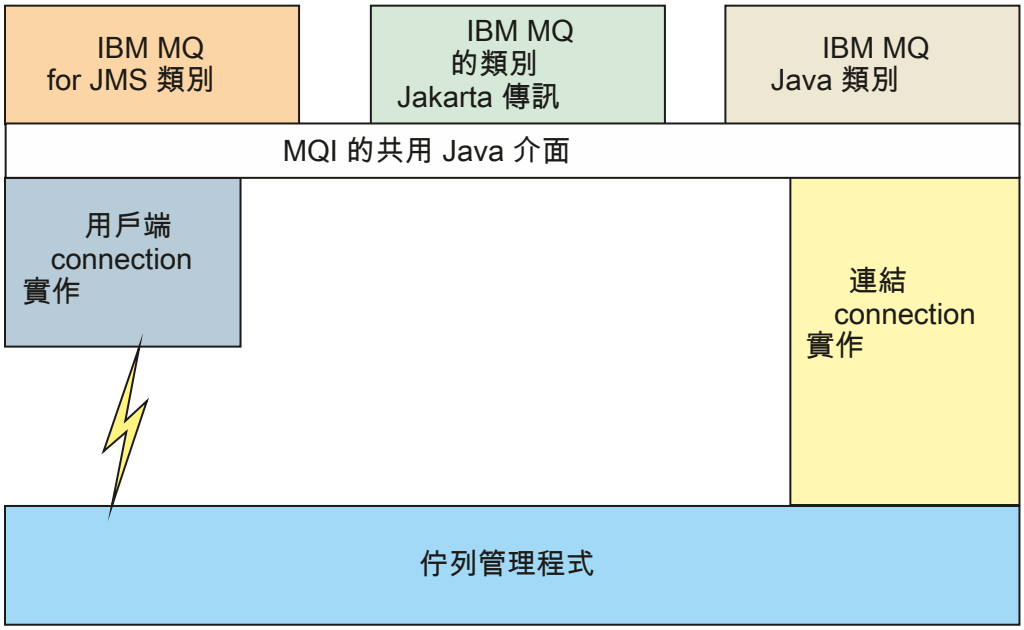


圖 *55: IBM MQ classes for JMS*、 *IBM MQ classes for Jakarta Messaging* 與 *IBM MQ classes for Java* 之間的關係

一般而言， Java 程式應該只使用一個介面來與 IBM MQ - IBM MQ classes for Java、 IBM MQ classes for Jakarta Messaging 或 IBM MQ classes for JMS 互動。 不支援混合介面，但有一個例外。 為保持與 IBM WebSphere MQ 7.0 之前的版次相容，以 Java 撰寫的通道結束程式類別仍然可以使用 IBM MQ classes for Java 介面，即使從 IBM MQ classes for JMS 呼叫通道結束程式類別。 不過，使用 IBM MQ classes for Java 介面表示您的應用程式仍然相依於下列任一項:

- JMS 2.0 IBM MQ classes for Java JAR 檔 `com.ibm.mq.jar`。 如果不想要類別路徑中的 `com.ibm.mq.jar`，您可以改用 `com.ibm.mq.exits` 套件中的介面集。
- JM 3.0 V 9.3.0 V 9.3.0 與 IBM MQ classes for Jakarta Messaging 交互作業時使用 `com.ibm.mq.jakarta.client.jar`。

相關概念

V 9.3.0 V 9.3.0 為何應該將 IBM MQ 類別用於 Jakarta Messaging?
為何應該使用適用於 JMS 的 IBM MQ 類別?
為何應該將 IBM MQ 類別用於 Java?

IBM MQ 傳訊提供者

IBM MQ 傳訊提供者具有三種作業模式：標準模式、有限制的標準模式及移轉模式。

IBM MQ 傳訊提供者具有以下三種作業模式:

- IBM MQ 傳訊提供者標準模式
- 有限制的 IBM MQ 傳訊提供者標準模式
- IBM MQ 傳訊提供者移轉模式

IBM MQ 傳訊提供者標準模式使用 IBM MQ 佇列管理程式的所有特性實作 JMS。 此模式已最佳化為使用 JMS 2.0 JM 3.0 V 9.3.0 V 9.3.0 或 Jakarta Messaging 3.0 API 及功能。

如果:

- 用戶端在 **ConnectionFactory** 上指定提供者版本 6，用戶端的行為與 IBM WebSphere MQ 6.0 提供的用戶端相容。 僅支援 JMS 1.1 和 JMS 2 介面，但部分 JMS 2 功能 (例如共用訂閱、遞送延遲和非同步傳送) 已停用。 沒有連線共用。
- 用戶端在 **ConnectionFactory** 上指定提供者版本 7，完全支援 JMS 1.1 和 JMS 2 介面。
- 未指定提供者版本，嘗試與提供者第 7 版連接。 如果失敗，則會對提供者第 6 版進一步嘗試。

如果您想要使用 IBM MQ Enterprise Transport 來連接至 IBM Integration Bus，請使用移轉模式。 如果您使用 IBM MQ Real-Time Transport，則會自動選取轉移模式，因為您已明確選取 Connection Factory 物件中的內容。 使用 IBM MQ Enterprise Transport 來連線至 IBM Integration Bus 時，會遵循 配置 JMS **PROVIDERVERSION** 內容中所說明的模式選擇一般規則。

相關工作

配置 JMS 資源

z/OS IBM MQ for z/OS 概念

IBM MQ for z/OS 所使用的部分概念對於 z/OS 平台而言是唯一的。 例如，記載機制、儲存體管理技術、回復單元處置及佇列共用群組僅隨 IBM MQ for z/OS 提供。 請使用本主題作為簡介，以取得這些概念的進一步相關資訊。

概念包括 IBM MQ for z/OS 所使用物件的概觀，包括:

- 佇列管理程式
- 通道起始程式
- 共用佇列及佇列共用群組
- 內部群組佇列

下列主題也涵蓋您需要的各種程序，包括:

- z/OS 上的系統定義
- 儲存管理
- 回復及重新啟動
- IBM MQ for z/OS 中的安全概念

相關概念

第 134 頁的『z/OS 上的佇列管理程式』
您必須先安裝 IBM MQ for z/OS 產品並啟動佇列管理程式，然後才能讓應用程式在 z/OS 系統上使用 IBM MQ 。 佇列管理程式會擁有並管理 IBM MQ 所使用的資源集。

第 135 頁的『z/OS 上的通道起始程式』
通道起始程式提供並管理啟用 IBM MQ 分散式佇列作業的資源。 IBM MQ 使用 訊息通道代理程式 (MCA) 將訊息從一個佇列管理程式傳送至另一個佇列管理程式。

第 137 頁的『用於管理 IBM MQ for z/OS 的術語及作業』
請使用本主題作為 IBM MQ for z/OS 專有名詞及特定作業的簡介。

第 139 頁的『共用佇列及佇列共用群組』
您可以使用共用佇列及佇列共用群組，來實作 IBM MQ 資源的高可用性。 在 z/OS 平台上，共用佇列和佇列共用群組是 IBM MQ for z/OS 特有的功能。

第 176 頁的『內部群組佇列』
本節說明內部群組佇列作業，這是 z/OS 平台特有的 IBM MQ for z/OS 功能。 此功能僅適用於定義給佇列共用群組的佇列管理程式。

第 187 頁的『z/OS 上的儲存體管理』
IBM MQ for z/OS 需要永久及暫時資料結構，並使用頁集及記憶體緩衝區來儲存此資料。 這些主題提供更多關於 IBM MQ 如何使用這些頁集和緩衝區的詳細資料。

第 191 頁的『登入 IBM MQ for z/OS』
當發生資料變更及重要事件時， IBM MQ 會維護這些事件的 日誌 。 必要的話，這些日誌可用來將資料回復至前一個狀態。

第 210 頁的『在 z/OS 上回復並重新啟動』
請使用本主題中的鏈結，以瞭解 IBM MQ for z/OS 的重新啟動及回復功能。

第 223 頁的『IBM MQ for z/OS 中的安全概念』
請利用這個主題來瞭解 IBM MQ 安全的重要性，以及在系統上沒有足夠安全設定的含意。

第 229 頁的『z/OS 上的可用性』
IBM MQ for z/OS 具有許多高可用性特性。 本主題說明可用性的部分考量。

第 232 頁的『z/OS 上的回復單元處置』
當連接至佇列共用群組 (QSG) 中的佇列管理程式時，某些交易式應用程式可以使用 GROUP 而非 QMGR 回復處置單元，方法是在連接時指定 QSG 名稱，而不是佇列管理程式名稱。 這可移除重新連接至 QSG 中相同佇列管理程式的需求，讓交易回復更靈活且更健全。

相關參考

第 200 頁的『z/OS 上的系統定義』
IBM MQ for z/OS 使用許多預設物件定義，並提供範例 JCL 來建立那些預設物件。 請利用這個主題來瞭解這些預設物件和範例 JCL。

第 231 頁的『IBM MQ for z/OS 上的監視及統計資料』
IBM MQ for z/OS 具有一組機能，用於監視佇列管理程式及收集統計資料。

z/OS z/OS 上的佇列管理程式

您必須先安裝 IBM MQ for z/OS 產品並啟動佇列管理程式，然後才能讓應用程式在 z/OS 系統上使用 IBM MQ 。 佇列管理程式會擁有並管理 IBM MQ 所使用的資源集。

佇列管理程式

佇列管理程式 是向應用程式提供傳訊服務的程式。 使用「訊息佇列介面 (MQI)」的應用程式可以將訊息放到佇列中，以及從佇列取得訊息。 佇列管理程式還可以確定訊息是傳送到正確的佇列，或是遞送到另一個佇列管理程式。 也可以處理發給它的 MQI 呼叫，以及提送給它的指令（無論是來自哪一個來源）。 同時，它還可以針對每一個呼叫或指令，產生適當的完成碼。

佇列管理程式所管理的資源包括:

- 保留 IBM MQ 物件定義及訊息資料的頁面集
- 在佇列管理程式失敗時用來回復訊息及物件的日誌
- 處理器儲存體
- 不同應用程式環境 (CICS、 IMS 及批次) 可透過其存取 IBM MQ API 的連線
- IBM MQ 通道起始程式，容許在 z/OS 系統上的 IBM MQ 與其他系統之間進行通訊

佇列管理程式具有名稱，且應用程式可以使用此名稱來連接至該佇列管理程式。

第 135 頁的圖 56 說明佇列管理程式，其中顯示與不同應用程式環境的連線，以及通道起始程式。

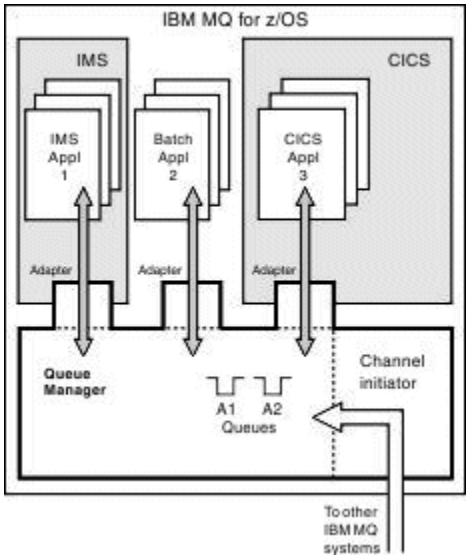


圖 *56: IBM MQ for z/OS* 的概觀

z/OS 上的佇列管理程式子系統

在 z/OS 上， IBM MQ 會以在 IPL 時啟動的 z/OS 子系統來執行。 在子系統中，佇列管理程式是透過執行 JCL 程序來啟動，該 JCL 程序指定包含日誌相關資訊且保留物件定義及訊息資料 (頁面集) 的 z/OS 資料集。 子系統與佇列管理程式具有相同的名稱，最多四個字元。 網路中的所有佇列管理程式都必須具有唯一名稱，即使它們位於不同的系統、Sysplex 或平台上也一樣。

z/OS 上的通道起始程式

通道起始程式提供並管理啟用 IBM MQ 分散式佇列作業的資源。 IBM MQ 使用 訊息通道代理程式 (MCA) 將訊息從一個佇列管理程式傳送至另一個佇列管理程式。

若要將訊息從佇列管理程式 A 傳送至佇列管理程式 B ，佇列管理程式 A 上的 傳送端 MCA 必須設定佇列管理程式 B 的通訊鏈結。 必須在佇列管理程式 B 上啟動 接收中 MCA ，才能從通訊鏈結接收訊息。 此單向路徑由傳送端 MCA、通訊鏈結及接收端 MCA 組成，稱為 通道。 傳送端 MCA 會從傳輸佇列取得訊息，並將它們透過通道傳送至接收端 MCA。 接收 MCA 會接收訊息，並將它們放到目的地佇列中。

在 IBM MQ for z/OS 中，傳送及接收 MCA 都在通道起始程式內執行 (通道起始程式也稱為 移轉裝置)。 通道起始程式在佇列管理程式的控制下以 z/OS 位址空間執行。 只能有單一通道起始程式連接至佇列管理程式，且它在與佇列管理程式相同的 z/OS 映像檔內執行。 可能有數千個 MCA 處理程序同時在通道起始程式內執行。

第 136 頁的圖 57 顯示 Sysplex 內的兩個佇列管理程式。 每一個佇列管理程式都有通道起始程式和本端佇列。 AIX 和 Windows 上的佇列管理程式所傳送的訊息會放在本端佇列中，由應用程式從中擷取。 回覆訊息由類似的路徑傳回。

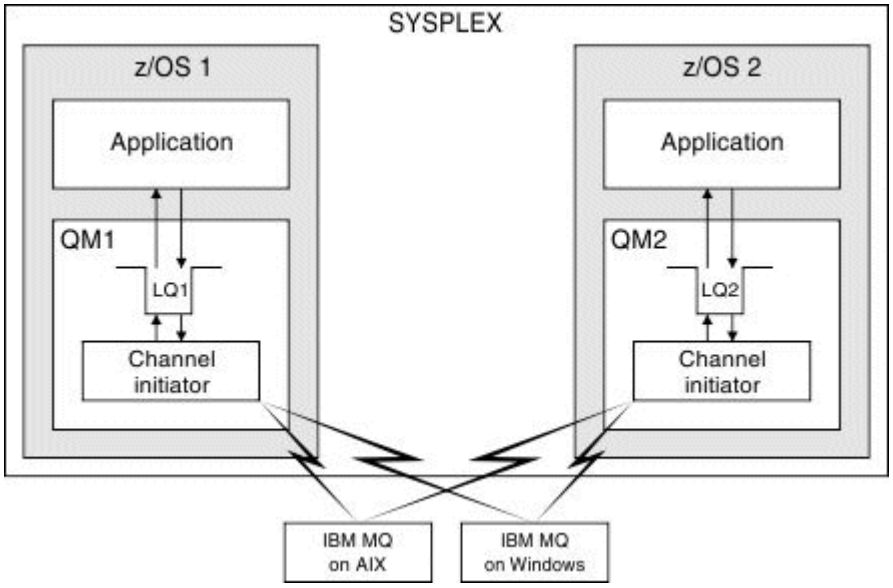


圖 *57:* 佇列管理程式之間的通訊

通道起始程式也包含與通道管理相關的其他處理程序。 這些程序包括:

接聽器
這些處理程序會接聽通訊子系統 (例如 TCP) 上的入埠通道要求，並在收到入埠要求時啟動具名 MCA。

監督人
這會管理通道起始程式位址空間，例如，它負責在失敗之後重新啟動通道。

名稱伺服器
這是用來將 TCP 名稱解析成位址。

TLS 作業
這些是用來執行加密和解密，以及檢查憑證撤銷清冊。

z/OS 通道起始程式的 SMF 記錄

通道起始程式 (CHINIT) 可以產生 SMF 統計資料記錄，以及含有作業和通道相關資訊的帳戶記錄。

CHINIT 可以使用下列類型的資訊來產生 SMF 統計資料記錄及統計記錄:

- 作業: 分派器、配接器、網域名稱伺服器 (DNS) 及 SSL。 這些作業形成稱為 CHINIT 統計資料的內容。
- 通道: 提供與 DIS CHSTATUS 指令提供的帳戶資訊類似的帳戶資訊。 這稱為通道統計。

IBM MQ for Multiplatforms 會將 PCF 訊息寫入 SYSTEM.ADMIN.STATISTICS.QUEUE。 如需如何在 IBM MQ for Multiplatforms 上記錄統計資料資訊的進一步資訊，請參閱 通道統計資料訊息資料 。

統計資料

您可以使用此資訊來找出下列資訊:

- 您是否需要更多 CHINIT 作業，例如 SSL TCB 數目，以及這些作業所使用的 CPU 數量。
- 這些作業上要求的平均時間。
- 針對 DNS 及 SSL 作業，間隔中的最長持續時間要求，以及發生此事件的當天時間。 您可以將此時間與您可能遇到的頻道問題產生關聯。

帳戶資料

您可以使用此資訊來監視通道使用情形，並找出下列資訊:

- 具有最高傳輸量的通道。
- 傳送訊息的速率，以及傳送資料的速率 (MB/ 秒)。
- 達到的批次大小。 如果達到的批次大小接近指定給通道的批次大小，通道可能接近傳送訊息的限制。

您可以使用 START TRACE 及 STOP TRACE 指令來控制帳戶追蹤及統計資料追蹤的收集。 您可以在通道及佇列管理程式上使用 STATCHL 及 STATACLS 選項，來控制通道是否產生 SMF 資料。

z/OS 用於管理 IBM MQ for z/OS 的術語及作業

請使用本主題作為 IBM MQ for z/OS 專有名詞及特定作業的簡介。

管理 IBM MQ for z/OS 所需的部分術語及作業是 z/OS 平台特有的。 下列清單包含其中一些術語及作業。

- 共用佇列
- 頁集及緩衝池
- 記載
- 修改佇列管理程式環境
- 重新啟動及回復
- 安全
- 可用性
- 操作物件
- 監視及統計資料
- 應用程式環境

共用佇列

佇列可以是 非共用(由一個佇列管理程式擁有且只能存取) 或 共用(由 佇列共用群組擁有)。 佇列共用群組由一些在單一 z/OS Sysplex 內執行的佇列管理程式組成，這些佇列管理程式可以同時存取相同的 IBM MQ 物件定義及訊息資料。 在佇列共用群組內，可共用物件定義儲存在共用 Db2 資料庫中。 共用佇列訊息保留在一或多個連結機能結構 (CF 結構) 內。 如果訊息資料太大而無法直接儲存在結構中 (大小超過 63 KB) ，或如果訊息夠大，而安裝定義規則選取它來卸載，則訊息控制資訊仍會儲存在連結機能項目中，但訊息資料會卸載至共用訊息資料集 (SMDS) 或共用 Db2 資料庫。 共用訊息資料集、共用 Db2 資料庫及連結機能結構是由群組中所有佇列管理程式共同管理的資源。

頁集及緩衝池

將訊息放入非共用佇列時，佇列管理程式會將資料儲存在頁集上，以便在後續作業從相同佇列取得訊息時可以擷取該資料。 如果從佇列中移除訊息，則稍後會釋放頁集中保留資料的空間以供重複使用。 隨著佇列上保留的訊息數目增加，因此頁集中使用的空間量會增加，且隨著佇列上的訊息數目減少，頁集中使用的空間也會減少。

為了降低將資料寫入頁集以及從頁集中讀取資料的效能成本，佇列管理程式會將更新項目緩衝到處理器儲存體中。 用來緩衝頁集存取的儲存體量是透過 IBM MQ 稱為 緩衝池的物件來控制。

如需頁集及緩衝池的相關資訊，請參閱 儲存體管理。

記載

對保留在頁集上的物件所做的任何變更，以及對持續訊息的作業，都會記錄為日誌記錄。 這些日誌記錄會寫入稱為 作用中日誌的日誌資料集。 作用中日誌資料集的名稱和大小保留在稱為 引導資料集 (BSDS) 的資料集中。

當作用中日誌資料集填滿時，佇列管理程式會切換至另一個日誌資料集，以便記載可以繼續，並將完整作用中日誌資料集的內容複製到 保存日誌 資料集。 這些動作的相關資訊 (包括保存日誌資料集的名稱) 保留在引導資料集中。 在概念上，佇列管理程式會循環使用一系列作用中日誌資料集; 當作用中日誌填滿時，日誌資料會卸載至保存日誌，且作用中日誌資料集可供重複使用。

如需日誌和引導資料集的相關資訊，請參閱 第 191 頁的『登入 IBM MQ for z/OS』。

修改佇列管理程式環境

當佇列管理程式啟動時，會讀取一組起始設定參數，以控制佇列管理程式的運作方式。 此外，還會讀取包含 IBM MQ 指令的資料集，並執行它們包含的指令。 一般而言，這些資料集包含 IBM MQ 執行所需的系統物件定義，您可以自訂這些定義，以定義或起始設定作業環境所需的 IBM MQ 物件。 當讀取這些資料集時，它們所定義的任何物件都會儲存在頁集或 Db2 中。

如需起始設定參數及系統物件的相關資訊，請參閱 第 200 頁的『z/OS 上的系統定義』。

回復及重新啟動

在 IBM MQ 作業期間的任何時間，處理器儲存體中可能有尚未寫入頁集的變更。 這些變更會寫出至佇列管理程式內背景作業最近最少使用的頁集。

如果佇列管理程式異常終止，則佇列管理程式重新啟動的回復階段可以回復遺失的頁集變更，因為持續訊息資料保留在日誌記錄中。 這表示 IBM MQ 可以回復持續訊息資料及物件變更，直到失敗點為止。

如果佇列共用群組成員的佇列管理程式發生連結機能失敗，則只有在您已備份連結機能結構時，才能回復該佇列上的持續訊息。

如需回復及重新啟動的相關資訊，請參閱 第 210 頁的『在 z/OS 上回復並重新啟動』。

安全

您可以使用外部安全管理程式，例如「安全伺服器」(先前稱為 RACF) 以保護 IBM MQ 擁有及管理的資源不受未獲授權使用者存取。 您也可以使用「傳輸層安全 (TLS)」來進行通道安全。 TLS 包含在 IBM MQ 產品中。

如需 IBM MQ 安全的相關資訊，請參閱 第 223 頁的『IBM MQ for z/OS 中的安全概念』。

可用性

IBM MQ 有數個特性設計為在佇列管理程式或通訊子系統失敗時增加系統可用性。 如需這些特性的相關資訊，請參閱 第 229 頁的『z/OS 上的可用性』。

操作物件

當佇列管理程式在執行中，您可以透過 z/OS 主控台介面，或透過使用 TSO 下 ISPF 服務的管理公用程式，來操作 IBM MQ 物件。 這兩種機制都可讓您定義、變更或刪除 IBM MQ 物件。 您也可以控制及顯示各種 IBM MQ 及佇列管理程式功能的狀態。

如需這些機能的相關資訊，請參閱 您可以從中在 IBM MQ for z/OS 上發出 MQSC 及 PCF 指令的來源。

您也可以使用「 IBM MQ 探險家」來操作 IBM MQ 物件，這是一個圖形使用者介面，提供視覺化方式來使用佇列、佇列管理程式及其他物件。

監視及統計資料

有數個機能可用來監視佇列管理程式及通道起始程式。 您也可以收集用於績效評估及會計用途的統計資料。

如需這些機能的相關資訊，請參閱 第 231 頁的『IBM MQ for z/OS 上的監視及統計資料』。

應用程式環境

當佇列管理程式已啟動時，應用程式可以連接至佇列管理程式，並使用 IBM MQ API 開始。 這些可以是 CICS、 IMS、Batch 或 WebSphere Application Server 應用程式。 IBM MQ 應用程式也可以使用 CICS 及 IMS 橋接器，來存取 CICS 及 IMS 系統上不知道 IBM MQ 的應用程式。

如需這些機能的相關資訊，請參閱 第 234 頁的『IBM MQ 及其他 z/OS 產品』。

如需撰寫 IBM MQ 應用程式的相關資訊，請參閱下列文件:

- 開發應用程式
- 使用 C++
- 使用 IBM MQ classes for Java

z/OS 共用佇列及佇列共用群組

您可以使用共用佇列及佇列共用群組，來實作 IBM MQ 資源的高可用性。 在 z/OS 平台上，共用佇列和佇列共用群組是 IBM MQ for z/OS 特有的功能。

本節說明屬性和好處，並提供數個佇列管理程式如何共用相同佇列及那些佇列上的訊息的相關資訊。

何謂共用佇列?

共用佇列是一種本端佇列類型。 Sysplex 中的一或多個佇列管理程式可以存取該佇列上的訊息。

佇列共用群組

可以存取同一組共用佇列的佇列管理程式會形成稱為 佇列共用群組的群組。

任何佇列管理程式都可以存取訊息

佇列共用群組中的任何佇列管理程式都可以存取共用佇列。 這表示您可以將訊息放置在一個佇列管理程式的共用佇列中，並從不同佇列管理程式的佇列中取得相同的訊息。 這為佇列共用群組內的通訊提供快速機制，不需要佇列管理程式之間的通道處於作用中狀態。

IBM MQ 支援將訊息卸載至 Db2 或共用訊息資料集 (SMDS)。 可配置任何大小的訊息卸載。

第 140 頁的圖 58 顯示三個佇列管理程式及一個連結機能，形成佇列共用群組。 這三個佇列管理程式都可以在連結機能中存取共用佇列。

應用程式可以連接至佇列共用群組內的任何佇列管理程式。 因為佇列共用群組中的所有佇列管理程式都可以存取所有共用佇列，所以應用程式並不取決於特定佇列管理程式的可用性; 佇列共用群組中的任何佇列管理程式都可以為佇列提供服務。

這會提供更大的可用性，因為如果其中一個佇列管理程式有問題，佇列共用群組中的所有其他佇列管理程式都可以繼續處理佇列。

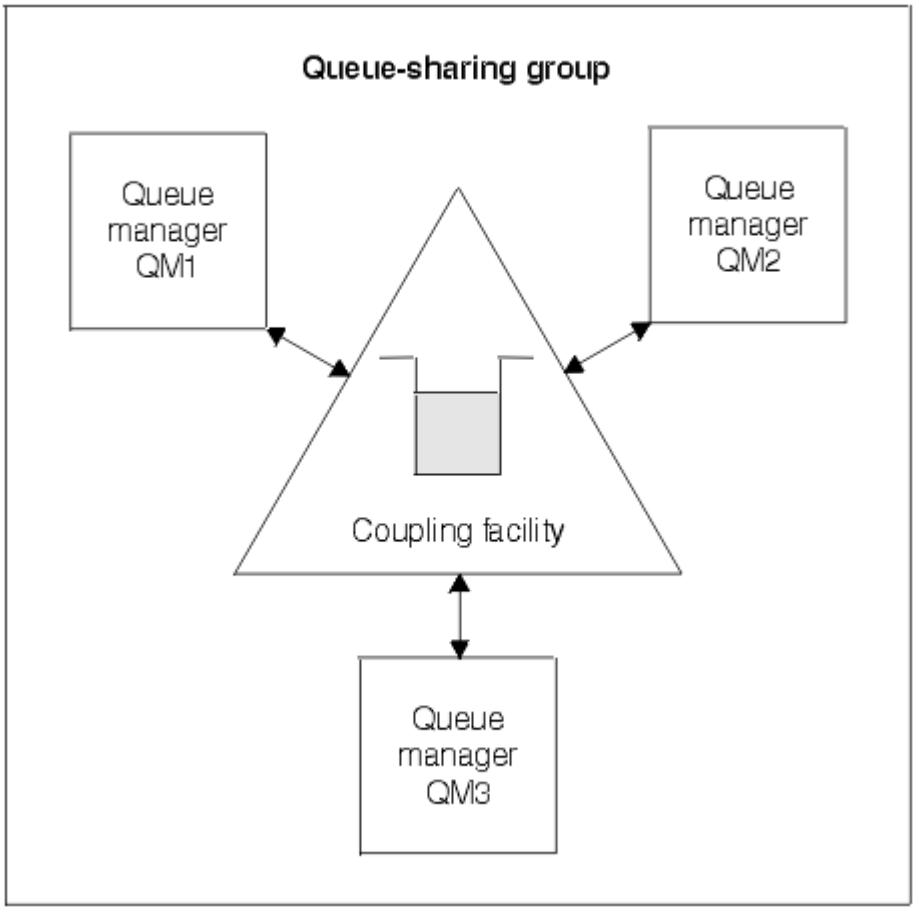


圖 *58:* 佇列共用群組

佇列定義由所有佇列管理程式共用

共用佇列定義儲存在 Db2 資料庫表格 OBJ_B_QUEUE 中。 因此，您只需要定義佇列一次，佇列共用群組中的所有佇列管理程式就可以存取它。 這表示要建立的定義較少。

相反地，非共用佇列的定義儲存在擁有佇列之佇列管理程式的頁集零 (如 頁集 中所述)。

如果已在定義佇列管理程式的頁集上定義具有該名稱的佇列，則無法定義共用佇列。 同樣地，如果存在同名的共用佇列，則您無法在佇列管理程式頁集上定義佇列的本端版本。

何謂佇列共用群組?

可以存取相同共用佇列的佇列管理程式群組稱為佇列共用群組。 佇列共用群組的每一個成員都有權存取同一組共用佇列。

佇列共用群組有一個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。

第 141 頁的圖 59 說明包含兩個佇列管理程式的佇列共用群組。 每一個佇列管理程式都有一個通道起始程式及其自己的本端頁集和日誌資料集。

佇列共用群組的每一個成員也必須連接至 Db2 系統。 Db2 系統必須全部位於相同的 Db2 資料共用群組中，佇列管理程式才能存取用來保留共用物件定義的 Db2 共用儲存庫。 這些是任何類型 IBM MQ 物件 (例如，佇列及通道) 的定義，它們只定義一次，然後群組中的任何佇列管理程式就可以使用它們。 這些稱為 廣域 定義，並在 專用及廣域定義中說明。

多個佇列共用群組可以參照特定的資料共用群組。 您可以在 IBM MQ 系統參數中指定 Db2 子系統的名稱，以及佇列管理程式在啟動時使用的資料共用群組。

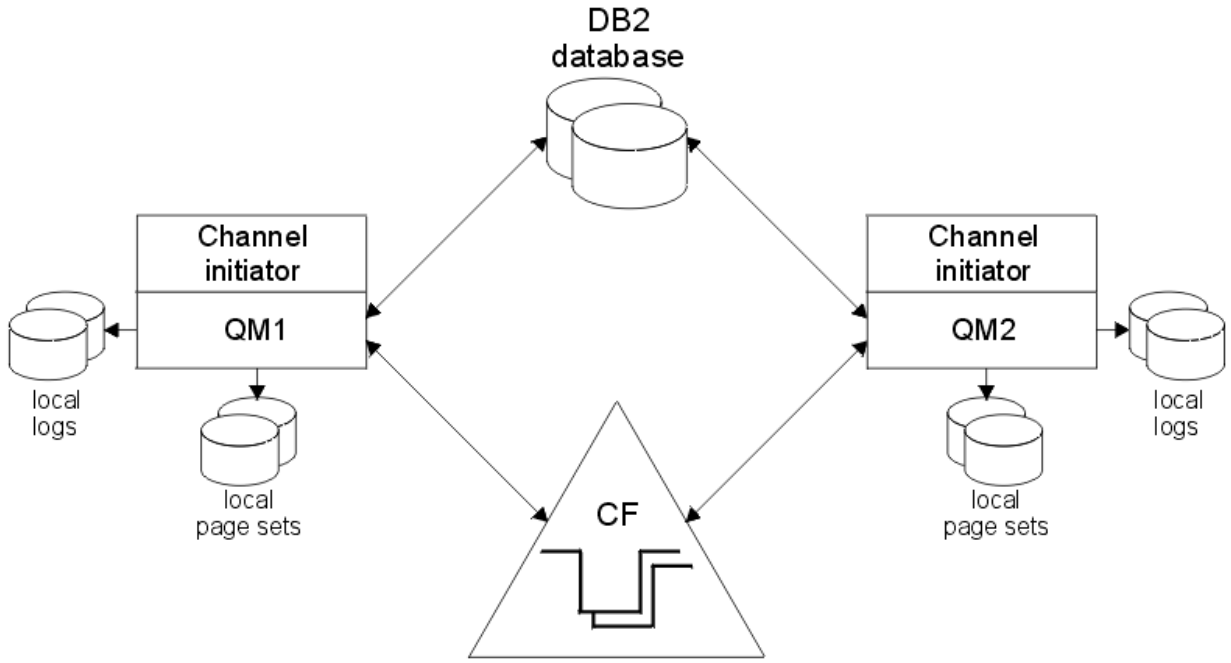


圖 *59:* 佇列共用群組中佇列管理程式的元件

當佇列管理程式已加入佇列共用群組時，它可以存取定義給該群組的共用物件，且您可以使用該佇列管理程式在群組內定義新的共用物件。 如果在群組內定義共用佇列，您可以使用此佇列管理程式，將訊息放置在那些共用佇列中，以及從那些共用佇列中取得訊息。 群組中的任何佇列管理程式都可以擷取保留在共用佇列上的訊息。

您可以輸入 MQSC 指令一次，並讓它在佇列共用群組內的所有佇列管理程式上執行，如同已個別在每一個佇列管理程式上輸入它一樣。 指令範圍 屬性用於此作業。 此屬性在 將指令導向不同的佇列管理程式中說明。

當佇列管理程式以佇列共用群組的成員身分執行時，必須能夠區分私密定義給該佇列管理程式的 IBM MQ 物件，以及廣域定義可供佇列共用群組中所有佇列管理程式使用的 IBM MQ 物件。 佇列共用群組處置 屬性用於此。 此屬性在 專用及廣域定義中說明。

您可以定義一組安全設定檔，以控制對群組內任何位置的 IBM MQ 物件的存取權。 這表示您必須定義的設定檔數目已大幅減少。

佇列管理程式只能屬於一個佇列共用群組，且群組中的所有佇列管理程式都必須位於相同的 Sysplex 中。 您可以在啟動時，在系統參數中指定佇列管理程式所屬的佇列共用群組。

相關概念

第 142 頁的『保留共用佇列訊息的位置?』
共用佇列上的每一則訊息都由 z/OS 連結機能清單結構中的一個項目代表。 如果訊息資料太大而無法放入相同項目中，則會將它卸載至共用訊息資料集 (SMDS) 或 Db2。

第 154 頁的『使用共用佇列的優點』
共用佇列容許 IBM MQ 應用程式具有可調式、高可用性，並容許實作工作量平衡。

第 171 頁的『分散式佇列及佇列共用群組』
分散式佇列及佇列共用群組是兩種可用來增加應用程式系統可用性的技術。 請利用這個主題來尋找這些技術的進一步相關資訊。

第 174 頁的『影響共用佇列的工作量配送』
請利用這個主題來瞭解影響佇列共用群組中共用佇列之工作量配送的因素。

相關參考

第 175 頁的『在何處尋找共用佇列及佇列共用群組的相關資訊』

請利用這個主題中的表格來尋找 IBM MQ for z/OS 如何使用共用佇列和佇列共用群組的相關資訊。

z/OS 保留共用佇列訊息的位置?

共用佇列上的每一則訊息都由 z/OS 連結機能清單結構中的一個項目代表。 如果訊息資料太大而無法放入相同項目中，則會將它卸載至共用訊息資料集 (SMDS) 或 Db2。

如果 CF 結構已配置為使用「系統類別記憶體 (SCM)」，則 IBM MQ 可以在不需要其他配置的情況下使用此結構。

重要: IBM z16 計劃成為最後一代 IBM Z®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

共用佇列訊息儲存體

放置在共用佇列上的訊息不會儲存在頁集上，且不會使用緩衝池。

共用佇列中的訊息在 z/OS 連結機能 (CF) 中的清單結構上具有項目。 相同 Sysplex 中的許多佇列管理程式可以使用 CF 清單結構來存取那些訊息。

小型共用佇列訊息的訊息資料通常包含在連結機能項目中。 若為較大的訊息，訊息資料可以儲存在共用訊息資料集 (SMDS) 中，或儲存在 Db2 資料共用群組所共用的 Db2 表格中的一或多個二進位大型物件 (BLOB)。 超過 63 KB 的訊息資料一律卸載至 SMDS 或 Db2。 較小的訊息也可以選擇性地以相同方式卸載，以節省連結機能結構中的空間。 如需詳細資料，請參閱第 143 頁的『指定共用訊息的卸載選項』。

放置在共用佇列上的訊息會在連結機能結構中參照，直到 MQGET 擷取為止。 連結機能作業用於:

- 搜尋下一個可擷取訊息
- 鎖定共用佇列上未確定的訊息
- 通知感興趣的佇列管理程式已確定訊息到達

持續訊息上的 MQPUT 及 MQGET 作業會記錄在執行該作業之佇列管理程式的日誌中。 這可將連結機能失敗時資料流失的風險降至最低。

連結機能

在連結機能內參照保留在共用佇列上的訊息。 連結機能位於 Sysplex 中任何 z/OS 映像檔之外，且通常配置為在不同的電源供應器上執行。 因此，連結機能在軟體故障時具有復原力，您可以將它配置成在硬體故障或停電時具有復原力。 這表示儲存在連結機能中的訊息具有高可用性。

IBM MQ 使用的每一個連結機能清單結構都專用於特定佇列共用群組，但連結機能可以保留多個佇列共用群組的結構。 不同佇列共用群組中的佇列管理程式無法共用資料。 佇列共用群組中最多有 32 個佇列管理程式可以同時連接至連結機能清單結構。

單一連結機能清單結構最多可以包含 512 個共用佇列。 結構中儲存的訊息資料總量受結構容量限制。 不過，使用 **CFLEVEL(5)** 時，您可以使用卸載參數來卸載小於 63 KB 之訊息的資料，從而增加可以儲存在結構中的訊息數，雖然每則訊息仍需要至少一個連結機能項目，以及至少 768 個位元組的資料 (由項目 256 個位元組及標頭及描述子兩個元素 512 個位元組組成)。

清單結構的大小受下列因素限制:

- 它必須位於單一連結機能內。
- 它可能會與 IBM MQ 及其他產品的其他結構共用可用的連結機能儲存體。

連結機能清單結構可以有相關聯的儲存類別記憶體。 在某些情況下，與共用佇列搭配使用時，此儲存類別記憶體可能很有用。 如需相關資訊，請參閱 第 155 頁的『使用具有共用佇列的儲存類別記憶體』。

規劃 CF 結構大小

如果您需要 CF 結構大小的指引，則可以使用 MP16: IBM MQ for z/OS 產能規劃及調整 supportpac。 您也可以使用 IBM 提供的 Web 型工具 CFSizer 來協助 CF 大小。

CF 結構物件

佇列管理程式使用的連結機能結構指定在 CF 結構 (CFSTRUCT) IBM MQ 物件中。

這些結構物件儲存在 Db2 中。

使用與連結機能結構相關的 z/OS 指令或定義時，需要佇列共用群組名稱的前四個字元。 不過， IBM MQ CFSTRUCT 物件一律存在於單一佇列共用群組中，因此其名稱不包括佇列共用群組名稱的前四個字元。 例如，在以 SQ03 開頭的佇列共用群組中定義的 CFSTRUCT (MYDATA) 將使用連結機能清單結構 SQ03MYDATA。

CF 結構具有 CFLEVEL 屬性，可判定其功能:

- 1、2-可用於小於 63 KB 的非持續訊息
- 3-可用於小於 63 KB 的持續及非持續訊息
- 4-可用於持續及非持續訊息，最多 100 MB
- 5-可用於持續及非持續訊息 (最多 100 MB) ，並選擇性地卸載至共用訊息資料集 (SMDS) 或 Db2。

註: 使用 IBM MQ 時，您可以加密連結機能結構。 如需相關資訊，請參閱 加密連結機能結構資料 。

連結機能的備份及回復

您可以使用 IBM MQ 指令 BACKUP CFSTRUCT 來備份連結機能清單結構。 這會將目前在 CF 結構內的持續訊息副本放置到進行備份之佇列管理程式的作用中日誌資料集，並將備份記錄寫入 Db2。

如果連結機能失敗，您可以使用 IBM MQ 指令 RECOVER CFSTRUCT。 這會使用 Db2 中的備份記錄，從 CF 結構的備份中尋找並還原持續訊息。 自前次備份以來的任何活動都會使用佇列共用群組中所有佇列管理程式的日誌來重播，然後 CF 結構會還原到失敗之前的時間點。

如需詳細資料，請參閱 備份 CFSTRUCT 和 回復 CFSTRUCT 指令。

相關概念

第 143 頁的『指定共用訊息的卸載選項』
您可以選擇共用佇列訊息的訊息資料儲存在 Db2 表格或共用訊息資料集 (SMDS) 中的位置。 您也可以根據訊息大小及連結機能結構 (CF) 的現行使用情形，來選取要卸載的訊息。

第 145 頁的『管理共用訊息資料集 (SMDS) 環境』
如果您選取共用訊息資料集來卸載大型訊息，則也必須注意 IBM MQ 用來管理這些資料集的資訊，以及用來使用此資訊的指令。 請利用這個主題來瞭解如何管理共用訊息資料集。

z/OS 指定共用訊息的卸載選項

您可以選擇共用佇列訊息的訊息資料儲存在 Db2 表格或共用訊息資料集 (SMDS) 中的位置。 您也可以根據訊息大小及連結機能結構 (CF) 的現行使用情形，來選取要卸載的訊息。

共用佇列的訊息資料可以從連結機能卸載，並儲存在 Db2 表格或稱為 共用訊息資料集 (SMDS) 的 IBM MQ 受管理資料集中。

對於大於連結機能項目大小 63 KB 的訊息，與卸載至 Db2 相比，將訊息資料卸載至 SMDS 可能會有顯著的效能改善。

仍會使用連結機能結構中的清單項目來管理每個共用佇列訊息，但當訊息資料卸載至 SMDS 時，連結機能項目只會包含部分控制資訊，以及儲存訊息之相關磁碟區塊的參照清單。 使用此機制表示每一個訊息所需的連結機能元素儲存體數量僅為訊息實際大小的一小部分。

選取共用佇列訊息的儲存位置

SMDS 或 Db2 共用訊息儲存體的選擇由 **CFSTRUCT** 定義上的 **OFFLOAD(SMDS|DB2)** 參數控制。 **OFFLOAD(SMDS)** 是預設值。

此參數也需要 **CFSTRUCT** 使用 **CFLEVEL(5)** 或以上。

OFFLOAD 參數僅從 **CFLEVEL(5)** 開始有效。 如需詳細資料，請參閱 DEFINE CFSTRUCT 。

主要基於移轉目的支援 **OFFLOAD(DB2)** 。

選取要卸載哪些共用佇列訊息

根據訊息資料的大小，以及連結機能結構的現行使用情形，將訊息資料卸載至 SMDS 或 Db2 。 有三個規則，每一個規則指定一對相符的參數。 這些參數是對應的連結機能結構用量臨界值百分比 (**OFFLDnTH**) 及訊息大小限制 (**OFFLDnSZ**)。

這三個規則的現行實作是使用下列關鍵字配對來指定:

- OFFLD1TH 和 OFFLD1SZ
- OFFLD2TH 和 OFFLD2SZ
- OFFLD3TH 和 OFFLD3SZ

規則配對	預設值	說明
規則配對 1	OFFLD1TH(70) 和 OFFLD1SZ(32K)	對於超過 32 KB 的訊息，如果連結機能結構超過 70% 完整卸載資料
規則配對 2	OFFLD2TH(80) 和 OFFLD2SZ(4K)	對於超過 4 KB 的訊息，如果連結機能結構超過 80% 完整卸載資料
規則配對 3	OFFLD3TH(90) 和 OFFLD3SZ(0K)	如果連結機能結構超過 90% 完整卸載資料，則訊息超過 0 KB (所有訊息)

如果卸載規則具有 OFFLD *x* SZ 值 64K ，則表示規則未生效。 在此情況下，只有在另一個卸載規則生效時，或訊息大於 63.75 KB 且太大而無法儲存在結構中，才會卸載訊息。

每一個卸載的訊息仍需要連結機能中的 0.75 KB 儲存體。

可針對每一個結構指定的三個卸載規則，預期如下所示使用。

- 效能
  - 當應用程式結構中有大量空間時，只有在訊息資料太大而無法儲存在結構中，或超出某個較低的訊息大小臨界值，使得儲存在結構中的效能值不值得它需要的結構空間量時，才應該卸載訊息資料。
  - 如果需要特定的訊息大小臨界值，通常會使用第一個卸載規則來指定。
- 容量
  - 當應用程式結構中的空間非常小時，應該卸載最大的訊息資料量，以充分利用剩餘空間。
  - 第三個卸載規則通常用來指出當結構幾乎已滿時，應該卸載大部分訊息，因此應用程式結構中的項目通常會達到大小下限 (需要大約 0.75K 個位元組)。
  - 應該根據應用程式結構大小及預期的待辦事項數目上限來選擇使用臨界值參數。 例如，如果預期的待辦事項上限為 1M 個訊息，則此訊息數所需的結構儲存體量大約為 0.75G 個位元組。 例如，如果結構大約是 10G 個位元組，則卸載所有訊息的使用臨界值必須設為 92% 或更低。
  - 結構空間分成元素和項目，即使整體空間可能足夠，其中一個空間可能比另一個空間先用完。 系統會在必要時提供 AUTOALTER 功能來調整比例，但這並不十分敏感，因此實際可用的空間量可能較少。 因此，最好是使用不超過 90% 的最大結構空間，因此在前一個範例中，卸載所有訊息的使用臨界值最好設定為大約 80%。
- 緩衝過渡:

- 由於連結機能結構中剩餘的空間量會減少，因此不想要在效能性質中有較大的突然變更。 也不想要連結機能管理在所使用的一般項目與元素比例中突然臨界值變更。
- 第二個卸載規則通常用來在效能與容量偏誤卸載規則之間提供一些中間緩衝。 當連結機能結構中使用的空間超出中間臨界值時，可以將它設為導致卸載活動大幅增加。 這表示剩餘空間的使用速度會較慢，並讓連結機能自動變更處理程序有更多時間適應較高的使用層次。

如果無法擴充連結機能結構，且需要儲存至少一些預定數目的訊息，則可以視需要修改第三規則，以確保所有訊息的資料卸載以適當的臨界值開始，以確保為該預定數目的訊息保留空間。

例如，如果連結機能結構大小為 4 GB，且預先決定的訊息數目為 1 百萬，則需要 1,000,000 * 0.75 KB，即 768 MB，即 4 GB 的 18.75%。 在此情況下，卸載所有訊息的臨界值需要設定大約 80%，而不是 90%。 這會提供參數 OFFLD3TH(80) 和 OFFLD3SZ(0K)。 其他卸載參數也需要調整。

如果發現卸載非常小的訊息會對效能造成重大影響，但對於較大的訊息則相對影響較小，則可以減少其他規則的使用臨界值，以提早卸載較大的訊息，在需要卸載之前，在結構中為小型訊息留下更多空間。

例如，如果超過 32KB 的訊息經常出現，但卸載這些訊息的經歷時間效能 (從 RMF 統計資料或應用程式效能判定) 與將它們保留在連結機能中的經歷時間效能非常類似，則第一個規則的臨界值可以設為 0% 以卸載所有這類訊息。 這會提供參數 OFFLD1TH(0) 及 OFFLD1SZ(32K)。 同樣地，需要調整其他卸載參數。

如果有許多關於特定中間大小的訊息 (例如 16 KB 和 6 KB)，則變更第二個規則的訊息大小選項可能會很有用，這樣較大的規則會以相當低的使用臨界值卸載，節省大量空間，但較小的規則仍只會儲存在連結機能中。

z/OS 管理共用訊息資料集 *(SMDS)* 環境

如果您選取共用訊息資料集來卸載大型訊息，則也必須注意 IBM MQ 用來管理這些資料集的資訊，以及用來使用此資訊的指令。 請利用這個主題來瞭解如何管理共用訊息資料集。

SMDS 物件

在共用 SMDS 物件中追蹤每一個共用訊息資料集的內容及狀態，可透過佇列共用群組中的任何佇列管理程式來更新。

每一個佇列管理程式都有一個共用訊息資料集，可存取每一個連結機能應用程式結構。 共用訊息資料集由擁有端佇列管理程式名稱 (使用 SMDS 關鍵字指定) 及應用程式結構名稱 (使用 CFSTRUCT 關鍵字指定) 所識別。

註: 定義結構的 SMDS 資料集時，每一個佇列管理程式必須各有一個。

SMDS 物件儲存在陣列中 (群組中每個佇列管理程式有一個項目)，該陣列形成儲存在 Db2 中的對應 CFSTRUCT 物件的延伸。

沒有指令可用來 DEFINE 或 DELETE SMDS 物件，因為它是作為 CFSTRUCT 物件的一部分而建立或刪除，但有一個指令可用來 ALTER 它，以變更個別擁有佇列管理程式的設定。

如需 SMDS 指令的進一步資訊，請參閱 第 153 頁的『SMDS 相關指令』

SMDSCONN 資訊

共用訊息資料集可以處於正常狀態，但一個以上佇列管理程式無法連接至它，例如，因為安全定義或直接存取裝置連線功能有問題。 因此，每一個佇列管理程式都必須追蹤連線狀態，以及每一個共用訊息資料集的可用性資訊，例如，指出它目前是否可以連接至它，如果不是，則為何不可以連接至它。

SMDSCONN 資訊代表與共用訊息資料集的佇列管理程式連線。 至於共用訊息資料集本身，則由擁有共用訊息資料集 (如共用物件本身的 SMDS 關鍵字所指定) 且結合 CFSTRUCT 名稱的佇列管理程式所識別。

沒有可識別連接佇列管理程式的參數，因為定址至特定佇列管理程式的指令只能參照該相同佇列管理程式的 SMDSCONN 資訊。

SMDSCONN 資訊項目會維護在擁有端佇列管理程式的主儲存體中，並在重新啟動佇列管理程式時重建。 不過，如果已明確停止來自個別佇列管理程式的連線，則此資訊也會儲存為對應 CFSTRUCT 或 SMDS 物件中連線陣列的旗標，以便在佇列管理程式重新啟動時持續保存。

狀態及可用性資訊

狀態資訊指出資源或連線的狀態 (例如，是否尚未使用、是否正常使用或是否需要回復)。 通常會使用 STATUS 關鍵字來說明。 可能的值視物件類型而定。

狀態資訊通常會自動更新，例如在使用資源或連線時偵測到錯誤時。 不過，在某些情況下，也可以使用指令來更新狀態，以容許佇列管理程式無法自動判斷正確狀態的情況。

可用性資訊指出是否可以使用資源或連線，且通常主要由狀態資訊決定。 對於共用訊息資料集支援中使用的資源或連線類型，會實作三個可用性層次:

可用
這表示資源可供正常使用。 這不一定表示目前正在使用中 (可以從 STATUS 值來決定)。 對於資料集，如果它需要重新啟動處理，則容許擁有端佇列管理程式開啟它，但其他佇列管理程式必須等待直到資料集回到「作用中」狀態。

因錯誤而無法使用
這表示由於發生錯誤，資源已自動變成無法使用，且在執行某種形式的修復或回復處理之前，預期無法再次使用。 不過，允許在沒有操作員介入的情況下再次嘗試使其可供使用。 將資源標示為已啟用的指令，或以指出回復處理已完成的方式變更狀態的指令，也可以觸發這類嘗試。

資源變成無法使用的原因通常從相關的 STATUS 值中很明顯，但在某些情況下，可能會有其他原因讓資源變成無法使用，在這種情況下，會提供個別的 REASON 值來指出原因。

因為操作員指令而無法使用
這表示指令已明確停用對資源的存取權。 只能使用指令來重新啟用它，才能使它變成可用。

SMDS 可用性

對於共用 SMDS 物件， ACCESS 關鍵字會說明可用性，可能值為 ENABLED、SUSPENDED 及 DISABLED。

可以針對群組中任何佇列管理程式的相關共用物件使用 **RESET SMDS** 指令來更新可用性，以設定 ACCESS (ENABLED) 或 ACCESS (DISABLED)。

如果可用性先前是 ACCESS (SUSPENDED) ，則將它變更為 ACCESS (ENABLED) 會觸發新的嘗試來使用共用訊息資料集，但如果前一個錯誤仍然存在，則可用性會重設回 ACCESS (SUSPENDED)。

SMDSCONN 可用性

對於本端 SMDSCONN 資訊項目， AVAIL 關鍵字會說明可用性，可能值為 NORMAL、ERROR 或 STOPPED。 可以使用定址至特定佇列管理程式的 **START SMDSCONN** 或 **STOP SMDSCONN** 指令來更新可用性，以啟用或停用其連線。

如果可用性先前是 AVAIL (ERROR) ，將它變更為 AVAIL (NORMAL) 會觸發新的嘗試來使用共用訊息資料集，但如果先前的錯誤仍然存在，則可用性會重設回 AVAIL (ERROR)。

共用訊息資料集共用狀態及可用性

在群組內使用共用狀態資訊來管理每一個共用訊息資料集的可用性，可以使用 **DISPLAY CFSTATUS** 指令搭配 TYPE (SMDS) 來顯示此資訊。 這會顯示已針對每一個結構啟動資料集之每一個佇列管理程式的狀態資訊。 每一個資料集都可以處於下列其中一種狀態:

找不到
這表示尚未啟動對應的資料集。 只有在指定特定的佇列管理程式時，才會出現這個狀態，因為當選取所有佇列管理程式時，會跳過未啟動的資料集。

新建
第一次開啟並起始設定資料集，準備使其成為作用中。

ACTIVE
這表示資料集完全可用，且應該由結構的所有作用中佇列管理程式來配置及開啟。

FAILED
這表示資料集完全無法使用 (回復處理除外) ，且必須由所有佇列管理程式關閉並取消配置。

回復中
這表示此資料集正在進行媒體回復 (使用 RECOVER CFSTRUCT)。

已回復
這指出已發出將失敗資料集切回作用中狀態的指令，但需要尚未完成的進一步重新啟動處理程序，因此資料集只能由擁有端佇列管理程式開啟以進行重新啟動處理。

空白
資料集不包含任何訊息。 如果擁有端佇列管理程式在不包含任何訊息的情況下正常關閉資料集，則會將資料集置於此狀態。 當因為已清空應用程式結構 (使用具有 TYPE PURGE 的 **RECOVER CFSTRUCT**，或僅針對不可回復的結構，刪除該結構的前一個實例) 而捨棄前一個資料集內容時，也可以將它置於 EMPTY 狀態。 下次資料集由其擁有的佇列管理程式開啟時，空間對映會重設為空的，且狀態會變更為 ACTIVE。 由於不再需要先前的資料集內容，因此可以將處於此狀態的資料集取代為新配置的資料集，例如，變更空間配置或將它移至另一個磁區。

指令輸出包括啟用回復記載的日期和時間 (如果有的話)，以及資料集失敗的日期和時間 (如果目前不在作用中)。

共用訊息資料集可以透過 **RESET SMDS** 指令進入 FAILED 狀態，或在偵測到下列任何類型的錯誤時自動進入 FAILED 狀態:

- 擁有端佇列管理程式無法配置或開啟資料集。
- 在任何佇列管理程式順利開啟資料集標頭之後，驗證資料集標頭會失敗。
- 當擁有端佇列管理程式正在讀取或寫入資料時，會發生永久 I/O 錯誤。
- 當另一個佇列管理程式從已順利完成開啟處理及驗證的資料集中讀取資料時，會發生永久 I/O 錯誤。

當資料集處於 FAILED 或 INRECOVER 狀態時，無法正常使用，因此如果可用性狀態為 ACCESS (ENABLED)，則會變更為 ACCESS (SUSPENDED)。

如果資料集已進入 FAILED 狀態，但不需要任何媒體回復 (例如，因為資料仍然有效，但儲存裝置暫時離線)，則可以使用 **RESET SMDS** 指令來要求將狀態直接變更為 RECOVERY 狀態。

當資料集進入 RECOVERY 狀態時，不論是在回復處理完成時，還是在 **RESET SMDS** 指令的結果中，一旦重新啟動處理完成，就可以重新使用它。 如果它處於 ACCESS (SUSPENDED) 狀態，則會自動切回 ACCESS (ENABLED) 狀態，這容許擁有佇列管理程式執行重新啟動處理程序。 當重新啟動處理程序完成時，狀態會變更為 ACTIVE，然後所有其他佇列管理程式都可以重新連接至資料集。

共用訊息資料集連線狀態及可用性

每一個佇列管理程式會針對其與群組中其他佇列管理程式所擁有的每一個共用訊息資料集的連線，維護其本端狀態及可用性資訊。 可以使用 **DISPLAY SMDSCONN** 指令來顯示此資訊。

如果它無法存取屬於另一個佇列管理程式且處於 ACTIVE 狀態的共用訊息資料集，它會從自己的觀點將連線標示為無法使用。

如果錯誤明確指出資料集本身有問題，則佇列管理程式也會自動變更共用狀態，以指出資料集現在處於 FAILED 狀態。 不過，如果錯誤可能是環境問題所造成，例如未獲授權開啟資料集，則佇列管理程式會發出錯誤訊息，並將資料集視為無法使用，但不會修改共用資料集狀態。 如果環境錯誤無論如何都是資料集的問題 (例如，它已配置在某些佇列管理程式無法存取的裝置上)，則操作員可以使用指定 STATUS (FAILED) 的 RESET SMDS 指令，以容許在必要時回復或修復資料集。

如果無法建立共用訊息資料集的連線，但資料集似乎是有效的，則可以對擁有端佇列管理程式發出 **START SMDSCONN** 指令來觸發使用它的新嘗試。

如果作業需要暫時終止特定佇列管理程式與資料集之間的連線，但資料集本身未損壞，則可以使用 **STOP SMDSCONN** 指令來關閉並取消配置資料集。 如果資料集在使用中，佇列管理程式會正常關閉它 (雖然該資料集中的任何資料要求都會被拒絕，並傳回回覆碼)。 如果它是擁有的資料集，則佇列管理程式會在 CLOSE 處理期間儲存空間對映，而不需要重新啟動處理。

如果資料集需要暫時從所有佇列管理程式中取出 (例如移動它)，但未損壞，則最好對相關資料集使用 **STOP SMDSCONN** 搭配選項 CMDSCOPE (*)，以先停止使用它的佇列管理程式，因為這將避免在資料集恢復服務時需要重新啟動處理程序。 相反地，如果資料集標示為 FAILED，則會告知佇列管理程式必須立即停止使用它，這表示空間對映將不會儲存且需要透過重新啟動處理來重建。

如果重新啟動佇列管理程式，則將重試存取先前處於 ACCESS (SUSPENDED) 狀態的任何共用訊息資料集。

共用訊息資料集回復記載

基於媒體回復目的，會記載持續共用訊息。 這表示在連結機能結構或共用訊息資料集發生任何失敗之後，只要回復日誌仍然完整，就可以回復訊息。 您也可以從另一個站台的回復日誌重建持續訊息，以進行災難回復。

當訊息資料寫入共用訊息資料集時，寫入資料集的每一個區塊會分別記載在寫入連結機能的訊息項目 (包括資料對映) 之後。 回復程序一律會回復連結機能結構，但不需要回復個別共用訊息資料集，除非資料集狀態為 FAILED，或狀態為 ACTIVE 但資料集標頭記錄不再有效，指出已重建資料集。 如果資料集的狀態為「作用中」且資料集標頭仍然有效，則不會選取資料集進行回復，也不會選取資料集的狀態為 EMPTY，指出在失敗時未儲存任何訊息。

共用訊息資料集備份

當使用 BACKUP CFSTRUCT 來備份應用程式結構中的共用訊息時，會同時備份儲存在共用訊息資料集中的持續訊息的任何資料，以及先前儲存在 DB 中的持續共用訊息的任何資料。

共用訊息資料集回復

如果共用訊息資料集毀損或遺失，則需要將它置於 FAILED 狀態，以停止佇列管理程式使用它，直到修復它為止。 這通常會自動發生，但也可以使用指定 STATUS (FAILED) 的 **RESET SMDS** 指令來完成。

如果共用訊息資料集包含任何持續訊息，則可以使用 RECOVER CFSTRUCT 指令來回復這些訊息。 此指令會先從最近的 BACKUP CFSTRUCT 指令還原該共用訊息資料集的任何持續訊息資料，然後套用自該時間以來所記載的所有變更。 如果自第一次啟動資料集以來未執行任何 **BACKUP CFSTRUCT** 指令，則會將它重設為空，然後套用自啟動以來的所有變更。

如果 CFSTRUCT 內容及所有共用訊息資料集無法使用 (例如在災難回復狀況中)，則可以在單一 **RECOVER CFSTRUCT** 指令中全部回復。

如果共用訊息資料集已損壞，但 CFSTRUCT 的回復不在作用中，或包含最新 BACKUP CFSTRUCT 的日誌無法使用，則無法回復卸載至該資料集的訊息。 在此情況下，可以使用具有參數 TYPE (PURGE) 的 **RECOVER CFSTRUCT** 指令，將共用訊息資料集標示為空，並從具有該資料集中所儲存資料的結構中刪除任何訊息。

當發出 **RECOVER CFSTRUCT** 指令時，共用訊息資料集狀態會從 FAILED 變更為 INRECOVER。 如果回復順利完成，則狀態會自動變更為「已回復」，否則會變更回「失敗」。

當資料集變更為「已回復」狀態時，這會告知擁有端佇列管理程式，它現在可以嘗試開啟資料集並執行重新啟動處理程序。

共用訊息資料集回復及同步點

不論同步點為何，共用訊息資料集回復處理程序都會重新套用所有完整日誌記錄的變更，直到日誌結尾。

如果在同步點內進行變更，則 CFSTRUCT 的重新啟動或回復處理可能會導致取消未確定的要求，因此部分已回復的變更可能不會實際使用，但無論如何都不會對回復它們造成損害。

也可能是未確定的 MQPUT 訊息已寫入結構，但對應的資料可能未寫入資料集或日誌 (因為只有在開始同步點處理時才強制 I/O 完成)。 這是無害的，因為重新啟動處理程序會取消結構中的訊息項目，因此它參照未回復資料的事實並不重要。

共用訊息資料集重新啟動處理

如果 CFSTRUCT 的佇列管理程式連線正常終止，則在資料集關閉之前，佇列管理程式會將每一個共用訊息資料集的可用區塊空間對映寫入資料集內的檢查點區域。 然後可以在連線重新啟動時重新讀取空間對映，前提是 CFSTRUCT 或共用訊息資料集都不需要在下次重新啟動之前進行任何回復處理。

不過，如果佇列管理程式異常終止，或結構或資料集需要任何回復處理程序，則在重新啟動與結構的佇列管理程式連線時，需要其他處理程序來動態重建空間對映。

如果資料集本身不需要回復，則佇列管理程式重新啟動只會掃描結構的現行內容，以尋找現行佇列管理程式所擁有的訊息資料參照，並將空間對映中的相關資料區塊標示為所擁有。 重建空間對映時，其他佇列管理程式可以繼續使用結構，並讀取重新啟動佇列管理程式所擁有的資料。

在回復之後重新啟動共用訊息資料集

如果必須從備份回復共用訊息資料集，則會遺失資料集中儲存的所有非持續訊息，且如果使用 TYPE (PURGE) 回復資料集，則會遺失資料集中儲存的所有訊息。 在回復完成之前，資料集將標示為 FAILED 或 INRECOVER，因此任何從另一個佇列管理程式讀取受影響訊息的嘗試都會傳回錯誤碼，指出資料集暫時無法使用。

當資料集已回復時，狀態會變更為「已回復」，這可讓擁有端佇列管理程式開啟它來進行重新啟動處理，但資料集仍無法供其他佇列管理程式使用。 佇列管理程式重新啟動會掃描結構，以重建任何剩餘訊息的空間對映。 掃描也會檢查是否有遺失資料的訊息，並從結構中刪除它們 (或如果必要的話，將它們標示為遺失，稍後再刪除)。

當此重新啟動掃描完成時，資料集狀態會自動從「已回復」變更為「作用中」，此時其他佇列管理程式可以再次開始使用它。

共用訊息資料集使用情形資訊

DISPLAY USAGE 指令現在也會顯示任何目前開啟之共用訊息資料集的共用訊息資料集空間及緩衝池使用情形的相關資訊。 如果指定新選項 TYPE (SMDS) 或現存選項 TYPE (ALL)，則會顯示此資訊。

共用訊息資料效能及容量考量

監視資料集使用情形

DISPLAY USAGE 指令可以顯示每一個擁有的共用訊息資料集的現行百分比 (含 **TYPE(SMDS)**選項)。

當共用訊息資料集達到 90% 已滿時，佇列管理程式通常會自動擴充共用訊息資料集，但前提是選項 **DSEXPAND(YES)** 對 SMDS 定義有效。 當 SMDS 選項設定為 **DSEXPAND(YES)** 或 SMDS 選項設定為 **DSEXPAND(DEFAULT)** 且 CFSTRUCT 預設選項設定為 **DSEXPAND(YES)**時，此選項適用。

如果擴充嘗試失敗，因為建立資料集時未指定次要配置大小 (提供訊息 IEC070I，原因碼為 203) 佇列管理程式會使用置換次要配置 (大約現行大小的 20%) 來重複擴充要求。

當資料集展開時，新的資料集範圍會格式化為展開處理程序的一部分，這可能需要數十秒甚至數分鐘，以處理非常大的範圍。 在格式化完成且型錄已更新以顯示新的高使用控制間隔之後，新空間變成可供使用。

如果正在非常快速地建立新訊息，則在擴充處理完成之前，現有資料集可能會變滿。 在此情況下，任何無法配置空間的要求都會暫時暫停，直到擴充嘗試完成且新空間可供使用為止。 如果擴充成功，則會自動重試要求。

如果擴充嘗試因缺少可用空間或已達到延伸範圍上限而失敗，則會發出一則訊息，指出失敗的原因，然後受影響 SMDS 的置換選項會自動變更為 **DSEXPAND(NO)**，以防止進一步的擴充嘗試。 在此情況下，資料集可能會變滿，在此情況下可能需要進一步動作，如 資料集已滿中所述。

監視應用程式結構使用情形

可以使用 MVS **DISPLAY XCF,STRUCTURE** 指令來顯示應用程式結構的使用層次，並指定應用程式結構的完整名稱 (包括佇列共用群組字首)。 IXC360I 回應訊息顯示元素及項目的現行用法。

當結構用法超出 CFRM 原則中指定的 **FULLTHRESHOLD** 值時，系統會發出訊息 IXC585E，如果指定的話，系統可能會執行自動 **ALTER** 動作，這可能會變更項目與元素的比例或增加結構大小。

最佳化緩衝池大小

共用緩衝池中的每一個緩衝區都用來讀取或寫入一個訊息 (最多到邏輯區塊大小) 的連續頁面範圍。 如果訊息溢出到進一步的區塊中，則個別區塊中的每一個頁面範圍都需要個別緩衝區。

在寫入或讀取作業之後包含訊息資料的緩衝區會保留在儲存體中，並使用近期最少使用 (LRU) 快取方法來重複使用，以便稍後重新讀取相同資料的要求不需要移至磁碟。 當寫入共用訊息，然後在稍後由在相同系統上執行的應用程式重新讀取時，這會提供重要的最佳化。 如果基於選取目的而瀏覽另一個佇列管理程式所擁有的訊息，則也會避免從磁碟重新讀取訊息。

這表示每一個應用程式結構所需的緩衝區數目是每一個並行 API 要求的一個緩衝區數目，它會讀取或寫入該應用程式結構的大型訊息，以及一些額外的緩衝區數目，這些緩衝區將用來儲存最近存取的資料，以最佳化後續的讀取存取。

對於共用緩衝池，如果緩衝區不足， API 要求只會在緩衝區無法立即可用時等待。 不過，應該避免此狀況，因為它可能會導致效能大幅降低。

共用緩衝池的 **DISPLAY USAGE** 指令統計資料顯示現行統計資料間隔內是否有任何緩衝區等待，也會顯示可用緩衝區的最低數目 (或負值指出隨時等待緩衝區的執行緒數目上限)、已儲存資料的緩衝區數目，以及緩衝區要求在 LRU 鏈結上順利找到已儲存資料的次數百分比 ("LRU 命中數") 而不是必須讀取它 ("LRU 遺失")[1].

- 如果有任何等待，則應該增加緩衝區數目。
- 如果有許多未用的緩衝區，則可能會減少緩衝區數目，以讓區域中有更多儲存體可供其他用途使用。
- 如果有許多包含已儲存資料的緩衝區，但對該已儲存資料的點閱數所佔的比例非常小，則如果儲存體更適合用於其他用途，則緩衝區數目可能會減少。 不過，緩衝區數目不應減少超過可用緩衝區的最低數目，因為這可能會觸發等待，且最好是足夠高，使最低可用緩衝區計數通常遠高於零。

刪除共用訊息資料集

DELETE CFSTRUCT 指令 (只有在結構中的所有共用佇列都是空的且已關閉時才容許) 不會刪除共用訊息資料集本身，但可以在此指令完成之後以一般方式刪除它們。 如果要重複使用相同的資料集作為共用訊息資料集，則必須先將它重新格式化，以將它重設為空狀態。

共用訊息資料集的異常狀況

在正常使用期間，即使沒有軟體或硬體錯誤，也可能會發生一些異常狀況。

資料集已滿

如果資料集已滿但無法展開，或擴充嘗試失敗，使用對應佇列管理程式將大型訊息寫入對應應用程式結構的應用程式會收到錯誤 2192， MQRC_STORAGE_MEDIUM_FULL (也稱為 MQRC_PAGESET_FULL)。

資料集可能因為應用程式中應該處理資料的失敗而變成已滿，導致累積大量訊息待辦事項。 如果是這樣，進一步擴充資料集只會是暫時的解決方案，讓處理應用程式儘快重新開始是很重要的。

如果可以提供更多空間，則可以使用 **ALTER SMDS** 指令來設定 **DSEXPAND(YES)** 或 **DSEXPAND(DEFAULT)** (假設已設定或假設 YES 作為 CFSTRUCT 定義的 **DSEXPAND** 預設值)，以觸發重試。 不過，如果失敗的原因是已達到延伸範圍上限，則會拒絕新的擴充嘗試，並顯示一則訊息，重新設定 **DSEXPAND(NO)** 。 在此情況下，進一步擴充它的唯一方法是重新配置它，這包括讓它暫時無法使用，如下所述。

需要移動或重新配置資料集

如果資料集需要移動或展開，但在正常使用中，則可以暫時停止使用，以容許移動或重新配置。 任何在資料集無法使用時嘗試使用該資料集的 API 要求都會收到原因碼 MQRC_DATA_SET_NOT_AVAILABLE 。

1. 使用 **RESET SMDS** 指令，將資料集標示為 **ACCESS(DISABLED)**。 這會導致它正常關閉，並由所有目前連接的佇列管理程式取消配置。
2. 視需要移動或重新配置資料集，將舊內容複製到新配置的資料集，例如使用 Access Method Services (AMS) **REPRO** 指令。

   在將舊資料複製到新資料集之前，請勿嘗試預先格式化新資料集，因為這會導致將複製的資料附加到格式化資料集的結尾。
3. 再次使用 **RESET SMDS** 指令將資料集標示為 **ACCESS(ENABLED)** ，使其重新使用。

   如果舊內容小於新資料集的大小，則在開啟新資料集時，會自動預先格式化空間的其餘部分。

[1] (Hits / (Hits+Misses))* 100

如果舊內容大於新資料集的大小，則佇列管理程式必須掃描連結機能結構中的訊息，並重建空間對映，以確保未遺失任何作用中資料。 如果找到對新範圍之外的資料區塊的任何參照，則資料集會標示為 **STATUS(FAILED)**，且必須透過將資料集取代為正確大小的資料集，並將舊資料集重新複製到其中，或使用 **RECOVER CFSTRUCT** 來回復任何持續訊息來修復。

連結機能結構空間不足

如果連結機能結構空間不足，導致訊息 IXC585E， 則值得檢查卸載規則是否已設定，以確保在此情況下卸載的資料量上限。 否則，可以使用 **ALTER CFSTRUCT** 指令來修改卸載規則。

共用訊息資料集的錯誤狀況

有許多問題需要注意，這些問題只能由錯誤所造成，在正常作業狀況下不會發生。

無法開啟擁有的資料集

如果擁有共用訊息資料集的佇列管理程式無法配置或開啟它，或不支援資料集屬性，則佇列管理程式會設定適當的 **SMDSCONN** 狀態值 **ALLOCFAIL** 或 **OPENFAIL**， 並將 **SMDSCONN** 可用性設為 **AVAIL(ERROR)**。 它也會將 SMDS 可用性設為 **ACCESS(SUSPENDED)**。 更正錯誤之後，請使用 **RESET SMDS** 指令來設定 **ACCESS(ENABLED)** 以觸發重試，或向擁有端佇列管理程式發出 **START SMDSCONN** 指令。

無法開啟唯讀資料集

如果佇列管理程式無法配置或開啟另一個佇列管理程式所擁有並標示為 **STATUS(ACTIVE)**的共用訊息資料集，它會假設這可能是由於它與資料集 (由 **SMDSCONN** 物件代表) 的連線發生特定問題，而不是資料集本身的問題。

它會適當將 **SMDSCONN** 標示為 **STATUS(ALLOCFAIL)** 或 **STATUS(OPENFAIL)**， 並將 **SMDSCONN** 可用性標示為 AVAIL (ERROR)， 以防止進一步嘗試使用它。

如果可以更正問題而不影響資料集本身的狀態，請使用 **START SMDSCONN** 指令來觸發重試。

如果問題變成資料集本身的問題，則可以使用 **RESET SMDS** 指令將資料集標示為 **STATUS(FAILED)**， 直到回復為止。 當資料集已回復時，將狀態變更回 **STATUS(ACTIVE)** 的動作會導致通知其他佇列管理程式。 如果 **SMDSCONN** 標示為 **AVAIL(ERROR)**，則會自動變更回 **AVAIL(NORMAL)**， 以觸發新的嘗試來開啟資料集。

資料集標頭毀損

如果順利開啟資料集，但標頭資訊的格式不正確，則佇列管理程式會關閉並取消配置資料集，並將狀態集設為 **STATUS(FAILED)**， 並將可用性設為 **ACCESS(SUSPENDED)**。 這可讓 **RECOVER CFSTRUCT** 用來回復內容。

如果因為資料集包含來自其他用途的殘留資料而產生錯誤，且隨後未預先格式化，則請預先格式化資料集，並使用 **RESET SMDS** 指令將狀態變更為 **STATUS(RECOVERED)**。

否則，必須回復資料集。

資料集非預期地空白

如果佇列管理程式開啟標示為 **STATUS(ACTIVE)** 的資料集，但發現它未起始設定或新預先格式化但有效，則佇列管理程式會關閉並取消配置共用訊息資料集，然後將狀態設為 **STATUS(FAILED)**， 並將可用性設為 **ACCESS(SUSPENDED)**。

資料集有永久 I/O 錯誤

如果資料集在 **OPEN** 處理成功之後有永久 I/O 錯誤，則可能需要回復。 佇列管理程式會將資料集標示為 **STATUS(FAILED)**， 以便所有目前連接的佇列管理程式都會關閉並取消配置它。

資料集有可回復的 I/O 錯誤

如果資料集有硬體問題，這可能會導致無法回復的 I/O 錯誤，這些錯誤不會反映回佇列管理程式，但會導致效能大幅降低，而且也會指出在不久的將來發生永久 I/O 錯誤的風險。

在此情況下，可以使用 **RESET SMDS** 指令將資料集標示為 **STATUS(FAILED)**，以離線進行回復。 這會導致所有佇列管理程式關閉並取消配置它，因此例如，它可以在重新變成可用之前移至新磁區。

以這種方式使資料集無法使用時，不會儲存空間對映，因此佇列管理程式連線重新啟動處理程序將需要掃描連結機能結構，以找出資料集中的訊息並重建空間對映，然後才能再次提供資料集。 作為替代方案，如果共用訊息資料集仍可使用，則可以使用 **RESET SMDS** 指令將資料集標示為 **ACCESS(DISABLED)**，讓它更輕地變成無法使用。

資料集內容不正確

佇列管理程式無法直接偵測到資料集包含不正確的資料或不是最新的，例如，因為包含該資料集的磁區必須從備份還原。 不過，它會執行完整性檢查，這使得任何這類錯誤極不可能導致應用程式看到不正確的訊息資料。

為了完整性檢查的目的，在將訊息資料傳遞給使用者程式之前，資料集中的每一個訊息區塊都以對應連結機能項目 ID 的副本作為字首，包括每次讀取訊息區塊時都會檢查的唯一時間戳記。 如果訊息區塊字首不符合項目 ID (且在平均時間內未刪除連結機能項目)，則會假設訊息區塊已損壞且無法使用。

如果損壞的訊息持續存在，則資料集會標示為 **STATUS(FAILED)**，且必須使用 **RECOVER CFSTRUCT** 指令來回復結構內容。 如果損壞的訊息是非持續性，則無法回復它，因此會發出診斷訊息，並刪除對應的連結機能訊息項目。

如果開啟資料集時沒有可用的已儲存空間對映，則會透過掃描連結機能結構來重建資料集中資料的參照。 在此掃描期間，佇列管理程式會執行一些動作:

1. 佇列管理程式會決定資料集中目前剩餘的最新訊息 (如果有的話) 的位置。
2. 然後佇列管理程式會從資料集讀取該訊息，以確保區塊字首符合訊息項目 ID

這些動作可確保佇列管理程式會偵測資料集為舊版的任何案例，並將資料集標示為 FAILED。 不過，此檢查可容忍下列情況: 從先前副本還原資料集，且自那時以來未新增任何新訊息，或自後續讀取並刪除該副本以來所新增的所有訊息。

在正常關閉資料集的情況下，為了防止發生舊版資料，佇列管理程式會執行一些動作:

1. 當正常關閉資料集時，佇列管理程式會將空間對映時間戳記副本儲存在 Db2 內的 SMDS 物件中。
2. 然後，當重新開啟資料集時，佇列管理程式會檢查空間對映時間戳記是否相同

如果時間戳記不符，這表示可能已使用資料集的舊版副本，因此佇列管理程式會忽略現有的空間對映並重新建置它，只有在未實際遺失任何訊息資料時才會成功。

註: 這些完整性檢查不保證在所有理論上可能的情況下偵測舊版或損壞的資料集。 例如，它們將偵測不到訊息區塊開頭有效但其餘資料已部分改寫的情況。

共用訊息資料集的回復實務範例

本節說明共用訊息資料集回復實務範例。

未遺失任何資料的資料集回復

在某些情況下，可以還原失敗資料集的正確內容，而不需要實際回復。 其中一個範例是資料集包含先前使用的殘留資料，且尚未重新預先格式化，可以透過預先格式化來修正。 另一種情況是資料集已移動，但在複製資料的過程中發生錯誤，可以重新正確複製資料來修正此錯誤。

在這種情況下，可以使用 **RESET SMDS** 指令來設定 **STATUS(RECOVERED)**，讓已更正的資料集再次可用。 如果目前可用性是 **ACCESS(SUSPENDED)**，則會自動將它設回 **ACCESS(ENABLED)**。

當通知擁有端佇列管理程式已回復資料集時，它會掃描結構內容以重新建構空間對映，然後將狀態變更為 **STATUS(ACTIVE)**。 然後，其他佇列管理程式可以重新開始讀取資料集。

TYPE (NORMAL) 的資料集回復

如果資料集的內容已遺失，但應用程式結構已使用 **RECOVER(YES)** 定義，且有適當的回復日誌可用，則可以使用 **RECOVER CFSTRUCT** 指令來回復儲存在結構中的任何持續訊息，包括卸載至共用訊息資料集的持續訊息資料。 這個指令會使用 **BACKUP CFSTRUCT** 指令所記載的資訊，加上自備份時間以來對持續訊息所記載的所有變更，來還原現行狀態。

RECOVER CFSTRUCT 指令一律會回復連結機能結構中的所有持續訊息，以及儲存在 Db2 中的卸載訊息資料。 對於儲存在共用訊息資料集中的已卸載資料，只有在每一個資料集已標示為 **STATUS(FAILED)**，或在由回復處理程序開啟時發現非預期的空白或無效時，才會選取該資料集進行回

復處理。 任何標示為作用中且通過驗證檢查的共用訊息資料集都不需要回復，因為現有的訊息資料已正確，但標頭會更新，以指出在回復之後需要重建任何已儲存的空間對映。

只有在結構已標示為失敗時，才能進行回復處理，因為回復處理需要重新建構結構的完整內容。 不過，如果至少有一個共用訊息資料集標示為失敗，必要的話， **RECOVER CFSTRUCT** 指令會自動將結構標示為失敗，以便繼續進行回復處理程序。

可以從佇列共用群組中的任何佇列管理程式執行回復，前提是它已獲得相關資料集的寫入權。

只會備份及記載持續訊息，因此正常回復處理程序會還原所有持續訊息，但會導致結構中的任何非持續訊息遺失。

當回復完成時，已選取進行回復的任何資料集都會自動變更為 **STATUS(RECOVERED)**，且如果可用性為 **ACCESS(SUSPENDED)**，則會變更為 **ACCESS(ENABLED)**。 佇列管理程式會重新建置每一個資料集的空間對映，方法是掃描連結機能中的訊息，然後將資料集標示為 **STATUS(ACTIVE)**，以便再次使用它。

具有 TYPE (PURGE) 的資料集回復

對於可回復結構，如果資料集內容已遺失，但由於某些原因而無法回復 (例如，因為回復日誌無法使用或回復可能花費太長時間)，則 **RECOVER CFSTRUCT** 指令可以與 **TYPE(PURGE)** 搭配使用，讓結構回到可用狀態。 這會將結構重設為空狀態，並將所有相關聯的資料集標示為 **STATUS(EMPTY)**。

刪除應用程式結構

如果使用 MVS **SETXCF FORCE** 指令刪除不可回復的應用程式結構，或由於結構失敗，則下次連接結構時，會發出訊息 CSQE028I，指出已重設結構且捨棄所有現有訊息，且任何現有資料集也會自動重設為 **STATUS(EMPTY)**。 在失去結構或任何相關聯資料集中的資料之後，此動作會讓不可回復的結構再次可用。

如果刪除可回復的應用程式結構，其處理方式與結構失敗的處理方式相同。

資料集回復失敗

如果 **RECOVER CFSTRUCT** 因某些原因而無法完成，例如因為日誌資料集不再可用，或因為佇列管理程式在進行回復時已終止，則任何至少已啟動回復的資料集都會在標頭中標示，以顯示已嘗試局部回復，且資料集將保持 **STATUS(FAILED)** 狀態。

在此情況下，選項是重複原始回復要求，或改為使用 **TYPE(PURGE)** 進行回復，並捨棄現有資料。

如果嘗試將資料集標示為 **STATUS(RECOVERED)** 而未實際回復，則下次開啟佇列管理程式時，會看到標頭指出回復不完整，並再次將它標示為 **STATUS(FAILED)**。

離站災難回復

對於離站災難回復，只能使用日誌及包含 CFSTRUCT 定義及相關聯 SMDS 狀態資訊的 Db2 共用物件來重建持續共用訊息。

在設定包含定義的 Db2 表格之後，應用程式結構及共用訊息資料集可以設定為空白。 當佇列管理程式連接至它們，並發現它們非預期地是空的時，它會將它們標示為失敗，在此之後，可以使用單一 **RECOVER CFSTRUCT** 指令來回復所有受影響結構的所有持續訊息。

z/OS *SMDS* 相關指令

本主題說明並提供共用訊息資料集相關指令的存取權。

顯示並變更與大型訊息卸載 (**OFFLOAD** 及卸載規則) 及共用訊息資料集相關的 **CFSTRUCT** 選項 (**DSGROUP, DSBLOCK, DSBUFS, DSEXPAND**):

- 顯示 CFSTRUCT
- DEFINE CFSTRUCT
- ALTER CFSTRUCT
- DELETE CFSTRUCT

顯示 **CFSTRUCT** 與大型訊息卸載相關的狀態 **(OFFLDUSE)**:

- DISPLAY CFSTATUS

顯示及變更置換資料集選項 (**DSEXPAND** 和 **DSBUFS**) 針對個別佇列管理程式:

- 顯示 SMDS
- ALTER SMDS

顯示或修改佇列共用群組內資料集的狀態及可用性:

- DISPLAY CFSTATUS TYPE (SMDS)
- 重設 SMDS

顯示佇列管理程式的 SMDS 資料集空間使用情形及緩衝區使用情形資訊:

- 顯示使用類型 (SMDS)

顯示或修改連線的狀態及可用性 (**SMDSCONN**) 至個別佇列管理程式中的資料集:

- 顯示 SMDSCONN
- 開始 SMDSCONN
- 停止 SMDSCONN

備份及回復共用訊息，必要時包括 SMDS 中的大型訊息資料:

- 備份 CFSTRUCT
- 回復 CFSTRUCT

z/OS 使用共用佇列的優點

共用佇列容許 IBM MQ 應用程式具有可調式、高可用性，並容許實作工作量平衡。

共用佇列的優點

共用佇列架構 (其中所複製伺服器從單一共用佇列取回工作) 具有一些有用的內容:

- 它可透過新增伺服器應用程式的新實例，甚至新增具有佇列管理程式 (在佇列共用群組中) 及應用程式副本的 z/OS 映像檔，來進行調整。
- 它具有高可用性。
- 它會根據佇列共用群組中每一個佇列管理程式的可用處理容量，自然執行 取回 工作量平衡。

使用共用佇列以取得高可用性

下列範例說明如何使用共用佇列來增加應用程式可用性。

假設有一個 IBM MQ 實務範例，在網路中執行的用戶端應用程式想要對 z/OS 上執行的伺服器應用程式提出要求。 用戶端應用程式會建構要求訊息，並將它放在要求佇列上。 然後，用戶端會等待來自伺服器的回覆，該回覆會傳送至要求訊息的訊息描述子中指名的回覆目的地佇列。

IBM MQ 會管理從用戶端機器到 z/OS 上伺服器輸入佇列的要求訊息傳輸，以及從伺服器到用戶端的回應傳輸。 透過將伺服器的輸入佇列定義為共用佇列，可以在佇列共用群組中的任何佇列管理程式上擷取放入佇列的任何訊息。 這表示您可以在 Sysplex 中每一個 z/OS 映像檔上配置佇列管理程式，並透過將它們全部連接至相同的佇列共用群組，其中任何一個都可以存取伺服器輸入佇列上的訊息。

伺服器輸入佇列上的訊息仍然可用，即使其中一個佇列管理程式異常終止，或您因為管理原因而必須停止它。 您可以讓整個 z/OS 映像檔離線，訊息仍可使用。

若要利用共用佇列上訊息的這種可用性，請在 Sysplex 中每一個 z/OS 映像檔上執行伺服器應用程式的實例，以提供更高的伺服器應用程式容量及可用性，如 第 155 頁的圖 60 中所示。

伺服器應用程式的一個實例會從共用佇列擷取要求訊息，並根據內容來執行其處理，以產生傳回用戶端作為 IBM MQ 訊息的結果。 回應訊息是針對要求訊息的訊息描述子中所指名的回覆目的地佇列及回覆目的地佇列管理程式。

您可以使用一些選項來配置傳回路徑。 如需這些選項的相關資訊，請參閱第 171 頁的『分散式佇列及佇列共用群組』。

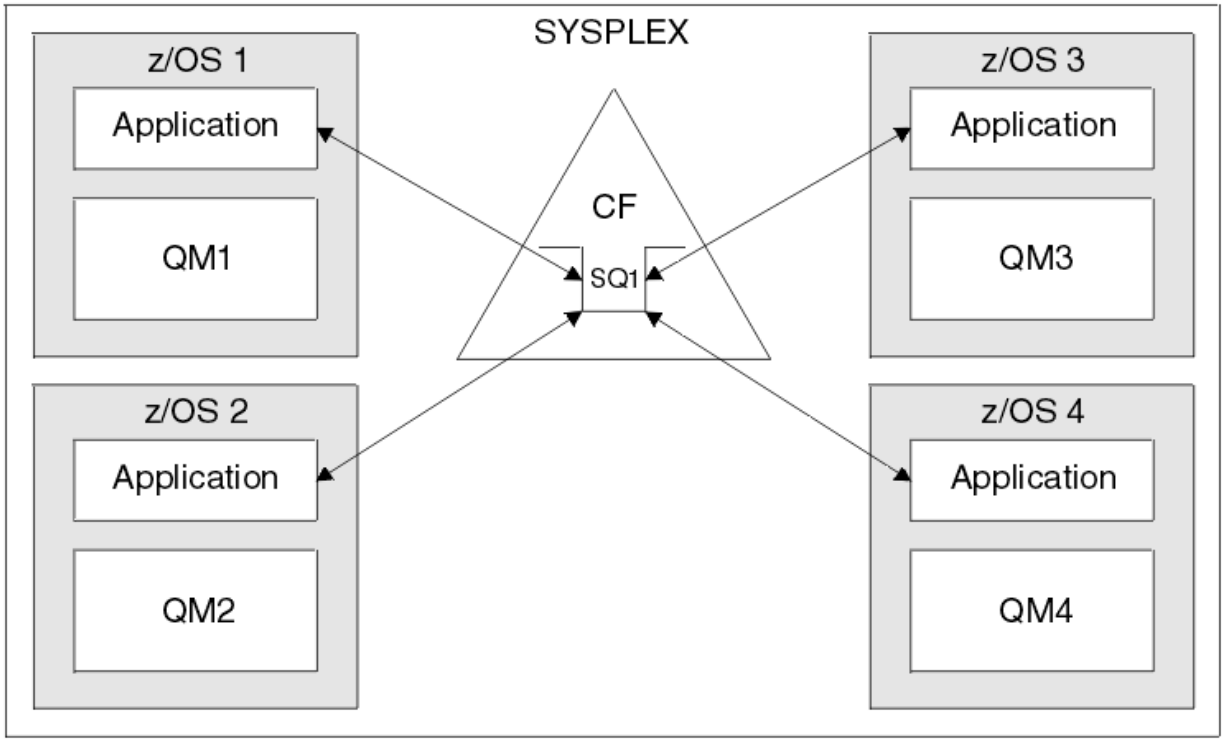


圖 *60:* 處理共用佇列之應用程式的多個實例

同層級回復

為了進一步加強佇列共用群組中訊息的可用性， IBM MQ 會偵測群組中是否有另一個佇列管理程式與連結機能異常中斷連線，並在可能時完成該佇列管理程式仍在擱置中的工作單元。 此特性稱為 同層級回復 。

假設佇列管理程式在應用程式已從同步點的佇列中擷取要求訊息，但尚未放置回應訊息或確定工作單元時，異常終止。 佇列共用群組中的另一個佇列管理程式會偵測到失敗，並取消正在失敗佇列管理程式上執行的進行中工作單元。 這表示要求訊息會放回要求佇列，且可供其中一個其他伺服器實例處理，而不等待失敗的佇列管理程式重新啟動。

如果 IBM MQ 無法自動解析工作單元，您可以手動解析共用部分，讓佇列共用群組中的另一個佇列管理程式繼續處理該工作。

z/OS 使用具有共用佇列的儲存類別記憶體

與 IBM MQ for z/OS 共用佇列一起使用時，使用儲存類別記憶體 (SCM) 會很有利。

重要: IBM z16 計劃成為最後一代 IBM Z ®， 以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

z13、 zEC12 及 zBC12 機器容許安裝 Flash Express 卡。 這些卡包含快閃記憶體固態硬碟 (SSD)。 安裝之後，可以將卡片中的快閃記憶體儲存體配置給一或多個通常稱為 SCM 的 LPAR。

就 I/O 延遲及成本而言， SCM 位於實際儲存體與直接存取儲存裝置 (DASD) 之間。 由於 SCM 沒有移動組件，其 I/O 延遲遠低於 DASD。

SCM 也比實際儲存體更便宜。 因此，可以以相對較低的成本安裝大量儲存體; 例如， 一對 Flash Express 卡包含 1424 GB 可用的儲存體。

這些性質表示當必須在短時間內從實際儲存體取得大量資料時， SCM 非常有用，因為該資料寫入 SCM 的速度比寫入 DASD 的速度快得多。 當使用包含 IBM MQ 共用佇列的連結機能 (CF) 清單結構時，這個特定點非常有用。

為何清單結構填滿

定義 CF 結構時，會以說明結構大小上限的 SIZE 屬性來配置它。 因為 CF 結構一律永久常駐在實際儲存體中，所以 CF 上所定義結構的 SIZE 屬性總和應該小於配置給 CF 的實際儲存體數量。

因此，任何給定結構的 SIZE 值都會維持在可能的最小值，因此有更多結構可以適合 CF。 然而，確保結構足夠大以達到其目的，可能會造成衝突的壓力，因為讓結構太小表示它可能填滿，會干擾使用它的應用程式或子系統。

非常需要根據預期的使用來精確調整結構的大小。 不過，這項作業很難執行，因為工作量可能會隨著時間而變更，而且要考慮到其波動並不容易。

IBM MQ 共用佇列使用 CF 清單結構來儲存訊息。 IBM MQ 會呼叫 CF 結構，其中包含訊息及應用程式結構。

使用儲存在 IBM MQ CFSTRUCT 物件中的資訊來參照應用程式結構。 將小於 63 KB 的訊息放置在共用佇列上時，該訊息會整個儲存在應用程式結構中作為單一清單項目，以及零個以上清單元素。

因為 IBM MQ 共用佇列使用清單結構，所說明的壓力也會影響共用佇列。 在此情況下，可以儲存在共用佇列上的訊息數目上限是下列函數的函數:

- 佇列上訊息的大小
- 結構大小上限
- 結構中可用的項目及元素數目

因為最多 512 個共用佇列可以使用相同的結構，而且有效競爭項目及元素，這甚至會使事件更加複雜

IBM MQ 佇列用於在應用程式之間傳送資料，因此當夥伴應用程式 (應該取得這些訊息) 未執行時，常見的狀況是應用程式將訊息放入佇列。

當發生此情況時，佇列上的訊息數會隨著時間而增加，直到發生下列一或多個狀況為止:

- 放置應用程式會停止放置訊息。
- 取得應用程式開始取得訊息。
- 佇列上的現有訊息會開始到期，並從佇列中移除。
- 佇列達到其深度上限，在此情況下， MQRC_Q_FULL 原因碼會傳回放置應用程式。
- 包含共用佇列的結構達到其大小上限，或包含該結構的 CF 用盡可用的儲存體。 在任一種情況下，都會將 MQRC_STORAGE_MEDIUM_FULL 原因碼傳回給放置應用程式。

在最後三種情況下，佇列已滿。 此時放置應用程式會有問題，因為它的訊息無法傳送。 放置應用程式通常會使用下列一或多個解決方案來解決此問題:

- 反覆重試放置訊息，選擇性地在重試之間延遲。
- 將訊息放置在其他位置，例如資料庫或檔案。 稍後可以存取這些訊息，並如常將訊息放入佇列。
- 如果訊息是非持續性，請捨棄它。

不過，對於某些類別的應用程式，例如具有大量送入訊息或無法存取檔案系統的應用程式，這些解決方案並不實際。 真正需要確保佇列永遠不會填滿，或極不可能填滿，這與共用佇列特別相關。

SMDS 及卸載規則

IBM WebSphere MQ 7.1 中引進的卸載規則提供一種方法，可減少應用程式結構填滿的可能性。

每一個應用程式結構都有三個相關聯的規則，使用三組關鍵字來指定:

- OFFLD1SZ 及 OFFLD1TH
- OFFLD2SZ 及 OFFLD2TH
- OFFLD3SZ 及 OFFLD3TH

每一個規則指定必須符合的條件，訊息資料才能卸載至與應用程式結構相關聯的儲存機制。 目前有兩種儲存機制可用:

- Db2
- 「虛擬儲存體存取方法 (VSAM)」線性資料集的群組， IBM MQ 會呼叫共用訊息資料集 (SMDS)。

下列範例顯示 MQSC 指令，使用 DEFINE CFSTRUCT 指令來建立名為 LIST1 的應用程式結構。

此結構已備妥預設卸載規則，並使用 SMDS 作為卸載機制。 這表示當結構已滿 70% (OFFLD1TH) 時，所有 32 KB 或更大 (OFFLD1SZ) 的訊息都會卸載至 SMDS。

同樣地，當結構已滿 80% (OFFLD2TH) 時，會卸載所有 4 KB 或更大 (OFFLD2SZ) 的訊息。 當結構已滿 90% (OFFLD3TH) 時，會卸載所有訊息 (OFFLD3SZ)。

```
DEFINE CFSTRUCT(LIST1)
CFLEVEL(5)
OFFLOAD(SMDS)
OFFLD1SZ(32K) OFFLD1TH(70)
OFFLD2SZ(4K) OFFLD2TH(80)
OFFLD3SZ(0K) OFFLD3TH(90)
```

卸載訊息儲存在卸載媒體中，訊息的指標儲存在結構中。 雖然卸載規則會減少結構填滿的機會，但在結構中放置較少的訊息資料，因為它用完儲存體，部分資料仍會寫入每一個訊息的結構。 亦即，卸載訊息的指標。

此外，卸載規則還隨附效能成本。 將訊息寫入結構相對快速，且主要由將寫入要求傳送至 CF 所花費的時間所支配。 實際寫入結構的速度很快，以實際儲存速度進行。

將訊息寫入 SMDS 會慢很多，因為它包括寫入訊息指標的結構，以及將訊息資料寫入 SMDS。 這項第二次寫入作業是以 DASD 速度完成，且有可能增加延遲。 如果使用 Db2 作為卸載機制，則效能成本會更高。

z/OS 儲存類別記憶體如何與 *IBM MQ for z/OS* 搭配運作

儲存類別記憶體 (SCM) 與 IBM MQ for z/OS 共用佇列搭配使用的概觀。

重要: IBM z16 計劃成為最後一代 IBM Z®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

CFLEVEL 19 或更高版本的連結機能 (CF) 可以將 SCM 配置給它。 然後可以配置 CF 中定義的結構，以利用 SCM 來減少結構填滿的機會 (稱為結構已滿條件)。 當配置為利用 SCM 的結構填滿超過系統決定點時， CF 會開始將資料從結構移至 SCM，這會釋放結構中的空間給新資料。

註: 因為 SCM 本身可以填滿，所以將 SCM 配置到結構只會減少結構完整條件的可能性，但不會完全排除發生一個情況的機會。

結構配置為使用 SCM，方法是在連結機能資源管理程式 (CFRM) 原則中同時指定 **SCMALGORITHM** 和 **SCMMAXSIZE** 關鍵字，其中包含該結構的定義。

請注意，在指定這些關鍵字並套用 CFRM 原則之後，必須重建或取消配置結構，使其生效。

SCMALGORITHM 關鍵字

因為 SCM 的輸入/輸出速度比實際儲存體的輸入/輸出速度慢，所以 CF 會使用針對預期的結構使用量身訂做的演算法，以減少寫入 SCM 或從 SCM 讀取的影響。

該演算法由結構的 CFRM 原則中的 **SCMALGORITHM** 關鍵字配置，使用 *KEYPRIORITY1* 值。 請注意，您應該只搭配使用 *KEYPRIORITY1* 值與 IBM MQ 共用佇列所使用的清單結構。

KEYPRIORITY1 演算法的運作方式是假設大部分應用程式將以優先順序從共用佇列中取得訊息; 亦即，當應用程式取得訊息時，它會取得具有最高優先順序的最舊訊息。

當結構開始填滿超過系統定義臨界值 90% 時， CF 會開始非同步移轉下一個最不可能取得的訊息。 這些是最近放入佇列的優先順序較低的訊息。

將訊息從結構非同步移轉至 SCM 稱為「預先暫置」。

預先暫置會降低使用 SCM 的效能成本，因為它會減少應用程式在同步輸入/輸出至 SCM 期間遭到封鎖的機會。

除了預先暫置之外， *KEYPRIORITY1* 演算法也會在有足夠可用空間時，以非同步方式從 SCM 帶回訊息，並將訊息帶回結構中。 對於 *KEYPRIORITY1* 演算法，這表示當結構小於或等於 70% 已滿時。

將訊息從 SCM 帶入結構的動作稱為「預先提取」。

預先提取可減少應用程式嘗試取得已預先暫置至 SCM 且在 CF 同步將訊息帶回結構時必須等待的訊息的可能性。

SCMMAXSIZE 關鍵字

SCMMAXSIZE 關鍵字定義結構可使用的 SCM 數量上限。 因為 SCM 在需要時由 CF 配置給結構，所以可以指定大於可用 SCM 總數的 **SCMMAXSIZE** 。 這被稱為 "過度投入"。

重要: 絕不過度確定 SCM。 如果您這麼做，則依賴它的應用程式將無法取得他們預期的行為。 例如，使用共用佇列的 IBM MQ 應用程式可能會取得非預期的 MQRC_STORAGE_MEDIUM_FULL 原因碼。

CF 使用各種資料結構來追蹤其對 SCM 的使用。 這些資料結構位於配置給 CF 的實際儲存體中，因此減少結構可以使用的實際儲存體數量。 這些資料結構所使用的儲存體稱為「擴增空間」。

使用 SCM 配置結構時，會將 CF 中的少量實際儲存體配置給稱為固定擴增空間的結構。 即使結構從未實際使用任何 SCM，也會配置此項。 由於結構中的資料儲存在 SCM 中，因此將從 CF 中的備用實際儲存體配置額外的動態擴增空間。

從 SCM 中移除資料時，會將動態擴增空間傳回 CF。 擴增空間 (固定或動態) 絕不會從配置給結構的實際儲存體中取得。

除了擴增儲存體之外，當結構配置為使用 SCM 時，該結構所使用的控制儲存體數量會增加。 這表示與未配置 SCM 的相同大小結構相比，以 SCM 所配置的清單結構所包含的項目和元素可能較少。

若要瞭解 SCM 對新結構或現有結構的影響，請使用 CFSizer 工具。

最後一個要注意的重要點是，在資料從結構移至 SCM 之後，已使用動態擴增空間，無法手動或自動變更結構。

也就是說，無法增加或減少配置給結構的儲存體數量，無法變更結構所使用的項目與元素比例，依此類推。若要再次使結構可變更，該結構不得在 SCM 中儲存任何資料，且不得使用動態擴增儲存體。

z/OS 為何使用 *SCM*

緊急儲存體和改良的效能是兩個搭配使用 SCM 與 IBM MQ for z/OS 的使用案例。

重要: IBM z16 計劃成為最後一代 IBM Z®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

本節介紹兩種可能情況背後的理論。 如需如何設定實務範例的進一步詳細資料，請參閱:

- 第 161 頁的『緊急儲存體-基本配置』
- 第 166 頁的『改良的效能-基本配置』

重要: 將 SCM 與 CF 結構搭配使用並不取決於任何特定版本的 IBM MQ。 不過，緊急儲存體實務範例僅適用於 IBM WebSphere MQ 7.1 以及更新版本，因為它需要 SMDS 及卸載規則。

緊急儲存體

SMDS 和訊息卸載可以與 SCM 一起使用，以減少在延伸中斷期間將 MQRC_STORAGE_MEDIUM_FULL 原因碼傳回 IBM MQ 應用程式的可能性。

概觀

在應用程式結構上配置單一共用佇列。 放置應用程式會將訊息放到共用佇列中; 取得應用程式會從共用佇列中取得訊息。

在正常執行期間，預期佇列深度接近零，但商業需求指出系統必須能夠容忍取得應用程式的兩小時中斷。 這表示共用佇列必須能夠包含來自放置應用程式的兩小時訊息。

目前，透過使用預設卸載規則及 SMDS 來達成此處理程序，從而最小化結構的大小，同時降低與卸載相關聯的效能成本。

傳送至共用佇列的訊息速率預期在中短期會加倍。 雖然系統能夠容忍兩小時中斷的需求仍然存在，但 CF 中沒有足夠的實際儲存體可用來將結構大小加倍。

因為包含應用程式結構的 CF 位於 zEC12 機器上，所以有可能將足夠的 SCM 與結構相關聯，以儲存足夠的訊息，以便可以容忍兩小時的中斷。

請考量一段時間發生的情況:

1. 一開始，系統處於穩定狀態。 放置及取得應用程式都正常執行，且佇列深度接近或為零。 結果是應用程式結構大部分是空的。
2. 在特定時間，取得應用程式發生非預期的失敗並停止。 放置應用程式會繼續將訊息放置到佇列，且應用程式結構會開始填滿。
3. 在結構達到 70% 已滿之後，符合第一個卸載規則的條件，且大小大於或等於 32 KB 的所有訊息都會卸載至 SMDS。

   如需卸載規則的概觀，請參閱 第 156 頁的『SMDS 及卸載規則』 。
4. 當訊息繼續放置到共用佇列時，結構會繼續填滿 (因為訊息資料儲存在結構中，或因為卸載訊息的指標儲存在結構中)。

   當結構達到 80% 已滿時，會開始套用第二個卸載規則，且會將 4 KB 或以上的訊息卸載至 SMDS。
5. 當結構超過 90% 已滿時，所有訊息都會卸載至 SMDS ，且只會將訊息指標放置在結構中。

   大約此時，預先暫置演算法會開始執行，並開始將資料從結構移至 SCM。 假設佇列上的所有訊息都具有相同的優先順序，則會預先暫置最新的訊息。

   因為現在正在將所有訊息卸載至 SMDS ，所以移入 SCM 的資料不是實際訊息資料，而是 SMDS 上訊息的指標。

   因此，可以儲存在結構組合上的訊息數，以及與結構相關聯的 SCM 和 SMDS 數非常大。

   效能: 在中斷執行的這個階段期間，放置應用程式可能會因為必須寫入 SMDS 而受到效能降低的程度。 在此情況下， SCM 的使用不應成為在效能方面放置應用程式的限制因素。 SCM 提供額外空間來防止結構填滿。
6. 最終取得應用程式會再次可用，且中斷已結束。

   不過，結構仍在使用 SCM。 取得應用程式會開始從佇列中讀取訊息，先取得最舊、最高優先順序的訊息。

   因為這些訊息是在結構開始填滿之前寫的，所以它們完全從結構的實際儲存部分出來。
7. 當結構開始清空時，它會低於預先暫置的作用中臨界值，因此預先暫置會停止。
8. 結構用量會減少到低於卸載規則生效的點，因此訊息不再卸載至 SMDS ，除非它們超過 63 KB。

   大約此時，預先提取演算法會開始將資料從 SCM 移入結構。 由於取得應用程式會依照 SCM 演算法所預期的順序，從佇列中取得訊息，因此在取得應用程式需要訊息之前，會先帶入訊息。

   結果是取得應用程式永不需要等待訊息從 SCM 同步帶入。
9. 當取得應用程式繼續向下移動佇列時，它會開始擷取已卸載至 SMDS 的訊息。
10. 最後，系統再次處於穩定狀態。 沒有訊息儲存在 SCM 或 SMDS 中，且佇列深度接近零。

提高效能

此實務範例說明使用 SCM 來增加可以儲存在共用佇列上的訊息數目，而不會產生使用 SMDS 的效能成本。

說明

在此情況下，放置及取得應用程式會透過儲存在應用程式結構中的共用佇列進行通訊。

放置應用程式通常會在短時間內放置大量訊息時激增執行。 然後，在一段很長的時間內，它完全不會產生任何訊息。

取得應用程式會循序處理每一則訊息，並對每一則訊息執行複雜的處理程序。 因此，除了放置應用程式開始執行的時間之外，大部分時間佇列深度都是零，其中佇列深度會隨著放置訊息的速度比取得訊息的速度更快而開始增加。

佇列深度會增加，直到放置應用程式停止，且取得應用程式有足夠時間處理佇列上的所有訊息。

附註:

1. 在此實務範例中，關鍵因素是效能。 傳送至佇列的訊息一律小於 63 KB ，因此永不需要卸載至 SMDS。
2. 應用程式結構已調整大小，其大小足以包含放置應用程式在單一「快速寄發」中所放置的所有訊息。
3. 卸載規則必須全部停用，以便即使在結構開始填滿時，也不會將訊息卸載至 SMDS。 這是因為與將訊息寫入 SMDS 及從中讀取訊息相關聯的效能成本被視為無法接受。

一段時間後，放置應用程式在激增中傳送的訊息數必須增加數個數量級。 因為取得應用程式必須循序處理每一個訊息，所以佇列上的訊息數會增加到結構填滿的點。

此時，放置應用程式會在放置訊息時收到原因碼 (MQRC_STORAGE_MEDIUM_FULL) ，且放置作業失敗。 當放置應用程式無法將訊息放置到佇列時，它只能短暫地容忍期間。 如果期間太長，則應用程式會結束。

假設您沒有時間或技能來重寫放置應用程式或取得應用程式，這個問題有三個可能的解決方案:

1. 增加應用程式結構的大小。
2. 將卸載規則新增至應用程式結構，以便在佇列開始填滿時將訊息卸載至 SMDS。
3. 建立 SCM 與結構的關聯。

第一個解決方案是快速實作，但 CF 上沒有足夠的實際儲存體可用。

第二個解決方案也可能快速實作，但卸載至 SMDS 的效能影響被認為太重要，無法使用此選項。

第三個解決方案 (將 SCM 與結構相關聯) 提供可接受的成本與效能平衡。

將 SCM 與結構相關聯會導致 CF 中實際儲存體的更高使用率，因為使用已擴增的儲存體來取得作業。 不過，實際儲存體的實際數量將小於第一個選項中使用的數量。

另一個考量是 SCM 的成本。 不過，此成本遠低於實際儲存體。 這些因素結合在一起，使第三個選項比第一個選項更便宜。

雖然第三個選項可能無法像第一個選項一樣執行，但 CF 所使用的預先提取及預先暫置演算法可以結合，讓效能差異可接受，或在某些情況下可忽略。

當然，效能會比使用 SMDS 來卸載訊息好得多。

請考量一段時間發生的情況:

1. 最初，取得應用程式處於作用中狀態，並等待將訊息遞送至共用佇列。 放置應用程式不在作用中，且共用佇列是空的。
2. 在特定時間，放置應用程式會變成作用中，並開始將大量訊息放置到共用佇列。 取得應用程式會開始取得訊息，但佇列深度會快速開始增加，因為取得應用程式比放置應用程式慢。

   因此，應用程式結構會開始填滿。
3. 隨著時間增加，放置應用程式仍在作用中。 應用程式結構最多可填滿大約 90%。

   這是 SCM 預先暫置演算法開始將訊息從結構移至 SCM 時，釋放結構中的空間。

   因為取得應用程式會先從佇列取得最舊、最高優先順序的訊息，所以一律會從結構取得訊息，且不需要等待訊息從中的 SCM 同步取得到結構。
4. 放置應用程式仍在作用中，並將訊息放置到共用佇列。 不過，應用程式永遠不會收到 MQRC_STORAGE_MEDIUM_FULL 原因碼，因為 SCM 中存在足夠空間來儲存所有不符合結構的訊息。
5. 最終，放置應用程式會停止，因為它沒有其他要放置的訊息。

   預先暫置演算法會停止，因為結構低於 90% 使用中，且取得應用程式會繼續處理佇列中的訊息。
6. 當取得應用程式開始釋放結構中的空間時，預先提取演算法會開始將訊息從 SCM 帶回結構。

因為取得應用程式會依照預先提取演算法所預期的順序來處理訊息，所以取得應用程式永不會變成封鎖，等待從 SCM 將訊息資料同步帶入結構。

7. 最後，取得應用程式會處理共用佇列上的所有訊息，並等待直到下一個訊息可用為止。 訊息的結構和 SCM 是空的。

z/OS 緊急儲存體-基本配置

如何在 IBM MQ 上設定緊急儲存體的基本實務範例。

關於這項作業

重要: IBM z16 計劃成為最後一代 IBM Z ®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

SMDS 和訊息卸載可以與 SCM 一起使用，以減少在延伸中斷期間將 MQRC_STORAGE_MEDIUM_FULL 原因碼傳回 IBM MQ 應用程式的可能性。

例如，您的企業具有將訊息放入佇列的應用程式，以及從佇列取得訊息的應用程式。 在正常執行期間，您預期佇列深度接近零，但商業需求指出系統能夠容忍取得訊息的應用程式中斷兩小時。

這表示所使用的共用佇列必須能夠包含來自放置應用程式的兩小時訊息。 目前您使用預設卸載規則及 SMDS 來達成此目的。

您預期傳送至共用佇列的訊息速率在中短期為兩倍。 雖然您要求系統仍可容忍兩小時中斷，但 CF 中可用的實際儲存體不足，無法將結構大小加倍。 因為包含應用程式結構的 CF 位於 zEC12 機器上，所以您能夠將足夠的 SCM 與儲存足夠訊息的結構相關聯，以便可以容忍兩小時中斷

此起始實務範例使用:

- 包含單一佇列管理程式 CSQ3 的佇列共用群組 IBM1。 除了管理結構之外，佇列共用群組還定義了單一應用程式結構 SCEN1。
- 連結機能 (CF) CF01，其中 SCEN1 應用程式結構儲存為 IBM1SCEN1 結構。 此結構的大小上限為 1 GB。
- 應用程式結構使用的單一共用佇列 SCEN1.Q 。

此配置在 第 161 頁的圖 61 中說明。

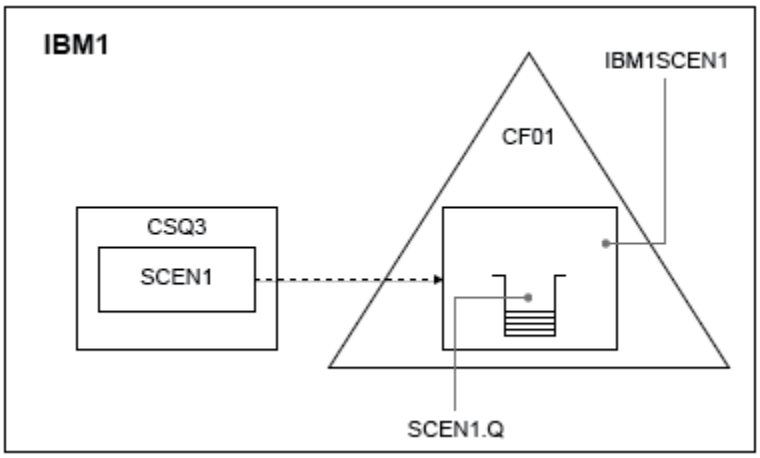


圖 *61:* 基本配置

此外，假設佇列管理程式 CSQ3 已是佇列共用群組 IBM1 的唯一成員。

您必須將結構 IBM1SCEN1 的定義新增至連結機能資源管理程式 (CFRM) 原則。 為了簡單起見，會定義結構，以便透過指定 PREFLIST(CF01)，只能在單一連結機能 CF01 中建立該結構。

小心: 若要容許正式作業系統中的高可用性，您應該針對 IBM MQ 所使用的任何結構，在 PREFLIST 中至少包含兩個 CF。

程序

1. 使用下列指令來重新整理 CFRM 原則:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN1
```

   結構 IBM1SCEN1 的 CFRM 原則範例:

```
STRUCTURE
NAME(IBM1SCEN1)
SIZE(1024M)
INITSIZE(512M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

2. 使用下列指令，驗證已正確建立結構:

```
D XCF,STR,STRNAME=IBM1SCEN1
```

   此時，您的結構尚未配置給佇列共用群組 (如 STATUS 行所示)。
3. 配置 IBM MQ 以使用 CFRM 原則中定義的結構。
   a. 使用 DEFINE CFSTRUCT 指令及 SCEN1 結構名稱來建立 IBM MQ CFSTRUCT 物件:

```
DEFINE CFSTRUCT(SCEN1)
CFCONLOS(TOLERATE)
CFLEVEL(5)
DESCR('Structure for SCM scenario 1')
RECOVER(NO)
RECAUTO(YES)
OFFLOAD(DB2)
OFFLD1SZ(64K) OFFLD1TH(70)
OFFLD2SZ(64K) OFFLD2TH(80)
OFFLD3SZ(64K) OFFLD3TH(90)
```

      .
   b. 使用 DISPLAY CFSTRUCT 指令來驗證結構。
   c. 使用下列 MQSC 指令，定義 SCEN1.Q 共用佇列，以使用 SCEN1 結構:

```
DEFINE QLOCAL(SCEN1.Q) QSGDISP(SHARED) CFSTRUCT(SCEN1) MAXDEPTH(999999999)
```

4. 使用 IBM MQ Explorer ，將單一訊息放入佇列 SCEN1.Q ，並再次將訊息關閉。
5. 發出下列指令，以確認現在已配置結構:

```
D XCF,STR,STRNAME=IBM1SCEN1
```

   移入指令的輸出， STATUS 行會顯示 ALLOCATED。

結果

您已建立基本配置。 現在，您可以使用您選取的任何方法來取得配置的基準線效能的構想。

下一步

將 SMDS 和 SCM 新增至起始結構

相關概念

第 155 頁的『使用具有共用佇列的儲存類別記憶體』

與 IBM MQ for z/OS 共用佇列一起使用時，使用儲存類別記憶體 (SCM) 會很有利。

z/OS 將 *SMDS* 和 *SCM* 新增至起始結構

如何在 IBM MQ 上為緊急儲存體新增 SMDS 及 SCM。

關於這項作業

重要: IBM z16 計劃成為最後一代 IBM Z®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

此部分作業使用 第 161 頁的『緊急儲存體-基本配置』中說明的基本配置。 此實務範例說明新增共用訊息資料集 (SMDS)，然後將 SCM 新增至起始結構。

這項最終配置在 第 163 頁的圖 62 中說明。

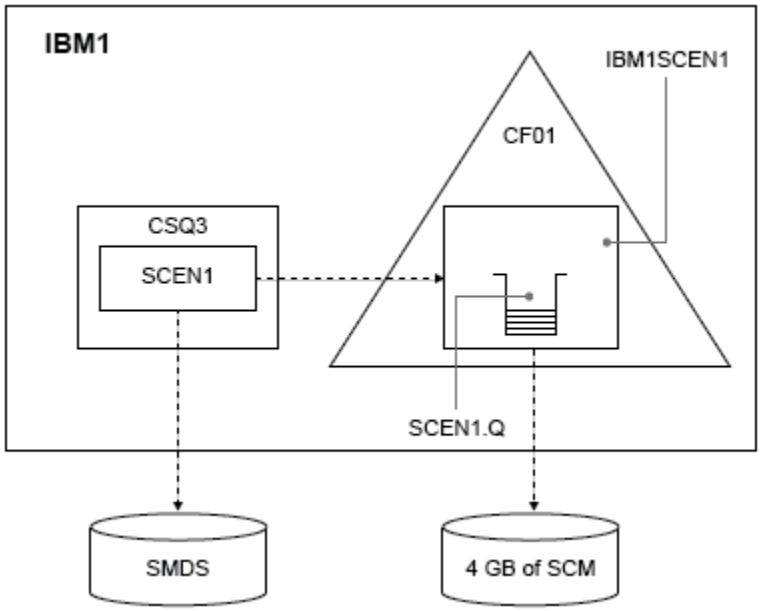


圖 *62:* 為緊急儲存體新增 *SMDS* 和 *SCM* 的配置

程序

1. 編輯 **CSQ4SMDS** 範例 JCL，以建立 SCEN1 應用程式結構使用的 SMDS 資料集，如下所示:

```
//CSQ4SMDS JOB NOTIFY=&SYSUID
//*
//* Allocate SMDS
//*
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE CLUSTER           -
(NAME(CSQSMDS.SCEN1.CSQ3.SMDS) -
MEGABYTES(5000 3000)     -
LINEAR             -
SHAREOPTIONS(2 3) )      -
DATA                -
(NAME(CSQSMDS.SCEN1.CSQ3.SMDS.DATA) )
/*
//*
//* Format the SMDS
//*
//FORM   EXEC PGM=CSQJUFMT,COND=(0,NE),REGION=0M
//STEPLIB DD DSN=MQ800.SCSQANLE,DISP=SHR
//     DD DSN=MQ800.SCSQAUTH,DISP=SHR
//SYSUT1  DD DISP=OLD,DSN=CSQSMDS.SCEN1.CSQ3.SMDS
//SYSPRINT DD SYSOUT=*
```

2. 發出 ALTER CFSTRUCT 指令來變更 SCEN1 應用程式結構，以使用 SMDS 來卸載，並實作預設卸載規則:

```
ALTER CFSTRUCT(SCEN1) OFFLOAD(SMDS) OFFLD1SZ(32K) OFFLD2SZ(4K) OFFLD3SZ(0K)
DSGROUP('CSQSMDS.SCEN1.*.SMDS') DSBLOCK(1M)
```

請注意下列項目:

- 因為 SCEN1.Q 是 SCEN1 應用程式結構上的唯一共用佇列，所以 **DSBLOCK** 值已設為 1M，這是可能的最大值。 這應該是我們的情境中最有效率的設定。
- 因為放置應用程式所傳送的訊息是 30 KB，所以在符合第二個卸載規則之前，當結構已滿 80% 時，才會開始卸載至 SMDS。

3. 重新執行您的測試應用程式。

   請注意佇列上訊息的儲存體已增加。

4. 透過執行下列程序，將 4 GB SCM 新增至結構 IBM1SCEN1 :

   a) 發出下列指令，檢查有多少 SCM 已安裝並配置給 CF01:

   ```
   D CF,CFNAME=CF01
   ```

   b) 檢查所顯示輸出的 STORAGE CONFIGURATION 區段中的 STORAGE-CLASS MEMORY 圖，以查看可用的儲存體。

   c) 使用 SCMMAXSIZE 及 SCMALGORITHM 關鍵字來更新 CFRM 原則，如下所示:

   ```
   STRUCTURE
   NAME(IBM1SCEN1)
   SIZE(1024M)
   INITSIZE(512M)
   ALLOWAUTOALT(YES)
   FULLTHRESHOLD(85)
   PREFLIST(CF01)
   ALLOWREALLOCATE(YES)
   DUPLEX(DISABLED)
   ENFORCEORDER(NO)
   SCMMAXSIZE(4G)
   SCMALGORITHM(KEYPRIORITY1)
   ```

5. 發出下列指令來啟動 CFRM 原則:

   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=polname
   ```

6. 重建 IBM1SCEN1 結構。

   您必須執行此程序，因為在您進行先前變更時已配置結構。

   發出下列指令以重建結構:

   ```
   SETXCF START,REBUILD,STRNM=IBM1SCEN1
   ```

結果

您已順利將 SCM 新增至配置。

下一步

將系統效能最佳化。 如需相關資訊，請參閱 第 164 頁的『最佳化儲存類別記憶體用量』。

z/OS 最佳化儲存類別記憶體用量

如何改善儲存類別記憶體 (SCM) 的使用。

重要: IBM z16 計劃成為最後一代 IBM Z®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

請執行下列指令:

```
D XCF,STR,STRNAME=IBM1SCEN1
```

由於結構已滿載訊息資料，由於先前的測試，部分重建涉及將結構中的部分訊息預先暫置到 SCM 中。 已使用前一個指令來起始此處理程序。

此指令的輸出會產生，例如:

```
ACTIVE STRUCTURE
----------------
ALLOCATION TIME: 06/17/2014 09:28:50
CFNAME : CF01
COUPLING FACILITY: 002827.IBM.02.00000000B8D7
PARTITION: 3B CPCID: 00
STORAGE CONFIGURATION ALLOCATED MAXIMUM %
ACTUAL SIZE: 1024 M 1024 M 100
AUGMENTED SPACE: 3 M 142 M 2
STORAGE-CLASS MEMORY: 88 M 4096 M 2
ENTRIES: 120120 1089536 11
ELEMENTS: 240240 15664556 1
SPACE USAGE IN-USE TOTAL %
ENTRIES: 84921 219439 38
ELEMENTS: 2707678 3149050 85
EMCS: 2 282044 0
LOCKS: 1024
SCMHIGHTHRESHOLD : 90
SCMLOWTHRESHOLD : 70
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CD5186A0 2BD8B85C
LOGICAL VERSION: CD515C50 CE2ED258
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME : IXCLO053
DISPOSITION : KEEP
ACCESS TIME : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS : 1
CONNECTION NAME ID VERSION  SYSNAME JOBNAME  ASID STATE
--------------- -- --------  ------- -------  ---- ----------------
CSQEIBM1CSQ301  01 00010059 SC61    CSQ3MSTR 0091 ACTIVE
```

請從指令輸出中注意下列事項:

- 該 STORAGE_CLASS MEMORY 可讓您確認已將 4096 MB 的 **MAXIMUM** 新增至結構。
- 用於預先暫置的 STORAGE-CLASS MEMORY 數量的 ALLOCATED 圖。 現在，結構中有可用空間，在新增 SCM 之前，沒有可用空間。
- 用來追蹤 SCM 使用情形的 AUGMENTED SPACE 數量。
- 當結構 90% 已滿時，預先暫置演算法開始將資料從結構移至 SCM 的點。 這由不可配置的 **SCMHIGHTHRESHOLD** 內容指出。
- 當結構 70% 已滿時，預先提取演算法開始將資料從 SCM 移至結構以下的點。 這由不可配置的 **SCMLOWTHRESHOLD** 內容指出。

現在您可以測試各種方式來最佳化 SCM 的使用。 請注意下列項目:

- 使用 SCM 來儲存訊息之後，除非您已從 SCM 中移除所有資料，否則無法變更結構。

  在此情況下，這表示項目與元素的比例會凍結在第一次使用 SCM 時的原值。 在預先暫置演算法開始將資料移入 SCM 之前，您必須小心確定結構處於您想要的狀態。
- 在使用 SCM 之前，現行結構大小是否正確?

  例如，您是否已將 **INITSIZE** 從 512 MB 增加到 1 GB 的 SIZE?

  如果您不這麼做，雖然您已啟用結構進行自動變更，但在變更有機會開始之前，預先暫置演算法將會開始將資料移至 SCM。 因此，會使用 512 MB 實際儲存體來凍結結構。
- 在使用 SCM 之前，項目與元素的比例是否正確?

  此實務範例的目標是增加可整體儲存在結構及 SCM 中的已卸載訊息指標數目，以及將盡可能多的訊息完全保留在結構儲存體中。 存取這些訊息比存取 SMDS 上的訊息更快。

  因此，您需要有一個結構，從有利於儲存訊息的項目與元素比例開始，然後在第一次啟動預先暫置演算法之前，轉移至有利於儲存訊息指標的比例。 在某種程度上，可以利用 IBM MQ 卸載規則來達成這項轉移。

發出下列指令來變更卸載規則:

```
ALTER CFSTRUCT(SCEN1) OFFLD1SZ(0K)
```

您可能必須執行數個執行，以最佳化項目與元素比例。

下表顯示在緊急儲存體實務範例的不同階段期間，放置在佇列上的訊息數可能的改進。

表 *18:* 比較緊急儲存體實務範例的結果

測試說明	**訊息數**	**填入佇列的時間 (秒)**
基本配置	27,850	3.2
具有預設卸載規則的 SMDS	205,000	158
具有預設卸載規則的 SCM	828,610	469
具有已調整卸載規則的 SCM	1,135,775	679

表格中的最後一列顯示調整卸載規則具有必要的效果。

您需要檢查您的系統，以查看您是否可以以任何方式改善這些數字。 例如，您可能沒有可用的 SMDS 儲存體。 如果您可以配置更多 SMDS 儲存體，則應該可以大幅增加佇列上的訊息數目。

z/OS 改良的效能-基本配置

如何設定基本實務範例，以在 IBM MQ 上使用共用佇列來改良效能。

關於這項作業

重要: IBM z16 計劃成為最後一代 IBM Z ®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

此實務範例說明如何使用 SCM 來增加可以儲存在共用佇列上的訊息數目，而不會產生使用 SMDS 的效能成本。

此起始實務範例非常類似於用於緊急儲存體的實務範例，並使用:

- 包含單一佇列管理程式 CSQ3 的佇列共用群組 IBM1。 除了管理結構之外，佇列共用群組還定義了單一應用程式結構 SCEN2。
- 連結機能 (CF) CF01，其中 SCEN2 應用程式結構儲存為 IBM1SCEN2 結構。 此結構的大小上限為 2 GB。
- 單一共用佇列 SCEN2.Q，配置為使用應用程式結構。

此配置在 第 166 頁的圖 63 中說明。

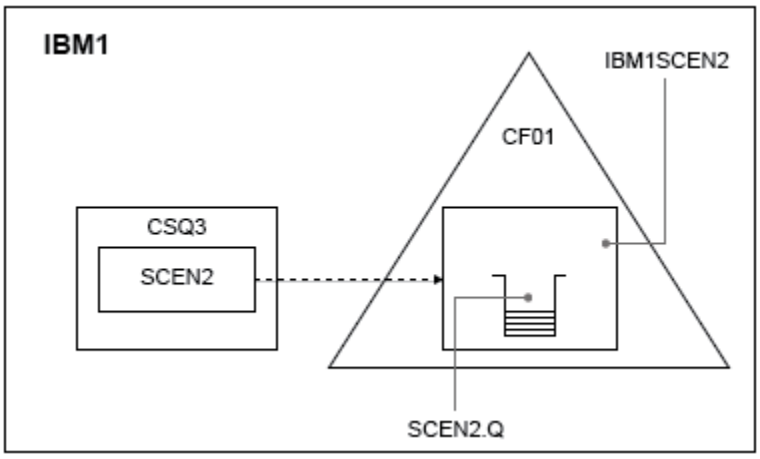


圖 *63:* 基本配置

此外，假設佇列管理程式 CSQ3 已是佇列共用群組 IBM1 的唯一成員。

您必須將結構 IBM1SCEN2 的定義新增至連結機能資源管理程式 (CFRM) 原則。 為了簡單起見，會定義結構，以便透過指定 PREFLIST(CF01)，只能在單一連結機能 CF01 中建立該結構。

結構 IBM1SCEN2 的 CFRM 原則範例:

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

INITSIZE 及 **SIZE** 關鍵字都具有值 2048M，因此結構無法調整大小。

程序

1. 使用下列指令來重新整理 CFRM 原則:

   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
   ```

2. 使用下列指令，驗證已正確建立結構:

   ```
   D XCF,STR,STRNAME=IBM1SCEN2
   ```

   發出前述指令會提供下列輸出:

   ```
   RESPONSE=SC61
   IXC360I 07.58.51 DISPLAY XCF 581
   STRNAME: IBM1SCEN2
   STATUS: NOT ALLOCATED
   POLICY INFORMATION:
   POLICY SIZE : 2048 M
   POLICY INITSIZE: 2048 M
   POLICY MINSIZE : 1536 M
   FULLTHRESHOLD : 85
   ALLOWAUTOALT : YES
   REBUILD PERCENT: N/A
   DUPLEX : DISABLED
   ALLOWREALLOCATE: YES
   PREFERENCE LIST: CF01
   ENFORCEORDER : NO
   EXCLUSION LIST IS EMPTY

   EVENT MANAGEMENT: MESSAGE-BASED   MANAGER SYSTEM NAME: SC53
   MANAGEMENT LEVEL : 01050107
   ```

   此時，您的結構尚未配置給佇列共用群組 (如 STATUS 行所示)。

3. 配置 IBM MQ 以使用 CFRM 原則中定義的結構。

   a. 使用 DEFINE CFSTRUCT 指令，並以 SCEN2 結構名稱來建立 IBM MQ CFSTRUCT 物件。

   ```
   DEFINE CFSTRUCT(SCEN2)
   CFCONLOS(TOLERATE)
   CFLEVEL(5)
   DESCR('Structure for SCM scenario 2')
   RECOVER(NO)
   RECAUTO(YES)
   OFFLOAD(DB2)
   OFFLD1SZ(64K) OFFLD1TH(70)
   OFFLD2SZ(64K) OFFLD2TH(80)
   OFFLD3SZ(64K) OFFLD3TH(90)
   ```

b. 使用 DISPLAY CFSTRUCT 指令檢查結構。

c. 使用下列 MQSC 指令，定義 SCEN2.Q 共用佇列，以使用 SCEN2 結構:

```
DEFINE QLOCAL(SCEN2.Q) QSGDISP(SHARED) CFSTRUCT(SCEN2) MAXDEPTH(999999999)
```

4. 使用「IBM MQ 探險家」將單一訊息放入佇列 SCEN2.Q，並再次將訊息關閉。
5. 發出下列指令，以確認現在已配置結構:

```
D XCF,STR,STRNAME=IBM1SCEN2
```

檢閱指令的輸出 (部分會顯示其中的部分)，並確保 STATUS 行顯示 ALLOCATED。

```
RESPONSE=SC61
IXC360I 08.31.27 DISPLAY XCF 703
STRNAME: IBM1SCEN2
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
POLICY SIZE : 2048 M
POLICY INITSIZE: 2048 M
POLICY MINSIZE : 1536 M
FULLTHRESHOLD : 85
ALLOWAUTOALT : YES
REBUILD PERCENT: N/A
DUPLEX : DISABLED
ALLOWREALLOCATE: YES
PREFERENCE LIST: CF01
ENFORCEORDER : NO
EXCLUSION LIST IS EMPTY
```

此外，請注意 SPACE USAGE 區段中欄位的值:

- 項目
- 元素
- EMCS
- LOCKS

這些值的範例如下:

```
SPACE USAGE   IN-USE      TOTAL    %
ENTRIES:      344686     345242   99
ELEMENTS:    6548455    6548467   99
EMCS:              2     780318    0
LOCKS:          1024
```

結果

您已建立基本配置。 現在，您可以使用您選取的任何方法來取得配置的基準線效能的構想。

下一步

您應該測試基本實務範例。 例如，您可以使用下列三個應用程式，以顯示的順序啟動應用程式，並同時執行它們。

1. 使用 PCF 應用程式來要求現行深度 (**CURDEPTH**) 每 5 秒 SCEN2.Q 的值。 輸出可用來繪製一段時間內佇列的深度。
2. 單一執行緒取得應用程式會使用具有無限等待的 get，反覆地從 SCEN2.Q 取得訊息。 為了模擬處理已移除的訊息，取得中的應用程式會針對它移除的每 10 則訊息暫停 4 毫秒。
3. 單一執行緒放置應用程式會將總計 100 萬個 4 KB 非持續訊息放置到 SCEN2.Q。 此應用程式不會在放置每一個訊息之間暫停，因此訊息放置在 SCEN2.Q 上的速度會比取得應用程式取得訊息的速度更快。

因此，當放置應用程式執行時， SCEN2.Q 的深度會增加。

當結構 IBM1SCEN2 已填滿，且放置應用程式收到 MQRC_STORAGE_MEDIUM_FULL 原因碼時，放置應用程式會休眠 5 秒，然後再嘗試將下一則訊息放置到佇列中。

您可以在一段時間內繪製 CURDEPTH 應用程式的結果。 您會取得某種鋸齒波輸出形式，因為放置應用程式會暫停，以容許佇列部分清空。

移至 第 169 頁的『將 SCM 新增至起始結構』。

相關概念

第 155 頁的『使用具有共用佇列的儲存類別記憶體』

與 IBM MQ for z/OS 共用佇列一起使用時，使用儲存類別記憶體 (SCM) 會很有利。

z/OS 將 *SCM* 新增至起始結構

如何新增 SCM 以改善 IBM MQ 上的效能。

關於這項作業

重要: IBM z16 計劃成為最後一代 IBM Z®，以支援將虛擬快閃記憶體 (也稱為儲存類別記憶體或 SCM) 用於「連結機能」映像檔。 如需相關資訊，請參閱: IBM Z 及 IBM LinuxONE 4Q 2023 方向陳述式。

作為替代方案，您應該使用較大的結構或將訊息卸載至 SMDS。

此部分作業使用 第 166 頁的『改良的效能-基本配置』中說明的基本配置。 此實務範例說明將 SCM 新增至起始結構。

這項最終配置在 第 169 頁的圖 64 中說明。

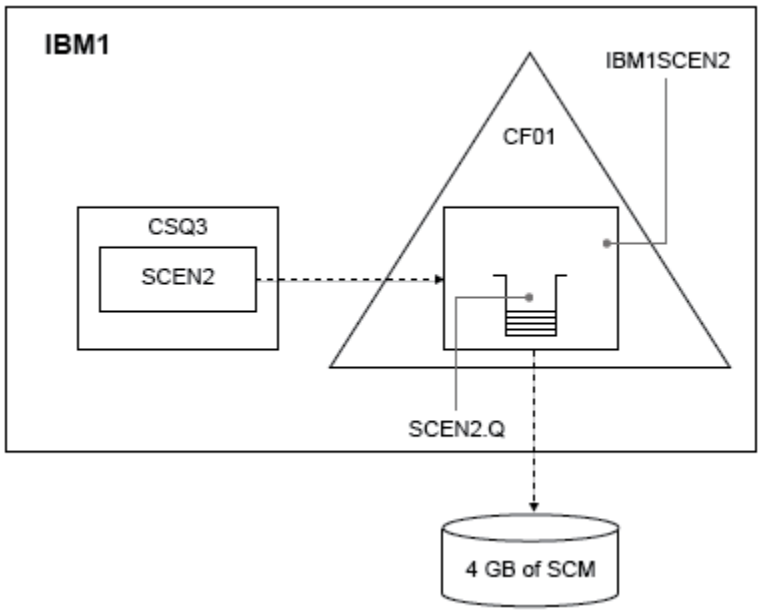


圖 *64:* 配置新增 *SCM* 以提高效能

程序

1. 透過執行下列程序，將 4 GB SCM 新增至結構 IBM1SCEN2 :
   a) 發出下列指令，檢查有多少 SCM 已安裝並配置給 CF01:

```
D CF,CFNAME=CF01
```

   b) 檢查所顯示輸出的 STORAGE CONFIGURATION 區段中的 STORAGE-CLASS MEMORY 圖，以查看可用的儲存體。
   c) 使用 SCMMAXSIZE 及 SCMALGORITHM 關鍵字來更新 CFRM 原則，如下所示:

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
SCMMAXSIZE(4G)
SCMALGORITHM(KEYPRIORITY1)
```

2. 發出下列指令來啟動 CFRM 原則:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
```

3. 重建 IBM1SCEN2 結構。
   您必須執行此程序，因為在您進行先前變更時已配置結構。
   發出下列指令以重建結構:

```
SETXCF START,REBUILD,STRNM=IBM1SCEN2
```

4. 發出下列指令，以確認結構的新配置:

```
D XCF,STR,STRNAME=IBM1SCEN2
```

   檢閱指令的輸出，其中一部分如下:

```
SPACE USAGE     IN-USE       TOTAL      %
ENTRIES:          33        342684      0
ELEMENTS:         48       6503697      0
EMCS:              2        575600      0
LOCKS:                        1024
```

結果

透過增加使用 SCM 所需的控制儲存體來計算實際儲存體使用的變更。

- 在 SCM 新增至結構之前，結構會有下列總計，如 第 166 頁的『改良的效能-基本配置』所示:
  - 345 242 個項目
  - 6,548,467 個要素
  - 780,318 EMCS
- 將 SCM 新增至結構之後，結構會有下列總計:
  - 342,684 個項目
  - 6,503,697 個元素
  - 575,600 EMCS

使用這些圖，在新增 SCM 之後，結構大小會減少下列項目:

- 2558 個項目
- 44,770 個元素
- 204,718 EMCS

用於管理 SCM 的結構儲存體數量，對於已配置 4 GB SCM 的 2 GB 結構，如下所示:

```
(2558 + 44,770 + 204,718) * 256 = 61.5 MB
```

請注意，新增更多 SCM 可能只會達到結構大小的邊際縮減，因為用來追蹤 SCM 的控制儲存體數量會隨著結構大小及配置的 SCM 數量增加而增加。

下一步

重複 第 166 頁的『改良的效能-基本配置』的最後一節中說明的測試。

您可以繪製一段時間內已修訂應用程式的結果。 將圖形與先前取得的圖形相比較，您現在會取得沒有鋸齒波的輸出，因為放置應用程式不再需要等待佇列部分清空。

如需相關資訊，請參閱 MP16: WebSphere MQ for z/OS -產能規劃與調整。

z/OS 分散式佇列及佇列共用群組

分散式佇列及佇列共用群組是兩種可用來增加應用程式系統可用性的技術。 請利用這個主題來尋找這些技術的進一步相關資訊。

為了補充共用佇列上訊息的高可用性， IBM MQ 的分散式佇列元件具有下列其他功能:

- 網路的更高可用性。
- 已增加佇列共用群組之入埠網路連線的容量。

第 171 頁的圖 65 說明分散式佇列及佇列共用群組。 它會顯示 Sysplex 內的兩個佇列管理程式，兩者都屬於相同的佇列共用群組。 他們都可以存取共用佇列 SQ1。 網路中的佇列管理程式 (例如，在 AIX 及 Windows 上) 可以透過任一佇列管理程式的通道起始程式將訊息放入此佇列。 這兩個佇列管理程式上的複製應用程式會處理佇列。

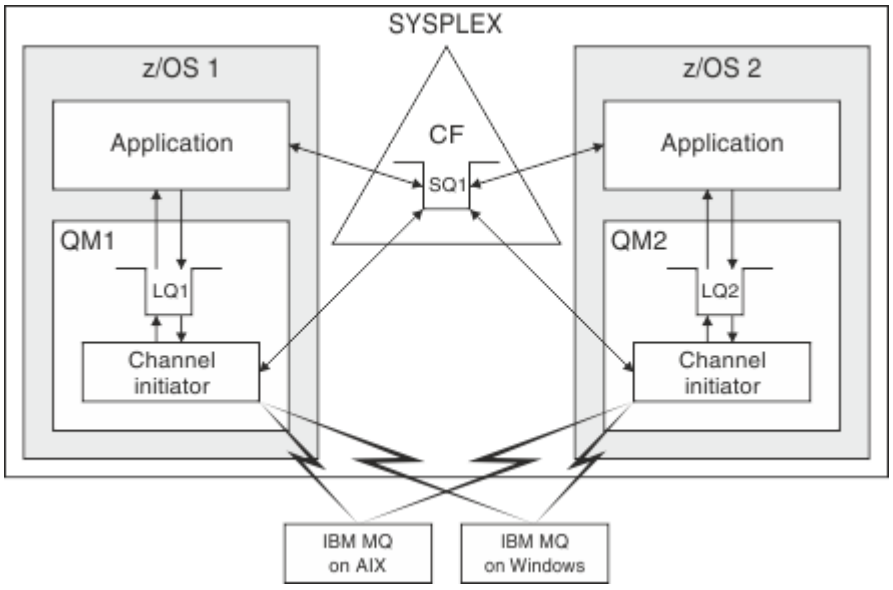


圖 *65:* 分散式佇列及佇列共用群組

相關概念

第 172 頁的『共用通道』
使用本主題來瞭解共用通道及其與 IBM MQ for z/OS 搭配使用的概念。

第 176 頁的『內部群組佇列』
本節說明內部群組佇列作業，這是 z/OS 平台特有的 IBM MQ for z/OS 功能。 此功能僅適用於定義給佇列共用群組的佇列管理程式。

第 173 頁的『叢集和佇列共用群組』
請利用這個主題來瞭解如何將佇列共用群組與叢集搭配使用。

z/OS 共用通道

使用本主題來瞭解共用通道及其與 IBM MQ for z/OS 搭配使用的概念。

許多網路產品提供一種機制，可隱藏網路上的伺服器故障，或平衡一組合格伺服器之間的入埠網路要求。 網路產品讓 一般埠 可用於入埠網路連線要求，並且可以透過連接其中一部合格伺服器來滿足入埠要求。

這些網路產品包括:

- VTAM 一般資源
- SYSPLEX 經銷商

通道起始程式利用這些產品來使用共用佇列的功能

共用通道有兩種類型: 共用入埠通道和 共用出埠通道。

- 共用入埠通道
- 共用出埠通道

如需通道的進一步相關資訊，請參閱

- 共用通道摘要
- 共用通道狀態

共用入埠通道

佇列共用群組中的每一個通道起始程式都會啟動額外的接聽器作業，以在 一般埠上接聽。 此一般埠由其中一種支援技術 (VTAM、TCP/IP) 提供給網路使用。 網路技術會將一般埠的入埠網路連接要求分派給佇列共用群組 (QSG) 中正在一般埠上接聽的任何一個接聽器。

如果通道起始程式可以存取具有該名稱之通道的通道定義，則您可以在通道起始程式上啟動入埠連接所導向的通道。 您可以將通道定義定義為專用於佇列管理程式，或儲存在共用儲存庫上，以便可以在任何位置使用 (廣域定義)。 這表示您可以透過將通道定義定義定義為廣域定義，在佇列共用群組中的任何通道起始程式上提供通道定義。

當透過一般埠來啟動通道時，還有另一個差異; 通道同步化是與佇列共用群組，而不是與個別佇列管理程式。 例如，考量遠端佇列管理程式透過一般埠啟動通道。 當通道第一次啟動時，它可能會在佇列管理程式 QM1 及訊息流程上啟動。 如果通道在佇列管理程式 QM2 上停止並重新啟動，則由於與佇列共用群組同步化，因此已傳送的訊息數相關資訊仍然正確。

您可以使用透過一般埠啟動的入埠通道，將訊息放入任何佇列。 遠端佇列管理程式不知道是否共用目標佇列。 如果目標佇列是共用佇列，則遠端佇列管理程式會以負載平衡方式透過任何可用的通道起始程式連接，並將訊息放入共用佇列。

如果目標佇列是專用佇列，則訊息會放入通道現行實例所連接的任何佇列管理程式所擁有的專用佇列中。 在此環境中 (稱為 抄寫的本端佇列) ，每一個佇列管理程式必須定義一組相同的專用佇列。

配置佇列共用群組的 SVRCONN 通道

佇列共用群組中 SVRCONN 通道的最佳配置是在每一個 CHINIT 中設定專用接聽器，以使用不同於點對點通道的埠號。 然後這些接聽器埠會用作新的工作量配送機制 (例如使用虛擬 IP 位址 (VIPA) 的 Sysplex Distributor) 的「後端」資源。 然後會使用外部 VIPA 位址作為網路中 CLNTCONN 定義的目標位址。 SVRCONN 通道可以使用 QSGDISP (GROUP) 來定義，因此 QSG 中的所有佇列管理程式都可以使用相同的定義。 此配置可避免使用共用接聽器，因此降低維護共用通道狀態的佇列共用群組的效能影響，主從式通道不需要此佇列共用群組。

共用出埠通道

如果出埠通道是從共用傳輸佇列取得訊息，則會被視為共用通道。 如果它是共用的，它會保留佇列共用群組層次的同步化資訊。 這表示如果通訊子系統、通道起始程式或佇列管理程式失敗，則可以在佇列共用群組內

的不同佇列管理程式及通道起始程式實例上重新啟動通道。 以此方式重新啟動失敗通道是共用通道的一項特性，稱為 同層級通道回復。

共用出埠通道的工作量平衡

如果您未指定要在特定通道起始程式上啟動出埠共用通道，則該通道可以在佇列共用群組內的任何通道起始程式上啟動。 IBM MQ 選取的通道起始程式是使用下列準則來決定:

- 通道起始程式目前是否需要通訊子系統?
- 通道起始程式是否可以使用 Db2 連線?
- 哪些通道起始程式具有最低現行工作量? 工作量包括作用中及重試中的通道。

共用通道摘要

共用通道與專用通道有下列不同:

專用通道
關聯於單一通道起始程式。

- 出埠通道使用本端傳輸佇列。
- 已透過本端埠啟動入埠通道。
- 同步化資訊保留在 SYSTEM.CHANNEL.SYNCQ 佇列。

共用通道
透過高可用性來平衡工作量。

- 出埠通道使用共用傳輸佇列。
- 已透過一般埠啟動入埠通道。
- 同步化資訊保留在 SYSTEM.QSG.CHANNEL.SYNCQ 佇列。

當您在 START CHANNEL 指令中使用 CHLDISP 選項來啟動通道時，您可以指定通道是專用還是共用。 共用通道可以透過與專用通道相同的方式觸發來啟動。 不過，當啟動共用通道時， IBM MQ 會執行工作量平衡，並在佇列共用群組內最適當的通道起始程式上啟動通道。 (必要的話，您可以指定在特定通道起始程式上啟動共用通道。)

共用通道狀態

佇列共用群組中的通道起始程式會在 Db2 中維護共用通道狀態表格。 這會記錄哪些通道在哪些通道起始程式上處於作用中。 如果有通道起始程式或通訊系統失敗，則會使用共用通道狀態表格。 它指出哪些通道需要在佇列共用群組中的不同通道起始程式上重新啟動。

z/OS 叢集和佇列共用群組

請利用這個主題來瞭解如何將佇列共用群組與叢集搭配使用。

您可以在單一定義中使共用佇列可供叢集使用。 若要這樣做，您可以在定義共用佇列時指定叢集的名稱。

網路中的使用者會將共用佇列視為由佇列共用群組內的每一個佇列管理程式所管理 (共用佇列未通告為由佇列共用群組所管理)。 用戶端可以啟動與佇列共用群組任何成員的階段作業，以將訊息放置到相同的共用佇列。

第 174 頁的圖 66 顯示叢集成員如何透過佇列共用群組的任何成員來存取共用佇列。

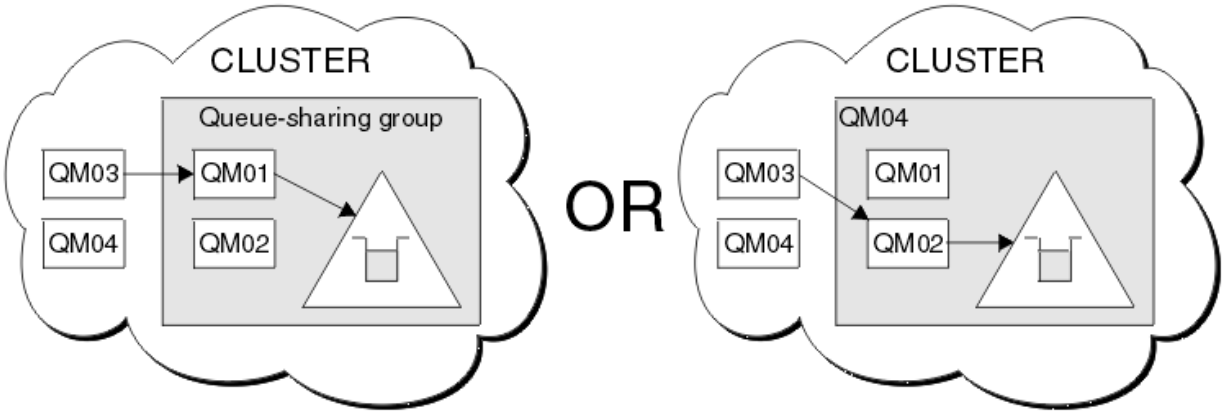


圖 *66:* 作為叢集一部分的佇列共用群組

z/OS 影響共用佇列的工作量配送

請利用這個主題來瞭解影響佇列共用群組中共用佇列之工作量配送的因素。

IBM MQ 不提供共用佇列的工作量平衡。 不過，佇列共用群組 (QSG) 中的工作量配送可能會以 基於取回的方式受到影響。 佇列共用群組中每一個佇列管理程式的可用處理容量，以及跨 Sysplex 定義的工作量管理目標，會影響佇列服務 (接收寫入共用佇列的訊息) 的佇列管理程式選擇。

不過，請務必瞭解，執行訊息 MQPUT 的佇列管理程式也可能對決定哪個佇列管理程式會取得訊息有很大影響。

本端佇列管理程式更有可能執行 MQGET

對於執行 MQPUT 的應用程式，本端佇列管理程式被認為是應用程式所連接的佇列管理程式。

下列考量會影響代表取得應用程式執行 MQGET 來提供訊息之 MQPUT 服務的確切佇列管理程式。

將訊息放入空的共用佇列時，通常會先公佈本端佇列管理程式，再通知佇列共用群組中的任何其他佇列管理程式。 如果本端佇列管理程式位於處理訊息的位置，則它會在 QSG 中任何其他佇列管理程式之前從連結機能 (CF) 接收清單轉移通知。 (清單轉移通知是共用佇列已從空狀態變更為非空狀態的通知。)

在此情況下，可能的實務範例如下:

1. 非持續訊息的 MQPUT 不同步點，並 快速放置到等待的 *getter*。

   如果有應用程式在佇列的本端佇列管理程式上具有 *MQGET with wait* ，則訊息的 MQPUT 會直接傳遞至取得應用程式的緩衝區，而不會寫入佇列。 這適用於共用及非共用佇列。 此特性通常稱為 快速放入等待中的 *getter* 機制。 如果是共用佇列，則不會通知 QSG 中的其他佇列管理程式，因為佇列沒有從空的轉移至非空的轉移。 例如，這表示如果提供此佇列管理程式可以處理來自此應用程式的所有放置，並假設沒有其他應用程式將訊息放入佇列，則佇列共用群組中沒有其他佇列管理程式協助排除此佇列。 不過，如果本端佇列管理程式沒有等待的 MQGET ，且將訊息放入共用佇列，則 CF 會根據其清單轉移通知規則來通知佇列共用群組中的其他佇列管理程式。
2. 持續或同步點訊息的 MQPUT。

   在此情況下，如果有應用程式在本端佇列管理程式上具有 *MQGET with wait* ，則會將訊息放入共用佇列，且 CF 會根據其清單轉移通知規則來通知佇列共用群組中的其他佇列管理程式。 不過，本端佇列管理程式不會等待來自 CF 的轉移通知，但會先允許任何本端 *MQGET with wait* ，且通常會先代表應用程式執行此訊息的取得，然後佇列共用群組中的任何其他佇列管理程式才能回應 CF 通知。 這取決於本端佇列管理程式忙碌的程度。 否則，由於訊息到達空佇列而由 CF 通知的任何佇列管理程式，將會嘗試先處理取得。 第一個回應處理新訊息的佇列管理程式。
3. 最後，如果佇列未排除訊息 (CF 已針對佇列傳送狀態從空變更為非空的通知) ，則所有連接的佇列管理程式都有機會協助處理佇列。 在此事件中，工作量據說是 以拉取為基礎。

此設計容許透過純取回型工作量分配來改善效能。 目的是利用 CF 中所保留佇列所提供的高可用性，同時儘可能容許佇列管理程式執行 MQGET ， 而不需要參照 CF ， 以盡可能有效率地處理訊息工作量。

如果強調工作量平衡比先前說明的效能加強功能更重要，則可以採用替代方法。 例如，確保未將任何取得的應用程式連接至放置應用程式所連接的相同佇列管理程式。 使用此設計時，會將所有訊息放入佇列，並在佇列從空白移至非空白時，根據 CF 演算法來處理此類轉移，通知 QSG 中的所有佇列管理程式。 此外， *fast put to waiting getter* 機制不適用。

z/OS 在何處尋找共用佇列及佇列共用群組的相關資訊

請利用這個主題中的表格來尋找 IBM MQ for z/OS 如何使用共用佇列和佇列共用群組的相關資訊。

表 *19:* 在何處尋找共用佇列及佇列共用群組的相關資訊

主題	**在何處尋找**
佇列共用群組回復	第 210 頁的『在 z/OS 上回復並重新啟動』
佇列共用群組安全	第 223 頁的『IBM MQ for z/OS 中的安全概念』
專用及廣域物件定義 將指令導向至不同的佇列 管理程式	您可以在 z/OS
規劃連結機能 環境	定義連結機能資源
規劃 SMDS 環境	規劃共用訊息資料集 (SMDS) 環境
規劃 Db2 環境	規劃 Db2 環境
設定共用佇列 系統參數	第 139 頁的『共用佇列及佇列共用群組』
公用程式 移轉佇列	z/OS 參照上的 IBM MQ 公用程式
主控台訊息	IBM MQ for z/OS 的訊息
MQSC 指令	MQSC 指令
IBM MQ 叢集	配置佇列管理程式叢集
IBM MQ 分散式佇列 (distributed queuing) 通道名稱	分散式佇列管理簡介
撰寫應用程式	應用程式設計概觀
MQCONNX 呼叫	MQCONNX

z/OS 內部群組佇列

本節說明內部群組佇列作業，這是 z/OS 平台特有的 IBM MQ for z/OS 功能。 此功能僅適用於定義給佇列共用群組的佇列管理程式。

如需佇列共用群組的相關資訊，請參閱 第 139 頁的『共用佇列及佇列共用群組』。

內部群組佇列作業概念

您可以在佇列共用群組中的佇列管理程式之間執行快速訊息傳送，而無需定義通道。 這會使用稱為 SYSTEM.QSG.TRANSMIT.QUEUE，這是共用傳輸佇列。 佇列共用群組中的每一個佇列管理程式都會啟動稱為內部群組佇列作業代理程式的作業，該代理程式會等待訊息到達此佇列，而這些訊息是送往其佇列管理程式。 當偵測到這類訊息時，會將它從佇列中移除，並放在正確的目的地佇列中。

會使用標準名稱解析規則，但如果已啟用內部群組佇列作業 (IGQ)，且目標佇列管理程式位於佇列共用群組內，則為 SYSTEM.QSG.TRANSMIT.QUEUE 用來將訊息傳送至正確的目的地佇列管理程式，而不是使用傳輸佇列及通道。

您可以透過佇列管理程式屬性來啟用內部群組佇列作業。 內部群組佇列作業會將非持續訊息移至同步點之外，並將持續訊息移至同步點內。 如果發現將訊息遞送至目標佇列時發生問題，則內部群組佇列作業會嘗試將訊息放入無法傳送郵件的佇列。 如果無法傳送郵件的佇列已滿或未定義，則會捨棄非持續訊息，但持續訊息會取消並回到 SYSTEM.QSG.TRANSMIT.QUEUE， IGQ 代理程式會嘗試遞送訊息，直到成功為止。

接收以佇列共用群組中不同佇列管理程式上的佇列為目的地之訊息的入埠共用通道，可以使用內部群組佇列作業來 躍點 訊息至正確的目的地。

有時，如果目標佇列是共用佇列，而不是最先傳送至目標佇列管理程式的訊息，您可能會想要本端佇列管理程式將訊息直接放置到目標佇列。 您可以使用佇列管理程式屬性 SQQMNAME 來控制此情況。 如果您將 SQQMNAME 的值設為 USE，則會在 **ObjectQMgrName** 指定的佇列管理程式上執行 MQOPEN 指令。

不過，如果目標佇列是共用佇列，且您將 SQQMNAME 的值設為 IGNORE，且 **ObjectQMgrName** 是佇列共用群組中另一個佇列管理程式的值，則會在本端佇列管理程式上開啟共用佇列。 如果本端佇列管理程式無法開啟目標佇列，或將訊息放入佇列，則會透過 IGQ 或 IBM MQ 通道，將訊息傳送至指定的 **ObjectQMgrName**。

內部群組佇列作業可用來更有效率地將小型訊息遞送至佇列共用群組內遠端佇列管理程式上的佇列。 內部群組佇列作業也支援大型訊息，最大為 100 MB 減去 傳輸佇列標頭長度。

註: 如果您使用此特性，使用者必須對佇列共用群組中每一個佇列管理程式上的佇列具有相同的存取權。

下圖顯示內部群組佇列作業的一般範例。

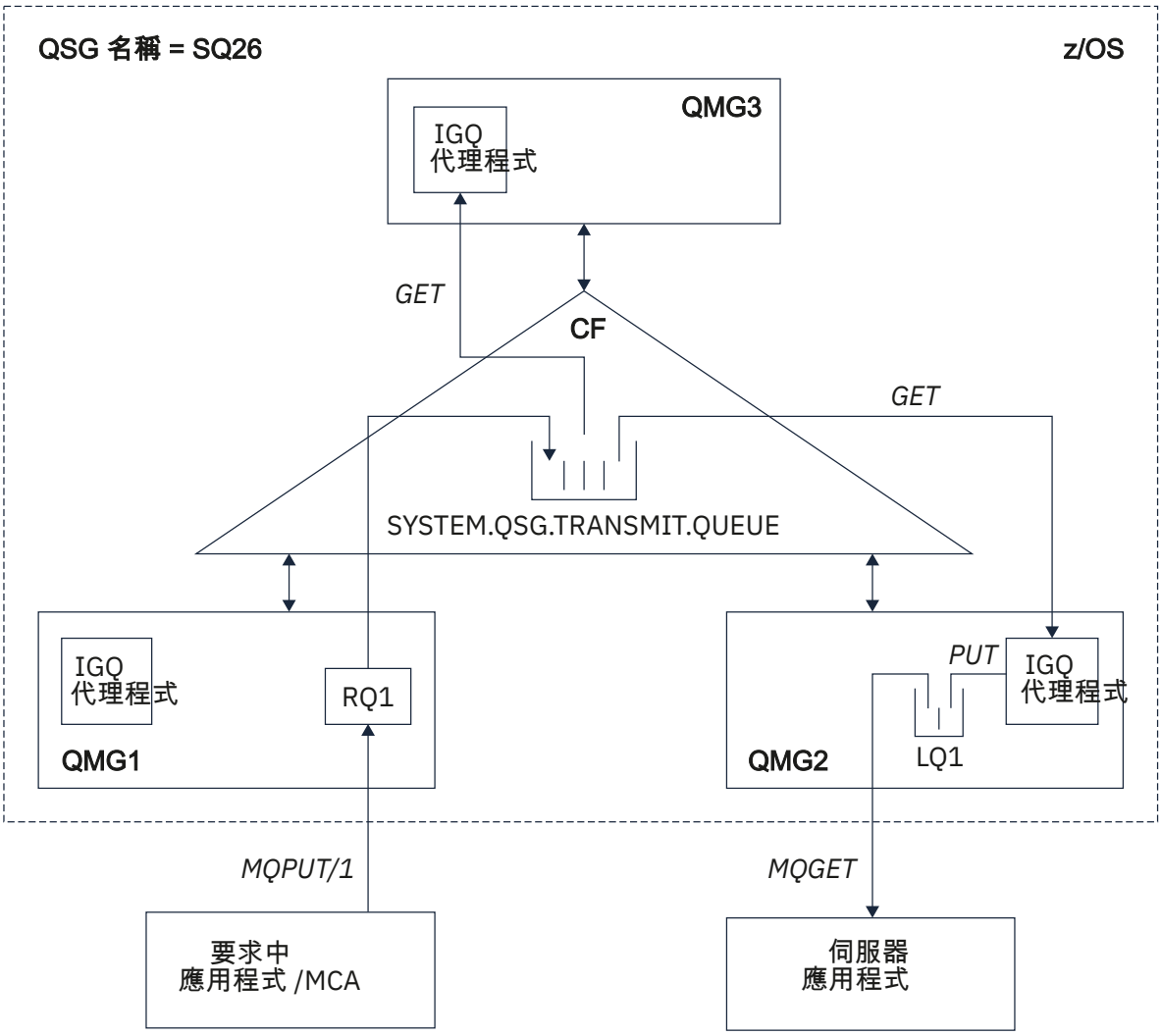


圖 67: 內部群組佇列作業的範例

圖表顯示:

- 在定義給佇列共用群組 SQ26 的三個佇列管理程式 (QMG1、 QMG2 及 QMG3) 上執行的 IGQ 代理程式。
- 共用傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE 。
- 在佇列管理程式 QMG1 中定義的遠端佇列定義。
- 在佇列管理程式 QMG2 中定義的本端佇列。
- 連接至佇列管理程式 QMG1 的要求應用程式 (此應用程式可以是「訊息通道代理程式 (MCA)」)。
- 連接至佇列管理程式 QMG2 的伺服器應用程式。
- 正在對 SYSTEM.QSG.TRANSMIT.QUEUE。

內部群組佇列作業及內部群組佇列作業代理程式

在佇列管理程式起始設定期間會啟動 IGQ 代理程式。 當應用程式開啟訊息並將訊息放置到遠端佇列時，本端佇列管理程式會決定是否使用內部群組佇列作業來進行訊息傳送。 如果要使用內部群組佇列作業，本端佇列管理程式會將訊息放入 SYSTEM.QSG.TRANSMIT.QUEUE。 目標遠端佇列管理程式上的 IGQ 代理程式會擷取訊息，並將它放在目的地佇列上。

內部群組佇列作業術語

術語的說明: 群組內佇列作業、群組內佇列作業使用的共用傳輸佇列，以及群組內佇列作業代理程式。

內部群組佇列

內部群組佇列作業可能會影響佇列共用群組中佇列管理程式之間的快速且較便宜的訊息傳送，而不需要定義通道。

供內部群組佇列作業使用的共用傳輸佇列

每一個佇列共用群組都有一個稱為 SYSTEM.QSG.TRANSMIT.QUEUE，供內部群組佇列作業使用。 如果啟用內部群組佇列作業，則為 SYSTEM.QSG.TRANSMIT.QUEUE 會出現在名稱解析路徑中。 當應用程式 (包括「訊息通道代理程式 (MCA)」) 將訊息放置到遠端佇列時，本端佇列管理程式會判定訊息快速傳送的資格，並將它們放置在 SYSTEM.QSG.TRANSMIT.QUEUE。

內部群組佇列作業代理程式

IGQ 代理程式是在佇列管理程式起始設定時啟動的作業，它會等待適當的訊息到達 SYSTEM.QSG.TRANSMIT.QUEUE。 IGQ 代理程式會從此佇列擷取適當的訊息，並將它們遞送至目的地佇列。

每一個佇列管理程式的 IGQ 代理程式一律會啟動，因為佇列管理程式本身會使用內部群組佇列作業來進行自己的內部處理。

z/OS 群組內佇列作業的好處

群組內佇列作業的好處如下: 減少系統定義、減少系統管理、改良效能、支援移轉，以及在佇列共用群組中佇列管理程式之間進行多跳時遞送訊息。

群組內佇列作業的好處如下:

減少系統定義
內部群組佇列作業不需要在佇列共用群組中的佇列管理程式之間定義通道。

減少系統管理
由於佇列共用群組中的佇列管理程式之間未定義任何通道，因此不需要通道管理。

提高效能
因為將訊息遞送至目標佇列只需要一個 IGQ 代理程式 (而不是兩個中間傳送端及接收端代理程式)，所以使用內部群組佇列作業來遞送訊息的成本可能比使用通道來遞送訊息的成本低。 在內部群組佇列作業中，只有一個接收元件，因為已移除傳送元件的需求。 這是因為當本端佇列管理程式上的放置作業完成時，訊息可供位於目的地佇列管理程式的 IGQ 代理程式遞送至目的地佇列，且如果訊息放置在同步點範圍內，則為已確定。

支援移轉

佇列共用群組外部的應用程式可以將訊息遞送至佇列共用群組中任何佇列管理程式上的佇列，同時只連接至佇列共用群組中的特定佇列管理程式。 這是因為到達接收端通道的訊息 (以遠端佇列管理程式上的佇列為目的地) 可以使用內部群組佇列作業透通地傳送至目的地佇列。 此機能可讓您在佇列共用群組之間部署應用程式，而不需要變更佇列共用群組外部的任何系統。

下圖說明一般配置，其中:

- 連接至佇列管理程式 QMG1 的要求應用程式需要將訊息傳送至佇列管理程式 QMG3 上的本端佇列。
- 佇列管理程式 QMG1 僅連接至佇列管理程式 QMG2。
- 先前使用通道連接的佇列管理程式 QMG2 及 QMG3 現在是佇列共用群組 SQ26 的成員。

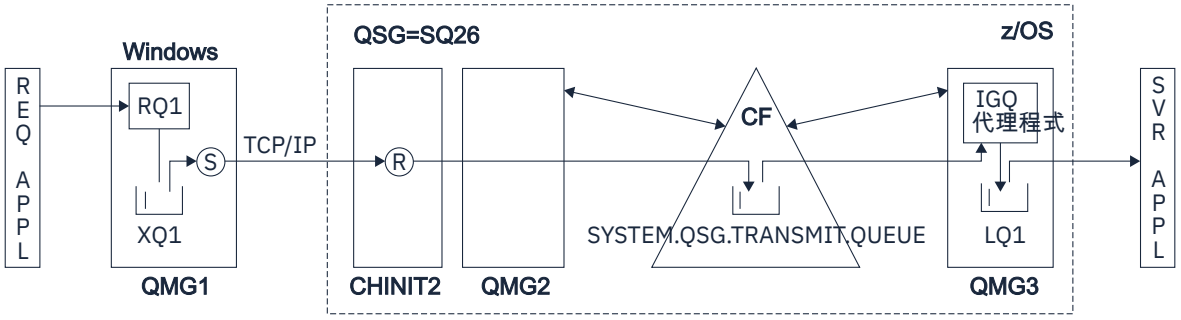


圖 *68:* 移轉支援範例

作業流程如下:

1. 發出要求的應用程式會將訊息 (以遠端佇列管理程式 QMG3 上的本端佇列 LQ1 為目的地) 放到遠端佇列定義 RQ1。
2. 在 Windows NT 工作站上執行的佇列管理程式 QMG1 會將訊息放入傳輸佇列 XQ1 中。
3. QM1 上的傳送端 MCA (S) 會使用 TCP/IP 將訊息傳輸至通道起始程式 CHINIT2 上的接收端 MCA (R)。
4. 通道起始程式 CHINIT2 上的接收端 MCA ® 會將訊息放入共用傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE。
5. 佇列管理程式 QMG3 上的 IGQ 代理程式會從 SYSTEM.QSG.TRANSMIT.QUEUE 並將它放置在目標本端佇列 LQ1 上。
6. 伺服器應用程式會從目標本端佇列擷取訊息並加以處理。

在佇列共用群組中的佇列管理程式之間多跳時遞送訊息

支援移轉 中的前一個圖表也說明在佇列共用群組中佇列管理程式之間進行多重跳躍時的訊息遞送。 到達佇列共用群組內的佇列管理程式，但目的地為佇列共用群組中另一個佇列管理程式上的佇列的訊息，可以使用內部群組佇列作業，輕鬆地傳輸至目的地佇列管理程式上的目的地佇列。

z/OS 內部群組佇列作業的限制

內部群組佇列作業的限制為: 可以使用內部群組佇列作業進行傳送的訊息、每個佇列管理程式的內部群組佇列作業代理程式數目，以及啟動和停止內部群組佇列作業代理程式。

本主題說明內部群組佇列作業的限制。

可使用內部群組佇列作業進行傳送的訊息

因為內部群組佇列作業使用連結機能 (CF) 中定義的共用傳輸佇列，所以內部群組佇列作業限制為遞送共用佇列支援訊息長度上限減去傳輸佇列標頭長度 (MQXQH) 的訊息。

每個佇列管理程式的內部群組佇列作業代理程式數目

在佇列共用群組中，每個佇列管理程式只會啟動一個 IGQ 代理程式。

啟動及停止內部群組佇列作業代理程式

IGQ 代理程式在佇列管理程式起始設定期間啟動，並在佇列管理程式關閉期間終止。 它設計為長時間執行、自我回復 (發生異常終止時)、作業。 如果 SYSTEM.QSG.TRANSMIT.QUEUE (例如，如果此佇列禁止「取得」)， IGQ 代理程式會繼續重試。 如果 IGQ 代理程式在佇列管理程式仍在作用中時發生導致代理程式正常終止的錯誤，則可以發出 ALTER QMGR IGQ (ENABLED) 指令來重新啟動它。 此指令可避免需要重新啟動佇列管理程式。

將佇列管理程式屬性 IGQ 設為 ENABLED 或 DISABLED

如果佇列管理程式屬性 IGQ 設為 ENABLED 或 DISABLED ，則現有的物件控點可能會失效，原因碼 MQRC_OBJECT_CHANGED。 如需相關資訊，請參閱 開始使用內部群組佇列 。

z/OS 開始使用內部群組佇列作業

您可以啟用、停用及使用內部群組佇列作業，如本主題所述。

啟用內部群組佇列作業

若要在佇列管理程式上啟用內部群組佇列作業，您需要執行下列動作:

- 定義稱為 SYSTEM.QSG.TRANSMIT.QUEUE。 此佇列的定義可在佇列共用群組的 SYSTEM 物件的 thlqual.SCSQPROC(CSQ4INSS) CSQINP2 範例中找到。 這個佇列必須依照 thlqual.SCSQPROC(CSQ4INSS) 中的說明，以正確的屬性來定義，群組內佇列作業才能正常運作。
- 因為 IGQ 代理程式一律在佇列管理程式起始設定時啟動，所以入埠訊息處理一律可以使用內部群組佇列作業。 IGQ 代理程式會處理放置在 SYSTEM.QSG.TRANSMIT.QUEUE。 不過，若要啟用出埠處理程序的內部群組佇列作業，佇列管理程式屬性 IGQ 必須設為 ENABLED。

  重要: 如果佇列管理程式屬性 IGQ 設為 ENABLED，則現有物件控點可能會失效，原因碼為 MQRC_OBJECT_CHANGED。 如需相關資訊，請參閱 第 186 頁的『內部群組佇列作業的特定內容』。 如此原因碼的「程式設計師回應」小節中所述，需要撰寫應用程式以處理此狀況 (如需詳細資料，請參閱 2041 (07F9) (RC2041) :MQRC_OBJECT_changed)。

此外，由於 IGQ 設計為長時間執行且自行回復的作業 (在起始設定期間啟動並在關閉時終止)，如需進一步資訊，請參閱 第 179 頁的『內部群組佇列作業的限制』。

停用內部群組佇列作業

若要停用出埠訊息傳送的內部群組佇列作業，請將佇列管理程式屬性 IGQ 設為 DISABLED。 如果針對特定佇列管理程式停用內部群組佇列作業，則該佇列管理程式上的 IGQ 代理程式仍然可以處理已放置在 SYSTEM.QSG.TRANSMIT.QUEUE，該佇列管理程式已啟用出埠傳送的內部群組佇列作業。

重要: 如果佇列管理程式屬性 IGQ 設為 ENABLED，則現有物件控點可能會失效，原因碼為 MQRC_OBJECT_CHANGED。 如需相關資訊，請參閱 第 186 頁的『內部群組佇列作業的特定內容』。 如此原因碼的「程式設計師回應」小節中所述，需要撰寫應用程式以處理此狀況 (如需詳細資料，請參閱 2041 (07F9) (RC2041) :MQRC_OBJECT_changed)。

此外，由於 IGQ 設計為長時間執行且自行回復的作業 (在起始設定期間啟動並在關閉時終止)，如需進一步資訊，請參閱 第 179 頁的『內部群組佇列作業的限制』。

使用內部群組佇列作業

一旦啟用內部群組佇列作業，即可使用它，且佇列管理程式會盡可能使用它。 亦即，當應用程式將訊息放入遠端佇列定義、完整遠端佇列或叢集佇列時，佇列管理程式會判斷訊息是否適合使用內部群組佇列作業來遞送，如果是，則會將訊息放入 SYSTEM.QSG.TRANSMIT.QUEUE。 不需要變更使用者應用程式或應用程式佇列，因為對於合格訊息，佇列管理程式會使用 SYSTEM.QSG.TRANSMIT.QUEUE，優先於任何其他傳輸佇列。

z/OS 內部群組佇列作業配置

除了一般內部群組佇列作業配置之外，還可以進行其他配置。

第 176 頁的『內部群組佇列作業概念』 說明一般配置。

相關概念

第 180 頁的『具有內部群組佇列作業的分散式佇列 (多個遞送路徑)』
對於處理短訊息的應用程式，可能只能配置內部群組佇列作業，以便在佇列共用群組中的佇列管理程式之間遞送訊息。

第 182 頁的『具有內部群組佇列作業 (多個遞送路徑) 的叢集作業』
可以配置佇列管理程式，使其位於叢集及佇列共用群組中。

第 184 頁的『叢集作業、群組內佇列作業及分散式佇列作業』
可以使用傳送端/接收端通道配對來配置佇列管理程式，該佇列管理程式是叢集及佇列共用群組的成員，並連接至分散式佇列管理程式。

z/OS 具有內部群組佇列作業的分散式佇列 (多個遞送路徑)

對於處理短訊息的應用程式，可能只能配置內部群組佇列作業，以便在佇列共用群組中的佇列管理程式之間遞送訊息。

通道通訊的內部群組佇列作業選擇可由 CFSTRUCT 類型層次控制。 (3，而不是 4 或 5)。 在 SYSTEM.QSQ.TRANSMIT.QUEUE。

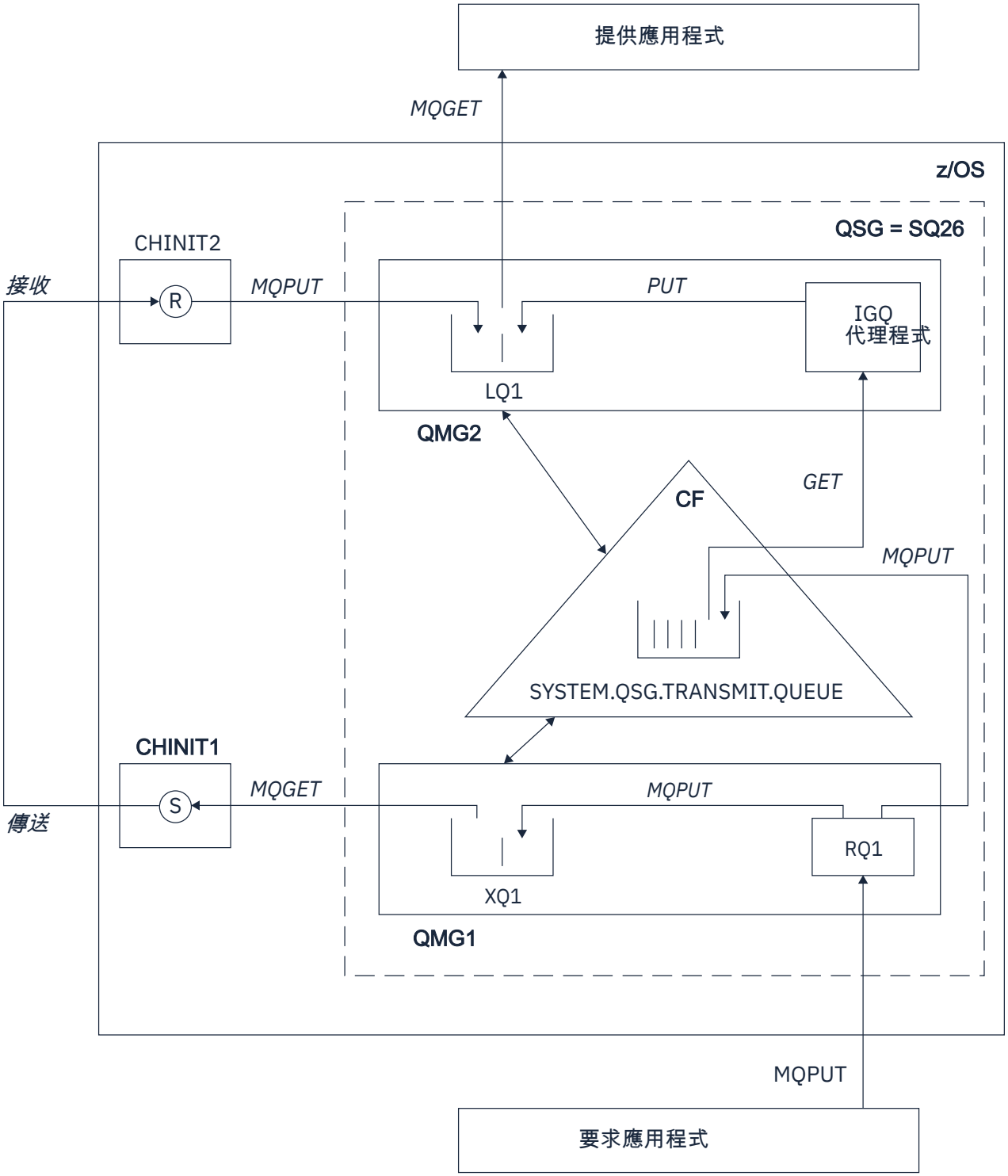


圖 *69:* 範例配置

開啟/放置處理

1. 請務必注意，當發出要求的應用程式開啟遠端佇列 RQ1 時，非共用傳輸佇列 XQ1 及共用傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE。
2. 當發出要求的應用程式將訊息放入遠端佇列時，會根據佇列管理程式上的出埠傳送是否啟用內部群組佇列作業，以及訊息性質，將訊息放入傳輸佇列 XQ1 或傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE。 佇列管理程式會將所有大型訊息放入傳輸佇列 XQ1，並將所有小型訊息放入傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE。

3. 如果傳輸佇列 XQ1 已滿或無法使用，則大型訊息的放置要求會同步失敗，並提供適當的回覆碼和原因碼。 不過，小型訊息的放置要求會繼續成功，並放置在傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE。
4. 如果是傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE 已滿或無法放置，小型訊息的放置要求會同步失敗，並傳回適當的回覆碼和原因碼。 不過，大型訊息的放置要求會繼續成功，並放置在傳輸佇列 XQ1 上。 在此情況下，不會嘗試將小型訊息放入傳輸佇列。

大型訊息的流程

1. 發出要求的應用程式會將大量訊息放入遠端佇列 RQ1。
2. 佇列管理程式 QMG1 會將訊息放入傳輸佇列 XQ1。
3. 佇列管理程式 QMG1 上的傳送端 MCA (S) 會從傳輸佇列 XQ1 中擷取訊息，並將它們傳送至佇列管理程式 QMG2。
4. 佇列管理程式 QMG2 上的接收端 MCA (R) 會接收訊息，並將它們放入目的地佇列 LQ1 中。
5. 負責處理的應用程式會擷取並處理來自佇列 LQ1 的訊息。

小型訊息的流程

1. 發出要求的應用程式會將小型訊息放入遠端佇列 RQ1。
2. 佇列管理程式 QMG1 會將訊息放入傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE。
3. 佇列管理程式 QMG2 上的 IGQ 會擷取訊息並將它們放入目的地佇列 LQ1 中。
4. 負責處理的應用程式會從佇列 LQ1 擷取訊息。

指向附註

1. 發出要求的應用程式不需要知道用於遞送訊息的基礎機制。
2. 對於小型訊息，可以達到可能更快的訊息遞送機制。
3. 有多個路徑可用於訊息遞送 (亦即，一般通道路徑及群組內佇列作業路徑)。
4. 相較於一般通道路徑，會選取可能更快的內部群組佇列作業路徑。 視訊息性質而定，訊息遞送可能分成兩條路徑。 因此，訊息可能不按順序遞送 (不過，如果只使用一般通道路徑來遞送訊息，也可以進行此遞送)。
5. 當已選取路徑，且已在傳輸佇列中放置訊息時，只會使用選取的路徑來遞送訊息。 SYSTEM.QSG.TRANSMIT.QUEUE 不會轉移至傳輸佇列 XQ1。

z/OS 具有內部群組佇列作業 (多個遞送路徑) 的叢集作業

可以配置佇列管理程式，使其位於叢集及佇列共用群組中。

當訊息傳送至叢集佇列，且本端及遠端目的地佇列管理程式位於相同的佇列共用群組時，會使用內部群組佇列作業來遞送小型訊息 (使用 SYSTEM.QSG.TRANSMIT.QUEUE) ，以及如果群組內佇列作業支援訊息大小，則遞送大型訊息。 此外， SYSTEM.CLUSTER.TRANSMIT.QUEUE 用於將訊息遞送至叢集中但在佇列共用群組外部的任何佇列管理程式。 下圖說明此配置 (未顯示通道起始程式)。

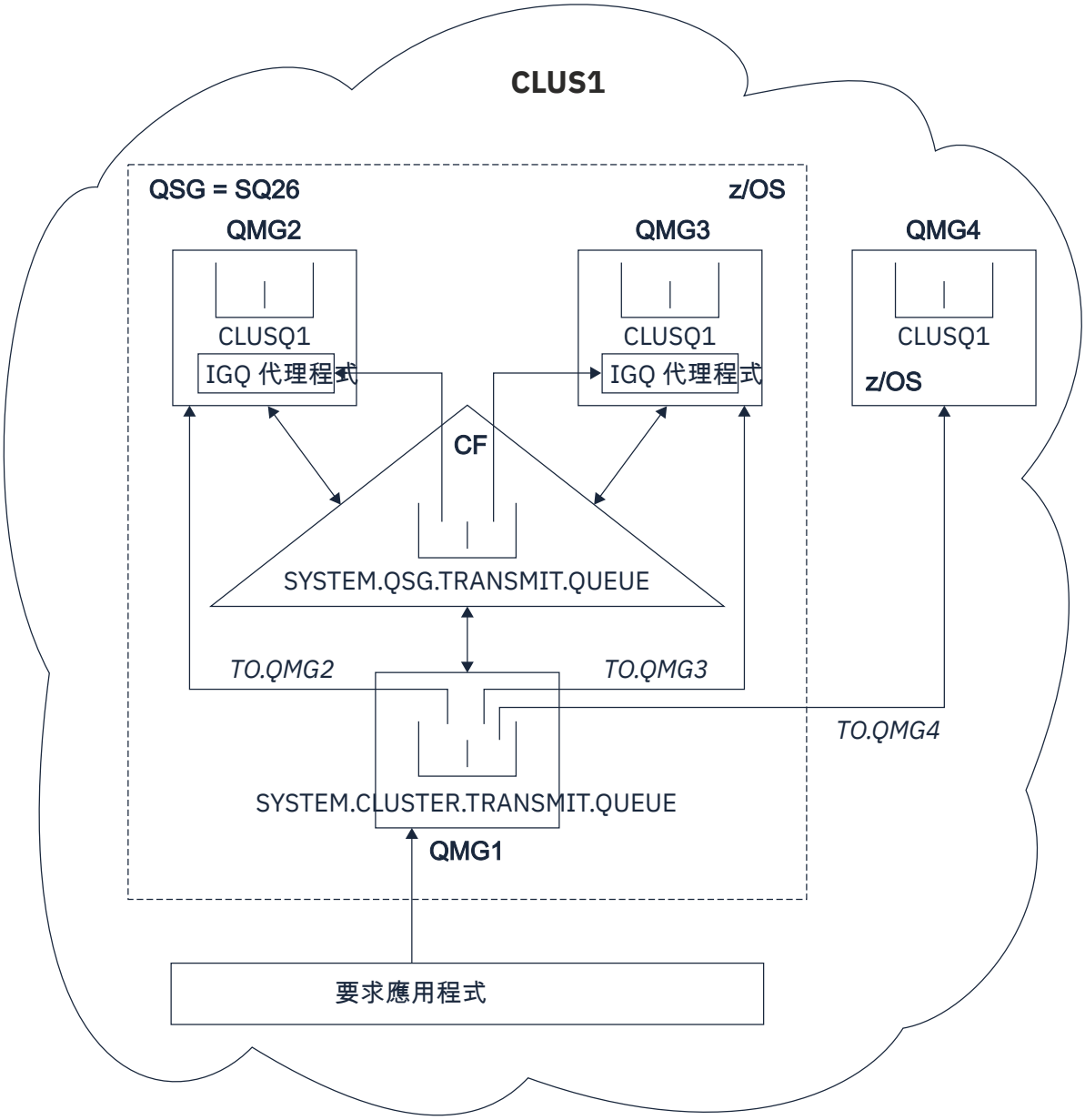


圖 *70:* 具有內部群組佇列作業的叢集作業範例

圖表顯示:

- 在叢集 CLUS1 中配置四個 z/OS 佇列管理程式 QMG1、 QMG2、 QMG3 及 QMG4 。
- 在佇列共用群組 SQ26 中配置佇列管理程式 QMG1、 QMG2 及 QMG3 。
- 在佇列管理程式 QMG2 及 QMG3 上執行的 IGQ 代理程式。
- QMG1 中定義的本端 SYSTEM.CLUSTER.TRANSMIT.QUEUE 。

  註: 為求明確，請參閱 SYSTEM.CLUSTER.TRANSMIT.QUEUE 。
- 共用 SYSTEM.QSG.TRANSMIT.QUEUE ， 其位於使用 CFLEVEL (3) RECOVER (YES) 屬性所配置的 IBM MQ 結構中。
- 叢集通道 TO.QMG2 (將 QMG1 連接至 QMG2) ， TO.QMG3 (將 QMG1 連接至 QMG3) 和 TO.QMG4 (將 QMG1 連接至 QMG4)。
- 在佇列管理程式 QMG2、 QMG3 及 QMG4 上管理叢集佇列 CLUSQ1 。

假設發出要求的應用程式使用 MQOO_BIND_NOT_FIXED 選項開啟叢集佇列，以便在放置時選取叢集佇列的目標佇列管理程式。

如果選取的目標佇列管理程式是 QMG2:

- 發出要求的應用程式所放置的所有大型訊息如下:
  - 放置到 QMG1 上的 SYSTEM.CLUSTER.TRANSMIT.QUEUE，因為 SYSTEM.QSG.TRANSMIT.QUEUE 是在 CFLEVEL (3) 結構中; 因此只支援大小高達 63 KB 的訊息。
  - 使用叢集通道 TO.QMG2 傳送至 QMG2 上的叢集佇列 CLUSQ1
- 發出要求的應用程式所放置的所有小型訊息
  - 放入共用傳輸佇列 SYSTEM.QSG.TRANSMIT.QUEUE。此佇列的結構已配置 RECOVER (YES) 屬性，因此同時用於持續性及非持續性小型訊息。
  - 由 QMG2 上的 IGQ 代理程式擷取
  - 放置到 QMG2 上的叢集佇列 CLUSQ1

如果選取的目標佇列管理程式是 QMG4:

- 因為 QMG4 不是佇列共用群組 SQ26 的成員，所以發出要求的應用程式所放置的所有訊息都是
  - 放置到 QMG1 上的 SYSTEM.CLUSTER.TRANSMIT.QUEUE
  - 使用叢集通道 TO.QMG4 傳送至 QMG4 上的叢集佇列 CLUSQ1

指向附註

- 發出要求的應用程式不需要知道用於遞送訊息的基礎機制。
- 在佇列共用群組中的佇列管理程式之間傳送小型非持續訊息 (即使相同佇列管理程式位於叢集中)，可達到可能更快的遞送機制。
- 有多個路徑可用於訊息遞送 (亦即，叢集路徑及內部群組佇列作業路徑)。
- 相較於叢集路徑，會優先選取可能更快的內部群組佇列作業路徑。視訊息性質而定，訊息遞送可能分成兩條路徑。因此，訊息可能不按順序遞送。請務必注意，不論應用程式指定的 MQOO_BIND_ * 選項，都可以進行這項遞送。根據是否在開啟時指定 MQOO_BIND_NOT_FIXED、MQOO_BIND_ON_OPEN、MQOO_BIND_ON_GROUP 或 MQOO_BIND_AS_Q_DEF，內部群組佇列作業會以與叢集作業相同的方式配送訊息。
- 當已選取路徑，且已在傳輸佇列中放置訊息時，只會使用選取的路徑來遞送訊息。SYSTEM.QSG.TRANSMIT.QUEUE 不會轉移至 SYSTEM.CLUSTER.TRANSMIT.QUEUE。

z/OS 叢集作業、群組內佇列作業及分散式佇列作業

可以使用傳送端/接收端通道配對來配置佇列管理程式，該佇列管理程式是叢集及佇列共用群組的成員，並連接至分散式佇列管理程式。

此配置是分散式佇列與內部群組佇列的組合，以及叢集與內部群組佇列的組合。

內部群組佇列作業在 第 180 頁的『具有內部群組佇列作業的分散式佇列 (多個遞送路徑)』中說明。

第 182 頁的『具有內部群組佇列作業 (多個遞送路徑) 的叢集作業』中說明了具有內部群組佇列作業的叢集作業。

z/OS 內部群組佇列作業訊息

本節說明放置到 SYSTEM.QSG.TRANSMIT.QUEUE。

訊息結構

與放置到傳輸佇列的所有其他訊息一樣，放置到 SYSTEM.QSG.TRANSMIT.QUEUE 以傳輸佇列標頭 (MQXQH) 為字首。

訊息持續性

在 IBM WebSphere MQ 5.3 及以上版本中，共用佇列同時支援持續及非持續訊息。

如果佇列管理程式在 IGQ 代理程式處理非持續訊息時終止，或 IGQ 代理程式在處理訊息時異常終止，則處理中的非持續訊息可能會遺失。 如果需要回復非持續訊息，應用程式必須安排回復非持續訊息。

如果 IGQ 代理程式發出的非持續訊息的放置要求非預期地失敗，則會遺失正在處理的訊息。

訊息遞送
IGQ 代理程式會擷取並遞送同步點範圍以外的所有非持續訊息，以及同步點範圍內的所有持續訊息。 在此情況下， IGQ 代理程式會作為同步點協調程式。 因此， IGQ 代理程式會處理非持續訊息，例如在訊息通道上處理快速非持續訊息的方式。 請參閱 快速、非持續訊息。

訊息批次
IGQ 代理程式使用 50 則訊息的固定批次大小。 在批次內擷取的任何持續訊息會以 50 則訊息的間隔來確定。 當 SYSTEM.QSG.TRANSMIT.QUEUE。

訊息大小
可以放置到 SYSTEM.QSG.TRANSMIT.QUEUE 是共用佇列支援的訊息長度上限減去傳輸佇列標頭 (MQXQH) 的長度。

預設訊息持續性和預設訊息優先順序
如果 SYSTEM.QSG.TRANSMIT.QUEUE 位於開啟時建立的佇列名稱解析路徑中，因此對於放置有預設持續性及預設優先順序 (或具有預設持續性或預設優先順序) 的訊息，會在選取具有預設優先順序及所使用的持續性值的佇列時套用一般規則。 (如需佇列選擇規則的相關資訊，請參閱 IBM MQ 訊息 一節)。

相關概念
第 185 頁的『未遞送/未處理的訊息』
本主題說明 SYSTEM.QSG.TRANSMIT.QUEUE。

第 185 頁的『報告訊息-內部群組佇列作業』
本主題說明報告訊息: 確認到達、確認遞送、到期報告及異常狀況報告。

z/OS 未遞送/未處理的訊息

本主題說明 SYSTEM.QSG.TRANSMIT.QUEUE。

如果 IGQ 代理程式無法將訊息遞送至目的地佇列， IGQ 代理程式:

- 允許使用 MQRO_DISCARD_MSG 報告選項 (如果未遞送訊息的 MQMD 的「報告選項」欄位指出它必須) ，並捨棄未遞送訊息。
- 嘗試將未遞送的訊息放入目的地佇列管理程式的無法傳送郵件的佇列 (如果尚未捨棄該訊息)。 IGQ 代理程式會以無法傳送郵件的佇列標頭 (MQDLH) 作為訊息字首。

如果未定義無法傳送的郵件佇列，或無法將無法遞送的訊息放入無法傳送的郵件佇列，且無法遞送的訊息為:

- 持續， IGQ 代理程式會取消它正在處理的現行持續訊息批次，並進入重試狀態。 如需相關資訊，請參閱 第 186 頁的『內部群組佇列作業的特定內容』。
- 非持續性， IGQ 代理程式會捨棄訊息並繼續處理下一則訊息。

如果佇列共用群組中的佇列管理程式在其相關聯 IGQ 代理程式有時間處理其所有訊息之前終止，則未處理的訊息會保留在 SYSTEM.QSG.TRANSMIT.QUEUE 直到佇列管理程式下一次啟動為止。 然後 IGQ 代理程式會擷取訊息並將其遞送至目的地佇列。

如果連結機能在 SYSTEM.QSG.TRANSMIT.QUEUE ， 任何未處理的非持續訊息都會遺失。

IBM 建議應用程式不要將訊息直接放置到傳輸佇列。 如果應用程式將訊息直接放置到 SYSTEM.QSG.TRANSMIT.QUEUE， IGQ 代理程式可能無法處理這些訊息，它們會保留在 SYSTEM.QSG.TRANSMIT.QUEUE。 然後，使用者必須使用自己的方法來處理這些未處理的訊息。

z/OS 報告訊息-內部群組佇列作業

本主題說明報告訊息: 確認到達、確認遞送、到期報告及異常狀況報告。

確認到達 (COA)/確認遞送 (COD) 報告訊息
使用內部群組佇列作業時，佇列管理程式會產生 COA 及 COD 訊息。

到期報告訊息
到期報告訊息由佇列管理程式產生。

異常狀況報告訊息

視未遞送訊息之訊息描述子的 報告選項 欄位中指定的 MQRO_EXCEPTION_ * 報告選項而定， IGQ 代理程式會產生必要的異常狀況報告，並將它放置在指定的回覆目的地佇列上。 內部群組佇列作業可用來將異常狀況報告遞送至目的地回覆目的地佇列。

報告訊息的持續性與未遞送訊息的持續性相同。 如果 IGQ 代理程式無法解析目的地回覆目的地佇列的名稱，或如果無法將回覆訊息放入傳輸佇列 (用於後續傳送至目的地回覆目的地佇列)，則會嘗試將異常狀況報告放入產生報告訊息之佇列管理程式的無法傳送郵件佇列。 如果無法執行，則未遞送的訊息為:

- 持續， IGQ 代理程式會捨棄異常狀況報告，取消現行訊息批次，並進入重試狀態。 如需相關資訊，請參閱 第 186 頁的『內部群組佇列作業的特定內容』。
- 非持續性， IGQ 代理程式會捨棄異常狀況報告，並繼續處理 SYSTEM.QSG.TRANSMIT.QUEUE。

z/OS 內部群組佇列作業的安全

本主題說明內部群組佇列作業的安全安排。

佇列管理程式屬性 IGQAUT (IGQ 權限) 及 IGQUSER (IGQ 代理程式使用者 ID) 可以設定為控制 IGQ 代理程式開啟目的地佇列時所執行的安全檢查層次。

內部群組佇列作業權限 (IGQAUT)

可以設定 IGQAUT 屬性以指出要執行的安全檢查類型，並因此決定 IGQ 代理程式在建立將訊息放入目的地佇列的權限時要使用的使用者 ID。

IGQAUT 屬性類似於通道定義上可用的 PUTAUT 屬性。

內部群組佇列作業使用者 ID (IGQUSER)

IGQUSER 屬性可用來指定 IGQ 代理程式在建立將訊息放入目的地佇列的權限時要使用的使用者 ID。

IGQUSER 屬性類似於通道定義上可用的 MCAUSER 屬性。

z/OS 內部群組佇列作業的特定內容

本節說明內部群組佇列作業的特定內容，包括物件控點的失效、內部群組佇列作業代理程式的自我回復及重試功能，以及內部群組佇列作業代理程式和序列化。

物件控點失效 (MQRC_OBJECT_CHANGED)

如果在開啟物件之後發現物件的屬性已變更，則佇列管理程式會使物件控點下次使用時具有 MQRC_OBJECT_CHANGED 的物件控點失效。

內部群組佇列作業引進下列物件控點失效規則:

- 如果 SYSTEM.QSG.TRANSMIT.QUEUE ，因為在開啟時內部群組佇列作業是 ENABLED ，但在放置時發現內部群組佇列作業是 DISABLED ，因此佇列管理程式會使物件控點失效，並使具有 MQRC_OBJECT_CHANGED 的放置要求失敗。
- 在開啟處理期間，如果 SYSTEM.QSG.TRANSMIT.QUEUE 未包含在名稱解析路徑中，因為開啟時內部群組佇列作業已停用，但在放置時發現內部群組佇列作業已啟用，則佇列管理程式會使物件控點失效，並使具有 MQRC_OBJECT_CHANGED 的放置要求失敗。
- 如果 SYSTEM.QSG.TRANSMIT.QUEUE 已併入名稱解析路徑中，因為在開啟時已啟用內部群組佇列作業，但 SYSTEM.QSG.TRANSMIT.QUEUE 定義已依放置時間變更，則佇列管理程式會使物件控點失效，並使具有 MQRC_OBJECT_CHANGED 的放置要求失敗。

內部群組佇列作業代理程式的自行回復

如果 IGQ 代理程式異常終止，則會發出 CSQM067E 訊息，且 IGQ 代理程式會重新啟動。

內部群組佇列作業代理程式的重試功能

如果 IGQ 代理程式在存取 SYSTEM.QSG.TRANSMIT.QUEUE (例如，因為未定義它，或已使用不正確的屬性定義它，或因「取得」或其他原因而禁止它)，所以 IGQ 代理程式會進入重試狀態。

IGQ 代理程式會觀察短及長重試次數及間隔。 這些計數和間隔的值無法變更，如下所示:

表 *20:* 短重試計數和長重試間隔值	
常數	**值**
短重試次數	10
短重試間隔	60 秒 = 1 分鐘
長重試次數	999,999,999
長重試間隔	1200 秒 = 20 分鐘

內部群組佇列作業代理程式及序列化

當同層級回復仍在進行中時， IGQ 代理程式嘗試將共用佇列的存取序列化，可能會失敗。

當 IGQ 代理程式處理一或多個共用佇列上未確定的訊息時，如果佇列共用群組中的佇列管理程式失敗， IGQ 代理程式會結束，且會對失敗的佇列管理程式進行共用佇列同層級回復。 因為共用佇列同層級回復是非同步活動，所以在共用佇列同層級回復完成之前，可能會讓失敗的佇列管理程式以及該佇列管理程式的 IGQ 代理程式重新啟動。 這會讓任何已確定的訊息有可能在仍在回復中的訊息之前及之後依序處理。 為了確保訊息未依順序處理， IGQ 代理程式會透過發出 MQCONNX API 呼叫來序列化共用佇列的存取權。

當同層級回復仍在進行中時， IGQ 代理程式嘗試將共用佇列的存取序列化，可能會失敗。 會發出錯誤訊息，且 IGQ 代理程式會進入重試狀態。 當佇列管理程式同層級回復完成時，例如在下次重試時， IGQ 代理程式可以啟動。

z/OS z/OS 上的儲存體管理

IBM MQ for z/OS 需要永久及暫時資料結構，並使用頁集及記憶體緩衝區來儲存此資料。 這些主題提供更多關於 IBM MQ 如何使用這些頁集和緩衝區的詳細資料。

相關概念

第 187 頁的『IBM MQ for z/OS 的頁集』
請利用這個主題來瞭解 IBM MQ for z/OS 如何使用頁面集來儲存訊息。

第 188 頁的『IBM MQ for z/OS 的儲存類別』
儲存類別是一個 IBM MQ for z/OS 概念，可讓佇列管理程式將佇列對映至頁集。 您可以使用儲存類別來控制哪些佇列使用哪些資料集。

第 189 頁的『IBM MQ for z/OS 的緩衝區及緩衝池』
IBM MQ for z/OS 使用緩衝區及緩衝池來暫時快取資料。 請利用這個主題來進一步瞭解如何組織及使用緩衝區。

相關參考

第 190 頁的『在何處尋找 IBM MQ for z/OS 儲存體管理的相關資訊』
請使用本主題作為參考，以尋找 IBM MQ for z/OS 儲存體管理的進一步相關資訊。

z/OS IBM MQ for z/OS 的頁集

請利用這個主題來瞭解 IBM MQ for z/OS 如何使用頁面集來儲存訊息。

頁集 是 VSAM 線性資料集，已特別格式化供 IBM MQ 使用。 頁集用來儲存大部分訊息及物件定義。

這方面的例外是廣域定義 (儲存在 Db2 上的共用儲存庫中)，以及共用佇列上的訊息。 這些未儲存在佇列管理程式頁集上。 如需共用佇列的相關資訊，請參閱 第 139 頁的『共用佇列及佇列共用群組』; 如需廣域定義的相關資訊，請參閱 專用和廣域定義。

IBM MQ 頁面集的大小最多可以為 64 GB。 每一個頁集都由頁集 ID (PSID) 識別，這是 00 到 99 範圍內的整數。 每一個佇列管理程式都必須有自己的頁集。

IBM MQ 使用頁集零 (PSID=00) 來儲存物件定義及與佇列管理程式相關的其他重要資訊。 對於 IBM MQ 的正常作業而言，頁集零不會變滿是很重要的，因此請勿使用它來儲存訊息。

若要改善系統效能，您也應該將短期訊息與長期訊息分開，方法是將它們放在不同的頁集上。

您必須格式化頁集，且 IBM MQ 為此提供 FORMAT 公用程式; 請參閱 格式化頁集 (FORMAT)。 頁集也必須定義給 IBM MQ 子系統。

IBM MQ for z/OS 可以配置成在頁面集已滿時動態展開它。 必要的話， IBM MQ 會繼續擴充頁集，直到有 123 個邏輯延伸範圍存在為止 (如果有足夠的磁碟儲存體空間可用的話)。 如果以這種方式定義線性資料集，則延伸範圍可以跨越磁區，不過， IBM MQ 無法擴充超過 64 GB 的頁集。

您無法使用不同 IBM MQ 佇列管理程式上某個 IBM MQ 佇列管理程式中的頁面集，或變更佇列管理程式名稱。 如果您要將資料從一個佇列管理程式傳送至另一個佇列管理程式，則必須從第一個佇列管理程式卸載所有物件及訊息，並將它們重新載入至另一個佇列管理程式。

在執行早於 V6 的版本的佇列管理程式中，無法使用大於 4 GB 的頁集。 在移轉期間，當您可能需要回復至舊版程式碼時:

- 請勿將頁集 0 變更為大於 4 GB。
- 使用舊版重新啟動佇列管理程式時，大於 4 GB 的其他頁集將會保持離線。

如需移轉可擴充超過 4 GB 之現有頁集的進一步相關資訊，請參閱 將頁集定義為大於 4 GB。

管理者可以將頁面集動態新增至執行中佇列管理程式，或從執行中佇列管理程式移除頁面集 (頁面集零除外)。 只有在指令包含 DSN 關鍵字時，才能在佇列管理程式重新啟動完成之後執行 DEFINE PSID 指令。

z/OS IBM MQ for z/OS 的儲存類別

儲存類別是一個 IBM MQ for z/OS 概念，可讓佇列管理程式將佇列對映至頁集。 您可以使用儲存類別來控制哪些佇列使用哪些資料集。

儲存類別簡介

儲存類別 會將一個以上佇列對映至頁集。 這表示該佇列的訊息會儲存在該頁集上。

儲存類別可讓您控制儲存非共用訊息資料的位置，以進行管理、資料集空間及負載管理或應用程式隔離。 如果您使用 IMS 橋接器 (如 第 236 頁的『IBM MQ 及 IMS』 所述) ，也可以使用儲存類別來定義 IMS 區域的 XCF 群組及成員名稱。

共用佇列不會使用儲存類別來取得頁集對映，因為其中的訊息不會儲存在頁集上。

儲存類別如何運作

- 您可以使用 DEFINE STGCLASS 指令來定義儲存類別，並指定頁集 ID (PSID)。
- 當您定義佇列時，您會在 STGCLASS 屬性中指定儲存類別。

在下列範例中，本端佇列 QE5 透過儲存類別 ARC2 對映至頁集 21。

```
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE QLOCAL(QE5) STGCLASS(ARC2)
```

這表示放置在佇列 QE5 上的訊息會儲存在頁集 21 上 (如果它們停留在佇列上的時間足以寫入 DASD)。

多個佇列可以使用相同的儲存類別，而且您可以定義任意數目的儲存類別。 例如，您可以延伸前一個範例，以包括更多儲存類別及佇列定義，如下所示:

```
DEFINE STGCLASS(ARC1) PSID(05)
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE STGCLASS(MAXI) PSID(05)
DEFINE QLOCAL(QE1) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE2) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE3) STGCLASS(MAXI) ...
DEFINE QLOCAL(QE4) STGCLASS(ARC2) ...
DEFINE QLOCAL(QE5) STGCLASS(ARC2) ...
```

在 第 189 頁的圖 71 中，儲存類別 ARC1 和 MAXI 都與頁集 05 相關聯。 因此，佇列 QE1、 QE2 及 QE3 會對映至頁集 05。 同樣地，儲存類別 ARC2 會將佇列 QE4 及 QE5 與頁集 21 相關聯。

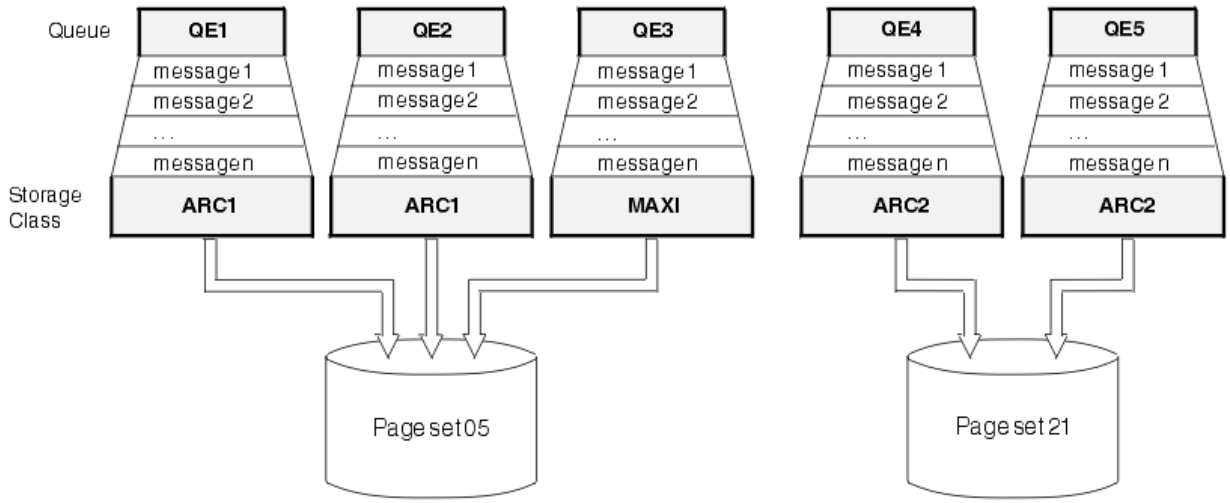


圖 *71:* 透過儲存類別將佇列對映至頁集

如果您定義佇列而未指定儲存類別，則 IBM MQ 會使用預設儲存類別。

如果將訊息放置在名為不存在儲存類別的佇列上，則應用程式會收到錯誤。 您必須變更佇列定義，為它提供現有的儲存類別名稱，或建立佇列指定的儲存類別。

只有在下列情況下，才能變更儲存類別:

- 使用此儲存類別的所有佇列都是空的，且沒有未確定的活動。
- 會關閉所有使用此儲存類別的佇列。

z/OS IBM MQ for z/OS 的緩衝區及緩衝池

IBM MQ for z/OS 使用緩衝區及緩衝池來暫時快取資料。 請利用這個主題來進一步瞭解如何組織及使用緩衝區。

為了提高效率， IBM MQ 會使用快取形式，因此訊息 (及物件定義) 會先暫時儲存在緩衝區中，再儲存在 DASD 上的頁集。 短期訊息 (即在收到之後不久從佇列擷取的訊息) 可能只會儲存在緩衝區中。 這個快取活動是由緩衝區管理程式所控制，緩衝區管理程式是 IBM MQ 的元件。

緩衝區會組織成 緩衝池。 您最多可以為每一個佇列管理程式定義 100 個緩衝池 (0 到 99)。

建議您使用與 第 190 頁的圖 72 中概述的物件及訊息類型分隔一致的緩衝池數目下限，以及應用程式可能具有的任何資料隔離需求。 每一個緩衝區的長度為 4 KB。 依預設，緩衝池會使用 31 位元儲存體，在此模式中，緩衝區數目上限是由佇列管理程式位址空間中可用的 31 位元儲存體數量決定; 緩衝區使用不超過大約 70%。 或者，可以從 64 位元儲存體進行緩衝池儲存體配置 (使用 **DEFINE BUFFPOOL** 指令的 LOCATION 屬性)。 使用 LOCATION (ABOVE) 來使用 64 位元儲存體有兩個好處。 首先，有更多 64 位元儲存體可用，因

此緩衝池可以更大; 其次，有 31 位元儲存體可供其他功能使用。 一般而言，您擁有的緩衝區越多，緩衝越有效率，且 IBM MQ 的效能越好。

第 190 頁的圖 72 顯示訊息、緩衝區、緩衝池及頁集之間的關係。 緩衝池與一或多個頁集相關聯; 每一個頁集與單一緩衝池相關聯。

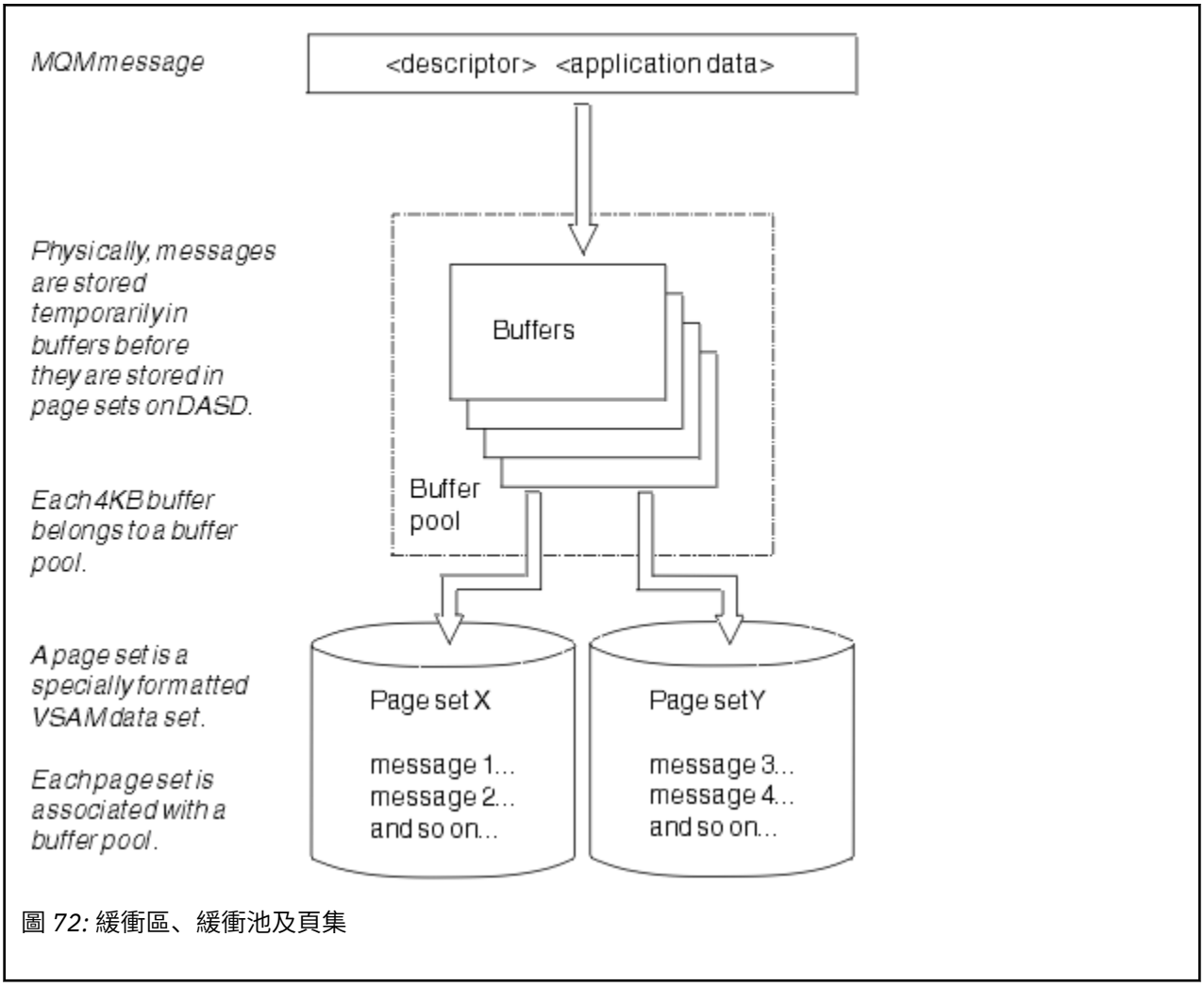


圖 72: 緩衝區、緩衝池及頁集

您可以使用 **ALTER BUFFPOOL** 指令，動態發出指令來修改緩衝池大小及位置。 您可以使用 **DEFINE PSID** 指令來動態新增頁集，或使用 **DELETE PSID** 指令來刪除頁集。

如果緩衝池太小， IBM MQ 會發出訊息 CSQP020E。 然後，您可以動態新增更多緩衝區至受影響的緩衝池 (請注意，您可能必須從其他緩衝池移除緩衝區才能執行此動作)。

您可以使用 **DEFINE BUFFPOOL** 指令來指定儲存區中的緩衝區數目，並且可以使用 **ALTER BUFFPOOL** 指令來動態調整緩衝池的大小。 您可以使用 **DISPLAY USAGE** 指令來顯示使用緩衝池的頁集，以動態地判定儲存區中緩衝區的現行數目。

基於效能原因，請勿將訊息及物件定義放置在相同的緩衝池中。 僅對保留物件定義的頁集零使用一個緩衝池 (例如數字零)。 同樣地，將短期訊息及長期訊息保留在不同的緩衝池中，因此保留在不同的頁集及不同的佇列中。

重新啟動之後，無法使用 **DEFINE BUFFPOOL** 指令來建立新的緩衝池。 相反地，如果 **DEFINE PSID** 指令使用 DSN 關鍵字，它可以明確識別目前未定義的緩衝池。 然後會建立新的緩衝池。

z/OS 在何處尋找 IBM MQ for z/OS 儲存體管理的相關資訊

請使用本主題作為參考，以尋找 IBM MQ for z/OS 儲存體管理的進一步相關資訊。

您可以從下列來源找到本節中主題的相關資訊:

表 21: 何處可找到儲存體管理的相關資訊	
主題	**在何處尋找**
您需要多少儲存體	在 z/OS 上規劃儲存體及效能需求
建立頁面集的大小 及緩衝池	規劃頁面集及緩衝池
管理頁面集	管理頁面集
MQSC 指令	MQSC 指令

z/OS 登入 IBM MQ for z/OS

當發生資料變更及重要事件時， IBM MQ 會維護這些事件的 日誌 。 必要的話，這些日誌可用來將資料回復至前一個狀態。

引導資料集 (BSDS) 會儲存包含日誌之資料集的相關資訊。

日誌不包含統計資料、追蹤資料或效能評估的資訊。 如需 IBM MQ 所收集之統計及監視資訊的進一步詳細資料，請參閱 監視及統計資料。

如需記載的相關資訊，請參閱下列主題:

- 第 191 頁的『IBM MQ for z/OS 中的日誌檔』
- 第 195 頁的『日誌的結構方式』
- 第 195 頁的『如何撰寫 IBM MQ for z/OS 日誌』
- 第 198 頁的『較大的日誌相對位元組位址』
- 第 199 頁的『引導資料集』

相關工作

規劃記載環境

使用系統參數模組來設定日誌

z/OS 管理 z/OS

相關參考

z/OS IBM MQ for z/OS 的訊息

z/OS IBM MQ for z/OS 中的日誌檔

日誌檔包含交易回復所需的資訊。 可以保存現行日誌檔，以便您可以長時間保留日誌資料。

何謂日誌檔

IBM MQ 會記錄在 作用中日誌中發生的所有重要事件。 日誌包含回復所需的資訊:

- 持續訊息
- IBM MQ 物件，例如佇列
- IBM MQ 佇列管理程式

主動日誌包括週期性使用的資料集集合 (最多 310 個)。

您可以啟用日誌保存，以便在作用中日誌填入保存資料集中的副本時。 使用保存可讓您長時間保留日誌資料。 如果您不使用保存，則會改寫日誌折返及先前的資料。 若要回復頁集，或回復 CF 結構中的資料，您需要從建立頁集或結構備份時開始的日誌資料。 可以在磁碟或磁帶上建立保存日誌。

保存

因為作用中日誌具有固定大小，所以 IBM MQ 會定期將每一個日誌資料集的內容複製到 保存日誌，這通常是直接存取儲存裝置 (DASD) 或磁帶上的資料集。 如果子系統或交易失敗， IBM MQ 會使用作用中日誌，並在必要時使用保存日誌進行回復。

保存日誌最多可以包含 1000 個循序資料集。 您可以使用 z/OS 整合型錄機能 (ICF) 來編目每一個資料集。

保存是 IBM MQ 回復的基本元件。 如果回復單元是長時間執行的單元，則在保存日誌中可能會找到該回復單元內的日誌記錄。 在此情況下，回復需要保存日誌中的資料。 不過，如果停用保存，則具有新日誌記錄的作用中日誌會覆蓋，並改寫先前的日誌記錄。 這表示 IBM MQ 可能無法回復回復單元，且訊息可能會遺失。 然後佇列管理程式會異常終止。

因此，在正式作業環境中， **永不關閉保存**。 如果您這麼做，則在系統或交易失敗之後，會有遺失資料的風險。 只有在測試環境中執行時，您才可以考慮關閉保存。 如果您需要這麼做，請使用 CSQ6LOGP 巨集，如使用 CSQ6LOGP 中所述。

為了協助防止發生意外長時間執行工作單元的問題， IBM MQ 會發出訊息 (CSQJ160I 或 CSQJ161I) 如果在作用中日誌卸載處理期間偵測到長時間執行的工作單元。

雙重記載

在雙重記載中，每一個日誌記錄都會寫入兩個不同的作用中日誌資料集，以將重新啟動期間發生資料流失問題的可能性降至最低。

您可以將 IBM MQ 配置成使用 單一記載 或 雙重記載來執行。 使用單一記載，日誌記錄會寫入作用中日誌資料集一次。 每一個作用中日誌資料集都是單一延伸範圍 VSAM 線性資料集 (LDS)。 使用雙重記載，每一個日誌記錄都會寫入兩個不同的作用中日誌資料集。 雙重記載可將重新啟動期間資料流失問題的可能性降到最低。

日誌分流

日誌分流會導致部分工作單元的日誌記錄進一步向下寫入日誌。 對於長時間執行或長期不確定的工作單元，這會減少在佇列管理程式重新啟動或取消時必須讀取的日誌資料量。

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。 此技術稱為 分流。 已處理整個工作單元時，工作單元會處於 已延遲 狀態。 與已延遲工作單元相關的任何取消或重新啟動活動都可以使用已延遲日誌記錄，而不是使用原始工作日誌記錄單元。

偵測長時間執行的工作單元是檢查點處理程序的功能。 在檢查點時間，會檢查每一個作用中工作單元，以確定是否需要延遲。 如果工作單元自建立以來，或自前次延遲以來，已通過兩個先前的檢查點，則工作單元適合延遲。 這表示單一工作單元可能會被多次延遲。 這稱為 多重延遲 工作單元。

每三個檢查點就會分流一個工作單元。 不過，會對日誌切換 (或寫入日誌記錄導致超出 LOGLOAD) 非同步執行檢查點。

一次只會執行單一檢查點，因此在檢查點完成之前可能會有多個日誌切換。

這表示如果沒有足夠的作用中日誌，或作用中日誌太小，則在填滿所有日誌之前，可能無法完成大型工作單元的分流。

如果無法完成分流，則會產生訊息 CSQR027I 。

如果日誌保存已關閉，則當嘗試取消擱置失敗的工作單元時，會發生 ABEND 5C6 ，原因為 00D1032A 。 若要避免此問題，您應該使用 OFFLOAD=YES。

日誌分流一律處於作用中，不論是否啟用日誌保存，都會執行。

註: 雖然工作單元的所有日誌記錄都會延遲，但每一筆記錄的整個內容不會延遲，只會延遲取消所需的部分。 這表示寫入的日誌資料量會維持在最小，且如果發生頁集失敗，則無法使用延遲的記錄。 長時間執行工作單元是指已執行超過三個佇列管理程式檢查點的工作單元。

如需日誌分流的相關資訊，請參閱 管理日誌。

日誌壓縮

您可以配置 IBM MQ for z/OS，以在寫入及讀取日誌資料集時壓縮及解壓縮日誌記錄。

日誌壓縮可用來減少專用佇列上持續訊息寫入日誌的資料量。 達到的壓縮量取決於訊息內包含的資料類型。 例如，「執行長度編碼 (RLE)」的運作方式是壓縮重複的位元組實例，這可以有效地為結構化或記錄導向資料提供良好的結果。

小心: 放置到共用佇列的持續訊息不會受到日誌壓縮的限制。

您可以使用「系統管理機能 115 (SMF)」記錄的「日誌管理程式」區段內的欄位，來監視達到多少資料壓縮。 如需 SMF 的相關資訊，請參閱 使用系統管理機能 和 結算和統計資料訊息。

日誌壓縮會增加系統的處理器使用率。 只有在佇列管理程式的傳輸量受限於寫入日誌資料集的 IO 頻寬，或受限於保留日誌資料集所需的磁碟儲存體時，才應該考慮使用壓縮。 如果您使用共用佇列，則可以透過將其他佇列管理程式新增至佇列共用群組，並將工作量分散至更多佇列管理程式，來解除 IO 頻寬限制。

可以視需要啟用及停用日誌壓縮選項，而不需要停止並重新啟動佇列管理程式。 不論現行日誌壓縮設定為何，佇列管理程式都可以讀取任何壓縮日誌記錄。

佇列管理程式支援 3 個日誌壓縮設定。

無
不使用日誌資料壓縮。 這是預設值。

RLE
使用執行長度編碼 (RLE) 來執行日誌資料壓縮。

ANY
啟用佇列管理程式，以選取提供最大日誌記錄壓縮程度的壓縮演算法。 此選項會導致 RLE 壓縮。

您可以使用下列其中一項來控制日誌記錄的壓縮:

- MQSC 中的 SET 和 DISPLAY LOG 指令; 請參閱 SET LOG 和 DISPLAY LOG
- PCF 介面中的「設定日誌」及「查詢日誌」函數; 請參閱 設定日誌 及 查詢日誌
- 系統參數模組中的 CSQ6LOGP 巨集; 請參閱 使用 CSQ6LOGP

此外，「日誌列印」公用程式 CSQ1LOGP 也支援擴充任何壓縮日誌記錄。

日誌資料

日誌最多可以包含 1,800 萬百萬 ($1.8 * 10^{19}$) 位元組。 每一個位元組都可以透過其從日誌開頭開始的偏移來定址，該偏移稱為其 相對位元組位址 (RBA)。

RBA 由 6 個位元組或 8 個位元組欄位參照，提供可定址範圍總計 2^{48} 個位元組或 2^{64} 個位元組，視使用中的是 6 個位元組還是 8 個位元組日誌 RBA 而定。

不過，當 IBM MQ 偵測到已使用的範圍超出 F00000000000 (如果 6 個位元組的 RBA 在使用中) 或 FFFF800000000000 (如果 8 個位元組的日誌 RBA 在使用中) 時，會發出訊息 CSQI045、 CSQI046、CSQI047 及 CSQJ032，警告您重設日誌 RBA。

如果 RBA 值達到 FFF800000000 (如果 6 個位元組的日誌 RBA 在使用中) 或 FFFFFFC000000000 (如果 8 個位元組的日誌 RBA 在使用中)，則佇列管理程式會終止，原因碼為 00D10257。

發出關於已使用日誌範圍的警告訊息之後，您應該規劃佇列管理程式中斷，在此期間可以將佇列管理程式轉換為使用 8 位元組日誌 RBA，或者可以重設日誌。 重設佇列管理程式的日誌中記載了重設日誌的程序。

如果您的佇列管理程式使用 6 位元組日誌 RBA，請遵循 實作較大的日誌相對位元組位址中所記載的程序，考慮將佇列管理程式轉換成使用 8 位元組日誌 RBA，而不是重設佇列管理程式的日誌。

日誌由 日誌記錄組成，每一個記錄都是視為單一單元的一組日誌資料。 日誌記錄由其標頭第一個位元組的 RBA 識別，或由其日誌記錄序號 (LRSN) 識別。 RBA 或 LRSN 可唯一識別在日誌中特定點開始的記錄。

您是否使用 RBA 或 LRSN 來識別日誌點，取決於您是否使用佇列共用群組。 在佇列共用環境中，您無法使用相對位元組位址來唯一識別日誌點，因為多個佇列管理程式可以同時更新相同的佇列，且每個佇列管理程式都有自己的日誌。 為了解決此問題，日誌記錄序號衍生自時間戳記值，且不一定代表日誌內日誌記錄的實體位移。

每一個日誌記錄都有一個標頭，提供其類型、建立記錄的 IBM MQ 子元件，以及回復單元記錄的回復單元 ID。

日誌記錄有四種類型，如下列標題所說明:

- 回復單元日誌記錄
- 檢查點記錄
- 頁集控制記錄
- CF 結構備份記錄

回復日誌記錄的單元

大部分日誌記錄說明 IBM MQ 佇列的變更。 所有這類變更都在回復單元內進行。

IBM MQ 使用特殊記載技術，包括 復原/重做 及 補償日誌記錄 ，以減少重新啟動時間並增進系統可用性。

其中一個效果是重新啟動時間有限。 如果在重新啟動期間發生失敗，因此必須第二次重新啟動佇列管理程式，則在第二次重新啟動期間，不需要重新套用完成至第一次重新啟動失敗點的所有回復活動。 這表示連續重新啟動不會花費逐漸較長的時間來完成。

檢查點記錄

為了減少重新啟動時間，在正常作業期間， IBM MQ 會採取定期檢查點。 這些發生方式如下:

- 已寫入預先定義數目的日誌記錄時。 此數字由系統參數巨集 CSQ6SYSP 的檢查點頻率運算元 LOGLOAD 所定義，如 使用 CSQ6SYSP 中所述。
- 在順利重新啟動結束時。
- 在正常終止時。
- 每當 IBM MQ 切換至循環中的下一個作用中日誌資料集時。

在取得檢查點時， IBM MQ 會發出 DISPLAY CONN 指令 (如 DISPLAY CONN 所述) 內部，以便將目前不確定的連線清單寫入 z/OS 主控台日誌。

頁集控制記錄

這些記錄會在每一個檢查點登錄 IBM MQ 佇列管理程式已知的頁集及緩衝池，並記錄在檢查點時執行頁集之媒體回復所需的日誌範圍相關資訊。

頁集及緩衝池的某些動態變更也會寫入為頁集控制記錄，以便在下一次佇列管理程式重新啟動時可以回復及自動恢復這些變更。

CF 結構備份記錄

這些記錄保留從連結機能清單結構讀取的資料，以回應 BACKUP CFSTRUCT 指令。 在連結機能結構失敗的不太可能事件中， RECOVER CFSTRUCT 指令會將這些記錄與回復單元記錄一起使用，以執行連結機能結構到失敗點的媒體回復。

相關工作

實作較大的日誌相對位元組位址

z/OS 日誌的結構方式

請利用這個主題來瞭解用來說明日誌記錄的術語。

每一個作用中日誌資料集都必須是 VSAM 線性資料集 (LDS)。 寫入作用中日誌資料集的實體輸出單元是 4 KB 控制間隔 (CI)。 每一個 CI 都包含一筆 VSAM 記錄。

實體和邏輯日誌記錄

一個 VSAM CI 是 實體 記錄。 在特定時間記載的資訊會形成 邏輯 記錄，其長度與 CI 中可用的空間無關。 因此，一個實體記錄可能包含:

- 數筆邏輯記錄
- 一個以上邏輯記錄及另一個邏輯記錄的一部分
- 僅屬於一個邏輯記錄的一部分

術語 日誌記錄 是指 邏輯 記錄，不論儲存它需要多少 實體 記錄。

z/OS 如何撰寫 IBM MQ for z/OS 日誌

請利用這個主題來瞭解 IBM MQ 如何處理日誌檔記錄。

IBM MQ 會將每一筆日誌記錄寫入稱為 作用中日誌的 DASD 資料集。 當作用中日誌已滿時， IBM MQ 會將其內容複製到稱為 保存日誌的 DASD 或磁帶資料集。 此程序稱為 卸載。

第 196 頁的圖 73 說明記載的程序。 日誌記錄通常會經歷下列循環:

1. IBM MQ 會在回復日誌記錄中記錄對資料及重要事件的變更。
2. IBM MQ 會處理回復日誌記錄，並在必要時將它們分成區段。
3. 日誌記錄會循序放置在 輸出日誌緩衝區中，其格式為「VSAM 控制間隔 (CI)」。 每筆日誌記錄都由範圍從零到 2^{64} -1 的相對位元組位址識別。
4. CI 會寫入一組預先定義的 DASD 作用中日誌資料集，這些資料集會循序使用並回收。
5. 如果保存處於作用中，則當每一個作用中日誌資料集已滿時，其內容會自動卸載至新的保存日誌資料集。

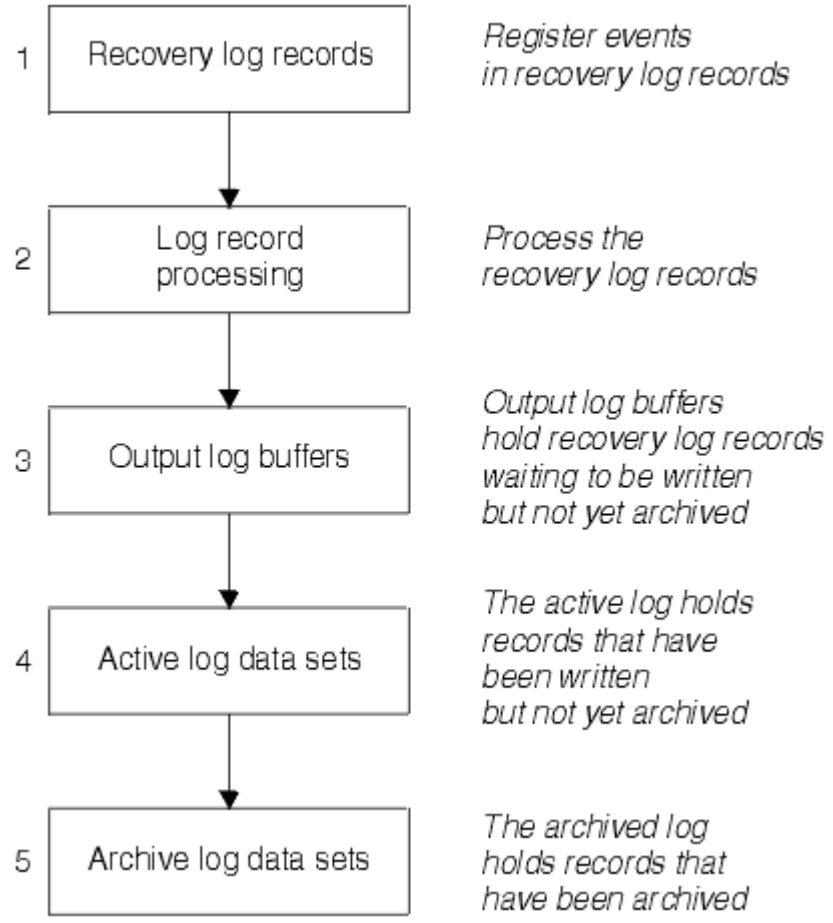


圖 *73:* 記載程序

寫入作用中日誌時

每當發生下列任何情況時，都會將儲存體內日誌緩衝區寫入作用中日誌資料集:

- 日誌緩衝區會變滿。
- 達到寫入臨界值 (如 CSQ6LOGP 巨集中所指定)。
- 某些重要事件會發生，例如確定點，或發出 IBM MQ BACKUP CFSTRUCT 指令時。

當佇列管理程式起始設定時， BSDS 中指定的作用中日誌資料集會動態配置供佇列管理程式專用，並維持專門配置給 IBM MQ ，直到佇列管理程式終止為止。

動態新增日誌資料集

在佇列管理程式執行時，可以動態定義新的作用中日誌資料集。 此特性可緩解保存因暫時性問題而無法卸載作用中日誌時佇列管理程式當掉的問題。 如需相關資訊，請參閱 DEFINE LOG 指令。

註: 若要重新定義或移除作用中日誌，您必須終止並重新啟動佇列管理程式。

IBM MQ 及儲存體管理子系統

IBM MQ 參數可讓您在動態配置 IBM MQ 保存日誌資料集時指定「儲存體管理子系統 (MVS/DFP SMS)」儲存類別。 IBM MQ 會起始保存日誌資料集，但您可以使用 SMS 來執行保存資料集的配置。

相關參考

第 197 頁的『寫入 IBM MQ for z/OS 保存日誌時』
請利用這個主題來瞭解將作用中日誌複製到保存日誌的處理程序，以及處理程序發生的時間。

z/OS 寫入 *IBM MQ for z/OS* 保存日誌時

請利用這個主題來瞭解將作用中日誌複製到保存日誌的處理程序，以及處理程序發生的時間。

將作用中日誌複製到保存日誌的處理程序稱為 卸載。 卸載至其他記載事件的關係在 第 197 頁的圖 74 中以圖表方式顯示。

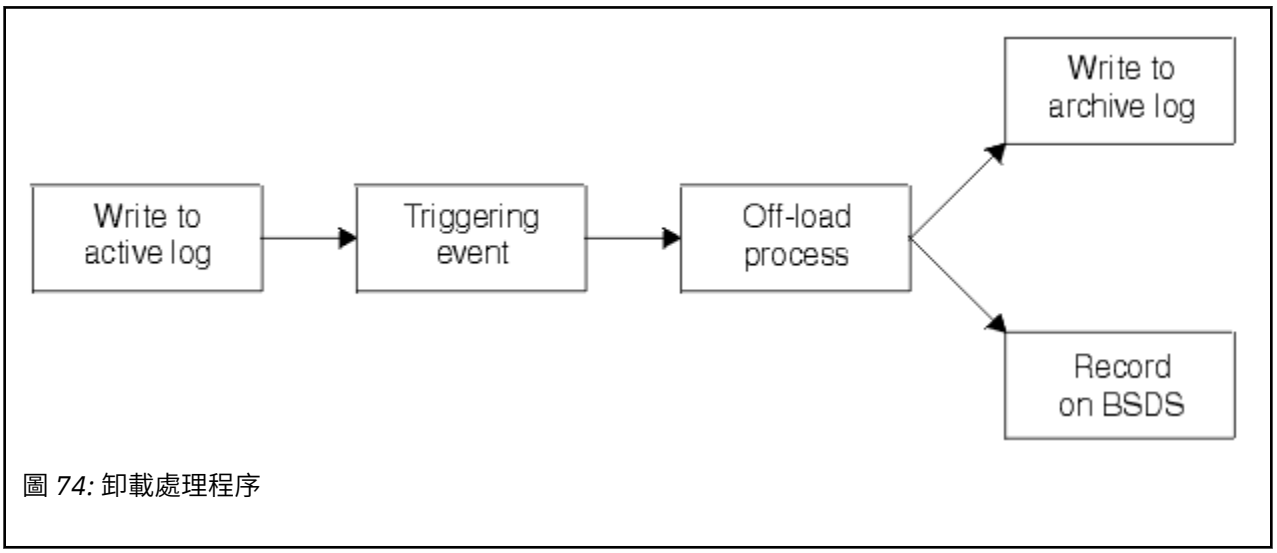


圖 *74:* 卸載處理程序

觸發卸載程序

數個事件可以觸發作用中日誌至保存日誌的卸載處理程序。 例如:

- 填入作用中日誌資料集。
- 使用 MQSC ARCHIVE LOG 指令。
- 寫入作用中日誌資料集時發生錯誤。

在失敗點之前會截斷資料集，且未寫入的記錄會變成新資料集的第一筆記錄。 會針對截斷的資料集觸發卸載，就像一般完整日誌資料集一樣。 如果有雙重作用中日誌，則會截斷兩個副本，使兩個副本保持同步。

當最後一個可用的作用中日誌已滿 5% ，之後增量為 5% 時，會發出 CSQJ110E 訊息，指出使用中日誌容量的百分比。 如果所有作用中日誌都已滿， IBM MQ 會停止處理，直到發生卸載為止，並發出下列訊息:

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

卸載處理程序

當所有作用中日誌都已滿時， IBM MQ 會執行卸載處理程序，並中止處理程序，直到卸載處理程序完成為止。 當作用中日誌已滿時，如果卸載處理失敗， IBM MQ 會異常終止。

當作用中日誌備妥可卸載時，會傳送要求給 z/OS 主控台操作員，以裝載磁帶或準備 DASD 裝置。 ARCWTOR 記載選項的值 (如需進一步資訊，請參閱 使用 CSQ6ARVP) 決定是否接收要求。 如果您使用磁帶來卸載，請指定 ARCWTOR=YES。 如果值為 YES ，則要求之前會有 WTOR (訊息號碼 CSQJ008E) 告訴操作員準備要配置的保存日誌資料集。

操作員不需要立即回應此訊息。 不過，延遲回應會延遲卸載處理程序。 除非操作員將回應延遲到 IBM MQ 用完作用中日誌為止，否則不會影響 IBM MQ 效能。

操作員可以透過取消卸載處理程序來回應。 在此情況下，如果配置是針對雙重保存資料集的第一份副本，則卸載處理程序只會延遲到下一個作用中日誌資料集已滿為止。 如果配置用於第二個副本，則保存處理程序會切換至單一副本模式，但僅適用於此資料集。

卸載時發生岔斷及錯誤

在卸載處理完成之前，停止佇列管理程式的要求不會生效。 如果 IBM MQ 在卸載進行中失敗，則在佇列管理程式重新啟動時，會重新開始卸載。

卸載處理期間的訊息

IBM MQ 及卸載處理程序會將卸載訊息傳送至 z/OS 主控台。 您可以使用這些訊息來尋找各種日誌資料集中的 RBA 範圍。

z/OS 較大的日誌相對位元組位址

此功能會在您必須重設日誌之前增加一段時間，以改善佇列管理程式的可用性。

回復資料會寫入日誌中，以便在佇列管理程式重新啟動時可以使用持續訊息。 術語日誌「相對位元組位址 (log RBA)」用於將資料位置作為與日誌開頭的偏移來參照。

在 IBM MQ 8.0 之前， 6 個位元組的日誌 RBA 最多可以處理 256 TB 的資料。 在寫入此數量的日誌記錄之前，您必須遵循 重設佇列管理程式的日誌中所記載的程序來重設佇列管理程式的日誌。

重設佇列管理程式的日誌不是快速處理程序，而且可能需要長時間中斷，因為需要在處理程序中重設頁面集。 對於高使用率佇列管理程式，此作業通常每年執行一次。

從 IBM MQ 8.0 開始，日誌 RBA 可以有 8 個位元組的長度，現在佇列管理程式可以在需要重設日誌 RBA 之前定址超過 64,000 倍的資料 (16 EB)。 使用較大的日誌 RBA 的影響是寫入的日誌資料大小會增加幾個位元組。

何時啟用此功能?

V 9.3.0 在 IBM MQ 9.3.0 或更新版本建立的佇列管理程式已啟用此功能。

如果現行日誌 RBA 接近日誌 RBA 範圍的結尾，請考量將佇列管理程式轉換為使用 8 個位元組的日誌 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，而且在您必須重設日誌之前會大幅增加時段。

在佇列管理程式起始設定期間發出的訊息 CSQJ034I 指出所配置佇列管理程式的日誌 RBA 範圍結束，並可用來判斷是否使用 6 個位元組或 8 個位元組的日誌 RBA。

如何啟用此功能?

以第 2 版格式 BSDS 啟動佇列管理程式，即可啟用 8 位元組日誌 RBA。 總之，實現這一目標的辦法是:

1. 確保佇列共用群組中的所有佇列管理程式都符合啟用 8 個位元組日誌 RBA 的需求
2. 完全關閉佇列管理程式
3. 執行 BSDS 轉換公用程式 ，以建立第 2 版格式的 BSDS 副本。
4. 以已轉換的 BSDS 重新啟動佇列管理程式。

一旦佇列管理程式已轉換為使用 8 個位元組的日誌 RBA ，就無法回復為使用 6 個位元組的日誌 RBA。

如需如何啟用 8 位元組日誌 RBA 的詳細程序，請參閱 實作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 引導資料集

IBM MQ 需要引導資料集作為參照日誌資料集和日誌記錄的機制。 在正常處理及重新啟動回復期間，需要此資訊。

引導資料集的目的

引導資料集 (BSDS) 是 VSAM 鍵序資料集 (KSDS)，用於保留 IBM MQ 所需的資訊。 它包含下列:

- IBM MQ 已知的所有作用中和保存日誌資料集的庫存。 IBM MQ 使用此庫存來執行下列動作:
  - 追蹤作用中和保存日誌資料集
  - 尋找日誌記錄，以便它可以在正常處理期間滿足日誌讀取要求
  - 尋找日誌記錄，以便它可以處理重新啟動處理程序

  每次定義保存日誌資料集或重複使用作用中日誌資料集時， IBM MQ 都會將資訊儲存在庫存中。 對於作用中日誌，庫存會顯示哪些已滿以及哪些可供重複使用。 庫存會保留該資料集中所保留日誌的每一個部分的相對位元組位址 (RBA)。
- 所有最近 IBM MQ 活動的 環繞 庫存。 如果您必須重新啟動佇列管理程式，則需要這樣做。

如果佇列管理程式發生錯誤，且您必須重新啟動它，則需要 BSDS。 IBM MQ **必須** 具有 BSDS。 若要將重新啟動期間發生問題的可能性降至最低，您可以使用雙重 BSDS 來配置 IBM MQ，每一個都會記錄相同的資訊。 使用雙重 BSDS 稱為在 雙重模式中執行。 可能的話，請將副本放在個別磁區上。 這可減少磁區毀損或毀損時兩者遺失的風險。 使用雙重 BSDS，而不是雙重寫入 DASD。

當自訂 IBM MQ 且您可以使用變更日誌庫存公用程式 (CSQJU003) 來管理庫存時，會設定 BSDS。 如需此公用程式的相關資訊，請參閱管理 IBM MQ for z/OS。 佇列管理程式啟動程序中的 DD 陳述式會參照它。

一般而言， IBM MQ 會保留 BSDS 的重複副本。 如果發生 I/O 錯誤，它會取消配置失敗的副本，並繼續執行單一 BSDS。 您可以還原雙重模式作業，這在 管理 IBM MQ for z/OS 中有所說明。

安裝 IBM MQ 時，會先在 BSDS 中登錄作用中日誌。 如果不終止並重新啟動佇列管理程式，則無法取代作用中日誌。

保存日誌資料集是動態配置的。 配置時，會在 BSDS 中登錄資料集名稱。 保存日誌資料集清單會在新增保存檔時展開，並在達到使用者決定的項目數時覆蓋。 單一保存記載的項目數上限是 1000，雙重記載則是 2000。

您可以使用磁帶管理系統來刪除保存日誌資料集 (IBM MQ 沒有自動化方法)。 因此，保存日誌資料集的相關資訊可以在系統管理者刪除保存日誌資料集之後很久的 BSDS 中。

相反地，可能已超出保存日誌資料集的數目上限，且 BSDS 中的資料在資料集達到其到期日之前很久被捨棄。

您可以使用下列 MQSC 指令來判斷日誌的範圍，以及各種媒體或佇列管理程式回復所需的作用中或保存日誌資料集 (保留最早日誌 RBA) 的名稱:

```
DISPLAY USAGE TYPE(DATASET)
```

如果系統參數模組指定在配置時編目保存日誌資料集，則 BSDS 會指向整合型錄機能 (ICF) 型錄，以取得稍後配置所需的資訊。 否則，每一個磁區的 BSDS 項目會登錄稍後配置所需的磁區序號及裝置資訊。

BSDS 版本

BSDS 的格式根據其版本而有所不同。 增加 BSDS 版本可讓您使用新特性。 IBM MQ 支援下列 BSDS 版本:

第 1 版
受所有 IBM MQ 版本支援。 第 1 版 BSDS 支援 6 個位元組的日誌 RBA 值。

第 2 版
受 IBM MQ 8.0 以及更新版本支援。 第 2 版 BSDS 會啟用 8 個位元組的日誌 RBA 值，且在每一個作用中日誌副本中最多可以有 310 個資料集。

V 9.3.0 依預設，會針對在 IBM MQ 9.3.0 以及更新版本建立的佇列管理程式啟用。

第 3 版
受 IBM MQ 8.0 以及更新版本支援。 當超過 31 個資料集新增至任一作用中日誌副本時， BSDS 會自動從第 2 版轉換至第 3 版。

您可以執行列印日誌對映公用程式 (CSQJU004)來判斷 BSDS 的版本。 若要將 BSDS 從第 1 版轉換至第 2 版，請執行 BSDS 轉換公用程式 (CSQJUCNV)。

如需 6 位元組及 8 位元組日誌 RBA 的相關資訊，請參閱 第 198 頁的『較大的日誌相對位元組位址』 。

保存日誌資料集和 BSDS 副本

每次建立新的保存日誌資料集時，也會建立 BSDS 的副本。 如果保存日誌位於磁帶上，則 BSDS 是第一個輸出磁區上的第一個資料集。 如果保存日誌位於 DASD 上，則 BSDS 是個別資料集。

保存日誌與 BSDS 副本的資料集名稱相同，但保存日誌名稱的最低層次限定元以 A 開頭，而 BSDS 副本以 B 開頭，例如:

保存日誌名稱
CSQ.ARCHLOG1.E00186.T2336229. *A* 0000001

BSDS 副本名稱
CSQ.ARCHLOG1.E00186.T2336229. *B* 0000001

如果複製 BSDS 時發生讀取錯誤，則不會建立副本，會發出 CSQJ125E 訊息，且在沒有 BSDS 副本的情況下會繼續卸載至新的保存日誌資料集。

z/OS z/OS 上的系統定義

IBM MQ for z/OS 使用許多預設物件定義，並提供範例 JCL 來建立那些預設物件。 請利用這個主題來瞭解這些預設物件和範例 JCL。

設定系統參數

在 IBM MQ for z/OS 中，系統參數模組會控制 IBM MQ 在其作業中使用的記載、保存、追蹤及連線環境。 系統參數由三個組譯器巨集指定，如下所示:

CSQ6SYSP
系統參數，包括設定連線及追蹤環境。

CSQ6LOGP
記載參數。

CSQ6ARVP
日誌保存參數。

IBM MQ for z/OS 提供預設參數模組。 如果這些沒有包含您要使用的值，您可以使用 IBM MQ 隨附的範例來建立自己的參數模組。 範例為 `thlqual.SCSQPROC(CSQ4ZPRM)`。

您可以在佇列管理程式執行時變更部分系統參數。 請參閱 MQSC 指令中的 SET SYSTEM、SET LOG 及 SET ARCHIVE 指令。

如需定義的相關資訊，請參閱下列主題:

- 第 201 頁的『定義 IBM MQ for z/OS 的系統物件』
- 第 204 頁的『在 IBM MQ for z/OS 上調整佇列管理程式』
- 第 205 頁的『IBM MQ for z/OS 提供的範例定義』

相關概念
自訂範例起始設定輸入資料集
您可以從中在 IBM MQ for z/OS 上發出 MQSC 及 PCF 指令的來源
相關工作
管理 z/OS

z/OS 定義 IBM MQ for z/OS 的系統物件

IBM MQ for z/OS 需要其他預先定義的物件，以用於發佈/訂閱應用程式、叢集及通道控制，以及其他系統管理功能。

IBM MQ for z/OS 所需的系統物件可以分為下列種類:

- 發佈/訂閱物件
- 系統預設物件
- 系統指令物件
- 系統管理物件
- 通道佇列
- 叢集佇列數
- 佇列共用群組佇列
- 儲存類別
- 定義系統物件無法傳送郵件的佇列
- 預設傳輸佇列
- 內部佇列
- 第 204 頁的『通道鑑別佇列』

發佈/訂閱物件

您需要先定義數個系統物件，然後才能將發佈/訂閱應用程式與 IBM MQ for z/OS 搭配使用。 IBM MQ 提供範例定義來協助您定義這些物件。 這些範例在 CSQ4INSG 中說明。

若要使用發佈/訂閱，您需要定義下列物件:

- 稱為 SYSTEM.RETAINED.PUB.QUEUE，用來保留佇列管理程式中每一個保留發佈資訊的副本。 每一個完整主題名稱最多可以在此佇列上儲存一個保留的發佈資訊。 如果您的應用程式將使用許多不同主題的保留發佈資訊，或您的保留發佈資訊是大型訊息，則應該小心規劃此佇列的儲存需求，包括將它指派給自己的頁面集 (如果它的儲存需求很大)。 若要增進效能，您應該使用 MSGID 的索引類型來定義此佇列 (如所提供的範例佇列定義所示)。
- 稱為 SYSTEM.DURABLE.SUBSCRIBER.QUEUE，用來保留佇列管理程式中可延續訂閱的持續性副本。 若要增進效能，您應該使用索引類型 CORRELID 來定義此佇列 (如所提供的範例佇列定義所示)。
- 稱為 SYSTEM.DURABLE.MODEL.QUEUE，用作受管理可延續訂閱的模型。
- 稱為 SYSTEM.NDURABLE.MODEL.QUEUE，用作受管理不可延續訂閱的模型。
- 稱為 SYSTEM.QPUBSUB.QUEUE.NAMELIST，包含佇列發佈/訂閱介面所監視的佇列名稱清單。
- 稱為 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST，包含已排入佇列的發佈/訂閱介面用來比對主題物件與訂閱點的主題物件清單。
- 稱為 SYSTEM.BASE.TOPIC，用作解析屬性的基本主題。
- 稱為 SYSTEM.BROKER.DEFAULT.STREAM，這是排入佇列的發佈/訂閱介面所使用的預設串流。
- 稱為 SYSTEM.BROKER.DEFAULT.SUBPOINT，這是排入佇列的發佈/訂閱介面所使用的預設 RFH2 訂閱點。
- 稱為 SYSTEM.BROKER.ADMIN.STREAM，這是已排入佇列的發佈/訂閱介面所使用的管理串流。
- 稱為 SYSTEM.DEFAULT.SUB，這是預設訂閱物件，用來提供 DEFINE SUB 指令的預設值。

系統預設物件

當您定義物件時，系統預設物件會用來提供預設屬性，且不會指定另一個物件的名稱作為定義的基礎。

預設系統物件定義的名稱以字元 "SYSTEM.DEFAULT"或"SYSTEM.DEF。" 例如，系統預設本端佇列命名為 SYSTEM.DEFAULT.LOCAL.QUEUE。

這些物件定義這些 IBM MQ 物件的屬性的系統預設值:

- 本端佇列
- 模型佇列
- 別名佇列
- 遠端佇列
- Processes
- 名單
- 通道
- 儲存類別
- 鑑別資訊

共用佇列是本端佇列的特殊類型，因此當您定義共用佇列時，會根據 SYSTEM.DEFAULT.LOCAL.QUEUE。 您需要記住提供「連結機能」結構名稱的值，因為預設定義中未指定該名稱。 或者，您可以定義自己的預設共用佇列定義，以作為共用佇列的基準，讓它們全部繼承必要的屬性。 請記住，您只需要在佇列共用群組中的一個佇列管理程式上定義共用佇列。

系統指令物件

系統指令物件的名稱以字元 SYSTEM.COMMAND。 您必須先定義這些物件，才能使用 IBM MQ 作業及控制台向 IBM MQ 子系統發出指令。

有兩個系統指令物件:

1. 系統指令輸入佇列是本端佇列，在 IBM MQ 指令處理器處理指令之前，會先在其中放置指令。 它必須稱為 SYSTEM.COMMAND.INPUT。 為了與 IBM MQ for Multiplatforms 相容，以及供 IBM MQ Console 和 administrative REST API 使用，也應該定義名為 SYSTEM.ADMIN.COMMAND.QUEUE 的別名。
2. SYSTEM.COMMAND.REPLY.MODEL 是定義系統指令回覆目的地佇列的模型佇列。

有兩個額外物件可供 IBM MQ Explorer 使用:

- SYSTEM.MQEXPLORER.REPLY.MODEL 佇列
- SYSTEM.ADMIN.SVRCONN 通道

SYSTEM.REST.REPLY.QUEUE 是 IBM MQ administrative REST API 所使用的回覆佇列。

通常會使用非持續訊息來傳送指令，因此兩個系統指令物件都應該具有 DEFPSIST (NO) 屬性，以便使用它們的應用程式 (包括提供的應用程式，例如公用程式及作業和控制台) 依預設會取得非持續訊息。 如果您的應用程式使用指令的持續訊息，請設定回覆目的地佇列的 DEFTYPE (PERMDYN) 屬性，因為這類指令的回覆訊息是持續的。

系統管理物件

系統管理物件的名稱以字元 SYSTEM.ADMIN。

有七個系統管理物件:

- SYSTEM.ADMIN.CHANNEL.EVENT 佇列
- SYSTEM.ADMIN.COMMAND.EVENT 佇列
- SYSTEM.ADMIN.CONFIG.EVENT 佇列
- SYSTEM.ADMIN.PERFM.EVENT 佇列

- SYSTEM.ADMIN.QMGR.EVENT 佇列
- SYSTEM.ADMIN.TRACE.ROUTE.QUEUE 佇列
- SYSTEM.ADMIN.ACTIVITY.QUEUE 佇列

通道佇列

若要使用分散式佇列，您需要定義下列物件:

- 名稱為 SYSTEM.CHANNEL.SYNCQ，用來維護通道的序號和邏輯工作單元 ID (LUWID)。 若要增進通道效能，您應該使用 MSGID 的索引類型來定義此佇列 (如所提供的範例佇列定義所示)。
- 名稱為 SYSTEM.CHANNEL.INITQ，用於通道指令。

您無法將這些佇列定義為共用佇列。

叢集佇列數

若要使用 IBM MQ 叢集，您需要定義下列物件:

- 稱為 SYSTEM.CLUSTER.COMMAND.QUEUE，用來在佇列管理程式之間傳達儲存庫變更。 寫入此佇列的訊息包含要套用至儲存庫本端副本之儲存庫資料的更新項目，或儲存庫資料的要求。
- 稱為 SYSTEM.CLUSTER.REPOSITORY.QUEUE，用來保留儲存庫的持續性副本。
- 稱為 SYSTEM.CLUSTER.TRANSMIT.QUEUE，這是叢集中所有目的地的傳輸佇列。 基於效能原因，您應該使用索引類型 CORRELID 來定義此佇列 (如範例佇列定義所示)。

這些佇列通常包含大量訊息。

您無法將這些佇列定義為共用佇列。

佇列共用群組佇列

若要使用共用通道及內部群組佇列作業，您需要定義下列物件:

- 名稱為 SYSTEM.QSG.CHANNEL.SYNCQ，用來保留共用通道的同步化資訊。
- 名稱為 SYSTEM.QSG.TRANSMIT.QUEUE，用作內部群組佇列作業的傳輸佇列。 如果您是在佇列共用群組中執行，則必須定義此佇列，即使您不是使用內部群組佇列作業。

儲存類別

建議您定義下列六個儲存類別。 您必須定義其中四個，因為 IBM MQ 需要它們。 建議使用其他儲存類別定義，因為它們在範例佇列定義中使用。

DEFAULT (必要)
此儲存類別用於所有效能不嚴重且不適合任何其他儲存類別的訊息佇列。 如果您在定義佇列時未指定預設儲存類別，則它也是提供的預設儲存類別。

NODEFINE (必要)
如果未定義您定義佇列時指定的儲存類別，則會使用此儲存類別。

REMOTE (必要)
此儲存類別主要用於傳輸佇列，即具有短期效能關鍵訊息的系統相關佇列。

SYSLNGLV
此儲存類別用於長時間執行的效能關鍵訊息。

SYSTEM (必要)
此儲存類別用於效能重要的系統相關訊息佇列，例如 SYSTEM.CHANNEL.SYNQ 和 SYSTEM.CLUSTER.* 佇列。

SYSVOLAT
此儲存類別用於短期的效能關鍵訊息。

您可以視需要修改其屬性，並新增其他儲存類別定義。

定義系統物件無法傳送郵件的佇列

如果訊息目的地無效，則會使用無法傳送郵件的佇列。 IBM MQ 會將這類訊息放在稱為無法傳送郵件之佇列的本端佇列中。 雖然無法傳送郵件的佇列不是必要項目，但您應該將其視為必要項目，尤其是當您使用分散式佇列或其中一個 IBM MQ 橋接器時。

請 **不要** 將無法傳送郵件的佇列定義為共用佇列。 放置到一個佇列管理程式上的本端佇列可能會放置到無法傳送的郵件佇列。 如果無法傳送的郵件佇列是共用佇列，則不同系統上的無法傳送的郵件佇列處理程式可以處理訊息，並將其放置在具有相同名稱的佇列上，但因為這是在不同佇列管理程式上，它會是錯誤佇列，或具有不同的安全設定檔。 如果佇列不存在，則無法重新處理它。

如果您決定定義無法傳送郵件的佇列，也必須告知佇列管理程式其名稱。 若要執行此動作，請使用 ALTER QMGR DEADQ (*queue-name*) 指令。 如需相關資訊，請參閱 顯示及變更佇列管理程式屬性。

預設傳輸佇列

當沒有其他適當的傳輸佇列可用來傳送訊息至另一個佇列管理程式時，會使用預設傳輸佇列。 如果您定義預設傳輸佇列，則必須也定義要提供佇列的通道。 如果您沒有執行這個動作，則不會將放到預設傳輸佇列的訊息傳輸至遠端佇列管理程式，而會保留在佇列上。

如果您決定定義預設傳輸佇列，也必須告知佇列管理程式其名稱。 若要執行此動作，請使用 ALTER QMGR 指令。

內部佇列

- **擱置資料佇列**
  - 定義供內部使用的佇列 SYSTEM.PENDING.DATA.QUEUE，支援在 JMS 發佈/訂閱環境中使用可延續訂閱。
- **JMS 2.0 遞送延遲暫置佇列**
  - 如果使用 JMS 2.0 提供的遞送延遲功能，則會使用內部暫置佇列 SYSTEM.DDELAY.LOCAL.QUEUE。 佇列管理程式會使用此佇列來暫時儲存以非零遞送延遲傳送的訊息，直到遞送延遲完成，並將訊息放置到其目標目的地為止。 在 CSQ4INSG 中提供此佇列的範例定義 (已註銷)。
  - 當您定義 SYSTEM.DDELAY.LOCAL.QUEUE 佇列，您必須針對將以遞送延遲傳送的預期訊息數設定 STGCLASS、MAXMSGL 及 MAXDEPTH 屬性。 此外，當定義 SYSTEM.DDELAY.LOCAL.QUEUE 佇列可確保只有佇列管理程式可以將訊息放入此佇列。 請小心確定沒有任何使用者 ID 具有將訊息放入此佇列的權限。

通道鑑別佇列

如需內部使用通道鑑別，請參閱 SYSTEM.CHLAUTH.DATA.QUEUE 佇列。 IBM MQ 提供範例定義來協助您定義這些物件。 此範例在 CSQ4INSA 中說明，其中也定義部分預設規則。

z/OS 在 IBM MQ for z/OS 上調整佇列管理程式

您可以採取幾個簡單步驟來確保佇列管理程式已調整，以避免基本效能問題。

您可以使用多種方法來增進佇列管理程式的效能，這些方法是由 ALTER QMGR 指令所設定的佇列管理程式屬性所控制。 本節包含如何透過設定佇列管理程式上容許的訊息數上限，或透過對佇列管理程式執行「內部管理」來執行此動作的相關資訊。 IBM MQ SupportPac MP16 - IBM MQ for z/OS 產能規劃與調整 提供效能及調整的相關資訊。

同步點

佇列管理程式的其中一個角色是應用程式內的同步點控制。 應用程式建構的工作單元包含以 MQCMIT 呼叫終止的任何 MQPUT 或 MQGET 呼叫數。

當一個 MQCMIT 範圍內的 MQPUT 或 MQGET 呼叫數增加時，確定的效能成本會大幅增加。 一般而言，應用程式應該設計成在單一同步點中不使用 MQPUT/MQGET 大量訊息。

您可以使用 MAXUMSGS 佇列管理程式屬性，以管理方式限制任何單一同步點內的訊息數。 如果應用程式超出此限制，則會在超出限制的 MQPUT,MQPUT1 或 MQGET 呼叫上收到 MQRC_SYNCPOINT_LIMIT_REACHED。 然後，應用程式應該適當地發出 MQCMIT 或 MQBACK。

MAXUMSGS 的預設值為 10000。 如果您想要施行下限，則可以降低此值，這也有助於防止應用程式迴圈。 在減少 MAXUMSGS 之前，請確定您瞭解現有的應用程式，以確保它們未超出限制，或可以容忍 MQRC_SYNCPOINT_LIMIT_REACHED 回覆碼

過期的訊息數

下一個適當的 MQGET 呼叫會捨棄已過期的訊息。 不過，如果未發生這類呼叫，則不會捨棄過期訊息，而且對於某些佇列 (特別是由 MessageId、 CorrelId 或 GroupId 完成訊息擷取並檢索佇列以取得效能的那些佇列)，許多過期訊息可能會累計。 佇列管理程式可以定期掃描任何佇列是否有過期訊息，然後會刪除這些訊息。 您可以選擇進行此掃描的頻率 (如果有的話)。 有兩種方法可以執行此動作:

明確要求
您可以控制掃描哪些佇列及掃描時間。 發出 REFRESH QMGR TYPE (EXPIRY) 指令，指定您要掃描的一或多個佇列。

定期掃描
您可以使用 EXPRYINT 屬性在佇列管理程式物件中指定期限間隔。 佇列管理程式會維護每一個佇列上過期訊息的相關資訊，並知道在何時值得掃描過期訊息。 每次達到 EXPRYINT 間隔時，佇列管理程式會尋找值得掃描過期訊息的候選佇列，並只掃描它認為值得的那些佇列。 它不會掃描所有佇列。 這可避免在不必要的掃描上浪費任何處理器時間。

只有佇列共用群組中的一個佇列管理程式才會掃描共用佇列。 一般而言，第一個要重新啟動的佇列管理程式或第一個要設定 EXPRYINT 的佇列管理程式會執行掃描。

註: 您必須為佇列共用群組內的所有佇列管理程式設定相同的 EXPRYINT 值。

z/OS IBM MQ for z/OS 提供的範例定義

請使用本主題作為 IBM MQ for z/OS 隨附的範例 JCL 及程式碼的參照。

下列範例定義隨附於 thlqual.SCSQPROC 程式庫中的 IBM MQ 。 您可以使用它們來定義系統物件，以及自訂您自己的物件。 您可以將其中部分包含在起始設定輸入資料集中 (如 起始設定指令 中所述)。

表 *22:* 系統物件的 *IBM MQ* 範例定義

起始設定輸入資料集 (initialization input data set)	**範例名稱**
CSQINP1	CSQ4INP1 CSQ4INPR

表 *22:* 系統物件的 *IBM MQ* 範例定義 *(*繼續*)*	
起始設定輸入資料集 (initialization input data set)	**範例名稱**
CSQINP2	CSQ4INSA CSQ4INYS [1] CSQ4INSX CSQ4INSG CSQ4INSR CSQ4INSS CSQ4INSJ CSQ4INSM CSQ4INYG CSQ4INYR CSQ4INYC CSQ4INYD CSQ4INSC
CSQINPT	CSQ4INST CSQ4INYT
其他	CSQ4DISP CSQ4INPX CSQ4IVPQ CSQ4IVPG CSQ4MSTR CSQ4MSRR CSQ4QMIN

註:

1. 這些範例定義的順序很重要: 如果 INYS、INSX 和 INSG 的順序不正確，則會發生錯誤。

CSQINP1 範例

當您針對每一個訊息類別使用一個頁面集時，請使用範例 CSQINP1 資料集 thlqual.SCSQPROC(CSQ4INP1); 當針對主要訊息類別使用多個頁面集時，請使用 thlqual.SCSQPROC(CSQ4INPR)。 它包含緩衝池的定義、設為緩衝池關聯的頁面，以及 ALTER SECURITY 指令。 將範例包含在佇列管理程式啟動型作業程序的 CSQINP1 連結中。

CSQINP2 範例

CSQ4INSG 系統物件範例

範例 CSQINP2 資料集 thlqual.SCSQPROC(CSQ4INSG) 包含下列系統物件的定義，以供一般使用:

- 系統預設物件
- 系統指令物件
- 系統管理物件
- 供系統使用的其他物件

您必須在此範例中定義物件，但您只需要在第一次啟動子系統時執行一次。 將定義併入 CSQINP2 資料集中是最佳作法。 它們在佇列管理程式關閉及重新啟動之間進行維護。 您不得變更物件名稱，但可以在必要時變更其屬性。

當符合下列條件時，會將一則訊息放置到 SYSTEM.DURABLE.SUBSCRIBER.QUEUE 佇列 (即使發佈訂閱不在作用中):

- QMGR 屬性 PSMODE 設為 DISABLED
- 範例物件 CSQ4INST 陳述式 `DEFINE SUB('SYSTEM.DEFAULT.SUB')` 存在。

若要避免此情況，請刪除或註銷 `DEFINE SUB('SYSTEM.DEFAULT.SUB')` 陳述式。

只有在使用 JMS 2.0 遞送延遲時， SYSTEM.DDELAY.LOCAL.QUEUE 才需要定義 JMS 2.0 遞送延遲暫置佇列。 依預設，佇列定義會註銷，您可以在必要時解除註解。

CSQ4INSA 系統物件及鑑別範例

範例 CSQINP2 資料集 thlqual.SCSQPROC(CSQ4INSA) 包含通道鑑別系統佇列定義。 此佇列會保留通道鑑別記錄。 它也包含預設通道鑑別規則。

如果 CHLAUTH 在佇列管理程式上「已啟用」且您想要執行通道，或者您想要 SET 或 DISPLAY CHLAUTH 記錄，則必須在此範例中定義物件。 您只需要在第一次啟動子系統時定義它們一次。 將定義併入 CSQINP2 資料集中是最佳作法。 它們是透過佇列管理程式關閉並重新啟動進行維護，您不得變更佇列名稱。

CSQ4INSS 系統物件範例

如果您使用佇列共用群組，則可以定義其他系統物件。

範例資料集 thlqual.SCSQPROC(CSQ4INSS) 包含與 CF 結構搭配使用的範例指令，以及一組共用通道及內部群組佇列作業所需的系統物件定義。

您無法依現狀使用此範例; 您必須在使用之前自訂它。 然後，您可以將此成員併入佇列管理程式啟動程序的 CSQINP2 DD 連結中，也可以使用它作為 CSQUTIL 公用程式的 COMMAND 函數輸入，以發出必要的指令。

當您定義群組或共用物件時，您只需要將它們包含在佇列共用群組中一個佇列管理程式的 CSQINP2 DD 連結中。

CSQ4INSX 系統物件範例

如果您使用分散式佇列及叢集作業，則必須定義其他系統物件。

範例資料集 thlqual.SCSQPROC(CSQ4INSX) 包含所需的佇列定義。 您可以在佇列管理程式啟動程序的 CSQINP2 DD 連結中包含此成員，也可以使用它作為 CSQUTIL 公用程式中 COMMAND 函數的輸入，以發出必要的 DEFINE 指令。

物件定義有兩種類型:

- SYSTEM.CHANNEL.xx，任何分散式佇列都需要
- SYSTEM.CLUSTER.xx，需要用於叢集作業

CSQ4INSJ 系統 JMS 物件範例

定義 JMS 發佈/訂閱網域中使用的佇列。

CSQ4INSM 系統物件範例

如果您使用進階訊息安全，則必須定義其他系統物件。 範例資料集 thlqual.SCSQPROC(CSQ4INSM) 包含必要的佇列定義。

CSQ4INSR 物件範例

定義 WebSphere Application Server 及分配管理系統使用的佇列。

CSQ4INYD 物件範例

如果您使用分散式佇列，且需要設定您自己的佇列、處理程序及通道。

範例資料集 thlqual.SCSQPROC(CSQ4INYD) 包含範例定義，可用來自訂分散式佇列物件。 它包括:

- 傳送端的一組定義
- 接收端的一組定義
- 使用用戶端的一組定義

您無法依現狀使用此範例-您必須在使用之前自訂它。 然後，您可以將此成員併入佇列管理程式啟動程序的 CSQINP2 DD 連結中，也可以使用它作為 CSQUTIL 公用程式的 COMMAND 函數輸入，以發出必要的 DEFINE 指令。 (這是較佳的作法，因為這表示您不必在每次重新啟動佇列管理程式時重新定義這些物件。)

CSQ4INYC 物件範例

如果您使用叢集作業，當需要時，會自動建立相等於分散式佇列的通道定義及遠端佇列定義的定義。 不過，需要一些手動通道定義-叢集的叢集接收端通道，以及至少一個叢集儲存庫佇列管理程式的叢集傳送端定義。

範例資料集: thlqual.SCSQPROC(CSQ4INYC) 包含下列範例定義，可用來自訂叢集作業物件:

- 佇列管理程式的定義
- 接收端通道的定義
- 傳送端通道的定義
- 叢集佇列的定義
- 叢集清單的定義

您無法依現狀使用此範例-您必須在使用之前自訂它。 然後，您可以將此成員併入佇列管理程式啟動程序的 CSQINP2 DD 連結中，也可以使用它作為 CSQUTIL 公用程式的 COMMAND 函數輸入，以發出必要的 DEFINE 指令。 這是較好的作法，因為這表示您不必在每次重新啟動 IBM MQ 時重新定義這些物件。

CSQ4INYG 物件範例

範例資料集: thlqual.SCSQPROC(CSQ4INYG) 包含下列範例定義，可用來自訂您自己的物件以供一般使用:

- 無法傳送郵件的佇列
- 預設傳輸佇列
- CICS 配接器物件

您無法依現狀使用此範例-您必須在使用之前自訂它。 然後，您可以將此成員併入佇列管理程式啟動程序的 CSQINP2 DD 連結中，也可以使用它作為 CSQUTIL 公用程式的 COMMAND 函數輸入，以發出必要的 DEFINE 指令。 這是較好的作法，因為這表示您不必在每次重新啟動 IBM MQ 時重新定義這些物件。

除了這裡的範例定義之外，您還可以使用系統物件定義作為自己資源定義的基礎。 例如，您可以製作 SYSTEM.DEFAULT.LOCAL.QUEUE 並將它命名為 MY.DEFAULT.LOCAL.QUEUE。 然後，您可以視需要變更此副本中的任何參數。 然後，您可以透過您選擇的任何方法來發出 DEFINE 指令，前提是您有權建立該類型的資源。

預設傳輸佇列

在決定是否要定義預設傳輸佇列之前，請先閱讀 預設傳輸佇列 說明。

- 如果您決定要定義預設傳輸佇列，請記住您也必須定義通道來處理它。
- 如果您決定不要定義 DEFXMITQ 陳述式，請記得在範例中從 ALTER QMGR 指令移除 DEFXMITQ 陳述式。

CICS 配接器物件

範例定義名為 CICS01.INITQ。 此佇列由 IBM MQ 提供的 CKTI 交易使用。 您可以變更此佇列的名稱; 不過，它必須符合 CICS 系統起始設定表格 (SIT) 中指定的名稱，或 INITPARM 陳述式中的 SYSIN 置換。

CSQ4INYS/CSQ4INYR 物件範例

使用的儲存類別定義:

- 每一個訊息類別各一個頁面集
- 訊息主要類別的多個頁集

例如， SYSTEM.COMMAND.INPUT 使用 STGCLASS ('SYSVOLAT') 及 SYSTEM.CLUSTER.TRANSMIT.QUEUE 使用 STGCLASS ('REMOTE')。 在 CSQ4INYS 中，這兩個儲存類別都使用相同的頁集。 在 CSQ4INYR 中，那些儲存類別會使用不同的頁集，以減少傳輸佇列填入的影響。

CSQINPT 範例

CSQ4INST

範例資料集: thlqual.SCSQPROC(CSQ4INST) 包含系統預設訂閱的定義。

您必須在此範例中定義物件，但只需要在第一次啟動發佈/訂閱引擎時執行一次。 在 CSQINPT 資料集中包含定義是最佳作法。 在佇列管理程式關閉並重新啟動期間，會維護它。 您不得變更物件名稱，但可以在必要時變更其屬性。

CSQ4INYT

範例資料集: thlqual.SCSQPROC(CSQ4INYT) 包含一組指令，您可能想要在啟動發佈/訂閱引擎時執行。 此範例顯示主題及訂閱資訊。

其他

CSQ4DISP 顯示範例

範例資料集: thlqual.SCSQPROC(CSQ4DISP) 包含一組通用 DISPLAY 指令，可顯示佇列管理程式上所有已定義的資源。 這包括所有 IBM MQ 物件及定義 (例如儲存體類別及追蹤) 的定義。 這些指令可以產生大量輸出。 您可以在 CSQINP2 資料集中使用此範例，或作為 CSQUTIL 公用程式之 COMMAND 函數的輸入。

CSQ4INPX 範例

範例資料集: thlqual.SCSQPROC(CSQ4INPX) 包含您每次啟動通道起始程式時可能想要執行的一組指令。 您必須先自訂此範例，然後才能使用; 然後您可以將它併入通道起始程式的 CSQINPX 資料集。

CSQ4IVPQ 和 CSQ4IVPG 範例

範例資料集: thlqual.SCSQPROC(CSQ4IVPQ) 及 thlqual.SCSQPROC(CSQ4IVPG) 包含執行安裝驗證程式 (IVP) 所需的 DEFINE 指令集。

您可以在 CSQINP2 資料集中包括這些範例。 當您順利執行 IVP 時，不需要在每次重新啟動佇列管理程式時再次執行它們。 因此，您不需要將這些範例永久保留在 CSQINP2 連結中。

CSQ4MSTR 及 CSQ4MSRR 範例

這些是佇列管理程式的範例已啟動作業程序: thlqual.SCSQPROC(CSQ4MSTR) 及 thlqual.SCSQPROC(CSQ4MSRR)。

CSQ4MSRR 會在 CSQINP2 連結中使用 CSQ4INYR ， 以便重要佇列分散在不同的頁集。

您可以移除註解，以便在必要時對新建立的佇列管理程式使用 CSQMINI 卡。

CSQ4QMIN 範例

範例 QMINI 資料集 thlqual.SCSQPROC(CSQ4QMIN)。

如需 QMINI 資料集及 **TransportSecurity** 段落的詳細資料，請參閱 QMINI 資料集 。

z/OS 在 z/OS 上回復並重新啟動

請使用本主題中的鏈結，以瞭解 IBM MQ for z/OS 的重新啟動及回復功能。

IBM MQ for z/OS 具有強大的重新啟動及回復功能。 如需佇列管理程式在停止之後如何回復，以及重新啟動時會發生什麼情況的相關資訊，請參閱下列子主題:

- 第 210 頁的『如何在 IBM MQ for z/OS 中對資料進行變更』
- 第 212 頁的『如何在 IBM MQ for z/OS 中維護一致性』
- 第 214 頁的『在 IBM MQ for z/OS 中終止期間發生的情況』
- 第 215 頁的『在 IBM MQ for z/OS 中重新啟動及回復期間發生的情況』
- 第 216 頁的『如何解決不確定的回復單元』
- 第 218 頁的『共用佇列回復』

相關概念

z/OS IBM MQ for z/OS 回復動作

相關工作

規劃備份及回復

z/OS 管理 z/OS

相關參考

z/OS IBM MQ for z/OS 的訊息

z/OS 如何在 IBM MQ for z/OS 中對資料進行變更

IBM MQ for z/OS 必須與其他子系統互動，以保持所有資料一致。 本主題包含 回復單元、它們是什麼以及如何在 回復中使用它們的相關資訊。

回復單元

回復單元 是由應用程式的單一佇列管理程式所執行的處理，它會將 IBM MQ 資料從一個一致點變更為另一個一致點。 一致性點 (也稱為 同步點 或 確定點) 是應用程式所存取的所有可回復資料一致的時間點。

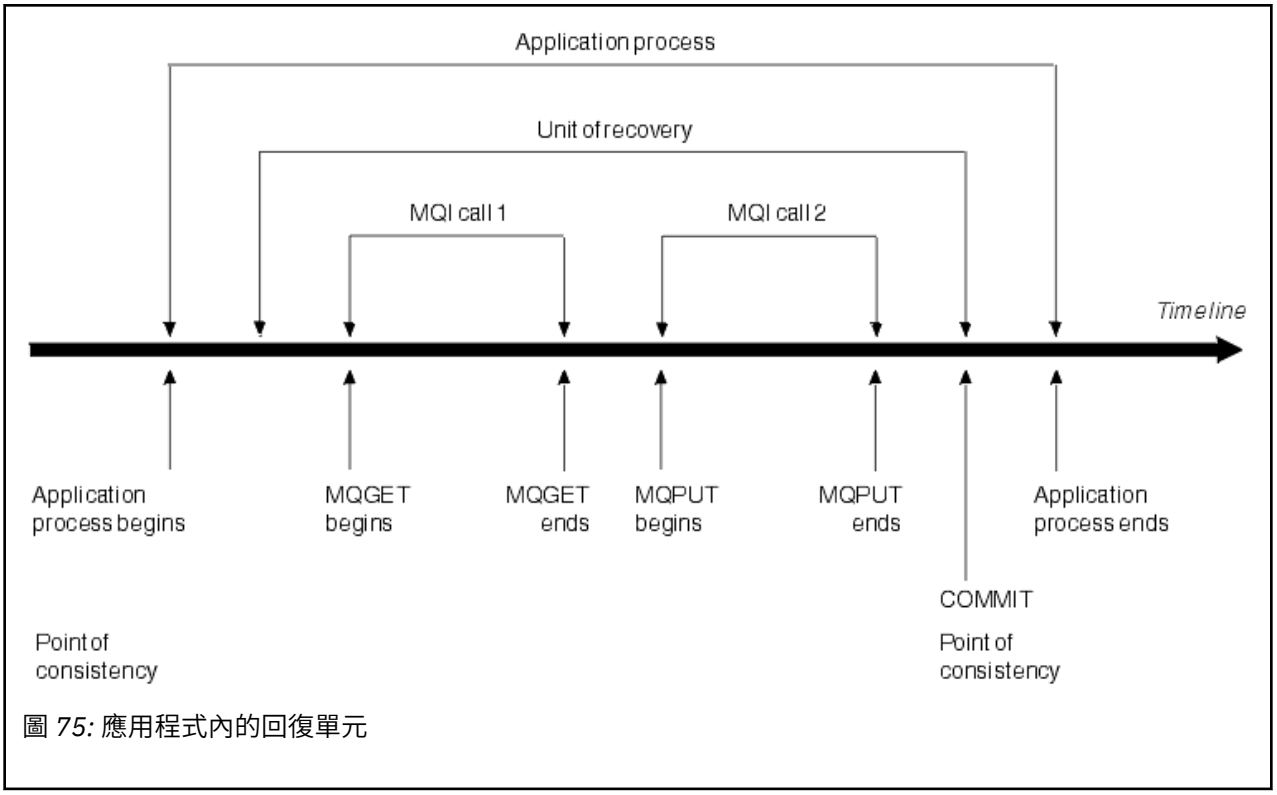


圖 *75:* 應用程式內的回復單元

回復單元在程式開始之後或在前一個一致性點之後，以資料的第一個變更開始; 它以後一個一致性點結束。 第 211 頁的圖 75 顯示回復單元、一致性點及應用程式之間的關係。 在此範例中，應用程式透過 MQI 呼叫 1 及 2 對佇列進行變更。 應用程式可以包括多個回復單元或僅一個回復單元。 不過，任何完整的回復單元都會在確定點結束。

例如，銀行交易將資金從一個帳戶轉移到另一個帳戶。 首先，程式會從第一個科目 (科目 A) 中扣除金額。 然後，它會將金額新增至第二個帳戶 B。 從 A 減去金額之後，兩個帳戶不一致， IBM MQ 無法確定。 當金額新增至科目 B 時，它們會變成一致。 當這兩個步驟都完成時，程式可以透過確定來發表一致性點，讓其他應用程式可以看見變更。

應用程式的正常終止會自動導致一致性點。 CICS 和 IMS 程式中的部分程式要求也會導致一致性點，例如 EXEC CICS SYNCPOINT。

取消工作

如果回復單元內發生錯誤， IBM MQ 會移除對資料所做的任何變更，並在回復單元開始時將資料傳回其狀態; 亦即， IBM MQ 會取消工作。 事件顯示在 第 212 頁的圖 76 中。

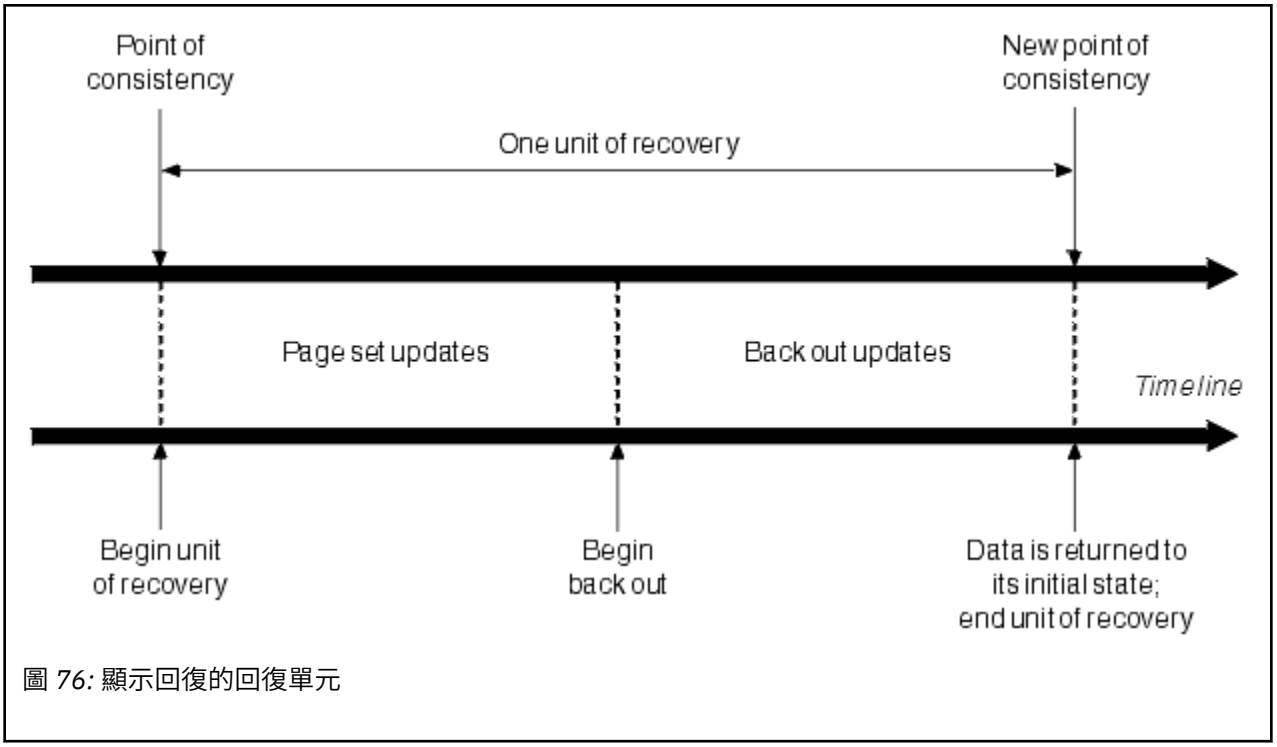


圖 *76:* 顯示回復的回復單元

z/OS 如何在 IBM MQ for z/OS 中維護一致性

IBM MQ for z/OS 中的資料必須與批次、 CICS、 IMS 或 TSO 一致。 在其中一個中變更的任何資料必須與在另一個中變更的資料相符。

在一個系統確定已變更的資料之前，它必須知道另一個系統可以進行對應的變更。 因此，系統必須進行通訊。

在 兩階段確定 期間 (例如在 CICS 下)，一個子系統會協調處理程序。 該子系統稱為 協調程式; 另一個是 參與者。 CICS 或 IMS 一律是與 IBM MQ 互動的協調程式，而 IBM MQ 一律是參與者。 在批次或 TSO 環境中， IBM MQ 可以參與 z/OS RRS 協調的兩階段確定通訊協定。

在 單一階段確定 期間 (例如在 TSO 或批次下)， IBM MQ 一律是互動中的協調者，並完全控制確定處理程序。

在 WebSphere Application Server 環境中， JMS 階段作業物件的語意會決定是否使用單階段或兩階段確定協調。

與 CICS 或 IMS 的一致性

IBM MQ 與 CICS 或 IMS 之間的連線支援下列同步點通訊協定:

- 兩階段確定-針對更新多個資源管理程式所擁有資源的交易。

  這是標準分散式同步點通訊協定。 它涉及比單階段確定更多的記載和訊息流程。
- 單一階段確定-用於更新單一資源管理程式 (IBM MQ) 所擁有之資源的交易。

  此通訊協定已針對記載及訊息流程進行最佳化。
- 略過同步點-適用於涉及 IBM MQ 但在佇列管理程式中不需要同步點 (例如，瀏覽佇列) 的交易。

在每一個情況下， CICS 或 IMS 都會充當同步點管理程式。

IBM MQ 用來與 CICS 或 IMS 通訊的兩階段確定階段如下:

1. 在階段 1 中，每一個系統會獨立判斷是否在其日誌中記錄了足夠的回復資訊，並且可以確定其工作。

   在階段結束時，系統會進行通訊。 如果他們同意的話，每一個都會開始下一個階段。

2. 在第 2 階段中，這些變更是永久的。 如果其中一個系統在階段 2 期間異常終止，則回復程序會在重新啟動期間完成作業。

兩階段確定程序的圖解

第 213 頁的圖 77 說明兩階段確定程序。 CICS 或 IMS 協調程式中的事件會顯示在上方行，而 IBM MQ 中的事件則會顯示在下方行。

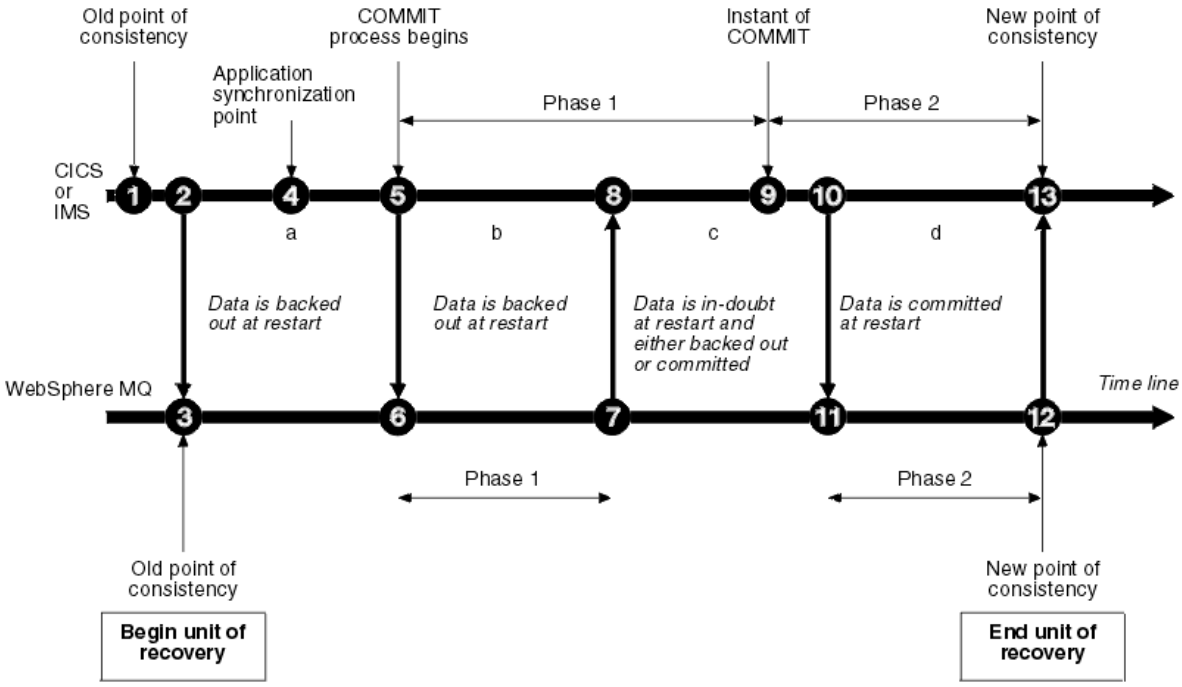


圖 77: 兩階段確定處理程序

下列區段中的數字會鏈結至圖中所示的數字。

1. 協調程式中的資料處於一致性點。
2. 協調程式中的應用程式會呼叫 IBM MQ ， 以新增訊息來更新佇列。
3. 這會在 IBM MQ 中啟動回復單元。
4. 在協調程式中繼續執行處理程序，直到達到應用程式同步點為止。
5. 然後，協調程式會啟動確定處理程序。 CICS 程式使用 SYNCPOINT 指令或一般應用程式終止來啟動確定。 IMS 程式可以使用 CHKP 呼叫、SYNC 呼叫、對 IOPCB 的 GET UNIQUE 呼叫或一般應用程式終止來啟動確定。 確定處理的階段 1 開始。
6. 當協調程式開始階段 1 處理程序時， IBM MQ 也會這樣做。
7. IBM MQ 順利完成階段 1 ， 在其日誌中寫入此事實，並通知協調程式。
8. 協調程式會接收通知。
9. 協調程式已順利完成其階段 1 處理。 現在兩個子系統都同意確定資料變更，因為這兩個子系統都已完成階段 1 ， 且可以從任何錯誤回復。 協調程式會在其日誌中記錄確定的瞬間-兩個子系統進行變更的不可撤銷決策。

   協調程式現在開始處理第 2 階段，即實際承諾。
10. 協調程式會通知 IBM MQ 開始其階段 2。
11. IBM MQ 記載階段 2 的開始。
12. 階段 2 已順利完成，現在這是 IBM MQ 的新一致性點。 然後， IBM MQ 會通知協調程式已完成其階段 2 處理。
13. 協調程式完成其階段 2 處理。 這兩個子系統所控制的資料現在一致且可供其他應用程式使用。

異常終止之後如何維護一致性

當佇列管理程式在異常終止之後重新啟動時，它必須決定是否要確定或取消在終止時處於作用中的回復單元。 對於部分回復單元， IBM MQ 具有足夠資訊來做出決策。 對於其他使用者，它不會，且在重新建立連線時必須從協調程式取得資訊。

第 213 頁的圖 77 顯示兩個階段內的四個期間 :a、b、c 及 d。 回復單元的狀態視發生終止的期間而定。 狀態可為下列其中一項:

進行中
佇列管理程式在完成階段 1 (期間 a 或 b) 之前結束; 在重新啟動期間， IBM MQ 會取消更新。

不能確定
佇列管理程式在完成階段 1 之後及開始階段 2 之前結束 (期間 c); 只有協調程式知道在確定之前或之後發生錯誤 (點 9)。 如果之前發生過，則 IBM MQ 必須取消其變更; 如果之後發生，則 IBM MQ 必須進行其變更並確定它們。 重新啟動時， IBM MQ 會在處理此回復單元之前等待來自協調程式的資訊。

確定中
佇列管理程式在開始其自己的階段 2 處理 (期間 d) 之後結束; 它會進行已確定的變更。

在取消中
在開始取消回復單元之後，但在重新啟動期間完成處理程序 (未顯示在圖中) 之前，佇列管理程式已結束， IBM MQ 會繼續取消變更。

z/OS 在 IBM MQ for z/OS 中終止期間發生的情況

佇列管理程式正常終止，以回應 STOP QMGR 指令。 如果佇列管理程式因任何其他原因而停止，則終止會異常。

請注意，在佇列管理程式終止期間， IBM MQ 會在內部發出指令

```
DISPLAY CONN(*) TYPE(CONN) ALL WHERE (APPLTYPE NE SYSTEMAL)
```

讓您知道哪些執行緒可能會阻止佇列管理程式完成關閉。

SYSTEMAL 符合 SYSTEM 或 CHINIT 的 APPLTYPES ，因此 DISPLAY CONN 指令會過濾不符合 SYSTEMAL 的應用程式類型，並將可能阻止正常關機的執行緒相關資訊傳回至工作日誌。

正常終止

在正常終止中， IBM MQ 會依序停止所有活動。 您可以使用靜止、強制或重新啟動模式來停止 IBM MQ 。 效果在 第 214 頁的表 23 中提供。

表 *23:* 使用 *QUIESCE*、*FORCE* 及 *RESTART* 終止

執行緒類型	**QUIESCE**	**強制**	**重新啟動**
作用中執行緒數	執行至完成	取消	取消
新建執行緒	可以啟動	不允許	不允許
新建連線	不允許	不允許	不允許

如果在應用程式仍連接時發生終止，則會通知批次應用程式。

使用 CICS，現行執行緒只會執行到回復單元結束為止。 使用 CICS，以靜止模式停止佇列管理程式會停止 CICS 配接卡，因此如果作用中作業包含多個回復單元，則作業不一定會執行到完成為止。

如果您在強制或重新啟動模式下停止佇列管理程式，則不會配置新的執行緒，且會回復連接的執行緒上的工作。 使用這些模式可以為確定處理階段之間的執行緒建立不確定的回復單元。 當 IBM MQ 與控制 CICS、 IMS 或 RRS 子系統重新連接時，會解決這些問題。

當您在任何模式下停止佇列管理程式時，步驟如下:

1. 連線已結束。
2. IBM MQ 不再接受指令。
3. IBM MQ 可確保頁面集的任何未完成更新都已完成。
4. IBM MQ 會在內部發出 DISPLAY USAGE 指令，以便在 z/OS 主控台日誌上記錄重新啟動 RBA。
5. 會取得關閉檢查點，並更新 BSDS。

指定靜止模式的終止不會影響不確定的回復單元。 任何處於不確定狀態的單位仍會處於不確定狀態。

異常終止

異常終止可能會使資料處於不一致狀態，例如:

- 在達到一致性點之前，已岔斷回復單元。
- 已確定的資料尚未寫入頁集。
- 未確定的資料已寫入頁集。
- 應用程式已在確定處理程序的階段 1 與階段 2 之間岔斷，使回復單元不確定。

IBM MQ 解決在重新啟動及回復期間因異常終止而產生的任何資料不一致。

z/OS 在 IBM MQ for z/OS 中重新啟動及回復期間發生的情況

IBM MQ 會使用其回復日誌及引導資料集 (BSDS) 來決定重新啟動時要回復的內容。 BSDS 會識別作用中及保存日誌資料集，以及日誌中最新 IBM MQ 檢查點的位置。

重新啟動及回復簡介

起始設定 IBM MQ 之後，會執行佇列管理程式重新啟動程序，如下所示:

- 日誌起始設定
- 現行狀態重建
- 轉遞日誌回復
- 反向日誌回復
- 佇列索引重建

完成回復時:

- 已確定的變更會反映在資料中。
- 不確定的活動會反映在資料中。 不過，在 IBM MQ 辨識不確定的決策並採取行動之前，資料會被鎖定且無法使用。
- 已從佇列中移除已岔斷的進行中及中斷中變更。 訊息一致且可以使用。
- 已使用新的檢查點。
- 已針對包含持續訊息的已檢索佇列建置新的索引 (如 第 216 頁的『重建佇列索引』 中所述)。

如果使用雙重 BSDS， IBM MQ 會檢查 BSDS 中時間戳記的一致性:

- 如果 BSDS 的兩個副本都是最新的， IBM MQ 會測試兩個時間戳記是否相等。 如果沒有， IBM MQ 會發出訊息 CSQJ120E 並終止。 當在個別 DASD 磁區上維護 BSDS 的兩個副本，且在佇列管理程式停止時還原其中一個磁區時，就會發生這種情況。 IBM MQ 會在重新啟動時偵測到狀況。
- 如果取消配置 BSDS 的一個副本，並且繼續使用單一 BSDS 進行記載，則可能會發生問題。 如果 BSDS 的兩個副本都在單一磁區上維護，且磁區已還原，或者如果兩個 BSDS 副本都個別還原，則 IBM MQ 可能不會偵測到還原。 在這種情況下，系統將不知道 BSDS 中未註明的日誌記錄。

在應用程式要求連線 之後 重新啟動時，不會通知批次應用程式。

瞭解回復所需的日誌範圍

在重新啟動期間，必須讀取的日誌資料範圍取決於許多因素:

- 在異常終止時，系統中通常有許多不完整的工作單元。 如先前所述，重新啟動處理程序將使系統進入一致性狀態，這可能包括取消進行中的工作單元，或在不確定的工作單元上回復鎖定。 工作單元回復需要進行中、取消中及不確定工作單元的所有工作單元日誌記錄都可用。 IBM MQ 會「分流」舊的工作單元，因此可以使用較小範圍的日誌資料來執行工作單元回復。
- 在異常終止時，通常會有許多持續更新項目只保留在緩衝池快取中。 它們尚未寫入磁碟。 這些變更必須從日誌讀取，並重新套用至頁集中保留的資料。 檢查點中的頁集回復 RBA 說明將頁集更新為一致狀態所需的最低日誌 RBA。
- 如果已在系統中引進舊頁集 (例如，已引進頁集備份以從媒體失敗中回復)，則必須從建立備份的日誌中讀取所有變更。 這些變更會重新套用至要回復的頁集中所保留的資料。 頁集第 0 頁中保留的頁集回復 RBA 說明頁集媒體回復所需的最低日誌 RBA。
- 如果在共用佇列上使用持續訊息，則需要日誌資料範圍才能回復保留持續訊息的 CFSTRUCT。 執行 CFSTRUCT 回復所需的最早日誌資料，大約來自舊 CFSTRUCT BACKUP 的時間。

在正常執行期間，可以使用 DISPLAY USAGE TYPE (DATASET) 指令來檢視與這些因素相關聯的回復日誌範圍 (當然因為重新引進舊頁集而無法提供資訊)。 若要避免在異常終止時可能延長佇列管理程式重新啟動的任何問題，請定期監視 DISPLAY USAGE TYPE (DATASET) 的值輸出。

此外，佇列管理程式會發出與下列因素相關的參考訊息:

- CSQJ160I 及 CSQJ161I 會警告長時間執行工作單元。
- CSQR026I 及 CSQR027I 提供這些長時間執行工作單元是否已順利延遲的相關資訊。
- CSQE040I 及 CSQE041E 警告結構備份已過時，因此 RECOVER CFSTRUCT 作業將會花費很長時間。

判斷哪個應用程式具有長時間執行的工作單元

可以判定具有長時間執行工作單元的應用程式。 如果要這麼做，請使用 DISPLAY CONN 指令。

DISPLAY CONN 指令會傳回連接至佇列管理程式之所有應用程式的連線資訊，以及可協助您判斷哪些應用程式目前具有長時間執行工作單元的其他資訊。 DISPLAY CONN 指令傳回的資訊與 DISPLAY QSTATUS 指令傳回的資訊類似，但主要差異在於 DISPLAY CONN 會顯示物件的相關資訊，以及特定連線的交易式資訊，而不是與特定物件相關聯之連線的詳細資料。

對於每一個連接的應用程式， DISPLAY CONN 指令會傳回下列資訊:

- 基本資訊，包括連線 ID 和 PID。
- 該連線的交易式資訊，包括建立交易的時間和日期 (亦即，在同步點下建立第一個 MQGET/PUT 的時間)，以及交易第一次寫入日誌的時間。
- 記載時間資訊，指出哪個應用程式仍有長時間執行的工作單元。
- 連線目前已開啟的所有物件清單。 每一個物件的詳細資料會以個別訊息傳回，並以連線 ID 作為索引鍵。因為有不同類型的物件 (例如佇列及佇列管理程式)，所以隨物件顯示的資訊是其物件類型特有的。

重建佇列索引

若要在未循序擷取訊息的佇列上增加 MQGET 作業的速度，您可以指定 IBM MQ 維護該佇列上所有訊息的訊息或相關性 ID 或群組 ID 的索引。

重新啟動佇列管理程式時，會針對每一個佇列重建這些索引。 這僅適用於持續訊息; 非持續訊息會在重新啟動時刪除。 如果已編製索引的佇列包含大量持續訊息，這會增加重新啟動佇列管理程式所花費的時間。

您可以選擇使用 CSQ6SYSP 巨集的 QINDXBLD 參數，以非同步重建索引來啟動佇列管理程式。 如果您設定 QINDXBLD=NOWAIT， IBM MQ 會重新啟動，而不會等待索引重建。

z/OS 如何解決不確定的回復單元

如果 IBM MQ 失去與另一個資源管理程式的連線，它通常會在重新啟動時嘗試回復所有不一致的物件。

如果 IBM MQ 失去與 CICS、 IMS 或 RRS 的連線，它通常會在重新啟動時嘗試回復所有不一致的物件。 解決不確定回復單元所需的資訊必須來自協調系統。 下列各節說明不同環境的解決程序。

- 如何從 CICS 解決不確定的回復單元
- 如何從 IMS 解決不確定的回復單元
- 如何從 RRS 解決不確定的回復單元
- 如何解決具有 GROUP 回復單元處置的不確定回復單元

如何從 CICS 解決不確定的回復單元

在某些情況下， CICS 無法執行 IBM MQ 處理程序來解決不確定的回復單元。 發生此情況時， IBM MQ 會傳送下列其中一則訊息:

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E

後面接著訊息 CSQC408I。

如需這些訊息意義的詳細資料，請參閱 IBM MQ for z/OS 訊息、完成及原因碼 手冊。

解決不確定單元不會影響 CICS 資源。 CICS 會控制回復協調，當它重新啟動時，會根據是否有日誌記錄標示確定開始，自動確定或取消每一個裝置。 當 IBM MQ 正在重新連接時，不確定物件的存在不會鎖定 CICS 資源。

CICS 配接器的其中一項功能是保持 CICS 與 IBM MQ 之間的資料同步。 如果佇列管理程式在連接至 CICS 時異常終止，則 CICS 可以在不 IBM MQ 察覺的情況下確定或取消工作。 當佇列管理程式重新啟動時，該工作稱為 不確定。

在重新啟動或重新連接 CICS 的連線之前， IBM MQ 無法解決這些不確定的回復單元 (亦即，確定或取消對 IBM MQ 資源所做的變更)。

在 CICS 配接器啟動期間，會起始解決不確定回復單元的處理程序。 當配接器要求不確定的回復單元清單時，處理程序即會啟動。 然後:

- 配接器會從 IBM MQ 接收此連線 ID 的不確定回復單元清單，並將它們傳遞至 CICS 以進行解析。
- CICS 會比較此清單中的項目與其專屬日誌中的項目。 CICS 會從它自己的清單判定針對每一個不確定的回復單元所採取的動作。

對於所有已解析的裝置， IBM MQ 會視需要更新佇列並釋放對應的鎖定。 未解析的單元在重新啟動之後可以保留。 請使用 管理 IBM MQ for z/OS 中說明的方法來解決這些問題。

如何從 IMS 解決不確定的回復單元

在 IMS 中解析不確定的回復單元不會影響 DL/I 資源。 IMS 可控制回復協調，並在重新啟動時自動確定或取消不完整的 DL/I 工作。 確定或取消線上區域 (非捷徑) 的決策是否有 IMS 日誌記錄類型 X'3730 '及 X'3801' 存在。 存在不確定的回復單元並不表示在 IBM MQ 連接之前鎖定 DL/I 記錄。

在佇列管理程式重新啟動期間， IBM MQ 會建立不確定的回復單元清單。 IMS 會建置自己的殘留回復項目 (RRE) 清單。 RRE 會記載在 IMS 檢查點，直到解決所有項目為止。

在將 IMS 區域重新連線至 IBM MQ 期間， IMS 會向 IBM MQ 指出是否確定或取消 IBM MQ 中標示為不確定的工作單元。

解決不確定單元時:

1. 如果 IBM MQ 辨識出它已標示要確定的項目，且 IMS 已將它標示為要取消，則 IBM MQ 會發出訊息 CSQQ010E。 IBM MQ 會針對 IBM MQ 與 IMS 之間此類型的所有不一致發出此訊息。
2. 如果 IBM MQ 有任何剩餘的不確定裝置，則配接器會發出訊息 CSQQ008I。

對於所有已解決的裝置， IBM MQ 會視需要更新佇列並釋放對應的鎖定。

IBM MQ 會對不確定的工作維護尚未解決的鎖定。 如果保留重要鎖定，這會在系統中造成待辦事項。 連線會保持作用中狀態，因此您可以解析 IMS RRE。 透過 管理 IBM MQ for z/OS 中說明的方法來回復不確定的執行緒。

除非有軟體或作業問題 (例如 IMS 冷開機)，否則應該解決所有不確定的工作。 在兩種情況下， IMS 控制區域會進行不確定的解決:

1. 在啟動與 IBM MQ 的連線時，同步完成解析。
2. 當程式異常終止時，在此期間會非同步執行解析。

如何從 RRS 解決不確定的回復單元

RRS 配接器的其中一項功能是讓 IBM MQ 與其他 RRS 參與的資源管理程式之間的資料保持同步。 當 IBM MQ 已完成第一階段確定，且正在等待 RRS (確定協調程式) 的決策時，如果發生失敗，則回復單元會進入不確定狀態。

在 RRS 與 IBM MQ 之間重新建立通訊時， RRS 會根據是否有日誌記錄標示確定開始，自動確定或取消每一個回復單元。 在重新建立與 RRS 的連線之前， IBM MQ 無法解決這些不確定的回復單元 (亦即，確定或取消對 IBM MQ 資源所做的變更)。

在某些情況下， RRS 無法解決不確定的回復單元。 發生此情況時， IBM MQ 會將下列其中一則訊息傳送至 z/OS 主控台:

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

如需這些訊息意義的詳細資料，請參閱 IBM MQ for z/OS 訊息、完成及原因碼 手冊。

對於所有已解析的回復單元， IBM MQ 會視需要更新佇列並釋放對應的鎖定。 無法解析的回復單元可以在重新啟動之後保留。 請使用 管理 IBM MQ for z/OS 中說明的方法來解決這些問題。

如何解決具有 GROUP 回復單元處置的不確定回復單元

在啟用 GROUPUR 佇列管理程式屬性的佇列共用群組 (QSG) 中，交易協調程式可以解決具有 GROUP 回復單元處置的不確定交易。 每當交易協調程式重新連接時，它通常會要求任何未完成的不確定交易清單，然後使用其日誌中的資訊來解析它們。

當已連接 GROUP 單元回復處置的交易協調程式要求不確定的交易清單時，傳回的清單包含所有具有 GROUP 單元回復處置 (存在於整個佇列共用群組中) 的不確定交易。 此清單不取決於啟動不確定交易的佇列管理程式。 處理此類要求的佇列管理程式會使用 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 然後，佇列管理程式會從其最後一個檢查點讀取任何非作用中佇列管理程式的日誌，以識別它們如果處於作用中狀態將會報告的任何其他不確定交易。

當交易協調程式要求解決不確定的交易時，它所連接的佇列管理程式會識別交易是否源自本身，如果如此，則會以與具有 QMGR 回復處置單元的交易相同的方式來解決該交易。 如果交易源自 QSG 中的另一個作用中佇列管理程式，則會使用 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 如果交易是在 QSG 的非作用中佇列管理程式上產生，則會立即解決任何共用佇列工作，並將解決任何剩餘專用佇列工作的要求放置在 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 在接受新工作之前，非作用中佇列管理程式會在啟動時處理此要求。 在此實務範例中，原始佇列管理程式的日誌仍會反映回復單元不確定，直到它重新啟動並處理要求為止。

z/OS 共用佇列回復

請利用這個主題來瞭解佇列共用群組環境中各種元件的 IBM MQ 回復及回復。

- 第 219 頁的『交易式回復』
- 第 219 頁的『同層級回復』

交易式回復

當應用程式發出 MQBACK 呼叫或異常終止 (例如，因為 EXEC CICS ROLLBACK 或 IMS 異常終止) 佇列管理程式中儲存的執行緒層次資訊，可確保回復進行中的工作單元。 共用佇列上同步點內的 MQPUT 及 MQGET 作業會以與非共用佇列更新相同的方式回復。

同層級回復

如果佇列管理程式失敗，它會與目前所連接的連結機能結構異常中斷連線。 如果 z/OS 實例與連結機能之間的連線失敗 (例如，連結機能或分割區的實體鏈結失敗或關閉電源) ，則也會偵測到佇列管理程式與所涉及連結機能結構之間的連線異常終止。 相同佇列共用群組中仍連接至該結構的其他佇列管理程式會偵測異常斷線，並嘗試對該結構上失敗的佇列管理程式起始 同層級回復 。 只有其中一個佇列管理程式會順利起始同層級回復，但所有其他佇列管理程式會合作回復失敗的佇列管理程式所擁有的工作單元。

當沒有同層級連接至結構時，如果佇列管理程式失敗，則會在另一個佇列管理程式連接至該結構時，或失敗的佇列管理程式重新啟動時，執行回復。

同層級回復 (通常稱為「同層級回復 (PLR)」) 是依結構在結構上執行，且單一佇列管理程式可以同時參與多個結構的回復。 不過，在不同結構的回復中合作的同層級集可能會因失敗時哪些佇列管理程式連接至不同結構而有所不同。

當失敗的佇列管理程式重新啟動時，它會重新連接至失敗時所連接的結構，並回復同層級回復未回復的任何剩餘未解決工作單元。

同層級回復是一個多階段處理程序。 在第一個階段期間，會回復已在進行中階段之外進行的工作單元; 這可能涉及確定工作單元的訊息，以及鎖定不確定工作單元的訊息。 在第二個階段期間，會檢查在失敗佇列管理程式中具有作用中執行緒的佇列、回復與進行中工作單元相關的未確定的訊息，以及重設失敗佇列管理程式中共用佇列上作用中控點的相關資訊。 這表示 IBM MQ 會重設任何指示器，指出失敗的佇列管理程式已開啟共用佇列以供輸入專用，容許其他作用中佇列管理程式開啟佇列供輸入。

共用佇列定義

代表共用佇列屬性的佇列物件會保留在佇列共用群組所使用的共用 Db2 儲存庫中。 請確定有足夠的程序可用來備份及回復用來保留 IBM MQ 物件的 Db2 表格。 您也可以使用 IBM MQ CSQUTIL 公用程式來建立 MQSC 指令，以在佇列管理程式中重播，以重新定義 IBM MQ 物件，包括儲存在 Db2 中的共用佇列及群組定義。

記載

佇列共用群組可以支援持續訊息，因為共用佇列上的訊息可以記載在佇列管理程式日誌中。

連結機能及結構故障

可以針對連結機能 (CF) 結構報告兩種類型的失敗: 結構失敗及連線中斷。 用於資料共用 (XES) 的 Sysplex 服務會通知 IBM MQ CF 結構失敗或 CF 失敗與結構失敗事件。 如果 XES 造成失去連線功能事件，這不一定表示結構有問題，可能是沒有連線可用來與結構通訊。 可能並非所有佇列管理程式都會收到結構的連線功能中

斷事件; 這取決於 CF 連線的配置。 由於操作員指令 (例如 VARY PATH OFFLINE 或 CONFIG CHP OFFLINE) ，也可能收到失去連線功能事件。

IBM MQ 使用的 CF 結構可以配置為使用系統管理的雙工。 這表示如果單一失敗，系統管理的失效接手處理程序會隱藏結構失敗或失去連線功能，且不會通知佇列管理程式失敗。 如果雙工結構或連線的兩個實例都失敗，則佇列管理程式會接收適當的事件，並以與單工結構的失敗事件相同的方式來處理該事件。 實務範例中說明佇列管理程式如何處理事件的詳細資料。

在 CF 或結構失敗的不太可能事件中，任何儲存在受影響應用程式結構中的非持續訊息都會遺失。 您可以使用 RECOVER CFSTRUCT 指令來回復持續訊息。 如果可回復的應用程式結構失敗，則在回復結構之前，會阻止對此結構進行任何進一步的應用程式活動。

若要確保您可以在合理時段內回復 CF 結構，請使用 BACKUP CFSTRUCT 指令進行頻繁備份。 您可以選擇在佇列共用群組中的任何佇列管理程式上執行備份，或指定一個佇列管理程式來執行所有備份。 自動化取得備份的程序，以確保定期取得備份。

每一個備份都會寫入進行備份之佇列管理程式的作用中日誌資料集。 共用佇列 Db2 儲存庫會記錄所備份的 CF 結構名稱、執行備份的佇列管理程式名稱、該佇列管理程式日誌上此備份的 RBA 範圍，以及備份時間。

管理結構包含在應用程式結構失敗時共用佇列上的工作單元不完整的相關資訊，因此在 RECOVER CFSTRUCT 處理期間必須可以使用管理結構。 如果管理結構失敗，佇列共用群組中的所有佇列管理程式必須已重建其管理結構項目，您才能發出 RECOVER CFSTRUCT 指令。

佇列管理程式會自動且不終止地重建其管理結構項目。 如果失敗時佇列管理程式不在執行中，其管理結構項目可以由在相同或更高層次執行的佇列共用群組中的另一個佇列管理程式重建。

若要回復應用程式結構，請對您要執行回復的佇列管理程式發出 RECOVER CFSTRUCT 指令。 您可以回復單一 CF 結構，也可以同步回復數個 CF 結構。 您可以使用佇列共用群組中的任何佇列管理程式來回復，它不必是執行備份的佇列管理程式，或先前已連接至失敗結構的佇列管理程式。

RECOVER CFSTRUCT 指令會使用透過 Db2 儲存庫資訊 (因此必須在執行回復的佇列管理程式上提供 Db2) 所找到的備份，並將此回復至失敗點。

RECOVER CFSTRUCT 指令會執行此動作，方法是將在備份開始與失敗時間之間執行 MQPUT 或 MQGET 之佇列共用群組中每一個佇列管理程式的日誌記錄，套用至對映至 CF 結構的任何共用佇列。 所產生的日誌合併可能需要讀取大量日誌資料，因為自讀取備份以來，參與佇列管理程式所寫入的所有日誌資料。 強烈建議您進行頻繁 (例如，每小時) 備份，尤其是在備份內有大量訊息時。

結構失敗實務範例

實務練習

如果針對 CF 結構報告失敗，則所連接佇列管理程式所採取的動作取決於下列各項:

- z/OS 的 XES 元件向 IBM MQ 報告的失敗類型。
- 結構類型 (應用程式或管理)
- 佇列管理程式層次
- IBM MQ CFSTRUCT 物件 (2、3、4 或 5) 的 CFLEVEL。 這不是 CFCC 微碼的 CFLEVEL)
- CFLEVEL 中 IBM MQ CFSTRUCT 物件的 RECAUTO 屬性 (5)

下列實務說明報告管理結構失敗時所發生的情況:

- 如果接收到管理結構的結構失敗事件，則會自動重新配置並重建結構，而不會終止佇列管理程式。 當佇列管理程式嘗試連接時， XES 會配置新的結構實例。

  當佇列管理程式已連接至結構的新實例時，佇列管理程式會將其本身的項目寫入結構中。 此處理由佇列管理程式執行，且不是 XES 重建處理程序的一部分。

  如果佇列管理程式在失敗時未執行，或在完成其部分管理結構的回復之前終止，則佇列共用群組中的另一個佇列管理程式可以重建其管理結構項目。

  佇列管理程式的管理結構項目只能由在相同層次或更高層次執行的另一個佇列管理程式來重建。 如果佇列共用群組中的另一個佇列管理程式無法重建佇列管理程式的管理結構項目，請重新啟動佇列管理程式，以便它可以完成其部分結構的重建。

在重建所有佇列管理程式的管理結構項目之前，會暫停某些動作。 已暫停的動作包括下列:

- 開啟及關閉共用佇列。
- 確定或取消回復單元。
- 與佇列管理程式連接或中斷連線的序列化應用程式。
- 備份或回復應用程式結構。

任何已連接至佇列管理程式的序列化應用程式都可以繼續處理。 任何嘗試使用 MQCNO_SERIALZE_CONN_TAG_QSG 或 MQCNO_RESTRICT_CONN_TAG_QSG 參數進行連接的序列化應用程式都會收到 MQRC_CONN_TAG_NOT_USABLE 回覆碼。

當已重建佇列管理程式的管理結構項目時，會回復已暫停的動作。

下列實務說明報告應用程式結構失敗時所發生的情況:

- 如果收到應用程式結構的結構失敗事件，且 CFLEVEL 是 1 或 2 ，則佇列管理程式會終止。 重新啟動佇列管理程式。 嘗試重新連接至結構的第一個佇列管理程式會導致 XES 配置結構的新實例。
- 如果接收到應用程式結構的結構失敗事件，且 CFLEVEL 是 3、4 或 5 ，則會繼續執行連接至結構的佇列管理程式。 未使用失敗結構中佇列的應用程式可以繼續正常處理。

  不過，嘗試在失敗結構中的佇列上執行作業的應用程式會收到 MQRC_CF_STRUC_FAILED 錯誤，直到順利重建結構為止，此時應用程式可以重新開啟佇列。

  針對以 RECAUTO (YES) 定義的 CFLEVEL (5) 應用程式結構，會自動啟動結構重建。 否則，當發出 RECOVER CFSTRUCT 指令時，將重建結構。

連結機能連線功能失敗的復原力

什麼是連結機能連線功能失敗的備援?

對連結機能連線功能失敗的恢復是指佇列共用群組中的佇列管理程式能夠容忍失去與連結機能結構的連線功能而不終止。 此功能也會嘗試在另一個連結機能中使用更好的連線功能重建結構，以儘快重新取得共用佇列的存取權。

何謂局部失去連線功能?

IBM MQ 定義局部失去連線功能的狀況，是 Sysplex 中的一個以上系統失去與連結機能的連線功能，系統所存取的結構已配置在該連結機能中，但 Sysplex 中至少有一個系統維持與相同連結機能的連線功能。

什麼是完全失去連線功能?

IBM MQ 定義完全失去連線功能的狀況，即 Sysplex 中沒有任何系統可連接至連結機能及其內配置的結構。

您為何要啟用此功能?

連結機能連線功能失敗的復原力可增進 IBM MQ 的可用性，在佇列管理程式失去與一或多個連結機能結構的連線功能之後，可讓非共用佇列保持可用。 此外，失去連結機能結構連線功能的佇列管理程式會自動嘗試在另一個可用的連結機能中重建結構，以改善佇列共用群組內共用佇列的可用性。

啟用此功能時的考量

如果沒有可用的替代連結機能，則容忍失去連結機能結構連線而不終止的佇列管理程式可能在一段時間內無法重新連接至連結機能結構。 在已失去連線功能的結構上定義的共用佇列會保持無法使用，直到還原該結構的連線功能為止。 在此狀況下，連接至佇列共用群組成員以執行共用佇列工作的應用程式可能會發現他們需要存取的共用佇列無法使用。 為了避免這種狀況，建議將佇列管理程式配置成在失去連結機能結構的連線功能時終止。 此終止會強制應用程式連接至佇列共用群組的另一個成員，該佇列共用群組可連接至定義應用程式所需共用佇列的連結機能結構。

管理連結機能連線功能失敗的復原力

如何啟用此功能?

必須執行下列步驟，才能啟用連結機能連線功能的備援

1. 請確定 CFRM 連結資料集已格式化，以支援系統管理的重建。 這可讓佇列管理程式起始系統管理的重建，以在可用的連結機能中重建結構。 請使用 **DISPLAY XCF,COUPLE,TYPE=CFRM** 指令來判斷 CFRM 連結資料集的格式。 若要支援系統管理的重建，應該透過指定下列指令來格式化 CFRM 連結資料集:

   ```
   "ITEM NAME(SMREBLD) NUMBER(1)"
   ```

   如需格式化 CFRM 連結資料集的相關資訊，請參閱 z/OS MVS 設定 Sysplex 文件。
2. 請確定替代連結機能可供使用，且位於所有 IBM MQ 連結機能結構的 CFRM 喜好設定清單中。 這可讓佇列管理程式嘗試將結構重建為替代可用的連結機能，以儘快還原對結構的存取權。

   IBM MQ 結構必須在 CFRM 原則中使用 ENFORCE_順序 (NO) 定義，以便在 IBM MQ 需要重新配置結構時， XCF 能夠在配置中選擇最佳 CF。

   如需結構喜好設定清單的相關資訊，請參閱 z/OS MVS 設定 Sysplex 文件。
3. 變更所有需要容忍失去 CFLEVEL (5) 連線功能的應用程式連結機能結構。 這是可容忍失去連線功能的最低層次。
4. 判定 **QMGR CFCONLOS** 及 **CFSTRUCT CFCONLOS** 屬性所需的值，並相應地變更這些值。 **QMGR CFCONLOS** 屬性控制是否容忍失去與管理結構的連線功能，而 **CFSTRUCT CFCONLOS** 屬性控制每一個應用程式連結機能結構是否容忍失去連線功能。 如果保留這些屬性的預設值，在失去任何連結機能結構的連線功能之後，佇列管理程式會終止。
5. 決定每一個應用程式連結機能結構的 **CFSTRUCT RECAUTO** 屬性所需值，並相應地變更這些值。 此屬性控制在完全失去連線功能之後，是否應該使用記載的資料自動回復連結機能結構。 如果保留此屬性的預設值，則在完全失去連線功能之後，不會對應用程式結構執行自動回復。

實務範例 1-失去與管理結構的連線功能

佇列管理程式可以容忍失去與管理結構的連線功能，而不會終止。

當任何佇列管理程式 (已配置為容許失去與管理結構的連線功能) 失去與管理結構的連線功能時，佇列共用群組的所有成員都會與管理結構中斷連線。 然後，佇列共用群組中的所有作用中佇列管理程式會嘗試重新連接至管理結構，使其在連結機能中重新配置，並具有與 Sysplex 中所有系統的最佳連線功能，然後重建管理結構資料。

註: 這不一定是與具有作用中佇列管理程式的所有系統具有最佳連線功能的連結機能。

如果佇列管理程式無法重新連接至管理結構 (例如，因為管理結構的 CFRM 喜好設定清單中沒有可用的連結機能)，則在佇列管理程式可以順利重新連接至管理結構並重建其管理結構資料之前，部分共用佇列作業仍無法使用。 當系統上有適當的連結機能可用時，會自動重新連線。

由於缺少連結機能的連線功能，或沒有適當的連結機能可用來配置結構，因此在佇列管理程式啟動期間無法連接至管理結構。 然後，佇列共用群組中的所有作用中佇列管理程式會嘗試重新連接至管理結構，使其重新配置在另一個連結機能中 (如果有的話)，並重建管理結構資料。

實務範例 2-失去與應用程式結構的連線功能

在不終止佇列管理程式的情況下，可以容忍失去與 **CFLEVEL(5)** 或更高版本的應用程式結構的連線功能。 佇列管理程式已連接至位於 **CFLEVEL(4)** 或更低版本的應用程式結構，或位於 **CFLEVEL(5)** 且未配置為容許失去連線功能的結構，當失去與結構的連線功能時，會異常終止，原因碼為 00C510AB 。

當連線功能遺失至已配置為容許失去連線功能的應用程式結構時，失去與結構連線功能的所有佇列管理程式都會斷線。 佇列管理程式的後續行為視失去連線功能是局部還是全部而定。

局部失去與應用程式結構的連線功能

如果判定失去連線功能是局部的，則失去與結構的連線功能的佇列管理程式會嘗試起始系統管理的重建，以將結構移至具有改良連線功能的另一個連結機能。 如果此重建成功，則結構中的持續及非持續訊息都會複製到其他連結機能，並還原對結構上佇列的存取權。 未失去連線功能的佇列管理程式仍會連接至該結構，不過，在系統管理的重建處理程序期間，會延遲存取該結構的作業。

如果應用程式結構無法重建至具有改良連線功能的另一個連結機能，或在另一個連結機能中重建之後，部分佇列管理程式仍無法連線至該結構，則在該結構上定義的佇列在連線功能還原至該連結機能之前，仍無法在沒有該結構連線功能的佇列管理程式上使用。 當結構變成可用且還原對結構上所定義共用佇列的存取權時，佇列管理程式會自動重新連接至結構。

失去與應用程式結構的連線功能總計

如果 Sysplex 中的所有 MVS 系統都與應用程式結構配置所在的連結機能失去連線，則每當嘗試重新連接結構時， z/OS 會從連結機能取消配置該結構。 佇列管理程式可能因數個原因而嘗試重新連接至結構，例如應用程式嘗試開啟共用佇列，或系統發出通知，指出新的連結機能資源可能已可用。 因此，在完全失去與應用程式結構的連線功能之後，可能會遺失受影響結構中的所有非持續訊息。

在完全失去連線功能之後，如果已使用 **RECAUTO(YES)**定義可回復的應用程式結構，則會自動回復它們。 如果可使用替代連結機能來配置中的結構，或只要這類連結機能變成可用，則幾乎會立即開始回復。 如果尚未使用 **RECAUTO(YES)**定義結構，則可以發出 **RECOVER CFSTRUCT** 指令來啟動回復。 這會回復結構中的所有持續訊息，但會遺失所有非持續訊息。 由於此處理程序涉及讀取佇列管理程式日誌，因此可能需要一些時間才能完成，因此建議定期進行結構備份，以減少在還原對結構上共用佇列的存取權之前的時間。

只要應用程式嘗試開啟在結構上定義的共用佇列，或從系統收到新的連結機能資源變成可用的通知，佇列管理程式就會嘗試重新連接至不可回復的應用程式結構。 如果有適當的連結機能可用來在中配置結構，則會配置新的結構，並還原對該結構上所定義共用佇列的存取權。 因為持續訊息無法放入定義在不可回復結構中的佇列，所以共用佇列上的所有訊息都會遺失。

作業行為

如果「IBM WebSphere MQ 7.1」或更新版本的佇列管理程式已配置成容忍遺失與特定連結機能結構的連線功能，則佇列共用群組的成員會嘗試自動從失敗中回復，並重新連接至該結構。 此活動可能包括在另一個連結機能中重新配置具有較佳連線功能的結構 (如果有的話)。 不過，可能仍需要操作員介入，才能從失去連線功能中回復。

一般而言，必要的操作員動作是:

1. 解決導致失去連線功能的失敗原因。
2. 請確定可以配置 IBM MQ 結構的連結機能可在 Sysplex 中的所有系統上使用

在失去連線功能事件之後自動重新配置在另一個連結機能中的任何結構，都可以移至具有與佇列共用群組中所有佇列管理程式最佳連線功能的連結機能。 必要的話，可以透過起始系統管理的重建指令 **SETXCF START,REBUILD** 來完成此作業，如 z/OS MVS 系統指令參考手冊中所記載。

如果部分失去與應用程式結構的連線功能，則失去與結構的連線功能的佇列管理程式會嘗試起始系統管理的重建。 只有在連結機能與目前連接至結構的所有作用中佇列管理程式具有連線功能時，此處理程序才會在另一個連結機能中配置結構。 因此，當佇列共用群組中的大部分佇列管理程式已失去與應用程式結構的連線功能時，由於仍連接至原始結構的佇列管理程式，它們可能無法將結構重建成另一個連結機能。 在此狀況下，仍連接至原始結構的佇列管理程式可以關閉，以容許重建結構，或發出 **RESET CFSTRUCT ACTION(FAIL)** 指令使結構失敗。 可以透過發出 **RECOVER CFSTRUCT** 指令，在適用的結構上起始回復。

註: 當失敗並回復結構時，結構上的所有非持續訊息都會遺失。

z/OS IBM MQ for z/OS 中的安全概念

請利用這個主題來瞭解 IBM MQ 安全的重要性，以及在系統上沒有足夠安全設定的含意。

為何您必須保護 IBM MQ 資源

IBM MQ 會處理可能有價值的資訊傳送。 套用安全可確保 IBM MQ 擁有及管理的資源受到保護，不會遭到未獲授權的存取。 這類存取可能會導致遺失或揭露資訊。

您應該確保任何未獲授權的使用者或處理程序都不會存取或變更下列任何資源:

- IBM MQ 的連線
- IBM MQ 物件，例如佇列、處理程序及名稱清單

- IBM MQ 傳輸鏈結，亦即 IBM MQ 通道
- IBM MQ 系統控制指令 (system control commands)
- IBM MQ 訊息
- 與訊息相關聯的環境定義資訊

為了提供必要的安全， IBM MQ 會使用 z/OS 系統授權機能 (SAF) ，將授權要求遞送至「外部安全管理程式 (ESM)」，例如「安全伺服器」(先前稱為 RACF)。 IBM MQ 不會自行進行安全驗證。 在使用分散式佇列或用戶端時，您可能需要其他安全措施， IBM MQ 會提供通道鑑別記錄、通道結束程式、MCAUSER 通道屬性及 TLS。

容許存取物件的決策由 ESM 及 IBM MQ 遵循該決策。 如果 ESM 無法做出決策，則 IBM MQ 會阻止存取物件。

如果您未保護 IBM MQ 資源會發生什麼情況

如果您未對安全執行任何動作，最可能的效果是 所有 使用者都可以存取及變更 每個 資源。 這不僅包括本端使用者，也包括遠端系統上使用分散式佇列或用戶端的使用者，其中登入安全控制可能比一般 z/OS 的情況更嚴格。

若要啟用安全檢查，您必須執行下列動作:

- 安裝並啟動 ESM (例如， Security Server)。
- 如果您使用安全伺服器以外的 ESM ，請定義 MQADMIN 類別。
- 啟動 MQADMIN 類別。

您必須考量使用大小寫混合的資源名稱是否有利於您的企業。 如果您在 ESM 設定檔中使用大小寫混合的資源名稱，則必須定義並啟動 MXADMIN 類別。

z/OS 資料集加密

「資料集加密 (DSE)」提供加密 z/OS 資料集的功能，因此它們包含的資料只能由獲授與特定許可權的使用者 ID 進行檢視或修改。 這會提供檔案系統中靜態資料的加密，並防止不慎將機密性資訊揭露給具有合法商業需求及管理資料集本身許可權的使用者。

在 IBM MQ for z/OS 9.1.4 之前， IBM MQ for z/OS 不支援搭配使用 DSE 與作用中日誌、頁集及共用訊息資料集 (SMDS) ，以提供 IBM MQ 訊息的主要持續性機制。

相反地， Advanced Message Security 提供 IBM MQ 傳訊的端對端加密解決方案，其中包含整個 IBM MQ 網路、動態資料加密、靜態資料加密，甚至在執行時期 IBM MQ 程序內。

IBM MQ 子系統中使用的其他 VSAM 及循序資料集可以使用 DSE 進行加密。 例如:

- Bootstrap 資料集 (bootstrap data set, BSDS)
- 使用 CSQINPx DDNAME 啟動時讀取的保留系統配置 (MQSC) 指令的循序檔
- IBM MQ 保存日誌，通常用於長期保存 IBM MQ 日誌資料以進行審核。

您可以透過配置使用資料集金鑰標籤定義的資料類別，來使用 DSE 進行加密。 如需相關資訊，請參閱 規劃日誌保存儲存體。

從 IBM MQ for z/OS 9.1.4 開始，除了舊版中提供的支援之外， IBM MQ for z/OS 還支援搭配使用 DSE 與作用中日誌及頁集。

IBM MQ for z/OS 不支援將 DSE 用於共用訊息資料集 (SMDS)。

請參閱 IBM MQ for z/OS 上具有資料集加密之靜態資料的機密性 一節。 的文件以取得相關資訊。

相關概念
安全概念
通道鑑別記錄
在 z/OS 上使用 IBM MQ 物件的權限
加密安全通訊協定: TLS

相關工作

在 z/OS 上設定安全

比較鏈結層次安全和應用程式層次安全

相關參考

IBM MQ for z/OS 的訊息

z/OS IBM MQ for z/OS 中的安全控制項和選項

您可以指定是否開啟整個 IBM MQ 子系統的安全，以及是否要在佇列管理程式或佇列共用群組層次執行安全檢查。 您也可以控制針對 API 資源安全所檢查的使用者 ID 數目。

子系統安全

子系統安全是一個控制項，可指定是否對整個佇列管理程式執行任何安全檢查。 如果您不需要安全檢查 (例如，在測試系統上)，或如果您對可連接至 IBM MQ 之所有資源 (包括用戶端及通道) 的安全層次感到滿意，則可以關閉佇列管理程式或佇列共用群組的安全檢查，以便不會進行進一步的安全檢查。

這是唯一可以完全關閉安全並判斷是否執行任何其他安全檢查的檢查。 也就是說，如果您關閉佇列管理程式或佇列共用群組的檢查，則不會執行其他 IBM MQ 檢查; 如果您保持開啟狀態，則 IBM MQ 會檢查其他 IBM MQ 資源的安全需求。

您也可以開啟或關閉特定資源集 (例如指令) 的安全。

佇列管理程式或佇列共用群組層次檢查

您可以在佇列管理程式層次或佇列共用群組層次實作安全。 如果您在佇列共用群組層次實作安全，群組中的所有佇列管理程式都會共用相同的設定檔。 這表示要定義及維護的設定檔較少，使安全管理更容易。 它也可讓您輕鬆將新的佇列管理程式新增至佇列共用群組，因為它會繼承現有的安全設定檔。

如果您的安裝需要，例如在移轉期間，或如果佇列共用群組中有一個佇列管理程式需要群組中其他佇列管理程式的不同安全層次，也可以實作兩者的組合。

佇列共用群組層次安全

對整個佇列共用群組執行佇列共用群組層次安全檢查。 它可讓您簡化安全管理，因為它需要您定義較少的安全設定檔。 使用者 ID 使用特定資源的授權是在佇列共用群組層次處理，與使用者 ID 用來存取資源的佇列管理程式無關。

例如，假設伺服器應用程式在使用者 ID SERVER 下執行，且想要存取稱為 SERVER.REQUEST，且您想要在 Sysplex 中的每一個 z/OS 映像檔上執行 SERVER 實例。 而不是允許 SERVER 開啟 SERVER.REQUEST (佇列管理程式層次安全)，您只能在佇列共用群組層次允許存取。

您可以使用佇列共用群組層次安全設定檔來保護所有類型的資源，不論是本端或共用。

佇列管理程式層次安全 (queue manager level security)

您可以使用佇列管理程式層次安全設定檔來保護所有類型的資源，不論是本端或共用。

兩個層次的組合

您可以使用佇列管理程式及佇列共用群組層次安全的組合。

您可以針對屬於該群組成員的特定佇列管理程式，置換佇列共用群組層次安全設定。 這表示您可以在個別佇列管理程式上執行不同層次的安全檢查，與在群組中其他佇列管理程式上執行的安全檢查。

如需相關資訊，請參閱 控制佇列共用群組或佇列管理程式層次安全的設定檔。

控制檢查的使用者 ID 數目

RESLEVEL 是安全伺服器設定檔，用來控制針對 IBM MQ 資源安全所檢查的使用者 ID 數目。 通常，當使用者嘗試存取 IBM MQ 資源時，「安全伺服器」會檢查相關使用者 ID，以查看是否容許存取該資源。 透過定義 RESLEVEL 設定檔，您可以控制是否要檢查零、一或兩個使用者 ID (如果適用的話)。

這些控制是在連線的基礎上執行，並在連線的有效期限內持續執行。

每一個佇列管理程式只有一個 RESLEVEL 設定檔。 控制是由使用者 ID 對此設定檔具有的存取權來實作。

大小寫混合的 IBM MQ RACF 類別

您現在可以使用大小寫混合的 RACF 設定檔支援，這可讓您使用大小寫混合的資源名稱，並定義 IBM MQ RACF 設定檔來保護它們。

您可以選擇:

- 繼續只使用大寫 IBM MQ RACF Classes as in previous releases， or
- 使用新的大小寫混合格式 IBM MQ RACF 類別。

如果不使用大小寫混合的 RACF 設定檔，您仍然可以在 IBM MQ for z/OS 中使用大小寫混合的資源名稱; 不過，這些資源名稱只能由大寫 IBM MQ 類別中的通用 RACF 設定檔來保護。 使用大小寫混合的 IBM MQ RACF 設定檔支援時，您可以在大小寫混合的 IBM MQ 類別中定義 IBM MQ RACF 設定檔，以提供更精細的保護層次。

z/OS 您可以在 IBM MQ for z/OS 中保護的資源

當佇列管理程式啟動時，或在操作員指令指示時， IBM MQ for z/OS 會決定您要保護哪些資源。

您可以控制針對每一個個別佇列管理程式執行哪些安全檢查。 例如，您可以在正式作業佇列管理程式上實作一些安全檢查，但在測試佇列管理程式上則沒有。

連線安全

當應用程式嘗試連接佇列管理程式時，都會執行連線安全檢查。 其作法是發出 MQCONN 或 MQCONNX 要求，或當通道起始程式或 CICS 或 IMS 配接器發出連線要求時。

如果您使用佇列管理程式層次安全，則可以關閉特定佇列管理程式的連線安全檢查。 不過，如果您這麼做，任何使用者都可以連接至該佇列管理程式。

對於 CICS 配接器，只會使用 CICS 位址空間使用者 ID 來進行連線安全檢查，而不會使用個別 CICS 終端機使用者 ID。 對於 IMS 配接器，當 IMS 控制項或相依區域連接至 IBM MQ 時，會檢查 IMS 位址空間使用者 ID。 對於通道起始程式，會檢查通道起始程式位址空間所使用的使用者 ID。

您可以在佇列管理程式或佇列共用群組層次開啟或關閉連線安全檢查。

指令安全

當使用者從 可在 IBM MQ for z/OS 上發出 MQSC 及 PCF 指令的來源中說明的任何來源發出 MQSC 指令時，即會執行指令安全檢查。 您可以對指令指定的資源進行個別檢查，如 第 226 頁的『指令資源安全』中所述。

如果您關閉指令檢查，則不會檢查指令的發證者，以查看他們是否具有發出指令的權限。

如果從主控台輸入 MQSC 指令，則主控台必須具有 z/OS SYS 主控台權限屬性。 從 CSQINP1 或 CSQINP2 資料集發出的指令，或由佇列管理程式在內部發出的指令，會免於所有安全檢查，而 CSQINPX 的指令則會使用通道起始程式位址空間的使用者 ID。 您必須控制容許誰透過一般資料集保護來更新這些資料集。

您可以在佇列管理程式或佇列共用群組層次開啟或關閉指令安全檢查。

指令資源安全

部分 MQSC 指令 (例如，定義本端佇列) 涉及 IBM MQ 資源的操作。 當指令資源安全處於作用中狀態時，每次發出涉及資源的指令時， IBM MQ 會檢查是否容許使用者變更該資源的定義。

您可以使用指令資源安全來協助施行命名標準。 例如，薪資管理者可能只能刪除及定義名稱以 "PAYROLL" 開頭的佇列。 如果指令資源安全處於非作用中狀態，則不會對指令所操作的資源進行安全檢查。 請勿混淆指令資源安全與指令安全; 兩者是獨立的。

關閉指令資源安全檢查不會影響特別針對其他不涉及指令之處理類型所執行的資源檢查。

您可以在佇列管理程式或佇列共用群組層次開啟或關閉指令資源安全檢查。

通道安全考量

通道安全性

當您使用通道時，可用的安全特性取決於您要使用的通訊協定。 如果您使用 TCP，雖然您可以使用 TLS，但通訊協定並未提供任何安全特性。 如果您使用 APPC，則可以透過網路將傳送端 MCA 的使用者 ID 資訊傳送至目的地 MCA 以進行驗證。

對於這兩種通訊協定，您可以指定要檢查哪些使用者 ID 以確保安全，以及要檢查多少個使用者 ID。 同樣地，可供您使用的選項取決於您使用的通訊協定、您在定義通道時指定的內容，以及通道起始程式的 RESLEVEL 設定。

如需可用通道安全類型的相關資訊，請參閱 通道鑑別記錄 及 安全結束程式概觀

相關參考

第 227 頁的『IBM MQ for z/OS 中的 API 資源安全』
當應用程式使用 MQOPEN 或 MQPUT1 呼叫開啟物件時，會檢查資源。 開啟物件所需的存取權視開啟佇列時指定的開啟選項而定。

z/OS IBM MQ for z/OS 中的 API 資源安全

當應用程式使用 MQOPEN 或 MQPUT1 呼叫開啟物件時，會檢查資源。 開啟物件所需的存取權視開啟佇列時指定的開啟選項而定。

API-資源安全細分為下列檢查:

- 佇列
- 程序
- 名單
- 替代使用者
- 環境定義

開啟佇列管理程式物件或存取儲存類別物件時，不會執行安全檢查。

佇列

佇列安全檢查會控制容許開啟哪個佇列，以及容許開啟哪個佇列的選項。 例如，可能容許使用者開啟稱為 PAYROLL.INCREASE.SALARY 可瀏覽佇列上的訊息 (使用 MQOO_BROWSE 選項)，但不從佇列中移除訊息 (使用其中一個 MQOO_INPUT_ * 選項)。 如果您關閉佇列檢查，任何使用者都可以使用任何有效的開啟選項 (即 MQOPEN 或 MQPUT1 呼叫上任何有效的 MQOO_ * 選項) 來開啟任何佇列。

您可以在佇列管理程式或佇列共用群組層次開啟或關閉佇列安全檢查。

處理程序

當使用者開啟程序定義物件時，會執行程序安全檢查。 如果您關閉檢查處理程序，則任何使用者都可以開啟任何處理程序。

您可以在佇列管理程式或佇列共用群組層次開啟或關閉處理程序安全檢查。

名稱清單

當使用者開啟名稱清單時，會執行名稱清單安全檢查。 如果您關閉檢查名稱清單，任何使用者都可以開啟任何名稱清單。

您可以在佇列管理程式或佇列共用群組層次開啟或關閉名單安全檢查。

替代使用者

替代使用者安全控制一個使用者 ID 是否可以使用另一個使用者 ID 的權限來開啟 IBM MQ 物件。

例如:

- 以使用者 ID PAYSERV 執行的伺服器程式會從使用者 ID USER1 放置在佇列上的佇列中擷取要求訊息。
- 當伺服器程式取得要求訊息時，它會處理要求，並將回覆放回要求訊息所指定的回覆佇列中。
- 伺服器可以指定其他使用者 ID (在此情況下為 USER1)，而不是使用自己的使用者 ID (PAYSERV) 來授權開啟回覆目的地佇列。 在此範例中，替代使用者安全會控制在開啟回覆目的地佇列時，是否容許使用者 ID PAYSERV 指定使用者 ID USER1 作為替代使用者 ID。

替代使用者 ID 指定在物件描述子 (MQOD) 的 *AlternateUserId* 欄位中。

您可以在任何 IBM MQ 物件 (例如，處理程序或名稱清單) 上使用替代使用者 ID。 它不會影響任何其他資源管理程式所使用的使用者 ID，例如 CICS 安全或 z/OS 資料集安全。

如果替代使用者安全不在作用中，則任何使用者都可以使用任何其他使用者 ID 作為替代使用者 ID。

您可以在佇列管理程式或佇列共用群組層次開啟或關閉替代使用者安全檢查。

環境定義

環境定義是適用於特定訊息的資訊，並且包含在屬於訊息一部分的訊息描述子 (MQMD) 中。 環境定義資訊分為兩個區段:

身分區段
首先將訊息放入佇列的應用程式使用者。 它由下列欄位組成:

- *UserIdentifier*
- *AccountingToken*
- *ApplIdentityData*

原始區段
將訊息放置在目前儲存所在佇列上的應用程式。 它由下列欄位組成:

- *PutApplType*
- *PutApplName*
- *PutDate*
- *PutTime*
- *ApplOriginData*

當發出 MQPUT 或 MQPUT1 呼叫時，應用程式可以指定環境定義資料。 依預設，應用程式可能會產生資料、從另一個訊息傳遞資料，或佇列管理程式可能會產生資料。 例如，伺服器程式可以使用環境定義資料來檢查要求程式的身分，亦即，此訊息是否來自正確的應用程式? 通常會使用 *UserIdentifier* 欄位來決定替代使用者的使用者 ID。

您可以使用環境定義安全來控制使用者是否可以在任何 MQOPEN 或 MQPUT 呼叫上指定任何環境定義選項。 如需環境定義選項的相關資訊，請參閱 與訊息環境定義相關的 MQOPEN 選項。 如需與環境定義相關之訊息描述子欄位的說明，請參閱 MQMD-訊息 descriptorMQMD -訊息描述子。

如果您關閉環境定義安全檢查，任何使用者都可以使用佇列安全所容許的任何環境定義選項。

您可以在佇列、佇列管理程式或佇列共用群組層次開啟或關閉環境定義安全檢查。

z/OS z/OS 上的可用性

IBM MQ for z/OS 具有許多高可用性特性。 本主題說明可用性的部分考量。

如果佇列管理程式或通道起始程式失敗， IBM MQ 的數個特性可以增加系統可用性。 如需這些特性的相關資訊，請參閱下列各節:

- Sysplex 考量
- 共用佇列
- 共用通道
- IBM MQ 網路可用性
- 使用 z/OS Automatic Restart Manager (ARM)
- 使用 z/OS 延伸回復機能 (XRF)
- 在佇列共用群組中使用 z/OS GROUPUR 屬性進行回復
- 何處可找到可用性的相關資訊

Sysplex 考量

在 *Sysplex* 中，許多 z/OS 作業系統映像檔在單一系統映像檔中分工合作，並使用連結機能進行通訊。 IBM MQ 可以使用 Sysplex 環境的機能來加強可用性。

移除佇列管理程式與特定 z/OS 映像檔之間的親緣性，可在映像檔失敗時在不同的 z/OS 映像檔上重新啟動佇列管理程式。 如果您確定下列各項，重新啟動機制可以是手動、使用 ARM 或使用系統自動化:

- 所有頁面集、日誌、引導資料集、程式碼庫及佇列管理程式配置資料集都定義在共用磁區上。
- 子系統定義具有 Sysplex 範圍及 Sysplex 內的唯一名稱。
- 在 IPL 時安裝在每個 z/OS 映像檔上的 早期程式碼 層次是相同層次。
- Sysplex 中的每一個 TCP 堆疊都有 TCP 虛擬 IP 位址 (VIPA) ，且您已配置 IBM MQ TCP 接聽器和入埠連線來使用 VIPA ，而不是預設主機名稱。

如需在 Sysplex 中使用 TCP 的相關資訊，請參閱 *Sysplex* 中的 *TCP/IP* SG24-5235(IBM Redbooks 出版品)。

此外，您可以配置在 Sysplex 中不同作業系統映像檔上執行的多個佇列管理程式，以作為佇列共用群組來運作，這可以利用共用佇列及共用通道來達到更高可用性及工作量平衡。

共用佇列

在佇列共用群組環境中，應用程式可以連接至佇列共用群組內的任何佇列管理程式。 由於佇列共用群組中的所有佇列管理程式都可以存取同一組共用佇列，因此應用程式並不取決於特定佇列管理程式的可用性; 佇列共用群組中的任何佇列管理程式都可以為任何佇列提供服務。 如果佇列管理程式因為佇列共用群組中的所有其他佇列管理程式都可以繼續處理佇列而停止，則這會提供更大的可用性。 如需共用佇列高可用性的相關資訊，請參閱 第 154 頁的『使用共用佇列的優點』。

為了進一步加強佇列共用群組中的訊息可用性， IBM MQ 會偵測群組中的另一個佇列管理程式是否異常中斷與連結機能的連線，並在可能時完成該佇列管理程式仍在擱置中的工作單元。 這稱為 同層級回復 ，並在 第 219 頁的『同層級回復』中說明。

同層級回復無法回復失敗時不確定的工作單元。 您可以使用「自動重新啟動管理程式 (ARM)」來重新啟動失敗所涉及的所有系統 (例如 CICS、 Db2 及 IBM MQ) ，並確保它們都在相同的新處理器上重新啟動。 這表示它們可以重新同步化，並可快速回復不確定的工作單元。 這說明於第 230 頁的『使用 z/OS Automatic Restart Manager (ARM)』。

共用通道

在佇列共用群組環境中， IBM MQ 提供可提供網路高可用性的功能。 通道起始程式可讓您使用網路產品，在一組合格伺服器之間平衡網路要求，並隱藏網路上的伺服器故障 (例如， VTAM 一般資源)。 IBM MQ 使用一般埠來處理入埠要求，以便連接要求可以遞送至佇列共用群組中任何可用的通道起始程式。 這說明於第 172 頁的『共用通道』。

共用出埠通道會取得它們從共用傳輸佇列傳送的訊息。 對於整個佇列共用群組層次，共用通道狀態的相關資訊會保留在一個位置。 這表示如果通道起始程式、佇列管理程式或通訊子系統失敗，則可以在佇列共用群組中的不同通道起始程式上自動重新啟動通道。 這稱為 同層級通道回復 ，並在 共用出埠通道中說明。

IBM MQ 網路可用性

在 IBM MQ 網路中，會使用通道將 IBM MQ 訊息從佇列管理程式傳送至佇列管理程式。 您可以在許多層次變更配置，以改善佇列管理程式的網路可用性，以及 IBM MQ 通道偵測網路問題及重新連接的能力。

TCP *Keepalive* 適用於 TCP/IP 通道。 它會導致 TCP 在階段作業之間定期傳送封包，以偵測網路失敗。 KAINT 通道屬性決定通道的這些封包頻率。

AdoptMCA 容許因網路中斷而在接收處理中封鎖的通道終止並取代為新的連線要求。 您可以搭配使用 ADOPTMCA 佇列管理程式內容與 MQSC 公用程式或 AdoptNewMCAType 內容與「可程式指令格式」介面，來控制 AdoptMCA 。

ReceiveTimeout 可防止在網路接收呼叫中永久封鎖通道。 RCVTIME 和 RCVTMIN 通道起始程式參數會根據通道的活動訊號間隔來決定通道的接收逾時性質。 如需詳細資料，請參閱 佇列管理程式參數 。

使用 z/OS Automatic Restart Manager (ARM)

您可以將 IBM MQ for z/OS 與 z/OS 自動重新啟動管理程式 (ARM) 一起使用。 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 失敗，則整個相關子系統及應用程式群組也會失敗。 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。

ARM 可讓您在共用佇列環境中快速回復不確定的交易。 如果您沒有使用佇列共用群組，它也會提供更高的可用性。

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。

若要啟用自動重新啟動，您必須執行下列動作:

1. 設定 ARM 連結資料集。
2. 在 *ARM* 原則中定義您要 z/OS 執行的自動重新啟動動作。
3. 啟動 ARM 原則。

如果您要自動重新啟動不同 z/OS 映像檔中的佇列管理程式，則每一個可能重新啟動該佇列管理程式之 z/OS 映像檔中的每一個佇列管理程式，都必須定義一個全 Sysplex 的唯一 4 個字元子系統名稱。

在 IBM MQ 網路中使用 ARM 中說明 ARM 與 IBM MQ 搭配使用。

使用 z/OS 延伸回復機能 (XRF)

您可以在延伸回復機能 (XRF) 環境中使用 IBM MQ 。 所有 IBM MQ 擁有的資料集 (執行碼、BSDS、日誌及頁集) 必須位於作用中及替代 XRF 處理器之間共用的 DASD 上。

如果您使用 XRF 進行回復，則必須停止作用中處理器上的佇列管理程式，並在替代處理器上啟動它。 對於 CICS，您可以使用 CICS 提供的指令清單表 (CLT) 來執行此動作，或者系統操作員可以手動執行此動作。 對於 IMS，這是手動作業，您必須在協調 IMS 系統完成處理器交換器之後執行。

必須先完成或終止 IBM MQ 公用程式，然後才能將佇列管理程式切換至替代處理器。 在規劃 XRF 回復計劃時，請仔細考量此潛在岔斷的影響。

在作用中處理器上的佇列管理程式終止之前，請小心防止在替代處理器上啟動佇列管理程式。 過早啟動可能會導致資料、型錄及日誌中發生嚴重的完整性問題。 使用廣域資源序列化 (GRS) 可防止在兩個系統上同時使用 IBM MQ ，有助於避免完整性問題。 您必須併入 BSDS 作為受保護資源，且必須在 GRS 環中併入作用中及替代 XRF 處理器。

在佇列共用群組中使用 z/OS GROUPUR 屬性進行回復

佇列共用群組 (QSG) 容許本主題中說明的其他交易式機能。 GROUPUR 屬性容許 XA 用戶端應用程式在 QSG 的任何成員上執行任何可能需要的不確定交易回復。

如果 XA 用戶端應用程式透過 Sysplex 連接至佇列共用群組 (QSG) ，則無法保證它所連接的特定佇列管理程式。 佇列共用群組內的佇列管理程式使用 GROUPUR 屬性可以啟用任何不確定的交易回復，而這些回復可能需要在 QSG 的任何成員上發生。 即使應用程式最初連接的佇列管理程式無法使用，也可以進行交易回復。

此特性會從對 QSG 特定成員的任何相依關係中釋放 XA 用戶端應用程式，因此延伸佇列管理程式的可用性。在交易式應用程式中，佇列共用群組會呈現為單一實體，提供所有 IBM MQ 特性，且沒有單一佇列管理程式失敗點。

此功能在交易式應用程式中並不明顯。

何處可找到可用性的相關資訊

您可以從下列來源找到這些主題的相關資訊:

表 *24:* 何處可找到可用性的相關資訊

主題	**在何處尋找**
佇列共用群組	第 139 頁的『共用佇列及佇列共用群組』
系統參數	配置系統參數
使用自動重新啟動管理程式公用程式	在 IBM MQ 網路中使用 ARM
MQSC 指令	MQSC 指令

z/OS IBM MQ for z/OS 上的監視及統計資料

IBM MQ for z/OS 具有一組機能，用於監視佇列管理程式及收集統計資料。

IBM MQ 提供用來監視系統及收集統計資料的機能。 如需這些機能的進一步相關資訊，請參閱下列各節:

- 第 231 頁的『連線監視』
- 第 232 頁的『IBM MQ 追蹤』
- 第 232 頁的『事件』

連線監視

IBM MQ 包括下列指令，用於監視 IBM MQ 物件的狀態:

- DISPLAY CHSTATUS 顯示指定通道的狀態。
- DISPLAY QSTATUS 顯示指定佇列的狀態。
- DISPLAY CONN 顯示指定連線的狀態。

如需這些指令的相關資訊，請參閱 MQSC 指令。

IBM MQ 追蹤

IBM MQ 提供追蹤機能，您可以在佇列管理程式執行時用來收集下列資訊:

效能統計資料
統計資料追蹤會收集下列資訊，以協助您監視效能並調整系統:

- 不同 MQI 要求的計數 (訊息管理程式統計資料)
- 不同物件要求的計數 (資料管理程式統計資料)
- Db2 用法的相關資訊 (Db2 管理程式統計資料)
- 「連結機能」用法的相關資訊 (「連結機能」管理程式統計資料)
- SMDS 使用情形 (共用訊息資料集統計資料) 的相關資訊
- 緩衝池使用情形 (緩衝區管理程式統計資料) 的相關資訊
- 記載的相關資訊 (日誌管理程式統計資料)
- 儲存體用量 (儲存體管理程式統計資料) 的相關資訊
- 鎖定要求的相關資訊 (鎖定管理程式統計資料)

帳戶資料

- 結算追蹤會收集處理 MQI 呼叫所花費處理器時間的相關資訊，以及特定使用者所提出的 MQPUT 及 MQGET 要求數目的相關資訊。
- IBM MQ 也可以使用 IBM MQ 來收集每一個作業的相關資訊。 此資料會收集為執行緒層次帳戶記錄。 對於每一個執行緒， IBM MQ 也會收集該執行緒所使用之每一個佇列的相關資訊。

追蹤所產生的資料會傳送至「系統管理機能 (SMF)」或一般性追蹤機能 (GTF)。

事件

IBM MQ 事件提供佇列管理程式中錯誤、警告及其他重要事件的相關資訊。 透過將這些事件合併至您自己的系統管理應用程式，您可以針對多個 IBM MQ 應用程式，監視多個佇列管理程式之間的活動。 特別是您可以從單一佇列管理程式監視系統中的所有佇列管理程式。

事件可以透過使用者撰寫的報告機制，向支援將事件呈現給操作員的管理應用程式報告。 事件也可讓應用程式作為其他管理網路 (例如 NetView) 的代理程式，以監視報告並建立適當的警示。

相關工作
使用 IBM MQ 追蹤
使用 IBM MQ 事件

z/OS z/OS 上的回復單元處置

當連接至佇列共用群組 (QSG) 中的佇列管理程式時，某些交易式應用程式可以使用 GROUP 而非 QMGR 回復處置單元，方法是在連接時指定 QSG 名稱，而不是佇列管理程式名稱。 這可移除重新連接至 QSG 中相同佇列管理程式的需求，讓交易回復更靈活且更健全。

由使用佇列共用群組名稱連接的應用程式所啟動的交易，也具有 GROUP 單元回復處置。

當交易式應用程式以 GROUP 單元回復處置方式連接時，它會以邏輯方式連接至佇列共用群組，且與任何特定佇列管理程式沒有親緣性。 在連接至 QSG 內的任何佇列管理程式時，可以查詢並解決它已啟動且已完成確定處理程序 phase-1 的任何兩階段確定交易 (亦即，它們是不確定的)。 在回復實務範例中，這表示交易協調程式不必重新連接至當時可能無法使用的相同佇列管理程式。

使用 QMGR 回復單元處置來連接的應用程式，與它們所連接的佇列管理程式具有直接親緣性。 在回復實務範例中，不論佇列管理程式是否屬於佇列共用群組，交易協調程式都必須重新連接至相同的佇列管理程式，以解決任何不確定的交易。

當應用程式指定佇列共用群組名稱，並因此以 GROUP 單元回復處置來連接至 QSG 中的佇列管理程式時，佇列共用群組在邏輯上是個別的資源管理程式。 這表示只有在應用程式以相同的回復單元處置重新連接時，才能看見不確定的交易。 具有 QMGR 回復處置單元的不確定交易對於已連接 GROUP 回復處置單元的應用程式不可見，反之亦然。

相關概念

第 233 頁的『啟用 GROUP 回復單元』
佇列共用群組可以配置及啟用 GROUP 單元回復的支援。

第 233 頁的『應用程式支援』
請利用這個頁面來判斷哪些應用程式可以連接 GROUP 單元的回復處置。

z/OS 啟用 GROUP 回復單元

佇列共用群組可以配置及啟用 GROUP 單元回復的支援。

若要在 QSG 內的佇列管理程式上使用 GROUP 回復單元，請啟用 GROUPUR 佇列管理程式屬性。 如需此概念的相關資訊，請先參閱 第 232 頁的『z/OS 上的回復單元處置』，然後再閱讀本主題的其餘部分。

啟用 GROUPUR 佇列管理程式屬性時，佇列管理程式會接受具有 GROUP 單元回復處置的新連線。 如果您停用此屬性，則不會接受具有此處置的新連線，雖然已連接的應用程式在中斷連線之前不會受到影響。

當應用程式以 GROUP 回復處置單元連接並查詢不確定的交易，或嘗試解析已在佇列共用群組 (QSG) 中其他位置啟動的交易時，它現在所連接的佇列管理程式必須能夠與佇列共用群組的其他成員通訊，以便它可以處理要求。 為此，它會使用稱為 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 此佇列必須位於稱為 CSQSYSAPPL 的可回復應用程式結構上。 結構必須是可回復的，因為在處理解析要求時，持續訊息會儲存在此佇列中。

您必須先確定已定義連結機能結構及共用佇列，才能啟用 GROUP 回復單元。 您可以使用 CSQ4INSS 範例中的定義。 在啟動期間定義或偵測到佇列時，佇列共用群組中的每一個佇列管理程式都會開啟佇列，以便它可以接收送入的要求。 如果您因為未正確定義佇列而想要刪除或移動佇列，您可以更新佇列物件來禁止 MQGET 要求，以要求佇列管理程式關閉其開啟控點。 當您進行必要的更正時，允許應用程式再次從佇列取得訊息會指示每一個佇列管理程式重新開啟它。 請使用 DISPLAY QSTATUS 指令來識別在佇列上開啟的控點。

完成此設定之後，您可以在每一個佇列管理程式上啟用 GROUP 回復單元，讓交易式應用程式能夠以 GROUP 回復單元處置來連接。 這不需要是佇列共用群組內的所有佇列管理程式，但如果您選擇只在佇列共用群組的子集上啟用此功能，則必須確定您的應用程式只嘗試連接至已啟用此功能的佇列管理程式。 如需相關資訊，請參閱 第 233 頁的『應用程式支援』。

當您嘗試啟用 GROUPUR 佇列管理程式屬性時，會執行一些配置檢查。 佇列管理程式會檢查下列各項:

- 它屬於佇列共用群組。
- 根據 CSQ4INSS 中的定義，已定義稱為 SYSTEM.QSG.UR.RESOLUTION.QUEUE 的共用佇列。
- SYSTEM.QSG.UR.RESOLUTION.QUEUE 位於稱為 CSQSYSAPPL 的可回復 CF 結構上。

如果上述任何檢查失敗， GROUPUR 屬性仍會停用，並傳回訊息碼。

如果已啟用佇列管理程式屬性，則也會在佇列管理程式啟動時執行這些配置檢查。 如果在啟動 GROUP 回復單元期間有任何檢查失敗，且佇列管理程式會發出訊息來識別失敗的檢查。 當您執行必要的更正動作之後，必須重新啟用佇列管理程式屬性。

z/OS 應用程式支援

請利用這個頁面來判斷哪些應用程式可以連接 GROUP 單元的回復處置。

對 GROUP 回復單元處置的支援僅限於特定類型的交易式應用程式，其中 IBM MQ for z/OS 是其資源管理程式，而不是交易協調程式。 目前支援的交易式應用程式如下:

- IBM MQ 延伸交易式用戶端應用程式
- 在應用程式伺服器中執行的 IBM MQ classes for JMS 應用程式，例如 WebSphere Application Server。
- 當使用 RESYNCMEMBER (GROUPRESYNC) 配置 CICS MQCONN 資源定義時，在 CICS Transaction Server 4.2 或更新版本中執行的 CICS 應用程式。

相關概念

第 234 頁的『IBM MQ 延伸交易式用戶端應用程式』
請利用這個頁面來決定 IBM MQ 延伸交易式用戶端應用程式如何使用 GROUP 單元回復處置。

第 234 頁的『CICS 應用程式』
請利用這個頁面來決定 CICS 如何使用 GROUP 單元回復處置。

z/OS *IBM MQ* 延伸交易式用戶端應用程式

請利用這個頁面來決定 IBM MQ 延伸交易式用戶端應用程式如何使用 GROUP 單元回復處置。

IBM MQ 延伸交易式用戶端應用程式的範例是使用 JMS 並在 WebSphere Application Server 中執行，透過 TCP/IP 而非本端連結連接至 IBM MQ 。 這些用戶端應用程式透過網路連線 (例如透過 TCP/IP) 連接至 IBM MQ for z/OS 。 對於這些應用程式，它是 xa_open 呼叫中所傳遞之 xa_info 字串的 QMNAME 參數指定值，指定是否使用 QMGR 或 GROUP 回復單元處置。 如需 xa_open 的相關資訊，請參閱 xa_open 字串的格式 和 xa_open 的其他錯誤處理。 對於 JMS 應用程式，這是透過在 ConnectionFactory 中指定佇列共用群組 (QSG) 的名稱而非特定佇列管理程式的名稱來完成。

若要讓 XA 用戶端應用程式利用 GROUP 回復單元處置，您必須配置 TCP/IP 設定，以容許用戶端應用程式遞送至佇列共用群組中已啟用 GROUPUR 屬性的佇列管理程式，而不是特定佇列管理程式。 您可以用來執行此動作的其中一個動態虛擬 IP 位址技術是 z/OS SysPlex 經銷商。 如需詳細資料，請參閱 z/OS Communications Server 和 z/OS 基本技能: 動態虛擬定址 。 如果您要在佇列共用群組中的佇列管理程式子集上啟用 GROUP 回復單元，請確保用戶端應用程式無法遞送至未啟用它的那些應用程式。

您的用戶端應用程式不需要使用共用通道連接至佇列共用群組。

z/OS *CICS* 應用程式

請利用這個頁面來決定 CICS 如何使用 GROUP 單元回復處置。

CICS 4.2 以及更新版本在 MQCONN 資源定義中提供群組重新同步選項 RESYNCMEMBER (GROUPRESYNC)。 使用此選項配置的「 CICS 」可以連接至佇列共用群組中任何適當的佇列管理程式，該佇列共用群組與該 CICS 區域在相同的 LPAR 上執行。 若要支援 CICS GROUPRESYNC 選項，佇列管理程式必須在 MQ V7.1 或更新版本上執行，並啟用 GROUPUR 支援。

在使用 GROUPRESYNC 連接至 MQ 的 CICS 區域內執行的交易，會建立具有 GROUP 單元回復處置的工作單元。

在佇列管理程式失敗之後，您可以使用 RESYNCMEMBER (GROUPRESYNC) 來啟用更快速的回復，因為它可讓 CICS 區域立即連接至在相同 LPAR 上執行的替代合格佇列管理程式，並視需要解決任何不確定的交易，而無需等待佇列管理程式重新啟動。

RESYNCMEMBER (GROUPRESYNC) 也會為 CICS 啟用更靈活的重新啟動選項。 如果 CICS 區域的 MQ 連線配置為使用 GROUPRESYNC 和 MQ 共用佇列，則可以在佇列管理程式以相同佇列共用群組成員身分執行的任何 LPAR 上重新啟動。

z/OS IBM MQ 及其他 z/OS 產品

請利用這個主題來瞭解 IBM MQ 如何使用其他 z/OS 產品。

相關概念

第 235 頁的『IBM MQ 及 CICS』
IBM MQ 9.0.0 以及更新版本所支援的所有 CICS 版本，都使用 CICS 提供的配接器及橋接器版本。

第 240 頁的『IBM MQ for z/OS 及 WebSphere Application Server』
請利用這個主題來瞭解 WebSphere Application Server 如何使用 IBM MQ for z/OS 。

相關參考

第 236 頁的『IBM MQ 及 IMS』
請利用這個主題來瞭解 IBM MQ 如何使用 IMS。 IMS 配接器可讓您將佇列管理程式連接至 IMS，並可讓 IMS 應用程式使用 MQI。

第 239 頁的『IBM MQ 及 z/OS 批次、TSO 和 RRS 配接器』
請利用這個主題來瞭解 IBM MQ 如何使用「 z/OS 批次」、TSO 和 RRS 配接器。

z/OS IBM MQ 及 CICS

IBM MQ 9.0.0 以及更新版本所支援的所有 CICS 版本，都使用 CICS 提供的配接器及橋接器版本。

如需配置 IBM MQ CICS 配接器及 IBM MQ CICS bridge 元件的相關資訊，請參閱 CICS 說明文件的 配置與 IBM MQ 的連線 一節。

相關工作

將 IBM MQ 與 CICS 搭配使用

z/OS CICS 群組連接

CICS 群組連接可讓 CICS 區域連接至相同 LPAR 上 IBM MQ 佇列共用群組的任何作用中成員，而不指定個別佇列管理程式。 CICS 仍會一次連接至單一佇列管理程式。

LPAR 上至少需要兩個佇列管理程式，才能支援 CICS 群組連接。 使用群組連接可提供較高的可用性，因為您不需要特定佇列管理程式處於作用中狀態。 CICS 會連接至 LPAR 上佇列共用群組中的任何佇列管理程式。

如需相關資訊，請參閱 MQCONN 資源的 CICS 文件。

CICS 會嘗試連接至所傳遞的 MQNAME ，如同它是佇列管理程式一樣:

- 如果佇列管理程式存在且在作用中，則連線將運作。
- 如果連線失敗， CICS 會查詢群組中佇列管理程式的狀態，以確定哪些在相同 LPAR 上處於作用中。
- 如果有多個佇列管理程式在作用中， CICS 會檢查 RESYNCMEMBER (YES) 和 UOW 狀態，以判斷 CICS 是否需要或應該連接至特定成員，或在非作用中時等待。
- 如果不需要連接至特定成員， CICS 會選取佇列管理程式 (使用隨機化演算法)。
- CICS 會嘗試連接至選擇的佇列管理程式。
- 如果嘗試失敗，則視回覆碼而定， CICS 會選擇下一個成員，然後再次進入選擇迴圈。
- 如果沒有作用中的佇列管理程式， CICS 會對佇列管理程式清單發出多個連線，並在 ECBLIST 上等待，直到第一個佇列管理程式變成可用為止。

相關概念

第 235 頁的『CICS 的群組回復單元 (GROUPUR)』

IBM MQ GROUPUR for CICS 提供佇列共用群組 (QSG) 中不確定工作單元的同層級回復。 一個 IBM MQ 佇列管理程式可以代表佇列共用群組中的另一個佇列管理程式，解決不確定的工作單元。 這表示如果 CICS 透過群組連接重新連接至 QSG 中的不同佇列管理程式，它可以從前一個 IBM MQ 連線解決不確定的交易。

相關資訊

支援 IBM MQ 佇列共用群組

z/OS CICS 的群組回復單元 (GROUPUR)

IBM MQ GROUPUR for CICS 提供佇列共用群組 (QSG) 中不確定工作單元的同層級回復。 一個 IBM MQ 佇列管理程式可以代表佇列共用群組中的另一個佇列管理程式，解決不確定的工作單元。 這表示如果 CICS 透過群組連接重新連接至 QSG 中的不同佇列管理程式，它可以從前一個 IBM MQ 連線解決不確定的交易。

如果 CICS 區域正在使用佇列管理程式，且佇列管理程式異常結束，則會回復任何不確定的交易。 這樣就不需要 CICS 區域等待它正在使用的佇列管理程式重新啟動，然後解決任何不確定的工作單元。 這表示您在 LPAR 上至少需要兩個佇列管理程式，以便在第一個佇列管理程式異常終止時，「 CICS 」可以連接至另一個佇列管理程式。

CICS MQCONN 定義上的新 RESYNCMEMBER (GROUPRESYNC) 設定:

- 使用 IBM MQ 群組連接功能及同層級回復。
- 需要已啟用 GROUPUR 屬性的佇列管理程式。
- 仍然支援現有的 CICS MQCONN RESYNCMEMBER 設定 (YES 和 NO):
  - 使用現有的 CICS 群組連接功能，而不使用同層級回復。

– 下次 CICS 連接至 IBM MQ 時，變更 RESYNCMEMBER 設定會生效。

相關概念

第 233 頁的『啟用 GROUP 回復單元』

佇列共用群組可以配置及啟用 GROUP 單元回復的支援。

z/OS IBM MQ 及 IMS

請利用這個主題來瞭解 IBM MQ 如何使用 IMS。 IMS 配接器可讓您將佇列管理程式連接至 IMS，並可讓 IMS 應用程式使用 MQI。

選用額外的 IBM MQ - IMS 橋接器可讓應用程式執行不使用 MQI 的 IMS 應用程式。 這表示您可以搭配使用舊式應用程式與 IBM MQ，而不需要重新編寫它們。

如需這些元件的相關資訊，請參閱下列子主題:

相關概念

IBM MQ for z/OS 上的 IMS 及 IMS 橋接器應用程式

相關工作

設定 IMS 配接器

設定 IMS 橋接器

操作 IMS 配接器

相關參考

MQIIH- IMS 資訊標頭

z/OS IMS 配接器

IMS 配接器是 IMS 應用程式與 IBM MQ 子系統之間的介面。

IBM MQ 配接器可讓不同的應用程式環境透過訊息佇列作業網路來傳送及接收訊息。 IMS 配接器是 IMS 應用程式與 IBM MQ 子系統之間的介面。 它可讓 IMS 應用程式使用 MQI。

IMS 配接器使用 IMS 提供的 外部子系統連接機能 (ESAF) 來接收及解譯存取 IBM MQ 的要求。 通常， IMS 會自動連接至 IBM MQ ，而無需操作員介入。

IMS 配接器為以下列模式或狀態執行的程式提供對 IBM MQ 資源的存取權:

- 作業 (TCB) 模式
- 問題狀態
- 非跨記憶體模式
- 非存取登錄模式

配接器提供從應用程式作業控制區塊 (TCB) 到 IBM MQ 的連線執行緒。

對於對 IBM MQ 所擁有的資源所做的變更，配接器支援兩段式確定通訊協定，並以 IMS 作為同步點協調程式。 IMS 配接卡不支援 IMS 不是同步點協調程式的交談，例如 APPC 保護 (SYNCLVL = SYNCPT) 交談。

配接器也提供觸發監視器異動 (CSQQTRMN)。 這說明於第 237 頁的『IMS 觸發監視器』。

您可以搭配使用 IBM MQ 與 IMS 延伸回復機能 (XRF) ，以協助從 IMS 錯誤進行回復。

註: 從 IMS 15.2 不再支援「延伸回復機能 (XRF)」。 如需相關資訊，請參閱 IMS 文件。

使用配接器

應用程式和 IMS 配接器在相同的位址空間中執行。 佇列管理程式在其自己的位址空間中是分開的。

您必須鏈結編輯每一個向適當的 IMS 語言介面模組發出一個以上 MQI 呼叫的程式，以及 (除非它使用動態 MQI 呼叫) IBM MQ 提供的 API Stub 程式 CSQQSTUB。 當應用程式發出 MQI 呼叫時， Stub 會透過 IMS 外部子系統介面，將控制傳送至配接器，該介面會管理訊息佇列管理程式對要求的處理。

使用 IMS 進行系統管理及作業

獲授權的 IMS 終端機操作員可以發出 IMS 指令，以控制及監視與 IBM MQ 的連線。 不過， IMS 終端機操作員無法控制 IBM MQ 位址空間。 例如，操作員無法從 IMS 位址空間關閉 IBM MQ 。

限制

在使用 IMS 配接器的應用程式內不支援下列 IBM MQ API 呼叫:

- MQCB
- MQCB_XX_ENCODE_CASE_ONE function
- MQCTL

IMS 觸發監視器

IMS 觸發監視器 (**CSQQTRMN**) 是 IBM MQ 提供的 IMS 應用程式，可在發生 IBM MQ 事件時 (例如，將訊息放入特定佇列時) 啟動 IMS 交易。

如何運作

當訊息放入應用程式訊息佇列時，如果符合觸發條件，則會產生觸發程式。 然後，佇列管理程式會將訊息 (包含部分使用者定義資料) (稱為 觸發訊息) 寫入已對該訊息佇列指定的起始佇列。 在 IMS 環境中，您可以啟動 CSQQTRMN 的實例，以監視起始佇列，並在觸發訊息到達時從中擷取觸發訊息。 一般而言， CSQQTRMN 透過 INSERT (ISRT) 排定另一個 IMS 交易至 IMS 訊息佇列。 已啟動的 IMS 應用程式會從應用程式訊息佇列讀取訊息，然後處理它。 CSQQTRMN 必須以非訊息 BMP 執行。

CSQQTRMN 服務的每一個副本都是單一起始佇列。 當它啟動時，觸發監視器會一直執行，直到 IBM MQ 或 IMS 結束為止。

CSQQTRMN 的 APPLCTN 巨集必須指定 SCHDTYP=PARALLEL。

因為觸發監視器是批次導向的 BMP ，所以觸發監視器所啟動的 IMS 交易包含下列項目:

- IOPCB 的 LTERM 欄位中的空白
- IOPCB 的使用者 ID 欄位中觸發監視器 BMP 的 PSB 名稱

如果目標 IMS 交易受到「安全伺服器」(先前稱為 RACF) 保護，您可能需要將 CSQQTRMN 定義為「安全伺服器」的使用者 ID。

z/OS IBM MQ - IMS 橋接器

IBM MQ - IMS 橋接器是 IBM MQ for z/OS 的元件，容許從 IBM MQ 應用程式直接存取 IMS 系統上的應用程式。

IBM MQ - IMS 橋接器會啟用 隱含的 *MQI* 支援。 這表示您可以將由 3270 連接的終端機所控制的舊式應用程式重新設計成由 IBM MQ 訊息所控制，而無需重新撰寫、重新編譯或重新鏈結它們。 橋接器是 IMS *Open Transaction Manager Access* (OTMA) 用戶端。

在橋接器應用程式中， IMS 應用程式內沒有 IBM MQ 呼叫。 應用程式使用 GET UNIQUE (GU) 取得其輸入至 IOPCB ，並使用 ISRT 將其輸出傳送至 IOPCB。 IBM MQ 應用程式會使用訊息資料中的 IMS 標頭 (MQIIH 結構) ，以確保應用程式可以在由不可程式化終端機驅動時執行。 如果您使用處理多區段訊息的 IMS 應用程式，請注意所有區段都應該包含在一個 IBM MQ 訊息內。

第 238 頁的圖 78 中說明 IMS 橋接器。

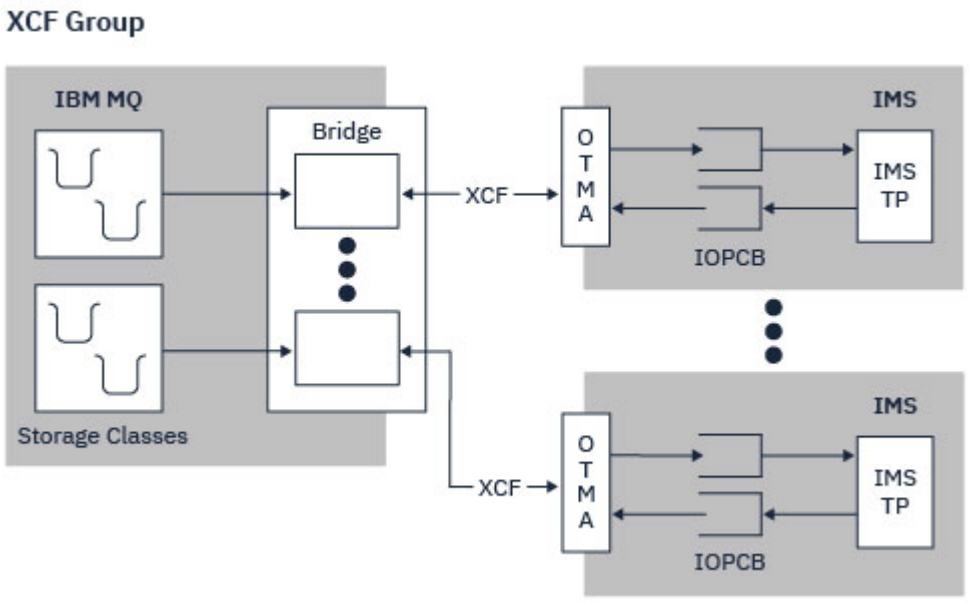


圖 *78: IBM MQ - IMS* 橋接器

佇列管理程式可以連接至一或多個 IMS 系統，且多個佇列管理程式可以連接至一個 IMS 系統。 唯一的限制是它們必須全部屬於相同的 XCF 群組，且必須全部位於相同的 Sysplex 中。

如需設定 IMS 橋接器及將其他 IMS 連線新增至相同佇列管理程式的相關資訊，請參閱 設定 IMS 橋接器 。

什麼是 OTMA?

IMS OTMA 機能是在 IMS 上執行的交易型無連線主從式通訊協定。 它充當主機型通訊伺服器的介面，透過 z/OS 跨系統連結機能 (XCF)存取 IMS TM 應用程式。

OTMA 可讓用戶端連接至 IMS ，以針對大型網路或大量階段作業，為用戶端與 IMS 之間的互動提供高效能。 OTMA 在 z/OS Sysplex 環境中實作。 因此， OTMA 的網域限制為 XCF 的網域。

OTMA 資源監視

IMS v10 或更新版本中提供的 x '3C' OTMA 通訊協定訊息支援，包含在 IBM MQ for z/OS 的 IBM MQ - IMS 橋接器中。 IMS 會將這些訊息傳送至 OTMA 用戶端，以報告其性能狀態。

如果 IMS 夥伴無法處理正在傳送的交易要求量，則會通知 IBM MQ 發生洪水警告。 在回應中， IBM MQ 會降低透過橋接器傳送要求的速率。

如果 IMS 仍無法處理交易要求，且發生完全癱瘓狀況，則 IMS 夥伴的所有 TPIPE 都會暫停。 在 IMS 夥伴通知已緩解洪水或洪水警告狀況 IBM MQ 之後，如果適當的話，將回復所有已暫停的 TPIPE ，並逐漸增加傳送交易要求的速率，直到達到速率上限為止。 IBM MQ 會發出主控台訊息，以回應 IMS 夥伴的狀態變更。

如果正在使用 IMS v10 夥伴，您應該確定已套用 PTF UK45082 。

從 IBM MQ 提交 IMS 交易

若要提交使用橋接器的 IMS 交易，應用程式會如常將訊息放置在 IBM MQ 佇列上。 訊息包含 IMS 交易資料; 它們可以具有 IMS 標頭 (MQIIH 結構) ，或容許 IBM MQ - IMS 橋接器對訊息中的資料進行假設。

然後， IBM MQ 會將訊息放入 IMS 佇列 (它會先在 IBM MQ 中排入佇列，以便能夠使用同步點來確保資料完整性)。 IBM MQ 佇列的儲存類別會決定佇列是否為 *OTMA* 佇列 (亦即，用來將訊息傳輸至 IBM MQ - IMS 橋接器的佇列) ， 以及訊息資料傳送至其中的特定 IMS 友機。

遠端佇列管理程式也可以透過寫入 IBM MQ for z/OS 上的這些 OTMA 佇列來啟動 IMS 交易。

從 IMS 系統傳回的資料會直接寫入訊息描述子結構 (MQMD) 中指定的 IBM MQ 回覆目的地佇列。 (這可能是 MQMD 的 **ReplyToQMgr** 欄位中指定之佇列管理程式的傳輸佇列。)

相關概念

IBM MQ for z/OS 上的 IMS 及 IMS 橋接器應用程式

相關工作

自訂 IMS 橋接器

相關參考

第 236 頁的『IBM MQ 及 IMS』

請利用這個主題來瞭解 IBM MQ 如何使用 IMS。 IMS 配接器可讓您將佇列管理程式連接至 IMS，並可讓 IMS 應用程式使用 MQI。

z/OS IBM MQ 及 z/OS 批次、TSO 和 RRS 配接器

請利用這個主題來瞭解 IBM MQ 如何使用「 z/OS 批次」、TSO 和 RRS 配接器。

批次配接器簡介

批次 /TSO 配接器是 IBM MQ 與在 JES、TSO 或 z/OS UNIX System Services 下執行的 z/OS 應用程式之間的介面。 這些配接器可讓 z/OS 應用程式使用 MQI。

配接器為以下列模式或狀態執行的程式提供對 IBM MQ 資源的存取權:

- 作業 (TCB) 模式
- 問題或監督者狀態
- 非跨記憶體模式
- 非存取登錄模式

應用程式與 IBM MQ 之間的連線位於作業層次。 配接器提供從應用程式作業控制區塊 (TCB) 到 IBM MQ 的連線執行緒。

對於 IBM MQ 所擁有資源的變更， Batch/TSO 配接器支援一段式確定通訊協定。 它不支援多階段確定通訊協定。 RRS 配接器可讓 IBM MQ 應用程式與其他啟用 RRS 的產品一起參與兩段式確定通訊協定，並由 z/OS 資源回復服務 (RRS) 協調。

配接器使用 z/OS STIMERM 服務來每秒排程非同步事件。 此事件執行的岔斷要求區塊 (IRB) 不涉及批次應用程式作業的任何等待。 此 IRB 會檢查是否已公佈 IBM MQ 終止 ECB。 如果已公佈終止 ECB， IRB 會公佈任何正在 IBM MQ 中等待事件 (例如，信號或等待) 的應用程式 ECB。

批次 /TSO 配接器

IBM MQ Batch/TSO 配接器為 z/OS Batch 和 TSO 應用程式提供 IBM MQ 支援。 在「 z/OS 批次」或 TSO 下執行的所有應用程式都必須使用它們來編輯 API Stub 程式 CSQBSTUB 鏈結。 Stub 可讓應用程式存取所有 MQI 呼叫。 您可以透過發出 MQI 呼叫 **MQCMIT** 及 **MQBACK**，對應用程式使用單一階段確定及取消。

RRS 配接卡

資源回復服務 (RRS) 是 z/OS 的子元件，提供全系統服務來協調 z/OS 產品之間的兩段式確定。 IBM MQ Batch/TSO RRS 配接器 (RRS 配接器) 為想要使用這些服務的 z/OS 批次和 TSO 應用程式提供 IBM MQ 支援。 RRS 配接卡可讓 IBM MQ 成為 RRS 協調中的完整參與者。 應用程式可以與支援 RRS 的其他產品 (例如 Db2) 一起參與兩段式確定處理。

RRS 配接器提供兩個 Stub; 您必須將想要使用 RRS 的應用程式鏈結至其中一個 Stub。

CSQBRSTB
此 Stub 可讓您使用 RRS 可呼叫資源回復服務，而非 MQI 呼叫 **MQCMIT** 和 **MQBACK**，對應用程式使用兩段式確定和取消。

您也必須從檔案庫 SYS1.CSSLIB 搭配您的應用程式。 如果您使用 MQI 呼叫 **MQCMIT** 及 **MQBACK**，則會收到回覆碼 MQRC_ENVIRONMENT_ERROR。

CSQBRRSI
此 Stub 可讓您使用 MQI 呼叫 **MQCMIT** 及 **MQBACK** ; IBM MQ 實際上會將這些呼叫實作為 **SRRCMIT** 和 **SRRBACK** RRS 呼叫。

如需建置使用 RRS 配接器的應用程式的相關資訊，請參閱 RRS 批次配接器。

在何處尋找「z/OS 批次」、TSO 和 RRS 配接器的相關資訊

您可以在下列來源中找到本節主題的相關資訊:

表 *25:* 在何處尋找使用 *z/OS Batch with IBM MQ* 的相關資訊	
主題	**在何處尋找**
設定批次配接器	作業 19: 設定批次、TSO 和 RRS 配接器
RRS 可呼叫資源回復服務	*MVS Programming: Callable Services for High Level Languages*

z/OS IBM MQ for z/OS 及 WebSphere Application Server

請利用這個主題來瞭解 WebSphere Application Server 如何使用 IBM MQ for z/OS 。

以 Java 撰寫且在 WebSphere Application Server 下執行的應用程式可以使用 Java Message Service (JMS) 規格來執行傳訊。 此環境中的點對點傳訊可以由 IBM MQ for z/OS 佇列管理程式提供。

使用 IBM MQ for z/OS 佇列管理程式來提供傳訊的好處是連接 JMS 應用程式可以完全參與 IBM MQ 網路的功能。 例如，他們可以使用 IMS 橋接器，或與在其他平台上執行的佇列管理程式交換訊息。

WebSphere Application Server 與佇列管理程式之間的連線

如需相關資訊，請參閱 一起使用 IBM MQ 和 WebSphere Application Server 。

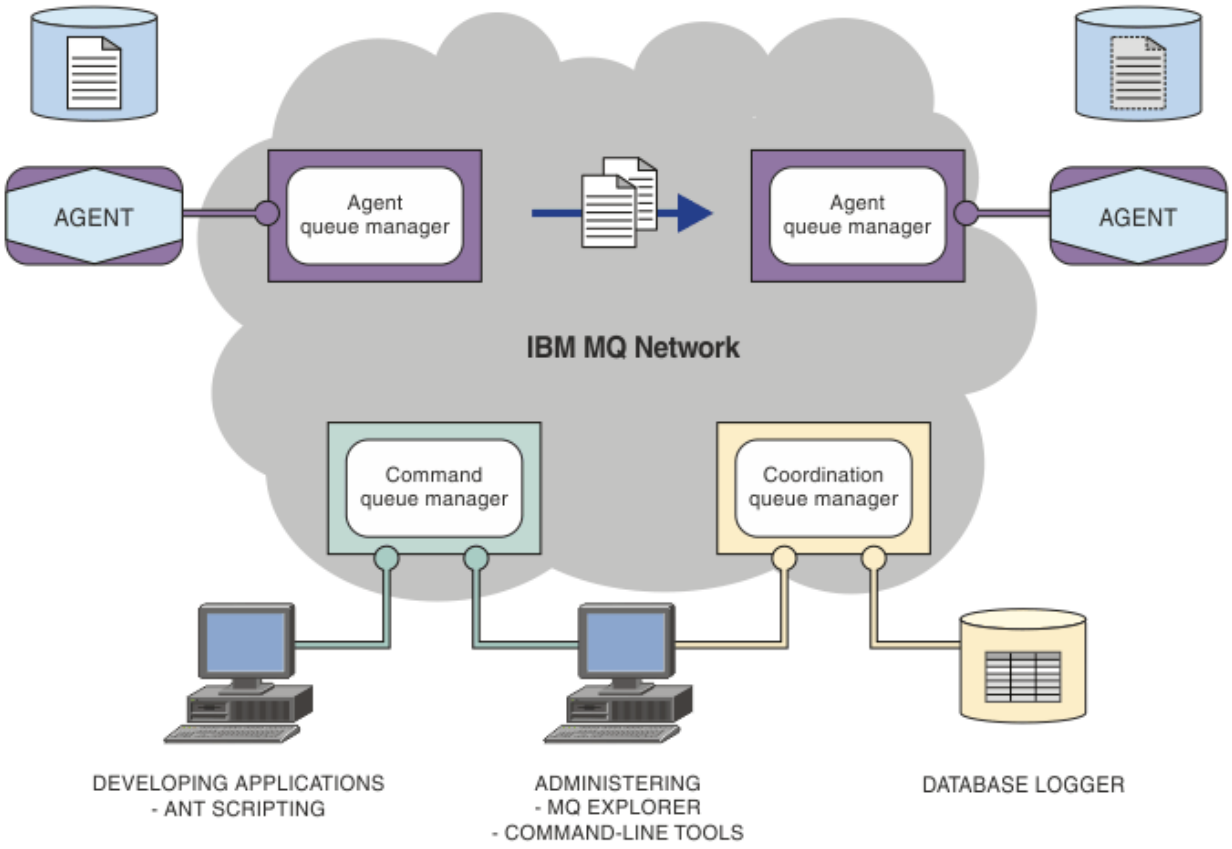
從 JMS 應用程式使用 IBM MQ 函數

依預設， IBM MQ 佇列上保留的 JMS 訊息會使用 MQRFH2 標頭來保留部分 JMS 訊息標頭資訊。 許多舊式 IBM MQ 應用程式無法處理具有這些標頭的訊息，且需要自己的性質標頭，例如 MQCIH for CICS Bridge , 或 MQWIH for IBM MQ Workflow 應用程式。 如需這些特殊考量的詳細資料，請參閱 將 JMS 訊息對映至 IBM MQ 訊息。

Managed File Transfer

不論檔案大小或使用的作業系統為何，Managed File Transfer 皆會以受管理且可審核的方式在系統之間傳送檔案。

您可以使用 Managed File Transfer 來建置自訂、可擴充及自動化的解決方案，讓您能夠管理、信任及維護檔案傳送的安全。 Managed File Transfer 刪除了高成本的備用項目、降低維護成本，並使現有的 IT 投資獲得最大的效益。



此圖表顯示簡易 Managed File Transfer 拓蹼。 在 IBM MQ 網路中有兩個代理程式，各自連接到專屬的代理程式佇列管理程式。 檔案從圖表一側的代理程式，經由 IBM MQ 網路，傳送至圖表另一側的代理程式。 此外，在 IBM MQ 網路中，還有協調佇列管理程式及指令佇列管理程式。 應用程式及工具會連接至這些佇列管理程式，以在 IBM MQ 網路中配置、管理、操作及記載 Managed File Transfer 活動。

視您的作業系統及整體設定而定，Managed File Transfer 可安裝為四個不同的選項。 這些選項是 Managed File Transfer Agent、Managed File Transfer Logger、Managed File Transfer Service 或 Managed File Transfer Tools。 如需相關資訊，請參閱 Managed File Transfer 產品選項。

您可以使用 Managed File Transfer 來執行下列作業：

- 建立 Managed File Transfer
  - Windows Linux 在 Linux 或 Windows 平台上，從 IBM MQ Explorer 建立新的檔案傳送。
  - 從所有受支援平台的指令行建立新的檔案傳送。
  - 將檔案傳送功能整合至 Apache Ant 工具。
  - 撰寫應用程式，以透過將訊息放置在代理程式指令佇列上來對 Managed File Transfer 進行控制。
  - 將檔案傳送排定於稍後進行。 您也可以根據一系列檔案系統事件（例如建立新檔案）來觸發排定的檔案傳送。
  - 持續監視資源，例如目錄，然後當該資源的內容符合一些預先定義的條件時，啟動作業。 此作業可以是檔案傳送、 Ant Script 或 JCL 工作。
  - 與 IBM MQ 佇列來回傳送檔案。
  - 與 FTP、FTPS 或 SFTP 伺服器來回傳送檔案。
  - 與 Connect:Direct 節點來回傳送檔案。
  - 傳送文字及二進位檔。 會在來源與目的地系統的字碼頁及行尾使用慣例之間自動轉換文字檔。
  - 可使用以 Secure Socket Layer (SSL) 基礎連線的業界標準來維護傳送安全。

- 檢視進行中的傳送，並記載網路中所有傳送的相關資訊
  - Windows Linux 在 Linux 或 Windows 平台上，從 IBM MQ Explorer 檢視進行中傳送的狀態。
  - Windows Linux 在 Linux 或 Windows 平台上使用 IBM MQ Explorer 來檢查已完成傳送的狀態。
  - 使用 Managed File Transfer 資料庫日誌程式特性，將日誌訊息儲存至 Db2 或 Oracle 資料庫。

Managed File Transfer 建置在 IBM MQ 上，其提供應用程式之間保證唯一一次的訊息遞送。 您可以充分運用 IBM MQ 的各種特性。 例如，您可以使用通道壓縮來壓縮您在代理程式之間透過 IBM MQ 通道傳送的資料，以及使用 SSL 通道來維護在代理程式之間傳送的資料的安全。 可靠的檔案傳送，並可容忍執行檔案傳送的基礎架構失敗。 如果您發生網路中斷，當連線功能還原時，檔案傳送會從它停止的位置重新啟動。

合併檔案傳送與現有的 IBM MQ 網路，您可以避免浪費維護兩個不同基礎架構所需的資源。 如果您還不是 IBM MQ 客戶，則可以透過建立 IBM MQ 網路來支援 Managed File Transfer，以建置未來 SOA 實作的骨幹。 如果您已經是 IBM MQ 客戶，Managed File Transfer 可以充分運用您現有的 IBM MQ 基礎架構，包括 IBM MQ Internet Pass-Thru 及 IBM Integration Bus。

您可以利用 IBM MQ 高可用性解決方案來改善 Managed File Transfer 配置的備援。 如果您的代理程式使用抄寫的資料佇列管理程式 (RDQM)，則必須將它們配置為使用浮動 IP 位址特性。 這表示代理程式會使用相同的 IP 位址來與三個 RDQM 實例中的任何一個進行通訊，並在失效接手時自動重新連接 (請參閱 RDQM 高可用性 及 建立及刪除浮動 IP 位址)。 如果您使用多重實例佇列管理程式解決方案，則應用程式會使用不同的 IP 位址來與每一個實例進行通訊，失效接手時由用戶端重新連接處理 (請參閱 多重實例佇列管理程式 及 通道與用戶端重新連線)。

整合 Managed File Transfer 與一些其他 IBM 產品：

IBM Integration Bus

在 IBM Integration Bus 流程中由 Managed File Transfer 傳送的處理程序檔案。 如需相關資訊，請參閱 從 IBM Integration Bus 使用 MFT。

IBM Sterling Connect:Direct

使用 Managed File Transfer Connect:Direct 橋接器，在現有 Connect:Direct 網路中來回傳送檔案。 如需相關資訊，請參閱 Connect:Direct 橋接器。

IBM Tivoli Composite Application Manager

IBM Tivoli Composite Application Manager 提供的代理程式可讓您用來監視已發佈至協調佇列管理程式的資訊。

相關概念

Managed File Transfer 產品選項

第 243 頁的『MFT 拓蹼概觀』
有關 Managed File Transfer 代理程式如何連接 IBM MQ 網路中的協調佇列管理程式的概觀。

第 242 頁的『MFT 如何與 IBM MQ 配合運作? 』
Managed File Transfer 以數種方式與 IBM MQ 進行互動。

MFT 如何與 IBM MQ 配合運作?

Managed File Transfer 以數種方式與 IBM MQ 進行互動。

- Managed File Transfer 透過以下方式在代理程式程序之間傳送檔案：將每個檔案分割為一則以上的訊息，然後透過 IBM MQ 網路傳輸訊息。
- 代理程式程序使用非持續性訊息來移動檔案資料，以最大程度降低對 IBM MQ 日誌的影響。 透過彼此間進行通訊，代理程式程序可以限定包含檔案資料的訊息流。 如此可避免包含檔案資料的訊息聚集在 IBM MQ 傳輸佇列，並確保如果有任何非持續性訊息未遞送，則再次傳送檔案資料。
- Managed File Transfer 代理程式使用數個 IBM MQ 佇列。 如需相關資訊，請參閱 MFT 系統佇列及系統主題。

- 雖然其中部分佇列嚴格限於內部使用，但代理程式可以接受傳送至它可讀取的特定佇列，且為特殊格式化指令訊息形式的要求。 指令行指令及「IBM MQ Explorer」外掛程式，都可以將 IBM MQ 訊息傳送至代理程式，以指示代理程式執行所需的動作。 您可以撰寫以這種方式與代理程式進行互動的 IBM MQ 應用程式。 如需相關資訊，請參閱 透過將訊息放置在代理程式指令佇列上來控制 MFT。
- Managed File Transfer 代理程式會將其傳送狀態、進度及結果的相關資訊，傳送至已指定為協調佇列管理程式的 MQ 佇列管理程式。 此類資訊由協調佇列管理程式發佈，可由想要監視傳送進度或記錄已執行傳送的應用程式訂閱。 指令行指令及「IBM MQ Explorer」外掛程式，都可以使用所發佈的資訊。 您可以撰寫使用這項資訊的 IBM MQ 應用程式。 如需將資訊發佈至其中之主題的相關資訊，請參閱 SYSTEM.FTE topic。
- Managed File Transfer 的主要元件可利用 IBM MQ 佇列管理程式的功能，來儲存及轉遞訊息。 這意味著如果發生停電，基礎架構中未受影響的部分可以繼續傳送檔案。 此情況適用於協調佇列管理程式，其中結合使用儲存及轉遞與可延續訂閱可讓協調佇列管理程式容忍無法使用情況，且不會遺失已執行檔案傳送的主要資訊。

MFT 拓蹼概觀

有關 Managed File Transfer 代理程式如何連接 IBM MQ 網路中的協調佇列管理程式的概觀。

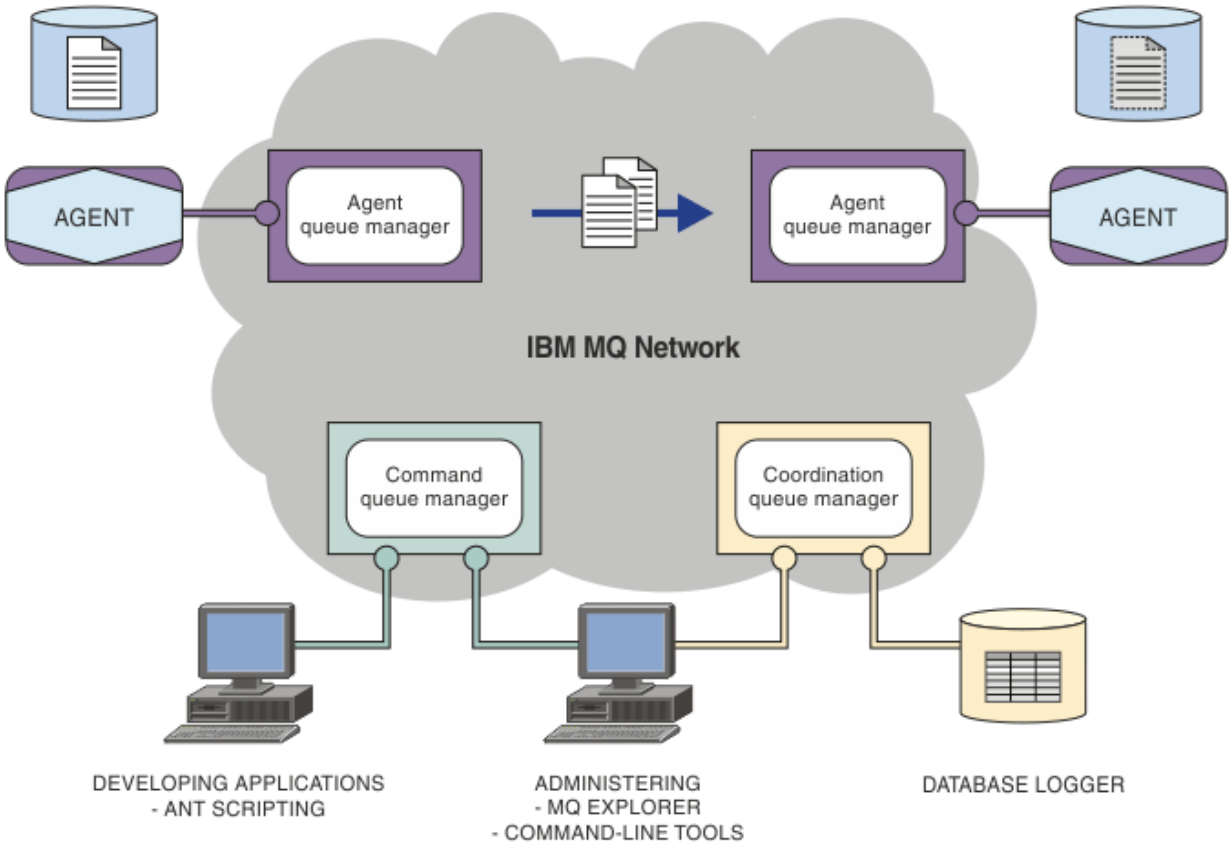
Managed File Transfer 代理程式會傳送及接收已傳送的檔案。 每一個代理程式在其關聯的佇列管理程式上都有其專屬的一組佇列，且代理程式是以連結或用戶端模式附加至其佇列管理程式。 代理程式也可以使用協調佇列管理程式作為其佇列管理程式。

協調佇列管理程式會播送審核及檔案傳送資訊。 協調佇列管理程式代表收集代理程式、傳送狀態及傳送審核資訊的單一點。 進行傳送時，不一定要有協調佇列管理程式。 如果協調佇列管理程式暫時變成無法使用，傳送會像平常一樣繼續進行。 審核及狀態訊息會儲存在代理程式佇列管理程式中，直到協調佇列管理程式變成可用，然後就可以正常地進行處理。

代理程式會向協調佇列管理程式登錄，並將其詳細資料發佈給該佇列管理程式。 Managed File Transfer 外掛程式會使用此代理程式資訊，開始從「IBM MQ Explorer」進行傳送。 這些指令也會使用在協調佇列管理程式上收集的代理程式資訊，以顯示代理程式資訊及代理程式狀態。

傳送狀態及傳送審核資訊會發佈在協調佇列管理程式上。 Managed File Transfer 外掛程式使用傳送狀態及傳送審核資訊，以監視從「IBM MQ Explorer」傳送的進度。 可保留儲存在協調佇列管理程式上的傳送審核資訊，以提供審核性。

指令佇列管理程式可以用來連接至 IBM MQ 網路，這也是您發出 Managed File Transfer 指令時所連接的佇列管理程式。



相關概念

第 240 頁的『Managed File Transfer』
不論檔案大小或使用的作業系統為何，Managed File Transfer 皆會以受管理且可審核的方式在系統之間傳送檔案。

第 242 頁的『MFT 如何與 IBM MQ 配合運作？』
Managed File Transfer 以數種方式與 IBM MQ 進行互動。

Managed File Transfer 實務範例

MFT REST API 概觀

REST API 支援某些 Managed File Transfer 指令，包括列出傳送，以及檔案傳送代理程式的詳細資料。

從 IBM MQ 9.1.0 開始， REST API 包括列出所有現行 Managed File Transfer 傳送及查詢 Managed File Transfer 代理程式狀態的選項。 如需相關資訊，請參閱 開始使用 REST API MFT。

IBM MQ Internet Pass-Thru

IBM MQ Internet Pass-Thru (MQIPT) 是 IBM MQ 的選用元件，可用來跨網際網路在遠端網站之間實作傳訊解決方案。

從 IBM MQ 9.2.0 開始， MQIPT 是 IBM MQ 的選用元件。 若要取得 IBM MQ 9.3.x 的 MQIPT 安裝檔案，請跳至 IBM Fix Central for IBM MQ。 在 IBM MQ 9.2.0 之前， MQIPT 提供作為支援套件。

您不需要執行 IBM MQ 9.3 ，即可在 IBM MQ 9.3 中使用 MQIPT 。 您可以使用 MQIPT 來連接任何受支援的 IBM MQ 版本，且不需要安裝與 MQIPT 版本相同的任何其他 IBM MQ 元件。

如果您已購買 IBM MQ 授權，則可以安裝所需數量的 MQIPT 副本。 MQIPT 安裝不計入您購買的 IBM MQ 授權。 如需 IBM MQ 授權的相關資訊，請參閱 IBM MQ 授權資訊。

註: 本文件與 IBM MQ 9.3 中的 MQIPT 相關。 如需 IBM Documentation 中的 MQIPT 支援套件 (2.1) 說明文件，請參閱 IBM MQ 9.0 說明文件中的 MQIPT (SupportPac MS81) 。

註: 如果您使用 MQIPT 2.1 或更舊版本，建議您升級至 MQIPT for IBM MQ 9.3，因為 MQIPT 支援套件的支援結束日期是 2020 年 9 月 30th 。

IBM MQ Internet Pass-Thru 以獨立式服務方式執行，可接收及轉遞兩個 IBM MQ 佇列管理程式之間或 IBM MQ 用戶端與 IBM MQ 佇列管理程式之間的 IBM MQ 訊息流程。

當用戶端和伺服器不在相同的實體網路上時， MQIPT 會啟用此連線。

MQIPT 的一個以上實例可以放置在兩個 IBM MQ 佇列管理程式之間或 IBM MQ 用戶端與 IBM MQ 佇列管理程式之間的通訊路徑中。 MQIPT 的實例可讓兩個 IBM MQ 系統交換訊息，而在兩個系統之間不需要直接 TCP/IP 連線。 如果防火牆配置禁止兩個系統之間的直接 TCP/IP 連線，這會很有用。

MQIPT 會在一或多個 TCP/IP 埠上接聽送入的連線，這些連線可以攜帶一般 IBM MQ 訊息、在 HTTP 內導通的 IBM MQ 訊息，或使用「傳輸層安全 (TLS)」或 Secure Sockets Layer (SSL) 加密的訊息。 MQIPT 可以處理多個並行連線。

發出起始 TCP/IP 連線要求的 IBM MQ 通道稱為 呼叫端，它嘗試以 回應端身分連接的通道，以及它最終嘗試以 目的地佇列管理程式身分聯絡的佇列管理程式。

當 MQIPT 將資料從其來源轉遞至其目的地時，它會將資料保留在記憶體中。 磁碟上未儲存任何資料 (作業系統分頁至磁碟的記憶體除外)。 MQIPT 明確存取磁碟的唯一時間是讀取其配置檔，以及寫入連線日誌和追蹤記錄。

IBM MQ 通道類型的完整範圍可以透過一或多個 MQIPT 實例來建立。 通訊路徑中存在 MQIPT 不會影響已連接 IBM MQ 元件的功能性質，但可能會影響訊息傳送的效能。

MQIPT 可以與 IBM MQ 和 IBM Integration Bus 一起使用，如 第 248 頁的『MQIPT 的可能配置』中所述。

若要安裝 MQIPT，請參閱 安裝 MQIPT。

相關工作

正在配置 IBM MQ Internet Pass-Thru

管理及配置 IBM MQ Internet Pass-Thru

相關參考

IBM MQ Internet Pass-Thru 配置參考資料

MQIPT 的使用

IBM MQ Internet Pass-Thru (MQIPT) 有許多可能的用途。

MQIPT 可以用作通道集中器

透過以這種方式使用 MQIPT ，與多個個別主機之間的通道可以在防火牆中出現，就好像它們都是與 MQIPT 主機之間的通道一樣。 這可讓您更容易定義及管理防火牆過濾規則。

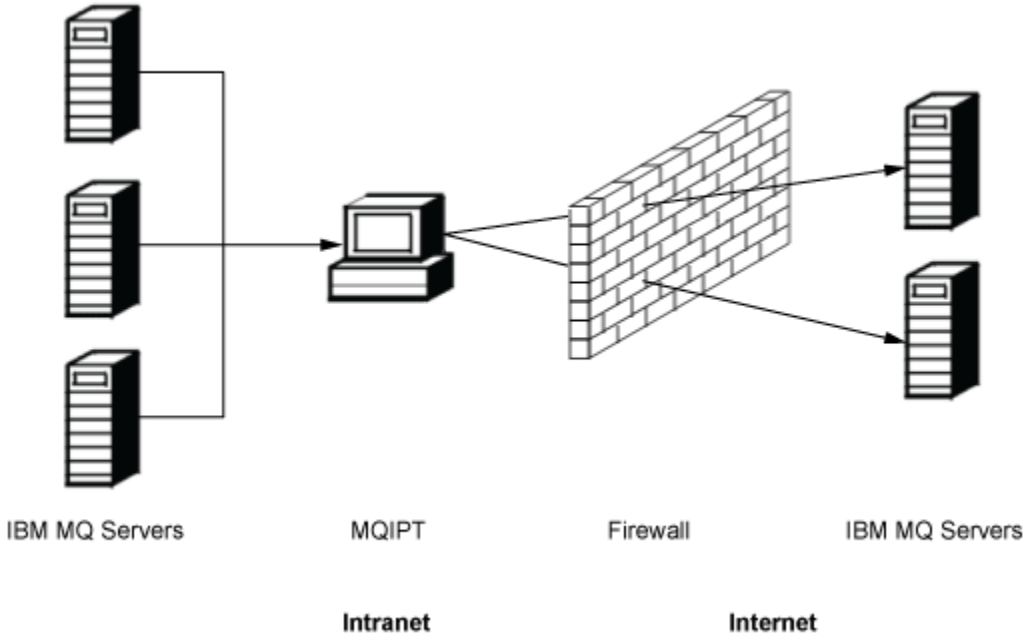


圖 *79: MQIPT* 作為通道集中器的範例

MQIPT 可以放在 DMZ 中，以提供單一存取點

在具有已知且授信網際網路通訊協定 (IP) 位址的電腦上，如果 MQIPT 放置在 DMZ 防火牆 (用來保護區域網路安全的防火牆配置) 內，則 MQIPT 可以用來接聽送入的 IBM MQ 通道連線，然後它可以轉遞至授信內部網路; 內部防火牆必須容許此授信電腦建立入埠連線。 在此配置中， MQIPT 會防止外部存取要求接收授信內部網路中電腦的真實 IP 位址。 以此方式， MQIPT 提供單一存取點。 必要的話， MQIPT 可以配置為接受 TLS 連線，並使用個別 TLS 連線將資料轉遞至目的地，因此在 DMZ 中終止 TLS 階段作業。

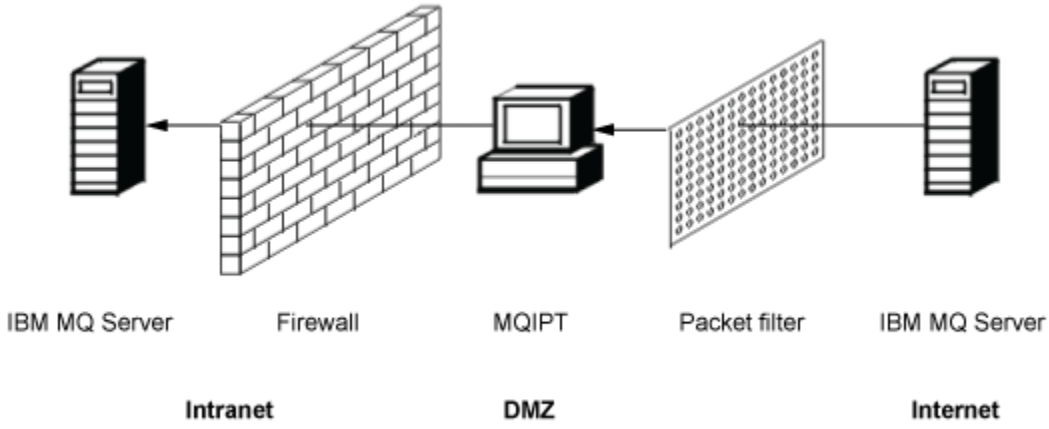


圖 *80: DMZ* 防火牆中的 *MQIPT* 範例

MQIPT 可以透過 HTTP 通道作業進行通訊

如果在行中部署兩個 MQIPT 實例，則它們可以使用 HTTP 進行通訊。 HTTP 通道作業特性可讓您利用現有的 HTTP Proxy ，透過防火牆來傳輸要求。 第一個 MQIPT 會將 IBM MQ 通訊協定插入 HTTP 中，第二個會從其 HTTP 封套中擷取 IBM MQ 通訊協定，並將它轉遞至目的地佇列管理程式。

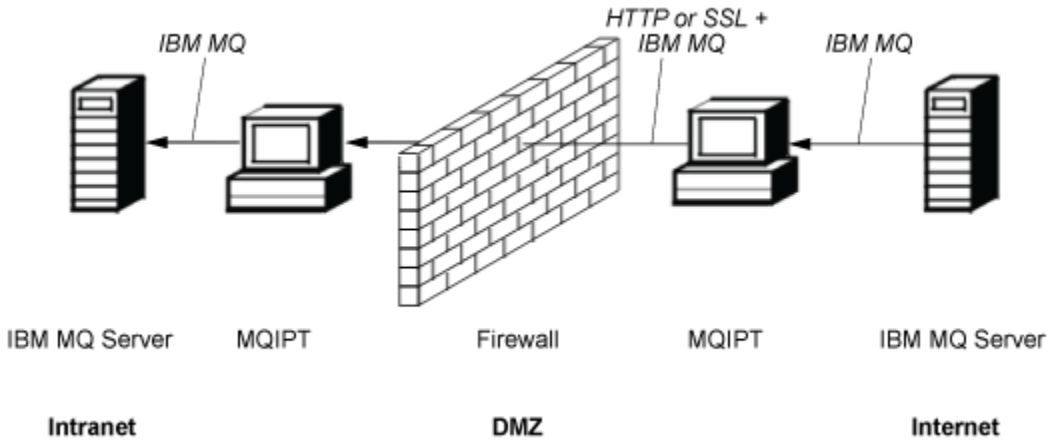


圖 *81: MQIPT* 和 *HTTP* 通道作業的範例

MQIPT 可以加密訊息

如果如上述範例般配置 MQIPT，在透過防火牆傳輸之前，可以先加密要求。 第一個 MQIPT 會加密資料，第二個會先使用 SSL/TLS 來解密資料，然後再將它傳送至目的地佇列管理程式。

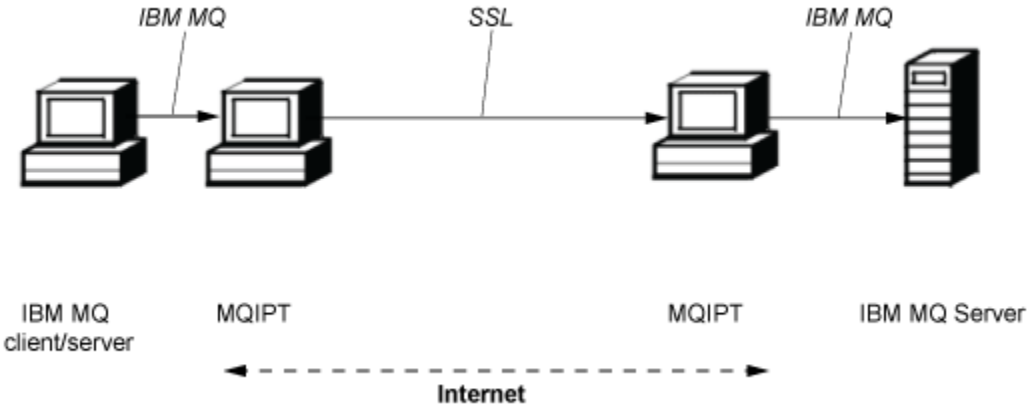


圖 *82: MQIPT* 及 *SSL/TLS* 的範例

MQIPT 工作原理

在其最簡單的配置中， MQIPT 充當 IBM MQ 通訊協定轉遞程式。 它會在 TCP/IP 埠上接聽，並接受來自 IBM MQ 通道的連線要求。

如果收到形式完整的要求， MQIPT 會在本身與目的地 IBM MQ 佇列管理程式之間建立進一步的 TCP/IP 連線。 然後，它會將從其送入連線接收的所有通訊協定封包傳遞至目的地佇列管理程式，並將來自目的地佇列管理程式的通訊協定封包傳回至原始送入連線。

不會涉及 IBM MQ 通訊協定 (用戶端/伺服器或佇列管理程式至佇列管理程式) 的變更，因為兩端都不直接知道中介的存在。 不需要新版本的 IBM MQ 用戶端或伺服器程式碼。

如果要使用 MQIPT，呼叫端通道必須配置成使用 MQIPT 主機名稱和埠，而不是目的地佇列管理程式的主機名稱和埠。 這是使用 IBM MQ 通道的 **CONNAME** 內容來定義。 MQIPT 會讀取送入的資料，並將其直接傳遞至目的地佇列管理程式。 其他配置欄位 (例如主從式通道中的使用者 ID 及密碼) 也會以類似方式傳遞至目的地佇列管理程式。

多個佇列管理程式

MQIPT 可用來容許存取多個目的地佇列管理程式。 若要這樣做，必須有一種機制可告知 MQIPT 要連接的佇列管理程式，因此 MQIPT 會使用送入的 TCP/IP 埠號來決定要連接的佇列管理程式。

因此，您可以配置 MQIPT 在多個 TCP/IP 埠上接聽。 每一個接聽埠都會透過 MQIPT 路徑對映至目的地佇列管理程式。 您最多可以定義 100 個這類路徑，以將接聽 TCP/IP 埠與目的地佇列管理程式的主機名稱及埠相關聯。 這表示目的地佇列管理程式的主機名稱 (IP 位址) 絕不會對原始通道可見。 每一個路徑可以處理其接聽埠與目的地之間的多個連線，每一個連線都獨立運作。

MQIPT 配置檔

MQIPT 使用稱為 `mqipt.conf` 的配置檔。 此檔案包含所有路徑及其相關聯內容的定義。 如需 `mqipt.conf` 的相關資訊，請參閱 管理及配置 IBM MQ Internet Pass-Thru 。

當啟動 MQIPT 時，它會啟動配置檔中列出的每一個路徑。 訊息會寫入系統主控台，以顯示每一個路徑的狀態。 當針對路徑顯示訊息 MQCPI078 時，該路徑已備妥可接受連線要求。

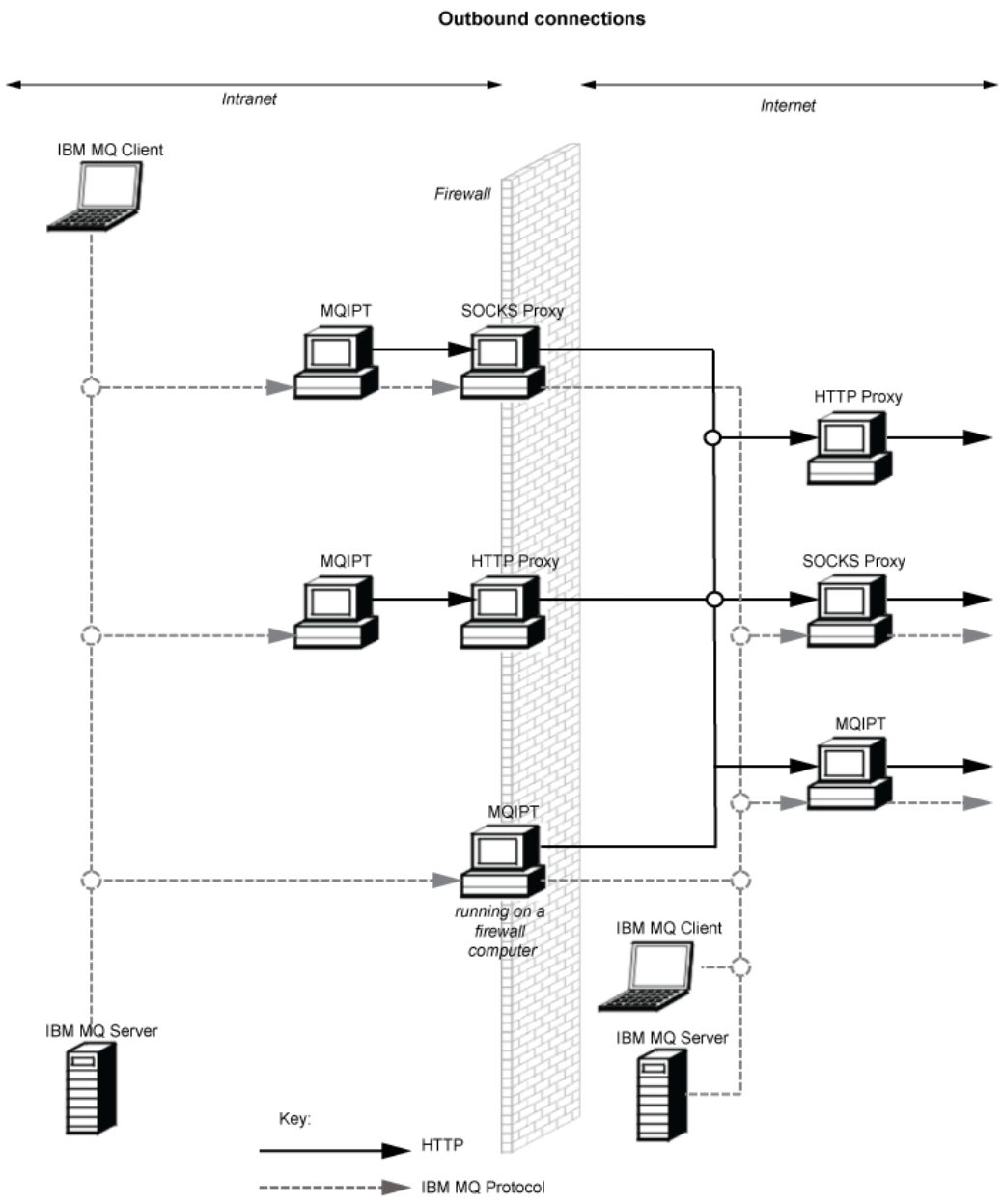
MQIPT 的可能配置

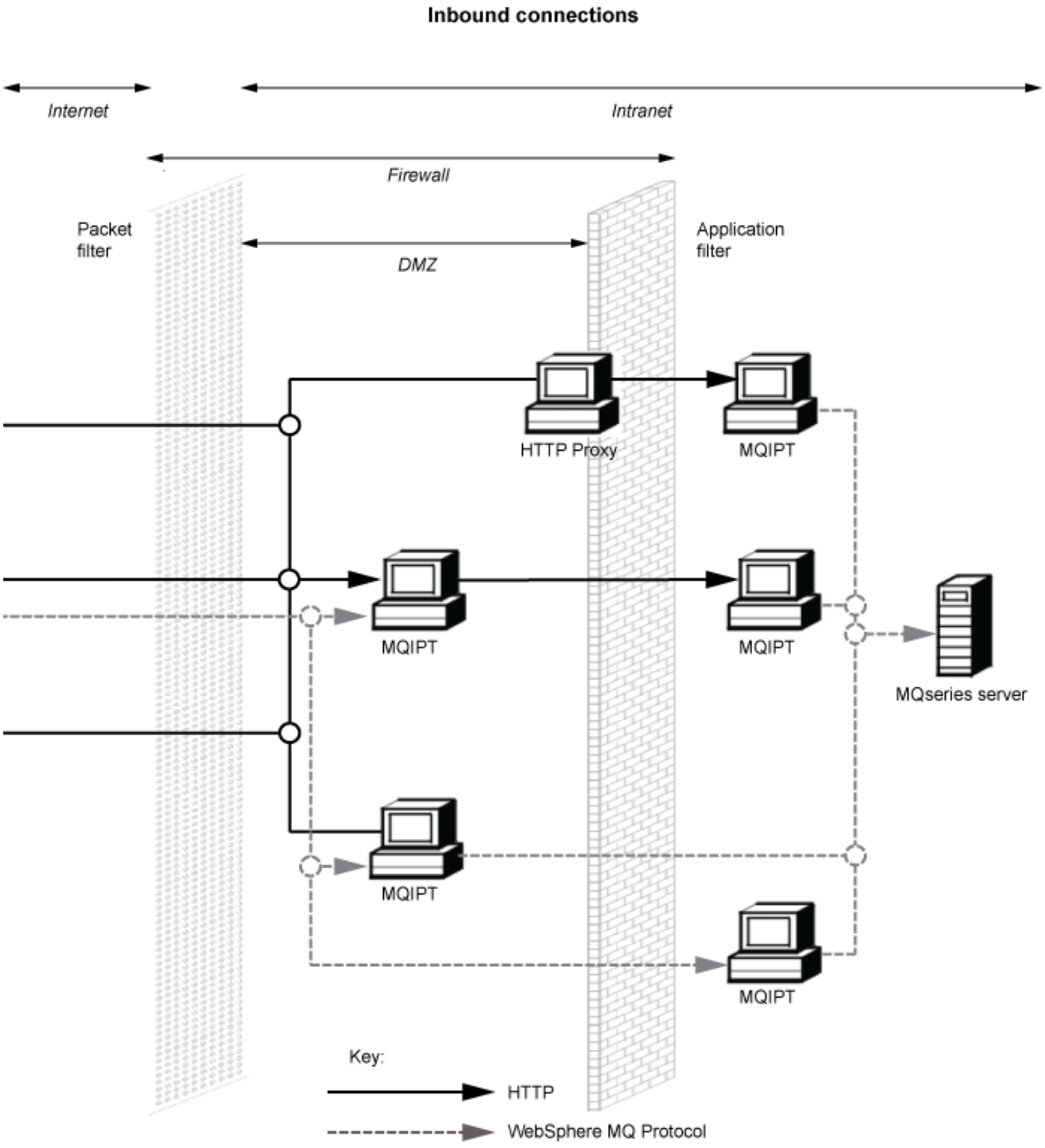
MQIPT 可以與 IBM MQ 和 IBM Integration Bus 一起使用。

下列多組件圖顯示 IBM MQ 拓蹼中 MQIPT 的許多可能配置。 它說明 MQIPT 傳送訊息的不同方式。 它顯示內部網路、防火牆內及防火牆外網際網路上的用戶端及伺服器，將訊息傳遞至轉遞它們的 MQIPT、 HTTP Proxy 或 SOCKS Proxy。

在透過入埠防火牆將訊息傳遞至伺服器之前， DMZ 中的 MQIPT Proxy 或 HTTP Proxy 會先接收訊息。

請注意，防火牆內部網路端上的 HTTP Proxy、SOCKS Proxy 及 MQIPT 電腦，代表多部電腦在網際網路上鏈結在一起的可能性。 例如，在達到目標之前， MQIPT 電腦可以透過一或多個 SOCKS 或 HTTP Proxy 電腦，或進一步 MQIPT 電腦進行通訊。


Outbound connections
Intranet
Internet
IBM MQ Client
Firewall
MQIPT
SOCKS Proxy
HTTP Proxy
MQIPT
HTTP Proxy
SOCKS Proxy
MQIPT
MQIPT
running on a firewall computer
IBM MQ Client
IBM MQ Server
IBM MQ Server
Key:
HTTP
IBM MQ Protocol



相容配置

相容的連線實務範例，其中 IBM MQ 用戶端或佇列管理程式會與 MQIPT 通訊。 使用相同或第二個 MQIPT 路徑來與目的地佇列管理程式進行通訊。

與單一 MQIPT 路徑相容的配置

您可以使用單一 MQIPT 路徑來與 IBM MQ 通訊。

第 251 頁的表 26 中的直欄包含下列資訊:

1. 在 IBM MQ 與 MQIPT 路徑之間使用的通訊協定。 連線可以由 IBM MQ 用戶端或佇列管理程式建立，並且可以使用 IBM MQ 格式及通訊協定 (FAP) 或 SSL/TLS 通訊協定。
2. MQIPT 路徑的運作模式。 MQIPT 與 IBM MQ 之間透過網際網路進行通訊的格式，由 MQIPT 路徑的配置決定。 請注意，當表格提及 SSL 時，您也可以使用 TLS。
3. 在 MQIPT 路徑與目的地佇列管理程式之間使用的通訊協定。

表 *26:* 具有單一 *MQIPT* 實例的有效配置

1. IBM MQ 來源通訊協定	**2. MQIPT 路徑的模式**	**3. IBM MQ 目的地通訊協定**
FAP	FAP-Proxy (預設值)	FAP
	FAP-伺服器和 SSL-用戶端	SSL/TLS
SSL/TLS	SSL-Proxy	SSL/TLS
	SSL 伺服器及 FAP 用戶端	FAP
	SSL 伺服器和 SSL 用戶端	SSL/TLS

與多個 MQIPT 路徑相容的配置

您可以選擇在一個以上 MQIPT 實例上使用多個路徑，以與 IBM MQ 進行通訊。

第 251 頁的表 27 中的直欄包含下列資訊:

1. 在 IBM MQ 與第一個 MQIPT 路徑之間使用的通訊協定。 連線可以由 IBM MQ 用戶端或佇列管理程式建立，並且可以使用 IBM MQ 格式及通訊協定 (FAP) 或 SSL/TLS 通訊協定。
2. 第一個 MQIPT 路徑運作的模式。 MQIPT 與 IBM MQ 之間透過網際網路進行通訊的格式，由 MQIPT 路徑的配置決定。 請注意，當表格提及 SSL 時，您也可以使用 TLS。
3. 第二個 MQIPT 路徑的運作模式。
4. 在第二個 MQIPT 路徑與目的地佇列管理程式之間使用的通訊協定。

表 *27:* 具有多個 *MQIPT* 實例的有效配置

1. IBM MQ 來源通訊協定	**2. 第一個 MQIPT 路徑的模式**	**3. 第二個 MQIPT 路徑的模式**	**4. IBM MQ 目的地通訊協定**
FAP (預設值)	FAP-Proxy (預設值)	FAP-Proxy (預設值)	FAP
	FAP-伺服器和 SSL-用戶端	SSL-Proxy	SSL/TLS
		SSL 伺服器及 FAP 用戶端	FAP
		SSL 伺服器和 SSL 用戶端	SSL/TLS
	HTTP-client	HTTP-伺服器及 SSL-用戶端	SSL/TLS
	HTTPS-client	HTTPS-伺服器和 SSL-用戶端	SSL/TLS
	HTTP-client	http-server	FAP
	HTTPS-client	HTTPS-server	FAP

表 27: 具有多個 MQIPT 實例的有效配置 (繼續)			
1. IBM MQ 來源通訊協定	**2. 第一個 MQIPT 路徑的模式**	**3. 第二個 MQIPT 路徑的模式**	**4. IBM MQ 目的地通訊協定**
SSL/TLS	SSL-Proxy	SSL-Proxy	SSL/TLS
		SSL 伺服器及 FAP 用戶端	FAP
		SSL 伺服器和 SSL 用戶端	SSL/TLS
	HTTP-client	http-server	FAP
	HTTPS-client	HTTPS-server	SSL/TLS
	HTTP-client	HTTP-伺服器及 SSL-用戶端	FAP
	HTTPS-client	HTTPS-伺服器和 SSL-用戶端	SSL/TLS

支援的通道配置

支援所有 IBM MQ 通道類型，但配置限制為 TCP/IP 連線。 對於 IBM MQ 用戶端或佇列管理程式， MQIPT 會像它是目的地佇列管理程式一樣出現。 當通道配置需要目的地主機和埠號時，會指定 MQIPT 主機名稱和接聽器埠號。

主從架構通道
MQIPT 會接聽送入的用戶端連線要求，然後使用 HTTP 通道作業、SSL/TLS 或作為標準 IBM MQ 通訊協定封包來轉遞它們。 如果 MQIPT 使用 HTTP 通道作業或 SSL/TLS ，則會將它們在連線上轉遞至第二個 MQIPT。 如果未使用 HTTP 通道作業，它會在連線上將它們轉遞至它所看到的目的地佇列管理程式 (雖然這可能是進一步的 MQIPT)。 當目的地佇列管理程式已接受用戶端連線時，會在用戶端與伺服器之間傳送封包。

叢集傳送端/接收端通道
如果 MQIPT 收到來自叢集傳送端通道的送入要求，則會假設佇列管理程式已啟用 SOCKS ，並在 SOCKS 信號交換程序期間取得真正的目的地位址。 它會以與用戶端連線通道完全相同的方式，將要求轉遞至下一個 MQIPT 或目的地佇列管理程式。 這也包括自動定義的叢集傳送端通道。

傳送端/接收端
如果 MQIPT 收到來自傳送端通道的送入要求，它會以與用戶端連線通道完全相同的方式，將它轉遞至下一個 MQIPT 或目的地佇列管理程式。 目的地佇列管理程式會驗證送入的要求，並在適當時啟動接收端通道。 傳送端與接收端通道之間的所有通訊 (包括安全流程) 都會傳送。

要求端/伺服器
此組合的處理方式與之前的配置相同。 連線要求的驗證由目的地佇列管理程式上的伺服器通道執行。

要求者/傳送者
如果不容許兩個佇列管理程式彼此建立直接連線，但同時容許連接至 MQIPT 並接受來自它的連線，則可以使用「回呼」配置。

伺服器/要求程式及伺服器/接收端
這些由 MQIPT 以處理 `Sender/Receiver` 配置的相同方式來處理。

通道終止及失敗狀況

當 MQIPT 偵測到 IBM MQ 通道關閉 (正常或異常) 時，它會傳播通道關閉。 如果您使用 MQIPT 來關閉路徑，則會關閉所有通過該路徑的通道。

MQIPT 提供選用的閒置-逾時機能。 如果 MQIPT 偵測到通道已閒置超過逾時的時間，則會對有問題的兩個連線執行立即關閉。

通道任一端的 IBM MQ 系統會在網路故障或其夥伴終止通道時，觀察到這些異常關閉狀況。 然後，通道可以重新啟動並回復 (如果在通訊協定不確定期間發生失敗) ，如同未使用 MQIPT 一樣。

訊息安全

IBM MQ 分散式佇列管理可確保適當地遞送訊息。 這仍然是通道兩端之間存在 MQIPT 的情況。 MQIPT 不會儲存任何訊息資料或參與同步點程序，以確保正確的訊息遞送。

使用快速、非持續性 IBM MQ 訊息時，如果 MQIPT 路徑失敗，或在 IBM MQ 訊息傳輸時重新啟動，則訊息可能會遺失。 在重新啟動路徑之前，請確定所有使用 MQIPT 路徑的 IBM MQ 通道都處於非作用中狀態。

如需 IBM MQ 中訊息安全的相關資訊，請參閱 訊息安全。

多重實例佇列管理程式及高可用性

在高可用性環境中， MQIPT 可以與多重實例佇列管理程式搭配使用。

MQIPT 沒有持續性狀態，因此將 MQIPT 失效接手至另一個系統沒有好處。 相反地，有多個 MQIPT 實例在不同系統上執行，且具有相同的 `mqipt.conf` 配置檔。 監視每一個 MQIPT 實例的可用性，並在必要時重新啟動它 (在相同系統上)。 這提供一組可用來遞送連線的相同 MQIPT 實例。 然後，您必須確保 IBM MQ 可以將連線遞送至 MQIPT ，並且 MQIPT 可以將這些連線轉遞至目的地佇列管理程式。

出埠 IBM MQ 通道可以透過各種方式導向至可用的 MQIPT 實例，例如:

- 使用負載平衡器或高可用性路由器，例如 WebSphere Edge Components 產品中的 IBM Network Dispatcher 。
- 使用逗點區隔清單，在 IBM MQ 通道定義中指定多個連線名稱。 然後， IBM MQ 會嘗試依序連接至每一個 MQIPT 位址，直到找到可用的 MQIPT 實例為止。

您也必須將連線從 MQIPT 導向至目的地佇列管理程式。 如果高可用性配置確保 IP 位址由目的地佇列管理程式進行失效接手，則不需要特殊 MQIPT 配置: 在 **Destination** 路徑內容中指定目的地 IP 位址，並容許失效接手作業隨佇列管理程式一起移動 IP 位址。

不過，如果佇列管理程式的 IP 位址在失效接手之後變更，則您必須安排 MQIPT 將連線轉遞至正確的目的地。 這可以透過下列數種方式之一來完成:

- 撰寫遞送結束程式，以檢查可存取的 IP 位址及埠號，然後置換每一個連線的遞送目的地。 部分範例遞送結束程式隨 MQIPT 提供; 它們可以針對此目的進行調整。
- 使用高可用性負載平衡器來重新導向連線。
- 定義多個 MQIPT 路徑，佇列管理程式可能執行所在的每一個 IP 位址及埠各一個。 然後將 IBM MQ 連線導向至各種 MQIPT 路徑，例如，在出埠通道的連線名稱中以逗點區隔清單列出所有路徑 IP 位址及埠號。

調整網路路徑上的所有端對端元件也很重要:

1. 無法使用系統的連線嘗試必須立即失敗，以便重新連接嘗試可以移至第一個可用的目的地。

   對於 MQIPT SSL 路徑，請調整 **SSLClientConnectTimeout** 路徑內容，以確保無法使用目的地的提示連線失敗。 如需 IBM MQ 調整參數的詳細資料，請參閱 IBM MQ 文件。 此外，如需作業系統的 TCP/IP 調整詳細資料，請參閱作業系統說明文件。 無論如何，失敗的連線嘗試應該會快速傳回網路失敗 (例如，TCP 重設封包) ，或應該在沒有過度延遲的情況下逾時。
2. 必須立即切斷失效系統的作用中連線，才能建立新的連線。

   當連線主動使用 MQIPT 時，您也應該考量失效接手的影響。 在失效接手期間，可能會中斷網路連線。 對於用戶端應用程式，您可以使用 IBM MQ 自動用戶端重新連線功能來重新建立已中斷的連線。 對於訊息通道，您可以指定短重試間隔，以便通道立即重新連接。 如需自動用戶端重新連線及訊息通道重試配置的相關資訊，請參閱 IBM MQ 文件。

V 9.3.5 IBM MQ Console 和 REST API

您可以使用 IBM MQ Console 和 REST API 來管理 IBM MQ，並使用 HTTP 來執行傳訊作業。

- 您可以使用 IBM MQ Console ，從 Web 瀏覽器執行基本管理作業。 如需相關資訊，請參閱 使用 IBM MQ Console 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。 如需相關資訊，請參閱 使用 REST API 管理。

- 您可以使用 messaging REST API 來執行簡式點對點和發佈傳訊。 如需相關資訊，請參閱 使用 REST API 進行傳訊。

安裝選項

IBM MQ Console 和 REST API 在稱為 mqweb 的 WebSphere Liberty 伺服器中執行。 從 IBM MQ 9.3.5，您可以將 mqweb 伺服器安裝為 IBM MQ 安裝中的選用元件，或安裝為獨立式 IBM MQ Web Server 安裝。

Linux V 9.3.5 獨立式 IBM MQ Web Server 安裝

從 IBM MQ 9.3.5 開始， mqweb 伺服器可以在 IBM MQ Web Server 的獨立式安裝中執行。 獨立式 IBM MQ Web Server 安裝可讓您在與 IBM MQ 安裝分開的系統上安裝並執行 mqweb 伺服器。 安裝獨立式 IBM MQ Web Server 可讓您更靈活地選擇在哪些系統上執行 mqweb 伺服器，以及選擇在哪些系統上執行 mqweb 伺服器的系統數目。 必要的話，可以在不同機器上執行 mqweb 伺服器的數個實例，以提供您需要的可調整性及可用性。

如果您已購買 IBM MQ 授權，則可以安裝所需數量的獨立式 IBM MQ Web Server 副本。 IBM MQ Web Server 安裝不計入您購買的 IBM MQ 授權。 如需 IBM MQ 授權的相關資訊，請參閱 IBM MQ 授權資訊。

下列限制適用於獨立式 IBM MQ Web Server 安裝架構:

- 「IBM MQ Console」只能用來管理遠端佇列管理程式。
- 「messaging REST API」只能與遠端佇列管理程式搭配使用。
- administrative REST API 無法使用。

獨立式 IBM MQ Web Server 僅在 Linux 平台上受支援。

如需安裝獨立式 IBM MQ Web Server 的相關資訊，請參閱 安裝獨立式 IBM MQ Web Server。

IBM MQ 安裝架構的選用元件

您可以選擇在 IBM MQ 安裝過程中安裝 IBM MQ Console 和 REST API 元件。

當 mqweb 伺服器在 IBM MQ 安裝中執行時，所有 IBM MQ Console 和 REST API 特性都可用。

- 「IBM MQ Console」可用來管理本端及遠端佇列管理程式。
- 「messaging REST API」可以與本端及遠端佇列管理程式搭配使用。
- 「administrative REST API」可用來管理本端及遠端佇列管理程式。

若要使用 IBM MQ Console 和 REST API 元件，請在 IBM MQ 安裝過程中安裝下列元件:

- AIX 在 AIX 上，安裝 `mqm.web.rte` 檔案集。
- IBM i 在 IBM i 上，安裝 WEB 元件。
- Linux 在 Linux 上，安裝 `MQSeriesWeb` 元件。
- Windows 在 Windows 上，安裝 `Web Administration` 特性。
- z/OS 在 z/OS 上，安裝 `IBM MQ for z/OS UNIX System Services Web Components` 特性。

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。

在其他國家或地區中，IBM 可能未提供本文件所提及的各項產品、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。 本文件在提及 IBM 產品、程式或服務時，不表示或暗示只能使用 IBM 產品、程式或服務。 只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。 不過，任何非 IBM 之產品、程式或服務，使用者必須自行負責作業之評估和驗證責任。

本文件所說明之主題內容，IBM 可能擁有其專利或專利申請案。 提供本文件不代表提供這些專利的授權。 您可以書面提出授權查詢，來函請寄到：

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

如果是有關雙位元組 (DBCS) 資訊的授權查詢，請洽詢所在國的 IBM 智慧財產部門，或書面提出授權查詢，來函請寄到：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

下列段落不適用於英國，若與任何其他國家之法律條款抵觸，亦不適用於該國： International Business Machines Corporation 只依 "現況" 提供本出版品，不提供任何明示或默示之保證，其中包括且不限於不侵權、可商用性或特定目的之適用性的隱含保證。 有些地區在特定交易上，不允許排除明示或暗示的保證，因此，這項聲明不一定適合您。

本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。 同時，IBM 得隨時改進並（或）變動本書中所提及的產品及（或）程式。

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，在其他作業環境下取得的結果可能大不相同。 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。

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 未來動向的任何陳述，僅代表 IBM 的目標而已，並可能於未事先聲明的情況下有所變動或撤回。

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。 所有這些名稱都是虛構的，如有任何類似實際企業所用的名稱及地址之處，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、服務性或功能。

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。 其他產品及服務名稱可能是 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: