9.3

IBM MQ の技術概要

注記

本書および本書で紹介する製品をご使用になる前に、315 ページの『特記事項』に記載されている情報をお読みください。

本書は、 IBM® MQ バージョン 9 リリース 3、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様が IBM に情報を送信する場合、お客様は IBM に対し、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で情報を使用または配布する非独占的な権利を付与します。

目次

IBM MQ の技術概要

IBM MQ を使用して、アプリケーションに接続し、組織全体での情報の分散を管理します。

IBM MQ により、複数のプログラムは、一貫性のあるアプリケーション・プログラミング・インターフェースを使用して、異なるコンポーネント (プロセッサー、オペレーティング・システム、サブシステム、および通信プロトコル) のネットワーク間で相互に通信することができます。 このインターフェースを使用して設計および作成されるアプリケーションは、メッセージ・キューイング・アプリケーションとして知られています。

以下のサブトピックを使用して、IBM MQ で提供されるメッセージ・キューイングおよび他の機能についての情報を得ることができます。

関連概念

IBM MQ の概要

製品の要件とサポート情報を確認できる場所

関連タスク

IBM MQ アーキテクチャーの計画

関連資料

7 ページの『メッセージ・キューイングの主な機能とメリット』

ここでは、メッセージ・キューイングの主な機能とメリットを中心に説明します。 メッセージ・キューイングのセキュリティーやデータ保全性などについて説明します。

メッセージ・キューイングの概要

IBM MQ 製品では、整合性のあるアプリケーション・プログラミング・インターフェースを使用して、性質の異なるコンポーネント (プロセッサー、オペレーティング・システム、サブシステム、および通信プロトコル) のネットワーク内でプログラムが相互に通信できるようにしています。

メッセージング とキューイング のスタイルをとるため、このインターフェースを使用して設計、作成されるアプリケーションをメッセージ・キューイング・アプリケーションといいます。

- メッセージングでは、プログラムは、相互に直接呼び出す代わりに、メッセージでデータを送信し合うことにより通信します。
- キューイングでは、メッセージがストレージ内のキューに配置されるので、複数のプログラムが互いに独立して、異なるスピードで、異なる時刻に、異なる場所で、プログラム間の論理接続をもたずに稼働できるようになります。

メッセージ・キューイングはデータ処理の分野で長年用いられてきました。 現在では、電子メールの分野で広く使われています。 キューイングがない場合、電子メッセージをリモート側で送信するには、経路上のすべてのノードがメッセージを転送でき、受信局をログオンさせて、メッセージを送信しようとしていることを常に認識させることが必要です。 キューイング・システムでは、メッセージはシステムがそれを転送できるようになるまで中間ノードに格納されます。 最終の宛先では、受信局が読み取り可能になるまで、メッセージは電子メールボックスに格納されます。

しかし、今日では多くの複雑な業務トランザクションがキューイングを使用しないで処理されています。 大規模ネットワークでは、システムが膨大な数の接続を使用可能な状態にしていなければなりません。 システムの一部で問題が発生すると、システムの多くの部分が使用できなくなります。

メッセージ・キューイングは、プログラム用の電子メールと考えることができます。 メッセージ・キューイング環境では、アプリケーション・スイートの各部分を構成するそれぞれのプログラムが、特定の要求に応答する形で、明確に定義された自己完結型の機能を実行します。 他のプログラムと通信するには、プログラムは事前定義キューにメッセージを書き込む必要があります。 他のプログラムは、そのメッセージをキューから取り出し、そこに入っている要求や情報を処理します。 このため、メッセージ・キューイングはプログラム間通信の 1 つのスタイルと言えます。

キューイングとは、アプリケーションがメッセージの処理が行えるようになるまでメッセージを保留する機能のことです。 キューイングによって次のことが可能になります。

- 通信コードを記述することなく、プログラム間で通信を行うことができる (これらのプログラムはそれぞれ異なった環境で実行中でも構わない)。
- プログラムがメッセージを処理する順序を選択することができる。
- メッセージの数がしきい値を超えるときは、複数のプログラムが 1 つのキューをサービスするようにして、システム上の負荷のバランスをとることができる。
- 基本システムが使用不可のときは、代替システムがキューをサービスするようにして、アプリケーションの使用可能度を向上させることができる。

メッセージ・キューとは

メッセージ・キューは、単にキューと呼ばれており、メッセージを送信することができる名前付きの宛先です。 メッセージは、それらのキューを取り扱うプログラムによって取り出されるまで、キューに蓄積されます。

キューは、キュー・マネージャー内にあり、キュー・マネージャーによって管理されます (9 ページの『メッセージ・キューイングの用語』を参照)。 キューの物理的な特性は、キュー・マネージャーが実行されているオペレーティング・システムに依存します。 キューはコンピューターのストレージ内の揮発性バッファー域にあっても、あるいはディスクなどの永続記憶装置上のデータ・セットにあっても構いません。 キューの物理的な管理は、キュー・マネージャーの役目であり、関連するアプリケーション・プログラムには明らかにされません。

プログラムは、キュー・マネージャーの外部サービスを通じてのみキューにアクセスします。 外部サービスは、キューのオープン、キューへのメッセージの書き込み、キューからのメッセージの読み取り、およびキューのクローズを行うことができます。 また、キューの属性を設定したり、照会したりすることもできます。

さまざまなスタイルのメッセージ・キューイング

Point-to-Point

1 つのメッセージがキューに入れられ、1 つのアプリケーションがそのメッセージを受け取ります。

Point-to-Point メッセージングでは、送信側アプリケーションが受信側アプリケーションにメッセージを送信する前に、送信側アプリケーションが受信側アプリケーションについての情報を認識している必要があります。 例えば、送信側アプリケーションは、情報を送信するキューの名前を認識していて、キュー・マネージャー名を指定しなければならない場合があります。

パブリッシュ/サブスクライブ

パブリッシング・アプリケーションによってパブリッシュされた各メッセージのコピーは、各インタレスト・アプリケーションに送信されます。 インタレスト・アプリケーションは多数ある場合も、1 つある場合も、まったくない場合もあります。 パブリッシュ/サブスクライブでは、インタレスト・アプリケーションはサブスクライバーと呼ばれ、メッセージはサブスクリプションによって識別されるキューに入れられます。

パブリッシュ/サブスクライブ・メッセージングを使用すると、情報の提供者とその情報の利用者 を分離することができます。 送信側アプリケーションと受信側のアプリケーションは、情報を送受信するために、お互いについて何も知る必要がありません。 詳しくは、63 ページの『パブリッシュ/サブスクライブ・メッセージング』を参照してください。

アプリケーションの設計担当者および開発者にとってのメッセージ･キューイングの利点

IBM MQ により、アプリケーション・プログラムは、メッセージ・キューイング を使用してメッセージ・ドリブン処理に加わることができます。 適切なメッセージ・キューイング・ソフトウェア製品を使用すれば、プラットフォームが異なっていても、アプリケーション・プログラム相互間で通信することができます。 例えば、z/OS® アプリケーションは IBM MQ for z/OS を介して通信できます。 アプリケーションは、基礎となる通信機構からは保護されています。 その他に、メッセージ・キューイングには次のような利点があります。

- 多くのアプリケーション間で共用できる小規模のプログラム群を使用して、アプリケーションを設計できる。

- これらの構築ブロックを再使用することによって、新しいアプリケーションを速やかに構築できる。
- メッセージ・キューイング技法を用いるため作成されたアプリケーションは、キュー・マネージャーの作業の仕方が変更されても影響を受けない。
- 通信プロトコルを使用する必要がない。 キュー・マネージャーが通信のすべての局面を処理します。
- メッセージを受信するプログラムは、メッセージが送信されてくる時に実行中である必要はない。 これらのメッセージはキューに保存されます。

設計担当者はアプリケーションのコストを下げることができます。なぜなら、開発の速度が速くなり、開発担当者の必要人数が少なくてすみ、しかもメッセージ・キューイングを使わないアプリケーションの場合よりもプログラミング・スキルの要求レベルが低いからです。

IBM MQ は、 メッセージ・キュー・インターフェース (または MQI) と呼ばれる共通アプリケーション・プログラミング・インターフェースを、アプリケーションがどこで実行されていても実装します。 このため、アプリケーション・プログラムを、あるプラットフォームから 別のプラットフォームに移植するのが容易になります。

MQI の詳細については、Message Queue Interface の概要を参照してください。

メッセージ・キューイングの主な機能とメリット

ここでは、メッセージ・キューイングの主な機能とメリットを中心に説明します。 メッセージ・キューイングのセキュリティーやデータ保全性などについて説明します。

メッセージ・キューイング方法を用いるアプリケーションの主な機能は、次のとおりです。

- 7 ページの『プログラム間に直接の接続がない』
- 8 ページの『時間に依存しない通信』
- 8 ページの『小規模のプログラム』
- 8 ページの『メッセージ・ドリブン型の処理』
- 9 ページの『イベント主導型の処理』
- 9 ページの『メッセージ優先順位』
- 9 ページの『セキュリティー』
- 9 ページの『データ整合性』
- 9 ページの『リカバリー・サポート』

注 : IBM MQ クライアントおよびサーバーについて検討する際に、新しいプラットフォームの追加の IBM MQ MQI clients をサポートするためにサーバー・アプリケーションを変更する必要はありません。 同様に、IBM MQ MQI client は、変更しなくても、追加タイプのサーバーで機能することができます。

プログラム間に直接の接続がない

メッセージ・キューイングは、間接的なプログラム間通信の技法です。 これは、プログラムが相互に通信し合うアプリケーション内で用いることができます。 1 つのプログラムがメッセージを (キュー・マネージャーが所有する) キューに書き込んで、次に別のプログラムがそのメッセージをキューから読み取ることによって通信が行われます。

プログラムは、他のプログラムによってキューに書き込まれたメッセージを読み取ることができます。 他のプログラムは、受信プログラムとして同じキュー・マネージャーに接続することも、または別のキュー・マネージャーに接続することもできます。 この別のキュー・マネージャーは別のシステム、異なったコンピューター・システム、またはたとえ異なった企業にあっても構いません。

メッセージ・キューを用いて通信するプログラム間には、物理的な接続はありません。 一方のプログラムが、キュー・マネージャーが所有するキューにメッセージを送信し、もう一方のプログラムがキューからメッセージを取り出します (8 ページの図 1 を参照)。

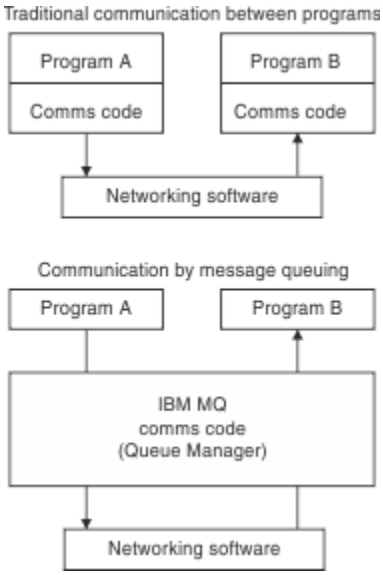


図 *1.* 従来の通信方式と比較したメッセージ・キューイング

電子メールの場合と同様に、トランザクションの一部である個々のメッセージは、蓄積交換方式に基づいてネットワーク内を移動します。基盤。 ノード間のリンクに障害が起こった場合は、そのリンクが復旧するか、オペレーターまたはプログラムがそのメッセージの宛先を変更するまで、メッセージは保留されます。

メッセージがキューからキューへ移動する機構はプログラムには表れません。 そのため、プログラムはより単純化されます。

時間に依存しない通信

他のプログラムに作業を要求したプログラムは、要求に対する応答を待つ必要はありません。 これらのプログラムは他の作業を行うことができ、応答の処理をそれが到着したとき、または後で行うことができます。 メッセージング・アプリケーションを作成するときは、プログラムがいつメッセージを送信するか、または相手がいつメッセージを受信できるかを知る (あるいは関与する) 必要はありません。 メッセージが失われることはありません。メッセージは、相手がそれを処理できるようになるまで、キュー・マネージャーによって保存されます。 メッセージは、プログラムが削除するまでキューにとどまっています。 これは、送信と受信のアプリケーション・プログラムは分離されていることを意味します。送信側は、メッセージの受信に関する受信側の応答を待たずに処理を継続できます。 ターゲット・アプリケーションが、メッセージの送信時に、実行中である必要もありません。 ターゲット・アプリケーションは、始動後にメッセージを取り出すことができます。

小規模のプログラム

メッセージ・キューイングには、小規模の自己完結型のプログラムを使用できるという利点があります。単一の大規模なプログラムに作業のすべての部分を順次に実行させる代わりに、その作業をいくつかの小規模で独立したプログラムに分散させることができます。 要求側プログラムは、個別の各プログラムにメッセージを送信して、それぞれの機能を実行するよう要求します。各プログラムは、機能を完了すると、1つまたは複数のメッセージとしてその結果を戻します。

メッセージ・ドリブン型の処理

トリガー操作 と呼ばれるメカニズムを使用して、メッセージがキューに到着したときに自動的にアプリケーションを開始できます。 必要に応じて、メッセージの処理が完了したときに、アプリケーションを停止させることもできます。

イベント主導型の処理

プログラムを、キューの状態に応じて制御できます。 例えば、メッセージがキューに到着した直後にプログラムを開始するようにしたり、またはある優先順位より上あるいはあらゆる優先順位のメッセージが、例えば 10 個キューに入るまではプログラムを開始しないようにすることもできます。

メッセージ優先順位

プログラムは、メッセージをキューに書き込むときに優先順位を割り当てることができます。 これにより、新しいメッセージが追加されるキュー内の位置が決まります。

プログラムがキューからメッセージを取得する方法は、キューに入っているメッセージの順番に取得するか、特定のメッセージを取得するかのいずれかです。 (プログラムが以前に送信した要求に対する応答を探している場合は、特定のメッセージを取得する必要が生じることがあります。)

セキュリティー

キュー・マネージャーを使用する際のアプリケーションの認証、キュー・マネージャー上のキューなどのリソースを使用する際の許可検査、ネットワークを介して転送されるメッセージ・データおよびキュー上に存在するメッセージ・データの暗号化、といったセキュリティー機能が用意されています。 セキュリティーの詳細については、セキュリティーの概要を参照してください。

データ整合性

データ保全性は、作業単位によって提供されます。 各 MQGET や MQPUT では、作業単位の開始と終了の同期は、オプションとして完全にサポートされており、作業単位の結果をコミットしたり、ロールバックしたりすることができます。 同期点のサポートは、そのアプリケーションに選択された同期点調整の形式によって、IBM MQ の外側または内側のどちらかで動作します。

リカバリー・サポート

リカバリーを可能にするため、IBM MQ の持続的な更新はすべてログに記録されます。 リカバリーが必要な場合、すべてのパーシスタント・メッセージがリストアされ、すべての未完了トランザクションがロールバックされ、すべての同期点コミットおよびバックアウトは制御中の同期点マネージャーが通常ハンドルする方法で処理されます。 持続メッセージの詳細については、メッセージの持続性を参照してください。

メッセージ・キューイングの用語

ここでは、メッセージ・キューイングで使われている一部の用語について説明します。

次の方法があります。

- チャネル
- クラスター
- IBM MQ MQI client
- z/OS グループ内キューイング
- メッセージ
- メッセージ・チャネル・エージェント
- メッセージ記述子
- Point-to-Point
- publish/subscribe
- キュー
- キュー・マネージャー
- z/OS キュー共用グループ
- z/OS 共用キュー

- サブスクリプション
- トピック

チャネル

あるキュー・マネージャーから別のキュー・マネージャーにメッセージを移すためにチャネルが使用され、それによってアプリケーションは基礎をなす通信プロトコルから遮蔽されます。 キュー・マネージャーは、同一のシステム上に存在する場合もあれば、同一のプラットフォーム上の異なるシステムや、異なるプラットフォーム上に存在する場合もあります。 メッセージの送信元は次のように多岐にわたります。

- あるノードから別のノードにデータを転送するユーザー作成のアプリケーション・プログラム。
- PCF コマンドまたは MQAI を使用するユーザー作成の管理アプリケーション。
- IBM MQ Explorer。
- インスツルメンテーション・イベント・メッセージを別のキュー・マネージャーに送信するキュー・マネージャー。
- リモート管理コマンドを別のキュー・マネージャーに送信するキュー・マネージャー。 例えば、MQSC コマンドや administrative REST API を使用します。

チャネルについて詳しくは、32 ページの『チャネル定義』を参照してください。

クラスター

クラスター とは、何らかの方法で論理的に関連付けられているキュー・マネージャーのネットワークです。

クラスター化せずに分散キューイングを使用する IBM MQ のネットワークでは、キュー・マネージャーはすべて独立しています。 あるキュー・マネージャーから別のキュー・マネージャーにメッセージを送信する必要がある場合、送信元のキュー・マネージャーには、伝送キューとリモート・キュー・マネージャーへのチャネルが定義されていなければなりません。

クラスターの使用には、システム管理の単純化、および可用性とワークロードの平衡化の向上という 2 つの異なる目的があります。

設定したクラスターが最小のものであっても、システム管理の単純化は実現されます。 1 つのクラスターを構成しているキュー・マネージャーは多くの定義を必要としないので、間違った定義を作成する状況は少なくなります。

クラスタリングの詳細については、クラスターを参照してください。

IBM MQ MQI client

IBM MQ MQI クライアントは、独立してインストール可能な IBM MQ のコンポーネントです。 MQI クライアントを使用することにより、通信プロトコルを使って IBM MQ アプリケーションを実行し、他のプラットフォーム上の 1 つ以上の Message Queue Interface (MQI) サーバーと対話して、それらのキュー・マネージャーに接続することができます。

IBM MQ MQI client コンポーネントをインストールして使用する方法の詳細については、以下のトピックを参照してください。

- AIX AIX®での IBM MQ クライアントのインストール
- Linux Linux®での IBM MQ クライアントのインストール
- Windows Windows での IBM MQ クライアントのインストール
- IBM i IBM i での IBM MQ クライアントのインストール

サーバーとクライアント間の接続の構成

グループ内キューイング

z/OS

キュー共用グループ中のキュー・マネージャーは、通常のチャネルを使って通信できます。または、グループ内キューイング (IGQ) と呼ばれる、チャネルを定義しなくても高速でメッセージ転送を実行できる技法を使うことができます。 これは、IBM MQ for z/OS のみに適用されます。

グループ内キューイングについて詳しくは、221 ページの『グループ内キューイング』を参照してください。

メッセージ

メッセージ・キューイングでのメッセージとは、1 つのプログラムから別のプログラムに向けて送られるデータの集合です。 IBM MQ メッセージ を参照してください。

メッセージ・タイプの詳細については、メッセージのタイプを参照してください。

MSG チャネル・エージェント

メッセージ・チャネル・エージェントは、チャネルの一端です。 1 組のメッセージ・チャネル・エージェント (送信エージェントと受信エージェント) によって 1 つのチャネルが構成され、キュー・マネージャーから別のキュー・マネージャーにメッセージが移動されます。

メッセージ・チャネル・エージェントの使用方法については、分散キュー管理の概要を参照してください。

メッセージ記述子

IBM MQ メッセージは制御情報とアプリケーション・データから構成されています。

制御情報は、メッセージ記述子構造体 (MQMD) で定義され、次のような項目で構成されています。

- メッセージのタイプ
- メッセージの ID
- メッセージ送達の優先順位

アプリケーション・データの構造と内容は、IBM MQ ではなく、関連するプログラムによって決定されます。

詳しくは、 MQMD を参照してください。

Point-to-Point メッセージング

Point-to-Point メッセージングでは、各メッセージは、これを生成する 1 つのアプリケーションから、これをコンシュームする 1 つのアプリケーションへ移動します。 メッセージは、生成側のアプリケーションがメッセージをキューに入れることで転送され、コンシューム側のアプリケーションはそのキューからメッセージを取得します。

パブリッシュ/サブスクライブ・メッセージング

パブリッシュ/サブスクライブ・メッセージングでは、パブリッシング・アプリケーションによってパブリッシュされた各メッセージのコピーは、各インタレスト・アプリケーションに送信されます。 インタレスト・アプリケーションは多数ある場合も、1 つある場合も、まったくない場合もあります。 パブリッシュ/サブスクライブでは、インタレスト・アプリケーションはサブスクライバーと呼ばれ、メッセージはサブスクリプションによって識別されるキューに入れられます。

詳細については、63 ページの『パブリッシュ/サブスクライブ・メッセージング』を参照してください。

キュー

メッセージの送信先になる指定された宛先です。 メッセージは、それらのキューを取り扱うプログラムによって取り出されるまで、キューに蓄積されます。

詳細については、19 ページの『キュー』を参照してください。

キュー・マネージャー

キュー・マネージャー は、アプリケーションに対してキューイング・サービスを提供するシステム・プログラムです。

これは、アプリケーション・プログラミング・インターフェースを提供し、それによってプログラムはメッセージのキューへの書き込みおよびキューからのメッセージの読み取りができます。 キュー・マネージャーは管理者が新しいキューを生成したり、既存のキューの属性を変更したり、キュー・マネージャーの操作を制御したりできる付加的な機能も提供します。

IBM MQ メッセージ・キューイング・サービスをシステムで利用する場合、キュー・マネージャーを実行する必要があります。 1 つのシステムに複数のキュー・マネージャーを稼働させることができます (例えば、テスト・システムと実働 システムを区別するため)。 各キュー・マネージャーは、アプリケーションに対して、接続ハンドル (*Hconn*) で識別される。

種々のアプリケーションがキュー・マネージャーのサービスを同時に使用でき、しかもこれらのアプリケーションは互いにまったく無関係でも構いません。 プログラムがキュー・マネージャーのサービスを利用するためには、そのキュー・マネージャーとの接続を確立する必要があります。

アプリケーションが他のキュー・マネージャーに接続しているアプリケーションにメッセージを送信するためには、そのキュー・マネージャーはこれらのキュー・マネージャー間で通信できなければなりません。 IBM MQ は、蓄積交換 プロトコルを実装することにより、このようなアプリケーションの間で安全にメッセージが送達されるようにしています。

詳細については、28 ページの『キュー・マネージャー』を参照してください。

キュー共用グループ

共用キューの同じセットにアクセスできるキュー・マネージャーは、キュー共用グループ (QSG) というグループを形成します。 これらは、共用キューを格納するカップリング・ファシリティー (CF) を使って互いに通信を行います。 これは、IBM MQ for z/OS のみに適用されます。

詳細については、175 ページの『共用キューとキュー共用グループ』を参照してください。

共用キュー

共用キューとはローカル・キューのタイプで、シスプレックス内の 1 つ以上のキュー・マネージャーによってメッセージにアクセスすることができます。 これは、同一のキュー・マネージャーを使用して複数のアプリケーションによって共用されるキューとは異なります。 これは、IBM MQ for z/OS のみに適用されます。

サブスクリプション

パブリッシュ/サブスクライブ・アプリケーションは、特定のトピックに関して、インタレストをメッセージに登録することができます。 これを行ったアプリケーションはサブスクライバーと呼ばれ、マッチング・メッセージが処理のためにキューに入れられる方法を、用語サブスクリプションが定義します。

サブスクリプションには、サブスクライバーの ID、パブリケーションを配置する宛先キューの ID についての情報が含まれます。 また、パブリケーションを宛先キューに配置する方法についての情報も含まれます。

詳細については、66 ページの『サブスクライバーとサブスクリプション』を参照してください。

トピック

トピックは、パブリッシュ/サブスクライブ・メッセージでパブリッシュされる情報の主題を示す文字ストリングです。

トピックは、パブリッシュ/サブスクライブ・システムでメッセージを正常に送達するうえで、重要な役割を果たします。 パブリッシャーは、各メッセージに特定の宛先アドレスを含める代わりに、各メッセージにトピックを割り当てます。 キュー・マネージャーは、トピックと、そのトピックをサブスクライブして

いるサブスクライバーのリストとを突き合わせ、それらのサブスクライバーの各々にメッセージを送達します。

詳細については、69 ページの『トピック』を参照してください。

メッセージとキュー

メッセージとキューは、メッセージ・キューイング・システムの基本構成要素です。

メッセージとは

メッセージ を使用するアプリケーションにとって、メッセージは意味のあるバイト・ストリングです。 メッセージは、ある 1 つのアプリケーションから別のアプリケーションに (または、同じアプリケーションの異なる部分の間で) 情報を転送するために使用されます。 伝達に関係するアプリケーションは、 同じプラットフォーム上で実行されていても、別のプラットフォーム上で実行されていてもかまいません。

IBM MQ メッセージの構成要素を次に示します。

- アプリケーション・データ。アプリケーション・データの内容と構造は、そのデータを使用するアプリケーション・プログラムによって定義されます。
- メッセージ記述子。メッセージ・ディスクリプターは、メッセージを識別し、 メッセージのタイプおよび送信側アプリケーションによってメッセージに割り当てられた優先順位などの 追加の制御情報を含んでいます。

  メッセージ記述子の形式は、IBM MQ によって定義されます。 メッセージ記述子の詳細な説明については、MQMD - メッセージ記述子を参照してください。
- メッセージ・プロパティー。 メッセージに関するメタデータです。 メッセージ・プロパティーの内容は、それらを使用するアプリケーション・プログラムが定義します。 詳細については、メッセージ・プロパティーを参照してください。

メッセージ長

デフォルトの最大メッセージ長は 4 MB ですが、この最大長を 100 MB まで増やすことができます (1 MB は 1 048 576 バイトです)。 ただし、実際には、次のものによってメッセージ長は制限されます。

- 受信側のキュー用に定義した最大メッセージ長
- キュー・マネージャー用に定義した最大メッセージ長
- キューによって定義された最大メッセージ長
- 送信側のアプリケーションまたは受信側のアプリケーションで定義した 最大メッセージ長
- メッセージ用として使用可能なストレージの量

1 つのアプリケーションが必要とするすべての情報を送るには、複数のメッセージが必要な場合があります。

アプリケーションによるキューの送信/受信方法

アプリケーションでは、**MQI 呼び出し**を使用してメッセージを送受信します。

例えば、キューにメッセージを書き込むために、アプリケーションでは以下の処理が行われます。

1. MQOPEN 呼び出しを発行して必要なキューをオープンする。
2. MQPUT 呼び出しを発行してキューにメッセージを書き込む。

別のアプリケーションがそのキューからメッセージを取り出すときには、MQI MQGET 呼び出しを発行します。

MQI 呼び出しについての詳細は、MQI 呼び出しを参照してください。

キューとは

キュー とは、メッセージを保管するためのデータ構造体です。

キューはすべて、キュー・マネージャー に属しています。 キュー・マネージャーは、所有するキューを管理し、受信したすべてのメッセージを適切なキューに格納します。 メッセージは、アプリケーション・プログラムまたはキュー・マネージャーによって、通常の操作の一部としてキューに書き込まれます。

事前定義キューと動的キュー

キューは、その作成方法によって次のような特徴があります。

- **事前定義キュー**は、管理者が該当する MQSC または PCF コマンドを使用して作成します。事前定義キューは、永続キューであって、それらを使用するアプリケーションとは無関係に存在し、IBM MQ が再始動しても存続します。
- **動的キュー**が作成されるのは、アプリケーションが モデル・キュー の名前を指定して MQOPEN 要求を出した場合です。 作成されたキューは、モデル・キューというテンプレート・キュー定義 に基づきます。 MQSC コマンド DEFINE QMODEL を使用してモデル・キューを作成することができます。 モデル・キューの属性 (例えば、キューに保管できるメッセージの最大数) は、そのモデル・キューから作成される動的キューが継承します。

  モデル・キューは、作成される動的キューが永続キューになるか一時キュー になるかを指定する属性を持っています。 永続キューは、アプリケーションやキュー・マネージャーが 再始動しても存続しますが、一時キューは、再始動すると失われます。

キューからのメッセージの取り出し

MQSeries では、アプリケーションに適切な許可が与えられている場合、次の取り出しアルゴリズ ムに従ってキューからメッセージを取り出すことができます。

- 先入れ先出し法 (first-in-first-out (FIFO))。
- メッセージ記述子に定義されたメッセージ優先順位。 同じ優先順位を持つメッセージは、FIFO 順に取り出されます。
- 特定のメッセージについてのプログラム要求。

アプリケーションからの MQGET 要求によって、使用される方式が決まります。

IBM MQ オブジェクト

キュー・マネージャーは、IBM MQ オブジェクトのプロパティーを定義します。 それらのプロパティーの値は、IBM MQ がこれらのオブジェクトを処理する方法に影響します。 IBM MQ のコマンドとインターフェースを使用して、オブジェクトを作成したり管理したりできます。 アプリケーションから、Message Queue Interface (MQI) を使用してオブジェクトを制御します。 プログラムからアドレッシングされるとき、オブジェクトは IBM MQ オブジェクト記述子 (MQOD) によって識別されます。

オブジェクト管理

オブジェクトの管理には、以下のタスクが含まれます。

- キュー・マネージャーの始動および停止
- アプリケーション用のオブジェクト (特にキュー) の作成
- オブジェクトの属性の表示または変更
- オブジェクトの削除
- チャネルを使用して、他の (リモート) システムにあるキュー・マネージャーへの通信パスを作成。
- キュー・マネージャーのクラスター を作成することによって、管理プロセス全体を簡易化し、ワークロードのバランスをとる。

動的キューの場合を除き、オブジェクト (動的キューを除く) は、処理の前にキュー・マネージャーに定義されていなければなりません。

IBM MQ コマンドを使用してオブジェクト管理操作を行う場合、キュー・マネージャーは、ユーザーがそれらの操作を実行するのに必要なレベルの権限を持っているかどうかを検査します。 同様に、アプリケーションが MQOPEN 呼び出しを使用してオブジェクトをオープンするとき、キュー・マネージャーは、そのオブジェクトへのアクセスを許可する前に、アプリケーションが必要なレベルの権限を持っているかどうかを検査します。 検査は、オープンされているオブジェクトの名前に対して行われます。

以下のメソッドを使用してオブジェクトを定義したり管理したりできます。

- プログラマブル・コマンド・フォーマット・リファレンスおよび管理タスクの自動化で説明されている PCF コマンド
- MQSC コマンドで説明されている MQSC コマンド
- z/OS IBM MQ for z/OS 操作および制御パネル (IBM MQ for z/OS の管理 を参照)
- Windows Linux IBM MQ Explorer (Windows および Linux システム専用の Intel)。 詳しくは、MQ エクスプローラーの概要を参照してください。

以下のメソッドを使用してオブジェクトを管理することもできます。

- キーボードから入力する制御コマンド。 制御コマンドを使用した IBM MQ for Multiplatforms の管理を参照してください。
- プログラムからの IBM MQ 管理インターフェース (MQAI) の呼び出し。 IBM MQ 管理インターフェース (MQAI) を参照してください。

ALW AIX, Linux, and Windows 上の IBM MQ コマンドのシーケンスについては、MQSC 機能を使用して、ファイルに保持されている一連のコマンドを実行できます。 詳しくは、 MQSC コマンドを使用した IBM MQ の管理を参照してください。

IBM i 定期的に使用する IBM MQ for IBM i コマンドのシーケンスについては、制御言語プログラムを作成することができます。 詳しくは、CL コマンドを使用した IBM MQ for IBM i の管理を参照してください。

z/OS 通常使用する一連の IBM MQ for z/OS コマンドの場合、コマンドの入ったメッセージを作成して、システム・コマンドの入力キューにこれらのメッセージを書き込む管理プログラムを作成することもできます。 キュー・マネージャーは、このキュー上のメッセージを、コマンド・ラインまたは操作および制御パネルから入力されたコマンドを処理するのと同じ方法で処理します。 この手法は、IBM MQ 管理のためのプログラムの作成に説明されており、IBM MQ for z/OS と共に提供されている Mail Manager サンプル・アプリケーションで示されています。 このサンプルについては、IBM MQ for z/OS 用のサンプル・プログラムに説明があります。

オブジェクトの属性

オブジェクトのプロパティーは、その属性によって定義されます。 属性の一部はユーザーが指定できますが、他の属性は表示のみ可能です。

例えば、キューが収容できる最大メッセージ長は、**MaxMsgLength** 属性によって定義されます。この属性は、キューの作成時に指定できます。 **DefinitionType** 属性は、キューが作成された方法を指定します。この属性は表示のみできます。

IBM MQ では、属性を参照する方法としては、次の 2 とおりの方法があります。

- 属性の PCF 名 (例えば、**MaxMsgLength**) を使用 する方法
- 属性の MQSC コマンド名 (例えば、MAXMSGL) を使用する方法

キュー共有グループ

z/OS

共用キューの同じセットにアクセスできるキュー・マネージャーは、キュー共用グループ (QSG) と呼ばれるグループを形成し、共用キューを保管するカップリング・ファシリティー (CF) を使用して互いに通信します。 QSG は厳密にはオブジェクトではないことに注意してください。

共用キューとは、キュー共用グループ内にある 1 つ以上のキュー・マネージャーがアクセスできるメッセージを持つ、一種のローカル・キューです。 これは、同一のキュー・マネージャーを使用して複数のアプリケーションによって共用されるキューとは異なります。

キュー共用グループには、最大 4 文字の名前があります。 この名前はネットワーク内で固有であり、かつ、キュー・マネージャー名とは異なるものである必要があります。

重要 : 共用キューおよびキュー共用グループは、IBM MQ for z/OS でのみサポートされます。

詳しくは、 175 ページの『共用キューとキュー共用グループ』 を参照してください。

システム・デフォルト・オブジェクト

システム・デフォルト・オブジェクト とは、キュー・マネージャーの作成時に各キュー・マネージャーごとに自動的に作成される 1 組のオブジェクト定義のことです。 ご使用のシステムのアプリケーションで使用するために、これらのオブジェクト定義はすべてコピーしたり、修正することができます。

デフォルト・オブジェクト名には、語幹である SYSTEM が付いています。例えば、デフォルト・ローカル・キューは SYSTEM.DEFAULT.LOCAL.QUEUE であり、デフォルト受信チャネルは SYSTEM.DEF.RECEIVER です。 これらのオブジェクトの名前を変更することはできません。これらの名前を 持つデフォルト・オブジェクトは必須です。

オブジェクトを定義する際に、明示的に指定されなかった属性は該当するデフォルト・オブジェクトからコピーされます。 例えば、ローカル・キューを定義する場合、指定しなかった属性は、デフォルト・キュー SYSTEM.DEFAULT.LOCAL.QUEUE から取られます。

詳しくは、 システムおよびデフォルト・オブジェクト を参照してください。

オブジェクト・タイプ

管理タスクの多くには、さまざまな種類の IBM MQ オブジェクト の操作が関係します。

IBM MQ オブジェクトの命名については、37 ページの『IBM MQ オブジェクトの命名』を参照してください。

キュー・マネージャー上に作成されるデフォルトのオブジェクトについては、16 ページの『システム・デフォルト・オブジェクト』を参照してください。

さまざまなタイプの IBM MQ オブジェクトについては、以下を参照してください。

認証情報オブジェクト

認証情報オブジェクトは、証明書取り消し検査を実行するために必要な定義を提供します。

キュー・マネージャー認証情報オブジェクトは、Transport Layer Security (TLS) の IBM MQ サポートの一部を構成しています。 このオブジェクトには、取り消された証明書の検査に必要な定義があります。 認証局は、信頼できなくなった証明書を取り消します。

認証情報オブジェクトを定義するには、MQSC コマンド **DEFINE AUTHINFO** を使用できます。 認証情報オブジェクトの属性について詳しくは、**DEFINE AUTHINFO** を参照してください。

認証情報オブジェクトでは、以下の IBM MQ 制御コマンドを使用できます。

- **setmqaut** (権限の付与または取り消し)
- **dspmqaut** (オブジェクト権限の表示)
- **dmpmqaut** (権限のダンプ)
- **rcrmqobj** (オブジェクトの再作成)
- **rcdmqimg** (メディア・イメージの記録)
- **dspmqfls** (ファイル名の表示)

TLS の概要、および認証情報オブジェクトの使用については、 Transport Layer Security (TLS) の概念 および IBM MQ での TLS セキュリティー・プロトコルを参照してください。

チャネル

チャネルとは、あるキュー・マネージャーと別のキュー・マネージャーを結ぶ通信パスを提供するオブジェクトのことです。

詳しくは、30 ページの『チャネル』 を参照してください。

通信情報オブジェクト

IBM MQ Multicast は、待ち時間が短く、ファンアウトが大きい、高信頼性マルチキャスト・メッセージングです。 マルチキャスト送信を使用するには、通信情報 (COMMINFO) オブジェクトが必要です。

詳しくは、111 ページの『IBM MQ マルチキャスト』 を参照してください。

COMMINFO オブジェクトは、マルチキャスト伝送に関連付けられた属性が含まれる IBM MQ オブジェクトです。 これらの属性の詳細については、DEFINE COMMINFO を参照してください。 COMMINFO オブジェクトの作成について詳しくは、マルチキャストの概要を参照してください。

リスナー

リスナー は、他のキュー・マネージャーまたはクライアント・アプリケーションからネットワーク要求を受け取るプロセスで、関連付けられたチャネルを始動します。

リスナー・プロセス は、 **runmqlsr** 制御コマンドを使用して開始できます。

リスナー・オブジェクト は、キュー・マネージャーの有効範囲内からリスナー・プロセスの開始と停止を管理することができるようにする、IBM MQ オブジェクトです。 リスナー・オブジェクトの属性を定義することにより、次の操作を実行できます。

- リスナー・プロセスを構成する。
- キュー・マネージャーが開始および停止したときに、リスナー・プロセスも自動的に開始および停止させるかどうかを指定する。

重要 : z/OS リスナー・オブジェクトは、IBM MQ for z/OS ではサポートされていません。 チャネル・イニシエーターを使用して、IBM MQ for z/OS が listen 機能を実装する方法について詳しくは、171 ページの『z/OS 上のチャネル・イニシエーター』 を参照してください。

名前リスト

名前リスト は、クラスター名、キュー名、または認証情報オブジェクト名のリストが含まれた IBM MQ オブジェクトです。 クラスターでは、このリストを使用して、キュー・マネージャーがリポジトリーを保持しているクラスターのリストを識別することができます。

名前リストは、IBM MQ オブジェクトの 1 つで、他の IBM MQ オブジェクトのリストが格納されています。 通常、名前リストはトリガー・モニターなどのアプリケーションによって、キュー・グループの識別に使用されます。 名前リストを使用した場合の利点は、アプリケーションとは別個に管理できるということです。更新時に、その名前リストを使用しているアプリケーションを停止する必要はありません。 また、あるアプリケーションで障害が起こった場合でも、名前リストには影響はなく、他のアプリケーションは引き続きその名前リストを使用できます。

名前リストは、複数の IBM MQ オブジェクトによって参照されるクラスターのリストを維持するために、キュー・マネージャー・クラスターでも使用されます。

MQSC コマンド DEFINE NAMELIST および ALTER NAMELIST を使用して、名前リストを定義および変更することができます。

注 : z/OS あるいは、 z/OS では、 IBM MQ for z/OS 操作および制御パネルを使用することもできます。

プログラムは MQI を使用することにより、これらの名前リストにどのキューが登録されているかを検知できます。 名前リストの編成は、アプリケーション設計担当者およびシステム管理者が行います。

使用可能な名前リスト属性のリストについては、 名前リストの属性を参照してください。

プロセス定義

プロセス定義オブジェクトを使用すると、キュー・マネージャーが使用するアプリケーションの属性を定義することによって、オペレーターによる介入がなくてもアプリケーションを開始することができます。

プロセス定義オブジェクトは、IBM MQ キュー・マネージャーでのトリガー・イベントに応答して開始されるアプリケーションを定義します。 プロセス定義の属性には、アプリケーション ID、アプリケーション・タイプ、およびアプリケーション特有のデータがあります。 詳しくは、27 ページの『IBM MQ によって特定の目的で使用されるキュー』の『開始キュー』を参照してください。

アプリケーションをオペレーターの介入なしで開始させる (これについては、トリガーによる IBM MQ アプリケーションの開始を参照) には、アプリケーションの属性をキュー・マネージャーに通知する必要があります。 これらの属性はプロセス定義オブジェクト で定義されます。

ProcessName 属性はオブジェクトの作成時に固定されます。 ただし、他の属性は、IBM MQ コマンドを使用して変更できます。

注 : z/OS あるいは、 z/OS では、 IBM MQ for z/OS 操作パネルおよび制御パネルを使用することもできます。

すべての属性の値について問い合わせるには、MQINQ - オブジェクト属性の照会を使用します。

使用可能なプロセス定義属性のリストについては、 プロセス定義の属性を参照してください。

キュー

IBM MQ キュー は名前付きオブジェクトで、アプリケーションはそこにメッセージを書き込んだり、そこからメッセージを読み取ったりできます。

詳しくは、 19 ページの『キュー』 を参照してください。

キュー・マネージャー

IBM MQ キュー・マネージャーは、アプリケーションにキューイング・サービスを提供し、そのキューを管理します。

詳しくは、 28 ページの『キュー・マネージャー』 を参照してください。

サービス

サービス・オブジェクトは、キュー・マネージャーが開始または停止したときに実行するプログラムを定義するための方法です。

プログラムは次のいずれかのタイプになります。

サーバー
サーバー は、SERVTYPE パラメーターが SERVER に指定されているサービス・オブジェクトです。 サーバー・サービス・オブジェクトは、指定したキュー・マネージャーの開始時に実行されるプログラムの定義です。 サーバー・プロセスの 1 つのインスタンスだけを並行して実行できます。 実行中は、サーバー・プロセスの状況を MQSC コマンド DISPLAY SVSTATUS を使用してモニターできます。 通常、サーバー・サービス・オブジェクトは、送達不能ハンドラーまたはトリガー・モニターなどのプログラムの定義ですが、実行可能なプログラムは IBM MQ によって提供されるプログラムに限定されません。 また、サーバー・サービス・オブジェクトを定義して、指定されたキュー・マネージャーがシャットダウンしてプログラムが終了したときに実行されるコマンドを含めることができます。

コマンド
コマンド は、SERVTYPE パラメーターが COMMAND に指定されているサービス・オブジェクトです。 コマンド・サービス・オブジェクトは、指定されたキュー・マネージャーが開始または停止したときに実行されるプログラムの定義です。 コマンド・プロセスの複数のインスタンスを並行して実行できます。 コマンド・サービス・オブジェクトは、プログラムが実行されるとキュー・マネージャーがプログラムをモニターしなくなる点で、サーバー・サービス・オブジェクトと異なっています。 通常、コマンド・サービス・オブジェクトは存続期間の短いプログラムの定義で、1 つ以上の他のタスクを開始するなどの特定のタスクを実行します。

重要 : z/OS サービス・オブジェクトは、IBM MQ for z/OS ではサポートされていません。

詳しくは、「サービスの操作」を参照してください。

ストレージ・クラス

z/OS

記憶域クラスは、1 つ以上のキューをページ・セットにマップします。

つまり、そのキューのメッセージがそのページ・セットに (バッファリングを条件として) 保管されます。

記憶域クラスは、IBM MQ for z/OS でのみサポートされています。

ストレージ・クラスについて詳しくは、 z/OS での IBM MQ 環境の計画を参照してください。

トピック・オブジェクト

トピック・オブジェクト は、特定の非デフォルト属性をトピックに割り当てることができるようにする IBM MQ オブジェクトです。

トピック は、特定のトピック・ストリング にパブリッシュまたはサブスクライブするアプリケーションによって定義されます。 トピック・ストリングでは、スラッシュ記号 (/) でトピックを分離することにより、トピックの階層を指定できます。階層はトピック・ツリーで視覚化されます。 例えば、アプリケーションがトピック・ストリングである `/Sport/American Football` および `/Sport/Soccer` にパブリッシュする場合、作成されるトピック・ツリーは、`Sport` という親ノードと、`American Football` および `Soccer` という 2 つの子ノードで構成されます。

各トピックは、トピック・ツリー内にある最初の親管理ノードから属性を継承します。 特定のトピック・ツリーに管理トピック・ノードが存在しない場合、全トピックは基本トピック・オブジェクトである SYSTEM.BASE.TOPIC から属性を継承します。

トピック・ツリー内の任意のノードでトピック・オブジェクトを作成できます。これは、トピック・オブジェクトの TOPICSTR 属性でそのノードのトピック・ストリングを指定することにより可能となります。また、管理トピック・ノードの他の属性も定義することができます。 これらの属性について詳しくは、『MQSC コマンド』または『PCF コマンドによる管理の自動化』を参照してください。 デフォルトでは、各トピック・オブジェクトは直近の親管理トピック・ノードから属性を継承します。

さらに、トピック・オブジェクトを使用することにより、アプリケーション開発者からトピック・ツリー全体を隠すことも可能です。 トピック `/Sport/American Football` に対して FOOTBALL.US という名前のトピック・オブジェクトが作成された場合、アプリケーションは、同じ結果を持つストリング `/Sport/American Football` ではなく、 FOOTBALL.US という名前のオブジェクトにパブリッシュまたはサブスクライブすることができます。

トピック・オブジェクトのトピック・ストリング中に文字 #、+、/、または * を 入力する場合、その文字はストリング中では通常文字として扱われ、 トピック・オブジェクトが関連付けられているトピック・ストリングの一部であると 見なされます。

トピック・オブジェクトについて詳しくは、63 ページの『パブリッシュ/サブスクライブ・メッセージング』を参照してください。

関連概念

5 ページの『メッセージ・キューイングの概要』
IBM MQ 製品では、整合性のあるアプリケーション・プログラミング・インターフェースを使用して、性質の異なるコンポーネント (プロセッサー、オペレーティング・システム、サブシステム、および通信プロトコル) のネットワーク内でプログラムが相互に通信できるようにしています。

関連資料

MQSC コマンド

キュー

IBM MQ のキューとキュー属性の紹介。

メッセージはキューに入れられるので、メッセージを書き込んだアプリケーションは、そのメッセージに対する応答を期待する場合でも、応答を待っている間に他の作業を自由に行うことができます。 アプリケーションは Message Queue Interface (MQI) を用いてキューにアクセスします。MQI については、Message Queue Interface の概要で説明します。

メッセージをキューに書き込むには、その前にキューが作成されている必要があります。 キューはキュー・マネージャーによって所有され、そのキュー・マネージャーは多数のキューを所有できます。 ただし、各キューはそのキュー・マネージャー内で固有の名前を持っていなければなりません。

キューはキュー・マネージャーを通じて保守されます。 ほとんどの場合、各キューはそのキュー・マネージャーによって物理的に管理されますが、このことはアプリケーション・プログラムからは認識されません。 IBM MQ for z/OS 共用キューは、キュー共用グループ内の任意のキュー・マネージャーによって管理できます。

キューを作成するには、IBM MQ コマンド (MQSC)、PCF コマンド、またはプラットフォーム固有のインターフェースを使用できます。 例えば、IBM MQ for z/OS 操作および制御パネルはプラットフォーム固有のものです。

一時的なジョブ用のローカル・キューを独自のアプリケーションから動的に 作成できます。 例えば、応答先 キュー (アプリケーションが終了したら必要なくなる) を作成できます。 詳しくは、25 ページの『動的キューとモデル・キュー』 を参照してください。

キューを使用する前に、そのキューで何を行いたいかを指定して、キューをオープンする必要があります。例えば、以下の目的でキューをオープンすることができます。

- メッセージのブラウズのみ (取り出しは行わない)
- メッセージの取り出し (他のプログラムとの共用アクセスか、または排他的アクセスで)
- キューへのメッセージの書き込み
- キューの属性の照会
- キューの属性の設定

キューを開くときに指定できるオプションの完全なリストについては、MQOPEN - オブジェクトのオープンを参照してください。

キューの属性

キューの属性には、そのキューが定義される時に指定され、後からは変更できないものがあります (例えば、キューのタイプ)。 キューのその他の属性は、次のいずれかの方法で変更可能な属性にグループ分けできます。

- キューの処理中にキュー・マネージャーによって (例えば、キューの現在のサイズ)
- コマンドによってのみ (例えば、キューのテキスト記述)
- アプリケーションが MQSET 呼び出しを使用して (例えば、そのキューに書き込み操作ができるかどうか)

MQINQ 呼び出しを用いてすべての属性の値を知ることができます。

複数のキュー・タイプに共通な属性として次のものがあります。

QName
キューの名前。

QType
キューのタイプ。

QDesc
キューのテキスト記述。

InhibitGet
プログラムがキューからのメッセージの取得を許可されているかどうか。 ただし、リモート・キューからメッセージを取得することはできません。

InhibitPut
プログラムがそのキューにメッセージを書き込むことができるかどうか。

DefPriority
キューに書き込まれたメッセージのデフォルト優先順位。

DefPersistence
キューに書き込まれたメッセージのデフォルト持続性

範囲
このキューに対するエントリーが名前サービスでも存在するか制御します。

z/OS **Scope** 属性は、z/OS ではサポートされません。

これらの属性の詳細な説明については、キューの属性を参照してください。

キューの定義方法

IBM MQ にキューを定義するには、MQSC の DEFINE コマンドまたは PCF の Create Queue コマンドを使用します。 これらのコマンドは、キューのタイプおよびキューの属性を指定します。 例えば、ローカル・キュー・オブジェクトは、アプリケーションがその キューを MQI 呼び出しで参照したときに何が発生するかを指定する属性を 持っています。 属性には、次のものがあります。

- アプリケーションがメッセージをキューから取り出せるかどうか (読み取り (GET) 可能)
- アプリケーションがメッセージをキューに書き込めるかどうか (書き込み (PUT) 可能)
- キューへのアクセスが、1 つのアプリケーション専用になるか、または複数のアプリケーションで共用されるか
- 同時にキューに保管できるメッセージの最大数 (キューの最大サイズ)
- キューに書き込むことのできる最大メッセージ長

また、キューの定義に使用できるプラットフォーム固有の各種インターフェースもあります。

関連概念
59 ページの『クラスター・キュー』
クラスター・キューとは、クラスター・キュー・マネージャーでホストされ、同じクラスター内の別のキュー・マネージャーで使用できるキューです。

50 ページの『送達不能キュー』
送達不能キュー (または未配布メッセージ・キュー) は、正しい宛先にメッセージを送信できない場合にそのメッセージが送信されるキューです。 一般的に、送達不能キューはキュー・マネージャーごとに 1 つ存在します。

PCF コマンドによる管理の自動化
IBM MQ Console でのキューの処理

関連タスク
MQSC コマンドを使用した IBM MQ の管理
MQ Explorer を使用したキュー・マネージャーおよびオブジェクトの作成と構成
IBM i CL コマンドを使用した IBM MQ for IBM i の管理
z/OS IBM MQ for z/OS で MQSC コマンドおよび PCF コマンドを発行できるソース

関連資料
59 ページの『共用キューとクラスター・キューの比較』
ここでは、共用キューおよびクラスター・キューを比較し、ご使用のシステムにどちらが適切かを判断できるようにすることを目的としています。

関連情報
175 ページの『共用キューとは』

ローカル・キュー

ローカル・キューのタイプは、伝送、開始、送達不能、コマンド、デフォルト、チャネル、およびイベント・キューです。

キューは、プログラムが接続されているキュー・マネージャーに所有される場合、プログラムではローカル と認知されます。 メッセージは、ローカル・キューから取得し、ローカル・キューに入れることができます。

キュー定義オブジェクトは、キューに入る物理メッセージと同様、そのキューの定義情報を保持します。

各キュー・マネージャーは、ある特別な目的のために用いられる次のようなローカル・キューを持つことができます。

伝送キュー

アプリケーションがメッセージをリモート・キューに送信するとき、ローカル・キュー・マネージャーはそのメッセージを、伝送キュー と呼ばれる特別のローカル・キューに保管します。 アプリケーションは、メッセージを伝送キューに直接書き込むことも、リモート・キュー定義を介して間接的に書き込むこともできます。

キュー・マネージャーは、メッセージをリモート・キュー・マネージャーに送信するときに、次の順序で伝送キューを決定します。

1. リモート・キューのローカル定義の XMITQ 属性に名前が指定されている伝送キュー。
2. リモート・キュー・マネージャーと同じ名前の伝送キュー。 この値は、リモート・キューのローカル定義の XMITQ のデフォルト値です。
3. ローカル・キュー・マネージャーの DEFXMITQ 属性に名前が指定されている伝送キュー。

メッセージ・チャネル・エージェントは、伝送キューと関連付けられたチャネル・プログラムで、次の宛先にメッセージを送達します。 次の宛先とは、メッセージ・チャネルが接続されるキュー・マネージャーです。 それはメッセージの最終宛先と同じキュー・マネージャーである必要はありません。 メッセージは、次の宛先に送達されると伝送キューから削除されます。 メッセージは、最終宛先までの経路上でいくつものキュー・マネージャーを通過しなければならない場合があります。 経路上の各キュー・マネージャーには、伝送キューを定義する必要があります。各伝送キューは、次の宛先に伝送されるのを待っているメッセージを保持します。 通常の伝送キューは、メッセージの最終的な宛先が異なる場合でも、次の宛先へのメッセージを保持します。 クラスター伝送キューは、複数の宛先へのメッセージを保持します。 各メッセージの `correlID` は、次の宛先に転送するためにメッセージが入れられるチャネルを識別します。

キュー・マネージャーには、複数の伝送キューを定義できます。 同じ宛先に対して、それぞれが異なるサービスのクラスに用いられる伝送キューをいくつか定義する場合もあります。 例えば、同じ宛先に送る小規模メッセージと大規模メッセージに異なる伝送キューを作成したい場合があります。 そうすれば異なるメッセージ・チャネルを使用してメッセージを転送できるため、大規模メッセージが小規模メッセージを遅らせてしまうことがありません。 デフォルトでは、クラスター・キューまたはクラスター・トピックへのメッセージはすべて、単一のクラスター伝送キュー `SYSTEM.CLUSTER.TRANSMIT.QUEUE` に入れられます。 オプションとして、デフォルトを変更し、異なるクラスター・キュー・マネージャーから異なるクラスター伝送キューに行くようメッセージ・トラフィックを分離することができます。 キュー・マネージャー属性 `DEFCLXQ` を `CHANNEL` に設定すると、各クラスター送信側チャネルが別々のクラスター伝送キューを作成します。 代わりに、クラスター送信側チャネルに使用するクラスター伝送キューを手動で定義することもできます。

伝送キューは、メッセージ・チャネル・エージェントを起動して、メッセージを転送することができます。トリガーによる IBM MQ アプリケーションの開始を参照してください。

z/OS IBM MQ for z/OS では、グループ内キューイングを使用している場合は、伝送キューはグループ内キューイング・エージェントによりサービスされます。 共用伝送キューは、IBM MQ for z/OS でグループ内キューイングを使用する際に使われます。

開始キュー

開始キュー は、トリガー・イベントがアプリケーション・キューで起きたときに、キュー・マネージャーがトリガー・メッセージを書き込むローカル・キューです。

トリガー・イベントとは、プログラムにキューの処理を開始させることを目的とするイベントのことです。 例えば、到着メッセージが 10 件を超えるというイベントなどがあります。 トリガー操作の詳細については、トリガーによる IBM MQ アプリケーションの開始を参照してください。

送達不能 (未配布メッセージ) キュー

送達不能 (未配布メッセージ) キュー は、キュー・マネージャーが送達不能なメッセージを書き込むローカル・キューです。

キュー・マネージャーは、メッセージを送達不能キューに書き込むときに、メッセージにヘッダーを追加します。 ヘッダー情報には、キュー・マネージャーがそのメッセージを送達不能キューに書き込んだ理由が入っています。 また、元のメッセージの宛先、キュー・マネージャーがそのメッセージを送達不能キューに書き込んだ日時なども入っています。

また、アプリケーションも送達不能のメッセージ用にキューを使用できます。 詳細については、送達不能 (未配布メッセージ) キューの使用を参照してください。

システム・コマンド・キュー

システム・コマンド・キュー は、適正な許可を持つアプリケーションが IBM MQ コマンドを送信できるキューです。 これらのキューはプラットフォームでサポートされている PCF、MQSC および CL コマンドを受信し、キュー・マネージャーがそれらを処理できるようにします。

z/OS IBM MQ for z/OS では、キューは SYSTEM.COMMAND.INPUT と呼ばれます。 他のプラットフォームでは、 SYSTEM.ADMIN.COMMAND.QUEUE と呼ばれます。 受け入れられるコマンドはプラットフォームによって変わります。 詳細については、プログラマブル・コマンド・フォーマット・リファレンスを参照してください。

システム・デフォルト・キュー

システム・デフォルト・キュー には、ご使用のシステム用のキューの初期定義が格納されています。 キュー定義を作成すると、キュー・マネージャーは該当のシステム・デフォルト・キューからその定義をコピーします。 キュー定義の作成は、動的キューの作成とは異なります。 動的キューの定義は、動的キューのテンプレートとして選択したモデル・キューに基づきます。

イベント・キュー

イベント・キュー はイベント・メッセージを保持します。 これらのメッセージは、キュー・マネージャーまたはチャネルによって報告されます。

リモート・キュー

プログラムが接続されているキュー・マネージャーとは別のキュー・マネージャーが所有するキューは、そのプログラムにとってリモート です。

通信リンクが確立されている場合、プログラムはリモート・キューにメッセージを送信できます。 ただし、プログラムはリモート・キューからメッセージを読み取ることはできません。

キュー定義オブジェクトはリモート・キューを定義する際に作成され、ローカル・キュー・マネージャーがメッセージの送信先のキューを見つけるために必要な情報だけを保持します。 このオブジェクトは、リモート・キューのローカル定義 と呼ばれます。 リモート・キューの属性は、すべてそれを所有するキュー・マネージャーによって保持されますが、これは、リモート・キューが、そのキュー・マネージャーにとってはローカル・キューになるためです。

リモート・キューをオープンするときは、キューを識別するために次のどちらかを指定しなければなりません。

- リモート・キューを定義するローカル定義の名前。 アプリケーションの視点からは、これはローカル・キューのオープンと同じです。 アプリケーションは、キューがローカルかリモートかを認識する必要はありません。

  IBM i 以外のすべてのプラットフォームでリモート・キューのローカル定義を作成するには、 DEFINE QREMOTE コマンドを使用します。

IBM i では、CRTMQMQ コマンドを使用します。

- リモート・キュー・マネージャーの名前と、そのリモート・キュー・マネージャーに認識されているキューの名前。

20 ページの『キューの属性』で説明した共通属性のほかに、リモート・キューのローカル定義には 3 つの属性があります。 以下の 3 つの属性が該当します。

RemoteQName
キューを所有するキュー・マネージャーが、そのキューを識別する名前。

RemoteQMgrName
所有キュー・マネージャーの名前。

XmitQName
メッセージを他のキュー・マネージャーに転送するときに使用するローカル伝送キューの名前。

これらの属性の詳細については、キューの属性を参照してください。

リモート・キューのローカル定義に対し MQINQ 呼び出しを使用する場合、キュー・マネージャーが戻すのは、リモート・キュー名、リモート・キュー・マネージャー名、および伝送キュー名のローカル定義の属性だけで、リモート・システム内で一致するローカル・キューの属性は戻しません。

伝送キューも参照してください。

別名キュー

別名キュー とは、別のキューまたはトピックにアクセスするために使用できる IBM MQ オブジェクトです。これは、複数のプログラムがその同じキューを別の名前でアクセスして、作業できることを意味しています。

別名を解決した結果として得られるキュー (基本キューと呼ばれる) には、 プラットフォームでサポートされる、以下のいずれかの種類のキューを指定できます。

- ローカル・キュー
- リモート・キューのローカル定義。
- 共用キュー。これは、IBM MQ for z/OS でのみ使用可能なローカル・キューのタイプです。
- 事前定義キュー
- 動的キュー

別名を解決した結果がトピックになることもあります。 現在はキューにメッセージを 書き込むアプリケーションがある場合、キューの名前をトピックの別名にすれば、そのアプリケーションはトピックへのパブリッシュを行う ようになります。 アプリケーション・コードを変更する 必要はありません。

注 : 別名を同一キュー・マネージャー上の別の別名に直接解決することはできません。

別名キューの使用例は、システム管理者が、基本キューの名前 (つまり、別名が解決された結果のキュー名) と別名キューの名前に異なったアクセス許可を与える場合です。 これは、プログラムまたはユーザーが、基本キューではなく、別名キューを使用するように許可されることを意味します。

または、別名の書き込み操作は禁止し、基本キューの書き込み操作は可能になるように許可を設定できます。

あるアプリケーションでは、別名キューの使用により、システム管理者がアプリケーションを変更しなくても、別名キュー・オブジェクトの定義を容易に変更できるようになります。

プログラムが別名を使用しようとすると、IBM MQ は、別名に対する許可検査を行います。 ただし、プログラムに、その別名が解決された結果の名前にアクセスする許可が与えられているかどうかは検査しません。 したがって、プログラムには別名キューの名前にアクセスする許可は与えられますが、解決された結果のキュー名にアクセスする許可は与えられません。

19 ページの『キュー』に記述されている一般的なキュー属性のほかに、別名キューは **BaseQName** 属性を持っています。 これは、別名が解決されたときの基本キューの名前です。 この属性の詳細については、BaseQName (MQCHAR48) を参照してください。

別名キューの *InhibitGet* 属性および **InhibitPut** 属性 (19 ページの『キュー』を参照) は、別名に属します。 例えば、別名キュー名 ALIAS1 が基本キュー名 BASE に解決される場合は、ALIAS1 に対する禁止事項は ALIAS1 だけを対象にし、BASE は禁止されません。 ただし、BASE に対する禁止事項は ALIAS1 にも影響を及ぼします。

DefPriority 属性および **DefPersistence** 属性も別名に属します。 したがって、例えば、同じ基本キューの異なった別名に異なるデフォルト優先順位を割り当てることができます。 さらに、別名を使用するアプリケーションを変更しなくても、これらの優先順位を変えることもできます。

動的キューとモデル・キュー

この情報は、動的キュー、一時および永続の動的キューのプロパティー、動的キューの使用方法、動的キューを使用する際の考慮事項、およびモデル・キューについて詳しく説明するものです。

アプリケーション・プログラムがモデル・キューをオープンするために MQOPEN 呼び出しを出すと、キュー・マネージャーは動的に、そのモデル・キューと同じ属性をもつローカル・キューのインスタンスを作成します。 モデル・キューの *DefinitionType* フィールドの値に応じて、キュー・マネージャーは一時または永続の動的キューを作成します (動的キューの作成を参照)。

一時動的キューの特性

一時動的キュー は次のような特性をもっています。

- z/OS キュー共用グループ内のキュー・マネージャーがアクセスできる共用キューではない。

  キュー共用グループが使用可能なのは、IBM MQ for z/OS のみです。
- 非持続メッセージだけを保持する。
- リカバリー不能である。
- キュー・マネージャーの始動時に削除される。
- キューを作成した MQOPEN 呼び出しを発行したアプリケーションがキューをクローズしたときか、そのアプリケーションが終了したときに、削除される。
  - コミットされたメッセージがキューにある場合には削除される。
  - この時に、キューに対して未解決で、コミットされない MQGET、MQPUT、または MQPUT1 呼び出しがある場合は、キューは論理的に削除されたものとしてマーク付けされる。さらに、このキューは、(これらの呼び出しがコミットされたあとに) クローズ処理の一部として、またはアプリケーションが終了した時に、物理的に削除される。
  - この時 (作成中のアプリケーションまたは他のアプリケーションが) キューを使用している場合、そのキューは物理的に削除されたものとしてマーク付けされ、それを使用する最後のアプリケーションによってクローズされたときに、物理的に削除される。
  - 論理的に削除されたキューに (クローズ以外の目的で) アクセスしようとすると、MQRC_Q_DELETED の理由コードで失敗する。
  - MQCO_NONE、MQCO_DELETE、および MQCO_DELETE_PURGE は、キューを作成した MQOPEN 呼び出しに対応する MQCLOSE 呼び出しで指定すると、すべて MQCO_NONE として処理される。

永続動的キューの特性

永続動的キュー は、次のような特性を持っています。

- 持続メッセージまたは非持続メッセージを保持する。
- システム障害が生じた場合にリカバリー可能である。
- アプリケーション (必ずしも、キューを作成した MQOPEN 呼び出しを発行したアプリケーションでなくてもよい) が、MQCO_DELETE または MQCO_DELETE_PURGE オプションを使用してキューを正常にクローズしたときに、削除される。
  - MQCO_DELETE オプションを指定したクローズ要求は、キュー上にメッセージ (コミットされた、またはコミットされない) がまだ残っている場合は、失敗する。 MQCO_DELETE_PURGE オプションを指定したクローズ要求は、コミットされたメッセージがキュー上にある場合でも、正常に実行される (メッセージはクローズの一部として削除される)。ただし、キューに対して未解決で、コミットされない MQGET、MQPUT、または MQPUT1 呼び出しがある場合は、失敗する。

  - 削除要求が成功しても (作成中の、または他のアプリケーションによって)、キューが使用中の場合は、そのキューは、論理的に削除されたものとしてマーク付けされ、それを使用する最後のアプリケーションによってクローズされた時に、物理的に削除される。
- キューを削除する権限のないアプリケーションがキューをクローズした場合は、キューをクローズしたアプリケーションがキューを作成した MQOPEN 呼び出しを発行しない限り、キューは削除されません。 許可検査は、対応する MQOPEN 呼び出しの妥当性検査に使用されたユーザー ID (MQOO_ALTERNATE_USER_AUTHORITY が指定されていた場合は、代替ユーザー ID) に対して行われる。
- 通常のキューと同様に削除できる。

動的キューの使用方法

次のような場合に動的キューを使用できます。

- 処理を終了した後、キューを保持する必要がないアプリケーション
- メッセージに対する応答を別のアプリケーションに処理させるアプリケーション。 このようなアプリケーションは、モデル・キューをオープンして応答先キューを動的に作成できます。 例えば、クライアント・アプリケーションは次のようにすることができます。

  1. 動的キューを作成する。
  2. 要求メッセージのメッセージ記述子構造体の **ReplyToQ** フィールドにその名前を指定する。
  3. サーバーによって処理されるキューにその要求を入れる。

これで、サーバーは応答メッセージを応答先キューに入れることができるようになります。 最後に、クライアントはその応答を処理し、削除オプションを使用して応答先キューをクローズできます。

動的キューを使用する際の考慮事項

動的キューを使用するときは次の点を考慮してください。

- クライアント/サーバー・モデルでは、各クライアントが各自の動的応答先キューを作成して使用しなければならない。 動的応答先キューが複数のクライアントで共用されている場合は、そのキューに対して未解決でコミットされない活動があったり、そのキューが他のクライアントによって使用されていたりして、応答先キューの削除が遅れることがあります。 さらに、キューは論理的に削除されたというマークが付けられたために、後続の (MQCLOSE 以外の) API 要求でアクセス不能になることもあります。
- 使用しているアプリケーションの環境により、動的キューをアプリケーション間で共用しなければならない場合は、必ず、そのキューに対するすべての活動がコミットされてから (削除オプションで)、そのキューがクローズされるようにする。 これは、最後のユーザーが行う必要があります。 これによって、キューの削除が遅れないようになり、論理的に削除されたものとしてマーク付けされたためにキューがアクセス不能になっている期間が最小限になります。

モデル・キュー

モデル・キュー はキュー定義のテンプレートで、動的キューを作成する際に使用します。

IBM MQ プログラムからローカル・キューを動的に作成し、キュー属性のテンプレートとして使用したいモデル・キューを命名できます。 この時点で、新規のキューのいくつかの属性を変更することができます。ただし、**DefinitionType** を変更することはできません。 例えば、永続キューが必要な場合には、定義タイプが永続的に設定されたモデル・キューを選択してください。 ある会話型のアプリケーションでは、動的キューを使用して照会に対する応答を入れておくことができます。なぜなら、多くの場合、応答を処理し終わったらそれらのキューを保持する必要がなくなるからです。

MQOPEN 呼び出しのオブジェクト記述子 (MQOD) に、モデル・キューの名前を指定します。 キュー・マネージャーは、指定されたモデル・キューの属性を使用して、ローカル・キューを動的に作成します。

動的キューの名前を (完全名で) 指定するか、名前の語幹 (例えば、ABC) を指定して、キュー・マネージャーが固有の部分をこれに追加するようにできます。または、キュー・マネージャーがユーザーの代わりに完全な固有名を割り当てるようにすることもできます。 キュー・マネージャーが名前を割り当てる場合、それを MQOD 構造体に書き込みます。

モデル・キューに直接 MQPUT1 呼び出しを発行することはできませんが、モデル・キューをオープンしたときに作成された動的キューに対して MQPUT1 を発行することは可能です。

モデル・キューに対して MQSET や MQINQ を発行することはできません。 MQOO_INQUIRE か MQOO_SET を使用してモデル・キューを開くと、それ以降の MQINQ 呼び出しや MQSET 呼び出しは、動的に作成されたキューに対して行われます。

モデル・キューの属性は、ローカル・キューの属性のサブセットです。 詳細については、キューの属性を参照してください。

IBM MQ によって特定の目的で使用されるキュー

IBM MQ は、一部のローカル・キューを、その操作に関連する特定の目的のために使用します。

これらのキューは、IBM MQ によって使用される前に、必ず定義する必要があります。

開始キュー

開始キュー は、トリガー操作に使用するキューです。 キュー・マネージャーは、トリガー・イベントが発生すると、開始キュー にトリガー・メッセージを書き込みます。 トリガー・イベントとは、キュー・マネージャーによって検出される条件の論理的 組み合わせのことです。 例えば、キュー上のメッセージの数が、あらかじめ定義されたキューのサイズに達したときにトリガー・イベントが生成される場合があります。 このイベントが発生すると、キュー・マネージャーは指定の開始キューに トリガー・メッセージを入れることになります。 このトリガー・メッセージは、開始キューをモニターする特殊アプリケーションであるトリガー・モニター によって取り出されます。 次に、トリガー・モニターは、トリガー・メッセージに指定されているアプリケーション・プログラムを開始します。

キュー・マネージャーでこのトリガー操作を使用する場合は、少なくとも 1 つの 開始キューを、そのキュー・マネージャー用に定義する必要があります。 トリガー操作のためのオブジェクトの管理、runmqtrm、およびトリガーによる IBM MQ アプリケーションの開始を参照してください。

伝送キュー

伝送キューは、リモート・キュー・マネージャー宛てのメッセージを一時的に保管するキューです。 ローカル・キュー・マネージャーがメッセージを直接送信する各リモート・ キュー・マネージャーごとに、少なくとも 1 つの伝送キューを定義する必要が あります。 これらのキューは、リモート管理にも使用されます (ローカル・キュー・マネージャーからのリモート管理を参照)。 分散キューイングでの伝送キューの使用については、 IBM MQ 分散キューイング技法を参照してください。

各キュー・マネージャーには、1 つのデフォルト伝送キューがあります。 クラスター外のキュー・マネージャーがリモート・キューにメッセージを書き込む場合、デフォルトでは、デフォルト伝送キューが使用されます。 宛先のキュー・マネージャーと同名の伝送キューがあれば、その伝送キューにメッセージが入れられます。 キュー・マネージャーの別名定義で、**RQMNAME** パラメーターが宛先キュー・マネージャーと一致し、**XMITQ** パラメーターが指定されている場合は、**XMITQ** により指定された伝送キューにメッセージが入れられます。 **XMITQ** パラメーターがない場合は、メッセージで指定されたローカル・キューにメッセージが入れられます。

クラスター伝送キュー

クラスター内のそれぞれのキュー・マネージャーには、SYSTEM.CLUSTER.TRANSMIT.QUEUE というクラスター伝送キューと、SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE というモデル・クラスター伝送キューがあります。 これらのキューの定義は、キュー・マネージャーを定義するときにデフォルトとして作成されます。 キュー・マネージャーの属性、**DEFCLXQ** が CHANNEL に設定された場合、作成された各クラスター送信側チャネルに、永続動的クラスター伝送キューが自動的に作成されます。 キューは SYSTEM.CLUSTER.TRANSMIT. *ChannelName* と呼ばれます。 クラスター伝送キューを手動で定義することもできます。

クラスター内のキュー・マネージャーは、同じクラスター内の他の キュー・マネージャーに、これらのキュー上のメッセージを送ります。

名前の解決時には、クラスター伝送キューがデフォルト伝送キューより優先され、特定のクラスター伝送キューが SYSTEM.CLUSTER.TRANSMIT.QUEUE より優先されます。

送達不能キュー
送達不能 (未配布メッセージ) キューは、正しい宛先に渡すことができないメッセージを保管します。 例えば、宛先キューが満杯である場合には、メッセージは転送されません。 システムに提供された送達不能キューは、SYSTEM.DEAD.LETTER.QUEUE と呼ばれます。

分散キューイングでは、関係する各キュー・マネージャーごとに 1 つの送達不能キューを定義してください。

コマンド・キュー
コマンド・キュー SYSTEM.ADMIN.COMMAND.QUEUE は、適切な許可を与えられたアプリケーションが処理対象の MQSC コマンドを送信する先のローカル・キューです。 次に、これらのコマンドは、コマンド・サーバーと呼ばれる IBM MQ コンポーネントによって取り出されます。 コマンド・サーバーは、それらのコマンドを検査し、正しいものをキュー・マネージャーの処理用に渡し、該当する応答先キューに応答を戻します。

キュー・マネージャーを作成すると、各キュー・マネージャーごとにコマンド ・キューが自動的に作成されます。

応答先キュー
あるアプリケーションが要求メッセージを送信した場合、そのメッセージを 受信するアプリケーションは、送信側のアプリケーションに応答メッセージを 戻すことができます。 この応答メッセージは、応答先キューと呼ばれるキューに書き込まれます。このキューは、通常は送信側のアプリケーションのローカル・キューです。 応答先キューの名前は、送信側のアプリケーションによってメッセージ記述子の 一部として指定されます。

イベント・キュー
観測イベントは、MQI アプリケーションとは無関係にキュー・マネージャーをモニターするときに使用できます。

観測イベントが発生すると、キュー・マネージャーはイベント・キューにイベント・メッセージを書き込みます。 書き込まれたこのメッセージは、モニター・アプリケーションにより読み取られます。このアプリケーションは、イベントが問題を提示すると、管理者に通知したり何らかの矯正処置を開始します。

注 : トリガー・イベントは観測イベントとは異なります。 トリガー・イベントは、同じ条件では発生せず、イベント・メッセージが生成されません。

観測イベントの詳細については、観測イベントを参照してください。

キュー・マネージャー

キュー・マネージャー およびキュー・マネージャーがアプリケーションに提供するキューイング・サービスの概要です。

プログラムは、キュー・マネージャーのサービスを使用する前に、そのキュー・マネージャーへ接続していなければなりません。 この接続は、プログラムが明示的に (MQCONN または MQCONNX 呼び出しを使用する) 行うこともできますし、暗示的に行われることもあります (これはプログラムを実行するプラットフォームおよび環境に依存します)。

IBM MQ キュー・マネージャーは、以下のアクションを実行します。

- オブジェクトの属性は、受け取ったコマンドに応じて変更されます。
- 該当する条件が満たされたときに、トリガー・イベントや観測イベントなどの特殊イベントが生成されます。
- メッセージが、MQPUT 呼び出しを行ったアプリケーションの要求により、正しいキューに書き込まれます。 正しいキューに入れられなかった場合には、アプリケーションに通知され、該当する理由コードが戻されます。

それぞれのキューは、1 つのキュー・マネージャーに属しており、そのキュー・マネージャーに対してローカル・キュー であるといいます。 アプリケーションが接続されているキュー・マネージャーは、そのアプリケーションに対してローカル・キュー・マネージャー であるといいます。 アプリケーションのローカル・キュー・マネージャーに属しているキューは、そのアプリケーションのためのローカル・キューです。

リモート・キュー とは、別のキュー・マネージャーに属しているキューのことです。 リモート・キュー・マネージャー とは、ローカル・キュー・マネージャー 以外の任意のキュー・マネージャーのことです。 リモート・キュー・マネージャーは、ネットワーク内のリモート・マシン上にある場合と、ローカル・キュー・マネージャーと同じマシン上にある場合があります。 IBM MQ では、1 つのマシン上で複数のキュー・マネージャーを使用することができます。

キュー・マネージャー・オブジェクトは、一部の MQI 呼び出しで使用することができます。 例えば、キュー・マネージャー・オブジェクトの属性について、MQI 呼び出しの MQINQ を使用して問い合わせることができます。

キュー・マネージャーの属性

各キュー・マネージャーに関連付けられているのは、その特性を定義する一連の属性 (または特性) です。 キュー・マネージャーの属性のいくつかは、作成される時に固定されます。他のものについては、IBM MQ コマンドを使用して変更できます。 すべての属性の値は、Transport Layer Security (TLS) 暗号化で使用されるものを除いて、MQINQ 呼び出しを使って問い合わせることができます。

固定されている属性は、次のとおりです。

- キュー・マネージャーの名前
- キュー・マネージャーが稼働するプラットフォーム (例えば、Windows)
- キュー・マネージャーがサポートするシステム制御コマンドのレベル
- キュー・マネージャーが処理するメッセージに割り当てられる最高優先順位
- プログラムが IBM MQ コマンドを送信できるキューの名前
- キュー・マネージャーが処理できるメッセージの最大長 z/OS (IBM MQ for z/OS でのみ固定)
- プログラムがメッセージを書き込んだり読み取ったりするときに、キュー・マネージャーが同期点機能をサポートするかどうか

変更可能な 属性は、次のとおりです。

- キュー・マネージャーのテキスト記述子
- キュー・マネージャーが MQI 呼び出しを処理するときに文字ストリングに使用する文字セットの ID
- トリガー・メッセージの数を制限するためにキュー・マネージャーが使用する時間間隔
- z/OS キュー・マネージャーがキュー内の期限切れメッセージをスキャンする頻度を決定するのに使用される時間間隔 (IBM MQ for z/OS のみ)
- キュー・マネージャーの送達不能 (未配布メッセージ) キュー
- キュー・マネージャーのデフォルトの伝送キューの名前
- 任意の接続のためのオープン・ハンドルの最大数
- イベント報告の各種カテゴリーの設定または解除
- 1 つの作業単位内のコミットされていないメッセージの最大数

キュー・マネージャーとワークロード管理

1 つのキューに対して複数の定義が作成されている場合は、いくつかのキュー・マネージャーからなる 1 つのクラスターを設定することができます (例えば、クラスター内のキュー・マネージャーは、それぞれが他のキュー・マネージャーの複製として機能できます)。 特定のキューに対するメッセージは、そのキューのインスタンスのホストになっているすべてのキュー・マネージャーで処理できます。 ワークロード管理アルゴリズムは、どのキュー・マネージャーがメッセージを処理するか決定し、キュー・マネージャー間でワークロードを分散します。詳細については、クラスター・ワークロード管理アルゴリズムを参照してください。

チャネル

チャネルは、分散キュー・マネージャーが使用する IBM MQ MQI client と IBM MQ サーバー間、または 2 つの IBM MQ サーバー間の論理通信リンクのことです。

あるキュー・マネージャーから別のキュー・マネージャーにメッセージを移すためにチャネルが使用され、それによってアプリケーションは基礎をなす通信プロトコルから遮蔽されます。 キュー・マネージャーは、同一のシステム上に存在する場合もあれば、同一のプラットフォーム上の異なるシステムや、異なるプラットフォーム上に存在する場合もあります。 メッセージの送信元は次のように多岐にわたります。

- あるノードから別のノードにデータを転送するユーザー作成のアプリケーション・プログラム。
- PCF コマンドまたは MQAI を使用するユーザー作成の管理アプリケーション。
- IBM MQ Explorer。
- インスツルメンテーション・イベント・メッセージを別のキュー・マネージャーに送信するキュー・マネージャー。
- リモート管理コマンドを別のキュー・マネージャーに送信するキュー・マネージャー。 例えば、MQSC コマンドや administrative REST API を使用します。

1 つのチャネルには、2 つの定義があり、それらは接続の両端に 1 つずつあります。 キュー・マネージャーが相互に通信できるようにするには、メッセージを送信するキュー・マネージャーに 1 つのチャネル・オブジェクトを定義し、メッセージを受信するキュー・マネージャーに別の補完的なチャネル・オブジェクトを定義する必要があります。 接続の両端には、同じチャネル名 を使用する必要があり、使用するチャネル・タイプ には互換性がなければなりません。

IBM MQ のチャネルは、次に示すように 3 つのカテゴリーに分かれます (これらのカテゴリー内に異なるチャネル・タイプがあります)。

- メッセージ・チャネル。単一方向のもので、あるキュー・マネージャーから他のキュー・マネージャーへメッセージを転送します。
- MQI チャネル。双方向のもので、IBM MQ MQI client からキュー・マネージャーに MQI 呼び出しを転送し、キュー・マネージャーから IBM MQ クライアントに応答を転送します。
- AMQP チャネルは両方向であり、AMQP クライアントをサーバー・マシン上のキュー・マネージャーに接続します。 IBM MQ は AMQP チャネルを使用して、AMQP アプリケーションとキュー・マネージャー間の AMQP 呼び出しおよび応答を転送します。

メッセージ・チャネル

メッセージ・チャネルの目的は、1 つのキュー・マネージャーから別のキュー・マネージャーにメッセージを移すことです。 メッセージ・チャネルは、クライアント・サーバー環境で必須ではありません。

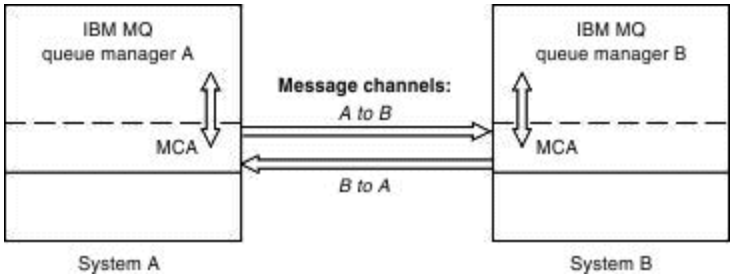


図 *2. 2* つのキュー・マネージャー間のメッセージ・チャネル

メッセージ・チャネルは、単一方向のリンクです。 ローカル・キュー・マネージャーから送信されたメッセージにリモート・キュー・マネージャーを応答させるには、ローカル・キュー・マネージャーに応答を返す別のチャネルをセットアップする必要があります。

メッセージ・チャネルは、メッセージ・チャネル・エージェント (MCA) を使用して 2 つのキュー・マネージャーを接続します。 チャネルの両側に、メッセージ・チャネル・エージェントが 1 つずつあります。 MCA は、複数のスレッドを使用してメッセージを転送できます。 このプロセスを、パイプライン と呼びま

す。 パイプラインを使用すると、MCA のメッセージ転送効率が向上し、その結果、チャネルのパフォーマンスが向上します。 パイプラインについて詳しくは、チャネルの属性を参照してください。

チャネルについて詳しくは、チャネル出口呼び出しおよびデータ構造体と、47 ページの『分散キューイング・コンポーネント』を参照してください。

MQI チャネル

メッセージ・キュー・インターフェース (MQI) チャネルは、 IBM MQ MQI client をサーバー・マシン上のキュー・マネージャーに接続し、IBM MQ MQI client アプリケーションから MQCONN または MQCONNX 呼び出しを発行すると確立されます。

これは両方向のリンクであり、メッセージ・データが入った MQPUT 呼び出しや、メッセージ・データが戻される結果となる MQGET 呼び出しなど、MQI 呼び出しおよび応答だけを転送するのに使用されます。 チャネル定義の作成と使用方法は 2 つあります (MQI チャネルの定義を参照してください)。

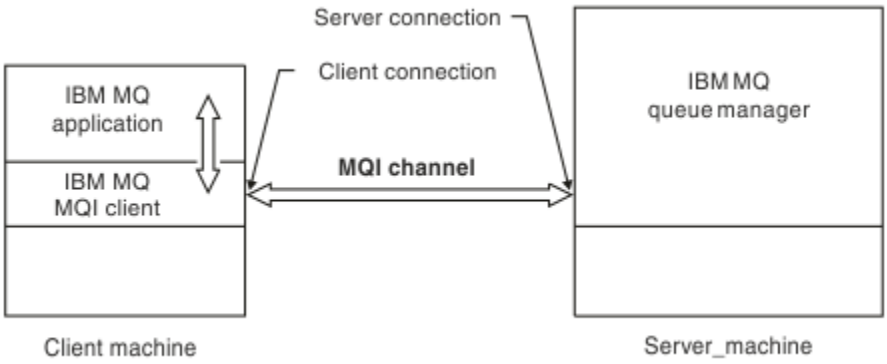


図 *3. MQI* チャネル上のクライアント接続およびサーバー接続

z/OS MQI チャネルを使用して、単一のキュー・マネージャー、あるいはキュー共用グループの一部となっているキュー・マネージャーにクライアントを 接続することができます (キュー共用グループへのクライアントの接続を参照)。

MQI チャネル定義のチャネル・タイプには 2 種類あります。 これらのチャネル・タイプでは、両方向 MQI チャネルを定義します。

クライアント接続チャネル
このタイプは、IBM MQ MQI client 用です。

サーバー接続チャネル
このタイプは、 IBM MQ MQI client 環境で実行されている IBM MQ アプリケーションが通信するキュー・マネージャーを実行しているサーバー用です。

AMQP チャネル

Multi

AMQP チャネルのタイプは 1 つだけです。

チャネルを使用して AMQP メッセージング・アプリケーションをキュー・マネージャーと接続し、アプリケーションが IBM MQ アプリケーションとメッセージを交換できるようにします。 AMQP チャネルにより、MQ Light を使用してアプリケーションを開発し、IBM MQ によって提供されるエンタープライズ・レベルの機能を利用して、そのアプリケーションをエンタープライズ・アプリケーションとしてデプロイすることができます。

クライアント接続チャネル

クライアント接続チャネル とは、IBM MQ MQI client からキュー・マネージャーへの通信パスを提供するオブジェクトのことです。

クライアント接続チャネルは、キュー・マネージャーとクライアントとの間でメッセージを移動するために、分散キューイングで使用されます。 チャネルは、基礎をなす通信プロトコルからアプリケーションを保護します。 クライアントは、キュー・マネージャーと同一のプラットフォーム上に存在する場合もあれば、異なるプラットフォーム上に存在する場合もあります。

チャネル定義

それぞれのチャネルのタイプについては、32 ページの『チャネル定義』を参照してください。

関連概念

43 ページの『分散キューイングとクラスター』
分散キューイングとは、あるキュー・マネージャーから別のキュー・マネージャーにメッセージを送信することです。 受信キュー・マネージャーは、同じマシン上に置くことも別のマシン上に置くこ ともでき、近くにあっても地球の裏側にあってもメッセージを受信できます。 このプログラムは、ローカル・キュー・マネージャーと同じプラットフォーム、または IBM MQ がサポートするいずれかのプラットフォームで実行することができます。 分散キューイング環境内のすべての接続を手動で定義することもできますし、クラスターを作成して IBM MQ で接続詳細の多くが定義されるようにすることもできます。

Message Queue Interface の概要

関連タスク

リモート IBM MQ オブジェクトの管理
MQI チャネルの停止中
サーバーとクライアント間の接続の構成

関連資料

チャネル出口呼び出しおよびデータ構造体

36 ページの『通信』
IBM MQ MQI clients は、サーバーとの通信に MQI チャネルを使用します。

チャネル定義

IBM MQ が使用するメッセージ・チャネルおよび MQI チャネルのさまざまなタイプを説明する表。

メッセージ・チャネルに言及する際、チャネルという言葉はよくチャネル定義の同義語として使用されます。 2 つの側を持つ完全チャネルのことか、または 1 つの側しかないチャネル定義のことなのかは、ふつうはその文脈から明らかです。

メッセージ・チャネル

メッセージ・チャネル定義は、以下のタイプのいずれかです。

メッセージ・チャネル定義のタイプ	説明
送信側	送信側チャネルは、キュー・マネージャーがメッセージを他のキュー・マネージャーに送信するために使用するメッセージ・チャネルです。 送信側チャネルを使用してメッセージを送信するには、 他のキュー・マネージャー上に送信側チャネルと同じ名前の受信側チャネルも作成しなければなりません。 「コールバック」メカニズムを実装している場合は、送信側チャネルを要求側チャネルと共に使用できます。

メッセージ・チャネル定義のタイプ	説明
サーバー	サーバー・チャネルは、キュー・マネージャーがメッセージを他のキュー・マネージャーに送信するために使用するメッセージ・チャネルです。 サーバー・チャネルを使用してメッセージを送信するには、 他のキュー・マネージャー上にサーバー・チャネルと同じ名前の受信側チャネルも作成しなければなりません。 要求側チャネルを持つサーバー・チャネルを使用することもできます。 その場合、 チャネルの他方の終端の要求側チャネル定義によって、サーバー・チャネル定義の開始が要求されます。 サーバーは要求側にメッセージを送信します。 サーバーにパートナー・チャネルの接続名が知られている限り、サーバーも通信を開始することができます。
受信側	受信側チャネルは、キュー・マネージャーがメッセージを他のキュー・マネージャーから受信するために使用するメッセージ・チャネルです。 受信側チャネルを使用してメッセージを受信するには、他のキュー・マネージャー上にこの受信側チャネルと同じ名前の送信側またはサーバー・チャネルも作成しなければなりません。
リクエスター	要求側チャネルは、キュー・マネージャーが他のキュー・マネージャーからメッセージを受信するために使用するメッセージ・チャネルです。 要求側チャネルは、リモート・エンドで定義されているパートナー・チャネルに開始を要求できます。 パートナー・チャネルがサーバー・チャネルの場合、サーバー・チャネルは開始要求を受け入れて、 サーバー・チャネル定義で識別されている伝送キューから要求側チャネルへのメッセージの送信を開始します。 パートナー・チャネルが送信側チャネルの場合、送信側チャネルは開始要求を受け入れますが、その時点で要求側との接続をクローズします。 その後、送信側チャネルが開始し、要求側であるパートナー・チャネルとのセッションをネゴシエーションして、送信側チャネル定義で識別されている伝送キューからのメッセージの送信を開始します。 要求側チャネルが送信側チャネルにコールバックを要求するという点で、この後者のケースは本質的にコールバック・メカニズムを提供しています。

メッセージ・チャネル定義のタイプ	説明
クラスター送信側	クラスター送信側 (CLUSSDR) チャネル定義は、クラスター・キュー・マネージャーがいずれかの フル・リポジトリーにクラスター情報を送信できるチャネルの送信側を定義します。 クラスター送信側チャネルを使用して、キューの追加や削除など、キュー・マネージャーの状況の変化をリポジトリーに 通知します。 また、このチャネルは、メッセージの送信にも使用されます。 フル・リポジトリー・キュー・マネージャー自体に、お互いを指し示すクラスター送信側チャネルがあります。 フル・リポジトリー・キュー・マネージャーは、このチャネルを使用してクラスター状況の変更を相互に通信します。 キュー・マネージャーの CLUSSDR チャネル定義がどのフル・リポジトリーを指しているかは、あまり重要ではありません。 最初の接続が行われた後に、必要に応じて、自動的にクラスター・キュー・マネージャー・オブジェクトがさらに定義されます。 これで、キュー・マネージャーがすべてのフル・リポジトリーにクラスター情報を送信し、 すべてのキュー・マネージャーにメッセージを送信することができるようになります。
クラスター受信側	クラスター受信側 (CLUSRCVR) チャネル定義は、 クラスター・キュー・マネージャーがクラスター内の他のキュー・マネージャーからメッセージを受信できるチャネルの受信側を定義します。 クラスター受信側チャネルは、リポジトリー宛てのクラスター情報に関する情報も伝送できます。 クラスター受信側チャネルを定義すると、キュー・マネージャーは、 メッセージを受信することができることを他のクラスター・キュー・マネージャーに示します。 各クラスター・キュー・マネージャーごとに、少なくとも 1 つのクラスター受信側チャネルが必要です。

それぞれのチャネルごとに両端を定義し、チャネルのそれぞれの終端にチャネル定義があるようにする必要があります。 チャネルの両端のタイプは互換タイプでなければなりません。

以下のチャネル定義の組み合わせが可能です。

- 送信側 - 受信側
- サーバー - 受信側
- 要求側 - サーバー
- 要求側 - 送信側 (コールバック)
- クラスター送信側 - クラスター受信側

メッセージ・チャネル・エージェント

作成する各チャネル定義は、特定のキュー・マネージャーに属している必要があります。 キュー・マネージャーは、同じまたは異なるタイプのいくつかのチャネルを持つことができます。 チャネルの両端に、プログラム、メッセージ・チャネル・エージェント (MCA) があります。 チャネルの片側で、呼び出し側 MCA が伝送キューからメッセージを取り出し、それらをチャネルを通して送信します。 チャネルのもう一方の側で、応答側 MCA がメッセージを受信し、リモート・キュー・マネージャーへそれらを配信します。

呼び出し側 MCA は、送信側、サーバー、または要求側チャネルと関連している場合があります。 応答側 MCA は、どのタイプのメッセージ・チャネルとも関連できます。

IBM MQ は、接続の両側で次のチャネル・タイプの組み合わせをサポートしています。

呼び出し側		メッセージ・フローの方向	応答側	
チャネル・タイプ	**リスナーが必要か**		**リスナーが必要か**	**チャネル・タイプ**
送信側	いいえ	呼び出し側から応答側	はい	受信側
サーバー	いいえ	呼び出し側から応答側	はい	受信側
サーバー	いいえ	呼び出し側から応答側	はい	リクエスター
リクエスター	いいえ	応答側から呼び出し側	はい	サーバー
リクエスター	はい	応答側から呼び出し側	はい	送信側

MQI チャネル

MQI チャネルは次のタイプの 1 つです。

MQI チャネル・タイプ	説明
サーバー接続	サーバー接続チャネルは、双方向の MQI チャネルで、IBM MQ クライアントを IBM MQ サーバーに接続するために使用されます。 サーバー接続チャネルとは、チャネルのサーバー側です。
クライアント接続	クライアント接続チャネルは、双方向の MQI チャネルで、IBM MQ クライアントを IBM MQ サーバーに接続するために使用されます。 IBM MQ Explorer は、リモート・キュー・マネージャーへの接続に、クライアント接続も使用します。 クライアント接続チャネルとは、チャネルのクライアント側です。 クライアント接続チャネルを作成すると、 キュー・マネージャーをホストするコンピューターにファイルが作成されます。 その後、クライアント接続ファイルを IBM MQ クライアント・コンピューターにコピーする必要があります。

Multi マルチスレッド・サポート - パイプライン

オプションで、メッセージ・チャネル・エージェント (MCA) により、複数のスレッドを使用してメッセージを転送できます。 このプロセスのことをパイプライン といい、このプロセスを使用すると、MCA によるメッセージ転送の効率が上がり、待ち状態が少なくなり、チャネルのパフォーマンスが向上します。 MCA 当たり最大 2 つのスレッドに限定されています。

パイプラインを制御するには、qm.ini ファイル中で *PipeLineLength* パラメーターを使用します。 このパラメーターは、 Channels スタンザに追加されます。

注 : パイプラインは TCP/IP チャネルの場合だけ有効です。

パイプラインを使用する場合は、*PipeLineLength* が 1 より大きくなるようにチャネルの両側のキュー・マネージャーを構成する必要があります。

チャネル出口に関する考慮事項

次の理由で、パイプラインによって一部の出口プログラムが失敗します。

- 出口が逐次に呼び出されない。
- 出口が別のスレッドから代替呼び出しされる。

パイプラインを使用する場合は、その前に以下の点について出口プログラムの設計を確認してください。

- 出口はすべての実行段階で再入可能でなければならない。
- MQI 呼び出しを使用する場合は、別々のスレッドから出口が呼び出されるときは同一の MQI ハンドルを使用できないことを念頭に置かなければならない。

あるメッセージ出口が、キューをオープンし、それ以降のすべての出口呼び出しでこのハンドルを使用して MQPUT 呼び出しを行うとします。 この出口は別のスレッドから呼び出されるので、パイプライン・モードでは失敗します。 失敗しないようにするには、スレッドごとにキュー・ハンドルを保持し、出口が呼び出されるたびにスレッド ID を検査してください。

通信

IBM MQ MQI clients は、サーバーとの通信に MQI チャネルを使用します。

チャネル定義は、IBM MQ MQI client とサーバーの接続の両端で作成する必要があります。 チャネル定義の作成方法については、MQI チャネルの定義で説明されています。

次のテーブルに使用可能な伝送プロトコルを示します。

表 *1. MQI* チャネルの伝送プロトコル

クライアント・プラットフォーム	LU 6.2	TCP/IP	NetBIOS	SPX
IBM i IBM i		はい		
Linux AIX AIX and Linux システム	はい [1]	はい		
Windows Windows	はい	はい	はい	はい

注 :

1. Linux 以下のプラットフォーム上では、LU.2 はサポートされていません。
   - Linux (POWER プラットフォーム)
   - Linux (x86-64 プラットフォーム)
   - Linux (zSeries s390x プラットフォーム)

伝送プロトコル - IBM MQ MQI client とサーバーのプラットフォームの組み合わせに、これらの伝送プロトコルを使用して実現できる IBM MQ MQI client およびサーバーのプラットフォームの組み合わせが示されています。

IBM MQ MQI client 上の IBM MQ アプリケーションは、キュー・マネージャーがローカルの場合と同じ方法ですべての MQI 呼び出しを使用できます。 **MQCONN** または **MQCONNX** は、選択したキュー・マネージャーに IBM MQ アプリケーションを関連付け、接続ハンドル を作成します。 接続ハンドルを使用するその他の呼び出しは、その後、接続されたキュー・マネージャーに処理されます。 IBM MQ MQI client 通信は、クライアントとサーバーの間にアクティブな接続を必要とします。対照的に、キュー・マネージャー間の通信は、接続にも時間にも依存しません。

伝送プロトコルはチャネル定義を使用することで指定され、アプリケーションに影響しません。 例えば、Windows アプリケーションは、TCP/IP を介してあるキュー・マネージャーと接続し、NetBIOS を介して別のキュー・マネージャーと接続することができます。

パフォーマンス上の考慮事項

使用する伝送プロトコルが、IBM MQ クライアント/サーバー・システムのパフォーマンスに影響することがあります。 伝送速度が遅い特定の状況では、 IBM MQ チャネル圧縮を使用できます。

IBM MQ オブジェクトの命名

IBM MQ オブジェクトに適用される命名規則は、オブジェクトごとに異なります。 また、IBM MQ で使用するマシンの名前およびユーザー ID は、いくつかの命名規則に従います。

キュー・マネージャーの各インスタンスは、その名前で見分けられます。 この名前は、相互接続されたキュー・マネージャーのネットワーク内で固有である必要があります。 固有になっていると、あるキュー・マネージャーは、所定のメッセージを送るべきターゲットのキュー・マネージャーを明確に識別することができます。

他のタイプのオブジェクトの場合、各オブジェクトにはそれぞれ関連付けられた名前があり、各オブジェクトはその名前で参照できます。 これらの名前は、1 つのキュー・マネージャーおよびオブジェクト・タイプ内において固有のものである必要があります。 例えば、同じ名前のキューとプロセスを持つことはできますが、同じ名前の 2 つのキューを持つことはできません。

IBM MQ では、名前には最大 48 文字まで使用することができます。ただし、チャネル は例外で、最大 20 文字まで使用できます。 IBM MQ オブジェクトの命名規則の詳細については、37 ページの『IBM MQ オブジェクトの命名規則』 を参照してください。

IBM MQ で使用するマシンの名前およびユーザー ID は、以下のいくつかの命名規則にも従います。

- マシン名にスペースが含まれていないことを確認します。 IBM MQ は、スペースが含まれているマシン名をサポートしていません。 名前にスペースが含まれているマシンに IBM MQ をインストールした場合は、キュー・マネージャーを作成できなくなります。
- IBM MQ 権限のためのユーザー ID およびグループの名前は、20 文字以内にする必要があります (スペースは使用できません)。
- Windows IBM MQ for Windows サーバーでは、@ 文字を含むユーザー ID (例えば abc@d) の下で IBM MQ MQI client が実行されている場合、そのクライアントの接続はサポートされません。

関連概念

40 ページの『IBM MQ ファイル名』
IBM MQ のキュー・マネージャー、キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、および認証情報オブジェクトは、それぞれファイルで表されます。 これらのオブジェクト名は必ずしも有効なファイル名ではないので、キュー・マネージャーは、必要に応じてそのオブジェクト名を有効なファイル名に変換します。

関連資料

37 ページの『IBM MQ オブジェクトの命名規則』
IBM MQ オブジェクト名には最大長があり、大/小文字が区別されます。 すべてのオブジェクト・タイプにすべての文字がサポートされているわけではなく、多くのオブジェクトには名前の固有性に関する規則があります。

IBM MQ オブジェクトの命名規則

IBM MQ オブジェクト名には最大長があり、大/小文字が区別されます。 すべてのオブジェクト・タイプにすべての文字がサポートされているわけではなく、多くのオブジェクトには名前の固有性に関する規則があります。

IBM MQ オブジェクトには多種多様なタイプがあり、オブジェクトはタイプごとに別個のオブジェクト名前空間に存在するので、タイプの異なる複数オブジェクトが同じ名前を持つことができます。例えば、ローカル・キューと送信側チャネルとが同じ名前であってもかまいません。 しかし、あるオブジェクトが同じ名前空間にある別のオブジェクトと同じ名前を持つことはできません。例えば、ローカル・キューはモデル・キューと同じ名前を持つことができず、送信側チャネルは受信側チャネルと同じ名前を持つことができません。

以下の IBM MQ オブジェクトは、別個のオブジェクト名前空間に存在します。

- 認証情報
- チャネル
- クライアント・チャネル

- リスナー
- 名前リスト
- プロセス
- キュー
- サービス
- ストレージ・クラス
- サブスクリプション
- トピック

オブジェクト名の文字長

一般に、IBM MQ オブジェクト名の最大長は 48 文字です。 この規則は、次のオブジェクトに当てはまります。

- 認証情報
- クラスター
- リスナー
- 名前リスト
- プロセス定義
- キュー
- キュー・マネージャー
- サービス
- サブスクリプション
- トピック

以下の制限があります。

1. z/OS z/OS システムでは、キュー・マネージャー名には、最大 4 文字の大文字および数字のみ使用可能です。
2. チャネル・オブジェクト名とクライアント接続チャネル名の最大長は 20 文字です。 チャネルについて詳しくは、チャネルの定義を参照してください。
3. トピック・ストリングに使用できるのは、最大 10240 バイトです。 すべての IBM MQ オブジェクト名は大/小文字を区別します。
4. サブスクリプション名には、最大 10240 バイトのサイズにすることができ、スペースを含めることができます。
5. ストレージ・クラス名の最大長は 8 文字です。
6. CF 構造名の最大長は 12 文字です。

オブジェクト名の文字

IBM MQ オブジェクト名に有効な文字は、次のとおりです。

文字	制約事項
大文字の A から Z	• なし

文字	制約事項
小文字の a から z	• MQSC スクリプトでは、小文字を含む名前を単一引用符で囲む必要があります。 これにより、小文字が大文字に変換されなくなります。 • EBCDIC カタカナを使用するシステムは、オブジェクト名に小文字の a から z の文字を使用できません。 • z/OS z/OS システムで小文字を使用する場合、キュー・マネージャー名に小文字を含めることができないなどの制限があることがあります。 • IBM i IBM i システムで CL コマンドを使用する場合は、小文字を含む名前を単一引用符で囲む必要があります。 これにより、小文字が大文字に変換されなくなります。
数字 0 から 9	• なし
ピリオド (.)	• なし
下線 (_)	• Multi なし • z/OS 名前の前または後に下線を付加することは、IBM MQ for z/OS 操作および制御パネルで処理できないため、避けてください。
スラッシュ (/)	• Windows Windows システムでは、キュー・マネージャー名の最初の文字をスラッシュにすることはできません。 • IBM i IBM i システムで CL コマンドを使用する場合は、スラッシュを含む名前を単一引用符で囲む必要があります。 • z/OS なし
パーセント記号 (%)	• ALW なし • z/OS IBM MQ for z/OS の外部セキュリティー・マネージャーとして RACF® を使用している場合は、オブジェクト名に% を使用しないでください。 これは、RACF 総称プロファイルの使用時に名前がセキュリティー検査に含まれないためです。 • IBM i IBM i システムで CL コマンドを使用する場合は、% 記号を含む名前を単一引用符で囲む必要があります。

オブジェクト名の文字に関しては、いくつかの一般的な規則もあります。

1. 先行ブランクや組み込みブランクは使用できません。
2. 各国語文字は使用不可です。
3. フィールド長全体に長さが満たない名前は、右側にブランクが埋め込まれる場合があります。 キュー・マネージャーから返されるすべてのショート・ネームには、必ず右側にブランクが埋め込まれます。

キュー名

キューの名前には次の 2 つの部分があります。

- キュー・マネージャーの名前。
- そのキュー・マネージャーによって認識されているキューのローカル名

キュー名の各部分の長さは 48 文字です。

ローカル・キューを参照する場合は、キュー・マネージャーの名前を (ブランク文字に置き換えたり、先行ヌル文字を使用したりして) 省略できます。 しかし、IBM MQ によってプログラムに返されるすべてのキュー名には、キュー・マネージャーの名前が含まれます。

z/OS 共用キュー (キュー共用グループ内のどのキュー・マネージャーにもアクセス可能) は、同じキュー共用グループ内のどの非共用ローカル・キューとも同じ名前にすることはできません。 この制限により、アプリケーションがローカル・キューをオープンするはずが間違えて共用キューをオープンしてしまうこと (またはその逆) を防げます。 共用キューおよびキュー共用グループは IBM MQ for z/OS でのみ使用可能です。

リモート・キューを参照する場合、プログラムは完全なキュー名にキュー・マネージャーの名前を含めるか、またはリモート・キューのローカル定義が存在する必要があります。

アプリケーションがキュー名を使用する場合、その名前はローカル・キューの名前 (またはその別名) またはリモート・キューのローカル定義の名前いずれかにすることができますが、アプリケーションがキューからメッセージを受け取る必要がない場合 (キューがローカルでなければならない場合) は、アプリケーションがそのいずれであるかを認識する必要はありません。 アプリケーションがキュー・オブジェクトをオープンすると、MQOPEN 呼び出しはネーム・レゾリューション機能を実行して、それ以降の操作をどのキューに対して実行するかを判別します。 ここで大切な点は、キュー・マネージャーのネットワーク内で特定の場所に定義されている特定のキューに対する依存関係がアプリケーションに組み込まれてはいないということです。 したがって、システム管理者がネットワーク内でキューを再配置し、その定義を変更しても、そのキューを使用するアプリケーションを変更する必要はありません。

予約オブジェクト名

SYSTEM. で始まるオブジェクト名は、キュー・マネージャーで定義されるオブジェクト用に予約されています。 **Alter**、**Define**、および **Replace** コマンドを使用して、ご使用のインストール環境に合わせてこれらのオブジェクト定義を変更できます。 IBM MQ 用に定義されている名前は、キュー名にすべてリストされています。

z/OS IBM MQ for z/OS では、カップリング・ファシリティー・アプリケーション構造名 CSQSYSAPPL は予約済みです。

関連概念
AIX, Linux, and Windows でのインストール名

IBM MQ ファイル名

IBM MQ のキュー・マネージャー、キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、および認証情報オブジェクトは、それぞれファイルで表されます。 これらのオブジェクト名は必ずしも有効なファイル名ではないので、キュー・マネージャーは、必要に応じてそのオブジェクト名を有効なファイル名に変換します。

キュー・マネージャー・ディレクトリーへのデフォルトのパスは、次のものから作られます。

- 接頭部。次のように、IBM MQ 構成情報に定義されています。
  - Linux AIX AIX and Linux では、デフォルトの接頭部は `/var/mqm` です。 これは、mqs.ini 構成ファイルの `DefaultPrefix` スタンザで構成されます。

– Windows Windows 32 ビット・システムでは、デフォルトの接頭部は C:¥Program Files (x86)¥IBM¥WebSphere MQ です。 Windows 64 ビット・システムでは、デフォルトの接頭部は C:¥Program Files¥IBM¥MQ です。 32 ビットおよび 64 ビットのどちらのインストール済み環境でも、データ・ディレクトリーは C: ¥ProgramData¥IBM¥MQ にインストールされます。 これは、mqs.ini 構成ファイルの DefaultPrefix スタンザで構成されます。

使用可能な場合、接頭部は「IBM MQ エクスプローラーの IBM MQ プロパティー」ページを使用して変更できます。そうでない場合は、手動で mqs.ini 構成ファイルを編集します。

- キュー・マネージャー名は、有効なディレクトリー名に変換されます。 例えば、次のとおりです。

```
queue.manager
```

このキュー・マネージャーは、次のように表されます。

```
queue!manager
```

この処理を、名前変換 と呼びます。

IBM MQ では、キュー・マネージャーに 48 文字までの名前を付けることが できます。

例えば、キュー・マネージャーに次の名前を付けることができます。

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

しかし、各キュー・マネージャーはファイルで表されるため、ファイル名の最大長および名前に使用できる文字に制限があります。 そのため、オブジェクトを表すファイルの名前は、ファイル・システムの要件に合うように自動的に変換されます。

キュー・マネージャー名の変換の規則を以下に示します。

1. 個々の文字が次のように変換されます。
   - 開始。 先へ!
   - / が & に変換される
2. 名前が正しくない場合は次のようになります。
   a. 8 文字に切り捨てられる。
   b. 3 文字の数値接尾部が付加される。

例えば、デフォルト接頭部であるとすると、queue.manager という名前のキュー・マネージャーは次のようになります。

- Windows NTFS または FAT32 の Windows では、キュー・マネージャー名は次のようになります。

```
C:&#xa5;Program Files&#xa5;IBM&#xa5;MQ\qmgrs\queue!manager
```

- Windows FAT の Windows では、キュー・マネージャー名は次のようになります。

```
C:&#xa5;Program Files&#xa5;IBM&#xa5;MQ\qmgrs\queue!ma
```

- Linux AIX AIX and Linux では、キュー・マネージャー名は次のようになります。

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

変換アルゴリズムでは、大文字小文字の区別のないファイル・システム上で、大文字小文字の違いのみの名前を区別することもできます。

オブジェクト名の変換

オブジェクト名は、必ずしも有効なファイル・システム名になっていません。 オブジェクト名を変換することが必要な場合があります。 使用される変換方法は、キュー・マネージャー名の場合とは異なります。 つまり、1 つのマシンにつきキュー・マネージャー名はわずかしかありませんが、各キュー・マネージャーごとに相当数の他のオブジェクトが存在する可能性があるためです。 キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、および認証情報オブジェクトは、ファイル・システム内に表示されます。

変換処理で新しい名前が生成された場合、元のオブジェクト名との関係は簡単には分かりません。 **dspmqfls** コマンドを使用して、本当のオブジェクト名と変換されたオブジェクト名との間での変換を行うことができます。

関連資料

dspmqfls (ファイル名の表示)

関連情報

mqs.ini ファイルの AllQueueManagers スタンザ

IBM i でのオブジェクト名

キュー・マネージャーには、固有の名前を持つ関連するキュー・マネージャー・ライブラリーがあります。 IBM i 統合ファイル・システム (IFS) の要件を満たすために、キュー・マネージャー名およびオブジェクト名の変換が必要になる場合があります。

キュー・マネージャーが作成されると、IBM MQ はキュー・マネージャー・ライブラリーをそれに関連付けます。 このキュー・マネージャー・ライブラリーには、主にユーザーが定義したキュー・マネージャーの名前に基づいた 10 文字以下の固有の名前が付けられます。 キュー・マネージャーとキュー・マネージャー・ライブラリーは共に、接頭部が /QIBM/UserData/mqm であるキュー・マネージャーの名前に基づいたディレクトリーに置かれます。 キュー・マネージャー、キュー・マネージャー・ライブラリー、およびディレクトリーの例を以下に示します。

キュー・マネージャー名	ORANGE
キュー・マネージャー・ライブラリー名	QMORANGE
ディレクトリー	/QIBM/UserData/mqm/ORANGE

すべてのキュー・マネージャー名およびキュー・マネージャー・ライブラリー名は、ファイル /QIBM/UserData/mqm/mqs.ini 内のスタンザに書き込まれます。

IBM MQ IFS ディレクトリーおよびファイル

IBM i Integrated File System (IFS) は、データを保管するために IBM MQ によって広範囲に使用されます。 IFS の詳細については、「*Integrated File System Introduction*」を参照してください。

各 IBM MQ オブジェクト (例えば、チャネルやキュー・マネージャーなど) は、ファイルで表されます。 これらのオブジェクト名は必ずしも有効なファイル名ではないので、キュー・マネージャーは、必要に応じてそのオブジェクト名を有効なファイル名に変換します。

キュー・マネージャー・ディレクトリーへのパスは、次のものから作られます。

- 接頭部は、キュー・マネージャー構成ファイル qm.ini に定義されています。 デフォルトの接頭部は /QIBM/UserData/mqm です。
- リテラル - qmgrs。
- コード化されたキュー・マネージャー名。 これは、有効なディレクトリー名に変換されたキュー・マネージャー名です。 例えば、キュー・マネージャー queue/manager は、queue&manager のように表されます。

この処理のことを、名前変換と呼びます。

IFS キュー・マネージャー名の変換

IBM MQ では、キュー・マネージャーに 48 文字までの名前を付けることが できます。

例えば、キュー・マネージャーには QUEUE/MANAGER/ACCOUNTING/SERVICES という名前を付けることができます。 ライブラリーがそれぞれのキュー・マネージャーに対して作成されるのと同じ方法で、それぞれのキュー・マネージャーもファイルによって表されます。 EBCDIC のコード・ポイントには変種があるため、その名前で使用できる文字には制限があります。 そのため、オブジェクトを表す IFS ファイルの名前は、ファイル・システムの要件に合うように自動的に変換されます。

例えば、queue/manager という名前のキュー・マネージャーで、文字 / を & に変換し、デフォルト接頭部がある場合、IBM MQ for IBM i でのキュー・マネージャー名は /QIBM/UserData/mqm/qmgrs/queue&manager になります。

オブジェクト名の変換

オブジェクト名は、必ずしも有効なファイル・システム名になっていません。 そのため、オブジェクト名を変換しなければならない場合があります。 使用される変換方法は、キュー・マネージャー名の場合とは異なります。 つまり、各マシンにキュー・マネージャー名はわずかしかありませんが、各キュー・マネージャーごとに相当数の他のオブジェクトが存在する可能性があるためです。 ファイル・システムには、プロセス定義、キュー、名前リスト、のみ提示されます。 チャネルは関係しません。

変換処理で新しい名前が生成された場合、元のオブジェクト名との関係は簡単には分かりません。 DSPMQMOBJN コマンドを使用すると、IBM MQ オブジェクトの変換後の名前を表示することができます。

分散キューイングとクラスター

分散キューイングとは、あるキュー・マネージャーから別のキュー・マネージャーにメッセージを送信することです。 受信キュー・マネージャーは、同じマシン上に置くことも別のマシン上に置くこ ともでき、近くにあっても地球の裏側にあってもメッセージを受信できます。 このプログラムは、ローカル・キュー・マネージャーと同じプラットフォーム、または IBM MQ がサポートするいずれかのプラットフォームで実行することができます。 分散キューイング環境内のすべての接続を手動で定義することもできますし、クラスターを作成して IBM MQ で接続詳細の多くが定義されるようにすることもできます。

分散キュー

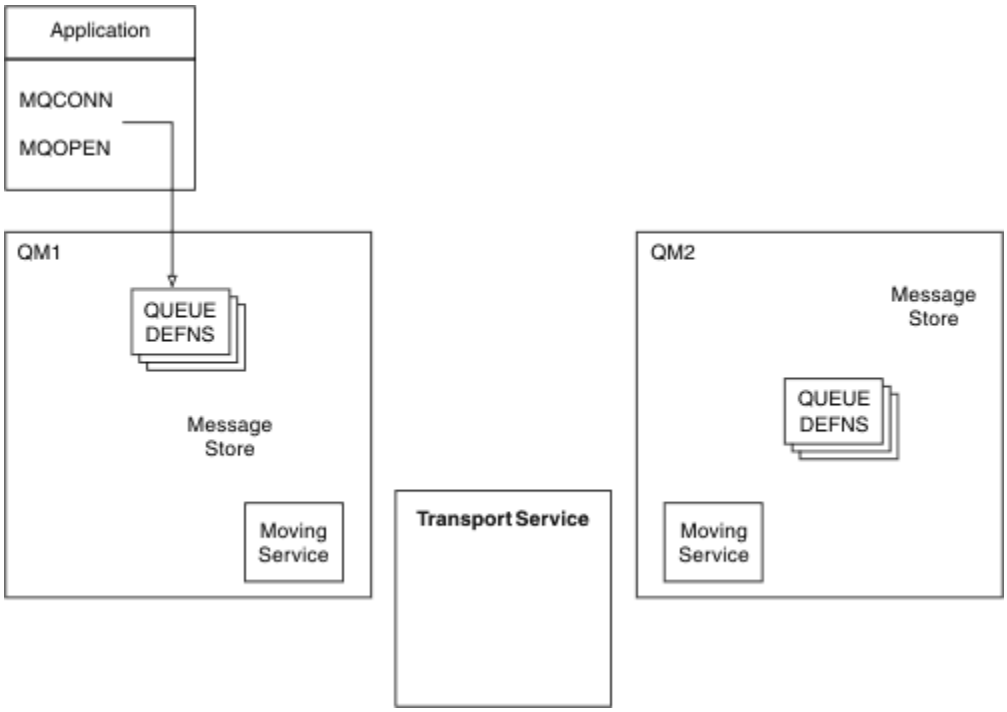


図 4. 分散キューイングのコンポーネントの概要

図の説明:

- アプリケーションは、MQCONN 呼び出しを使用してキュー・マネージャーに接続します。 次に、アプリケーションは、メッセージをキューに書き込めるように MQOPEN 呼び出しを使用してキューをオープンします。
- キュー・マネージャーごとに、所有する各キューの定義を持っています。 ローカル・キュー (このキュー・マネージャーによってホストされるキュー) の定義と、リモート・キュー (他のキュー・マネージャーによってホストされるキュー) の定義を持つことができます。
- リモート・キュー宛のメッセージは、ローカル・キュー・マネージャーが伝送キュー 上に保持し、伝送キューは、それらのメッセージをリモート・キュー・マネージャーに転送できるようになるまでメッセージ・ストアに保持します。
- キュー・マネージャーにはそれぞれ移動サービス と呼ばれる通信ソフトウェアが備えられています。 このソフトウェアを介してキュー・マネージャーは他のキュー・マネージャーと通信することができます。
- トランスポート・サービス は、キュー・マネージャーが関与しないサービスであり、次のいずれかになります (プラットフォームにより異なります)。
  - SNA APPC
  - 伝送制御プロトコル / インターネット・プロトコル (TCP/IP) (Transmission Control Protocol/Internet Protocol (TCP/IP))
  - NetBIOS
  - SPX

メッセージの送信に必要なコンポーネント

メッセージがリモート・キュー・マネージャーに送信される場合、ローカル・キュー・マネージャーには伝送キューとチャネルに関する定義が必要です。 チャネルは、2 つのキュー・マネージャーをつなぐ一方向の通信リンクです。 チャネルを使用して、リモート・キュー・マネージャーの任意の数のキュー宛てのメッセージを送ることができます。

チャネルの両側には、例えば送信側や受信側として定義される独立した定義が備えられています。 簡単なチャネルは、ローカル・キュー・マネージャーの送信側 チャネル定義とリモート・キュー・マネージャーの受信側 チャネル定義から構成されます。 これらの 2 つの定義は同じ名前でなければならず、両方の定義を合わせて 1 つのチャネルが構成されます。

メッセージの送受信を処理するソフトウェアは、メッセージ・チャネル・エージェント (MCA) と呼ばれます。 チャネルの両側に、メッセージ・チャネル・エージェント (MCA) が 1 つずつあります。

それぞれのキュー・マネージャーには 送達不能キュー (未配布メッセージ・キュー とも呼びます) が入っていなければなりません。 メッセージを宛先に送達できない場合、メッセージはこのキューに書き込まれます。

次の図は、キュー・マネージャー、伝送キュー、チャネル、および MCA の関係を示しています。

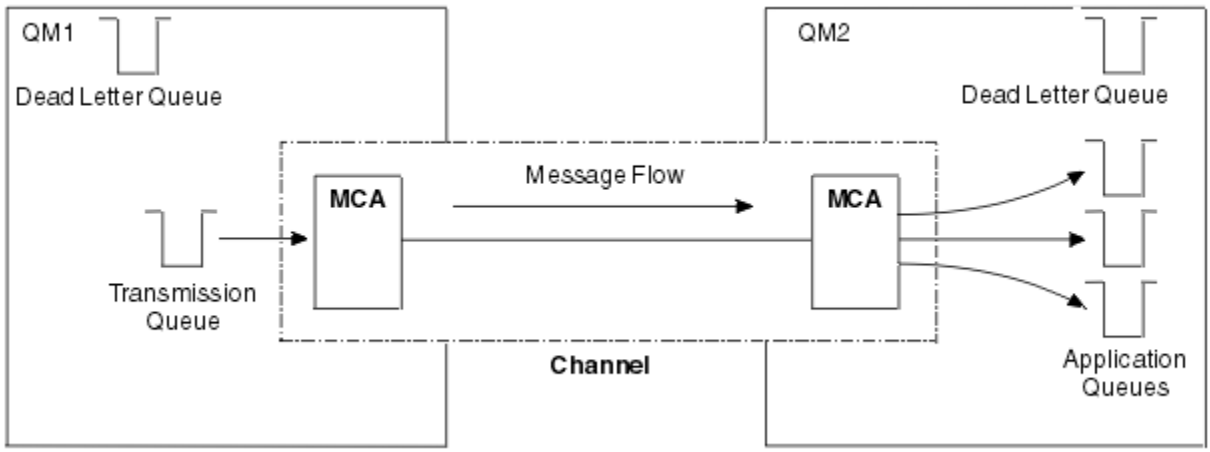


図 5. メッセージの送信

メッセージが戻るために必要なコンポーネント

リモート・キュー・マネージャーからアプリケーションにメッセージを返す必要がある場合は、以下の図に示すように、キュー・マネージャー間で反対方向に機能する別のチャネルを定義する必要があります。

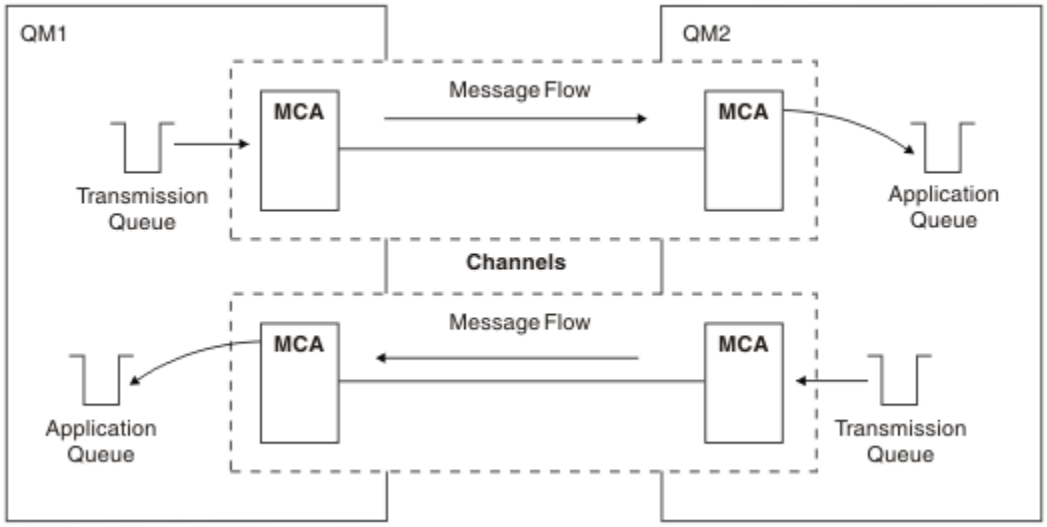


図 6. 両方向のメッセージ送信

クラスター

分散キューイング環境内のすべての接続を手動で定義する代わりに、一連のキュー・マネージャーをクラスターにグループ化することができます。 こうすると、キュー・マネージャーのホストするキューは、宛先ごとに明示的なチャネル定義、リモート・キュー定義、または伝送キューを用意しなくても、クラスター内の他のキュー・マネージャーで使用できるようになります。 クラスター内のすべてのキュー・マネージャーは単一の伝送キューを備えており、その伝送キューによってクラスター内の別のキュー・マネージャーへメッセージが伝送されます。 キュー・マネージャーごとに、1 つのクラスター受信側チャネルと 1 つのクラスター送信側チャネルのみ定義する必要があります。 追加のチャネルはクラスターによって自動的に管理されます。

IBM MQ クライアントは、他のキュー・マネージャーに接続できるのと同じように、クラスターを構成しているキュー・マネージャーに接続できます。 手動で構成した分散キューイングと同様に、任意のキュー・マネージャーのキューにメッセージを書き込むには、MQPUT 呼び出しを使用します。 ローカル・キューからメッセージを取り出すには、MQGET 呼び出しを使用します。

クラスターをサポートするプラットフォーム上にあるキュー・マネージャーは、クラスター構成にしなくても構いません。 クラスターを使用せずに分散キューイングを手動で構成することも、クラスターを使用しながら分散キューイングの構成を行うこともできます。

クラスターを使用する利点

クラスター化には、次の 2 つの主な利点があります。

- クラスター化により、チャネル、伝送キュー、およびリモート・キューの構成のために通常は多くのオブジェクト定義を必要とする IBM MQ ネットワークの管理が簡略化されます。 これは、多数のキュー・マネージャーを相互接続する必要があり、変更される可能性のある大規模ネットワークに特に当てはまります。 そのようなアーキテクチャーでは、構成や頻繁な保守が特に困難です。
- また、クラスターを使用すると、クラスター内のキューおよびキュー・マネージャーの間でメッセージ・トラフィックのワークロードを分散させることができます。 このような分散により、1 つのキューのメッセージ・ワークロードを、複数のキュー・マネージャー上にあるそのキューの複数の同等インスタンスに分散させることができます。 ワークロードの分散は、システム障害に対する回復力、およびシステム内で特にアクティブなメッセージ・フローのスケーリング・パフォーマンスを向上させる上で役立ちます。 そのような環境では、分散キューの各インスタンスに、メッセージを処理するコンシューム側アプリケーションが存在します。 詳細については、クラスターによるワークロードの管理を参照してください。

クラスター内でメッセージが経路指定される方法

クラスターは、信頼できるシステム管理者によって管理されているキュー・マネージャーのネットワークと見なすことができます。 クラスター・キューを定義すると、システム管理者は対応するリモート・キューの定義を必要に応じて他のキュー・マネージャーに対して自動的に作成します。

IBM MQ のクラスター内の各キュー・マネージャーには伝送キューが 1 つ設定されているので、伝送キューを定義する必要はありません。 この単一の伝送キューにより、クラスター内のその他のキュー・マネージャーにメッセージを送信することができます。 伝送キューを 1 つしか使用できないという制限はありません。 キュー・マネージャーは複数のクラスターを使用して、クラスター内の各キュー・マネージャーにメッセージを別々に送信できます。 通常、1 つのキュー・マネージャーは 1 つのクラスター伝送キューを使用します。 キュー・マネージャー属性 DEFCLXQ を変更して、キュー・マネージャーがクラスター内のキュー・マネージャーごとに異なるクラスター伝送キューを使用するように設定することもできます。 クラスター伝送キューを手動で定義することもできます。

クラスターを構成するキュー・マネージャーはすべて、上述のような方法で処理を実行します。 各キュー・マネージャーは、各キュー・マネージャー自体に関する情報とホストしているキューに関する情報を送信し、同じクラスター内にあるその他のキュー・マネージャーに関する情報を受信します。

キュー・マネージャーが使用できなくなった場合に情報の損失を防ぐには、クラスター内の 2 つのキュー・マネージャーをフル・リポジトリー として動作するように指定します。 これらのキュー・マネージャーは、クラスター内のすべてのキュー・マネージャーとキューに関するすべての情報を保管します。 クラスター内の他のすべてのキュー・マネージャーは、これらのキュー・マネージャーに関する情報と、メッセージの交換に使用するキューに関する情報のみを保管します。 このようなキュー・マネージャーは、部分リポジトリー として知られています。 詳しくは、57 ページの『クラスター・リポジトリー』 を参照してください。

クラスターの一部になるために、キュー・マネージャーには、クラスター送信側チャネルおよびクラスター受信側チャネルの 2 つのチャネルが必要です。

- クラスター送信側チャネルとは、送信側チャネルに似た通信チャネルです。 キュー・マネージャーに手動で 1 つのクラスター送信側チャネルを作成し、既にクラスターのメンバーである完全リポジトリーに接続する必要があります。
- クラスター受信側チャネルとは、受信側チャネルに似た通信チャネルです。 クラスター受信側チャネルを 1 つ手動で作成する必要があります。 このチャネルはキュー・マネージャーの仕組みとして機能し、クラスター通信を受信します。

このキュー・マネージャーとクラスターの他のメンバーとの通信に必要となる他のすべてのチャネルは自動的に作成されます。

以下の図は、CLUSTER というクラスターのコンポーネントを示しています。

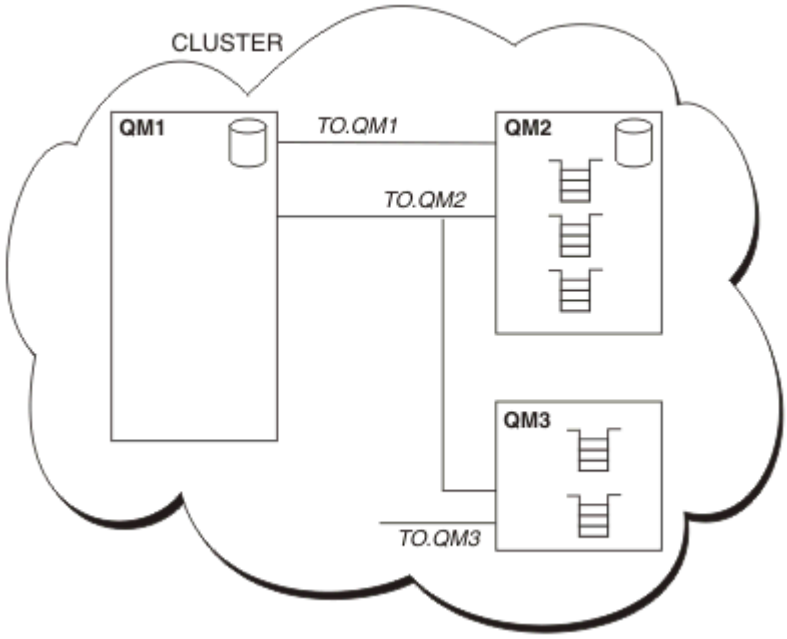


図 7. 複数のキュー・マネージャーで構成されるクラスター

- CLUSTER には、QM1、QM2、および QM3 という 3 つのキュー・マネージャーが備えられています。
- QM1 と QM2 は、クラスター内にあるキュー・マネージャーとキューに関する情報の全リポジトリーのホストです。
- QM2 と QM3 は、いくつかのクラスター・キュー (つまり、このクラスター内にある別のキュー・マネージャーからアクセス可能なキュー) のホストです。
- それぞれのキュー・マネージャーは、メッセージを受信する TO.qmgr というクラスター受信側チャネルを 1 つずつ備えています。
- また、それぞれのキュー・マネージャーは、リポジトリー・キュー・マネージャーの 1 つに情報を送信するクラスター送信側チャネルも 1 つずつ備えています。
- QM1 と QM3 は、QM2 のリポジトリーへ送信し、QM2 は QM1 のリポジトリーへ送信します。

分散キューイング・コンポーネント

分散キューイングのコンポーネントは、メッセージ・チャネル、メッセージ・チャネル・エージェント、伝送キュー、チャネルのイニシエーターとリスナー、およびチャネル出口プログラムです。 メッセージ・チャネルのそれぞれの側の定義は、複数のタイプのいずれか 1 つになります。

メッセージ・チャネルは、あるキュー・マネージャーから別のキュー・マネージャーへメッセージを送るチャネルです。 メッセージ・チャネルと MQI チャネルを混同しないようにしてください。 MQI チャネルには、サーバー接続 (SVRCONN) とクライアント接続 (CLNTCONN) の 2 つのチャネルがあります。 詳しくは、チャネルを参照してください。

メッセージ・チャネルのそれぞれの側の定義は、次のタイプのいずれかです。

- 送信側 (SDR)
- 受信側 (RCVR)
- サーバー (SVR)
- 要求側 (RQSTR)
- クラスター送信側 (CLUSSDR)
- クラスター受信側 (CLUSRCVR)

メッセージ・チャネルは、一方の側で定義されたタイプと、他方の側の互換性のあるタイプを使用して定義されます。 可能な組み合わせは以下のとおりです。

- 送信側と受信側
- 要求側とサーバー
- 要求側と送信側 (コールバック)
- サーバーと受信側
- クラスター送信側とクラスター受信側

送受信チャネル作成の詳しい手順は、チャネルの定義に記載されています。 送受信チャネルのセットアップに必要なパラメーターの例は、ご使用のプラットフォーム用の構成情報の例を参照してください。 チャネルの定義 (タイプを問わず) に必要なパラメーターについては、DEFINE CHANNEL を参照してください。

送信側 - 受信側チャネル

あるシステムで送信側がチャネルを開始すると、他のシステムにメッセージを送信できるようになります。 送信側は、チャネルのもう一方の側の受信側に開始を要求します。 送信側は伝送キューから受信側にメッセージを送信します。 受信側はメッセージを宛先キューに書き込みます。 これは、48 ページの図 8 に示されています。

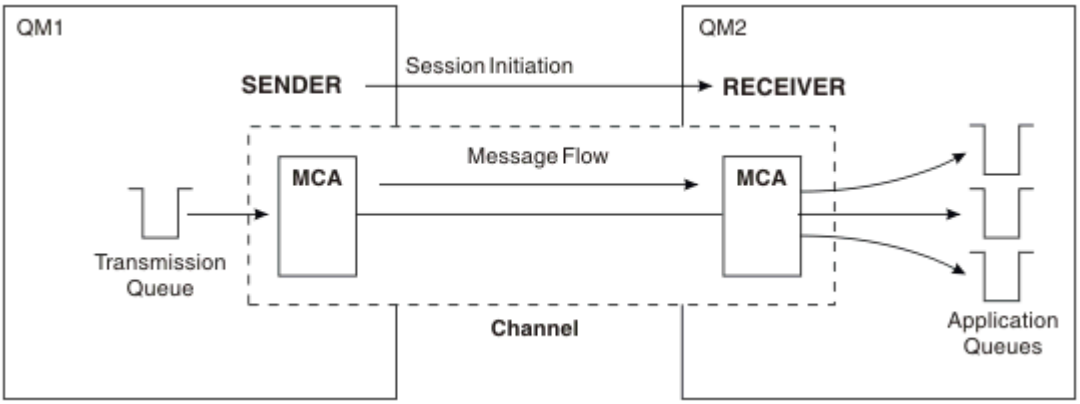


図 *8.* 送信側チャネルと受信側チャネル

要求側 - サーバー・チャネル

あるシステムの要求側がチャネルを開始すると別のシステムからのメッセージを受信できるようになります。 要求側は、チャネルのもう一方の側のサーバーに開始を要求します。 サーバーは、そのチャネル定義に定義された伝送キューから要求側にメッセージを送信します。

サーバー・チャネルは、通信を開始して要求側にメッセージを送信することもできます。 これは完全修飾サーバー、すなわちチャネル定義で指定されたパートナーの接続名を持つサーバー・チャネルにのみ適用されます。 要求側が完全修飾サーバーを開始することができますが、完全修飾サーバーが要求側との通信を開始することもできます。

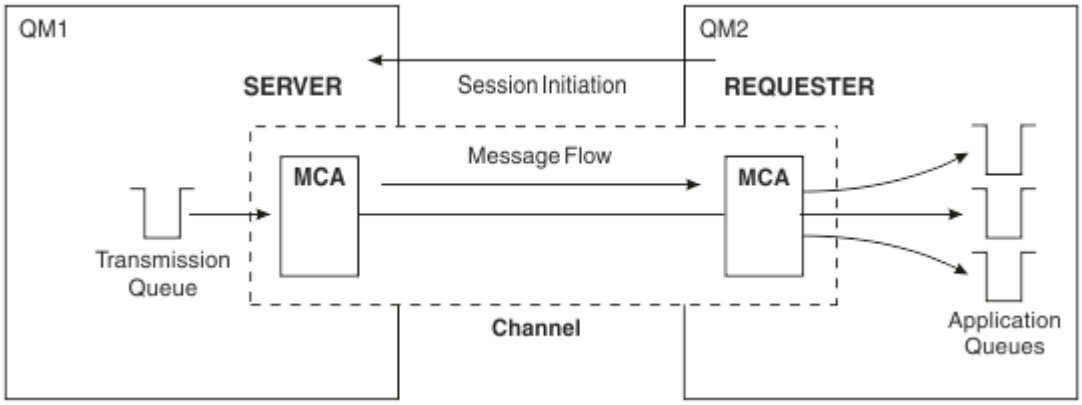


図 *9.* 要求側チャネルとサーバー・チャネル

要求側 - 送信側チャネル

要求側はチャネルを開始し、送信側は呼び出しを終了します。 次に、送信側はチャネル定義に入っている情報によって通信を再開します (コールバック と呼ばれます)。 その後、伝送キューから要求側にメッセージを送ります。

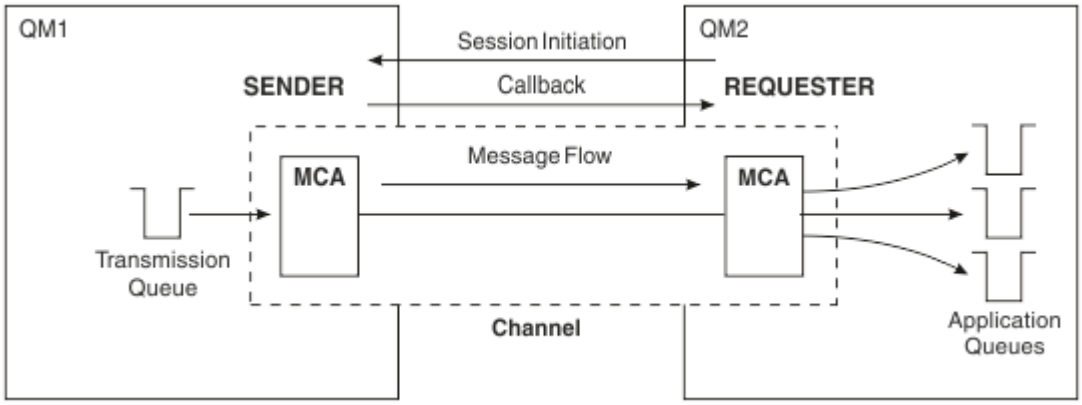


図 *10.* 要求側チャネルと送信側チャネル

サーバー・チャネルと受信側チャネル

これは、送信側チャネルと受信側チャネルに似ていますが、完全修飾 サーバー、すなわちチャネル定義で指定されたパートナーの接続名をもつサーバー・チャネルにだけ適用されます。 チャネルは、リンクのサーバー側で開始する必要があります。 次の図は、48 ページの図 8 に似ています。

クラスター送信側チャネル

クラスター内では、それぞれのキュー・マネージャーは全リポジトリー・キュー・マネージャーの 1 つに情報を送信するクラスター送信側チャネルを 1 つずつ備えています。 キュー・マネージャーは、クラスター送信側チャネル上の別のキュー・マネージャーへメッセージを送信することもできます。

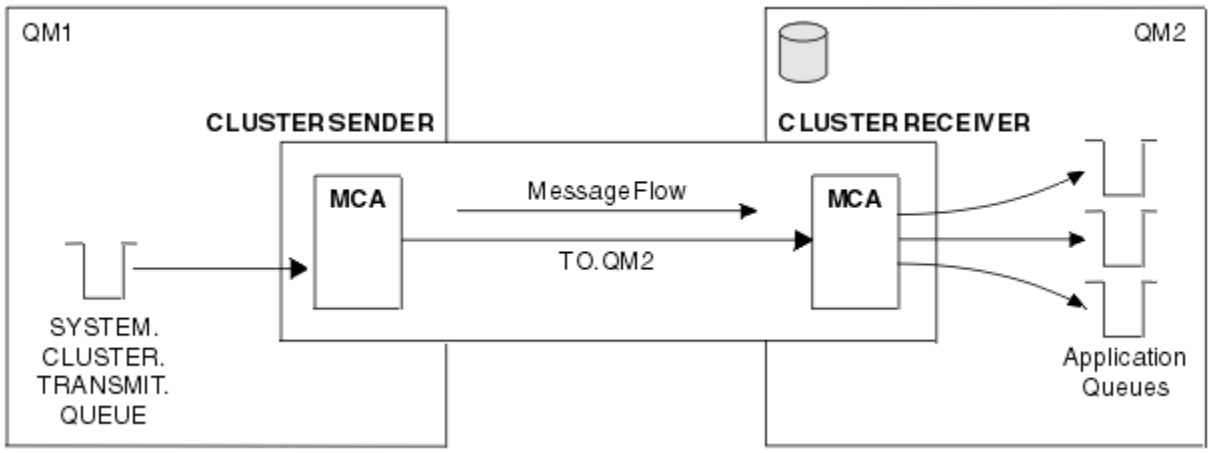


図 *11.* クラスター送信側チャネル

クラスター受信側チャネル

クラスター内の各キュー・マネージャーは、そのクラスターに関する情報とメッセージを受信するクラスター受信側チャネルを 1 つずつ備えています。 次の図は、50 ページの図 11 に似ています。

送達不能キュー

送達不能キュー (または未配布メッセージ・キュー) は、正しい宛先にメッセージを送信できない場合にそのメッセージが送信されるキューです。 一般的に、送達不能キューはキュー・マネージャーごとに 1 つ存在します。

送達不能キュー (DLQ) とは、例えば、宛先キューが存在しなかったり満杯だったりしてメッセージを宛先キューに送達できない場合に、メッセージを入れる保留キューのことで、未配布メッセージ・キュー とも言われます。 送達不能キューは、チャネルの送信側のデータ変換エラーの場合にも使用されます。 一般的には、ネットワーク内のすべてのキュー・マネージャーが、正しい宛先に送達できないメッセージを保管して後で取り出せるように、送達不能キューとして使用するローカル・キューを持っています。

キュー・マネージャー、メッセージ・チャネル・エージェント (MCA)、およびアプリケーションは、メッセージを DLQ に書き込むことができます。 DLQ 上のすべてのメッセージの先頭には、送達不能ヘッダー 構造体 MQDLH を付ける必要があります。 MQDLH 構造体の *Reason* フィールドには、メッセージが DLQ 上にある理由を識別する理由コードが入ります。

一般的には、キュー・マネージャーごとに送達不能キューを定義する必要があります。 定義しない場合、MCA はメッセージを書き込むことができず、伝送キューに残ったままとなり、チャネルは停止してしまいます。 また、高速の非持続メッセージ (Fast, nonpersistent messages を参照) を送達できず、ターゲット・システム上に送達不能キューが存在しない場合、これらのメッセージは破棄されます。

ただし、送達不能キューを使用すると、メッセージが送達される順序に影響するので、これらを使用しなくてもかまいません。

関連タスク

送達不能キューの取り扱い

未配布メッセージのトラブルシューティング

関連資料

runmqdlq (送達不能キュー・ハンドラーの実行)

リモート・キュー定義

リモート・キュー定義とは、別のキュー・マネージャーに所有されるキューの定義です。

アプリケーションはローカル・キューからしかメッセージを取り出せませんが、ローカル・キューまたはリモート・キューのどちらにもメッセージを書き込むことができます。 したがって、キュー・マネージャーには、ローカル・キューごとの定義だけでなくリモート・キュー定義 も備えていることがあります。 リ

モート・キュー定義の利点は、リモート・キューやリモート・キュー・マネージャーの名前、または伝送キューの名前を指定しなくても、アプリケーションがリモート・キューにメッセージを書き込めることです。 リモート・キュー定義を使用すると、ロケーションに依存しなくて済みます。

リモート・キュー定義は、他にも使用できますが、それについては後述します。

リモート・キュー・マネージャーへのアクセス方法

それぞれの発信元キュー・マネージャーとターゲット・キュー・マネージャーをつなぐチャネルは常に 1 つとは限りません。 2 つをつなぐいくつかの別の方法として、マルチ・ホップ、チャネルの共用、個別のチャネルおよびクラスタリングの使用などがあります。

マルチ・ホップ

ソース・キュー・マネージャーとターゲット・キュー・マネージャーをつなぐ直接の通信リンクがない場合、ターゲット・キュー・マネージャーへの経路の途中で 1 つまたは複数の中間キュー・マネージャー を介して伝送することができます。 これは、マルチ・ホップ と呼ばれます。

中間キュー・マネージャーでは、すべてのキュー・マネージャー同士、および伝送キュー同士をつなぐチャネルを定義する必要があります。 これについては、51 ページの図 12 に示してあります。

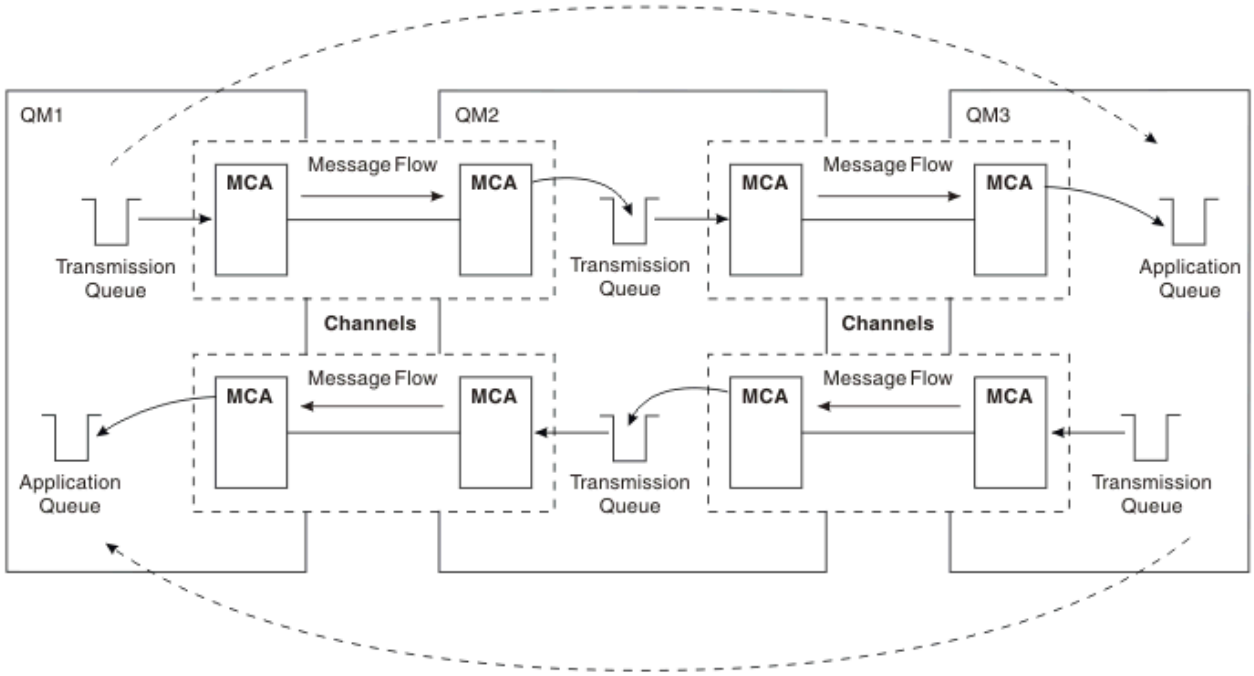


図 *12.* 中間キュー・マネージャーを介した伝送

チャネルの共用

アプリケーションを設計する場合は、キュー名と共にリモート・キュー・マネージャー名をアプリケーションに指定させるか、各リモート・キューごとにリモート・キュー定義 を作成するかを選択できます。 この定義には、リモート・キュー・マネージャー名、キュー名、および伝送キューの名前が入っています。 どちらの方法を使用しても、同じリモート・ロケーションのキューをアドレッシングするすべてのアプリケーションからのすべてのメッセージは、同じ伝送キューを介して送られます。 これについては、52 ページの図 13 に示してあります。

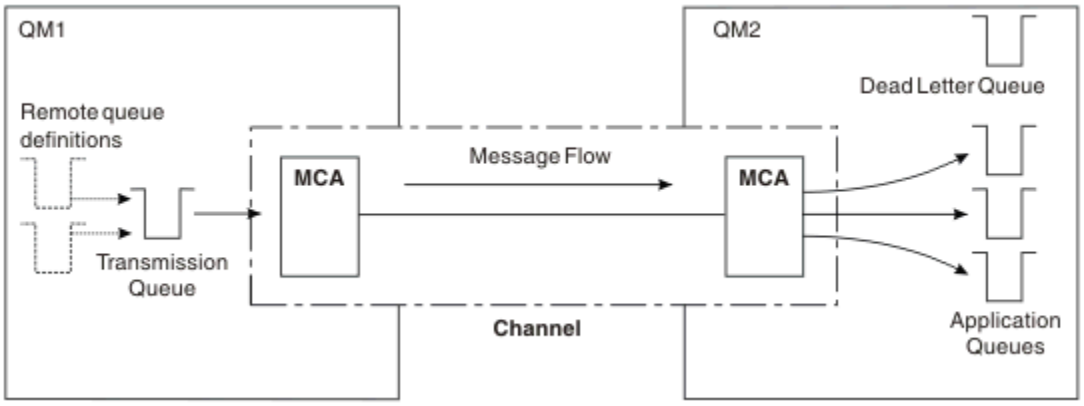


図 *13.* 伝送キューの共用

52 ページの図 13 は、複数のアプリケーションから同じチャネルを使用できる複数のリモート・キューへのメッセージの伝達を表しています。

個別のチャネルの使用

2 つのキュー・マネージャー間で異なるタイプのメッセージを送信する場合、その 2 つのキュー・マネージャー間に複数のチャネルを定義することができます。 セキュリティーのため、あるいは配送速度を犠牲にしてもメッセージ・トラフィックを一括させるために、代替チャネルが必要な場合があります。

2 番目のチャネルをセットアップするには、別のチャネルと伝送キューを定義し、ロケーションと伝送キュー名を指定するリモート・キュー定義を作成する必要があります。 これらを定義した後、アプリケーションではどちらのチャネルも使用できますが、メッセージは依然として同じターゲット・キューに送達されます。 これについては、52 ページの図 14 に示してあります。

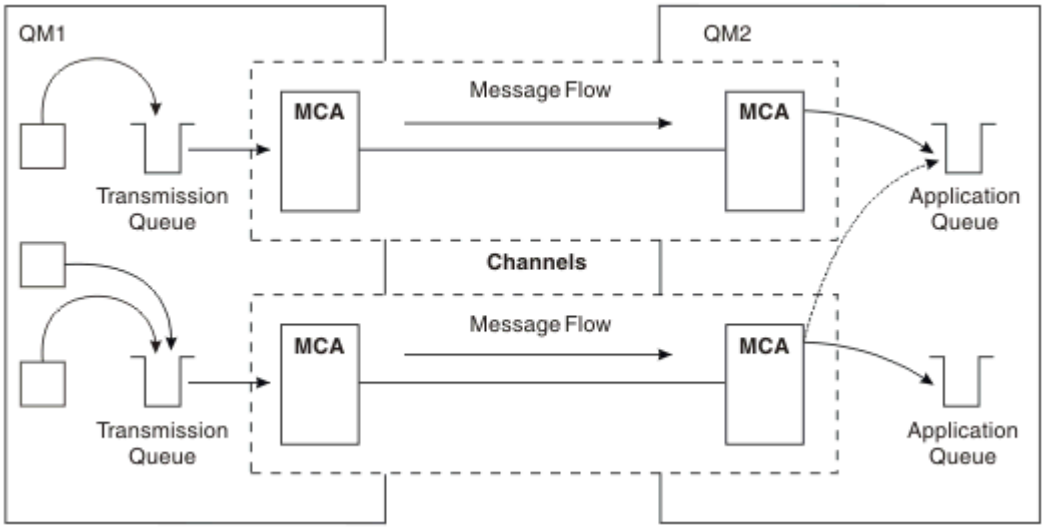


図 *14.* 複数チャネルの使用方法

伝送キューを指定するのにリモート・キュー定義を使用する場合、アプリケーションではロケーション (つまり、ターゲット・キュー・マネージャー) を**指定してはなりません**。 指定した場合、キュー・マネージャーはリモート・キュー定義を使用しません。 リモート・キュー定義を使用すると、ロケーションに依存しなくて済みます。 アプリケーションは、このキューのロケーションを知らなくても、メッセージを論理 キューに書き込むことができます。したがって、アプリケーションを変更しなくても、物理 キューを変更することができます。

クラスターの使用

クラスター内のすべてのキュー・マネージャーは、クラスター受信側チャネルを定義します。 別のキュー・マネージャーは、そのキュー・マネージャーへメッセージを送信するときに自動的にクラスター送信側チャネルを定義します。 例えば、あるキューのインスタンスがクラスター内に 2 つ以上存在する場合、そのキューのホストとなるどのキュー・マネージャーに対してもクラスター送信側チャネルを定義できます。 IBM MQ では、メッセージの経路を指定するのに使えるキュー・マネージャーを選択するため、ラウンドロビン・ルーチンを使用する作業負荷管理アルゴリズムが使用されます。 詳細については、クラスターを参照してください。

アドレッシング情報

アプリケーションがリモート・キュー・マネージャーに定義されているメッセージを書き込むとき、それらを伝送キューに入れる前に、ローカル・キュー・マネージャーは伝送ヘッダーを追加します。 このヘッダーには、宛先キューとキュー・マネージャーの名前、すなわちアドレッシング情報 が収められています。

単一キュー・マネージャー環境では、宛先キューのアドレスは、メッセージを書き込むキューをアプリケーションがオープンしたときに設定されます。 宛先キューは同じキュー・マネージャー上にあるため、アドレッシングのための情報は必要ありません。

分散キューイング環境では、キュー・マネージャーは、宛先キューの名前だけでなくそのキューのロケーション (すなわち、キュー・マネージャー名)、およびそのリモート・ロケーションへの経路 (すなわち、伝送キュー) を知る必要があります。 このアドレッシング情報は、伝送ヘッダーに含まれています。 受信チャネルは、伝送ヘッダーを取り外して、その中の情報を使用して宛先キューの場所を見つけます。

リモート・キュー定義を使用すると、宛先キュー・マネージャーの名前をアプリケーションが指定する必要がなくなります。 この定義は、リモート・キューの名前、メッセージが送られるリモート・キュー・マネージャーの名前、およびメッセージの移送に使用される伝送キューの名前を指定します。

別名について

別名を使用すれば、メッセージ・サービスの質が向上します。 キュー・マネージャーの別名を使用すると、システム管理者はアプリケーションを変更しないでターゲット・キュー・マネージャーの名前を変更することができます。 また、システム管理者は、ターゲット・キュー・マネージャーへの経路を変更したり、多数の他のキュー・マネージャーを介する (マルチ・ホップ) 経路を設定したりできます。 応答キューに別名が付けられるため、サービスが使いやすくなります。

キュー・マネージャーの別名および応答キューの別名は、ブランクの RNAME の入ったリモート・キュー定義を使用して作成されます。 これらの定義は、実際のキューを定義するのではありません。これらの定義は、物理キュー名、キュー・マネージャー名、および伝送キューを解決するためにキュー・マネージャーが使用します。

別名定義はブランクの RNAME を備えているという特徴があります。

キュー名の解決

キュー名の解決は、キューがオープンされるごとにすべてのキュー・マネージャーにおいて生じます。 その目的は、ターゲット・キュー、ターゲット・キュー・マネージャー (ローカルの場合もあります)、およびそのキュー・マネージャーへの経路 (ヌルの場合もあります) の指定です。 解決された名前は、キュー・マネージャー名、キュー名、および伝送キュー (キュー・マネージャーがリモートの場合) の 3 つの部分から構成されます。

リモート・キュー定義がある場合、別名定義は参照されません。 アプリケーションから提供されたキュー名は、ターゲット・キューの名前、リモート・キュー・マネージャー、およびリモート・キュー定義に指定される伝送キューに解決されます。 キュー名の解決についての詳細は、キュー名の解決を参照してください。

リモート・キュー定義がなく、キュー・マネージャーが指定されているか、または名前サービスによって解決されている場合、キュー・マネージャーは、システムに提供されたキュー・マネージャー名と一致するキュー・マネージャー別名定義があるかどうかを探します。 その定義がある場合は、その中の情報を使用して、ターゲット・キュー・マネージャーの名前へのキュー・マネージャーを解決します。 キュー・マ

ネージャー別名定義は、ターゲット・キュー・マネージャーへの伝送キューを決定するために使用することもできます。

解決されたキュー名がローカル・キューではない場合、アプリケーションが伝送キューに入れる各メッセージの伝送ヘッダーには、キュー・マネージャー名とキュー名の両方が収められます。

リモート・キュー定義またはキュー・マネージャー別名定義によって変更されない限り、通常、使用される伝送キューには、解決されたキュー・マネージャーの名前が付けられます。 そのような伝送キューを定義していなくても、デフォルト伝送キューを定義していれば、そのキューが使われます。

z/OS z/OS で実行されるキュー・マネージャーの名前は 4 文字までに制限されています。

キュー・マネージャー別名定義

メッセージを書き込むためにキューをオープンするアプリケーションがキュー名、**および**キュー・マネージャー名を指定するとき、キュー・マネージャー別名定義が適用されます。

キュー・マネージャー別名の定義には、次の 3 つの使用法があります。

- メッセージを送信し、キュー・マネージャー名を再マップするとき
- メッセージを送信し、伝送キューを変更または指定するとき
- メッセージを受信し、ローカル・キュー・マネージャーがこれらのメッセージの宛先を指しているかどうか判定するとき

アウトバウンド・メッセージ - キュー・マネージャー名の再マップ

キュー・マネージャー別名定義は、MQOPEN 呼び出しに指定されるキュー・マネージャー名を再マップするために使用できます。 例えば、MQOPEN 呼び出しは、THISQ のキュー名と YOURQM のキュー・マネージャー名を指定します。 ローカル・キュー・マネージャーには、次の例のようなキュー・マネージャー別名定義があります。

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

これは、アプリケーションがキュー・マネージャー YOURQM にメッセージを書き込むときに、使用される実際のキュー・マネージャーが REALQM であることを示しています。 ローカル・キュー・マネージャーが REALQM の場合、メッセージはローカル・キューの、キュー THISQ に書き込まれます。 ローカル・キュー・マネージャーが REALQM ではない場合、メッセージは REALQM という伝送キューにルーティングされます。 キュー・マネージャーは、伝送ヘッダーを変更して YOURQM の代わりに REALQM を指定します。

アウトバウンド・メッセージ - 伝送キューの変更および指定

55 ページの図 15 は、キュー・マネージャー QM3 でのキュー名を示す伝送ヘッダーを持ったメッセージが、キュー・マネージャー QM1 に到着するシナリオを示しています。 このシナリオでは、QM3 へは QM2 を介するマルチ・ホップを使用して到達できます。

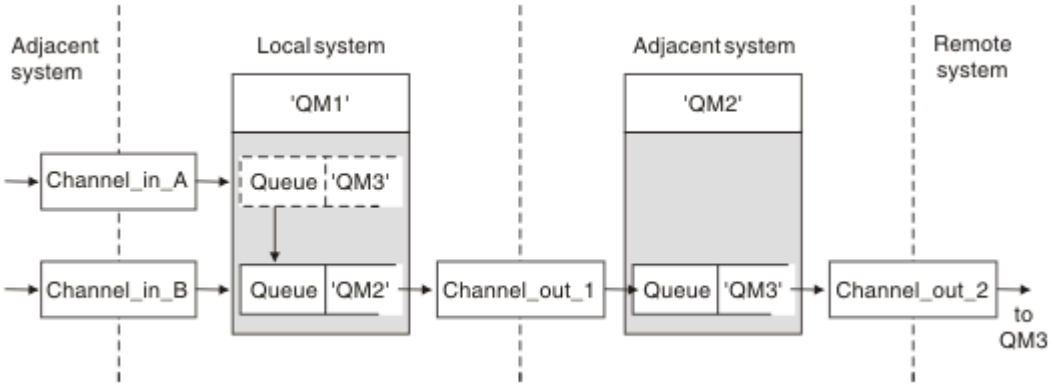


図 *15.* キュー・マネージャー別名

QM3 へのすべてのメッセージは、キュー・マネージャー別名を使用して QM1 で取り込まれます。 キュー・マネージャーの別名は QM3 であり、これには、伝送キュー QM2 を介した定義 QM3 が含まれています。 定義は、以下の例のようになります。

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

キュー・マネージャーはメッセージを伝送キュー QM2 に書き込みますが、宛先キュー・マネージャーの名前 QM3 が変わらないため、伝送キューのヘッダーを変更しません。

QM2 でのキュー名の入った伝送ヘッダーを示すメッセージが QM1 に着信すると、すべて QM2 伝送キューにも書き込まれます。 このようにして、宛先の異なるメッセージが共通の伝送キューに収集され、該当の隣接システムに送られて、さらに宛先に転送されます。

インバウンド・メッセージ - 宛先の決定

受信側 MCA は、伝送ヘッダーで参照されるキューをオープンします。 参照されるキュー・マネージャーと同じ名前のキュー・マネージャー別名定義が存在する場合、伝送ヘッダーに受信されるキュー・マネージャー名は、その定義からの RQMNAME と置き換わります。

このプロセスには、次の 2 つの使用法があります。

- 別のキュー・マネージャーにメッセージを送信する
- キュー・マネージャー名をローカル・キュー・マネージャーと同じ名前に変更する

応答先キュー別名の定義

応答先キューの別名定義は、メッセージ記述子の応答情報の代替名を指定します。 この利点は、アプリケーションを変更しないでキューまたはキュー・マネージャーの名前を変更できることです。

キュー名の解決

アプリケーションは、メッセージに応答するとき、受信したメッセージのメッセージ記述子 内のデータを使用して、応答先のキューの名前を確かめます。 送信側アプリケーションは、応答が送信される場所を示し、この情報をそのメッセージに付加します。 この概念は、アプリケーション設計時に調整する必要があります。

キュー名の解決は、メッセージがキューに書き込まれる前にアプリケーションの送信側で行われます。 キュー名の解決は、メッセージの送信先のリモート・アプリケーションと対話する前に生じます。 これは、キューがオープンされていない時点でネーム・レゾリューションが実行される唯一の状況です。

キュー・マネージャー別名を使用するキュー名の解決

通常、アプリケーションは、応答先キューを指定し、応答先キュー・マネージャー名をブランクにしておきます。 キュー・マネージャーは、書き込み時にその固有の名前を書き込みます。 この方式が唯一首尾よくいかないのは、例えば、伝送キュー QM1 を使用するデフォルト返送チャネルの代わりに、伝送キュー QM1_relief を使うチャネルなどの、代替チャネルを応答で使用したい場合です。 この状況では、伝送キューのヘッダーに指定されるキュー・マネージャー名は「実際の」キュー・マネージャー名と一致しませんが、キュー・マネージャーの別名定義を使用して再定義されます。 代替経路で応答を返すには、応答先キュー別名定義を使用して、応答先キュー・データも同様にマップする必要があります。

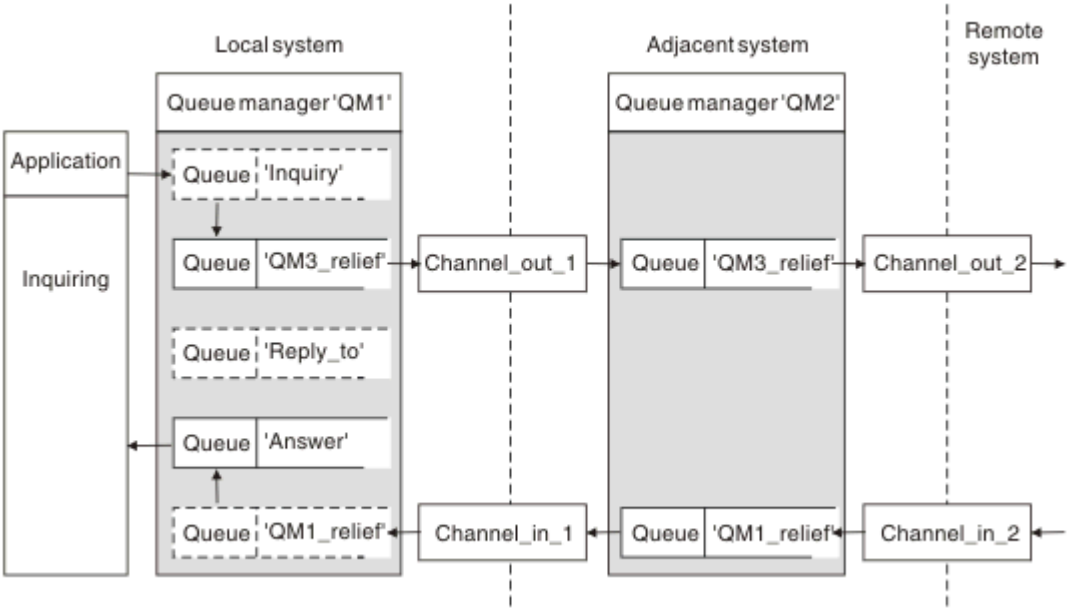


図 16. 応答ロケーションの変更に使用される応答先キューの別名

56 ページの図 16 の例の説明:

1. アプリケーションは、MQPUT 呼び出しを使用し、メッセージ記述子に以下のような情報を指定して、メッセージを書き込みます。

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

   ReplyToQMgr は、使用される応答先キュー別名用で、ブランクにしておく必要があります。
2. Answer という名前と QM1_relief というキュー・マネージャー名の付いた、応答先キュー別名定義 Reply_to を作成します。

   ```
   DEFINE QREMOTE ('Reply_to') RNAME ('Answer')
   RQMNAME ('QM1_relief')
   ```

3. メッセージは、ReplyToQ='Answer' および ReplyToQMgr='QM1_relief' を示すメッセージ記述子付きで送信されます。
4. アプリケーションの仕様には、応答が Reply_to ではなくキュー Answer になければならないという情報が必要です。

応答に備えるには、以下のように定義をして、並列戻りチャネルを作成する必要があります。

- QM2 で、伝送キュー定義 QM1_relief

```
DEFINE QLOCAL ('QM1_relief') USAGE(XMITQ)
```

- QM1 で、キュー・マネージャー別名 QM1_relief

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

  このキュー・マネージャー別名は、並列戻りチャネルのチェーンを終了させ、QM1 へのメッセージを取り込みます。

実際の運用が将来の場合でも、アプリケーションが最初から別名を使用することを確認してください。 現在は、これは応答先キューへの通常のキュー別名ですが、あとでキュー・マネージャー別名に変更することができます。

応答先キュー名

応答先キューの命名には注意が必要です。 アプリケーションが応答先キュー名をメッセージに組み込むのは、応答の送り先となるキューを指定できるようにするためです。 この名前を使用して応答先キュー別名定義を作成すると、実際の応答先キュー (すなわち、ローカル・キュー定義) に同じ名前を使用できなくなります。 したがって、応答先キュー別名定義には、キュー・マネージャー名の他に新しいキュー名を入れる必要があり、また、この別のキューに応答が入ることを示す情報がアプリケーションの仕様に入っていなければなりません。

こうして、アプリケーションは、元のメッセージを書き込むときに応答先キューとして指定したキューとは別のキューからメッセージを取り出す必要があります。

クラスターのコンポーネント

クラスターは、キュー・マネージャー、クラスター・リポジトリー、クラスター・チャネル、およびクラスター・キューで構成されます。

各クラスター・コンポーネントについて詳しくは、以下のサブトピックを参照してください。

関連概念
クラスター化と分散キューイングとの比較
関連タスク
キュー・マネージャー・クラスターの構成
新規クラスターのセットアップ

クラスター・リポジトリー

リポジトリーとは、クラスターを構成する各キュー・マネージャーについての情報の集まりを指します。

リポジトリー情報には、キュー・マネージャーの名前、場所、チャネル、そのキュー・マネージャーがホストするキュー、および他の情報が含まれます。 この情報は、SYSTEM.CLUSTER.REPOSITORY.QUEUE というキューにメッセージ形式で格納されています。 このキューは、デフォルト・オブジェクトの 1 つです。 Multi マルチプラットフォーム では、IBM MQ キュー・マネージャーを作成するときに定義されます。 z/OS IBM MQ for z/OS では、キュー・マネージャーのカスタマイズの一部として定義されます。

完全リポジトリーおよび部分リポジトリー

通常は、クラスターの中の 2 つのキュー・マネージャーに完全リポジトリーが保持されます。 それ以外のすべてのキュー・マネージャーには、部分リポジトリー が保持されます。

クラスター内の各キュー・マネージャーについての 全情報をホストしているキュー・マネージャーには、完全リポジトリーが含まれます。 クラスター内の他のキュー・マネージャーには、完全リポジトリーに格納されている情報のサブセットが入っている部分リポジトリーが含まれます。

部分リポジトリーには、メッセージを交換する必要があるキュー・マネージャーに関する情報だけが格納されています。 キュー・マネージャーが必要な情報に対する更新情報を要求すると、その情報が変更されている場合には、完全リポジトリー・キュー・マネージャーは要求元のキュー・マネージャーに新しい情報を送信します。 通常、 部分リポジトリーには、キュー・マネージャーがクラスター内で処理を実行するのに 必要な情報がすべて含まれています。 その他の情報が必要になると、キュー・マネージャーは完全リポジトリーにその情報を照会し、これにより、部分リポジトリーを更新します。 キュー・マネージャーは SYSTEM.CLUSTER.COMMAND.QUEUE キューを使用して、リポジトリーに対する更新の情報を要求し、これを受信します。

クラスターのメンバーであるキュー・マネージャーをマイグレーション する場合は、部分リポジトリーをマイグレーションする前に完全リポジトリーをマイグレーションしてください。 これは、新しいリリースで導入された新しい属性は、古いリポジトリーには格納できないためです。 これらは、許容されますが、格納されません。

クラスター・キュー・マネージャー

クラスター・キュー・マネージャーは、クラスターのメンバーであるキュー・マネージャーです。

キュー・マネージャーは、複数のクラスターのメンバーにすることができます。 各クラスター・キュー・マネージャーには、そのクラスターがメンバーとなっているすべてのクラスター全体で固有の名前がなければなりません。

1 つのクラスター・キュー・マネージャーで複数のキューをホストすることができます。 その場合、そのキュー・マネージャーはそれらのキューを同じクラスター内にあるその他のキュー・マネージャーに通知します。 ただし、これを行う必要はありません。 クラスター・キュー・マネージャーは、代わりにメッセージをクラスター内でホストされるキューに送信し、そのメッセージに対する応答のうち、そこに送るよう明示的に指定された応答だけを受信できます。

z/OS IBM MQ for z/OS では、クラスター・キュー・マネージャーはキュー共有グループのメンバーになることができます。 この場合、キュー・マネージャーは同じキュー共有グループ内の他のキュー・マネージャーとキュー定義を共有します。

クラスター・キュー・マネージャーは自律型のプログラムです。 各クラスター・キュー・マネージャーに定義されたキューおよびチャネルをそのキュー・マネージャーが完全に制御することができます。 各キュー・マネージャーの定義を別のキュー・マネージャーによって変更することはできません (同じキュー共有グループ内のキュー・マネージャーであれば可能です)。 リポジトリー・キュー・マネージャーは、クラスター内の他のキュー・マネージャーでの定義を制御しません。 リポジトリー・キュー・マネージャーは、必要に応じて使用できる、すべての定義の完全なセットを保持します。 クラスターは、キュー・マネージャーのフェデレーションです。

クラスター・キュー・マネージャーの定義を作成したり変更したりすると、その情報は完全リポジトリー・キュー・マネージャーに送信されます。 クラスター内の他のリポジトリーは後で更新されます。

完全リポジトリー・キュー・マネージャー

完全リポジトリー・キュー・マネージャーとは、クラスターのリソースの完全な表現を保持するクラスター・キュー・マネージャーです。 可用性を確保するために、各クラスターに 2 つ以上の完全リポジトリー・キュー・マネージャーをセットアップしてください。 完全リポジトリー・キュー・マネージャーは、クラスター内にあるその他のキュー・マネージャーから送信された情報を受信し、そのリポジトリーを更新します。 そして、互いにメッセージを交換することにより、すべての完全リポジトリー・キュー・マネージャーでクラスターに関する情報が常に最新に保たれるようにします。

キュー・マネージャーおよびリポジトリー

すべてのクラスターには、クラスター内のキュー・マネージャー、キュー、およびチャネルに関する情報のフル・リポジトリーを保持する、少なくとも 1 つの (できれば 2 つの) キュー・マネージャーがあります。 また、クラスター内のその他のキュー・マネージャーからこの情報を更新する要求が発行された場合は、その要求も完全リポジトリーに格納されます。

他のキュー・マネージャーは、それぞれ部分リポジトリーを保持します。このリポジトリーには、通信するために必要なキューおよびキュー・マネージャーのサブセットに関する情報が含まれています。 これら

のキュー・マネージャーは、別のキューまたはキュー・マネージャーに初めてアクセスする必要が生じたときに照会を行うことによって、それ自身の部分リポジトリーを作成します。 キュー・マネージャーは、そのキューまたはキュー・マネージャーに関する新しい情報があれば通知するように要求します。

各キュー・マネージャーは、リポジトリー情報を SYSTEM.CLUSTER.REPOSITORY.QUEUE というキューにメッセージ形式で格納します。 そして、各キュー・マネージャー間でメッセージ形式のリポジトリー情報を交換するときには SYSTEM.CLUSTER.COMMAND.QUEUE というキューを使用します。

クラスターに参加する各キュー・マネージャーは、いずれかのリポジトリーに対するクラスター送信側 (CLUSSDR) チャネルを定義します。 キュー・マネージャーはクラスター内のどのキュー・マネージャーが完全リポジトリーを保有しているのかをただちに認識します。 これ以降、キュー・マネージャーはどのリポジトリーに格納されている情報でも要求することができます。 また、キュー・マネージャーが、指定されたリポジトリーに情報を送信すると、その情報は別のリポジトリーがあればそのリポジトリーにも送信されます。

完全リポジトリーをホストしているキュー・マネージャーが、そのプログラムに接続されている別のキュー・マネージャーから新しい情報を受信すると、完全リポジトリーが更新されます。 この新しい情報は別のリポジトリーにも送信されます。そのため、稼働していないリポジトリー・キュー・マネージャーがあると情報の更新に遅れが生じるという可能性も少なくなります。 どの情報も 2 回送信されるので、各リポジトリーでは重複する情報を破棄する必要があります。 各情報には順序番号が付いているので、各リポジトリーではその番号によって重複する情報を識別することができます。 すべてのリポジトリー間では、メッセージを交換することによって互いに同期をとります。

クラスター・キュー

クラスター・キューとは、クラスター・キュー・マネージャーでホストされ、同じクラスター内の別のキュー・マネージャーで使用できるキューです。

クラスター・キュー定義は、クラスター内の他のキュー・マネージャーに通知されます。 クラスター内にあるその他のキュー・マネージャーは、対応するリモート・キュー定義がなくても、クラスター・キューにメッセージを書き込むことができます。 クラスター名前リストを使用して、クラスター・キューを複数のクラスターに通知できます。

キューが通知されると、クラスター内のキュー・マネージャーはそのキューにメッセージを書き込めるようになります。 メッセージを書き込むときには、キュー・マネージャーが、フルリポジトリーで、そのキューがホストされている場所を調べる必要があります。 格納場所が分かったら、宛先情報をメッセージに追加して、クラスター伝送キューにメッセージを書き込みます。

z/OS クラスター・キューは、IBM MQ for z/OS でのキュー共用グループの メンバーによって共用されるキューにすることができます。

関連タスク

クラスター・キューの定義

共用キューとクラスター・キューの比較

ここでは、共用キューおよびクラスター・キューを比較し、ご使用のシステムにどちらが適切かを判断できるようにすることを目的としています。

チャネル・イニシエーターのコスト

クラスター・キューでは、メッセージはチャネルによって送信されるため、アプリケーションのコストに加えてチャネル・イニシエーターのコストを考慮に入れてください。 チャネルがメッセージを取得し、書き込むため、ネットワークでコストが発生します。 共有キューにはこれらのコストがありません。そのため、キュー共有グループ内のキュー・マネージャー間でメッセージを移動するとき、共有キューでは、クラスター・キューよりも処理能力の使用が少なくなります。

メッセージの可用性

クラスター・キューの場合、キューに書き込みを行うと、メッセージは、ご使用のキュー・マネージャーへのアクティブなチャネルが接続されているキュー・マネージャーの 1 つへ送信されます。 リモート・キ

ュー・マネージャーでは、メッセージを処理するために使用するアプリケーションが機能していない場合、メッセージは処理されず、アプリケーションが始動するまで待機します。 同様に、キュー・マネージャーがシャットダウンすると、そのキュー・マネージャー上のメッセージは、キュー・マネージャーが再始動するまで使用可能になりません。 これらの事例から、共用キューを使用する場合に比べてメッセージの可用性が低いことが分かります。

共用キューを使用している場合、キュー共用グループ内のどのアプリケーションも、送信されたメッセージを取得することができます。 キュー共用グループで 1 つのキュー・マネージャーをシャットダウンした場合、メッセージは他のキュー・マネージャーで使用することができるので、クラスター・キューを使用する場合に比べてメッセージの可用性が高くなります。

キャパシティー

カップリング・ファシリティーは、ディスクよりも高価です。それで、ローカル・キューに 1,000,000 メッセージを格納するためのコストは、同じ数のメッセージを十分な容量のカップリング・ファシリティーに格納するよりも低くなります。

他のキュー・マネージャーへの送信

共用キュー・メッセージは、キュー共用グループでのみ使用可能です。 キュー共用グループの外でキュー・マネージャーを使用する場合、チャネルを使用する必要があります。 複数のリモートの分散キュー・マネージャー間でワークロード・バランスをとるために、クラスタリングを使用することができます。

ワークロード・バランシング

クラスタリングを使用して、チャネルおよびキュー・マネージャーに対して、送信されたメッセージを取得する比率に重みづけをすることができます。 例えば、1 つのキュー・マネージャーにメッセージの 60% を送信し、別のキュー・マネージャーにメッセージの 40% を送信することができます。 この事例は、リモート・キュー・マネージャーのプロセス処理能力に左右されません。 最初のキュー・マネージャーのシステムは過負荷になる可能性があり、2 番目のキュー・マネージャーのシステムはアイドル状態になる可能性があります。しかし、メッセージの多くは、最初のキュー・マネージャーに依然として送信されます。

共用キューを使用すると、2 つの CICS® システムがメッセージを取得できます。 1 つのシステムが過負荷になると、もう一方のシステムが大部分の作業負荷を引き継ぎます。

クラスター・チャネル

すべてのフル・リポジトリーで、手動でクラスター受信側チャネルと、クラスター内の他のすべてのフル・リポジトリーに接続するクラスター送信側チャネルのセットを手動で定義します。 部分リポジトリーを追加する場合、クラスター受信側チャネルと、いずれかのフル・リポジトリーに接続する単一のクラスター送信側チャネルを手動で定義します。 その他のクラスター送信側チャネルは、必要になった時にクラスターにより自動的に定義されます。 自動的に定義されたクラスター送信側チャネルの属性は、受信側のキュー・マネージャーの対応するクラスター受信側チャネル定義の属性を基にして設定されます。

クラスター受信側チャネル: CLUSRCVR

CLUSRCVR チャネル定義は、クラスター・キュー・マネージャーが同じクラスター内にある他のキュー・マネージャーからメッセージを受信することができるチャネルの終端を定義します。

CLUSRCVR チャネルは、各クラスター・キュー・マネージャーに 1 つ以上定義しなければなりません。 キュー・マネージャーは、CLUSRCVR チャネルを定義することにより、そのキュー・マネージャーがメッセージを受信可能であることを他のクラスター・キュー・マネージャーに示します。

CLUSRCVR チャネル定義は、他のキュー・マネージャーが対応するクラスター送信側チャネル定義を自動的に定義できるようにもします。 この記事の 61 ページの『自動定義されるクラスター送信側チャネル』 セクションを参照してください。

クラスター送信側チャネル: CLUSSDR

クラスターの各完全リポジトリー・キュー・マネージャーから、他の各完全リポジトリー・キュー・マネージャーへの CLUSSDR チャネルを手動で定義します。 完全リポジトリー間で交換されるすべての更新情報は、このチャネルでのみ送受信されます。 これらのチャネルを手動で定義して、完全リポジトリーのネットワークを明示的に制御します。

部分リポジトリー・キュー・マネージャーをクラスター追加する場合、いずれかのフル・リポジトリーに接続する単一の CLUSSDR チャネルを手動で定義します。 どのフル・リポジトリーを選択してもあまり変わりはありません。それは、初期接続が確立されれば、キュー・マネージャーのクラスター・キュー・マネージャー・オブジェクトが、CLUSSDR チャネルを含めて、必要に応じて自動的に定義されるからです。このため、キュー・マネージャーは、フル・リポジトリーにクラスター情報を送信し、クラスター内の任意のキュー・マネージャーにメッセージを送信できます。

この記事の該当セクションで説明したように、自動定義される送信側チャネルは、クラスター受信側チャネルの構成に基づきます。 したがって、クラスター・チャネルで設定するチャネル・プロパティーは、対応する CLUSSDR チャネルとクラスター受信側チャネルで同一に設定するか、クラスター受信側チャネルのみに設定する必要があります。

前述した理由で、CLUSSDR チャネルだけは手動で定義しなければなりません。 つまり、最初に部分リポジトリーを完全リポジトリーに接続する場合か、2 つの完全リポジトリーを相互に接続する場合です。 部分リポジトリーやクラスターに含まれないキュー・マネージャーを接続する CLUSSDR チャネルを手動で構成すると、AMQ9427 や AMQ9428 などのエラー・メッセージが出される原因になります。 これは一時的状況として避けられない場合もありますが (例えば完全リポジトリーの位置を変更するとき)、手動定義はできるだけ早く削除する必要があります。

自動定義されるクラスター送信側チャネル

部分リポジトリー・キュー・マネージャーをクラスター追加する場合、通常はキュー・マネージャーで 2 つのクラスター・チャネルだけを手動で定義します。

- クラスターのフル・リポジトリー・キュー・マネージャーへのクラスター送信側 (CLUSSDR) チャネル。
- クラスター受信側 (CLUSRCVR) チャネル。

CLUSSDR チャネルを定義することにより、キュー・マネージャーはクラスターとの初期接続を確立できるようになります。 初期接続が確立された後、他の CLUSSDR チャネルは必要に応じてクラスターにより自動的に定義されます。

自動的に定義される CLUSSDR チャネルは、受信側のキュー・マネージャーの対応する CLUSRCVR チャネル定義から自分の属性を取得します。 手動で定義された CLUSSDR チャネルがある場合でも、自動定義された CLUSSDR チャネルからの属性が使用されます。 例えば、**CONNAME** パラメーターでポート番号を指定せずに CLUSRCVR チャネルを定義し、ポート番号を指定して CLUSSDR チャネルを手動で定義したとします。 手操作で定義した CLUSSDR チャネルが自動定義されたもので置き換えられると、ポート番号 (CLUSRCVR チャネルから設定された) はブランクになります。 デフォルトのポート番号が使用され、チャネルで障害が発生します。

手動で定義された CLUSSDR チャネルと、対応する CLUSRCVR チャネル定義の間に構成の違いがある場合、いくつかの違いは即時に有効になり (例えば、ワークロード・バランシングのパラメーター)、いくつかのものはチャネル再始動 (例えば、TLS 構成) したときにのみ有効になります。

混乱を避けるために、可能な限り、以下のガイドラインを順守してください。

- 手動で定義するのは、完全リポジトリーを指す CLUSSDR チャネルのみにします。
- 手動で定義された CLUSSDR チャネルがある場合、受信側のキュー・マネージャーの対応する CLUSRCVR チャネル定義と完全に一致するように構成します。

自動定義チャネルの処理も参照してください。

関連概念
自動定義チャネルの処理
クラスター伝送キューとクラスター送信側チャネルの操作

関連タスク
新規クラスターのセットアップ
クラスターにキュー・マネージャーを追加する

クラスター・トピック

クラスター・トピックは、 **cluster** 属性が定義されている管理トピックです。 クラスター・トピックに関する情報は、クラスターのすべてのメンバーにプッシュされ、ローカル・トピックと結合されて、複数のキュー・マネージャーにわたるトピック・スペースの一部を形成します。 これにより、あるトピックに対して 1 つのキュー・マネージャーでパブリッシュされたメッセージが、クラスター内の他のキュー・マネージャーのサブスクリプションに配信されます。

キュー・マネージャーにクラスター・トピックを定義すると、そのクラスター・トピック定義が完全リポジトリー・キュー・マネージャーに送信されます。 完全リポジトリーは、そのクラスター・トピック定義をクラスター内のすべてのキュー・マネージャーに伝搬し、クラスター内のあらゆるキュー・マネージャーで、同じクラスター・トピックがパブリッシャーおよびサブスクライバーに使用可能になるようにします。 クラスター・トピックを作成するキュー・マネージャーは、クラスター・トピック・ホストと呼ばれます。 クラスター・トピックはクラスター内の任意のキュー・マネージャーで使用できますが、クラスター・トピックに対する変更は、そのトピックが定義されているキュー・マネージャー (クラスター・トピック・ホスト) で行う必要があります。変更を行った時点で、その変更は完全リポジトリーを介してクラスターのすべてのメンバーに伝搬されます。

直接ルーティング またはトピック・ホスト・ルーティング を使用するためのクラスター・トピックの構成について、またクラスター・トピックの継承やワイルドカード・サブスクリプションについては、クラスター・トピックの定義を参照してください。

クラスター・トピックを表示するために使用するコマンドについては、関連情報を参照してください。

関連概念
管理トピックの操作
サブスクリプションの操作
関連資料
表示トピック
DISPLAYTPSTATUS
表示サブ

デフォルトのクラスター・オブジェクト

Multi Multiplatforms では、キュー・マネージャーを定義するときに自動的に作成されるデフォルト・オブジェクトのセットに、デフォルトのクラスター・オブジェクトが含まれています。 z/OS z/OS の場合、デフォルトのクラスター・オブジェクト定義は、カスタマイズ・サンプルにあります。

注: デフォルト・チャネル定義は、他のすべてのチャネル定義と同様に、MQSC または PCF コマンドを実行することで変更できます。 SYSTEM.CLUSTER.HISTORY.QUEUE を除き、デフォルトのキュー定義は変更しないでください。

SYSTEM.CLUSTER.COMMAND.QUEUE
クラスター内のそれぞれのキュー・マネージャーには、SYSTEM.CLUSTER.COMMAND.QUEUE という、メッセージを完全リポジトリーに転送するために使用するローカル・キューがあります。 メッセージには、キュー・マネージャーに関する新しい情報や変更された情報、また他のキュー・マネージャーに関する情報の要求が格納されます。 通常、SYSTEM.CLUSTER.COMMAND.QUEUE は空です。

SYSTEM.CLUSTER.HISTORY.QUEUE
クラスター内の各キュー・マネージャーには、SYSTEM.CLUSTER.HISTORY.QUEUE という名前のローカル・キューがあります。 SYSTEM.CLUSTER.HISTORY.QUEUE は、サービス目的でクラスター状態情報の履歴を保管するために使用されます。

デフォルトのオブジェクト設定では、SYSTEM.CLUSTER.HISTORY.QUEUE は PUT (ENABLED) に設定されます。 履歴収集を抑止するには、設定を PUT (DISABLED) に変更します。

SYSTEM.CLUSTER.REPOSITORY.QUEUE
クラスター内の各キュー・マネージャーには、SYSTEM.CLUSTER.REPOSITORY.QUEUE という名前のローカル・キューがあります。 このキューはすべての完全リポジトリー情報の保管に使用します。 このキューは通常は空ではありません。

SYSTEM.CLUSTER.TRANSMIT.QUEUE
各キュー・マネージャーには、SYSTEM.CLUSTER.TRANSMIT.QUEUE というローカル・キューの定義があります。 SYSTEM.CLUSTER.TRANSMIT.QUEUE は、クラスター内のすべてのキューおよびキュー・マネージャーに対するすべてのメッセージのデフォルト伝送キューです。 キュー・マネージャー属性 DEFCLXQ を変更することにより、各クラスター送信側チャネルのデフォルト伝送キューを SYSTEM.CLUSTER.TRANSMIT. *ChannelName* に変更できます。
SYSTEM.CLUSTER.TRANSMIT.QUEUE を削除することはできません。 また、使用されるデフォルト伝送キューが SYSTEM.CLUSTER.TRANSMIT.QUEUE であるか SYSTEM.CLUSTER.TRANSMIT. *ChannelName* であるかの許可検査を定義するためにも使用されます。

SYSTEM.DEF.CLUSRCVR
それぞれのクラスターには、SYSTEM.DEF.CLUSRCVR というデフォルトの CLUSRCVR チャネル定義があります。 SYSTEM.DEF.CLUSRCVR は、クラスター内のキュー・マネージャーにクラスター受信側チャネルを作成するときに指定しないすべての属性にデフォルト値を提供するために使用されます。

SYSTEM.DEF.CLUSSDR
それぞれのクラスターには、SYSTEM.DEF.CLUSSDR というデフォルトの CLUSSDR チャネル定義があります。 SYSTEM.DEF.CLUSSDR は、クラスター内のキュー・マネージャーにクラスター送信側チャネルを作成するときに指定しないすべての属性にデフォルト値を提供するために使用されます。

関連概念
デフォルト・クラスター・オブジェクトの処理

パブリッシュ/サブスクライブ・メッセージング

パブリッシュ/サブスクライブ・メッセージングによって、情報の提供者をその情報の利用者から分離することができます。 送信側および受信側アプリケーションは、情報を送受信するために互いの情報を知っている必要はありません。

Point-to-Point IBM MQ アプリケーションが別のアプリケーションにメッセージを送信できるようにするためには、まずそのアプリケーションについて知る必要があります。 例えば、情報の送信先のキューの名前がわかっていなければなりませんし、場合によってはキュー・マネージャー名を指定する必要もあります。

IBM MQ のパブリッシュ/サブスクライブでは、ご使用のアプリケーションがターゲット・アプリケーションについて何も知る必要はありません。 送信側アプリケーションが行う必要があるのは、以下の処理だけです。

- アプリケーションが必要とする情報を含む IBM MQ メッセージを 書き込み ます。
- 情報のサブジェクトを示すトピックへのこのメッセージの割り当て。
- IBM MQ によるその情報の配布処理。

同様に、ターゲット・アプリケーションも、受け取る情報のソースについて何も知る必要はありません。

以下の図は、最も単純なパブリッシュ/サブスクライブ・システムを示しています。 パブリッシャーが 1 つ、キュー・マネージャーが 1 つ、サブスクライバーが 1 つあります。 サブスクライバーがサブスクリプションをキュー・マネージャー上に作成し、パブリッシャーがパブリケーションをキュー・マネージャーに送信します。 そのパブリケーションを、キュー・マネージャーがサブスクライバーに転送します。

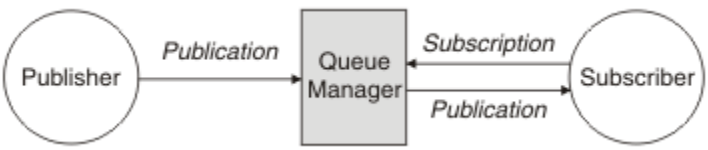


図 *17.* 簡単なパブリッシュ/サブスクライブの構成

標準的なパブリッシュ/サブスクライブ・システムには、さまざまなトピックに関する複数のパブリッシャーと複数のサブスクライバーがあり、多くの場合、キュー・マネージャーも複数あります。 アプリケーションは、パブリッシャーとサブスクライバーの両方にすることができます。

パブリッシュ/サブスクライブ・メッセージングと Point-to-Point のもう 1 つの重要な相違点は、Point-to-Point キューに送信されたメッセージを処理するのは、単一のコンシューム・アプリケーションのみであるという点です。 複数のサブスクライバーがインタレストを登録しているパブリッシュ/サブスクライブ・トピックにパブリッシュされたメッセージは、すべてのインタレスト・サブスクライバーによって処理されます。

パブリッシュ/サブスクライブの構成要素

パブリッシュ/サブスクライブは、サブスクライバーがパブリッシャーから情報をメッセージの形で受け取るためのメカニズムです。 パブリッシャーとサブスクライバーの間の相互作用は、IBM MQ の標準機能を使用して、キュー・マネージャーによって制御されます。

標準的なパブリッシュ/サブスクライブ・システムには、さまざまなトピックに関する複数のパブリッシャーと複数のサブスクライバーがあり、多くの場合、キュー・マネージャーも複数あります。 アプリケーションは、パブリッシャーとサブスクライバーの両方にすることができます。

情報のプロバイダーをパブリッシャー といいます。 パブリッシャーは主題に関する情報を提供しますが、その情報に関心のあるアプリケーションのことは何も知る必要はありません。 パブリッシャーは、その情報をパブリケーション というメッセージの形で生成します。このようなメッセージのトピックのパブリッシュと定義は、パブリケーションによって行われます。

情報のコンシューマーをサブスクライバー といいます。 サブスクライバーは、サブスクライバーが対象とするトピックを示したサブスクリプション を作成します。 したがって、どのパブリケーションがサブスクライバーに転送されるかは、サブスクリプションで決まります。 サブスクライバーは複数のサブスクリプションを行え、さまざまなパブリッシャーから情報を受け取ることができます。

パブリッシュされた情報は IBM MQ メッセージで送られ、情報の主題はそのトピック で識別されます。 パブリッシャーは情報をパブリッシュするときにトピックを指定し、サブスクライバーは受け取るパブリケーションのトピックを指定します。 サブスクライバーには、サブスクライバーがサブスクライブしたトピックに関する情報だけが送られます。

Point-to-Point メッセージングでは、メッセージごとに特定の宛先を含める必要がありますが、その必要性を排除してパブリッシュ/サブスクライブ・メッセージングで情報のプロバイダーとコンシューマーを分離できるようにしているのは、トピックの存在です。

パブリッシャーとサブスクライバーの間の相互作用はすべて、キュー・マネージャーによって制御されます。 キュー・マネージャーは、パブリッシャーからメッセージを受け取り、サブスクライバーから (一連のトピックの) サブスクリプションを受け取ります。 キュー・マネージャーのジョブは、パブリッシュされたメッセージを、メッセージのトピックへのインタレストを登録したサブスクライバーにルーティングすることです。

IBM MQ の標準機能を使用してメッセージが配布されるので、アプリケーションは既存の IBM MQ アプリケーションが使用できるすべてのフィーチャーを使用できます。 つまり、持続メッセージを使用して、一度だけ保証される配信を取得することや、メッセージをトランザクション作業単位の一部にして、パブリッシャーがコミットしたメッセージだけがサブスクライバーに配信されるようにすることが可能ということになります。

パブリッシャーとパブリケーション

IBM MQ パブリッシュ/サブスクライブでは、パブリッシャーとは、指定されたトピックに関する情報を、パブリケーションという標準的な IBM MQ メッセージの形式でキュー・マネージャーに対して使用可能にするアプリケーションです。 1 つのパブリッシャーが複数のトピックに関する情報をパブリッシュすることができます。

パブリッシャーは、MQPUT verb を使用して、事前にオープンしたトピックにメッセージを書き込みます。このメッセージがパブリケーションです。 次に、ローカル・キュー・マネージャーは、パブリケーションのトピックへのサブスクリプションを持つ任意のサブスクライバーに、パブリケーションを送付します。複数のサブスクライバーが、パブリッシュされたメッセージをコンシュームできます。

キュー・マネージャーは、適切なサブスクリプションを持つすべてのローカル・サブスクライバーにパブリケーションを配布することに加えて、自分に接続している他のキュー・マネージャーに、直接的にあるいはトピックのサブスクライバーを持つキュー・マネージャーのネットワークを介して、パブリケーションを配布することもできます。

IBM MQ パブリッシュ/サブスクライブ・ネットワークでは、パブリッシュ・アプリケーションはサブスクライバーにもなれます。

同期点の下にあるパブリケーション

パブリッシャーは同期点で MQPUT または MQPUT1 呼び出しを発行して、作業単位内でサブスクライバーに配信されたすべてのメッセージを含めることができます。 MQPMO_RETAIN オプション、または値が ALL または ALLDUR であるトピック配信オプション NPMSGDLV または PMSGDLV が指定された場合、キュー・マネージャーはパブリッシャー MQPUT または MQPUT1 呼び出しの範囲内で、内部的な MQPUT または MQPUT1 呼び出しを同期点で使用します。

状態情報とイベント情報

パブリケーションは、状態パブリケーション (例えば、現在の株価) またはイベント・パブリケーション (例えば、その株の取引) のどちらかに分類できます。

状態パブリケーション

状態パブリケーション には、何かの現在の状態 (例えば、株価やサッカーの試合の現在のスコア) に関する情報が含まれます。 何かが起こると (例えば、株価の変動やサッカーのスコアの変化)、それまでの状態情報は新しい情報に取って代わるので、不要になります。

サブスクライバーは、開始時に現行バージョンの状態情報を受信し、状態が変わるたびに新しい情報が送信されてくることを望むものです。

パブリケーションに状態情報が含まれる場合、そのパブリケーションは多くの場合に保存パブリケーションとしてパブリッシュされます。 新規サブスクライバーは通常、現在の状態情報を直ちに必要とします。 イベントが発生して情報がリパブリッシュされるのを待機することを望んではいません。 サブスクライバーが MQSO_PUBLICATIONS_ON_REQUEST または MQSO_NEW_PUBLICATIONS_ONLY オプションを使用しない限り、サブスクライブしたサブスクライバーは、トピックの保存パブリケーションを自動的に受信します。

イベント・パブリケーション

イベント・パブリケーション には、発生した個々のイベント (例えば、何かの株の取引や特定のゴールの得点) に関する情報が含まれます。 各イベントは他のイベントから独立しています。

サブスクライバーは、イベントが発生すると、そのイベントに関する情報を受信することを望むものです。

保存パブリケーション

デフォルトでは、関心を持つすべてのサブスクライバーにパブリケーションが送信された後、そのパブリケーションは廃棄されます。 ただし、パブリッシャーはパブリケーションのコピーを保存することを指定できます。その結果、そのトピックへのインタレストを登録する今後のサブスクライバーにパブリケーションのコピーを送信することができます。

関心を持つすべてのサブスクライバーにパブリケーションが送信された後にそのパブリケーションを削除するのは、イベント情報に適していますが、状態情報には必ずしも適しているわけではありません。 メッセージを保存することによって、新規サブスクライバーは、初回の状態情報を受信するのに情報が再びパブリッシュされるのを待機する必要がなくなります。 例えば、株価のサブスクリプションを登録したサブスクライバーは、株価が変動する (したがってリパブリッシュされる) のを待たずに、現在の株価を直ちに受信することになります。

キュー・マネージャーは、各トピックのパブリケーションを 1 つだけ保存できます。したがって、新しい保存パブリケーションがキュー・マネージャーに到着すると、トピックの既存の保存パブリケーションが削除されます。 ただし、既存パブリケーションが削除されるのが、新しい保存パブリケーションの到着と

同期しない場合もあります。 そのため、どのトピックについても可能な限り、保存パブリケーションを送信するパブリッシャーが 1 つを超えないようにしてください。

MQSO_NEW_PUBLICATIONS_ONLY サブスクリプション・オプションを使用することにより、サブスクライバーは保存パブリケーションを受信しないことを指定できます。 既存のサブスクライバーは、保存パブリケーションの複製コピーが送信されてくるよう要求することができます。

状態情報であっても、以下のように、パブリケーションを保存する必要がない場合があります。

- あるトピックへのすべてのサブスクリプションが行われた後にそのトピックのパブリケーションが行われ、新しいサブスクリプションが見込まれない、または新しいサブスクリプションを許可しない場合は、パブリケーションを保存する必要はありません。パブリケーションが初めてパブリッシュされるときに、サブスクライバーの完全セットにパブリケーションが配信されるからです。
- パブリケーションが頻繁に (例えば毎秒) 行われる場合、新しいサブスクライバー (または障害からの復旧サブスクライバー) は、初期サブスクリプションのほとんど直後に現在の状態を受信します。したがって、このようなパブリケーションを保存する必要はありません。
- 大規模なパブリケーションの場合は、各トピックの保存パブリケーションを保管するためのかなりのストレージ・スペースが最終的に必要になることがあります。 複数キュー・マネージャー環境では、一致サブスクリプションを持っているネットワーク内のすべてのキュー・マネージャーによって、保存パブリケーションが保管されます。

保存パブリケーションを使用するかどうかを決定するときは、サブスクライブ・アプリケーションが障害からどのように復旧するかを考慮してください。 パブリッシャーが保存パブリケーションを使用しない場合は、その現在の状態をサブスクライバー・アプリケーションがローカル保管しなければならないこともあります。

パブリケーションが保存されるようにするには、MQPMO_RETAIN メッセージ書き込みオプションを使用します。 このオプションを使用してもパブリケーションを保持できない場合、メッセージはパブリッシュされず、呼び出しは MQRC_PUT_NOT_RETAINED で失敗します。

メッセージが保存パブリケーションである場合、このことは MQIsRetained メッセージ・プロパティーで示されます。 メッセージの持続性は、それが最初にパブリッシュされた時の状態と同じです。

関連概念

パブリッシュ/サブスクライブ・クラスターでの保存パブリケーションに関する設計上の考慮事項

同期点の下にあるパブリケーション

IBM MQ パブリッシュ/サブスクライブにおいて、同期点はパブリッシャーが使用することも、キュー・マネージャーが内部的に使用することもできます。

パブリッシャーは MQPMO_SYNCPOINT オプション付きの MQPUT/MQPUT1 呼び出しを発行するときに同期点を使用します。 サブスクライバーに送達されるメッセージはすべて、作業単位内でコミットされていないメッセージの最大数までカウントされます。 MAXUMSGS キュー・マネージャー属性はこの上限を指定します。 上限に到達すると、パブリッシャーは 2024 (07E8) (RC2024): MQRC_SYNCPOINT_LIMIT_REACHED 理由コードを受け取ります。

MQPMO_RETAIN オプション付きの MQPMO_NO_SYNCPOINT を使って、またはトピック送達オプション NPMSGDLV/PMSGDLV に値 ALL または ALLDUR を指定してパブリッシャーが MQPUT/MQPUT1 呼び出しを行うと、キュー・マネージャーは内部同期点を使用して、要求のとおりにメッセージが送達されることを保証します。 パブリッシャーの MQPUT/MQPUT1 呼び出しの有効範囲内に制限値に達すると、パブリッシャーは 2024 (07E8) (RC2024): MQRC_SYNCPOINT_LIMIT_REACHED 理由コードを受け取ることができます。

サブスクライバーとサブスクリプション

IBM MQ パブリッシュ/サブスクライブにおけるサブスクライバーは、パブリッシュ/サブスクライブ・ネットワーク内のキュー・マネージャーに特定のトピックに関する情報を要求するアプリケーションです。 サブスクライバーは、同じまたは異なるトピックに関するメッセージを、複数のパブリッシャーから受信できます。

サブスクリプションは、MQSC コマンドを使用して手動で、またはアプリケーションで作成できます。 これらのサブスクリプションは、ローカル・キュー・マネージャーに送出されます。サブスクリプションには、サブスクライバーが受信を望んでいるパブリケーションに関する以下の情報が含まれます。

- サブスクライバーが関心のあるトピック。ワイルドカードが使用される場合は、複数のトピックとして解決されることがあります。
- パブリッシュされるメッセージに適用される任意指定の選択ストリング。
- 選択されたパブリケーションを置くキュー (サブスクライバー・キュー と呼ばれる) のハンドル、および任意指定の CorrelId。

ローカル・キュー・マネージャーはサブスクリプション情報を保管し、パブリケーションを受信すると、情報をスキャンして、パブリケーションのトピックと選択ストリングが一致するサブスクリプションがあるかどうかを判別します。 一致するサブスクリプションごとに、キュー・マネージャーはサブスクライバーのサブスクライバー・キューにパブリケーションを送信します。 キュー・マネージャーが保管しているサブスクリプション情報は、DIS SUB コマンドおよび DIS SBSTATUS コマンドを使用することによって表示できます。

サブスクリプションが削除されるのは、以下のいずれかのイベントが発生したときだけです。

- サブスクライバーが MQCLOSE 呼び出しを使用してアンサブスクライブした (サブスクリプションが非永続になっていた場合)。
- サブスクリプションの有効期限が切れた。
- システム管理者が DELETE SUB コマンドを使用してサブスクリプションを削除した。
- サブスクライバー・アプリケーションが終了した (サブスクリプションが非永続になっていた場合)。
- キュー・マネージャーが停止または再始動した (サブスクリプションが非永続になっていた場合)。

メッセージを入手する際には、MQGET 呼び出しで適切なオプションを使用します。 アプリケーションが 1 つのサブスクリプションのメッセージのみを処理する場合は、少なくとも、C サンプル・プログラム `amqssbxa.c` および 非管理 MQ サブスクライバーで示されているように、 `get-by-correlid` を使用する必要があります。 使用する **CorrelId** は、MQSD 内の MQSUB から返されます。**SubCorrelId** フィールド。

関連概念

複製サブスクリプションおよび共用サブスクリプション

関連資料

sharedSubscription プロパティーの定義方法を示す例

管理対象キューおよびパブリッシュ/サブスクライブ

サブスクリプションを作成する際、管理キューイングを使用するよう選択できます。 管理キューイングを使用する場合、サブスクリプションの作成時にサブスクリプション・キューが自動的に作成されます。 管理対象キューは、サブスクリプションの永続性に従って、自動的にタイディアップが行われます。 管理対象キューを使用すると、パブリケーションを受け取るキューの作成に関して心配する必要がなくなり、非永続のサブスクリプション接続が閉じられると、コンシュームされていないパブリケーションがサブスクライバー・キューから自動的に除去されます。

アプリケーションが特定のキューをサブスクライバー・キュー (受け取るパブリケーションの宛先) として使用する必要がない場合、MQSO_MANAGED サブスクリプション・オプションを使用して、管理対象サブスクリプション を使用できます。 管理対象サブスクリプションを作成する場合、キュー・マネージャーは、サブスクライバー・キュー用のサブスクライバーにオブジェクト・ハンドルを返します。このサブスクライバー・キューは、キュー・マネージャーによって作成され、そこでパブリケーションを受け取ります。 これは、管理対象サブスクリプションが、IBM MQ がサブスクリプションを処理する場所であるためです。 キュー上での参照、取得、または問い合わせを許可して、キューのオブジェクト・ハンドルが返されます (一時的な動的キューへのアクセス権限を明示的に与えられない限り、管理対象キューの属性の書き込みまたは設定を行うことはできません)。

サブスクリプションの永続性によって、キュー・マネージャーへのサブスクライブ・アプリケーションの接続が中断されたときに、管理対象キューが残るかどうかが決まります。

非永続サブスクリプションで使用される場合、管理対象サブスクリプションは特に便利です。これ以外の方法では、アプリケーションの接続が終了しても、コンシュームされていないメッセージはサブスクライバー・キューに残り、いつまでもキュー・マネージャー内のスペースを占めてしまうからです。 管理対象サブスクリプションを使用する場合、管理対象キューは一時的な動的キューになります。そのため、以下のいずれかの原因で接続が中断した場合、コンシュームされていないメッセージとともに削除されます。

- MQCO_REMOVE_SUB が指定された MQCLOSE が使用され、管理対象 Hobj が閉じられた。
- 非永続サブスクリプション (MQSO_NON_DURABLE) を使用しているアプリケーションへの接続が失われた。
- サブスクリプションの有効期限が切れ、管理対象 Hobj が閉じられたため、サブスクリプションが削除された。

管理対象サブスクリプションは永続サブスクリプションとともに使用できますが、接続が再オープンされたときに、コンシュームされていないメッセージを取得できるように、それらをサブスクライバー・キューに入れたままにするという場合も考えられます。 そのため、永続サブスクリプション用の管理対象キューは永続的な動的キューの形をとり、キュー・マネージャーへのサブスクライブ・アプリケーションの接続が中断されたときにも残ります。

永続的な動的管理対象キューを使用する場合にサブスクリプションに有効期限を設定して、接続が中断された後もそのキューを存続させるものの、無期限には存続させないようにすることができます。

管理対象キューを削除すると、エラー・メッセージを受け取ります。

作成される管理対象キューの名前の末尾には数値 (タイム・スタンプ) が付けられるため、それぞれが固有になります。

サブスクリプション永続性

サブスクリプションを永続的または非永続的として構成できます。 サブスクライブ・アプリケーションがキュー・マネージャーから切断された場合にサブスクリプションで行われる処理は、サブスクリプション永続性で決まります。

永続サブスクリプション

永続サブスクリプションは、キュー・マネージャーへのサブスクライブ・アプリケーションの接続が閉じられても存続します。 サブスクリプションが永続的である場合は、サブスクライブ・アプリケーションが切断されてもサブスクリプションは依然として有効であり、サブスクライブ・アプリケーションは、サブスクリプションを要求して改めて再接続すると使用することができます。このときサブスクライブ・アプリケーションは、サブスクリプションが作成されたときに返された **SubName** を使用します。

永続的にサブスクライブするときは、サブスクリプション名 (**SubName**) が必要です。 サブスクリプション名は、サブスクリプションの識別に使用できるように、キュー・マネージャー内で固有でなければなりません。 つまり、サブスクリプションに対する接続を意図的に閉じていても (MQCO_KEEP_SUB オプションを使用)、キュー・マネージャーから切断されていても、再開するサブスクリプションを指定するときは ID が必要であるということです。 MQSO_RESUME オプションを指定した MQSUB 呼び出しを使用することによって、既存のサブスクリプションを再開できます。 サブスクリプション名は、 SUBTYPE ALL または ADMIN を指定して DISPLAY SBSTATUS コマンドを使用した場合にも表示されます。

アプリケーションが必要としなくなった永続サブスクリプションは、MQCO_REMOVE_SUB オプションを指定した MQCLOSE 関数呼び出しを使用して削除するか、MQSC コマンド DELETE SUB を使用して手動で削除できます。

DURSUB トピック属性を使用して、トピックに対する永続サブスクリプションが可能かどうかを指定できます。

MQSO_RESUME オプションを使用した MQSUB 呼び出しから戻るときにサブスクリプション有効期限が設定されますが、サブスクリプションの残りの有効期限時間ではなく、元の有効期限に設定されます。

キュー・マネージャーは、永続サブスクリプションに対応するためにパブリケーションの送信を、そのサブスクライバー・アプリケーションが接続されていなくても、続行します。 そのため、サブスクライバー・キューにメッセージがたまることになります。 この問題を回避する最も簡単な方法は、適切なところでは非永続サブスクリプションを使用することです。 一方、永続サブスクリプションを使用する必要がある場合、サブスクライバーが保存パブリケーション・オプションを使用してサブスクライブすることで、メッ

セージが溜まるのを回避できます。 その場合、サブスクライバーは、パブリケーションをいつ受け取るかを MQSUBRQ 呼び出しを使用して制御できます。

非永続サブスクリプション

非永続サブスクリプションは、キュー・マネージャーへのサブスクライブ・アプリケーションの接続が開いている間だけ存在します。 サブスクライブ・アプリケーションが、意図的に、あるいは接続の損失により、キュー・マネージャーから切断されると、サブスクリプションは除去されます。 接続が閉じられると、キュー・マネージャーからサブスクリプションに関する情報が削除され、DISPLAY SBSTATUS コマンドを使用してサブスクリプションを表示しようとしても現れなくなります。 サブスクライバー・キューにはメッセージが書き込まれなくなります。

非永続サブスクリプションの場合にサブスクライバー・キュー上の未コンシューム・パブリケーションがどうなるかは、以下のように決まります。

- サブスクライブしているアプリケーションが 管理対象宛先を使用している場合、コンシュームされていないパブリケーションは自動的に削除されます。
- サブスクライブ・アプリケーションがサブスクライブ時に専用サブスクライバー・キューのハンドルを提供する場合は、未コンシューム・メッセージの自動削除は行われません。 適切な場合にキューをクリアするのは、アプリケーションが行います。 キューが複数のサブスクライバーまたは他の Point-to-Point アプリケーションによって共有されている場合は、キューを完全にクリアするのは適切でない可能性があります。

非永続サブスクリプションの必須ではありませんが、サブスクリプション名が定義されればキュー・マネージャーによって使用されます。 サブスクリプション名は、サブスクリプションの識別に使用できるように、キュー・マネージャー内で固有でなければなりません。

関連概念

複製サブスクリプションおよび共用サブスクリプション

関連タスク

JMS 2.0 共用サブスクリプションの使用

関連資料

sharedSubscription プロパティーの定義方法を示す例

選択ストリング

選択ストリングとは、サブスクリプションと一致するかどうかを判別するため、パブリケーションに適用される式のことです。 選択ストリングにはワイルドカード文字を含めることができます。

サブスクライブするときには、トピックを指定することに加えて、選択ストリングを指定して、メッセージ・プロパティーに従ってパブリケーションを選択することができます。

パブリッシャーによってメッセージが書き込まれると、各サブスクライバーに送達するために変更される前に、選択ストリングがそのメッセージに対して評価されます。 パブリッシュ操作の一部として変更される可能性のある、選択ストリング内のフィールドを使用する際には注意してください。 例えば、MQMD フィールドの `UserIdentifier`、`MsgId`、および `CorrelId` が該当します。

選択ストリングは、パブリッシュ操作の一部としてキュー・マネージャーが追加するメッセージ・プロパティー・フィールドを参照すべきではありません (パブリッシュ/サブスクライブのメッセージ・プロパティーを参照)。 ただし、パブリケーションのトピック・ストリングを含むメッセージ・プロパティー `MQTopicString` は例外です。

関連概念

選択ストリングの規則と制約事項

トピック

トピックは、パブリッシュ/サブスクライブ・メッセージでパブリッシュされる情報の主題です。

Point-to-Point システムでは、メッセージは特定の宛先アドレスに送信されます。 主題ベースのパブリッシュ/サブスクライブ・システムでは、メッセージの内容を表す主題に基づいて、サブスクライバーにメッセ

ージが送信されます。 内容ベースのシステムでは、メッセージ自体の内容に基づいてサブスクライバーにメッセージが送信されます。

IBM MQ パブリッシュ/サブスクライブ・システムは、主題ベースのパブリッシュ/サブスクライブ・システムです。 パブリッシャーはメッセージを作成し、パブリケーションの主題に最適なトピック・ストリングと共にパブリッシュします。 パブリケーションを受信するために、サブスクライバーは、パブリケーション・トピックを選択するためのパターン・マッチング・トピック・ストリングを指定したサブスクリプションを作成します。 キュー・マネージャーは、サブスクリプションがパブリケーション・トピックに一致していて、パブリケーションを受信する権限のあるサブスクライバーにパブリケーションを配信します。 71 ページの『トピック・ストリング』の文書では、パブリケーションの主題を識別するためのトピック・ストリングの構文について説明しています。 サブスクライバーも、受信するトピックを選択するためにトピック・ストリングを作成します。 サブスクライバーが作成するトピック・ストリングには、パブリケーションのトピック・ストリングとのパターン・マッチングのための 2 つの代替ワイルドカード・スキームのうちのどちらかを含めることができます。 パターン・マッチングについては、72 ページの『ワイルドカード・スキーマ』で説明しています。

主題ベースのパブリッシュ/サブスクライブでは、パブリッシャー (管理者) が主題をトピックに分類する必要があります。 主題は通常、トピック・ツリーとして階層的に編成されます。その場合、'/' 文字を使用してトピック・ストリング内にサブトピックを作成します。 トピック・ツリーの例については、78 ページの『トピック・ツリー』を参照してください。 トピックはトピック・ツリー内のノードです。 トピックはそれ以上サブトピックのないリーフ・ノードか、サブトピックのある中間ノードのいずれかになります。

主題を階層型トピック・ツリーに編成すると同時に、トピックを管理トピック・オブジェクトに関連付けることができます。 トピックを管理トピック・オブジェクトに関連付けることにより、トピックをクラスター内に配布するかどうかなどの属性をトピックに割り当てます。 関連付けは、管理トピック・オブジェクトの TOPICSTR 属性を使用してトピックを指定することによって行います。 管理トピック・オブジェクトをトピックに明示的に関連付けない場合、トピックは、管理トピック・オブジェクトに既に 関連付けられている、トピック・ツリー内の最も近い上位トピックの属性を継承します。 親トピックをまったく定義していない場合は、SYSTEM.BASE.TOPIC から継承します。 管理トピック・オブジェクトについては、79 ページの『管理トピック・オブジェクト』で説明しています。

注 : トピックのすべての属性を SYSTEM.BASE.TOPIC から継承する場合でも、SYSTEM.BASE.TOPIC から直接継承するトピックのルート・トピックを定義してください。 例えば、米国の州のトピック・スペース (USA/Alabama や USA/Alaska など) では、USA がルート・トピックです。 ルート・トピックの主な目的は、誤ったサブスクリプションにパブリケーションが一致することがないように、重複しない離散的トピック・スペースを作成することにあります。 ルート・トピックの属性を変更すれば、トピック・スペース全体に影響を及ぼせることにもなります。 例えば、**CLUSTER** 属性の名前を設定することができます。

パブリッシャーまたはサブスクライバーとしてトピックを表すときに、トピック・ストリングを指定するか、トピック・オブジェクトを参照することができます。 あるいはその両方を行うことができ、その場合、指定するトピック・ストリングはトピック・オブジェクトのサブトピックを定義します。 キュー・マネージャーは、トピック・オブジェクトで指定されたトピック・ストリング接頭部にトピック・ストリングを追加し、2 つのトピック・ストリングの間に追加の '/' を挿入することによって、トピックを識別します (例えば、 トピック・ストリング/オブジェクト・ストリング)。 これについては、76 ページの『トピック・ストリングの結合』で詳しく説明しています。 結果のトピック・ストリングを使用してトピックが識別され、管理トピック・オブジェクトに関連付けられます。 管理トピック・オブジェクトは、マスター・トピックに対応するトピック・オブジェクトと同じトピック・オブジェクトであるとは限りません。

内容ベースのパブリッシュ/サブスクライブでは、それぞれのメッセージの内容を検索する選択ストリングを指定することによって、受信するメッセージを定義します。 IBM MQ は、中間形式の内容ベース・パブリッシュ/サブスクライブを提供します。これは、メッセージの全内容ではなくメッセージ・プロパティーをスキャンするメッセージ・セレクターを使用します (セレクターを参照)。 メッセージ・セレクターの典型的な用途は、トピックをサブスクライブし、次に数値プロパティーで選択を限定することです。 セレクターを使用すると、特定の範囲に限定した値に関心があるということを指定できます。これは、文字ベースまたはトピック・ベースのワイルドカードを使用する場合にはできないことです。 メッセージの全内容に基づいてどうしてもフィルタリングを行う必要がある場合は、IBM Integration Bus を使用する必要があります。

トピック・ストリング

トピック・ストリングを使用してトピックとしてパブリッシュする情報にラベルを付けます。 文字ベースまたはトピック・ベースのワイルドカードのどちらかのトピック・ストリングを使用してトピックのグループにサブスクライブします。

トピック

トピック・ストリング は、パブリッシュ/サブスクライブ・メッセージのトピックを識別するための文字ストリングです。 トピック・ストリングを構成するときは、任意の文字を使用できます。

IBM WebSphere® MQ 7 のパブリッシュ/サブスクライブでは、3 文字は特別な意味を持ちます。 これらはトピック・ストリング内のどこにあってもかまいませんが、使用にあたって注意が必要です。 特殊文字の使用については、72 ページの『トピック・ベース・ワイルドカード・スキーム』で説明しています。

スラッシュ (/)

トピック・レベル分離文字。 トピックをトピック・ツリーとして構成するには、'/' 文字を使用します。

できれば、空のトピック・レベル ('//') は避けてください。 これは、トピック階層内のトピック・ストリングのないノードに相当します。 トピック・ストリング内の先頭または末尾の '/' は、先頭または末尾の空ノードに相当します。これも避けてください。

ハッシュ記号 (#)

サブスクリプションでマルチレベル・ワイルドカードを構成する場合に '/' と組み合わせて使用します。 パブリッシュされるトピックの指定に使用するトピック・ストリングで '/' に隣接して '#' を使用する場合は、注意が必要です。 71 ページの『トピック・ストリングの例』で、'#' の理にかなった使い方を示しています。

ストリング '.../#/...'、'#/...'、および '.../#' は、サブスクリプション・トピック・ストリングでは特別な意味を持ちます。 これらのストリングは、トピック階層の 1 つ以上のレベルのすべてのトピックに一致します。 したがって、これらのシーケンスのいずれかを使用してトピックを作成した場合は、同様にトピック階層の複数レベルのすべてのトピックのサブスクライブがなければ、そのトピックをサブスクライブできないことになります。

正符号 (+)

サブスクリプションで単一レベル・ワイルドカードを構成する場合に '/' と組み合わせて使用。 パブリッシュされるトピックの指定に使用するトピック・ストリングで '/' に隣接して '+' を使用する場合は、注意が必要です。

ストリング '.../+/...'、'+/...'、および '.../+' は、サブスクリプション・トピック・ストリングでは特別な意味を持ちます。 これらのストリングは、トピック階層の 1 つのレベルのすべてのトピックに一致します。 したがって、これらのシーケンスのいずれかを使用してトピックを作成した場合は、同様にトピック階層の 1 つのレベルのすべてのトピックのサブスクライブがなければ、そのトピックをサブスクライブできないことになります。

トピック・ストリングの例

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

関連資料

トピック

ワイルドカード・スキーマ
複数のトピックへのサブスクライブに使用される 2 つのワイルドカード方式が存在します。 方式の選択はサブスクリプション・オプションです。

MQSO_WILDCARD_TOPIC
トピック・ベースのワイルドカード方式を使用して、サブスクライブするトピックを選択します。

ワイルドカード・スキーマを明示的に選択しない場合は、これがデフォルトです。

MQSO_WILDCARD_CHAR
文字ベースのワイルドカード方式を使用して、サブスクライブするトピックを選択します。

DEFINE SUB コマンドで **wschema** パラメーターを指定して、いずれかのスキームを設定してください。 詳細については、DEFINE SUB を参照してください。

注 : IBM WebSphere MQ 7.0 より前に作成されたサブスクリプションは、常に文字ベースのワイルドカード方式を使用します。

例

```
IBM/+/Results
#/Results
IBM/Software/Results
IBM/*ware/Results
```

トピック・ベース・ワイルドカード・スキーム
トピック・ベースのワイルドカードを使用すると、サブスクライバーは同時に複数のトピックをサブスクライブできます。

トピック・ベースのワイルドカードは、IBM MQ パブリッシュ/サブスクライブのトピック・システムの強力なフィーチャーです。 マルチレベル・ワイルドカードと単一レベル・ワイルドカードはサブスクリプションに使用できますが、メッセージのパブリッシャーがトピック内で使用することはできません。

トピック・ベース・ワイルドカード・スキームでは、トピック・レベル別にグループ化されたパブリケーションを選択できます。 サブスクリプションのトピック・レベルのストリングがパブリケーションのストリングと完全に一致しなければならないかどうかを、トピック階層のレベルごとに選択できます。 例えば、サブスクリプションが `IBM/+/Results` の場合は、次のトピックがすべて選択されます。

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

ワイルドカードには 2 つのタイプがあります。

マルチレベル・ワイルドカード

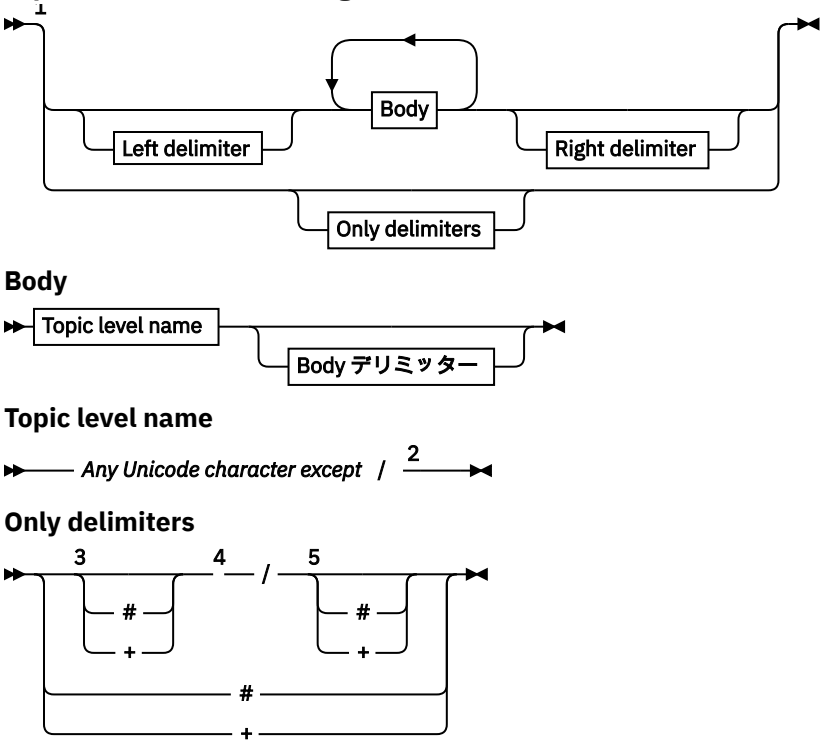
- マルチレベル・ワイルドカードは、サブスクリプションで使用されます。 パブリケーションで使用すると、リテラルとして扱われます。
- マルチレベル・ワイルドカード文字 '#' は、トピック内のレベルをいくつでも一致させる場合に使用します。 例えば、トピック・ツリー例を使用すると、'USA/Alaska/#' をサブスクライブした場合は、'USA/Alaska' トピックと 'USA/Alaska/Juneau' トピックに関するメッセージを受け取ります。
- マルチレベル・ワイルドカードはゼロ個以上のレベルを表すことができます。 したがって、'USA/#' は 'USA' 単独とも一致します。この場合、'#' はゼロ個のレベルを表しています。 このコンテキストでは、トピック・レベル分離文字は意味を持ちません。分離するレベルがないからです。
- マルチレベル・ワイルドカードは、単独で指定された場合、またはトピック・レベル分離文字に続いて指定された場合のみ有効です。 したがって、'#' と 'USA/#' は有効なトピックです。この場合、'#' 文字はワイルドカードとして扱われます。 一方、'USA#' も有効なトピック・ストリングではありますが、'#' 文字はワイルドカードと見なされず、特別な意味を持ちません。 詳しくは、74 ページの『トピック・ベースのワイルドカードが無効な場合』 を参照してください。

単一レベル・ワイルドカード

- 単一レベル・ワイルドカードは、サブスクリプションで使用されます。 パブリケーションで使用すると、リテラルとして扱われます。
- 単一レベル・ワイルドカード文字 '+' は、トピック・レベルが 1 つだけ一致します。 例えば、'USA/+' は 'USA/Alabama' と一致しますが、'USA/Alabama/Auburn' とは一致しません。 単一レベル・ワイルドカードは単一レベルのみと一致するので、'USA/+' は 'USA' と一致しません。
- 単一レベル・ワイルドカードはトピック・ツリーのどのレベルでも使用でき、マルチレベル・ワイルドカードとの併用が可能です。 単一レベル・ワイルドカードは、単独で指定する場合を除いて、トピック・レベル分離文字に続いて指定する必要があります。 したがって、'+' と 'USA/+' は有効なトピックです。この場合、'+' 文字はワイルドカードとして扱われます。 一方、'USA+' も有効なトピック・ストリングではありますが、'+' 文字はワイルドカードと見なされず、特別な意味を持ちません。 詳しくは、74 ページの『トピック・ベースのワイルドカードが無効な場合』 を参照してください。

トピック・ベース・ワイルドカード・スキームの構文には、エスケープ文字がありません。 '#' および '+' がワイルドカードとして扱われるかどうかは、それらのコンテキストによります。 詳しくは、74 ページの『トピック・ベースのワイルドカードが無効な場合』 を参照してください。

注 : トピック・ストリングの先頭と末尾は、特別に扱われます。 '$' を使用してストリングの終わりを示すと、 '$#/...' はマルチレベル・ワイルドカードで、 '$/#/..'になります。 ルートにある空のノードで、その後にマルチレベル・ワイルドカードが続きます。



注:

[1] ヌル・トピック・ストリングもゼロ長トピック・ストリングも無効です。

[2] char ベースのワイルドカード・スキームとトピック・ベースのワイルドカード・スキームの間の互換性のために、レベル名ストリングでは *, ?, % のいずれも使用しないことをお勧めします。

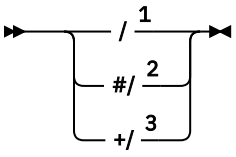
[3] これらのケースは *left delimiter* パターンと同等です。

[4] ワイルドカードなしの / は、単一の空トピックに相当します。

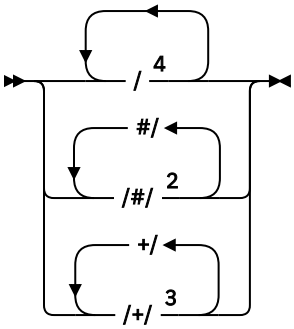
[5] これらのケースは *right delimiter* パターンと同等です。

[6] すべてのトピックと一致します。
[7] レベルが 1 つだけのすべてのトピックと一致します。

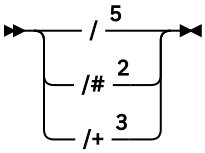
Left delimiter



Body delimiter



Right delimiter



注:

[1] トピック・ストリングは空トピックで始まります。
[2] ゼロ個以上のレベルと一致します。 複数のマルチレベル・マッチング・ストリングは、1 つのマルチレベル・マッチング・ストリングと同じ効果があります。
[3] 1 レベルだけ一致します。
[4] // は空トピック (トピック・ストリングのないトピック・オブジェクト) です。
[5] トピック・ストリングは空トピックで終わります。

トピック・ベースのワイルドカードが無効な場合

ワイルドカード文字の '+' と '#' は、あるトピック・レベル内で他の文字 (それら自身を含む) と混用されると、特別な意味を持たなくなります。

つまり、あるトピック・レベルに '+' または '#' が他の文字と一緒に含まれるトピックをパブリッシュできることになります。

例えば、次の 2 つのトピックを考えてみましょう。

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

最初の例では、'+' および '#' 文字はワイルドカードとして扱われるので、パブリッシュのトピック・ストリングでは無効ですが、サブスクリプションでは有効です。

2 番目の例では、'+' および '#' 文字はワイルドカードとして扱われないので、パブリッシュとサブスクライブの両方のトピック・ストリングにすることができます。

例

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

文字ベースのワイルドカード・スキーム
文字ベースのワイルドカード・スキームでは、 従来どおりの文字のマッチングに基づいてトピックを選択できます。

ストリング '*' を使用すると、トピック階層の複数レベルにあるすべてのトピックを選択できます。 文字ベースのワイルドカード・スキームで '*' を使用することは、トピック・ベース・ワイルドカード・ストリング '#' を使用することと同等です。

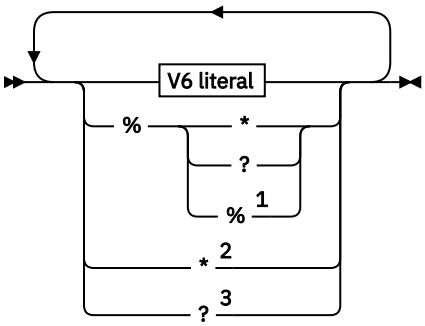
' *x*/*/*y* ' は、トピック・ベースのスキームでは ' *x*/#/*y* ' と同等であり、レベル ' *x* と *y* ' の間のトピック階層内のすべてのトピックを選択します。ここで、 ' *x* ' と ' *y* ' は、ワイルドカードによって返される一連のレベルに含まれていないトピック名です。

トピック・ベースのスキーマの '/+/' には、文字ベースのスキーマで完全に同等のものはありません。 'IBM/*/Results' は、'IBM/Patents/Software/Results'も選択します。 階層の各レベルにおいてトピック名セットが固有である場合にのみ、 2 つのスキーマを使用して、完全な一致をもたらす照会を常に構成できます。

一般的な使用法において、文字ベースのスキーマの '*' と '?' には、トピック・ベースのスキームで同等のものはありません。 トピック・ベースのスキーマでは、ワイルドカードを使用した部分マッチングを実行しません。 文字ベースのワイルドカード・サブスクリプション 'IBM/*ware/Results' には、トピック・ベースのスキームで同等のものはありません。

注 : 文字のワイルドカード・サブスクリプションを使用したマッチングは、 トピック・ベースのサブスクリプションを使用したマッチングよりも遅くなります。

Character-based wildcard string



V6 literal

►► *? を除く任意の Unicode 文字 *および* % ►◄

注:

[1] 「次の文字をエスケープ」を意味するため、リテラルとして扱われます。 '%' の後には '*'、'?' または '%' が続く必要があります。 71 ページの『トピック・ストリングの例』を参照してください。
[2] サブスクリプションで「0 文字以上一致」を意味します。
[3] サブスクリプションで「1 文字だけ一致」を意味します。

例

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

トピック・ストリングの結合

サブスクリプションを作成するとき、またはトピックを開いてメッセージをパブリッシュできるようにする場合、2 つの別個のサブトピック・ストリングまたは "サブトピック"を結合することにより、トピック・ストリングを形成することができます。 1 つのサブトピックは、アプリケーションまたは管理コマンドによってトピック・ストリングとして提供され、もう 1 つはトピック・オブジェクトに関連付けられたトピック・ストリングです。 サブトピック自体をトピック・ストリングとして使用することも、複数を組み合わせて新しいトピック名を形成することもできます。

例えば、MQSC コマンド **DEFINE SUB** を使用してサブスクリプションを定義する場合、コマンドは **TOPICSTR** (トピック・ストリング) または **TOPICOBJ** (トピック・オブジェクト) のいずれかまたは両方を属性として受け入れることができます。 **TOPICOBJ** のみが提供された場合、そのトピック・オブジェクトに関連付けられたトピック・ストリングがトピック・ストリングとして使用されます。 **TOPICSTR** のみが提供された場合、それがトピック・ストリングとして使用されます。 両方が提供されると、**TOPICOBJ** / **TOPICSTR** 形式でそれらが連結されて単一のトピック・ストリングになります。ここで、**TOPICOBJ** で構成されるトピック・ストリングが常に最初であり、ストリングの 2 つの部分は常に "/" 文字で区切られます。

同様に、MQI プログラムでは、MQOPEN によってフル・トピック名が作成されます。 これは、パブリッシュ/サブスクライブ MQI 呼び出しで使用される 2 つのフィールドにより、次にリストする順序で構成されます。

1. **ObjectName** フィールドで指定されたトピック・オブジェクトの **TOPICSTR** 属性。
2. アプリケーションが提供するサブトピックを定義する **ObjectString** パラメーター。

結果のトピック・ストリングは **ResObjectString** パラメーターで戻されます。

各フィールドの最初の文字がブランクや NULL 文字ではなく、フィールド長がゼロより大きい場合に、これらのフィールドは存在すると見なされます。 1 つのフィールドだけが存在する場合は、未変更のままトピック名として使用されます。 どちらのフィールドにも値が設定されていない場合、呼び出しは理由コード MQRC_UNKNOWN_OBJECT_NAME または MQRC_TOPIC_STRING_ERROR (フル・トピック名が無効な場合) により失敗します。

両方のフィールドが存在する場合、結果の結合されたトピック名の 2 つのエレメントの間に "/" 文字が挿入されます。

以下の表は、トピック・ストリング連結の例を示しています。

表 *2.* トピック・ストリング連結の例

トピック・オブジェクトの TOPICSTR	**アプリケーションまたは DEFINE SUB コマンドにより提供されるトピック・ストリング**	**フル・トピック名**	**コメント**
Football/Scores	''	Football/Scores	トピック・オブジェクトの TOPICSTR は単独で使用されます。
''	Football/Scores	Football/Scores	ObjectString/TOPICSTR は単独で使用されます。
Football	Scores	Football/Scores	"/" 文字が連結点に追加されます。

表 2. トピック・ストリング連結の例 (続き)

トピック・オブジェクトの **TOPICSTR**	アプリケーションまたは **DEFINE SUB** コマンドにより提供されるトピック・ストリング	フル・トピック名	コメント
Football	/Scores	Football//Scores	2 つのストリングの間に「空ノード」が生成されます。 これは、"Football/Scores" とは異なります。
/Football	Scores	/Football/Scores	トピックが「空ノード」で始まります。 これは、"Football/Scores" とは異なります。

"/" 文字は特殊文字と見なされ、78 ページの『トピック・ツリー』 内の完全なトピック名に構造を提供します。 "/" 文字は、トピック・ツリーの構造が影響を受けるため、他の理由で使用することはできません。 トピック "/Football" は、トピック "Football"と同じではありません。

注 : サブスクリプションの作成時にトピック・オブジェクトを使用する場合、トピック・オブジェクトのトピック・ストリングの値は、定義時にサブスクリプションで固定されます。 それ以降、トピック・オブジェクトが変更されても、サブスクリプションが定義されているトピック・ストリングには影響しません。

トピック・ストリングのワイルドカード文字

以下のワイルドカード文字が特殊文字です。

- 正符号 (+)
- 番号記号 (#)
- アスタリスク (*)
- 疑問符 (?)

ワイルドカード文字は、サブスクリプションで使用される場合にのみ特別な意味を持ちます。 これらの文字は、他の場所で使用した場合に無効とは見なされませんが、使用方法を確実に理解しておく必要があります。 トピック・オブジェクトをパブリッシュまたは定義するときには、トピック・ストリングでこれらの文字を使用しない方が良い場合があります。

1 つのトピック・レベル内で # または + が他の文字 (それらの文字自体を含む) と混在するようなトピック・ストリングでパブリッシュする場合、どちらかのワイルドカード体系を使用してトピック・ストリングをサブスクライブできます。

2 つの / 文字間の唯一の文字として # または + があるトピック・ストリングでパブリッシュする場合、ワイルドカード体系 MQSO_WILDCARD_TOPIC を使用するアプリケーションによってトピック・ストリングを明示的にサブスクライブできません。 この状態の結果、アプリケーションは予期していたよりも多くのパブリケーションを受け取ることになります。

定義されたトピック・オブジェクトのトピック・ストリングでワイルドカード文字を使用しないでください。 使用すると、オブジェクトがパブリッシャーによって使用されるときに、文字がリテラル文字として処理され、サブスクリプションによって使用されるときにワイルドカード文字として処理されます。 これにより混乱が発生する可能性があります。

コード・スニペットの例

このコード・スニペットは、サンプル・プログラムの例 2: 可変トピックへのパブリッシャーから抽出したもので、トピック・オブジェクトと可変トピック・ストリングを結合しています。

```
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);
```

トピック・ツリー

定義する各トピックは、トピック・ツリー内の要素、つまりノードです。 トピック・ツリーは、最初は空にしておくことも、MQSC または PCF コマンドを使用して既に定義されたトピックを含めることも可能です。 トピック作成コマンドを使用するか、パブリケーションまたはサブスクリプションで初めてトピックを指定することによって、新規トピックを定義できます。

トピックのトピック・ストリングを定義する際に任意の文字ストリングを使用できますが、階層ツリー構造に適合するトピック・ストリングにすることをお勧めします。 トピック・ストリングとトピック・ツリーを注意深く設計すると、次の操作が容易になります。

- 複数のトピックのサブスクライブ。
- セキュリティー・ポリシーの確立。

トピック・ツリーをフラットな線形構造で構成することは可能ですが、ルート・トピックが 1 つ以上ある階層構造のトピック・ツリーを構築したほうがより効果的です。 セキュリティーの計画およびトピックについて詳しくは、パブリッシュ/サブスクライブのセキュリティーを参照してください。

78 ページの図 18 は、ルート・トピックが 1 つあるトピック・ツリーの例を示しています。

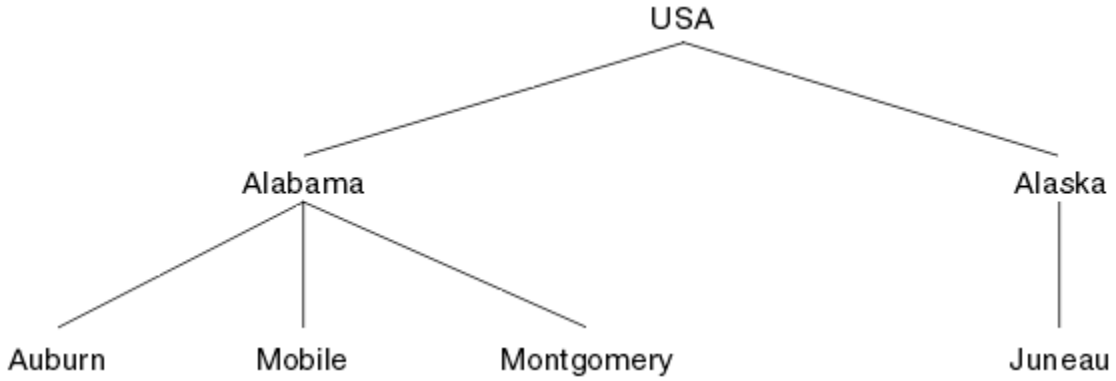


図 *18.* トピック・ツリーの例

図の中の各文字ストリングは、トピック・ツリー内のノードを表しています。 完全なトピック・ストリングは、トピック・ツリー内の 1 つ以上のレベルからノードを集約することによって作成されます。 レベルは「/」文字で分離されます。 完全指定トピック・ストリングの形式は、「root/level2/level3」になります。

78 ページの図 18 に示すトピック・ツリー内の有効なトピックは、以下のとおりです。

「USA」
「USA/Alabama」
「USA/Alaska」
「USA/Alabama/Auburn」
「USA/Alabama/Mobile」
「USA/Alabama/Montgomery」
「USA/Alaska/Juneau」

トピック・ストリングおよびトピック・ツリーを設計するときは、キュー・マネージャーがトピック・ストリングそのものを解釈したり、トピック・ストリングそのものから意味を引き出したりはしないということを覚えておいてください。 選択されたメッセージをそのトピックのサブスクライバーに送信するのにトピック・ストリングが使用されるだけです。

トピック・ツリーの構造と内容には、次の原則が適用されます。

- トピック・ツリー内のレベル数には制限がありません。
- トピック・ツリー内のレベルの名前の長さには制限がありません。
- 「ルート」ノードがいくつあってもかまいません。つまり、トピック・ツリーがいくつあってもかまいません。

関連タスク

トピック・ツリー内の不要なトピック数の削減

管理トピック・オブジェクト

管理トピック・オブジェクトを使用すると、デフォルトではない特定の属性をトピックに割り当てることができます。

79 ページの図 19 は、 さまざまなスポーツを扱う別個のトピックに分割された Sport というハイレベル・トピックを トピック・ツリーとして視覚化できる方法を示しています。

図 *19.* 視覚化したトピック・ツリー

79 ページの図 20 は、 トピック・ツリーをさらに細かく、各スポーツに関するさまざまなタイプの情報に分割できる方法を示しています。

図 *20.* 拡張したトピック・ツリー

この図のようなトピック・ツリーを作成するために、管理トピック・オブジェクトを定義する必要はありません。 このツリーの各ノードは、パブリッシュまたはサブスクライブ操作で作成されたトピック・ストリングによって定義されます。 このツリーの各トピックは、親から属性を継承します。 デフォルトではすべての属性が ASPARENT に設定されているため、属性は親トピック・オブジェクトから継承されます。 この例では、すべてのトピックは Sport トピックと同じ属性を持っています。 Sport トピックには管理トピック・オブジェクトがなく、 SYSTEM.BASE.TOPIC。

トピック・ツリーのルート・ノード (SYSTEM.BASE.TOPIC) で mqm 以外のユーザーに権限を付与することはお勧めできません。権限は継承されますが、制限はできないためです。 したがって、このレベルで権限を付与することは、ツリー全体に権限を付与することを意味します。 階層の下層にあるトピック・レベルで権限を付与してください。

トピック・ツリー内の特定のノードに特定の属性を定義するために、管理トピック・オブジェクトを使用できます。 次の例で、管理トピック・オブジェクトは「Soccer」トピックの 永続サブスクリプション・プロパティー DURSUB を値 NO に設定するように定義されています。

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

このトピック・ツリーは、以下のように視覚化できます。

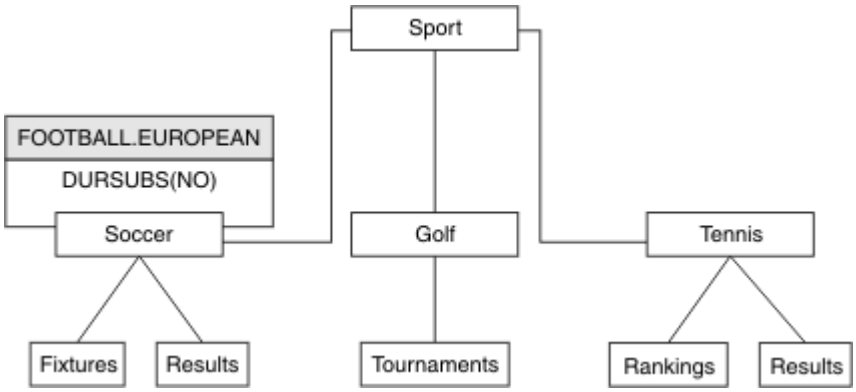


図 *21.*「*Sport/Soccer*」トピックに管理トピック・オブジェクトを関連付けたトピック・ツリーを視覚化した図

ツリー内の「Soccer」の下にあるトピックにサブスクライブするアプリケーションは、管理トピック・オブジェクトを追加する前に使用していたトピック・ストリングを引き続き使用できます。 ただし、ストリング /Sport/Soccer の代わりにオブジェクト名 FOOTBALL.EUROPEAN を使用してサブスクライブするようにアプリケーションを作成できるようになりました。 例えば、/Sport/Soccer/Results にサブスクライブするアプリケーションでは、MQSD.ObjectName として FOOTBALL.EUROPEAN、MQSD.ObjectString として Results を指定できます。

この機能を使用すると、トピック・ツリーの一部をアプリケーション開発者から隠すことができます。 トピック・ツリーの特定のノードで管理トピック・オブジェクトが定義された後、 アプリケーション開発者は独自のトピックをそのノードの子として定義できます。 開発者は親トピックについて知る必要がありますが、親ツリーの他のノードについて知る必要はありません。

属性の継承

トピック・ツリーに多数の管理トピック・オブジェクトがある場合、 各管理トピック・オブジェクトは、デフォルトで直近の親管理トピックから属性を継承します。 81 ページの図 22 は、前の例を拡張したものです。

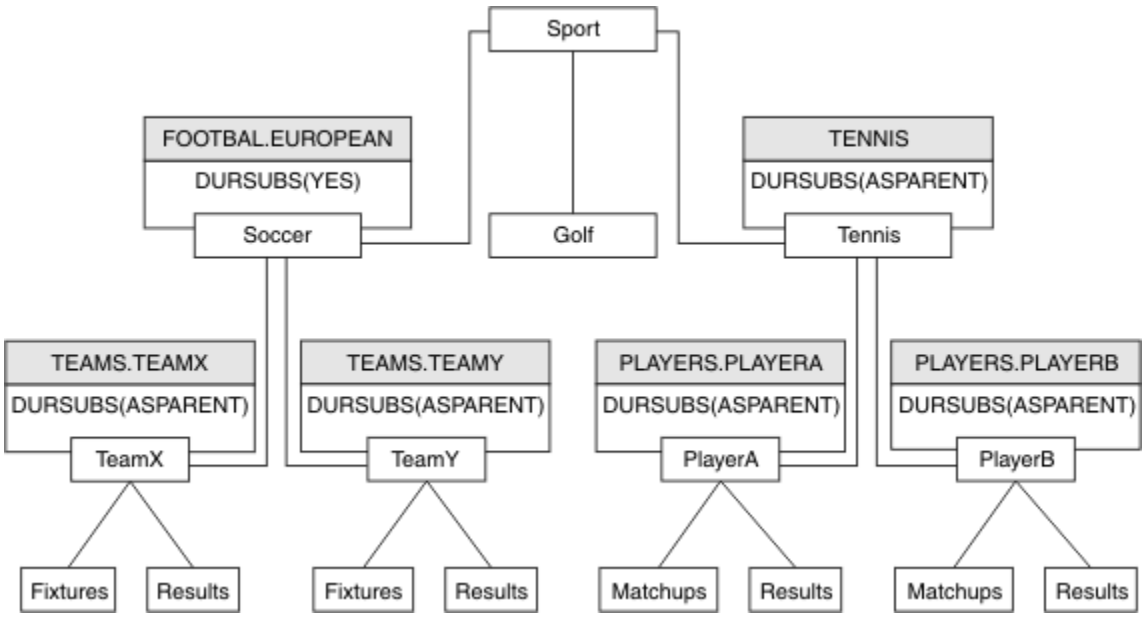


図 *22.* 複数の管理トピック・オブジェクトを定義したトピック・ツリー

例えば、継承を使用して、 /Sport/Soccer のすべての子トピックに、 サブスクリプションが非永続であることを示すプロパティーを設定するとします。 これには、FOOTBALL.EUROPEAN の DURSUB 属性を NO に変更します。

その属性を設定するには、以下のコマンドを使用できます。

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

Sport/Soccer の子トピックのすべての管理トピック・オブジェクトは、プロパティー DURSUB がデフォルト値 ASPARENT に設定されています。 FOOTBALL.EUROPEAN の DURSUB プロパティー値を NO に変更すると、 Sport/Soccer の子トピックは DURSUB プロパティー値 NO を継承します。 Sport/Tennis のすべての子トピックは、 SYSTEM.BASE.TOPIC オブジェクトから DURSUB の値を継承します。 SYSTEM.BASE.TOPIC の値は、YES です。

この状態でトピック Sport/Soccer/TeamX/Results に対する永続サブスクリプションを作成しようとすると失敗しますが、Sport/Tennis/PlayerB/Results に対する永続サブスクリプションを作成しようとする操作は成功します。

WILDCARD プロパティーによるワイルドカードの使用の制御

MQSC **Topic** WILDCARD プロパティーまたは 同等の PCF Topic WildcardOperation プロパティーを使用すると、 ワイルドカード・トピック・ストリング名を使用するサブスクライバー・アプリケーションへのパブリケーションの送達を制御できます。 WILDCARD プロパティーには、以下の 2 つの値のいずれかを指定できます。

WILDCARD
このトピックに対するワイルドカード・サブスクリプションの動作。

PASSTHRU
このトピック・オブジェクトのトピック・ストリングよりも具体的でないワイルドカード・トピックに対するサブスクリプションは、そのトピックまたはそのトピックよりも具体的なトピック・ストリングに対するパブリケーションを受信できるようになります。

BLOCK
このトピック・オブジェクトのトピック・ストリングよりも具体的でないワイルドカード・トピックに対するサブスクリプションは、このトピックまたはこのトピックよりも具体的なトピック・ストリングに対するパブリケーションを受信できなくなります。

サブスクリプションが定義されている場合に、この属性の値が使用されます。 この属性を変更しても、既存のサブスクリプションによってカバーされているトピック・セット は、変更による影響を受けません。 このシナリオは、トピック・オブジェクトが作成または削除されてトポロジーが変更された場合にも当てはまります。WILDCARD 属性の変更後に作成されたサブスクリプションに一致するトピックのセットは、変更後のトポロジーを使用して作成されます。 既存のサブスクリプションについて、一致するトピック・セットを強制的に再評価する場合は、キュー・マネージャーを再開する必要があります。

86 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターを作成する』 の例では、 82 ページの図 23 で示されるトピック・ツリー構造を作成するステップに従うことができます。

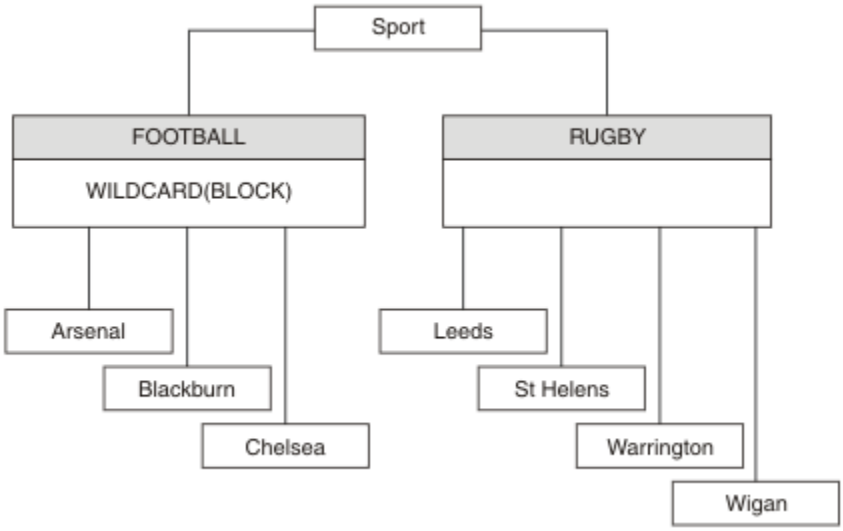


図 *23. WILDCARD* プロパティー *BLOCK* を使用するトピック・ツリー

ワイルドカード・トピック・ストリング # を使用するサブスクライバーは、 Sport トピックと Sport/Rugby サブツリーへのすべてのパブリケーションを受け取ります。 Sport/Football トピックの WILDCARD プロパティー値が BLOCK であるため、このサブスクライバーは Sport/Football サブツリーへのパブリケーションは受け取りません。

PASSTHRU は、デフォルトの設定値です。 Sport ツリーのノードには、WILDCARD プロパティー値 PASSTHRU を設定できます。 ノードで WILDCARD プロパティー値 BLOCK が設定されていない場合、PASSTHRU を設定しても、Sports ツリーのノードのサブスクライバーによって 観測される動作が変化することはありません。

この例では、サブスクリプションを作成して、 送達されるパブリケーションにワイルドカード設定が与える影響を確認します。 87 ページの図 27 を参照してください。 88 ページの図 30 でパブリッシュ・コマンドを実行して、 パブリケーションをいくつか作成します。

```
pub QMA
```

図 *24. QMA* へのパブリッシュ

83 ページの表 3 では結果が示されています。 WILDCARD プロパティー値 BLOCK を設定すると、 ワイルドカードを含むサブスクリプションは、 そのワイルドカードの有効範囲内にあるトピックへのパブリケーションを受信しなくなることに注意してください。

表 3. QMA で受信されたパブリケーション			
サブスクリプション	**トピック・ストリング**	**受信されたパブリケーション**	**注**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	「Football」サブツリーへのすべてのパブリケーションは、 Sports/Football の WILDCARD(BLOCK) によってブロックされます。
SARSENAL	Sports/#/Arsenal	-	Sports/Football の WILDCARD(BLOCK) により、 Arsenal でのワイルドカード・サブスクリプションは防止されます。
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	「Sports/Rugby」のデフォルトである WILDCARD は、Leeds でのワイルドカード・サブスクリプションを防止しません。

注:

あるサブスクリプションに、WILDCARD プロパティー値 BLOCK を持つトピック・オブジェクトに一致するワイルドカードがあるとします。 このサブスクリプションで、一致するワイルドカードの右側にトピック・ストリングもある場合、 サブスクリプションがパブリケーションを受信することはありません。 ブロックされないパブリケーションのセットは、 ブロックされたワイルドカードの親であるトピックへのパブリケーションです。 BLOCK プロパティー値を持つトピックの子であるトピックへのパブリケーションは、ワイルドカードによってブロックされます。 したがって、ワイルドカードの右側にトピックが含まれるサブスクリプション・トピック・ストリングは、 一致するパブリケーションを受信することがありません。

WILDCARD プロパティー値を BLOCK に設定しても、ワイルドカードが含まれるトピック・ストリングを使用したサブスクライブが実行できなくなるわけではありません。 このようなサブスクリプションは正常に行われます。 このサブスクリプションには、 WILDCARD プロパティー値 BLOCK を持つトピック・オブジェクトを含むトピックに一致する 明示的なトピックが含まれます。 これは、WILDCARD プロパティー値 BLOCK を持つトピックの親または子であるトピック用に、 ワイルドカードを使用します。 82 ページの図 23 の例では、 Sports/Football/# のようなサブスクリプションがパブリケーションを受信できます。

ワイルドカードとクラスター・トピック

クラスター・トピック定義はクラスター内のすべてのキュー・マネージャーに伝搬されます。 クラスター内のキュー・マネージャーでクラスター・トピックへのサブスクリプションを行うと、 そのキュー・マネージャーで複数のプロキシー・サブスクリプションが作成されます。 クラスター内の他のすべてのキュー・マネージャーで 1 つのプロキシー・サブスクリプションが作成されます。 ワイルドカードを含むトピック・ストリングを使用したサブスクリプションをクラスター・トピックと組み合わせると、 動作の予測が難しくなる可能性があります。 次の例は、この動作について説明しています。

86 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターを作成する』 の例にあるクラスター・セットアップでは、QMB が QMA と同じサブスクリプションのセットを持っているにもかかわらず、パブリッシャーが QMA にパブリッシュした後で QMB はパブリケーションを 受信しませんでした (82 ページの図 24 を参照)。 Sports/Football と Sports/Rugby のトピックは クラスター・トピックですが、fullsubs.tst で定義されているサブスクリプションは クラスター・トピックを参照しません。 QMB から QMA に伝搬されるプロキシー・サブスクリプションはありません。 プロキシー・サブスクリプションがないと、QMA へのパブリケーション は QMB に転送されません。

Sports/#/Leeds などの一部のサブスクリプションは、クラスター・トピック (このケースでは Sports/Rugby) を参照するように見えることがあります。 実際には、Sports/#/Leeds サブスクリプション はトピック・オブジェクト SYSTEM.BASE.TOPIC に解決されます。

Sports/#/Leeds などのサブスクリプションによって参照されるトピック・オブジェクトを解決するための規則は、 以下のとおりです。 トピック・ストリングが最初のワイルドカードの位置まで切り捨てられます。 トピック・ストリングを左方向にスキャンし、 関連付けられた管理トピック・オブジェクトを持つ最初のトピックを探します。 そのトピック・オブジェクトがクラスター名を指定するか、ローカル・トピック・オブジェクトを定義している可能性があります。 Sports/#/Leeds の例では、切り捨て後のトピック・ストリングは Sports であり、 トピック・オブジェクトを持たないため、Sports/#/Leeds は ローカル・トピック・オブジェクトである SYSTEM.BASE.TOPIC から継承します。

クラスター化されたトピックのサブスクライブによってワイルドカード伝搬の動作が変更される仕方を確認するには、 バッチ・スクリプト upsubs.bat を実行します。 このスクリプトはサブスクリプション・キューをクリアし、 fullsubs.tst 内にクラスター・トピック・サブスクリプションを追加します。 パブリケーションのバッチを作成するには、 puba.bat を再び実行します (82 ページの図 24 を参照してください)。

84 ページの表 4 は、パブリケーションがパブリッシュされたのと同じキュー・マネージャーに 2 つの新規サブスクリプションを追加した結果を示しています。 結果は予測どおりであり、新規サブスクリプションはそれぞれ 1 つのパブリケーションを受信し、 他のサブスクリプションによって受信されるパブリケーションの数は変更されません。 他のクラスター・キュー・マネージャーでは予測しない結果が発生します。 85 ページの表 5 を参照してください。

表 *4. QMA* で受信されたパブリケーション

サブスクリプション	トピック・ストリング	受信されたパブリケーション	注
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	「Football」サブツリーへのすべてのパブリケーションは、 Sports/Football の WILDCARD(BLOCK) によってブロックされます。
SARSENAL	Sports/#/Arsenal	-	Sports/Football の WILDCARD(BLOCK) により、 Arsenal でのワイルドカード・サブスクリプションは防止されます。
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	「Sports/Rugby」のデフォルトである WILDCARD は、Leeds でのワイルドカード・サブスクリプションを防止しません。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	サブスクリプションにワイルドカードがないため、 Arsenal はパブリケーションを受信します。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds は、いかなるイベントでもパブリケーションを受信します。

85 ページの表 5 は、 QMB で 2 つの新規サブスクリプションを追加して QMA でパブリッシュした結果を示しています。 これら 2 つの新規サブスクリプションがないときに QMB はパブリケーションを受信しませんでした。 Sports/FootBall と Sports/Rugby はどちらもクラスター・トピックであるため、 これら 2 つの新規サブスクリプションは予測どおりパブリケーションを受信します。 QMB から Sports/Football/Arsenal と Sports/Rugby/Leeds 用のプロキシー・サブスクリプション が QMA に転送された後、 そこからパブリケーションが QMB に送信されました。

予測しなかった結果として、 以前はパブリケーションを受信しなかった 2 つのサブスクリプション (Sports/# と Sports/#/Leeds) が、 パブリケーションを受信するようになりました。 これは、他のサブスクリプション用に QMB に転送されたパブリケーションである Sports/Football/Arsenal と Sports/Rugby/Leeds が、QMB に接続された任意のサブスクライバーから使用可能になったためです。 その結果、ローカル・トピックの Sports/# と Sports/#/Leeds へのサブスクリプションは、 Sports/

Rugby/Leeds パブリケーションを受信します。 「Sports/Football」は独自の WILDCARD プロパティー値が BLOCK に設定されているため、Sports/#/Arsenal は引き続きパブリケーションを受信しません。

表 *5. QMB* で受信されたパブリケーション

サブスクリプション	トピック・ストリング	受信されたパブリケーション	注
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*WILDCARD(BLOCK) on Sports/Football* によってブロックされる、フットボール・サブツリーへのすべてのパブリケーション
SARSENAL	Sports/#/Arsenal	-	Sports/Football の WILDCARD(BLOCK) により、Arsenal でのワイルドカード・サブスクリプションは防止されます。
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	「*Sports/Rugby*」のデフォルトである *WILDCARD* は、*Leeds* でのワイルドカード・サブスクリプションを防止しません。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	サブスクリプションにワイルドカードがないため、 Arsenal はパブリケーションを受信します。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds は、いかなるイベントでもパブリケーションを受信します。

ほとんどのアプリケーションにおいて、 あるサブスクリプションが別のサブスクリプションの動作に影響することは望ましくありません。 WILDCARD プロパティーの値 BLOCK の重要な使用法の 1 つは、 ワイルドカードを含む同じトピック・ストリングへのサブスクリプションをすべて同じように動作させることです。 サブスクリプションがパブリッシャーと同じキュー・マネージャー上にあるか別のキュー・マネージャー上にあるかにかかわらず、 サブスクリプションの結果は同じです。

ワイルドカードとストリーム

パブリッシュ/サブスクライブ API に書き込まれた新規アプリケーションの場合、結果として、* へのサブスクリプションがパブリケーションを受信しなくなります。 「Sports」へのすべてのパブリケーションを受信するには、Sports/* または Sports/# にサブスクライブするとともに、Business パブリケーションについても同様の処理を行う必要があります。

既存のキュー型パブリッシュ/サブスクライブ・アプリケーションの動作は、パブリッシュ/サブスクライブ・ブローカーを新しいバージョンの IBM MQ に移行しても変更されません。 **Publish**、**Register Publisher**、または **Subscriber** コマンドの **StreamName** プロパティーは、ストリームの移行先のトピックの名前にマップされます。

ワイルドカードとサブスクリプション・ポイント

パブリッシュ/サブスクライブ API に書き込まれた新規アプリケーションの場合、 移行の結果、* へのサブスクリプションがパブリケーションを受信しなくなります。 「Sports」へのすべてのパブリケーションを受信するには、Sports/* または Sports/# にサブスクライブするとともに、Business パブリケーションについても同様の処理を行う必要があります。

既存のキュー型パブリッシュ/サブスクライブ・アプリケーションの動作は、パブリッシュ/サブスクライブ・ブローカーを新しいバージョンの IBM MQ に移行しても変更されません。 **Publish**、**Register Publisher**、または **Subscriber** コマンドの **SubPoint** プロパティーは、サブスクリプションの移行先のトピックの名前にマップされます。

例: Sport パブリッシュ/サブスクライブ・クラスターを作成する

以降のステップでは、クラスター CL1 と ともに 4 つのキュー・マネージャー (CL1A および CL1B という 2 つの全リポジトリーと、 QMA および QMB という 2 つの部分リポジトリー) を作成します。 全リポジトリーは、クラスター定義を保持するためにのみ使用されます。 QMA は、クラスター・トピック・ホストに指定されます。 永続サブスクリプションは QMA と QMB の両方で定義されます。

注 : この例は、Windows 用にコーディングされています。 他のプラットフォームでこの例を構成してテストするには、Create qmgrs.bat と create pub.bat を再コーディングする必要があります。

1. 以下のスクリプト・ファイルを作成します。
   a. Create topics.tst
   b. Create wildsubs.tst
   c. Create fullsubs.tst
   d. Create qmgrs.bat
   e. create pub.bat
2. Create qmgrs.bat を実行して構成を作成します。

```
qmgrs
```

82 ページの図 23 でトピックを作成します。 図 5 のスクリプトは、クラスター・トピック Sports/Football および Sports/Rugby を作成します。

注 : REPLACE オプションは、トピックの TOPICSTR プロパティーを置き換えません。 TOPICSTR は、この例でさまざまなトピック・ツリーをテストするために役立つプロパティーです。 トピックを変更するには、最初にトピックを削除します。

```
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')
```

図 *25.* トピックの削除と作成*: topics.tst*

注 : REPLACE によってトピック・ストリングが置き換えられることはないため、トピックを削除します。

ワイルドカードが含まれるサブスクリプションを作成します。 ワイルドカードは、82 ページの図 23 で示されているトピック・オブジェクトを持つトピックに対応します。 各サブスクリプション用のキューを作成します。 スクリプトが実行または再実行されると、キューがクリアされてサブスクリプションは削除されます。

注 : REPLACE オプションによって サブスクリプションの TOPICOBJ または TOPICSTR プロパティーが置き換えられることはありません。 TOPICOBJ または TOPICSTR は、さまざまなサブスクリプションをテストするために変更される便利なプロパティーです。 これらを変更するには、最初にサブスクリプションを削除します。

```
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)
```

図 *26.* ワイルドカード・サブスクリプションの作成*: wildsubs.tst*

クラスター・トピック・オブジェクトを 参照するサブスクリプションを作成します。

注 :

TOPICOBJ によって参照されるトピック・ストリング と TOPICSTR によって定義されるトピック・ストリングの間には、 デリミッター / が自動的に挿入されます。

定義 DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) は、同じサブスクリプションを作成します。 TOPICOBJ は、すでに定義したトピック・ストリングを迅速に参照する方法として使用されます。 サブスクリプションは作成された後、トピック・オブジェクトを参照しなくなります。

```
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)
```

図 *27.* サブスクリプションの削除と作成*: fullsubs.tst*

2 つのリポジトリーが含まれる クラスターを作成します。 パブリッシュとサブスクライブ用に 2 つの部分リポジトリーを作成します。 すべてを削除してやり直すには、スクリプトを再実行します。 このスクリプトでは、トピック階層と初期ワイルドカード・サブスクリプションも作成されます。

注 :

他のプラットフォームでは、同様のスクリプトを作成するか、すべてのコマンドを入力します。 スクリプトを使用すると、迅速にすべてを削除して同一の構成でやり直すことができます。

```
@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
```

図 *28.* キュー・マネージャーの作成*: qmgrs.bat*

サブスクリプションをクラスター・トピックに追加して、構成を更新します。

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

図 *29.* サブスクリプションの更新*: upsubs.bat*

パブリケーション・トピック・ストリングが含まれる メッセージをパブリッシュするには、キュー・マネージャーをパラメーターとして pub.bat を実行します。 Pub.bat は、サンプル・プログラム **amqspub** を使用します。

```
@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
```

図 *30.* パブリッシュ*: pub.bat*

ストリームおよびトピック

キュー型パブリッシュ/サブスクライブには、統合パブリッシュ/サブスクライブ・モデルには存在しないパブリケーション・ストリームの概念があります。 キュー型パブリッシュ/サブスクライブにおいてストリームとは、異なるトピックの情報の流れを分離する手段を提供するものです。 ストリームは、管理上別のトピック ID にマップできる最上位トピックとして実装されます。

ネットワーク上のすべてのブローカーおよびキュー・マネージャーに対して、デフォルトのストリーム SYSTEM.BROKER.DEFAULT.STREAM が自動的にセットアップされます。 このデフォルトのストリームを使用するために追加の構成を行う必要はありません。 デフォルトのストリームを、名前の付けられていないデフォルトのトピック・スペースであると考えてみます。 デフォルト・ストリームにパブリッシュされたトピックは、キューに入れられたパブリッシュ/サブスクライブを有効にすると、接続されているすべてのキュー・マネージャーですぐに使用可能になります。 名前付きのストリームは、名前付きの別個のトピ

ック・スペースに類似しています。 名前付きのストリームは、それを使用するブローカーごとに定義する必要があります。

パブリッシャーとサブスクライバーが別々のキュー・マネージャー上にあり、それらのブローカーが同じブローカー階層内で接続した場合、追加で構成を行う必要なしに、パブリケーションおよびサブスクリプションはブローカー間を流れます。 逆の場合にも同じインターオペラビリティーが働きます。

名前付きストリーム

キュー型パブリッシュ/サブスクライブ・プログラミング・モデルを使用するソリューション・デザイナーが、すべてのスポーツ・パブリケーションを Sport という 1 つの名前付きストリーム内に入れることにしたとします。 キューに入れられたパブリッシュ/サブスクライブが有効になっている IBM MQ 上で実行されるキュー・マネージャーがストリームを使用できるようにするには、ストリームを手動で追加する必要があります。

ストリーム Sport の Soccer/Results にサブスクライブするキュー型パブリッシュ/サブスクライブ・アプリケーションは、変更なしで機能します。 MQSUB を使用してトピック Sport にサブスクライブし、トピック・ストリング Soccer/Results を提供する統合パブリッシュ/サブスクライブ・アプリケーションも、同じパブリケーションを受け取ります。

ストリームを追加する作業の説明は、ストリームの追加のトピックに記載されています。 ストリームを手動で追加しなければならない場合、それには以下の 2 つの理由があります。

1. 後のバージョンのキュー・マネージャー上で作動するキュー型パブリッシュ/サブスクライブ・アプリケーションを引き続き開発し、それらのアプリケーションを統合パブリッシュ/サブスクライブ MQI インターフェースには移行しない場合。
2. トピックに対するストリームのデフォルトのマッピングによりトピック・スペース内で「衝突」が発生し、ストリーム上のパブリケーションに、他の場所にあるパブリケーションと同じトピック・ストリングがある場合。

権限

デフォルトでは、トピック・ツリーのルートに複数のトピック・オブジェクト SYSTEM.BASE.TOPIC, SYSTEM.BROKER.DEFAULT.STREAM および SYSTEM.BROKER.DEFAULT.SUBPOINT があります。 権限 (例えば、パブリッシュやサブスクライブ) は、SYSTEM.BASE.TOPIC 上の権限によって決定されます。 SYSTEM.BROKER.DEFAULT.STREAM または SYSTEM.BROKER.DEFAULT.SUBPOINT 上の権限はどれも無視されます。 SYSTEM.BROKER.DEFAULT.STREAM または SYSTEM.BROKER.DEFAULT.SUBPOINT のいずれかが削除され、空ではないトピック・ストリングで再作成された場合、これらのオブジェクトに定義された権限は通常のトピック・オブジェクトと同じ方法で使用されます。

ストリームとトピックの間のマッピング

キューに入れられたパブリッシュ/サブスクライブ・ストリームは、キューを作成し、ストリームと同じ名前を付けることによって、 IBM MQ で模倣されます。 このキューはストリーム・キューと呼ばれることがあります。 キュー型パブリッシュ/サブスクライブ・アプリケーションではそのように見えるためです。 このキューを SYSTEM.QPUBSUB.QUEUE.NAMELIST という特別な名前リストに追加すると、パブリッシュ/サブスクライブ・エンジンがこのキューを識別します。 この名前リストに特別なキューを追加することによって、ストリームを必要な数だけ追加できます。 最後に、トピックにパブリッシュおよびサブスクライブできるように、ストリームと同じ名前、およびストリーム名と同じトピック・ストリングを持つトピックを追加する必要があります。

ただし、例外的な状況では、トピックを定義する際に、ストリームに対応するトピックに対して、任意のトピック・ストリングを指定できます。 トピック・ストリングの目的は、トピックに、そのトピック・スペース内で固有の名前を付けることです。 通常、その目的は、ストリーム名によって完全に果たされます。 時折、ストリーム名と既存のトピック名が衝突する場合があります。 この問題を解決するには、ストリームに関連付けられたトピックに対して別のトピック・ストリングを選択します。 いずれかのトピック・ストリングを、固有であることを確認して選択してください。

トピック定義で定義されたトピック・ストリングが、パブリッシャーおよびサブスクライバーが MQOPEN MQI 呼び出しまたは MQSUB MQI 呼び出しを使用して指定したトピック・ストリングに通常の方法で接頭部として付加されます。 トピック・オブジェクトを使用してトピックを参照するアプリケーションは、接頭部のトピック・ストリングの選択によって影響を受けることはありません。そのため、トピック・スペース内でパブリケーションが固有になるような任意のトピック・ストリングを選択できます。

各ストリームの各トピックへの再マップは、トピック・ストリングを固有にするために使用されている接頭部によって、あるトピック・セットが他のトピックと完全に区別されることを前提としています。 マッピングを機能させるために厳密に順守する、共通のトピック命名規則を定義する必要があります。

IBM MQ では、接頭部を付けるメカニズムを使用して、トピック・ストリングをトピック・スペース内の別の場所に再マップします。

注 : ストリームを削除するときには、最初に、そのストリームでのすべてのサブスクリプションを削除してください。 サブスクリプションのいずれかが、ブローカー階層内の他のブローカー由来のものである場合には、このアクションが最も重要です。

サブスクリプション・ポイントおよびトピック

名前付きサブスクリプション・ポイントはトピックとトピック・オブジェクトでエミュレートされます。

サブスクリプション・ポイントを手動で追加する場合は、サブスクリプション・ポイントの追加を参照してください。

IBM MQ におけるサブスクリプション・ポイント

IBM MQ は、IBM MQ トピック・ツリー内のさまざまなトピック・スペースにサブスクリプション・ポイントをマップします。 サブスクリプション・ポイントのないコマンド・メッセージのトピックは、未変更のまま IBM MQ トピック・ツリーのルートにマップされ、SYSTEM.BASE.TOPIC からプロパティーを継承します。

サブスクリプション・ポイントを持つコマンド・メッセージは、SYSTEM.QPUBSUB.SUBPOINT.NAMELIST のトピック・オブジェクトのリストを使用して処理されます。 コマンド・メッセージ内のサブスクリプション・ポイント名は、リスト内のトピック・オブジェクトそれぞれのトピック・ストリングと突き合わされます。 一致が検出されると、サブスクリプション・ポイント名がトピック・ノードとしてトピック・ストリングの前に付加されます。 このトピックは、SYSTEM.QPUBSUB.SUBPOINT.NAMELIST で検出された関連トピック・オブジェクトからそのプロパティーを継承します。

サブスクリプション・ポイントを使用することの効果は、サブスクリプション・ポイントごとに別々のトピック・スペースを作成する点にあります。 サブスクリプション・ポイントと名前が同じトピックが、トピック・スペースのルートになります。 各トピック・スペース内のトピックは、サブスクリプション・ポイントと名前が同じトピック・オブジェクトからそれぞれのプロパティーを継承します。

一致するトピック・オブジェクトに設定されていないプロパティーは、SYSTEM.BASE.TOPIC からの通常の方法で継承されます。

MQRFH2 メッセージ・ヘッダーを使用する、キューに入れられた既存のパブリッシュ/サブスクライブ・アプリケーションは、Publish または Register subscriber コマンド・メッセージで **SubPoint** プロパティーを設定することによって処理を続行します。 サブスクリプション・ポイントがコマンド・メッセージ内のトピック・ストリングと結合され、結果のトピックは他の場合と同様に処理されます。

IBM MQ アプリケーションは、サブスクリプション・ポイントの影響を受けません。 アプリケーションが、一致するトピック・オブジェクトのいずれかから情報を継承するトピックを使用する場合、そのアプリケーションは、一致するサブスクリプション・ポイントを使用するキュー型アプリケーションと相互運用できます。

例

既存の WebSphere Message Broker (現在は IBM Integration Bus と呼ばれる) IBM MQ に移行されたパブリッシュ/サブスクライブ・アプリケーションは、 GBP と USD の 2 つのトピック・オブジェクトを、対応するトピック・ストリング 'GBP' と 'USD'を使用して作成しました。

サブスクリプション・ポイント NYSE/IBM/SPOT を使用する IBM MQ 上で実行されるように移行された、トピック「USD」に対する既存のパブリッシャーは、トピック USD/NYSE/IBM/SPOT 上にパブリケーションを作成します。 同様に、サブスクリプション・ポイント NYSE/IBM/SPOT を使用して、USD への既存のサブスクライバーを USD/NYSE/IBM/SPOT に作成します。

MQSUB を呼び出して、IBM MQ パブリッシュ/サブスクライブ・プログラムでドル・スポット価格をサブスクライブします。「C」コード・フラグメントで説明されているように、USD トピック・オブジェクトおよびトピック・ストリング 'NYSE/IBM/SPOT' を使用してサブスクリプションを作成します。

```
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. クラスター・トピック・ホスト上の USD および GBP トピック・オブジェクトの CLUSTER 属性を設定します。
2. クラスター内の他のキュー・マネージャー上の USD および GBP トピック・オブジェクトのコピーをすべて削除します。
3. USD および GBP が、クラスター内のすべてのキュー・マネージャーで SYSTEM.QPUBSUB.SUBPOINT.NAMELIST に定義されていることを確認してください。

単一キュー・マネージャーのパブリッシュ/サブスクライブ構成の例

92 ページの図 31 は、基本的な単一キュー・マネージャーのパブリッシュ/サブスクライブ構成を表しています。 この例はニュース・サービスの構成を示すもので、ここではパブリッシャーからいくつかのトピックについての情報が提供されています。

- パブリッシャー 1 は、トピック「Sport」を使用してスポーツの試合結果の情報をパブリッシュしています。
- パブリッシャー 2 は、トピック「Stock」を使用して株価の情報をパブリッシュしています。
- パブリッシャー 3 はトピック「Films」を使用して映画のレビュー情報をパブリッシュし、トピック「TV」を使用してテレビ番組表をパブリッシュしています。

3 人のサブスクライバーは、それぞれ関心のある別々のトピックに登録しているため、キュー・マネージャーがそれぞれが興味を持っている情報を送信します。

- サブスクライバー 1 はスポーツの試合結果と株価を受信します。
- サブスクライバー 2 は映画のレビューを受信します。
- サブスクライバー 3 はスポーツの試合結果を受信します。

テレビ番組表に登録しているサブスクライバーはいないので、テレビ番組表は配布されません。

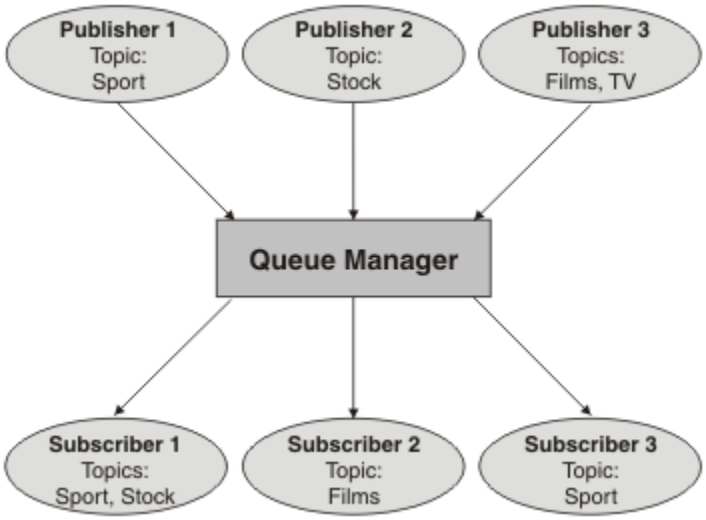


図 *31.* 単一キュー・マネージャーのパブリッシュ/サブスクライブの例

分散パブリッシュ/サブスクライブのネットワーク

各キュー・マネージャーは、トピックにパブリッシュされたメッセージと、そのトピックをサブスクライブしている、ローカルに作成されたサブスクリプションのマッチングを行います。 あるキュー・マネージャーに接続したアプリケーションからパブリッシュされたメッセージが、ネットワーク内の他のキュー・マネージャー上に作成された一致するサブスクリプションに配信されるように、キュー・マネージャーのネットワークを構成することができます。 このためには、キュー・マネージャー間のシンプルなチャネルを介する追加の構成が必要になります。

分散パブリッシュ/サブスクライブ構成とは、一連のキュー・マネージャーを接続したものです。 それらのキュー・マネージャーすべてを同じ物理システム上に配置することもできますし、いくつかの物理システムに分散することも可能です。 キュー・マネージャーどうしを接続すると、サブスクライバーは、1 つのキュー・マネージャーにサブスクライブした状態で、最初は別のキュー・マネージャーにパブリッシュされていたメッセージを受信することができます。 これを例示するため、次の図では、91 ページの『単一キュー・マネージャーのパブリッシュ/サブスクライブ構成の例』で説明した構成にキュー・マネージャーをもう 1 つ追加しています。

- キュー・マネージャー 2 は、パブリッシャー 4 が天気予報情報 (トピック「Weather」を使用) と主要道路の交通情報 (トピック「Traffic」を使用) をパブリッシュするために使用されます。
- サブスクライバー 4 もこのキュー・マネージャーを使用して、トピック「Traffic」を使用した交通情報にサブスクライブします。
- サブスクライバー 3 は、パブリッシャーからの別のキュー・マネージャーを使用しますが、やはり天気情報にサブスクライブします。 キュー・マネージャーが互いにリンクしているので、このようなことが可能になります。

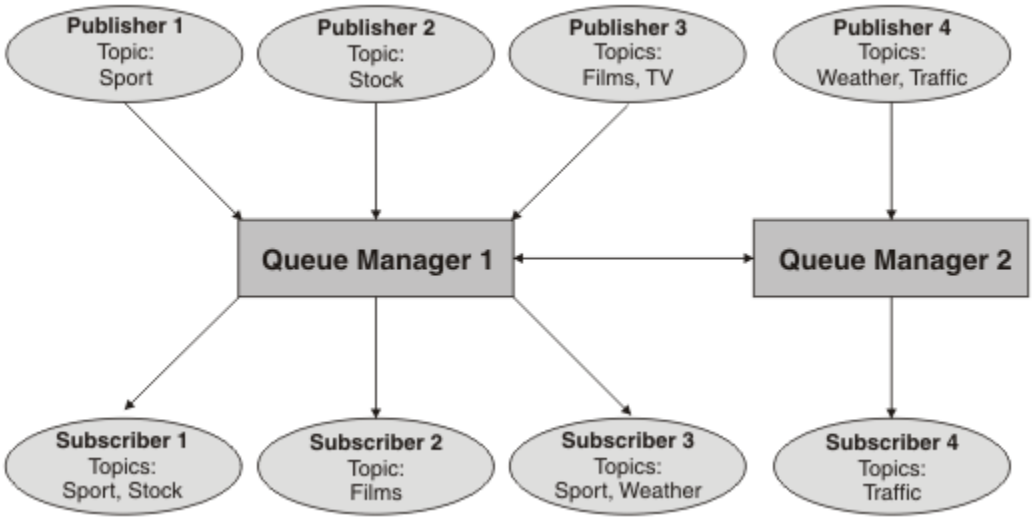


図 *32.* キュー・マネージャーが *2* つあるパブリッシュ/サブスクライブの例

手動によりキュー・マネージャーを親子階層で接続するか、パブリッシュ/サブスクライブ・クラスターを作成して IBM MQ に接続の詳細の大部分を自動定義させることができます。 両方のトポロジーを組み合わせて使用することもできます。 例えば、複数のクラスターを階層にして結合できます。

パブリッシュ/サブスクライブ・クラスターの概要

パブリッシュ/サブスクライブ・クラスターは、標準的なクラスターに、1 つ以上のトピック・オブジェクトが追加されたものです。 クラスター内のいずれかのキュー・マネージャーに管理トピック・オブジェクトを定義し、クラスター名を指定してそのトピック・オブジェクトをクラスター化すると、そのトピックのパブリッシャーとサブスクライバーは、クラスター内の任意のキュー・マネージャーに接続でき、パブリッシュされたメッセージは、キュー・マネージャー間のクラスター・チャネルを介してサブスクライバーにルーティングされます。

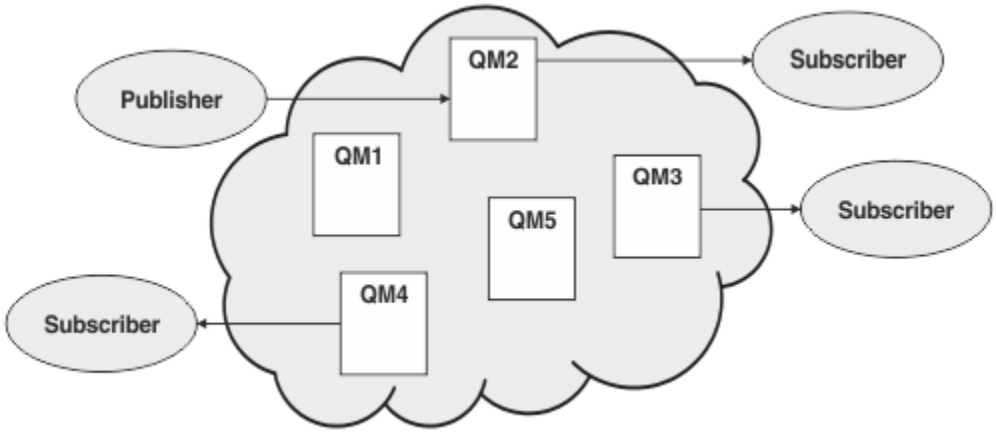


図 *33.* パブリッシュ/サブスクライブ・クラスター

クラスター内でどのようにパブリッシュ/サブスクライブ・メッセージをルーティングするかは、次の 2 つの方法で構成できます。

- 直接ルーティング (direct routing)
- トピック・ホスト・ルーティング (topic host routing)

直接ルーティング型のクラスター・トピックを構成すると、あるキュー・マネージャーでパブリッシュされたメッセージは、そのキュー・マネージャーからクラスター内の他のすべてのキュー・マネージャーのすべてのサブスクリプションに直接送信されます。 こうすると、最短パスによるパブリケーションが可能ですが、クラスター内のすべてのキュー・マネージャーが、そのキュー・マネージャー以外のすべてのキ

ュー・マネージャーを認識するようになるため、それぞれにクラスター・チャネルをキュー・マネージャー間に確立する可能性があります。

トピック・ホスト・ルーティングを使用する場合、あるキュー・マネージャーでパブリッシュされたメッセージは、そこから、管理対象トピック・オブジェクトの定義をホストするキュー・マネージャーに送信されます。 そのトピック・ホスト・キュー・マネージャー が、クラスター内の他のすべてのキュー・マネージャーのすべてのサブスクリプションにメッセージをルーティングします。 トピック・ホスト・キュー・マネージャー上にパブリッシャーまたはサブスクライバーが存在しないと、パブリケーションの経路が長くなります。 しかし、トピック・ホスト・キュー・マネージャーのみが、クラスター内の他のすべてのキュー・マネージャーを認識してキュー・マネージャー間にクラスター・チャネルを確立する可能性があるという利点があります。

詳しくは、95 ページの『パブリッシュ/サブスクライブ・クラスター』 を参照してください。

パブリッシュ/サブスクライブ階層の概要

パブリッシュ/サブスクライブ階層とは、一連のキュー・マネージャーをチャネルで接続して階層構造にしたものです。 パブリッシュ/サブスクライブ階層へのキュー・マネージャーの接続で説明されているように、各キュー・マネージャーはその 親 キュー・マネージャーを識別します。

トピックのパブリッシャーとサブスクライバーは、階層内の任意のキュー・マネージャーに接続できます。また、メッセージは、階層のキュー・マネージャー接続を使用してキュー・マネージャー間を流れます。

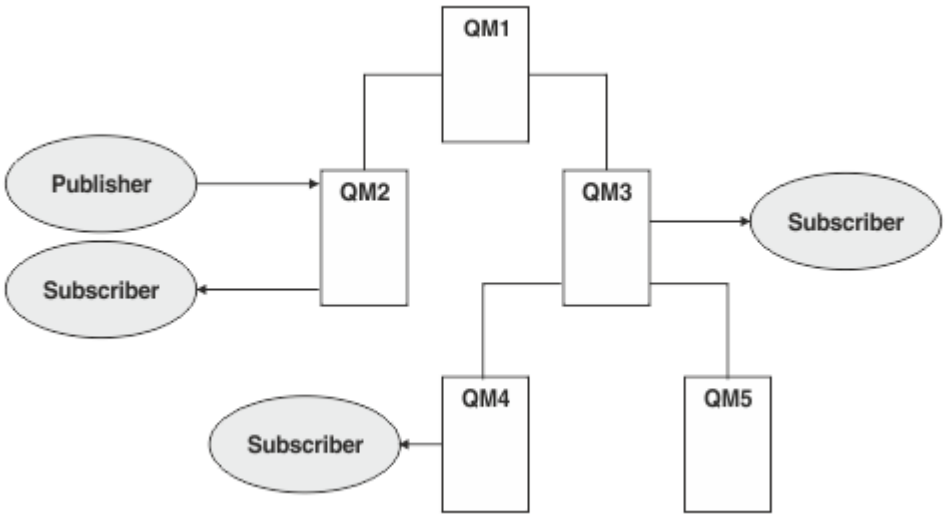


図 *34.* パブリッシュ/サブスクライブ階層

前の図では、QM3 および QM4 のサブスクライバーに配信されたパブリケーションは、QM2 から QM1 にルーティングされた後、QM3 に到達し、最後に QM4 に到達します。

階層を使用すると、階層内のすべてのキュー・マネージャーの関係を直接制御できます。 これは、パブリッシャーからサブスクライバーへのメッセージのルーティングをきめ細かく制御できるため、特に、接続が制限されたキュー・マネージャー・ネットワーク間でルーティングする場合に便利です。 パブリッシャーからサブスクライバーまでのメッセージのルーティング経路にあるすべてのキュー・マネージャーについて、その可用性および能力を慎重に考慮する必要があります。

詳しくは、98 ページの『パブリッシュ/サブスクライブの階層』 を参照してください。

キュー・マネージャー間のパブリケーションの分散

ルーティングの選択に加えて、次の 2 つのアプローチにより、キュー・マネージャーのネットワーク内にパブリケーションを分散させることができます。

- あるキュー・マネージャーからのパブリケーションは、そのパブリケーションのサブスクリプションを現在ホストしているキュー・マネージャーにのみ送信する。

- 各パブリケーションをすべてのキュー・マネージャーに送信し、各キュー・マネージャーにパブリケーションとサブスクリプションのマッチングを行わせる。

前者の場合、パブリケーション・メッセージは必要な場所にのみ送信されますが、サブスクリプションに関するある程度の情報をキュー・マネージャー間で共有する必要があります。 後者の場合、サブスクリプション情報を共有する必要はありませんが、不要なパブリケーション・メッセージがキュー・マネージャー間で送信される可能性があります。

デフォルトでは、IBM MQ は前者の方法を使用するため、パブリケーションは、該当するサブスクリプションを持つキュー・マネージャーにのみ送信されます。 サブスクリプション情報は、プロキシー・サブスクリプション の形でキュー・マネージャー間に伝搬します。 分散パブリッシュ/サブスクライブ・トポロジーにおいて最も使用効率が高い方法は、サブスクリプションの分散状況と存続期間、およびパブリケーションの頻度によって異なります。 『パブリッシュ/サブスクライブ・ネットワークでのサブスクリプションのパフォーマンス』を参照してください。

関連概念

78 ページの『トピック・ツリー』
定義する各トピックは、トピック・ツリー内の要素、つまりノードです。 トピック・ツリーは、最初は空にしておくことも、MQSC または PCF コマンドを使用して既に定義されたトピックを含めることも可能です。 トピック作成コマンドを使用するか、パブリケーションまたはサブスクリプションで初めてトピックを指定することによって、新規トピックを定義できます。

パブリッシュ/サブスクライブ階層のシナリオ

関連タスク

パブリッシュ/サブスクライブ・クラスターの設計

パブリッシュ/サブスクライブ・クラスター

パブリッシュ/サブスクライブ・クラスターとは、キュー・マネージャーを相互接続した標準的なクラスターであり、パブリケーションは、パブリッシュ元のアプリケーションから、クラスター内のキュー・マネージャー上に存在するサブスクリプションに自動的に移動します。 パブリッシュ/サブスクライブ・クラスター内のパブリケーションのルーティングには、直接ルーティング とトピック・ホスト・ルーティング という 2 つの選択肢があります。 どちらのルーティングを選択するかは、クラスターの規模および予測されるアクティビティーのパターンによって決まります。

パブリッシュ/サブスクライブ・メッセージングに使用されるクラスターは、標準 IBM MQ クラスターとまったく変わりはありません。 そのため、パブリッシュ/サブスクライブ・クラスター内のキュー・マネージャーは物理的に別々のコンピューター上に存在でき、キュー・マネージャーの各ペアは、必要に応じてクラスター・チャネルによって相互に接続されます。 詳しくは、クラスターを参照してください。

パブリッシュ/サブスクライブ・メッセージング用に標準的なキュー・マネージャー・クラスターを構成するには、クラスター内の 1 つのキュー・マネージャーに 1 つ以上の管理対象トピック・オブジェクトを定義します。 そのトピックをクラスター・トピックにするには、クラスターの名前を指定して **CLUSTER** プロパティーを構成します。 この構成を行うと、トピック・ツリー内のそのトピック以下の、パブリッシャーまたはサブスクライバーによって使用されるすべてのトピックが、クラスター内のすべてのキュー・マネージャー間で共用されます。また、そのトピック・ツリーのクラスター・ブランチにパブリッシュされたメッセージは、そのクラスター内の他のキュー・マネージャー上のサブスクリプションに自動的にルーティングされます。

ターゲット・キュー・マネージャー上に存在するメッセージのサブスクライバーの数にかかわらず、パブリッシャー・キュー・マネージャーと他の各キュー・マネージャーの間では、メッセージ 1 つにつきコピーが 1 つのみ送信されます。 1 つのキュー・マネージャーに 1 つ以上のサブスクリプションがある場合、メッセージが到達すると、すべてのサブスクリプションに複製されます。

このクラスターに属しているキュー・マネージャーはクラスター・トピックを自動的に認識するようになり、そのキュー・マネージャー上のパブリッシャーおよびサブスクライバーも自動的にクラスターに参加します。

クラスター・トピック・オブジェクト下にないトピック・ストリングで作業することで、非クラスター・パブリッシュ/サブスクライブ・アクティビティーをパブリッシュ/サブスクライブ・クラスターで行うこともできます。

パブリッシュ/サブスクライブ・クラスター内のパブリケーションのルーティングには、直接ルーティングとトピック・ホスト・ルーティング という 2 つの選択肢があります。 クラスター内で使用するメッセージ・ルーティングを選択するには、管理対象トピック・オブジェクトの **CLROUTE** プロパティーを以下のいずれかの値に設定します。

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

デフォルトでは、トピック・ルーティングは **DIRECT** です。 これは IBM MQ 8.0 より前の唯一のオプションでした。 直接経路指定されたクラスター・トピックをキュー・マネージャーで構成すると、クラスター内のすべてのキュー・マネージャーがクラスター内の他のすべてのキュー・マネージャーを認識するようになります。 各キュー・マネージャーは、パブリッシュ操作およびサブスクライブ操作を実行するときに、クラスター内の他のすべてのキュー・マネージャーに直接接続できます。

IBM MQ 8.0 以降、代わりにトピック・ルーティングを **TOPICHOST** として構成できるようになりました。トピック・ホスト経路指定を使用すると、クラスター内のすべてのキュー・マネージャーは、経路指定されたトピック定義をホストするクラスター・キュー・マネージャー (つまり、トピック・オブジェクトを定義したキュー・マネージャー) を認識するようになります。 パブリッシュ操作およびサブスクライブ操作を行うとき、クラスター内のキュー・マネージャーは、それらのトピック・ホスト・キュー・マネージャーにのみ接続し、相互に直接接続されることはありません。 トピック・ホスト・キュー・マネージャーは、パブリケーションがパブリッシュされるキュー・マネージャーから、一致するサブスクリプションがあるキュー・マネージャーへのパブリケーションの経路指定を担当します。

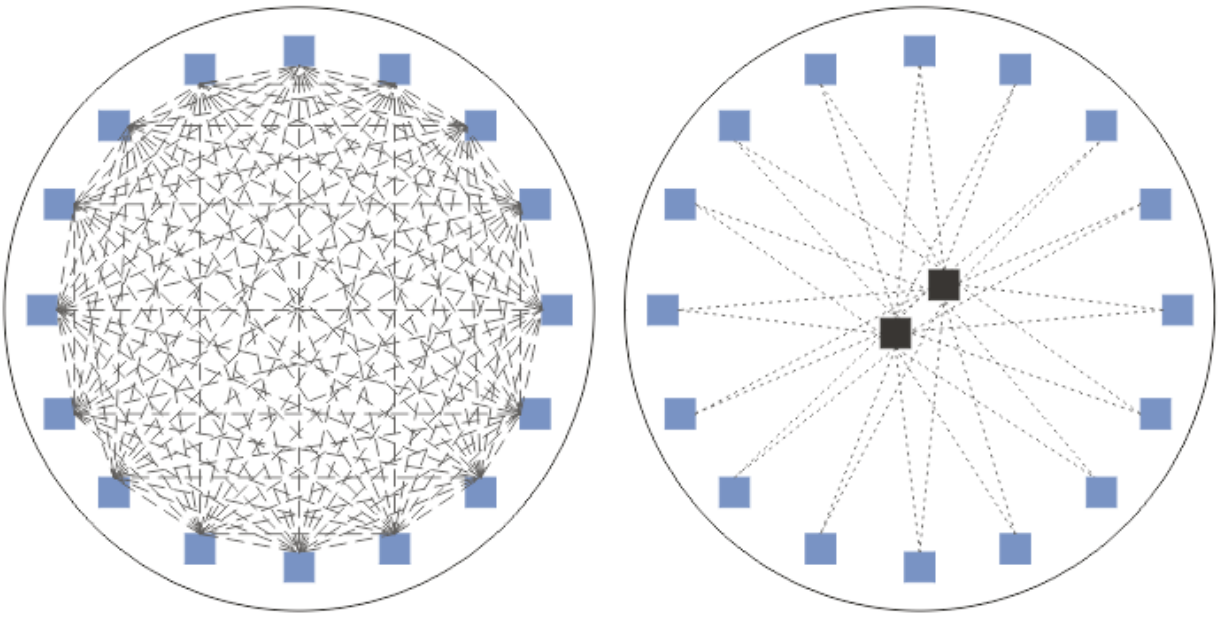


図 *35.* 直接ルーティングとトピック・ホスト・ルーティング

直接ルーティングの概要

管理対象トピック・オブジェクトに直接ルーティングを構成する場合、トピック・オブジェクトは、クラスター内のいずれかのキュー・マネージャーに定義するだけで、すべてのキュー・マネージャーから認識されるようになります。 どのキュー・マネージャーにトピックを定義しようと、そのトピックのパブリッシュ/サブスクライブ・メッセージングの動作には影響しません。

各メッセージは、中間キュー・マネージャーを経由して渡されるのではなく、パブリッシャー・キュー・マネージャーからクラスター内の他のキュー・マネージャー上の各サブスクリプションに直接流れます。

デフォルトでは、クラスター内でサブスクリプションを 1 つ以上ホストする他のキュー・マネージャーにのみ、メッセージが送信されます。

- これは、各キュー・マネージャーが、そこへの 1 つ以上のサブスクリプションが現在存在するすべてのトピックの情報を、クラスター内の他のすべてのキュー・マネージャーに直接通知することによって成り

立っています。 その結果、クラスター内のすべてのキュー・マネージャーが、サブスクライブされているすべてのトピックを認識するようになり、サブスクリプションをホストするすべてのキュー・マネージャーが、他のすべてのキュー・マネージャーに対してチャネルを確立します。 これは、各キュー・マネージャーにパブリッシャーが存在しているかどうかに関係なく行われます。

- サブスクライブされている個々のトピックについての情報をすべてのキュー・マネージャーで認識するという動作が発生しないようにするには、サブスクリプションが存在するかどうかにかかわらず、クラスター内のすべてのキュー・マネージャーにすべてのパブリケーションを送信するという方式に変更します。 こうすると、サブスクリプションに関する情報のトラフィックは削減されますが、パブリケーションのトラフィック、および各キュー・マネージャーが確立するチャネルの数は増加する可能性があります。 『パブリッシュ/サブスクライブ・ネットワークでのサブスクリプションのパフォーマンス』を参照してください。

直接ルーティングのクラスター・トピックを使用するパブリッシュ/サブスクライブのメッセージ・フローは、複数のパブリッシュ/サブスクライブ・クラスターをまたぐように設定することもできます。 これを行うには、各クラスターの中からそれぞれ 1 つのキュー・マネージャーをパブリッシュ/サブスクライブ階層に追加します。 複数のクラスターのトピック・スペースの結合を参照してください。

直接ルーティングの詳細な説明については、パブリッシュ/サブスクライブ・クラスターの直接ルーティングを参照してください。

トピック・ホスト・ルーティングの概要

管理対象トピック・オブジェクトにトピック・ホスト・ルーティングを構成すると、クラスター内のキュー・マネージャーのパブリケーションは、そのトピック・オブジェクトが構成されているキュー・マネージャー (トピック・ホスト) を経由し、そこからサブスクリプションが存在するキュー・マネージャーにルーティングされます。

- これは、各キュー・マネージャーが、そこへの 1 つ以上のサブスクリプションが現在存在するすべてのトピックの情報を、すべてのトピック・ホストに通知することによって成り立っています。 サブスクリプションをホストするすべてのキュー・マネージャーは、そのサブスクリプションに関係するトピックのすべてのトピック・ホストに対してチャネルを確立します。
- 非トピック・ホスト・キュー・マネージャーは、パブリッシュ/サブスクライブの目的でクラスター内の他の非トピック・ホスト・キュー・マネージャーを認識することはなく、それらのキュー・マネージャーとの間でこの目的のためのチャネルが確立されることもありません。
- パブリッシュ元のアプリケーションが、そのトピックをホストするキュー・マネージャーに接続されている場合、パブリッシュされたメッセージは、マッチングするサブスクリプションが作成されているキュー・マネージャーに直接ルーティングされるため、余分な「ホップ」は発生しません。 同様に、マッチングするサブスクリプションが、そのトピックをホストする 1 つのキュー・マネージャー上のみで作成されている場合、そのトピックにパブリッシュされるメッセージはそのキュー・マネージャーに直接ルーティングされるため、余分なホップは発生しません。
- パブリッシャーと同じキュー・マネージャー上にサブスクリプションが存在する場合は、最初にパブリケーションをトピック・オブジェクトのホストにルーティングする必要はありません。

クラスター・キューについては、複数のキュー・マネージャーが同じ管理トピック・オブジェクトを構成することができます。 これにより、メッセージ・ルーティングの可用性が高くなり、ワークロード・バランスを維持した水平方向のスケーラビリティーが得られます。 トピック・ホスト・ルーティングを使用するトピック・オブジェクトでは、複数のキュー・マネージャーで、同じ名前のトピックをトピック・ツリーの同じブランチに構成すると、各トピック・ホストは、サブスクリプションをホストするそれぞれのキュー・マネージャーでサブスクライブされるトピックを認識します。

- パブリッシュされたメッセージは、いずれかのトピック・ホスト・キュー・マネージャーに送信されてから、サブスクリプションをホストするキュー・マネージャーに転送されます。 どのトピック・ホスト・キュー・マネージャーが選択されるかは、Point-to-Point クラスター・キューの場合と同じデフォルトのワークロード・バランシング・ルールに従います。
- 1 つ以上のトピック・ホスト・キュー・マネージャーにパブリッシュ元のキュー・マネージャーから通信できない場合、他の使用可能なトピック・ホスト・キュー・マネージャーにメッセージがルーティングされます。

トピック・ツリーのルーティング対象のブランチに含まれているトピックへのすべてのパブリケーションは、そのトピックへのサブスクリプションがクラスター内のどこにも存在しない場合であっても、トピック・ホストの 1 つに転送されます。 デフォルトでは、クラスター内でサブスクリプションを 1 つ以上ホストする他のキュー・マネージャーにのみ、ここからメッセージが送信されます。

- これは、クラスター内の各キュー・マネージャー上のサブスクライブされているすべてのトピック・ストリングが、各トピック・ホスト・キュー・マネージャーに通知されることによって成り立っています。
- サブスクライブされている個々のトピックについての情報を認識するという動作が発生しないようにするには、サブスクリプションが存在するかどうかにかかわらず、クラスター内のすべてのキュー・マネージャーに、トピック・ホストにルーティングされたすべてのパブリケーションを送信するという方式に変更します。 こうすると、サブスクリプションに関する情報のトラフィックは削減されますが、パブリケーションのトラフィック、および各トピック・ホスト・キュー・マネージャーとの間に確立されるチャネルの数は増加する可能性があります。 『パブリッシュ/サブスクライブ・ネットワークでのサブスクリプションのパフォーマンス』を参照してください。

トピック・ホスト・ルーティングのクラスター・トピックを使用するパブリッシュ/サブスクライブのメッセージ・フローを、パブリッシュ/サブスクライブ階層を使用して複数のパブリッシュ/サブスクライブ・クラスターをまたぐフローにすることは**できません**。

トピック・ホスト・ルーティングの詳細な説明については、パブリッシュ/サブスクライブ・クラスターのトピック・ホスト・ルーティングを参照してください。

パブリッシュ/サブスクライブの階層

パブリッシュ/サブスクライブ階層を構築するには、チャネルを使用してキュー・マネージャーどうしをリンクした後に、キュー・マネージャーのペアの間に親子関係を定義します。 メッセージは、パブリッシャーから、階層内の直接的な関係を介してサブスクリプションまで流れます。 これは、複数の"ホップ"を意味する場合があることに注意してください。

ターゲット・キュー・マネージャー上に存在するメッセージのサブスクライバーの数にかかわらず、1 組のキュー・マネージャーの間で送信されるメッセージのコピーは 1 つのみです。 1 つのキュー・マネージャーに 1 つ以上のサブスクリプションがある場合、メッセージが到達すると、すべてのサブスクリプションに複製されます。

デフォルトでは、別のキュー・マネージャー上のサブスクリプションまでのルート上にある、階層内の他のキュー・マネージャーにのみメッセージが送信されます。

- これは、各キュー・マネージャーが、このキュー・マネージャー上またはこれに関係があるいずれかのキュー・マネージャー上に存在するトピックの情報のうち、1 つ以上のサブスクリプションが現在存在するものすべてを、直接関係する他のそれぞれのキュー・マネージャーに通知することによって成り立っています。 この結果、階層内のすべてのキュー・マネージャーが、サブスクライブされているすべてのトピックを認識するようになります。
- この動作を変更して、サブスクリプションの存在にかかわらず、常に階層内のすべてのキュー・マネージャーにパブリケーションを送信することもできます。 このようにした場合、サブスクリプション情報を階層の中で伝搬させる必要はなくなりますが、パブリケーションのトラフィックが増加する可能性があります。

クラスターを作成する場合は、ネットワーク内をメッセージが無限に循環するループを生成しないように注意する必要があります。 階層では、このようなループは生成されません。

キュー・マネージャーの名前はすべて固有でなければなりません。

パブリッシュ/サブスクライブのメッセージ・フローは、複数のパブリッシュ/サブスクライブ・クラスターをまたぐことができます。 これを行うには、各クラスターの中からそれぞれ 1 つのキュー・マネージャーをパブリッシュ/サブスクライブ階層に追加します。

詳細な説明については、パブリッシュ/サブスクライブ階層のルーティングを参照してください。

パブリッシュ/サブスクライブ・ネットワークでのプロキシー・サブスクリプション

プロキシー・サブスクリプションは、あるキュー・マネージャーによってパブリッシュされるトピックに関する、別のキュー・マネージャーによって行われるサブスクリプションです。 プロキシー・サブスクリプションは、サブスクリプションによってサブスクライブされている各トピック・ストリングのキュー・マネージャーの間で流れます。 ユーザーはプロキシー・サブスクリプションを明示的に作成しません。キュー・マネージャーが代わりにそれを行います。

キュー・マネージャーをまとめて、パブリッシュ/サブスクライブ・クラスターまたはパブリッシュ/サブスクライブ階層に接続できます。 プロキシー・サブスクリプションは、接続されているキュー・マネージャーの間で流れます。 プロキシー・サブスクリプションによって、あるキュー・マネージャーに接続されているパブリッシャーが作成したトピックへのパブリケーションを、他のキュー・マネージャーに接続されているそのトピックへのサブスクライバーが受け取ります。 92 ページの『分散パブリッシュ/サブスクライブのネットワーク』を参照。

パブリッシュ/サブスクライブ・トポロジーが個々のトピック・ストリングへの何千ものサブスクリプションを処理する場合、またはそのようなサブスクリプションの存在が急激に変化している可能性がある場合、プロキシー・サブスクリプション伝搬のオーバーヘッドについて検討する必要があります。 このトピックの残りの部分で説明する自動集約に加えて、手動の構成変更を行うことで、接続されたキュー・マネージャー間のプロキシー・サブスクリプションとパブリケーションのフローをさらに制限し、接続されたすべてのキュー・マネージャーにプロキシー・サブスクリプションが伝搬されるまでの待機遅延を縮小できます。 『パブリッシュ/サブスクライブ・ネットワークでのサブスクリプションのパフォーマンス』を参照してください。

プロキシー・サブスクリプションには、ローカル・サブスクリプションで使用されるセレクターは含まれないため、ワイルドカードを指定したサブスクリプション・トピック・ストリングが単純化される可能性があります。 その結果、実際のサブスクリプションにはマッチングしないパブリケーションがプロキシー・サブスクリプションとマッチングし、キュー・マネージャー間に余分のパブリケーション・フローが発生する可能性があります。 このような矛盾は、サブスクリプションをホストするキュー・マネージャーがフィルタリングにより除外するため、余分のパブリケーションはサブスクリプションには返されません。

プロキシー・サブスクリプションの集約

プロキシー・サブスクリプションは、重複除去システムを使用して集約されます。 特定の解決されたトピック・ストリングで、最初のローカル・サブスクリプションまたは最初に受け取ったプロキシー・サブスクリプションに関して、プロキシー・サブスクリプションが送信されます。 同じトピック・ストリングに対する後続のサブスクリプションは、その既存のプロキシー・サブスクリプションを利用します。

このプロキシー・サブスクリプションは、最後のローカル・サブスクリプションまたは最後に受け取ったプロキシー・サブスクリプションが取り消された後に取り消されます。

パブリケーションの集約

あるキュー・マネージャーに同じトピック・ストリングへの複数のサブスクリプションが存在する場合、そのトピック・ストリングに一致する各パブリケーションの単一コピーのみが、パブリッシュ/サブスクライブ・トポロジーのその他のキュー・マネージャーから送信されます。 メッセージが到着すると、ローカル・キュー・マネージャーはメッセージのコピーを一致する各サブスクリプションに送信します。

プロキシー・サブスクリプションにワイルドカードが含まれている場合、1 つのパブリケーションのトピック・ストリングに複数のプロキシー・サブスクリプションが一致する可能性があります。 接続されている 1 つのキュー・マネージャーによって作成された複数のプロキシー・サブスクリプションに一致するメッセージが、キュー・マネージャーでパブリッシュされた場合、複数のプロキシー・サブスクリプションを満たすために、パブリケーションの 1 つのコピーだけがリモート・キュー・マネージャーに転送されます。

関連概念

分散パブリッシュ/サブスクライブ・ネットワークでのループ検出

プロキシー・サブスクリプション内のワイルドカード

サブスクリプションでは、トピック・ストリングにワイルドカードを使用してパブリケーション内の複数のトピック・ストリングに突き合わせることができます。

サブスクリプションで使用できるワイルドカードのスキーマには、トピック・ベース と文字ベース の 2 つがあります。 『72 ページの『ワイルドカード・スキーマ』』を参照してください。

IBM MQ では、ワイルドカード・サブスクリプションのすべてのプロキシー・サブスクリプションは、トピック・ベースのワイルドカードを使用するように変換されます。 文字ベースのワイルドカードが検出されると、そのワイルドカードは、直前の / までさかのぼって # 文字に置き換えられます。 例えば、/aaa/bbb/c*d は /aaa/bbb/# に変換されます。 この変換の結果、リモート・キュー・マネージャーは、明示的にサブスクライブされた場合よりも少し多めにパブリケーションを送信することになります。追加のパブリケーションは、ローカル・キュー・マネージャーがそのローカル・サブスクライバーにパブリケーションを送信するときに、ローカル・キュー・マネージャーによってフィルターに掛けられます。

WILDCARD プロパティーによるワイルドカードの使用の制御

MQSC **Topic** WILDCARD プロパティーまたは 同等の PCF Topic `WildcardOperation` プロパティーを使用すると、 ワイルドカード・トピック・ストリング名を使用するサブスクライバー・アプリケーションへのパブリケーションの送達を制御できます。 WILDCARD プロパティーには、以下の 2 つの値のいずれかを指定できます。

WILDCARD
このトピックに対するワイルドカード・サブスクリプションの動作。

PASSTHRU
このトピック・オブジェクトのトピック・ストリングよりも具体的でないワイルドカード・トピックに対するサブスクリプションは、そのトピックまたはそのトピックよりも具体的なトピック・ストリングに対するパブリケーションを受信できるようになります。

BLOCK
このトピック・オブジェクトのトピック・ストリングよりも具体的でないワイルドカード・トピックに対するサブスクリプションは、このトピックまたはこのトピックよりも具体的なトピック・ストリングに対するパブリケーションを受信できなくなります。

サブスクリプションが定義されている場合に、この属性の値が使用されます。 この属性を変更しても、既存のサブスクリプションによってカバーされているトピック・セット は、変更による影響を受けません。 このシナリオは、トピック・オブジェクトが作成または削除されてトポロジーが変更された場合にも当てはまります。WILDCARD 属性の変更後に作成されたサブスクリプションに一致するトピックのセットは、変更後のトポロジーを使用して作成されます。 既存のサブスクリプションについて、一致するトピック・セットを強制的に再評価する場合は、キュー・マネージャーを再開する必要があります。

86 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターを作成する』 の例では、 82 ページの図 23 で示されるトピック・ツリー構造を作成するステップに従うことができます。

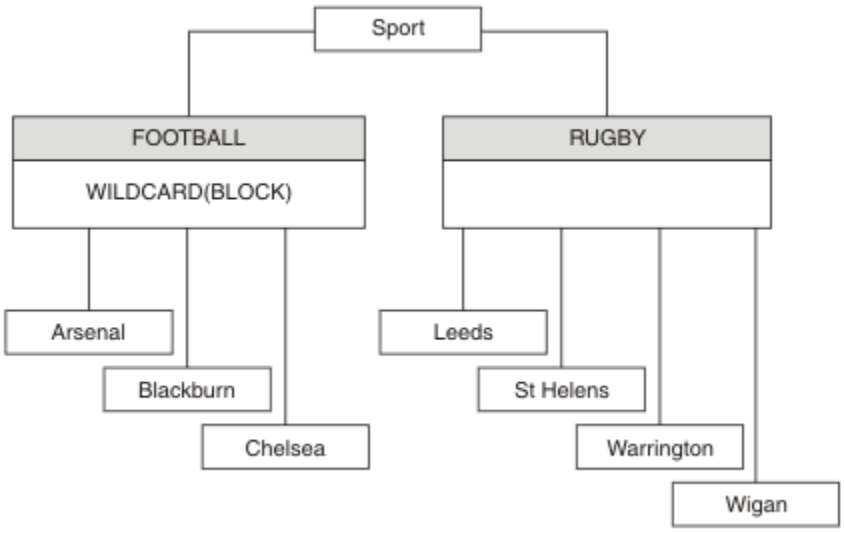


図 *36. WILDCARD* プロパティー *BLOCK* を使用するトピック・ツリー

ワイルドカード・トピック・ストリング *#* を使用するサブスクライバーは、 Sport トピックと Sport/Rugby サブツリーへのすべてのパブリケーションを受け取ります。 Sport/Football トピックの WILDCARD プロパティー値が BLOCK であるため、このサブスクライバーは Sport/Football サブツリーへのパブリケーションは受け取りません。

PASSTHRU は、デフォルトの設定値です。 Sport ツリーのノードには、WILDCARD プロパティー値 PASSTHRU を設定できます。 ノードで WILDCARD プロパティー値 BLOCK が設定されていない場合、PASSTHRU を設定しても、Sports ツリーのノードのサブスクライバーによって 観測される動作が変化することはありません。

この例では、サブスクリプションを作成して、 送達されるパブリケーションにワイルドカード設定が与える影響を確認します。 87 ページの図 27 を参照してください。 88 ページの図 30 でパブリッシュ・コマンドを実行して、 パブリケーションをいくつか作成します。

```
pub QMA
```

図 *37. QMA* へのパブリッシュ

83 ページの表 3 では結果が示されています。 WILDCARD プロパティー値 BLOCK を設定すると、ワイルドカードを含むサブスクリプションは、 そのワイルドカードの有効範囲内にあるトピックへのパブリケーションを受信しなくなることに注意してください。

表 *6. QMA* で受信されたパブリケーション

サブスクリプション	トピック・ストリング	受信されたパブリケーション	注
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	「Football」サブツリーへのすべてのパブリケーションは、 Sports/Football の WILDCARD(BLOCK) によってブロックされます。
SARSENAL	Sports/#/Arsenal	-	Sports/Football の WILDCARD(BLOCK) により、Arsenal でのワイルドカード・サブスクリプションは防止されます。

表 6. QMA で受信されたパブリケーション (続き)			
サブスクリプション	**トピック・ストリング**	**受信されたパブリケーション**	**注**
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	「Sports/Rugby」のデフォルトである WILDCARD は、Leeds でのワイルドカード・サブスクリプションを防止しません。

注:

あるサブスクリプションに、WILDCARD プロパティー値 BLOCK を持つトピック・オブジェクトに一致するワイルドカードがあるとします。 このサブスクリプションで、一致するワイルドカードの右側にトピック・ストリングもある場合、 サブスクリプションがパブリケーションを受信することはありません。 ブロックされないパブリケーションのセットは、 ブロックされたワイルドカードの親であるトピックへのパブリケーションです。 BLOCK プロパティー値を持つトピックの子であるトピックへのパブリケーションは、ワイルドカードによってブロックされます。 したがって、ワイルドカードの右側にトピックが含まれるサブスクリプション・トピック・ストリングは、 一致するパブリケーションを受信することがありません。

WILDCARD プロパティー値を BLOCK に設定しても、ワイルドカードが含まれるトピック・ストリングを使用したサブスクライブが実行できなくなるわけではありません。 このようなサブスクリプションは正常に行われます。 このサブスクリプションには、 WILDCARD プロパティー値 BLOCK を持つトピック・オブジェクトを含むトピックに一致する 明示的なトピックが含まれます。 これは、WILDCARD プロパティー値 BLOCK を持つトピックの親または子であるトピック用に、 ワイルドカードを使用します。 82 ページの図 23 の例では、 Sports/Football/# のようなサブスクリプションがパブリケーションを受信できます。

ワイルドカードとクラスター・トピック

クラスター・トピック定義はクラスター内のすべてのキュー・マネージャーに伝搬されます。 クラスター内のキュー・マネージャーでクラスター・トピックへのサブスクリプションを行うと、 そのキュー・マネージャーで複数のプロキシー・サブスクリプションが作成されます。 クラスター内の他のすべてのキュー・マネージャーで 1 つのプロキシー・サブスクリプションが作成されます。 ワイルドカードを含むトピック・ストリングを使用したサブスクリプションをクラスター・トピックと組み合わせると、 動作の予測が難しくなる可能性があります。 次の例は、この動作について説明しています。

86 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターを作成する』 の例にあるクラスター・セットアップでは、QMB が QMA と同じサブスクリプションのセットを持っているにもかかわらず、パブリッシャーが QMA にパブリッシュした後で QMB はパブリケーションを 受信しませんでした (82 ページの図 24 を参照)。 Sports/Football と Sports/Rugby のトピックは クラスター・トピックですが、fullsubs.tst で定義されているサブスクリプションは クラスター・トピックを参照しません。 QMB から QMA に伝搬されるプロキシー・サブスクリプションはありません。 プロキシー・サブスクリプションがないと、QMA へのパブリケーション は QMB に転送されません。

Sports/#/Leeds などの一部のサブスクリプションは、クラスター・トピック (このケースでは Sports/Rugby) を参照するように見えることがあります。 実際には、Sports/#/Leeds サブスクリプション はトピック・オブジェクト SYSTEM.BASE.TOPIC に解決されます。

Sports/#/Leeds などのサブスクリプションによって参照されるトピック・オブジェクトを解決するための規則は、 以下のとおりです。 トピック・ストリングが最初のワイルドカードの位置まで切り捨てられます。 トピック・ストリングを左方向にスキャンし、 関連付けられた管理トピック・オブジェクトを持つ最初のトピックを探します。 そのトピック・オブジェクトがクラスター名を指定するか、ローカル・トピック・オブジェクトを定義している可能性があります。 Sports/#/Leeds の例では、切り捨て後のトピック・ストリングは Sports であり、 トピック・オブジェクトを持たないため、Sports/#/Leeds は ローカル・トピック・オブジェクトである SYSTEM.BASE.TOPIC から継承します。

クラスター化されたトピックのサブスクライブによってワイルドカード伝搬の動作が変更される仕方を確認するには、 バッチ・スクリプト upsubs.bat を実行します。 このスクリプトはサブスクリプション・キ

ューをクリアし、 fullsubs.tst 内にクラスター・トピック・サブスクリプションを追加します。 パブリケーションのバッチを作成するには、 puba.bat を再び実行します (82 ページの図 24 を参照してください)。

84 ページの表 4 は、パブリケーションがパブリッシュされたのと同じキュー・マネージャーに 2 つの新規サブスクリプションを追加した結果を示しています。 結果は予測どおりであり、新規サブスクリプションはそれぞれ 1 つのパブリケーションを受信し、 他のサブスクリプションによって受信されるパブリケーションの数は変更されません。 他のクラスター・キュー・マネージャーでは予測しない結果が発生します。 85 ページの表 5 を参照してください。

表 7. *QMA* で受信されたパブリケーション

サブスクリプション	トピック・ストリング	受信されたパブリケーション	注
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	「Football」サブツリーへのすべてのパブリケーションは、 Sports/Football の WILDCARD(BLOCK) によってブロックされます。
SARSENAL	Sports/#/Arsenal	-	Sports/Football の WILDCARD(BLOCK) により、 Arsenal でのワイルドカード・サブスクリプションは防止されます。
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	「Sports/Rugby」のデフォルトである WILDCARD は、Leeds でのワイルドカード・サブスクリプションを防止しません。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	サブスクリプションにワイルドカードがないため、 Arsenal はパブリケーションを受信します。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds は、いかなるイベントでもパブリケーションを受信します。

85 ページの表 5 は、 QMB で 2 つの新規サブスクリプションを追加して QMA でパブリッシュした結果を示しています。 これら 2 つの新規サブスクリプションがないときに QMB はパブリケーションを受信しませんでした。 Sports/FootBall と Sports/Rugby はどちらもクラスター・トピックであるため、 これら 2 つの新規サブスクリプションは予測どおりパブリケーションを受信します。 QMB から Sports/Football/Arsenal と Sports/Rugby/Leeds 用のプロキシー・サブスクリプション が QMA に転送された後、 そこからパブリケーションが QMB に送信されました。

予測しなかった結果として、 以前はパブリケーションを受信しなかった 2 つのサブスクリプション (Sports/# と Sports/#/Leeds) が、 パブリケーションを受信するようになりました。 これは、他のサブスクリプション用に QMB に転送されたパブリケーションである Sports/Football/Arsenal と Sports/Rugby/Leeds が、QMB に接続された任意のサブスクライバーから使用可能になったためです。 その結果、ローカル・トピックの Sports/# と Sports/#/Leeds へのサブスクリプションは、 Sports/Rugby/Leeds パブリケーションを受信します。 「Sports/Football」は独自の WILDCARD プロパティー値が BLOCK に設定されているため、Sports/#/Arsenal は引き続きパブリケーションを受信しません。

表 *8. QMB* で受信されたパブリケーション			
サブスクリプション	**トピック・ストリング**	**受信されたパブリケーション**	**注**
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*WILDCARD(BLOCK) on Sports/Football* によってブロックされる、フットボール・サブツリーへのすべてのパブリケーション
SARSENAL	Sports/#/Arsenal	-	Sports/Football の WILDCARD(BLOCK) により、Arsenal でのワイルドカード・サブスクリプションは防止されます。
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	「*Sports/Rugby*」のデフォルトである *WILDCARD* は、*Leeds* でのワイルドカード・サブスクリプションを防止しません。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	サブスクリプションにワイルドカードがないため、 Arsenal はパブリケーションを受信します。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds は、いかなるイベントでもパブリケーションを受信します。

ほとんどのアプリケーションにおいて、 あるサブスクリプションが別のサブスクリプションの動作に影響することは望ましくありません。 WILDCARD プロパティーの値 BLOCK の重要な使用法の 1 つは、 ワイルドカードを含む同じトピック・ストリングへのサブスクリプションをすべて同じように動作させることです。 サブスクリプションがパブリッシャーと同じキュー・マネージャー上にあるか別のキュー・マネージャー上にあるかにかかわらず、 サブスクリプションの結果は同じです。

ワイルドカードとストリーム

パブリッシュ/サブスクライブ API に書き込まれた新規アプリケーションの場合、結果として、* へのサブスクリプションがパブリケーションを受信しなくなります。「Sports」へのすべてのパブリケーションを受信するには、Sports/* または Sports/# にサブスクライブするとともに、Business パブリケーションについても同様の処理を行う必要があります。

既存のキュー型パブリッシュ/サブスクライブ・アプリケーションの動作は、パブリッシュ/サブスクライブ・ブローカーを新しいバージョンの IBM MQ に移行しても変更されません。 **Publish**、**Register Publisher**、または **Subscriber** コマンドの **StreamName** プロパティーは、ストリームの移行先のトピックの名前にマップされます。

ワイルドカードとサブスクリプション・ポイント

パブリッシュ/サブスクライブ API に書き込まれた新規アプリケーションの場合、 移行の結果、* へのサブスクリプションがパブリケーションを受信しなくなります。「Sports」へのすべてのパブリケーションを受信するには、Sports/* または Sports/# にサブスクライブするとともに、Business パブリケーションについても同様の処理を行う必要があります。

既存のキュー型パブリッシュ/サブスクライブ・アプリケーションの動作は、パブリッシュ/サブスクライブ・ブローカーを新しいバージョンの IBM MQ に移行しても変更されません。 **Publish**、**Register Publisher**、または **Subscriber** コマンドの **SubPoint** プロパティーは、サブスクリプションの移行先のトピックの名前にマップされます。

例: Sport パブリッシュ/サブスクライブ・クラスターを作成する

以降のステップでは、クラスター CL1 と ともに 4 つのキュー・マネージャー (CL1A および CL1B という 2 つの全リポジトリーと、 QMA および QMB という 2 つの部分リポジトリー) を作成します。 全リポジトリーは、クラスター定義を保持するためにのみ使用されます。 QMA は、クラスター・トピック・ホストに指定されます。 永続サブスクリプションは QMA と QMB の両方で定義されます。

注 : この例は、Windows 用にコーディングされています。 他のプラットフォームでこの例を構成してテストするには、Create qmgrs.bat と create pub.bat を再コーディングする必要があります。

1. 以下のスクリプト・ファイルを作成します。
   a. Create topics.tst
   b. Create wildsubs.tst
   c. Create fullsubs.tst
   d. Create qmgrs.bat
   e. create pub.bat
2. Create qmgrs.bat を実行して構成を作成します。

```
qmgrs
```

82 ページの図 23 でトピックを作成します。 図 5 のスクリプトは、クラスター・トピック Sports/Football および Sports/Rugby を作成します。

注 : REPLACE オプションは、トピックの TOPICSTR プロパティーを置き換えません。 TOPICSTR は、この例でさまざまなトピック・ツリーをテストするために役立つプロパティーです。 トピックを変更するには、最初にトピックを削除します。

```
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')
```

図 *38.* トピックの削除と作成*: topics.tst*

注 : REPLACE によってトピック・ストリングが置き換えられることはないため、トピックを削除します。

ワイルドカードが含まれるサブスクリプションを作成します。 ワイルドカードは、82 ページの図 23 で示されているトピック・オブジェクトを持つトピックに対応します。 各サブスクリプション用のキューを作成します。 スクリプトが実行または再実行されると、キューがクリアされてサブスクリプションは削除されます。

注 : REPLACE オプションによって サブスクリプションの TOPICOBJ または TOPICSTR プロパティーが置き換えられることはありません。 TOPICOBJ または TOPICSTR は、さまざまなサブスクリプションをテストするために変更される便利なプロパティーです。 これらを変更するには、最初にサブスクリプションを削除します。

```
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)
```

図 *39.* ワイルドカード・サブスクリプションの作成*: wildsubs.tst*

クラスター・トピック・オブジェクトを 参照するサブスクリプションを作成します。

注 :

TOPICOBJ によって参照されるトピック・ストリング と TOPICSTR によって定義されるトピック・ストリングの間には、 デリミッター / が自動的に挿入されます。

定義 DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) は、同じサブスクリプションを作成します。 TOPICOBJ は、すでに定義したトピック・ストリングを迅速に参照する方法として使用されます。 サブスクリプションは作成された後、トピック・オブジェクトを参照しなくなります。

```
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)
```

図 *40.* サブスクリプションの削除と作成*: fullsubs.tst*

2 つのリポジトリーが含まれる クラスターを作成します。 パブリッシュとサブスクライブ用に 2 つの部分リポジトリーを作成します。 すべてを削除してやり直すには、スクリプトを再実行します。 このスクリプトでは、トピック階層と初期ワイルドカード・サブスクリプションも作成されます。

注 :

他のプラットフォームでは、同様のスクリプトを作成するか、すべてのコマンドを入力します。 スクリプトを使用すると、迅速にすべてを削除して同一の構成でやり直すことができます。

```
@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
```

図 *41.* キュー・マネージャーの作成*: qmgrs.bat*

サブスクリプションをクラスター・トピックに追加して、構成を更新します。

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

図 *42.* サブスクリプションの更新*: upsubs.bat*

パブリケーション・トピック・ストリングが含まれる メッセージをパブリッシュするには、キュー・マネージャーをパラメーターとして pub.bat を実行します。 Pub.bat は、サンプル・プログラム **amqspub** を使用します。

```
@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
```

図 *43.* パブリッシュ*: pub.bat*

関連概念

ワイルドカード・サブスクリプションと保存パブリケーション

パブリケーション有効範囲

パブリッシュ/サブスクライブ・クラスターまたはパブリッシュ/サブスクライブ階層を構成する場合、パブリケーションの有効範囲により、キュー・マネージャーがリモート・キュー・マネージャーにパブリッシュを転送するかどうかをさらに制御することができます。 パブリケーションの有効範囲の管理には、**PUBSCOPE** トピック属性を使用します。

パブリケーションがリモート・キュー・マネージャーに転送されない場合、ローカル・サブスクライバーだけがそのパブリケーションを受け取ります。

パブリッシュ/サブスクライブ・クラスターを使用する場合、パブリケーションの有効範囲は、主に、トピック・ツリーの特定の場所にあるクラスター・トピック・オブジェクトの定義によって制御されます。 パブリケーションの有効範囲を設定するときには、パブリケーションがクラスター内の他のキュー・マネージャーに流れるようにしなければなりません。 クラスター・トピックのパブリケーションの有効範囲を制限するのは、特定のキュー・マネージャー上の特定のトピックを細かく制御する必要がある場合だけにしてください。

パブリッシュ/サブスクライブ階層を使用する場合、パブリケーションの有効範囲は、主に、この属性とサブスクリプションの有効範囲属性の組み合わせによって制御されます。

PUBSCOPE 属性は、特定のトピックに対して行われるパブリケーションの有効範囲を判別するために使用されます。 この属性には以下のいずれかの値を設定できます。

QMGR
: パブリケーションは、ローカル・サブスクライバーだけに送信されます。 これらのパブリケーションはローカル・パブリケーション と呼ばれます。 ローカル・パブリケーションはリモート・キュー・マネージャーに転送されないため、リモート・キュー・マネージャーに接続されたサブスクライバーはそれを受信しません。

ALL
: パブリケーションの送信先は、パブリッシュ/サブスクライブ・クラスターまたはパブリッシュ/サブスクライブ階層内のリモート・キュー・マネージャーに接続されたサブスクライバーと、ローカル・サブスクライバーになります。 これらのパブリケーションはグローバル・パブリケーション と呼ばれます。

ASPARENT
: トピック・ツリー内の親トピックの **PUBSCOPE** 設定を使用します。

MQPMO_SCOPE_QMGR メッセージ書き込みオプションを使用して、パブリッシャーはパブリケーションをローカルに行うかグローバルに行うかを指定することもできます。 このオプションを使用する場合には、**PUBSCOPE** トピック属性を使用して設定されているすべての動作がオーバーライドされます。

関連概念

79 ページの『管理トピック・オブジェクト』
管理トピック・オブジェクトを使用すると、デフォルトではない特定の属性をトピックに割り当てることができます。

関連タスク

分散パブリッシュ/サブスクライブ・ネットワークの構成

サブスクリプション有効範囲

サブスクリプションの有効範囲は、1 つのキュー・マネージャーのサブスクリプションが、パブリッシュ/サブスクライブ・クラスターまたは階層の別のキュー・マネージャーでパブリッシュされるパブリケーションを受け取るか、あるいはローカル・パブリッシャーからのみのパブリケーションを受け取るかを制御します。

サブスクリプションの有効範囲をキュー・マネージャーに制限することで、パブリッシュ/サブスクライブ・トポロジー内の別のキュー・マネージャーにプロキシー・サブスクリプションが転送されないようにします。 これにより、キュー・マネージャー間のパブリッシュ/サブスクライブ・メッセージ・トラフィックが削減されます。

パブリッシュ/サブスクライブ・クラスターを使用する場合、サブスクリプションの有効範囲は、主に、トピック・ツリーの特定の場所にあるクラスター・トピック・オブジェクトの定義によって制御されます。 サブスクリプションの有効範囲を設定するときには、プロキシー・サブスクリプションがクラスター内の他のキュー・マネージャーに流れるようにしなければなりません。 クラスター・トピックのサブスクリプションの有効範囲を制限するのは、特定のキュー・マネージャー上の特定のトピックを細かく制御する必要がある場合だけにしてください。

パブリッシュ/サブスクライブ階層を使用する場合、サブスクリプションの有効範囲は、主に、この属性とパブリケーションの有効範囲属性の組み合わせによって制御されます。

SUBSCOPE トピック属性は、特定のトピックに対して行われるサブスクリプションの有効範囲を決定するために使用されます。 この属性には以下のいずれかの値を設定できます。

QMGR
サブスクリプションはローカル・パブリケーションのみを受け取り、プロキシー・サブスクリプションはリモート・キュー・マネージャーに伝搬されません。

ALL
プロキシー・サブスクリプションは、パブリッシュ/サブスクライブ・クラスターまたはパブリッシュ/サブスクライブ階層内のリモート・キュー・マネージャーに伝搬され、サブスクライバーは、ローカルおよびリモートのパブリケーションを受け取ります。

ASPARENT
トピック・ツリー内の親トピックの **SUBSCOPE** 設定を使用します。

トピックのサブスクリプション有効範囲がすべてに設定されている場合、直接または ASPARENT によって解決される場合、サブスクリプションの作成時に MQSO_SCOPE_QMGR を指定することにより、そのトピックの個々のサブスクリプションの有効範囲を QMGR に制限できます。 有効範囲が QMGR であるトピックのサブスクリプションの有効範囲を ALL に広げることはできません。

関連概念
79 ページの『管理トピック・オブジェクト』
管理トピック・オブジェクトを使用すると、デフォルトではない特定の属性をトピックに割り当てることができます。

関連タスク
分散パブリッシュ/サブスクライブ・ネットワークの構成

トピック・スペース

トピック・スペースとは、サブスクライブとパブリッシュを実行できるトピックのセットです。 分散パブリッシュ/サブスクライブ・トポロジー内のキュー・マネージャーにはトピック・スペースがあり、そのトピック・スペースには、そのトポロジー内の接続されたキュー・マネージャーでサブスクライブおよびパブリッシュされたトピックが含まれている可能性があります。

注 : 管理トピック・オブジェクト、トピック・ストリング、トピック・ツリーなどの、キュー・マネージャー内のトピックの概要については、69 ページの『トピック』を参照してください。 特に指定がない限り、この記事におけるトピック という記述は、トピック・ストリング を指すものとします。

トピックは、以下のいずれかの方法によって初期作成されます。

- トピック・オブジェクトまたは永続サブスクリプションを定義するときに、管理者が作成する。
- アプリケーションが新しいトピックに対するパブリケーションまたはサブスクリプションを動的に作成するときに、動的に作成される。

トピックは、プロキシー・サブスクリプションを介して、および管理クラスター・トピック・オブジェクトを作成することによって、他のキュー・マネージャーに伝搬されます。 プロキシー・サブスクリプションの場合、パブリッシャーの接続先であるキュー・マネージャーからサブスクライバーのキュー・マネージャーにパブリケーションが転送されます。

プロキシー・サブスクリプションは、キュー・マネージャー階層で親子関係によって互いに接続されたすべてのキュー・マネージャーの間で伝搬されます。 このため、1 つのキュー・マネージャーにおいて、階層内の他の任意のキュー・マネージャーで定義されたトピックにサブスクライブすることができます。 キュー・マネージャー間に接続パスが存在する限り、それらのキュー・マネージャーの接続方法は問題となりません。

パブリッシュ/サブスクライブ・クラスター内のクラスター・トピックへのサブスクリプションについては、プロキシー・サブスクリプションも伝搬されます。 クラスター・トピックとは、**CLUSTER** 属性を持つ (または親から属性を継承する) トピック・オブジェクトに付加されたトピックです。 クラスター・トピックではないトピックをローカル・トピックといい、これらはクラスターに複製されません。 ローカル・トピックへのサブスクリプションからクラスターにプロキシー・サブスクリプションが伝搬することはありません。

要約すると、2 つの環境でプロキシー・サブスクリプションがサブスクライバー用に作成されます。

1. キュー・マネージャーが階層のメンバーであり、キュー・マネージャーの親と子にプロキシー・サブスクリプションが転送される。

2. キュー・マネージャーがクラスターのメンバーであり、クラスター・トピック・オブジェクトに関連したトピックにサブスクリプション・トピック・ストリングが解決される。 トピックが直接ルーティングのクラスター・トピックである場合、プロキシー・サブスクリプションは、クラスターのすべてのメンバーに転送されます。 トピックがトピック・ホスト・ルーティング のクラスター・トピックである場合、プロキシー・サブスクリプションは、クラスター・トピック・オブジェクトを定義したクラスター内のキュー・マネージャーにのみ転送されます。 詳しくは、95 ページの『パブリッシュ/サブスクライブ・クラスター』を参照してください。

キュー・マネージャーがクラスターと階層のメンバーである場合、プロキシー・サブスクリプションは両方のメカニズムによって伝搬されます。重複するパブリケーションはサブスクライバーに送信されません。

以下のリストでは、3 つのパブリッシュ/サブスクライブ・トポロジーのトピック・スペースについて説明しています。

- 110 ページの『ケース 1. パブリッシュ/サブスクライブ・クラスター』.
- 111 ページの『ケース 2. パブリッシュ/サブスクライブの階層』.

各トピックで、以下の構成タスクはトピック・スペースを結合する方法を説明しています。

- パブリッシュ/サブスクライブ・クラスターでの単一のトピック・スペースの作成。
- 複数のクラスターのトピック・スペースの結合。
- 複数のクラスター内でのトピック・スペースの結合および分離。
- 複数のクラスター内のトピック・スペースに対するパブリッシュおよびサブスクライブ。

ケース 1. パブリッシュ/サブスクライブ・クラスター

この例では、キュー・マネージャーがパブリッシュ/サブスクライブ階層に接続されていないと仮定します。

キュー・マネージャーがパブリッシュ/サブスクライブ・クラスターのメンバーである場合、そのトピック・スペースはローカル・トピックとクラスター・トピックで構成されます。 ローカル・トピックは **CLUSTER** 属性を持たないトピック・オブジェクトに関連付けられています。 あるキュー・マネージャーにローカル・トピック・オブジェクト定義が存在する場合、そのトピック・スペースは、ローカルに定義された独自のトピック・オブジェクトを持つ、クラスター内の別のキュー・マネージャーとは異なります。

パブリッシュ/サブスクライブ・クラスターでは、サブスクライブ先のトピックがクラスター・トピック・オブジェクトに解決される場合を除いて、別のキュー・マネージャー上に定義されたトピックにサブスクライブすることはできません。

トピック・ホスト・ルーティング を使用する場合のように、複数のキュー・マネージャー上に同じ名前のクラスター・トピック・オブジェクトの定義が必要になる場合は、必要な場所ですべての定義が一致することが重要です。 詳しくは、パブリッシュ/サブスクライブ・クラスターでの単一のトピック・スペースの作成を参照してください。

トピック・オブジェクトのローカル定義は、それがクラスター・トピックの定義かローカル・トピックの定義かにかかわらず、クラスター内の別の場所で定義された同じトピック・オブジェクトよりも優先されます。 別の場所で定義されたオブジェクトの方が新しい場合でも、ローカルに定義されたトピックが使用されます。

クラスター内のすべての場所で、同じトピック・ストリングにクラスター・トピック・オブジェクトを関連付けることが重要です。 トピック・オブジェクトが関連付けられたトピック・ストリングを変更することはできません。 同じトピック・オブジェクトを別のトピック・ストリングに関連付けるには、トピック・オブジェクトを削除し、新しいトピック・ストリングを使って再作成する必要があります。 トピックをクラスター化した場合、クラスターの他のメンバーに保管されているトピック・オブジェクトのコピーが削除された後、クラスター内のすべての場所で新しいトピック・オブジェクトのコピーが作成されます。 トピック・オブジェクトのコピーはすべて同じトピック・ストリングを参照します。

クラスター内の異なるキュー・マネージャーで、異なるトピック・ストリングを使って、同じ名前のトピック・オブジェクトの定義を誤って作成してしまうことがあります。 異なるトピック・ストリングを使って同じトピック・オブジェクトを多重定義した場合、トピックが参照される方法と場所に応じて異なる結果が生じる可能性があるので、動作が混乱する場合があります。 この重要なポイントについて詳しくは、同じ名前の複数のクラスター・トピック定義を参照してください。

ケース 2. パブリッシュ/サブスクライブの階層

この例では、キュー・マネージャーがパブリッシュ/サブスクライブ・クラスターのメンバーではないと仮定します。

IBM MQ では、キュー・マネージャーがパブリッシュ/サブスクライブ階層のメンバーである場合、そのトピック・スペースは、ローカルに定義されたすべてのトピックと、接続されているキュー・マネージャーで定義されたすべてのトピックで構成されます。 1 つの階層内のすべてのキュー・マネージャーのトピック・スペースは同じです。 トピックはローカル・トピックとグローバル・トピックに分かれていません。

階層内の別のキュー・マネージャーに接続されたサブスクライバーにトピックのパブリケーションがパブリッシャーから送られることを防ぐには、**PUBSCOPE** オプションおよび **SUBSCOPE** オプションのいずれかを QMGR に設定してください。

キュー・マネージャー QMA 上のトピック・ストリング Alabama を使用して、トピック・オブジェクト USA/Alabama を定義するとします。 結果は次のとおりです。

1. QMA のトピック・スペースには、トピック・オブジェクト Alabama とトピック・ストリング USA/Alabama が含まれるようになりました。
2. アプリケーションまたは管理者は、QMA でトピック・オブジェクト名 Alabama を使用してサブスクリプションを作成できます。
3. アプリケーションは、階層内の任意のキュー・マネージャーに、USA/Alabama を含む任意のトピックへのサブスクリプションを作成できます。 QMA がローカルに定義されていない場合、トピック「USA/Alabama」はトピック・オブジェクト SYSTEM.BASE.TOPIC に分解されます。

関連概念

107 ページの『パブリケーション有効範囲』
パブリッシュ/サブスクライブ・クラスターまたはパブリッシュ/サブスクライブ階層を構成する場合、パブリケーションの有効範囲により、キュー・マネージャーがリモート・キュー・マネージャーにパブリッシュを転送するかどうかをさらに制御することができます。 パブリケーションの有効範囲の管理には、**PUBSCOPE** トピック属性を使用します。

108 ページの『サブスクリプション有効範囲』
サブスクリプションの有効範囲は、1 つのキュー・マネージャーのサブスクリプションが、パブリッシュ/サブスクライブ・クラスターまたは階層の別のキュー・マネージャーでパブリッシュされるパブリケーションを受け取るか、あるいはローカル・パブリッシャーからのみのパブリケーションを受け取るかを制御します。

関連タスク

分散パブリッシュ/サブスクライブ・ネットワークの構成

IBM MQ マルチキャスト

IBM MQ Multicast は、待ち時間が短く、ファンアウトが大きい、高信頼性マルチキャスト・メッセージングです。

マルチキャストは、パフォーマンスに悪影響を与えずに非常に多数のサブスクライバーに拡張することが可能な、効率の良いパブリッシュ/サブスクライブ・メッセージング方式です。 IBM MQ は、肯定応答、否定応答、およびシーケンス番号を使用して、大きなファンアウトで待ち時間の短いメッセージングを実現することにより、信頼性の高いマルチキャスト・メッセージングを可能にします。

IBM MQ Multicast のフェア・デリバリーにより、ほぼ同時の配信が可能になり、特定の受信側が有利になることはありません。 IBM MQ Multicast ではメッセージの配信にネットワークを使用するため、データをファンアウトするためのパブリッシュ/サブスクライブ・エンジンは必要ありません。 トピックがグループ・アドレスにマップされたら、パブリッシャーとサブスクライバーはピアツーピア・モードで作動できるため、キュー・マネージャーは必要ありません。 これにより、キュー・マネージャー・サーバーの負荷が軽減され、キュー・マネージャー・サーバーが潜在的な障害点となることはなくなります。

初期マルチキャストの概念

通信情報 (COMMINFO) オブジェクトを使用して、IBM MQ Multicast を既存のシステムやアプリケーションに簡単に統合できます。 2 つの TOPIC オブジェクト・フィールドを使用して、マルチキャスト・トラフィックをサポートしたり無視したりするように既存の TOPIC オブジェクトを短時間で構成できます。

マルチキャストに必要なオブジェクト

以下の情報は、IBM MQ Multicast に必要な 2 つのオブジェクトの概要です。

COMMINFO オブジェクト
COMMINFO オブジェクトには、マルチキャスト伝送に関連付けられた属性が含まれます。 COMMINFO オブジェクト・パラメーターの詳細については、DEFINE COMMINFO を参照してください。

必ず設定しなければならない COMMINFO フィールドは、COMMINFO オブジェクトの名前だけです。 設定後、この名前はトピックに対して COMMINFO オブジェクトを識別するために使用されます。 COMMINFO オブジェクトの **GRPADDR** フィールドを調べて、値が有効なマルチキャスト・グループ・アドレスであることを確認しなければなりません。

TOPIC オブジェクト
トピックとは、パブリッシュ/サブスクライブ・メッセージでパブリッシュされる情報のサブジェクトのことで、トピックを定義するには TOPIC オブジェクトを作成します。 TOPIC オブジェクト・パラメーターの詳細については、DEFINE TOPIC を参照してください。

TOPIC オブジェクト・パラメーター **COMMINFO** および **MCAST** の値を変更すると、既存のトピックをマルチキャストで使用できます。

- **COMMINFO** パラメーターは、マルチキャスト通信情報オブジェクトの名前を指定します。
- **MCAST** パラメーターは、トピック・ツリー内のこの位置でマルチキャストを許容するかどうかを指定します。 デフォルトでは、**MCAST** は ASPARENT に設定されます。これは、トピックのマルチキャスト属性が親から継承されることを意味します。 **MCAST** を ENABLED に設定すると、このノードでマルチキャスト・トラフィックが許可されます。

マルチキャスト・ネットワークとトピック

以下の情報は、様々なタイプのサブスクリプションとトピック定義を持つサブスクリプションがどうなるかに関する概要です。 これらの例はすべて、TOPIC オブジェクトの **COMMINFO** パラメーターが有効な COMMINFO オブジェクトの名前に設定されていることを前提にしています。

マルチキャスト有効に設定されているトピック
トピック・ストリング **MCAST** パラメーターが ENABLED に設定されていると、以下の場合以外は、マルチキャスト可能なクライアントからのサブスクリプションが許可され、マルチキャスト・サブスクリプションが行われます。

- マルチキャスト可能なクライアントからの永続サブスクリプションである。
- マルチキャスト可能なクライアントからの非管理対象サブスクリプションである。
- マルチキャスト不能なクライアントからのサブスクリプションである。

これらの場合は、非マルチキャスト・サブスクリプションが行われ、サブスクリプションが通常のパブリッシュ/サブスクライブにダウングレードされます。

マルチキャスト無効に設定されているトピック
トピック・ストリング **MCAST** パラメーターが DISABLED に設定されていると、常に非マルチキャスト・サブスクリプションが行われ、サブスクリプションが通常のパブリッシュ/サブスクライブにダウングレードされます。

マルチキャスト専用に設定されているトピック
トピック・ストリング **MCAST** パラメーターが ONLY に設定されていると、以下の場合以外は、マルチキャスト可能なクライアントからのサブスクリプションが許可され、マルチキャスト・サブスクリプションが行われます。

- 永続サブスクリプションです。永続サブスクリプションは拒否され、理由コード 2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED が出されます。
- これは非管理サブスクリプションです。管理対象外サブスクリプションは拒否され、理由コード 2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR が出されます。
- これは、非マルチキャスト対応クライアントからのサブスクリプションです。これらのサブスクリプションは拒否され、理由コード 2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY が出されます。
- これは、ローカルにバインドされたアプリケーションからのサブスクリプションです。これらのサブスクリプションは拒否され、理由コードが 2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY が出されます。

Windows Linux AIX MQ Telemetry 概要

MQ Telemetry は、キュー・マネージャーの一部である遠隔測定 (MQXR) サービス、テレメトリー・クライアント (自分で作成することも、無償でダウンロードすることも可能)、コマンド・ラインインターフェース、およびエクスプローラー形式の管理インターフェースで構成されます。 テレメトリーとは、幅広い種類のリモート・デバイスからデータを収集し、それらのデバイスを管理することです。 MQ Telemetry を利用すると、データの収集と、デバイスの制御を Web アプリケーションに統合できます。

MQ Telemetry は IBM MQ のコンポーネントです。 これらのバージョンのアップグレードは、基本的に新しいバージョンの IBM MQ のインストールです。

サンプル・アプリケーションは、引き続き Eclipse Paho および MQTT.org から無料で入手できます。 IBM MQ Telemetry Transport サンプル・プログラムを参照してください。

MQ Telemetry は IBM MQ のコンポーネントであるため、 MQ Telemetry はメイン製品と一緒にインストールすることも、メイン製品のインストール後にインストールすることもできます。 移行情報ついては、「MQ Telemetry 上で移行 Windows」および「MQ Telemetry 上で移行 Linux」を参照してください。

MQ Telemetry には、以下のコンポーネントが含まれています。

遠隔測定チャネル
MQTT クライアントと IBM MQ との接続を管理するには、遠隔測定チャネルを使用します。 テレメトリー・チャネルは、SYSTEM.MQTT.TRANSMIT.QUEUE などの新しい IBM MQ オブジェクトを使用して IBM MQ と対話します。

遠隔測定 (MQXR) サービス
MQTT クライアントは、 SYSTEM.MQXR.SERVICE テレメトリー・サービスを使用してテレメトリー・チャネルに接続します。

MQ Telemetry の IBM MQ Explorer サポート
MQ Telemetry は IBM MQ Explorer を使用して管理されます。

資料
MQ Telemetry 資料は、標準の IBM MQ 製品資料に含まれています。 Java および C クライアント用の SDK 資料は、Javadoc および HTML 形式で製品資料の中に用意されています。

遠隔測定の概念

何をすべきかを決める際は、自分を取り巻く環境から情報を収集するものです。 例えば消費者の立場では、どの食料を買うかを決める前に、蓄えとして何があるかを確認します。 乗り継ぎを予約する前に、今出発すると移動にどれだけ時間がかかるかを知りたいでしょう。 医者に診てもらうかどうかを決める前に、自分の症状を確認します。 バスを待つかどうかを決める前に、バスの到着予定時刻を確認します。 こうした意思決定のための情報は、メーターと装置から、または紙に書かれた言葉または画面から、あるいは自分自身から、直接もたらされます。 どこにいようと、いつやらなければならないとしても、情報を収集してまとめ、分析し、それに基づいて行動します。

情報源が広く分散していたりアクセスできなかったりする場合は、最も正確な情報を収集することは難しくなり、コストもかかるようになります。 加える変更が多い場合や変更することが難しい場合は、変更がなされないか、あまり効果的でないときに変更が行われます。

広く分散した装置からの情報収集とそれらの装置の制御のコストが、デジタル・テクノロジーを搭載した装置をインターネットに接続することによって大幅に削減されるとしたら、どうなるでしょう。 インター

ネットと企業のリソースを使って情報を分析することができます。 情報で裏付けられた判断をし、それに基づいて行動する機会が増えます。

テクノロジー・トレンドと環境面、経済面のプレッシャーが、以下のような変化をもたらしています。

1. 標準化と低価格デジタル・プロセッサーへの接続により、センサーとアクチュエーターの接続と制御のコストが削減されています。
2. 装置を接続するのにインターネットとインターネット・テクノロジーがますます使われています。 国によっては、インターネット・アプリケーションへの接続数の点で、携帯電話がパーソナル・コンピューターを上回っています。 他の装置も確実に後を追っています。
3. インターネットとインターネット・テクノロジーは、アプリケーションによるデータの取得を格段に容易にします。 データに簡単にアクセスできるので、センサーからのデータをさらに多くのソリューションで役立つ情報に変えるためのデータ分析論が使われるようになっています。
4. リソースを賢く使うことは、多くの場合に、炭酸ガス放出とコストを削減するための、より手っ取り早く安上がりな方法です。 別の方法としては、新しいリソースを見つけるか、既存のリソースを利用するための新しいテクノロジーを開発することになりますが、この場合は長期的ソリューションになる可能性があります。 短期間のうちに新しいテクノロジーを開発したり新しいリソースを見つけたりするのは、たいていは既存のソリューションを改良するよりもリスクがあり、時間とコストもかかります。

例

このような傾向が、環境とインテリジェントに連携する新しい機会をいかに生み出すかを、1 つの例が示しています。

「海上における人命の安全のための国際条約」(SOLAS) は、多くの船舶に船舶自動識別装置 (AIS) を配備することを義務付けています。 AIS を義務付けられているのは、300 トンを超える商船と、客船です。 AIS は本来、沿岸運航を対象とした衝突回避システムです。 管海官庁による沿岸水域の監視と管理に使われます。

世界中の熱心な人たちが、低価格 AIS 追跡局を配備して、沿岸運航情報をインターネット上に流しています。 その一方で、AIS からの情報をインターネットからの他の情報と結合するアプリケーションを作成している熱心な人たちもいます。 結果は Web サイトに置かれ、Twitter や SMS を使って公開されます。

あるアプリケーションでは、サウサンプトン近辺の AIS 局からの情報が、船舶所有権および地理情報と結合されます。 このアプリケーションは、フェリーの到着と Twitter への逸脱に関する情報をフィードします。 サウサンプトンとワイト島を結ぶフェリーを使用している定期利用者は、Twitter や SMS を使ってニュース・フィードをサブスクライブしています。 フェリーが遅れている場合は、定期便の到着時間が予定されている時間より遅く、出発を遅らせ、フェリーに乗船することができます。

さらに別の例については、116 ページの『Telemetry のユースケース』を参照してください。

関連タスク

MQ Telemetry のインストール
MQ Telemetry の管理
Windows 上の MQ Telemetry のマイグレーション
Linux 上の MQ Telemetry のマイグレーション
MQ Telemetry 用アプリケーションの開発
MQ Telemetry の問題のトラブルシューティング

関連資料

MQ Telemetry リファレンス

Windows Linux AIX MQ Telemetry の概要

世間一般や企業、行政機関の間で、私たちが生活し、働いている環境との連携を、MQ Telemetry を使ってもっとスマートなものにしたいという願望がますます強くなっています。 MQ Telemetry は、あらゆる種類の装置をインターネットや企業に接続し、スマート・デバイス用アプリケーションの作成コストを削減します。

MQ Telemetry とは何ですか？

- これは IBM MQ の機能で、IBM MQ が提供する汎用のメッセージング・バックボーンを、広範なリモート・センサー、アクチュエーター、および遠隔測定装置に拡張します。 MQ Telemetry による拡張により IBM MQ は、ネットワークや機能の備わったデバイスを使用して、インテリジェントなエンタープライズ・アプリケーション、サービス、および意思決定者と相互接続できるようになります。
- MQ Telemetry のコア部分は、以下のとおりです。

  MQ Telemetry (MQXR) サービス。
  このサービスは、IBM MQ サーバー内部で実行され、IBM MQ Telemetry Transport (MQTT) プロトコルを使用してテレメトリー・デバイスと通信します。

  ユーザーが作成した MQTT アプリケーション。
  これらのアプリケーションは、遠隔測定装置と IBM MQ キュー・マネージャーの間で送受信される情報、およびその情報への応答として行われるアクションを制御します。 このようなアプリケーションを作成する場合は、MQTT クライアント・ライブラリーを使用すると役に立ちます。

利点

- MQTT は、幅広い種類のデバイス用に MQTT 実装環境を作成できるようにするオープン・メッセージング・トランスポートです。
- MQTT クライアントは、リソースが制限されている、フットプリントの小さいデバイスで実行できます。
- MQTT は、処理能力の低いネットワークや、データ送信のコストが高いネットワーク、または不安定なネットワークで効率的に機能します。
- メッセージ・デリバリーは確実であり、アプリケーションからは分離されています。
- アプリケーション・プログラマーには通信プログラミングの知識は 必要ありません。
- メッセージを、他のメッセージング・アプリケーションと交換することができます。 他のメッセージング・アプリケーションとは別のテレメトリー・アプリケーション、 MQI、JMS、または企業のメッセージング・アプリケーションにすることができます。

使用方法

- サンプル・スクリプトは、サンプルの IBM MQ Telemetry Transport v3 クライアント・アプリケーション (`mqttv3app.jar`) を処理する場合に提供されます。IBM MQ Telemetry Transport サンプル・プログラムを参照。
- IBM MQ Explorer とその関連ツールを使用して、IBM MQ のテレメトリー機能を管理します。
- クライアント・ライブラリーを使用すると、キュー・マネージャーに接続してパブリッシュ/サブスクライブ・メッセージングを使用する MQTT アプリケーションの作成に役立ちます。
- アプリケーションを実行するデバイスに、アプリケーションとクライアント・ライブラリーを配布します。

動作方法

- MQTT は、パブリッシュ/サブスクライブ・プロトコルです。 MQTT クライアント・アプリケーションは、MQTT サーバーにメッセージをパブリッシュしたり、MQTT サーバーに接続しているアプリケーションから送信されたメッセージをサブスクライブしたりできます。
- MQTT クライアント・アプリケーションは、MQTT メッセージ・トランスポートを実装するクライアント・ライブラリーを使用します。
- 基本的な MQTT クライアント・アプリケーションは、標準的な MQ クライアントと同じように動作しますが、より多様なプラットフォームおよびネットワークで実行できます。
- MQ Telemetry (MQXR) サービスが、IBM MQ キュー・マネージャーを MQTT サーバーに変えます。
- IBM MQ キュー・マネージャーが MQTT サーバーとして機能する場合、キュー・マネージャーに接続する他のアプリケーションは、MQTT クライアントからのメッセージのサブスクライブおよび受信を行うことができます。

- キュー・マネージャーは、パブリッシュするアプリケーションからサブスクライブするアプリケーションへメッセージを配布するルーターとして動作します。
- メッセージは、異なるタイプのクライアント・アプリケーション間で配布できます。 例えば、Telemetry クライアントと JMS クライアント間などです。

注 : MQ Telemetry は、WebSphere Message Broker のバージョン 7 で廃止された SCADA ノード (現在は IBM Integration Bus と呼ばれている) を置き換えます。そして、Windows、Linux、および AIX 上で実行されます。

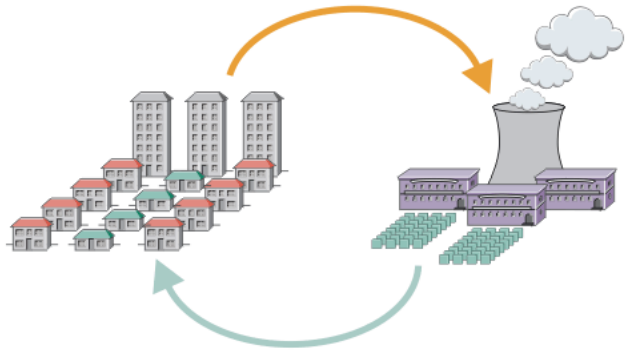
Windows Linux AIX Telemetry のユースケース

テレメトリーは、自動化されたセンシング、データの測定、およびリモート装置の制御を行うものです。特に重要なのは、装置から中央制御点へのデータ伝送です。 遠隔測定には、構成情報と制御情報を装置に送信することも含まれます。

MQ Telemetry は、MQTT protocol を使用して小型装置を接続し、IBM MQ を使用してそれらの装置を他のアプリケーションに接続します。 MQ Telemetry は、装置とインターネットの間のギャップを埋めて、「スマート・ソリューション」の作成を容易にします。 スマート・ソリューションは、装置をモニターして制御するアプリケーションのために、インターネットおよびエンタープライズ・アプリケーションで使用可能な情報の宝庫の錠を開けます。

以下の各図は、MQ Telemetry の代表的な使用例を示したものです。

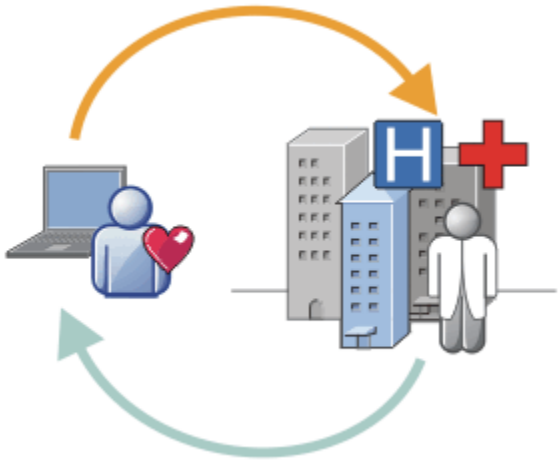
Telemetry: スマート電力

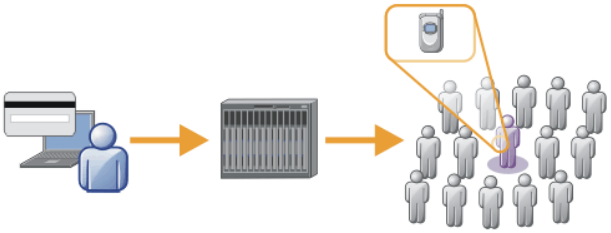


- エネルギー使用量データを含んだ MQTT メッセージがサービス・プロバイダーに送られます。
- MQ Telemetry は、エネルギー使用量データの分析に基づいて制御コマンドを送ります。
- 詳しくは、118 ページの『遠隔測定ユース・ケース: 家庭エネルギーのモニターと制御』のユース・ケースを参照してください。

Telemetry: スマート医療サービス

- MQ Telemetry は、ヘルス・データを病院と医師に送信します。
- 医療データの分析に基づいて、MQTT 警報やフィードバックが送られます。
- 詳しくは、117 ページの『遠隔測定ユース・ケース: 在宅患者モニター』のユース・ケースを参照してください。



Telemetry: 大勢の中の一人	
	• シンプルなカード・トランザクションが銀行のサーバーに送られます。 • MQ Telemetry は、数千名の中から一人の個人を特定して、その顧客のカードが使用されているという警報を返します。 • MQ Telemetry は、最もシンプルな入力情報に基づいてその個人を特定します。

サブトピックで説明されているユース・ケースは、実例に基づいたものです。 これらのユース・ケースは、遠隔測定のいくつかの使用法と、遠隔測定テクノロジーで解決されるはずの一般的な問題のいくつかを示しています。

Windows Linux AIX 遠隔測定ユース・ケース: 在宅患者モニター

心臓病患者ケア・システムに関する IBM と医療機関のコラボレーションでは、埋め込み除細動器が病院とコミュニケーションを取ります。 患者と埋め込みデバイスに関するデータが、RF 遠隔測定を使用して患者の自宅内の MQTT 装置に転送されます。

転送は通常、ベッドのそばに置かれた送信機に対して毎晩行われます。 送信機は電話システムを介してデータを病院に安全に転送し、病院でデータが分析されます。

このシステムにより、患者が医師に診てもらう回数が少なくなります。 患者またはデバイスが要注意であることが検出され、緊急の場合には待機医師に通報されます。

IBM と医療機関のこのコラボレーションは、さまざまな遠隔測定ユース・ケースに共通の以下の特性を持っています。

不可視性

電源と電話回線をつなぐことと一日の一部の時間に装置のすぐそばにいること以外に、装置はユーザー介入を必要としません。 確実に操作でき、使い方も簡単です。

患者が装置をセットアップしなくてもよいように、装置サプライヤーが装置を事前構成します。 患者は装置のプラグを差し込むだけでよいのです。 患者による構成がないので、装置の操作が単純になり、装置が誤って構成される可能性が低くなります。

MQTT クライアントが装置の一部として組み込まれます。 装置デベロッパーは装置に MQTT クライアント実装を組み込み、デベロッパーまたはサプライヤーが事前構成の一部として MQTT クライアントを構成します。

MQTT クライアントは、Java SE JAR ファイルとして出荷されます。この JAR ファイルは、開発者が Java アプリケーションに組み込んでいます。 このシナリオのような非 Java 環境の場合、装置デベロッパーは MQTT の公開フォーマットと公開プロトコルを使用して、異なる言語でクライアントを実装できます。 また、開発者は、Windows、Linux、および ARM プラットフォームの共有ライブラリーとして出荷された C クライアントの 1 つを使用することもできます。

不均等な接続性

除細動器と病院のコミュニケーションは、不均等なネットワーク特性を持ちます。 患者からデータを収集することとそのデータを病院に送信することにまつわるさまざまな問題を解決するために、2 つの異なるネットワークが使用されます。 患者と MQTT 装置の間には、近距離低出力 RF ネットワークが使用されます。 送信機は、低帯域幅電話回線を介する VPN TCP/IP 接続を使用して病院に接続します。

何とか工夫してすべての装置をインターネット・プロトコル・ネットワークに直接接続することは、多くの場合、実際的ではありません。 ハブによって接続された 2 つのネットワークを使用するのが、一般的なソリューションです。 MQTT 装置は単純なハブであり、患者からの情報を保管し、それを病院に転送します。

セキュリティー

医師は患者データの確実性を信用できなければならず、患者は自分のデータのプライバシーが尊重されることを望んでいます。

VPN または TLS を使用して接続を暗号化するだけで十分な状況もあります。 一方、保管されたデータもセキュリティー保護しておくことが望ましい状況もあります。

遠隔測定装置がセキュリティー保護されていないこともあります。 例えば、共用住居の中に置かれる可能性があります。 装置のユーザーは、正しい患者からのデータであることを確認するために認証されなければなりません。 装置自体は TLS を使用してサーバーに対して認証することができ、またそのサーバーを装置に対して認証できます。

装置とキュー・マネージャーの間の遠隔測定チャネルは、ユーザー認証のための JAAS と通信暗号化のための TLS、および装置認証をサポートします。 パブリケーションに対するアクセス権限は、IBM MQ のオブジェクト権限マネージャーによって制御されます。

ユーザーを認証するために使用される ID は、共通患者 ID などの異なる ID にマップできます。 共通 ID を使用すると、IBM MQ でパブリケーション・トピックに対する許可を構成するのが簡単になります。

接続性

MQTT 装置と病院の間の接続にはダイヤルアップが使用され、300 ボーという低帯域幅で動作します。

300 ボーで効果的に作動するように、MQTT protocol では、TCP/IP ヘッダーの他にさらにメッセージに追加されるのは数バイトだけです。

MQTT protocol には、待ち時間が少ない単一伝送の「応答不要送信」メッセージングが用意されています。 応答時間よりも送達が保証されることの方が重要である場合は、「最低 *1* 回」および「正確に *1* 回」の送達を保証する複数伝送も使用できます。 送達を保証するために、メッセージは正常に送達されるまで装置で保管されます。 装置がワイヤレス接続される場合に、保証送達は特に有用です。

拡張容易性

遠隔測定装置は通常、数万から数百万まで、大量に配備されます。

多数の装置をシステムに接続するとなると、大きな要求がソリューションに突きつけられます。 装置とそのソフトウェアのコストなどのビジネス上の要求と、ライセンスや装置、ユーザーの管理という管理上の要求があります。 技術上の要求としては、ネットワークおよびサーバーの負荷があります。

接続を開く際に、オープン接続を維持するよりも多くのサーバー・リソースを使用します。 しかし、このような電話回線を使用するユース・ケースでは、接続の費用は、必要な時間しか接続が開かれていないことが重要になります。 データ転送は、大体はバッチ処理される性質のものです。 就寝時間前の突然の接続ピークを回避するために、接続のスケジュールを夜間に分散させることができます。

クライアントに関しては、必要とするクライアント構成が最低限のものであることが、クライアントの拡張容易性に寄与します。 MQTT クライアントは装置に組み込まれます。 構成ステップや MQTT クライアント・ライセンス承諾ステップを患者への装置の配備に組み込まなければならないという要件はありません。

サーバーに関しては、MQ Telemetry の初期ターゲットは、キュー・マネージャー当たり 50,000 個のオープン接続です。

接続は IBM MQ Explorer を使用して管理されます。 IBM MQ Explorer は、表示する接続をフィルターに掛けて、管理しやすい数に絞り込みます。 ID をクライアントに割り振るためのスキームを適切に選択すれば、地理に基づいて、または患者の名前のアルファベット順に、接続をフィルターに掛けることができます。

Windows Linux AIX 遠隔測定ユース・ケース: 家庭エネルギーのモニターと制御

スマート・メーターは、従来型メーターよりも詳細にエネルギー使用量について収集します。

スマート・メーターはしばしば、家庭内の個々の電気製品をモニターおよび制御するために、ローカル遠隔測定ネットワークと結合されます。 ある距離を置いてモニターと制御を行うために、リモート接続されるものもあります。

リモート接続のセットアップを行う場合は、個人または電力事業者 (つまり中央制御点) が行うことになります。 リモート制御点は電力使用量を読み取り、使用量データを提供できます。 また、継続料金設定や気象情報など、使用量に影響するデータを提供できます。 全体発電効率を上げるために、負荷を制限することもできます。

スマート・メーターは、広く配備され始めています。 例えば英国政府は、2020 年までに英国のすべての家庭にスマート・メーターを配備することについて協議しています。

家庭メーターのユース・ケースは、以下のようないくつかの共通特性を持ちます。

不可視性

ユーザーがメーターを利用してエネルギー節約に踏み込むつもりでない限り、メーターはユーザー介入を必要としません。 メーターが個々の電気製品へのエネルギー供給の信頼性を落としてはなりません。

MQTT クライアントはメーターと一体で配備されたソフトウェアに組み込むことができ、別個のインストールや構成を必要としません。

不均等な接続性

電気製品とスマート・メーターの間の通信は、メーターとリモート接続ポイントの間とは異なる接続標準を必要とします。

スマート・メーターから電気製品への接続は可用性が高くなければならず、Home Area Network のネットワーク標準に準拠していなければなりません。

リモート・ネットワークでは、さまざまな物理接続が使用されるでしょう。 その中には、セルラーのように、伝送コストが高く、途切れることがあるものもあります。 MQTT v3 仕様は、リモート接続、およびローカル・アダプターとスマート・メーターの間の接続を対象としています。

電源コンセントと電気製品の間の接続とメーターでは、Zigbee のような Home Area Network が使用されます。 センサー・ネットワーク用 MQTT (MQTT-S) は、Zigbee などの低帯域幅ネットワーク・プロトコルと連動するように設計されています。 MQ Telemetry は、MQTT-S を直接サポートしていません。 MQTT-S を MQTT v3 に接続するためのゲートウェイが必要です。

在宅患者モニターのように、家庭エネルギーのモニターと制御のソリューションも複数のネットワークを必要とし、これらのネットワークはスマート・メーターをハブとして使用して接続されます。

セキュリティー

スマート・メーターに関連するいくつかのセキュリティー問題があります。 トランザクションの否認防止、開始された制御アクションの許可、電力使用量データのプライバシーが問題として挙げられます。

プライバシーを確保するために、メーターとリモート制御点の間で MQTT により転送されるデータを、TLS を使用して暗号化できます。 制御アクションの許可を確かなものにするために、メーターとリモート制御点の間の MQTT 接続を、TLS を使用して相互に認証できます。

接続性

リモート・ネットワークの物理的性質は、大きく異なる可能性があります。 既存のブロードバンド接続を使用する場合もあれば、通話コストが高く使用可能状態が断続的なモバイル・ネットワークを使用する場合もあります。 高コストで途切れることがある接続にとって、MQTT は効率的で信頼性の高いプロトコルです (117 ページの『遠隔測定ユース・ケース: 在宅患者モニター』を参照)。

拡張容易性

電力会社 (つまり中央制御点) は、最終的に数千万台のスマート・メーターを配備することを計画しています。 当初の配備ごとのメーター台数は、数万台から数十万台の規模です。 この数は、MQTT の初期ターゲットであるキュー・マネージャー当たりのオープン・クライアント接続数 50,000 に匹敵します。

家庭エネルギーのモニターと制御のアーキテクチャーの重大な側面は、スマート・メーターをネットワーク・コンセントレーターとして使用する点です。 電気製品用アダプターは、それぞれが別個のセンサーです。 それらを MQTT を使用するローカル・ハブに接続することにより、ハブは中央制御点との

単一 TCP/IP セッションにデータ・フローを集信することができます。また、セッション障害を乗り越えられるように、しばらくの間メッセージを保管できます。

家庭エネルギーのユース・ケースでは、2 つの理由により、リモート接続を開いたままにしておく必要があります。 第 1 に、接続を開くための時間が、要求を送信することと比較して長くかかるためです。 短い間隔で"「負荷制限」"要求を送信するために接続をいくつも開いていては、時間がかかりすぎてしまいます。 第 2 に、電力会社から負荷制限要求を受信するためには、まずクライアントが接続を開かなければなりません。 MQTT の場合、接続は常にクライアントによって開始されます。電力会社から負荷制限要求を受信するには、接続を開いたままにしておく必要があります。

接続を開く速度が重要である場合や、時間が重要な要求をサーバーが開始する場合、解決策としては通常、オープン接続を多数維持しておきます。

Windows Linux AIX 遠隔測定ユース・ケース: Radio Frequency Identification (RFID)

RFID とは、対象をワイヤレスで識別および追跡するために組み込み RFID タグを使用することです。 RFID タグは、RFID リーダーの照準に入っていれば、最大で数メートルの距離から読み取ることができます。 パッシブ・タグは RFID リーダーによってアクティブ化されます。 アクティブ・タグは、外部からの活動化がなくても送信します。 アクティブ・タグには給電部がなければなりません。 パッシブ・タグにも、距離を広げるために給電部を組み込むことができます。

RFID は多くのアプリケーションで使用されており、ユース・ケースのタイプは実にさまざまです。 RFID のユース・ケース、在宅患者モニターのユース・ケース、家庭エネルギーのモニターおよび制御のユース・ケースには、類似点と相違点がいくつかあります。

不可視性

多くのユース・ケースでは、RFID リーダーは大量に配備されるため、ユーザー介入なしで動作しなければなりません。 リーダーには、中央制御点と通信するための組み込み MQTT クライアントが搭載されます。

例えば、流通倉庫では、リーダーはモーション・センサーを使用してパレットを検出します。 パレットに積まれたアイテムの RFID タグをアクティブ化し、中央アプリケーションにデータと要求を送信します。 データは、在庫の場所を更新するために使用されます。 要求は、特定のベイに移動するなどの、パレットに対する次のアクションを制御します。 航空会社と空港のバゲージ (荷物) システムでは、RFID がこのように使用されています。

一部の RFID ユース・ケースでは、リーダーには Java Platform, Micro Edition (Java ME) などの標準的なコンピューティング環境があります。これらのケースでは、MQTT クライアントは、製造後に別個の構成ステップにデプロイされる場合があります。

不均等な接続性

RFID リーダーが MQTT クライアント搭載のローカル制御装置とは別になっている場合もあれば、各リーダーに MQTT クライアントが組み込まれている場合もあります。 通常は、地理的要因や通信上の要因から、トポロジーを選択することになります。

セキュリティー

RFID タグの添付におけるセキュリティー上の懸念は、プライバシーと認証性です。 RFID タグは人目に付かないため、モニターやスプーフ、改ざんがひそかに行われるおそれがあります。

RFID のセキュリティー問題のソリューションにより、新しい RFID ソリューションを配備する機会が増えています。 RFID タグとローカル・リーダーにセキュリティー上の脆弱性があるものの、中央情報処理を使用することが、さまざまな脅威に対抗するためのアプローチになります。 例えば、タグの改ざんは、デリバリーとディスパッチの在庫基準を動的に相互に関連付けることにより、検出される可能性があります。

接続性

RFID アプリケーションには通常、RFID リーダーと即時照会から集められた情報のバッチ型ストア・アンド・フォワードが含まれます。 流通倉庫ユース・ケースでは、RFID リーダーは常に接続されています。 タグが読み取られると、リーダーに関する情報と共にパブリッシュされます。 倉庫アプリケーションが、リーダーに応答をパブリッシュして戻します。

倉庫アプリケーションでは、ネットワークは一般的に信頼性が高いので、即時要求ではパフォーマンス面で待ち時間の少ない「応答不要送信」メッセージが使用されることがあります。 データのバッチ型ストア・アンド・フォワードでは、データ脱落に伴う管理コストを最小にするために、「正確に *1* 回」メッセージングが使用されることがあります。

拡張容易性

RFID アプリケーションが 1 秒か 2 秒程度の即時応答を必要とする場合、RFID リーダーは接続されたままでなければなりません。

Windows Linux AIX 遠隔測定ユース・ケース: 環境センシング

環境センシングでは、遠隔測定を使用して、河川の水位や質、大気汚染物質などの環境データに関する情報を収集します。

センサーはしばしば、有線通信への到達経路のない遠隔地に置かれます。 無線帯域幅はコストがかかり、信頼性が低い場合があります。 通常は、地理上の小区域内のいくつかの環境センサーが、安全な場所に設置されたローカル・モニター装置に接続されます。 このローカル接続は、有線の場合もあれば、無線の場合もあります。

不可視性

センサー・デバイスは、中央モニター装置よりも、その場所に行きにくく、低電力駆動であり、多数配備される傾向があります。 センサーが「ダム」(無言) の場合もあり、ローカル・モニター装置には、センサー・データを変換して保管するためのアダプターが組み込まれます。 モニター・デバイスには、Java Platform, Standard Edition (Java SE) または Java Platform, Micro Edition (Java ME) をサポートする汎用コンピューターが組み込まれている可能性があります。MQTT クライアントを構成するときに、非可視性が主要な要件になる可能性は低いです。

不均等な接続性

センサーの能力と、リモート接続のコストおよび帯域幅は通常、中央サーバーに接続されたローカル・モニター・ハブに依存することになります。

セキュリティー

このソリューションが軍や防衛のユース・ケースで使用されない限り、セキュリティーは大きな要件ではありません。

接続性

多くの場合、連続してモニターすることも、データを直ちに使用できることも、必須ではありません。洪水水位警報などの例外データは、直ちに転送されなければなりません。 センサー・データは、接続コストと通信コストを削減するためにローカル・モニターで集約され、予定接続を使用して転送されます。 例外データは、モニターで検出されると直ちに転送されます。

拡張容易性

センサーはローカル・ハブの周囲に集められ、センサー・データはパケットに集約されて、スケジュールに従って送信されます。 これらの要素は両方とも、直接接続されたセンサーを使用した場合に中央サーバーにかかる負荷を軽減します。

Windows Linux AIX 遠隔測定ユース・ケース: モバイル・アプリケーション

モバイル・アプリケーションとは、ワイヤレス・デバイス上で動作するアプリケーションのことです。 デバイスは、汎用アプリケーション・プラットフォームかカスタム・デバイスのどちらかです。

汎用プラットフォームとしては、電話や携帯情報端末などのハンドヘルド・デバイスと、ノートブック・コンピューターなどのポータブル・デバイスがあります。 カスタム・デバイスでは、特定のアプリケーションに合わせた特殊目的のハードウェアが使用されます。 「受領署名」小荷物配達を記録するためのデバイスは、カスタム・モバイル・デバイスの 1 例です。 カスタム・モバイル・デバイス上のアプリケーションは、多くの場合、汎用ソフトウェア・プラットフォーム上で作成されます。

不可視性

カスタム・モバイル・アプリケーションのデプロイメントは管理されるので、MQTT クライアント・アプリケーションの構成を含めることができます。 不可視性は、MQTT クライアントを構成するときの大きな要件にはならないでしょう。

不均等な接続性

これまでのユース・ケースのローカル・ハブ・トポロジーとは異なり、モバイル・クライアントはリモートで接続します。 クライアント・アプリケーション層は、中央ハブでアプリケーションに直接接続します。

セキュリティー

物理的セキュリティーがほとんどないため、モバイル・デバイスとモバイル・ユーザーは認証されなければなりません。 デバイスの ID の確認には TLS、ユーザーの認証には JAAS が使用されます。

接続性

モバイル・アプリケーションは、無線可能区域に依存する場合は、オフラインで作動できなければならず、また中断接続に効率的に対処できなければなりません。 こうした環境では、接続されたままであることが目標とはいえ、アプリケーションはメッセージのストア・アンド・フォワードを実行できなければなりません。 メッセージは注文や配達確認であることが多く、重要なビジネス価値を持っています。 メッセージのストア・アンド・フォワードを確実に行う必要があります。

拡張容易性

拡張容易性は大きな課題ではありません。 アプリケーション・クライアントの数は、カスタム・モバイル・アプリケーションのユース・ケースの場合、数千あるいは数万を超えることはないと思われます。

Windows Linux AIX キュー・マネージャーへの遠隔測定装置の接続

遠隔測定装置は、MQTT v3 クライアントを使用してキュー・マネージャーに接続しています。 MQTT v3 クライアントは、遠隔測定 (MQXR) サービスと呼ばれる TCP/IP リスナーに、TCP/IP を使用して接続します。

遠隔測定装置をキュー・マネージャーに接続すると、MQTT クライアントが `MqttClient.connect` メソッドを使用して TCP/IP 接続を開始します。 IBM MQ クライアントのように、MQTT クライアントは、メッセージを送受信するにはキュー・マネージャーに接続されなければなりません。 接続は、遠隔測定 (MQXR) サービスと呼ばれる TCP/IP リスナー (MQ Telemetry と共にインストールされる) を使用してサーバーで行われます。 各キュー・マネージャーは、最大 1 つの遠隔測定 (MQXR) サービスを実行します。

遠隔測定 (MQXR) サービスは、各クライアントによって `MqttClient.connect` メソッドで設定されたリモート・ソケット・アドレスを使用して、接続を遠隔測定チャネルに割り振ります。 ソケット・アドレスは、TCP/IP ホスト名とポート番号の組み合わせです。 同じリモート・ソケット・アドレスを使用する複数のクライアントが、遠隔測定 (MQXR) サービスによって同一遠隔測定チャネルに接続されます。

サーバー上に複数のキュー・マネージャーがある場合は、キュー・マネージャー間で遠隔測定チャネルを分割します。 これらのキュー・マネージャー間で、リモート・ソケット・アドレスを割り振ります。 各遠隔測定チャネルに、固有のリモート・ソケット・アドレスを定義します。 2 つの遠隔測定チャネルが同じソケット・アドレスを使用してはなりません。

複数のキュー・マネージャーの遠隔測定チャネルに同じリモート・ソケット・アドレスを構成した場合は、最初に接続する遠隔測定チャネルが成功します。 それ以降に同じアドレスで接続するチャネルは失敗します。

サーバー上に複数のネットワーク・アダプターがある場合は、遠隔測定チャネル間でリモート・ソケット・アドレスを分割します。 特定のソケット・アドレスが 1 つの遠隔測定チャネルのみに構成されている限り、ソケット・アドレスの割り振りは完全に任意となります。

Configure IBM MQ 用の MQTT サプリメントで提供されたウィザードを使用して、MQ Telemetry クライアントへ接続するために IBM MQ Explorer を構成します。 あるいは、Telemetry 対応キュー・マネージャーの構成 (Linux および AIX) および Windows 上のテレメトリー用キュー・マネージャーの構成の説明に従って、手動で遠隔測定を構成します。

関連資料
MQXR プロパティー

Windows Linux AIX Telemetry 接続プロトコル

MQ Telemetry は、TCP/IP IPv4 と IPv6、および TLS をサポートします。

Windows Linux AIX 遠隔測定 (MQXR) サービス

遠隔測定 (MQXR) サービスは TCP/IP リスナーであり、IBM MQ サービスとして管理されます。 このサービスは、IBM MQ Explorer のウィザード、または **runmqsc** コマンドを使用して作成します。

MQ Telemetry (MQXR) サービスは SYSTEM.MQXR.SERVICE

IBM MQ Explorer の MQ Telemetry 機能に用意されているテレメトリー・サンプル構成ウィザードは、テレメトリー・サービスとサンプル・テレメトリー・チャネルを作成します。 IBM MQ Explorer を使用した MQ Telemetry のインストールの検査 を参照してください。

コマンド・ラインからサンプル構成を作成します (『コマンド・ラインを使用した MQ Telemetry のインストールの検査』を参照)。

遠隔測定 (MQXR) サービスは、キュー・マネージャーと同時に自動的に開始および停止します。 IBM MQ Explorer でサービス・フォルダーを使用して、サービスを制御します。 サービスを表示するには、IBM MQ Explorer が表示から SYSTEM オブジェクトをフィルターで除外するのを停止するためのアイコンをクリックする必要があります。

サービスを手動で作成する方法の例については、以下を参照してください。

- Linux AIX Linux での SYSTEM.MQXR.SERVICE の作成。
- Windows Windows での SYSTEM.MQXR.SERVICE の作成。

V 9.3.0 IBM MQ 9.3.0 以降、 Linux での SYSTEM.MQXR.SERVICE の作成および Windows での SYSTEM.MQXR.SERVICE の作成 が更新され、MQTT TLS チャネルのパスフレーズを暗号化する必要があることを示すデフォルト鍵が指定されるようになりました。 詳しくは、MQTT TLS チャネルのパスフレーズの暗号化を参照してください。

Windows Linux AIX 遠隔測定チャネル

遠隔測定チャネルを作成して、Java 認証・承認サービス (JAAS) または TLS 認証などのさまざまなプロパティーを持つ接続を作成したり、クライアントのグループを管理したりします。

IBM MQ Explorer の MQ Telemetry 関数で提供される **New Telemetry Channel** ウィザードを使用して、テレメトリー・チャネルを作成します。 このウィザードを使用して、特定の TCP/IP ポートで MQTT クライアントからの接続を受け入れるようにチャネルを構成します。 IBM WebSphere MQ 7.1 以降、コマンド行プログラム **runmqsc** を使用して MQ Telemetry を構成できます。

さまざまなポートについて複数の遠隔測定チャネルを作成し、クライアントをグループに分けることによって、多数のクライアント接続を管理しやすくします。 遠隔測定チャネルごとに別々の名前を持ちます。

セキュリティー属性が異なる遠隔測定チャネルを構成して、異なるタイプの接続を作成することができます。 複数のチャネルを作成して、さまざまな TCP/IP アドレスでクライアント接続を受け入れるようにします。 TLS を使用して、メッセージを暗号化し、遠隔測定チャネルとクライアントを認証するようにします (MQTT クライアントおよびテレメトリー・チャネルの TLS 構成を参照)。 ユーザー ID を指定して、IBM MQ オブジェクトへのアクセスの許可を単純化します。 JAAS 構成を指定して、MQTT ユーザーを JAAS で認証するようにします (MQTT クライアントの識別、許可、および認証を参照)。

Windows Linux AIX IBM MQ Telemetry Transport プロトコル

IBM MQ Telemetry Transport (MQTT) v3 プロトコルは、低帯域幅または高コスト接続を使用する小型デバイス間のメッセージ交換用に設計されており、メッセージを確実に送信することを目的としています。 TCP/IP を使用します。

MQTT protocol がパブリッシュされます。IBM MQ Telemetry Transport のフォーマットおよびプロトコルを参照してください。 プロトコルのバージョン 3 は、パブリッシュ/サブスクライブを使用し、「応答不要送信」、「最低 *1* 回」、「正確に *1* 回」の 3 種類のサービス品質をサポートします。

プロトコル・ヘッダーのサイズが小さいことと、メッセージ・ペイロードがバイト配列であることから、小さいメッセージになります。 ヘッダーは、2 バイト固定ヘッダーと、最大 12 バイトの追加可変ヘッダーから成ります。 プロトコルは 12 バイトの可変ヘッダーを使用してサブスクライブと接続を行い、ほとんどのパブリケーションには、わずか 2 バイトの可変ヘッダーを使用します。

3 種類のサービス品質がサポートされているので、待ち時間の少なさと信頼性の間でトレードオフが可能です (MQTT クライアントによって提供されるサービス品質を参照)。 「応答不要送信」は持続的デバイス・ストレージを使用せず、パブリケーションの送受信に伝送を 1 回だけ行います。 「最低 *1* 回」と「正確に *1* 回」は、プロトコル状態を維持して確認応答があるまでメッセージを保存するための持続的ストレージをデバイス上に必要とします。

Windows Linux AIX MQTT クライアント

MQTT クライアント・アプリは、遠隔測定装置からの情報収集、サーバーへの接続、およびサーバーへの情報のパブリッシュを担います。 さらに、トピックにサブスクライブする、パブリケーションを受信する、遠隔測定装置を制御するという処理も行えます。

IBM MQ クライアントアプリケーションとは異なり、MQTT クライアント・アプリケーションは IBM MQ アプリケーションではありません。 MQTT クライアントは接続先のキュー・マネージャーを指定しません。 特定の IBM MQ プログラミング・インターフェースを使用しなければならないという制限を受けません。 代わりに、MQTT クライアントに MQTT 3 プロトコルが実装されます。 MQTT protocol とのインターフェースを提供する独自のクライアント・ライブラリーを、選択したプログラミング言語で、選択したプラットフォーム上に作成することができます。 IBM MQ Telemetry Transport のフォーマットおよびプロトコルを参照してください。

MQTT クライアント・アプリケーションを簡単に作成できるように、さまざまなプラットフォームを対象とする Java をカプセル化した C、JavaScript、および MQTT protocol クライアント・ライブラリーを使用します。 これらのライブラリーを MQTT アプリに取り込むと、完全に機能する MQTT クライアントを、15 行ほどの短いコードで書くことができます。 MQTT クライアント・ライブラリーを Eclipse Paho および MQTT.org から無料で入手することができます。 IBM MQ Telemetry Transport サンプル・プログラムを参照してください。

MQTT クライアント・アプリは、遠隔測定チャネルとの接続の開始を常に担っています。 接続されると、MQTT クライアント・アプリか IBM MQ アプリケーションのどちらかがメッセージの交換を開始できます。

MQTT クライアント・アプリと IBM MQ アプリケーションは、同じトピック・セットにパブリッシュとサブスクライブを行います。 IBM MQ アプリケーションは、MQTT クライアント・アプリが最初にサブスクリプションを作成しなくても、クライアント・アプリに直接メッセージを送信することもできます。 メッセージを MQTT クライアントに送信するように分散キューイングを構成を参照してください。

MQTT クライアント・アプリは、遠隔測定チャネルを使用して IBM MQ に接続されます。 テレメトリー・チャネルは、MQTT および IBM MQ によって使用されるさまざまなタイプのメッセージとの間のブリッジとして機能します。 MQTT クライアント・アプリのためにキュー・マネージャーにパブリケーションとサブスクリプションを作成します。 遠隔測定チャネルは、キュー・マネージャーから MQTT クライアント・アプリへ、MQTT クライアント・アプリのサブスクリプションと一致するパブリケーションを送信します。

Windows Linux AIX MQTT クライアントへのメッセージの送信

IBM MQ アプリケーションは、クライアントによって作成されたサブスクリプションにパブリッシュするか、あるいはメッセージを直接送信して、MQTT v3 クライアントにメッセージを送信できます。 MQTT クライアントは、他のクライアントからサブスクライブされたトピックにパブリッシュすることで、互いにメッセージを送信できます。

MQTT クライアントがパブリケーションにサブスクライブする (このパブリケーションは IBM MQ から受信)

タスク 127 ページの『MQTT クライアント・ユーティリティーへのメッセージを IBM MQ Explorer からパブリッシュする』 を実行して、 IBM MQ から MQTT クライアントにパブリケーションを送信します。

MQTT v3 クライアントがメッセージを受信する標準的な方法は、次のように、クライアントがトピックまたはトピックのセットへのサブスクリプションを作成することです。 125 ページの図 44 のコード・スニペットの例で、MQTT クライアントは、トピック・ストリング "MQTT Examples" を使用してサブスクライブを行います。 IBM MQ C アプリケーション 126 ページの図 45 は、トピック・ストリング "MQTT Examples"を使用してトピックにパブリッシュします。 126 ページの図 46 のコード・スニペットでは、MQTT クライアントは、コールバック・メソッド messageArrived でパブリケーションを受け取ります。

IBM MQ クライアントからのサブスクリプションへの応答としてパブリケーションを送信するように WebSphere MQ を構成する方法 MQTT について詳しくは、MQTT クライアント・サブスクリプションへの応答としてのメッセージのパブリッシュを参照してください。

IBM MQ アプリケーションが MQTT クライアントにメッセージを直接送信する

IBM MQ から MQTT クライアントにメッセージを直接送信するには、131 ページの『MQTT を使用した IBM MQ Explorer クライアントへのメッセージの送信』 のタスクを実行します。

この方法で MQTT クライアントに送信されるメッセージは、非送信請求メッセージと呼ばれます。 MQTT v3 クライアントは、トピック名のセットを持つパブリケーションとして非送信請求メッセージを受け取ります。 遠隔測定 (MQXR) サービスは、トピック名をリモートのキュー名に設定します。

IBM MQ クライアントにメッセージを直接送信するように MQTT を構成する方法について詳しくは、クライアントへのメッセージの直接送信を参照してください。

MQTT クライアントがメッセージをパブリッシュする

MQTT v3 クライアントは、別の MQTT v3 クライアントが受信するメッセージをパブリッシュすることはできますが、非送信請求メッセージを送信することはできません。 コード・スニペットの 126 ページの図 47 は、 Java で作成された MQTT v3 クライアントがメッセージをパブリッシュする方法を示しています。

ある特定の MQTT v3 クライアントにメッセージを送信する場合、各クライアントが独自の ClientIdentifier へのサブスクリプションを作成するのが標準的なパターンです。 トピック・ストリングとして ClientIdentifier を使用して、ある MQTT クライアントから別の MQTT クライアントにメッセージをパブリッシュするには、タスク 133 ページの『特定の MQTT v3 クライアントへのメッセージのパブリッシュ』 を実行します。

コード・スニペットの例

125 ページの図 44 のコード・スニペットは、 Java で作成された MQTT クライアントがサブスクリプションを作成する方法を示しています。 これには、そのサブスクリプションに対するパブリケーションを受信するためのコールバック・メソッド、messageArrived も必要です。

```
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);
```

図 *44. MQTT v3* クライアント・サブスクライバー

126 ページの図 45 のコード・スニペットは、C で書かれた IBM MQ アプリケーションがパブリケーションを送信する方法を示しています。 このコード・スニペットは、可変トピックへのパブリッシャーの作成のタスクから抜き出したものです。

```
/* 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);
```

図 *45. IBM MQ* パブリッシャー

パブリケーションが到着すると、 MQTT クライアントは MQTT アプリケーション・クライアント `MqttCallback` ・クラスの `messageArrived` メソッドを呼び出します。

```
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
}
```

図 *46. messageArrived* メソッド

126 ページの図 47 は、125 ページの図 44 で作成されたサブスクリプションにメッセージをパブリッシュする MQTT v3 を示しています。

```
      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);
```

図 *47. MQTT v3* クライアント・パブリッシャー

Windows Linux AIX MQTT クライアント・ユーティリティーへのメッセージを IBM MQ Explorer からパブリッシュする

この作業のステップに従って、IBM MQ Explorer を使用してメッセージをパブリッシュし、MQTT クライアント・ユーティリティーでそのメッセージにサブスクライブします。 追加作業では、デフォルトの伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定するのではなく、キュー・マネージャー別名を構成する方法を示します。

始める前に

この作業は、ユーザーが IBM MQ および IBM MQ Explorer に精通しており、IBM MQ および MQ Telemetry 機能がインストールされていることを前提としています。

この作業用にキュー・マネージャー・リソースを作成するユーザーには、そのための十分な権限が必要です。 デモンストレーションを目的として、IBM MQ Explorer ユーザー ID は、mqm グループのメンバーであると想定されます。

このタスクについて

この作業では、IBM MQ 内にトピックを作成し、MQTT クライアント・ユーティリティーを使用してそのトピックにサブスクライブします。 IBM MQ Explorer を使用してトピックへのパブリッシュを行うと、MQTT クライアントがパブリケーションを受信します。

手順

以下の作業のいずれかを実行します。

- MQ Telemetry は既にインストールされているが、まだ始動していない場合。 127 ページの『作業の開始時に遠隔測定 (MQXR) サービスが定義されていない場合』の作業を実行します。
- 以前は IBM MQ Telemetry を実行していたが、新しいキュー・マネージャーを使用してデモンストレーションを行う場合。 127 ページの『作業の開始時に遠隔測定 (MQXR) サービスが定義されていない場合』の作業を実行します。
- テレメトリー・リソースが定義されていない既存のキュー・マネージャーを使用して、この作業を実行する場合。 「**サンプル構成の定義**」ウィザードを実行しない場合。
  a. 以下の作業のいずれかを実行して、テレメトリーをセットアップします。
     - Telemetry 対応キュー・マネージャーの構成 (Linux および AIX)
     - Telemetry 対応キュー・マネージャーの構成 (Windows)
  b. 128 ページの『作業の開始時に遠隔測定 (MQXR) サービスが実行されている場合』の作業を実行します。
- テレメトリー・リソースが既に定義されている既存のキュー・マネージャーを使用してこの作業を行う場合には、128 ページの『作業の開始時に遠隔測定 (MQXR) サービスが実行されている場合』の作業を実行します。

次のタスク

131 ページの『MQTT を使用した IBM MQ Explorer クライアントへのメッセージの送信』を実行して、クライアント・ユーティリティーにメッセージを直接送信します。

作業の開始時に遠隔測定 *(MQXR)* サービスが定義されていない場合

キュー・マネージャーを作成し、「**サンプル構成の定義**」を実行して、キュー・マネージャー用にサンプルのテレメトリー・リソースを定義します。 IBM MQ Explorer を使用してメッセージをパブリッシュし、MQTT クライアント・ユーティリティーでそのメッセージにサブスクライブします。

このタスクについて

「**サンプル構成の定義**」を使用してサンプルのテレメトリー・リソースをセットアップすると、このウィザードによってゲスト・ユーザー ID のアクセス権が設定されます。 この方法でゲスト・ユーザー ID に権限

を与えることについては、注意深く検討してください。 Windows 上の guest 、および Linux 上の nobody には、トピック・ツリーのルートへのパブリッシュとサブスクライブ、および SYSTEM.MQTT.TRANSMIT.QUEUE へのメッセージの書き込みを行う権限が付与されています。

またこのウィザードは、デフォルトの伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定します。これを行うと、既存のキュー・マネージャー上で実行されているアプリケーションに干渉する可能性があります。 難しい作業になりますが、デフォルトの伝送キューを使用せずにテレメトリーを構成することも可能です。これを行う場合は、130 ページの『キュー・マネージャー別名の使用』の作業を実行してください。 この作業では、既存のデフォルト伝送キューに干渉する可能性を回避するように、キュー・マネージャーを作成します。

手順

1. IBM MQ Explorer を使用して、新しいキュー・マネージャーを作成および開始します。
   a) Queue Managers「フォルダー」 >「**新規**」 >「**キュー・マネージャー ...**」を右クリックします。 キュー・マネージャーの名前を入力し、>「**完了**」を選択します。

      キュー・マネージャー名 (例えば、MQTTQMGR) を作成します。
2. 遠隔測定 (MQXR) サービスを作成および開始し、サンプルの遠隔測定チャネルを作成します。
   a) Queue Managers*QmgrName*\Telemetry フォルダーを開きます。
   b)「**サンプル構成の定義...**」>「**完了**」をクリックします。

      「**MQTT クライアント・ユーティリティーを起動**」チェック・ボックスは、チェック・マークを付けたままにしておきます。
3. MQTT クライアント・ユーティリティーを使用して、 MQTT Example のサブスクリプションを作成します。
   a)「**接続**」 をクリックします。

      「**クライアント・ヒストリー**」 には、 Connected イベントが記録されます。
   b)「**サブスクリプション \ トピック**」 フィールド>「**サブスクライブ**」に MQTT Example と入力します。

   「**クライアント・ヒストリー**」 には、 Subscribed イベントが記録されます。
4. MQTTExampleTopic 内で IBM MQ を作成します。
   a) **MQ エクスプローラー** >「**新規**」 >「**トピック**」内の Queue Managers*QmgrName*\Topics フォルダーを右クリックします。
   b)「**名前**」に MQTTExampleTopic と入力し、>「**次へ**」を選択します。
   c)「**トピック・ストリング**」 >「**終了**」として MQTT Example と入力します。
   d)「**OK**」をクリックして、確認応答ウィンドウを閉じます。
5. IBM MQ Explorer を使用して、 Hello World! をトピック MQTT Example にパブリッシュします。
   a) Queue Managers*QmgrName*\Topics 内の IBM MQ Explorer フォルダーをクリックします。
   b)「右クリック」MQTTExampleTopic >「**パブリケーションのテスト ...**」
   c)「**メッセージ・データ**」 フィールド>「**メッセージのパブリッシュ**」 >「MQTT クライアント・ユーティリティーへの切り替え」ウィンドウに Hello World! と入力します。

   「**クライアント・ヒストリー**」 には、 Received イベントが記録されます。

作業の開始時に遠隔測定 *(MQXR)* サービスが実行されている場合

テレメトリー・チャネルおよびトピックを作成します。 トピックおよびテレメトリー伝送キューを使用する権限をユーザーに与えます。 IBM MQ Explorer を使用してメッセージをパブリッシュし、MQTT クライアント・ユーティリティーでそのメッセージにサブスクライブします。

始める前に

この作業バージョンでは、キュー・マネージャー *QmgrName* が定義および実行されています。 遠隔測定 (MQXR) サービスは定義および実行されています。 遠隔測定 (MQXR) サービスは、手動で作成されたか、あるいは「**サンプル構成の定義**」ウィザードを実行して作成された可能性があります。

このタスクについて

この作業では、既存のキュー・マネージャーを構成して、MQTT クライアント・ユーティリティーにパブリケーションを送信します。

この作業のステップ 129 ページの『1』では、デフォルトの伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定します。これを行うと、既存のキュー・マネージャー上で実行されているアプリケーションに干渉する可能性があります。 難しい作業になりますが、デフォルトの伝送キューを使用せずにテレメトリーを構成することも可能です。これを行う場合は、130 ページの『キュー・マネージャー別名の使用』の作業を実行してください。

手順

1. SYSTEM.MQTT.TRANSMIT.QUEUE をデフォルトの伝送キューとして設定します。
   a) Queue Managers*QmgrName* folder > 「**プロパティー...**」 を右クリック
   b) ナビゲーターで「**通信**」をクリックします。
   c)「**選択...**」をクリックし、> SYSTEM.MQTT.TRANSMIT.QUEUE を選択して、>「**OK**」>「**OK**」を選択します。
2. テレメトリー・チャネル MQTTExampleChannel を作成して、 MQTT クライアント・ユーティリティーを IBM MQ に接続し、 MQTT クライアント・ユーティリティーを開始します。
   a)「「**MQ エクスプローラー**」 >「**新規**」 >「**テレメトリー・チャネル...**」の中の Queue Managers*QmgrName* \Telemetry\Channels フォルダーを右クリックします。
   b)「**チャネル名**」フィールドに MQTTExampleChannel と入力し、>「**次へ**」>「**次へ**」を選択します。
   c) クライアント認証パネル上の「**固定ユーザー ID** 」を、「MQTTExample」 >「 **次へ**」にパブリッシュおよびサブスクライブする予定のユーザー ID に変更します。
   d)「**クライアント・ユーティリティーを起動する**」にチェック・マークを付けたままにして、>「**完了**」を選択します。
3. MQTT クライアント・ユーティリティーを使用して、 MQTT Example のサブスクリプションを作成します。
   a)「**接続**」 をクリックします。

   「**クライアント・ヒストリー**」 には、 Connected イベントが記録されます。
   b)「**サブスクリプション \ トピック**」 フィールド>「**サブスクライブ**」に MQTT Example と入力します。

   「**クライアント・ヒストリー**」 には、 Subscribed イベントが記録されます。
4. MQTTExampleTopic 内で IBM MQ を作成します。
   a) **MQ エクスプローラー** >「**新規**」 >「**トピック**」内の Queue Managers*QmgrName*\Topics フォルダーを右クリックします。
   b)「**名前**」に MQTTExampleTopic と入力し、>「**次へ**」を選択します。
   c)「**トピック・ストリング**」 >「**終了**」として MQTT Example と入力します。
   d)「**OK**」をクリックして、確認応答ウィンドウを閉じます。
5. mqm グループ内ではなく、ユーザーが MQTTExample トピックをパブリッシュおよびサブスクライブする場合は、以下のようにします。
   a) 以下のようにして、ユーザーにトピック MQTTExampleTopic のパブリッシュおよびサブスクライブを許可します。

```
setmqaut -m qMgrName -t topic -n MQTTExampleTopic -p User ID -all +pub +sub
```

b) メッセージを SYSTEM.MQTT.TRANSMIT.QUEUE に入れる権限をユーザーに与えます。

```
setmqaut -m qMgrName -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p User ID -all +put
```

6. IBM MQ Explorer を使用して、 Hello World! をトピック MQTT Example にパブリッシュします。
   a) Queue Managers*QmgrName*\Topics 内の IBM MQ Explorer フォルダーをクリックします。
   b)「右クリック」MQTTExampleTopic >「**パブリケーションのテスト ...**」
   c)「**メッセージ・データ**」 フィールド>「**メッセージのパブリッシュ**」 >「 MQTT クライアント・ユーティリティーへの切り替え」ウィンドウに Hello World! と入力します。

「**クライアント・ヒストリー**」 には、 Received イベントが記録されます。

キュー・マネージャー別名の使用

デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定せずに、 IBM MQ Explorer を使用して MQTT クライアント・ユーティリティーにメッセージをパブリッシュします。

この作業は前の作業の続きであり、キュー・マネージャー別名を使用することで、デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定することを回避します。

始める前に

127 ページの『作業の開始時に遠隔測定 (MQXR) サービスが定義されていない場合』または 128 ページの『作業の開始時に遠隔測定 (MQXR) サービスが実行されている場合』のいずれかの作業を実行します。

このタスクについて

MQTT クライアントがサブスクリプションを作成すると、IBM MQ は ClientIdentifier をリモート・キュー・マネージャー名として使用して、応答を送信します。 このタスクでは、クライアント ID、MyClient を使用します。

MyClient という名前の伝送キューまたはキュー・マネージャー別名がない場合、応答はデフォルト伝送キューに入れられます。 デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定すると、 MQTT クライアントは応答を受け取ります。

キュー・マネージャー別名を使用することで、デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定することを回避できます。 キュー・マネージャー別名は、ClientIdentifier ごとにセットアップする必要があります。 通常は、多数のクライアントが存在しており、キュー・マネージャー別名を使用するのは現実的ではありません。 ClientIdentifier が予測不能な場合も多く、この方法でテレメトリーを構成することはできません。

それでもなお、デフォルトの伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE 以外に構成しなければならない状況は存在します。 手順 のステップでは、デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に設定する代わりに、キュー・マネージャーの別名を構成します。

手順

1. デフォルトの伝送キューとしての SYSTEM.MQTT.TRANSMIT.QUEUE を削除します。
   a) Queue Managers*QmgrName* folder >「**プロパティー...**」 を右クリック
   b) ナビゲーターで「**通信**」をクリックします。
   c)「**デフォルト伝送キュー**」フィールドから SYSTEM.MQTT.TRANSMIT.QUEUE を削除し、>「**OK**」を選択します。
2. MQTT クライアント・ユーティリティーを使用して、サブスクリプションを作成できなくなったことを確認します。

a)「**接続**」 をクリックします。

「**クライアント・ヒストリー**」 には、 Connected イベントが記録されます。

b)「**サブスクリプション \ トピック**」 フィールド>「**サブスクライブ**」に MQTT Example と入力します。

「**クライアント・ヒストリー**」 には、 Subscribe failed イベントと Connection lost イベントが記録されます。

3. ClientIdentifier のキュー・マネージャー別名、MyClient を作成します。
   a)「Queue Managers*QmgrName*\Queues フォルダー」 >「**新規**」 >「**リモート・キュー定義**」を右クリックします。
   b) 定義に MyClient という名前を付け、>「**次へ**」を選択します。
   c)「**リモート・キュー・マネージャー**」フィールドに MyClient と入力します。
   d)「**伝送キュー**」フィールドに SYSTEM.MQTT.TRANSMIT.QUEUE と入力し、>「**完了**」を選択します。
4. MQTT クライアント・ユーティリティーに再度接続します。
   a)「**クライアント ID**」が MyClient に設定されていることを確認します。
   b) **接続**

「**クライアント・ヒストリー**」 には、 Connected イベントが記録されます。

5. MQTT クライアント・ユーティリティーを使用して、 MQTT Example のサブスクリプションを作成します。
   a)「**接続**」 をクリックします。

「**クライアント・ヒストリー**」 には、 Connected イベントが記録されます。

b)「**サブスクリプション \ トピック**」 フィールド>「**サブスクライブ**」に MQTT Example と入力します。

「**クライアント・ヒストリー**」 には、 Subscribed イベントが記録されます。

6. IBM MQ Explorer を使用して、 Hello World! をトピック MQTT Example にパブリッシュします。
   a) Queue Managers*QmgrName*\Topics 内の IBM MQ Explorer フォルダーをクリックします。
   b)「右クリック」MQTTExampleTopic >「**パブリケーションのテスト ...**」
   c)「**メッセージ・データ**」 フィールド>「**メッセージのパブリッシュ**」 >「 MQTT クライアント・ユーティリティーへの切り替え」ウィンドウに Hello World! と入力します。

「**クライアント・ヒストリー**」 には、 Received イベントが記録されます。

Windows Linux AIX MQTT を使用した IBM MQ Explorer クライアントへのメッセージの送信

MQTT を使用して IBM MQ キューにメッセージを入れることで、IBM MQ Explorer クライアント・ユーティリティーにメッセージを送信します。 この作業では、MQTT クライアントにメッセージを直接送信するように、リモート・キュー定義を構成する方法を示します。

始める前に

127 ページの『MQTT クライアント・ユーティリティーへのメッセージを IBM MQ Explorer からパブリッシュする』のタスクを実行します。 MQTT クライアント・ユーティリティーは、接続したままにします。

このタスクについて

この作業では、トピックへのパブリッシュを使用するのではなく、キューを使用して MQTT クライアントへメッセージを送信する方法について説明します。 クライアントでのサブスクリプションの作成は行いません。 この作業のステップ 132 ページの『2』では、以前のサブスクリプションが削除されていることを確認します。

手順

1. MQTT クライアント・ユーティリティーへの接続を切断してから再接続することで、既存のサブスクリプションをすべて廃棄します。

   デフォルトが変更されていなければ、MQTT クライアント・ユーティリティーは新規セッションで接続するため、サブスクリプションは廃棄されます。クリーン・セッションを参照してください。

   タスクを簡単に実行できるようにするには、 MQTT クライアント・ユーティリティーによって作成された生成済み ClientIdentifier を使用するのではなく、独自の ClientIdentifier を入力します。

   a)「**切断**」をクリックして、テレメトリー・チャネルから MQTT クライアント・ユーティリティーを切断します。

   「**クライアント履歴**」 には、 Disconnected イベントが記録されます。

   b)「**クライアント ID**」を MyClient に変更します。

   c)「**接続**」 をクリックします。

   「**クライアント履歴**」 には、 Connected イベントが記録されます。

2. MQTT クライアント・ユーティリティーが MQTTExampleTopic 用のパブリケーションを受信しなくなっていることを確認します。

   a) Queue Managers*QmgrName*\Topics 内の IBM MQ Explorer フォルダーをクリックします。

   b)「右クリック」MQTTExampleTopic >「**パブリケーションのテスト ...**」

   c)「**メッセージ・データ**」 フィールド>「**メッセージのパブリッシュ**」 >「 MQTT クライアント・ユーティリティーへの切り替え」ウィンドウに Hello World! と入力します。

   「**クライアント・ヒストリー**」にはイベントは何も記録されません。

3. クライアントのリモート・キュー定義を作成します。

   リモート・キュー定義で、リモート・キュー・マネージャー名として、ClientIdentifier である MyClient を設定します。 リモート・キュー名には任意の名前を使用できます。 リモート・キュー名は、トピック名として MQTT クライアントに渡されます。

   a)「Queue Managers*QmgrName*\Queues フォルダー」 >「**新規**」 >「**リモート・キュー定義**」を右クリックします。

   b) 定義に MyClientRemoteQueue という名前を付け、>「**次へ**」を選択します。

   c)「**リモート・キュー**」フィールドに MQTTExampleQueue と入力します。

   d)「**リモート・キュー・マネージャー**」フィールドに MyClient と入力します。

   e)「**伝送キュー**」フィールドに SYSTEM.MQTT.TRANSMIT.QUEUE と入力し、>「**完了**」を選択します。

4. テスト・メッセージを MyClientRemoteQueue に書き込みます。

   a) **MyClientRemoteQueue** を右クリックし、>「**テスト・メッセージの書き込み...**」を選択します。

   b)「メッセージ・データ」フィールドに Hello queue! と入力し、「**メッセージの書き込み**」>「**閉じる**」を選択します。

   「**クライアント・ヒストリー**」 には、 Received イベントが記録されます。

5. デフォルトの伝送キューとしての SYSTEM.MQTT.TRANSMIT.QUEUE を削除します。

   a) Queue Managers*QmgrName* folder >「**プロパティー...**」 を右クリック

   b) ナビゲーターで「**通信**」をクリックします。

   c)「**デフォルト伝送キュー**」フィールドから SYSTEM.MQTT.TRANSMIT.QUEUE を削除し、>「**OK**」を選択します。

6. ステップ 132 ページの『4』をやり直します。

   MyClientRemoteQueue は、伝送キューの名前を明示的に指定するリモート・キュー定義です。 MyClient にメッセージを送信するためにデフォルト伝送キューを定義する必要はありません。

次のタスク

デフォルト伝送キューが SYSTEM.MQTT.TRANSMIT.QUEUE に設定されなくなったため、 MQTT クライアント・ユーティリティーは、 ClientIdentifier、 MyClient にキュー・マネージャー別名が定義されていない限り、新しいサブスクリプションを作成できません。 デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に復元します。

Windows Linux AIX 特定の MQTT v3 クライアントへのメッセージのパブリッシュ

トピック名として ClientIdentifier を使用し、パブリッシュ/サブスクライブ・ブローカーとして IBM MQ を使用して、ある MQTT v3 クライアントから別のクライアントにメッセージをパブリッシュします。

始める前に

127 ページの『MQTT クライアント・ユーティリティーへのメッセージを IBM MQ Explorer からパブリッシュする』のタスクを実行します。 MQTT クライアント・ユーティリティーは、接続したままにします。

このタスクについて

この作業では、以下の 2 つについて説明します。

1. 1 つの MQTT クライアントでトピックへのサブスクライブを行い、別の MQTT クライアントからのパブリケーションを受け取ります。
2. トピック・ストリングとして ClientIdentifier を使用して、「Point-to-Point」サブスクリプションをセットアップします。

手順

1. MQTT クライアント・ユーティリティーへの接続を切断してから再接続することで、既存のサブスクリプションをすべて廃棄します。

   デフォルトが変更されていなければ、MQTT クライアント・ユーティリティーは新規セッションで接続するため、サブスクリプションは廃棄されます。クリーン・セッションを参照してください。

   タスクを簡単に実行できるようにするには、 MQTT クライアント・ユーティリティーによって作成された生成済み ClientIdentifier を使用するのではなく、独自の ClientIdentifier を入力します。

   a)「**切断**」をクリックして、テレメトリー・チャネルから MQTT クライアント・ユーティリティーを切断します。

   「**クライアント履歴**」 には、 Disconnected イベントが記録されます。

   b)「**クライアント ID**」を MyClient に変更します。

   c)「**接続**」 をクリックします。

   「**クライアント履歴**」 には、 Connected イベントが記録されます。
2. トピック MyClient へのサブスクリプションを作成します。

   MyClient は、このクライアントの ClientIdentifier です。

   a)「**サブスクリプション¥トピック**」フィールドに MyClient と入力し、>「**サブスクライブ**」を選択します。

   「**クライアント・ヒストリー**」 には、 Subscribed イベントが記録されます。
3. 別の MQTT クライアント・ユーティリティーを開始します。

   a) Queue Managers*QmgrName*\Telemetry\channels フォルダーを開きます。

   b)「**PlainText**」チャネルを右クリックし、>「**MQTT クライアント・ユーティリティーを実行...**」を選択します。

   c)「**接続**」 をクリックします。

「**クライアント履歴**」 には、 Connected イベントが記録されます。

4. トピック MyClient に Hello MyClient! をパブリッシュします。
   a) クライアント ID、MyClient で実行されている MQTT クライアント・ユーティリティーから、定期購読トピック MyClient をコピーします。
   b) MyClient クライアント・ユーティリティー・インスタンスのそれぞれの **パブリケーション\トピック** フィールドに MQTT を貼り付けます。
   c)「**パブリケーション¥メッセージ**」フィールドに Hello MyClient! と入力します。
   d) 両方のインスタンスで「**パブリッシュ**」をクリックします。

タスクの結果

クライアント ID、MQTT を使用する MyClient クライアント・ユーティリティーのクライアント・ヒストリーは、2 つの**受信済み**イベントおよび 1 つの**公開済み**イベントを記録します。 もう一方の MQTT クライアント・ユーティリティー・インスタンスには、1 つの「**パブリッシュ済み**」イベントが記録されます。

1 つの「**受信済み**」イベントが表示されている場合は、以下の原因が考えられますので確認してください。

1. キュー・マネージャーのデフォルト伝送キューが SYSTEM.MQTT.TRANSMIT.QUEUE に設定されている。
2. 他の演習を実行する際に、MyClient を参照するキュー・マネージャー別名またはリモート・キュー定義を作成しましたか？構成上の問題が発生した場合は、MyClient を参照するリソース (キュー・マネージャー別名や伝送キューなど) を削除します。 クライアント・ユーティリティーへの接続を切断し、遠隔測定 (MQXR) サービスを停止して再開します。

Windows Linux AIX IBM MQ クライアントから MQTT アプリケーションへのメッセージの送信

IBM MQ アプリケーションは、トピックにサブスクライブすることにより、MQTT v3 クライアントからメッセージを受信できます。 MQTT クライアントは、遠隔測定チャネルを使用して IBM MQ に接続し、同じトピックにパブリッシュすることにより、メッセージを IBM MQ アプリケーションに送信します。

134 ページの『IBM MQ クライアントからの MQTT へのメッセージのパブリッシュ』のタスクを実行して、MQTT クライアントから IBM MQ で定義されているサブスクリプションにパブリケーションを送信する方法を学習してください。

トピックがクラスター化されている場合、またはパブリッシュ/サブスクライブ階層を使用して分散されている場合は、サブスクリプションを、MQTT クライアントが接続されているキュー・マネージャーとは異なるキュー・マネージャー上に置くことができます。

Windows Linux AIX IBM MQ クライアントからの MQTT へのメッセージのパブリッシュ

IBM MQ Explorer を使用して、トピックへのサブスクリプションを作成し、MQTT クライアント・ユーティリティーを使用して、トピックにパブリッシュします。

始める前に

127 ページの『MQTT クライアント・ユーティリティーへのメッセージを IBM MQ Explorer からパブリッシュする』のタスクを実行します。 MQTT クライアント・ユーティリティーは、接続したままにします。

このタスクについて

その作業は MQTT クライアントを使用してメッセージのパブリッシングと IBM MQ Explorer を使用して作成された非管理永続サブスクリプションを使用してパブリケーションの受入を実演します。

手順

1. トピック・ストリング MQTT Example に対する永続サブスクリプションを作成します。

   IBM MQ Explorer を使用して、以下のステップを実行し、キューおよびサブスクリプションを作成します。

   a) Queue Managers*QmgrName*\Queues フォルダーから「IBM MQ Explorer」>「**新規**」>「**ローカル・キュー...**」をクリックします。

   b) キュー名として MQTTExampleQueue と入力し、「**終了**」をクリックします。

   c) Queue Managers*QmgrName*\Subscriptions フォルダーから「IBM MQ Explorer」>「**新規**」>「**サブスクリプション...**」を右クリックします。

   d) キュー名として MQTTExampleSubscription と入力し、「**次へ**」をクリックします。

   e)「**選択...**」>「MQTTExampleTopic」>「**OK**」をクリックします。

   MQTTExampleTopic のステップ 128 ページの『4』 でトピック「127 ページの『MQTT クライアント・ユーティリティーへのメッセージを IBM MQ Explorer からパブリッシュする』」が既に作成されています。

   f) 宛先名として MQTTExampleQueue と入力し、「**終了**」をクリックします。

2. 任意指定のステップとして、mqm 権限を持たずに、異なるユーザーが使用できるようにキューをセットアップします。

   mqm 以外の少ない権限を有するユーザーのために設定をセットアップしている場合、put と get 権限を MQTTExampleQueue に付与すべきです。 トピックと伝送キューへのアクセス権限は 127 ページの『MQTT クライアント・ユーティリティーへのメッセージを IBM MQ Explorer からパブリッシュする』 で構成されています。

   a) 以下のようにして、ユーザーにキュー MQTTExampleQueue の書き込みと取得を認可します。

   ```
   setmqaut -m qMgrName -t queue -n MQTTExampleQueue -p User ID -all +put +get
   ```

3. MQTT クライアント・ユーティリティーを使用して、 Hello IBM MQ! をトピック MQTT Example にパブリッシュします。

   MQTT クライアント・ユーティリティーが接続されたままの状態でない場合は、「**PlainText**」チャネルを右クリックして、「**MQTT クライアント・ユーティリティーを実行...**」>「**接続**」をクリックします。

   a)「**パブリケーション¥トピック**」フィールドに MQTT Example と入力します。

   b)「**パブリケーション \ メッセージ**」 フィールド>「**パブリッシュ**」に Hello IBM MQ! と入力します。

4. Queue Managers*QmgrName*\Queues フォルダーを開き、MQTTExampleQueue を見つけます。

   「**現行キュー項目数**」フィールド値は 1 です。

5. MQTTExampleQueue >「**メッセージの参照 ...**」 を右クリックします。 資料を調べてください。

Windows Linux AIX MQTT パブリッシュ/サブスクライブ・アプリケーション

トピック・ベースのパブリッシュ/サブスクライブを使用して MQTT アプリケーションを作成します。

MQTT クライアントが接続されると、クライアントとサーバーの間でどちらかの方向にパブリケーションがフローされます。 クライアントで情報がパブリッシュされる場合、パブリケーションはクライアントから送信されます。 クライアントによって作成されたサブスクリプションと一致するトピックにメッセージがパブリッシュされた場合、パブリケーションはクライアントで受信されます。

IBM MQ パブリッシュ/サブスクライブ・ブローカーは、MQTT クライアントによって作成されたトピックとサブスクリプションを管理します。 MQTT クライアントによって作成されたトピックは、IBM MQ アプリケーションによって作成されたトピックと同じトピック・スペースを共有します。

MQTT クライアント・サブスクリプションのトピック・ストリングと一致するパブリケーションは、リモート・キュー・マネージャー名がクライアントの ClientIdentifier に設定された SYSTEM.MQTT.TRANSMIT.QUEUE に置かれます。 遠隔測定 (MQXR) サービスは、サブスクリプションを作成したクライアントにパブリケーションを転送します。 その場合、クライアントを識別するリモート・キュー・マネージャー名として設定されている ClientIdentifier が使用されます。

通常は、SYSTEM.MQTT.TRANSMIT.QUEUE をデフォルトの伝送キューとして定義する必要があります。 デフォルトの伝送キューを使用しないように MQTT を構成することはできますが、煩わしい作業です (メッセージを MQTT クライアントに送信するように分散キューイングを構成を参照)。

MQTT クライアントは持続セッションを作成できます (139 ページの『MQTT のステートレス・セッションとステートフル・セッション』を参照)。 持続セッションで作成されたサブスクリプションは永続的です。 持続セッションを持つクライアント宛に到着したパブリケーションは、SYSTEM.MQTT.TRANSMIT.QUEUE に保管され、クライアントが再接続した時点でクライアントに転送されます。

MQTT クライアントは、保存パブリケーションにパブリッシュとサブスクライブを行うこともできます (保存パブリケーションおよび MQTT クライアントを参照)。 保存パブリケーション・トピックのサブスクライバーは、トピックへの最新のパブリケーションを受信します。 サブスクライバーは、サブスクリプションの作成時、または以前のセッションへの再接続時に保存パブリケーションを受信します。

Windows Linux AIX 遠隔測定アプリケーション

IBM MQ または IBM Integration Bus メッセージ・フローを使用してテレメトリー・アプリケーションを作成します。

JMS や MQI などの IBM MQ プログラミング・インターフェースを使用して、IBM MQ の遠隔測定アプリケーションをプログラムします。

遠隔測定 (MQXR) サービスは、MQTT v3 メッセージと IBM MQ メッセージの間の変換を行います。 MQTT クライアントのためにサブスクリプションとパブリケーションを作成し、パブリケーションを MQTT クライアントに転送します。 パブリケーションは MQTT v3 メッセージのペイロードです。 ペイロードは、メッセージ・ヘッダーと jms-bytes フォーマットのバイト配列から成ります。 遠隔測定サーバーは、MQTT v3 メッセージのヘッダーと IBM MQ メッセージのヘッダーをマップします (136 ページの『MQ Telemetry とキュー・マネージャーの統合』を参照)。

Publication、MQInput、および JMSInput ノードを使用して、IBM Integration Bus と MQTT クライアントの間でパブリケーションを送受信します。

メッセージ・フローを使用すると、HTTP を使用して Web サイトとテレメトリーを統合し、IBM MQ および WebSphere アダプターを使用して他のアプリケーションと統合することができます。

Windows Linux AIX MQ Telemetry とキュー・マネージャーの統合

MQTT クライアントは、パブリッシュ/サブスクライブ・アプリケーションとして IBM MQ と統合されます。 新しいトピックを作成するか、既存のトピックを使用して、IBM MQ 内のトピックに対して、パブリッシュとサブスクライブのどちらも行えます。 自身のサブスクリプションのトピックにパブリッシュする IBM MQ クライアント (自身を含む) や他の MQTT アプリケーションの結果として、IBM MQ からパブリケーションを受信します。 パブリケーションの属性を決定するために規則が適用されます。

トピック、パブリケーション、サブスクリプション、メッセージに関連付けられる属性のうち、IBM MQ で定義された属性の多くはサポートされません。 137 ページの『MQTT クライアントから IBM MQ パブリッシュ/サブスクライブ・ブローカーへ』と 138 ページの『IBM MQ クライアントへの MQTT』は、パブリケーションの属性がどのように設定されるかを示しています。 設定は、パブリケーションが IBM MQ パブリッシュ/サブスクライブ・ブローカーに向かうものであるか、そこからのものであるかによって異なります。

IBM MQ 内のパブリッシュ/サブスクライブ・トピックは、管理可能トピック・オブジェクトに関連付けられます。 MQTT クライアントによって作成されるトピック間に違いはありません。 MQTT クライアントがパブリケーション用のトピック・ストリングを作成すると、IBM MQ パブリッシュ/サブスクライブ・ブローカーはそれを管理トピック・オブジェクトに関連付けます。 ブローカーはパブリケーションに含まれるトピック・ストリングを、最も近い管理可能トピック・オブジェクトの親にマップします。 このマッピングは、IBM MQ アプリケーションの場合と同じです。 ユーザーが作成したトピックがない場合、パブリケ

ーション・トピックは SYSTEM.BASE.TOPIC にマップされます。 パブリケーションに適用される属性は、トピック・オブジェクトに由来します。

IBM MQ アプリケーションまたは管理者がサブスクリプションを作成すると、そのサブスクリプションに名前が付けられます。 IBM MQ Explorer を使用するか、 **runmqsc** または PCF コマンドを使用して、サブスクリプションをリストします。 すべての MQTT クライアントのサブスクリプションの名前が示されます。 これらには、次の形式の名前が付けられます。 *ClientIdentifier*:*Topic name*

MQTT クライアントから IBM MQ パブリッシュ/サブスクライブ・ブローカーへ

MQTT クライアントが IBM MQ にパブリケーションを送信しました。 遠隔測定 (MQXR) サービスが、パブリケーションを IBM MQ メッセージに変換します。 IBM MQ メッセージは、以下の 3 つの部分で構成されます。

1. MQMD
2. RFH2
3. メッセージ

MQMD の各プロパティーは、137 ページの表 9 に示すものを除いて、それぞれのデフォルト値に設定されます。

表 *9. MQMD* プロパティーの設定

MQMD フィールド	**タイプ**	**値**
Format	MQCHAR8	MQFMT_RF_HEADER_2
UserIdentifier	MQCHAR12	以下のいずれかに設定されます。 MqttClient.ClientIdentifier MqttConnectOptions.UserName IBM MQ 管理者によって設定された遠隔測定チャネルのユーザー ID。
Priority	MQLONG	MQPRI_PRIORITY_AS_Q_DEF (4 のデフォルトを有する IBM MQ とは異なる JMS のデフォルト)
Persistence	MQLONG	QoS=0→MQPER_NOT_PERSISTENT QoS=1→MQPER_PERSISTENT QoS=2→MQPER_PERSISTENT

RFH2 ヘッダーには、 JMS メッセージのタイプを定義するための <msd> フォルダーは含まれていません。 遠隔測定 (MQXR) サービスによって、IBM MQ メッセージがデフォルトの JMS メッセージとして作成されます。 デフォルトの JMS メッセージ・タイプは jms-bytes メッセージです。 アプリケーションからは、メッセージ・プロパティーとして追加されたヘッダー情報にアクセス可能です (メッセージ・プロパティーを参照)。

RFH2 の値は、137 ページの表 10 に示すように設定されます。 Format プロパティーは RFH2 固定ヘッダーに設定され、他の値は RFH2 フォルダーに設定されます。

表 *10. RFH2* プロパティーの設定

RFH2 プロパティー	**タイプ/フォルダー**	**ヘッダー**
形式	MQCHAR8	MQFMT_NONE

表 *10. RFH2* プロパティーの設定 (続き)		
RFH2 プロパティー	**タイプ/フォルダー**	**ヘッダー**
ClientIdentifier	mqtt/clientId	長さが 1...23 バイトの MqttClient.ClientIdentifier をコピーします。
QoS	mqtt/qos	QoS を着信 MQTT メッセージからコピーします。
メッセージ ID	mqtt/msgid	QoS が 1 または 2 の場合は、着信 MQTT メッセージから 「メッセージ ID」 をコピーします。
MQIsRetained	mqps/Ret	元の MQTT パブリケーションが RETAIN プロパティー・セットが設定されて送信された場合に設定し、メッセージは保存パブリケーションとして受信されます。
MQTopicString	mqps/Top	MQTT メッセージがパブリッシュされたトピック。

MQTT パブリケーションに含まれるペイロードは、IBM MQ メッセージの内容に次のようにマップされます。

表 *11. MQTT* パブリケーションのペイロードと *IBM MQ* メッセージの内容のマッピング		
メッセージの内容	**タイプ**	**IBM MQ メッセージの内容**
BUFFER	MQBYTE *n*	着信 MQTT メッセージに含まれるバイトのコピー。 長さがゼロの場合もあります。

IBM MQ クライアントへの MQTT

クライアントがパブリケーション・トピックにサブスクライブしました。 IBM MQ アプリケーションがトピックにパブリッシュした結果、MQTT パブリッシュ/サブスクライブ・ブローカーによってパブリケーションが IBM MQ サブスクライバーに送信されました。 あるいは、IBM MQ アプリケーションが非送信請求メッセージを直接 MQTT クライアントに送信しました。 138 ページの表 12 は、MQTT クライアントに送信されるメッセージで固定メッセージ・ヘッダーがどのように設定されるかを示しています。 IBM MQ メッセージ・ヘッダーの他のデータ、または他のヘッダーは破棄されます。 IBM MQ メッセージのメッセージ・データは、変更されることなく、MQTT メッセージのメッセージ・ペイロードとして送信されます。 MQTT メッセージは、遠隔測定 (MQXR) サービスによって MQTT クライアントに送信されます。

表 *12. MQTT* クライアントに送信される *IBM MQ* メッセージでの固定メッセージ・ヘッダーの設定方法		
MQTT フィールド	**タイプ**	**値**
DUP	ブール値	QoS = 1 または 2 であり、前の伝送でこのクライアントにメッセージが送信されており、一定の時間が経過してもメッセージの確認応答がない場合に設定されます。

表 12. *MQTT* クライアントに送信される *IBM MQ* メッセージでの固定メッセージ・ヘッダーの設定方法 (続き)

MQTT フィールド	タイプ	値
QoS	int	IBM MQ のパブリッシュ/サブスクライブ・ブローカーからの発信パブリケーションにおける QoS の値の設定方法は、着信パブリケーションによって異なります。 それは、着信パブリケーションが MQTT クライアントから送信されたものであるか IBM MQ アプリケーションから送信されたものであるかによって異なります。 **MQTT** 着信パブリケーションでの QoS とサブスクライバーによって要求された QoS のうち、低い方の値。 **IBM MQ** 着信パブリケーションから派生した QoS の低い方の値 MQPER_NOT_PERSISTENT→QoS=0 MQPER_PERSISTENT→QoS=2 およびサブスクライバーにより要求された QoS。 サブスクリプションなしでクライアントにメッセージが送信される場合、QoS はデフォルトで 2 に設定されます。 クライアントはこの値を、異なる QoS を指定して DEFAULT.QoS にサブスクライブすることによって変更できます。
RETAIN	ブール値	着信パブリケーションに保存プロパティー・セットがある場合に設定されます。

139 ページの表 13 は、MQTT クライアントに送信される MQTT メッセージで可変メッセージ・ヘッダーがどのように設定されるかを示しています。

表 13. *MQTT* クライアントに送信される *MQTT* メッセージで *MQTT* 可変ヘッダー・プロパティーが設定される方法

MQTT フィールド	タイプ	値
Topic name	文字列	パブリッシュされたメッセージに含まれるトピック・ストリング。
Message ID	文字列	パブリケーションが SYSTEM.MQTT.TRANSMIT.QUEUE に入れられたときのパブリケーションの MQMD.MsgId プロパティーの最後の 2 バイト。
Payload	byte[]	着信パブリケーションからパブリッシュ/サブスクライブ・ブローカーへのバイトの直接コピー。 長さがゼロの場合もあります。

Windows Linux AIX MQTT のステートレス・セッションとステートフル・セッション

MQTT クライアントは、キュー・マネージャーを使用してステートフル・セッションを作成できます。 ステートフル MQTT クライアントが切断した場合、キュー・マネージャーは、クライアントによって作成されたサブスクリプションと未完了メッセージを維持します。 クライアントが再接続されると、未完了メッセージを解決します。 配信用のキューに入れられているすべてのメッセージを送信し、切断中にそのサブスクリプションに対してパブリッシュされたすべてのメッセージを受信します。

MQTT クライアントは、遠隔測定チャネルに接続すると、新しいセッションを開始するか、前のセッションを再開します。 新しいセッションには、確認応答のない未解決メッセージも、サブスクリプションも、

配信を待機しているパブリケーションもありません。 クライアントは接続するときに、クリーン・セッションから開始するか、既存のセッションを再開するかを指定します (クリーン・セッションを参照)。

クライアントが既存のセッションを再開すると、接続の切断はなかったかのようにセッションが続行されます。 配信を待っているパブリケーションはクライアントに送信され、コミットされていないメッセージの転送はすべて完了されます。 クライアントが持続セッションで遠隔測定 (MQXR) サービスから切断しても、クライアントが作成したサブスクリプションは残ります。 サブスクリプションに対応するパブリケーションが、クライアントが再接続した時点でクライアントに送信されます。 クライアントが前のセッションを再開せずに再接続した場合、パブリケーションは遠隔測定 (MQXR) サービスによって破棄されます。

セッション状態情報は、キュー・マネージャーによって SYSTEM.MQTT.PERSISTENT.STATE キューに保存されます。

IBM MQ 管理者は、セッションを切断してパージすることができます。

Windows Linux AIX MQTT クライアントが接続されていないとき

クライアントが接続されていないときも、キュー・マネージャーはクライアントのためにパブリケーションの受信を続行できます。 こうしたパブリケーションは、クライアントが再接続した時点でクライアントに転送されます。 クライアントが予期せず切断した場合、キュー・マネージャーはクライアントに代わって、この"「遺言」"をパブリッシュします。

クライアントが予期せず切断したときに通知されるようにする場合は、遺言パブリケーションを登録できます (遺言パブリケーションを参照)。 この「遺言」は、クライアントの要求なしにクライアントへの接続が切断されていることを遠隔測定 (MQXR) サービスが検出した場合に、遠隔測定サービスによって送信されます。

クライアントは保存パブリケーションをいつでもパブリッシュできます (保存パブリケーションおよび MQTT クライアントを参照)。 トピックへの新規サブスクリプションは、トピックに関連付けられた保存パブリケーションを送信するよう要求できます。 「遺言」を保存パブリケーションとして作成すると、それを使用してクライアントの状況をモニターすることができます。

例えば、クライアントは接続時に保存パブリケーションをパブリッシュして、自身が使用可能であることを広告します。 同時に、自身が使用不可であることを通告するための「遺言」保存パブリケーションも作成します。 さらに、計画的な切断を行う直前に、自身が保存パブリケーションとして使用不可であることをパブリッシュします。 クライアントが使用可能かどうかを知るには、その保存パブリケーションのトピックにサブスクライブすることになります。 常に、3 つのパブリケーションのうちの 1 つを受信することになります。

クライアントがその切断中にパブリッシュされたメッセージを受信するようにする場合は、クライアントを前のセッションに再接続します (139 ページの『MQTT のステートレス・セッションとステートフル・セッション』を参照)。 前のセッションのサブスクリプションは、削除されるかクライアントがクリーン・セッションを作成するまではアクティブです。

Windows Linux AIX MQTT クライアントと IBM MQ アプリケーションの疎結合

MQTT クライアントと IBM MQ アプリケーション間のパブリケーションのフローは疎結合されています。 パブリケーションは、MQTT クライアントまたは IBM MQ アプリケーションのいずれかから、順不同で発信されます。 パブリッシャーとサブスクライバーは疎結合です。 これらは、パブリケーションとサブスクリプションを介して間接的に対話します。 MQTT アプリケーションから IBM MQ クライアントにメッセージを直接送信することもできます。

MQTT クライアントと IBM MQ アプリケーションは、以下の 2 つの理由により、疎結合していると言えます。

1. パブリッシャーとサブスクライバーは、パブリケーションおよびサブスクリプションのトピックとの関連により疎結合しています。 パブリッシャーとサブスクライバーは、通常、パブリケーションまたはサブスクリプションの他のソースのアドレスまたは ID を認識しません。
2. MQTT クライアントは、パブリケーションのパブリッシュ、サブスクライブ、受信、および送達確認の処理を、別々のスレッドで行います。

MQTT クライアント・アプリケーションは、パブリケーションの送達を待機しません。 アプリケーションは、メッセージを MQTT クライアントに渡してから、自身のスレッドの実行を続けます。 アプリケーションとパブリケーションの送達の同期を取るのに送達トークンが使用されます (送達トークンを参照)。

MQTT クライアントにメッセージを渡した後、アプリケーションには送達トークンを待つという選択肢があります。 クライアントは、待機するのではなく、パブリケーションが IBM MQ に送達されると呼び出されるコールバック・メソッドを定義できます。 このコールバック・メソッドは、送達トークンを無視することもできます。

メソッドに関連付けられたサービス品質によって、送達トークンはコールバック・メソッドに直ちに返されるか、場合によってはかなり時間がたってから返されます。 送達トークンは、クライアントが切断して再接続した後でも返されることがあります。 サービス品質が「応答不要送信」である場合、送達トークンは直ちに返されます。 これ以外の 2 つの場合は、サブスクライバーにパブリケーションが送信されたという確認応答をクライアントが受信したときのみ、送達トークンが返されます。

クライアント・サブスクリプションの結果として MQTT クライアントに送信されたパブリケーションは、`messageArrived` コールバック・メソッドに送達されます。 `messageArrived` は、メインアプリケーションとは別スレッドで動作します。

MQTT クライアントへのメッセージの直接送信

2 つの方法のいずれかを使用して、特定の MQTT クライアントにメッセージを送信することができます。

1. IBM MQ アプリケーションは、サブスクリプションなしで MQTT クライアントにメッセージを直接送信できます (クライアントへのメッセージの直接送信を参照)。
2. 別の方法は、`ClientIdentifier` 命名規則を使用します。 すべての MQTT サブスクライバーに、それぞれの固有の `ClientIdentifier` をトピックとして使用して、サブスクリプションを作成させます。 *ClientIdentifier* にパブリッシュします。 トピック *ClientIdentifier* にサブスクライブしたクライアントに、パブリケーションが送信されます。 この手法を使用すれば、パブリケーションを特定の MQTT サブスクライバーに送信できます。

Windows Linux AIX MQ Telemetry のセキュリティー

遠隔測定装置をセキュリティー保護することが重要になる場合があります。 装置がポータブルであることと、綿密に管理できない場所で使用されることが予想されるからです。 VPN を使用して、MQTT 装置から遠隔測定 (MQXR) サービスへの接続をセキュリティー保護できます。 MQ Telemetry では、これとは別の 2 つのセキュリティー・メカニズム (TLS と JAAS) を使用できます。

TLS は主に、装置と遠隔測定チャネルの間の通信を暗号化するため、および正しいサーバーに装置が接続しようとしていることを認証するために使用されます (TLS を使用する遠隔測定チャネルの認証を参照)。 TLS を使用して、クライアント装置がサーバーに接続することを許可されているかどうかを検査することもできます (TLS を使用する MQTT クライアントの認証を参照)。

JAAS は主に、装置のユーザーがサーバー・アプリケーションを使用することを許可されているかどうかを検査するために使用されます (パスワードを使用する MQTT クライアントの認証を参照)。 JAAS を LDAP と一緒に使用することにより、シングル・サインオン・ディレクトリーを使用してパスワードを検査できます。

TLS と JAAS を併用することにより、2 因子認証を行えます。 TLS によって使用される暗号を FIPS 標準に適合する暗号に制限できます。

少なくとも何万というユーザーがいれば、個人のセキュリティー・プロファイルを用意することは必ずしも現実的とは限りません。 そしてまた、個別ユーザーが IBM MQ オブジェクトにアクセスするのをプロファイルを使用して許可するのも、必ずしも現実的ではありません。 代わりに、トピックへのパブリケーションとサブスクリプションを許可するためのクラス、およびクライアントにパブリケーションを送信するためのクラスに、ユーザーをグループ化します。

各遠隔測定チャネルを構成して、クライアントをクライアント共通ユーザー ID にマップします。 特定のチャネルで接続するすべてのクライアントに、共通ユーザー ID を使用します (MQTT クライアントの ID および許可を参照)。

ユーザーのグループを許可することが、各個人の認証を阻害するわけではありません。 各個別ユーザーをクライアントまたはサーバーでそれぞれのユーザー名とパスワードで認証した後、サーバーで共通ユーザー ID を使用して許可することができます。

Windows Linux AIX MQ Telemetry グローバリゼーション

MQTT v3 プロトコルでは、メッセージ・ペイロードはバイト配列としてエンコードされます。 一般に、テキストを扱うアプリケーションは、UTF-8 でメッセージ・ペイロードを作成します。 遠隔測定チャネルは、メッセージ・ペイロードを UTF-8 であると示しますが、コード・ページ変換はまったく行いません。 パブリケーション・トピックのストリングは UTF-8 でなければなりません。

アプリケーションは、英字データを正しいコード・ページに変換し、数値データを正しい数値エンコード方式に変換する必要があります。

MQTT Java クライアントには、便利なメソッド `MqttMessage.toString` があります。 このメソッドは、メッセージ・ペイロードを、ローカル・プラットフォームのデフォルト文字セット (一般に UTF-8) でエンコードされているものとして扱います。 これはペイロードを Java String に変換します。 Java には、ストリングをローカル・プラットフォームのデフォルト文字セットを使用してエンコードされたバイト配列に変換する String メソッド `getBytes` があります。 2 つの MQTT Java プログラムが、同じデフォルト文字セットを持つプラットフォーム間で、メッセージ・ペイロード内のテキストを UTF-8 で非常に簡単かつ効率的に交換できます。

いずれかのプラットフォームのデフォルト文字セットが UTF-8 でない場合、アプリケーションはメッセージの交換のためのきまりを確立しなければなりません。 例えば、パブリッシャーは、`getBytes("UTF8")` メソッドを使用して、ストリングから UTF-8 への変換を指定します。 メッセージのテキストを受け取るには、サブスクライバーはメッセージが UTF-8 文字セットでエンコードされているものと見なします。

遠隔測定 (MQXR) サービスは、MQTT クライアントのメッセージからのすべての着信パブリケーションのエンコード方式を、UTF-8 であると示します。 `MQMD.CodedCharSetId` を UTF-8 に、`RFH2.CodedCharSetId` を `MQCCSI_INHERIT` に設定します。 136 ページの『MQ Telemetry とキュー・マネージャーの統合』を参照。 パブリケーションのフォーマットが `MQFMT_NONE` に設定されるので、チャネルによる、つまり MQGET による変換は行えません。

Windows Linux AIX MQ Telemetry のパフォーマンスと拡張容易性

多数のクライアントを管理するときや MQ Telemetry の拡張容易性を高めるときには、以下の要因を考慮に入れてください。

キャパシティー・プランニング

MQ Telemetry のパフォーマンス・レポートについては、MQ Performance documents を参照してください。

接続

接続に関係するコストには、以下の要素が含まれます。

- プロセッサー使用率とプロセッサー時間の点から見た接続そのもののセットアップ・コスト。
- ネットワーク・コスト。
- 開いたままの接続を使用しないときの使用メモリー。

クライアントが接続されたままのときは、さらに負荷が発生します。 接続が開いたままの場合、TCP/IP フローと MQTT メッセージではネットワークを使用して、まだ接続されているかどうかが確認されます。 加えて、開いたままのクライアント接続ごとに、サーバーでメモリーが使用されます。

メッセージを毎分複数回の頻度で送信する場合は、新たに接続を開始するときのコストを回避するために、接続を開いたままにしておきます。 メッセージを 10 分から 15 分ごとに 1 回未満の頻度で送信する場合は、接続を開いたままにしておくときのコストを回避するために、切断することを検討してください。 TLS

接続は、セットアップにより多くの費用がかかるため、使用されない状態で長時間開いたままにしておいてもかまいません。

また、クライアントの能力も考慮に入れてください。 クライアントにストア・アンド・フォワード機能がある場合は、メッセージをバッチにまとめて、バッチの送信と送信の間は接続を切断することができます。ただし、クライアントが切断されると、クライアントはサーバーからのメッセージを受信できなくなります。 したがって、判断にはアプリケーションの目的が関係します。

システムのクライアントが 1 つで、それが多数のメッセージを送信する場合は (例えばファイル転送)、メッセージごとにサーバーの応答を待つことはしないでください。 代わりに、すべてのメッセージを送信し、最後にそれらがすべて受信されたことを検査します。 あるいは、サービス品質 (QoS) を使用します。

QoS をメッセージによって変更することができます。QoS 0 を使用する重要でないメッセージと、QoS 2 を使用する重要なメッセージを送達することができます。 メッセージのスループットは、QoS が 2 の場合よりも、QoS が 0 の 2 倍程度になることがあります。

命名規則

多数のクライアントに対応するアプリケーションを設計する場合は、実効的な命名規則を実施してください。 各クライアントを正しい `ClientIdentifier` にマップするために、`ClientIdentifier` を意味のある名前にしてください。 適切な命名規則により、管理者が実行中のクライアントを判別しやすくなります。 命名規則は、管理者が IBM MQ エクスプローラーで多数のクライアントのリストをフィルタリングする際に役立ち、また、問題の判別にも役立ちます (クライアント ID を参照)。

スループット

トピック名の長さは、ネットワーク上を流れるバイト数に影響します。 パブリッシュまたはサブスクライブ時に、メッセージに含まれるバイト数が重要になることがあります。 したがって、トピック名の文字数を制限してください。 MQTT クライアントがトピックをサブスクライブすると、IBM MQ はそれに次の形式の名前を付けます。

```
ClientIdentifier: TopicName
```

MQTT クライアントのサブスクリプションをすべて表示するには、IBM MQ MQSC **DISPLAY** コマンドを次のように使用します。

```
DISPLAY SUB(' ClientID1:*')
```

IBM MQ クライアントが使用するための MQTT の定義リソース

MQTT クライアントは、IBM MQ (リモート・キュー・マネージャー) に接続します。 IBM MQ アプリケーションが MQTT クライアントにメッセージを送信するには、2 つの基本的な方法があります。デフォルト伝送キューを `SYSTEM.MQTT.TRANSMIT.QUEUE` に設定する方法と、キュー・マネージャー別名を使用する方法です。 MQTT クライアントが多数ある場合は、キュー・マネージャーのデフォルト伝送キューを定義します。 伝送キューのデフォルト設定を使用すると、管理作業が簡単になります (メッセージを MQTT クライアントに送信するように分散キューイングを構成を参照)。

サブスクリプションを回避することによる拡張容易性の向上

MQTT V3 クライアントがトピックにサブスクライブすると、遠隔測定 (MQXR) サービスによってサブスクリプションが IBM MQ で作成されます。 サブスクリプションにより、クライアントのパブリケーションが `SYSTEM.MQTT.TRANSMIT.QUEUE` に送信されます。 各パブリケーションの伝送ヘッダーにあるリモート・キュー・マネージャー名は、サブスクリプションを作成した MQTT クライアントの `ClientIdentifier` に設定されます。 多数のクライアントがそれぞれ独自のサブスクリプションを行う場合は、IBM MQ パブリッシュ/サブスクライブのクラスターまたは階層の全体にわたって、多数のプロキシー・サブスクリプションが維持されることになります。 パブリッシュ/サブスクライブの代わりに、Point-

to-Point ベースのソリューションを使用する方法については、クライアントへのメッセージの直接送信を参照してください。

多数のクライアントの管理

同時に接続される多数のクライアントをサポートするには、JVM パラメーターの **-Xms** と **-Xmx** を設定して、遠隔測定 (MQXR) サービスで使用可能なメモリーを増やします。 以下のステップに従ってください: 。

1. テレメトリー・サービス構成ディレクトリー内の `java.properties` ファイルを見つけます。 Windows 上のテレメトリー (MQXR) サービス構成ディレクトリー または Linux 上のテレメトリー・サービス構成ディレクトリー を参照してください。
2. ファイルの指示に従います。同時に接続されるクライアントの数が 50,000 の場合は、1 GB のヒープで十分です。

```
# Heap sizing options - uncomment the following lines to set the heap to 1G
#-Xmx1024m
#-Xms1024m
```

3. `java.properties` ファイル内でテレメトリー (MQXR) サービスを実行している JVM に渡すために、他のコマンド行引数を追加します。テレメトリー (MQXR) サービスへの JVM パラメーターの引き渡し を参照してください。

Linux 上のオープン・ファイル記述子の数を増やすには、以下の行を `/etc/security/limits.conf/`に追加し、再度ログインしてください。

```
@mqm soft nofile 65000
@mqm hard nofile 65000
```

ソケットごとに 1 つのファイル記述子を必要とします。 遠隔測定サービスでさらにファイル記述子がいくつか必要になるため、この数は、必要なオープン・ソケットの数よりも多くする必要があります。

キュー・マネージャーは、非永続サブスクリプションごとに 1 つのオブジェクト処理を使用します。 多数のアクティブな非永続サブスクリプションをサポートするには、キュー・マネージャーで使用できるアクティブ・ハンドルの最大数を増やします。例えば、以下のようにします。

```
echo ALTER QMGR MAXHANDS(999999999) | runmqsc qMgrName
```

図 *48. Windows* でのハンドルの最大数の変更

```
echo "ALTER QMGR MAXHANDS(999999999)" | runmqsc qMgrName
```

図 *49. Linux* でのハンドルの最大数の変更

その他の考慮事項

システム要件を計画する際は、システムの再始動に要する時間の長さを考慮に入れてください。 計画したダウン時間が、処理待ちのキューに入れられるメッセージの数に影響する可能性があります。 許容時間内にメッセージを正常に処理できるように、システムを構成してください。 ディスク・ストレージ、メモリー、および処理能力を確認してください。 一部のクライアント・アプリケーションでは、クライアントの再接続時にメッセージを破棄できる場合があります。 メッセージを破棄するには、クライアント接続パラメーターで `CleanSession` を設定します (クリーン・セッションを参照)。 あるいは、 MQTT クライアントでベスト・エフォートのサービス品質 `0` を使用してパブリッシュおよびサブスクライブを行います。 サービス品質を参照してください。 IBM MQ からメッセージを送信する場合は、**非持続** メッセージを使用します。 これらのサービス品質を使用したメッセージは、システムまたは接続の再始動時には回復されません。

Windows Linux AIX MQ Telemetry によってサポートされる装置

MQTT クライアントは、センサーとアクチュエーターからハンドヘルド・デバイス、車両システムまで、さまざまな装置の上で実行できます。

MQTT クライアントは小さいので、小容量メモリーや低処理能力に制約される装置上で動作します。 MQTT protocol は信頼性が高くヘッダーが小さいので、低帯域幅、高コスト、および使用可能状態が断続的であることに制約されるネットワークに適しています。

MQ Telemetry は、MQTT クライアント・アプリケーションを介して遠隔測定装置と通信します。 このようなアプリケーションでは以下のリソースを使用します。どのリソースも MQTT v3 プロトコルを実装しています。

- 以下のクライアント・ライブラリー
  - *MQTT client for Java*。例えば、Android、OS X、Linux、Windows デバイスなどのネイティブ・アプリケーションの作成に使用されます。 このクライアント・ライブラリーを使用するアプリケーションは、最小の CLDC (Connected Limited Device Configuration)/MIDP (Mobile Information Device Profile) から、CDC (Connected Device Configuration)/Foundation、J2SE (Java Platform Standard Edition)、J2EE (Java Platform Enterprise Edition) に至るまで、あらゆるバリエーションの Java 上で実行可能です。 IBM jclRM カスタマイズ・クラス・ライブラリーもサポートされています。 Java ME プラットフォームは一般に、アクチュエーター、センサー、携帯電話などの組み込みデバイスのような小型装置上で使用されます。 Java SE プラットフォームは一般に、デスクトップ・コンピューターやサーバーなどのより高性能な組み込み装置にインストールされます。
  - *MQTT client for C*。これは、ネイティブ・アプリケーション (例えば、iOS、OS X、Linux、または Windows 装置) の作成に使用されます。 このクライアント・ライブラリーは、Windows および Linux システム用の事前ビルドされたネイティブ・クライアントと一緒に C 参照インプリメンテーションを提供します。 この C リファレンス実装に基づいて、広範囲の装置やプラットフォームに MQTT を移植できます。 WindowsIntel 7、RedHat、Ubuntu、および ARM プラットフォーム上のいくつかの Windows システム (Eurotech Viper など) の一部の Linux システムは、C クライアントを実行する Linux のバージョンを実装していますが、IBM ではプラットフォームに対するサービス・サポートは提供されません。 IBM サポート・センターに連絡を取る場合は、サポートされているプラットフォーム上でクライアントの問題を再現しておく必要があります。
  - *MQTT client for Java*。 ブラウザー・ベースの Web アプリケーションの作成に使用します。

MQTT クライアント・ライブラリーを Eclipse Paho および MQTT.org から無料で入手することができます。 IBM MQ Telemetry Transport サンプル・プログラムを参照してください。

IBM MQ のセキュリティー

IBM MQ では、セキュリティーを提供するいくつかの方法があります。許可サービス・インターフェース、ユーザー作成またはサード・パーティーのチャネル出口、Transport Layer Security (TLS) を使用したチャネル・セキュリティー、チャネル認証レコード、およびメッセージ・セキュリティーです。

許可サービス・インターフェース

MQI 呼び出しの使用、コマンドの発行、およびオブジェクトへのアクセスは、**オブジェクト権限マネージャー** (OAM) によって提供されます。OAM はデフォルトでは使用可能です。 IBM MQ エンティティーへのアクセスは、IBM MQ ユーザー・グループおよび OAM を通して制御されます。 管理者はコマンド行インターフェースを使用して、必要に応じて許可を与えたり取り消したりすることができます。

許可サービス・コンポーネントの作成の詳細については、AIX, Linux, and Windows システムでのセキュリティーのセットアップを参照してください。

ユーザー作成またはサード・パーティーのチャネル出口

チャネルでは、ユーザー作成出口またはサード・パーティー出口を使用できます。 詳細については、メッセージング・チャネルのためのチャネル出口プログラムを参照してください。

TLS を使用したチャネル・セキュリティー

Transport Layer Security (TLS) プロトコルは、業界標準のチャネル・セキュリティーを提供し、盗聴、改ざん、偽名の使用に対して保護します。

TLS は、公開鍵とシンメトリック手法を使用して、メッセージの機密性と保全性、および相互認証を提供します。

TLS の詳細情報を含む IBM MQ のセキュリティーに関する総合的なレビューについては、セキュリティーを参照してください。 このセクションで説明したコマンドのポインターを含めた TLS の概要については、暗号セキュリティー・プロトコル: TLS を参照してください。

チャネル認証レコード

チャネル認証レコードを使用して、チャネル・レベルで接続システムに許可されているアクセスに対して正確な制御を行います。 詳しくは、チャネル認証レコードを参照してください。

メッセージ・セキュリティー

Advanced Message Security (別個にインストールおよびライセンス交付される IBM MQ のコンポーネント) を使用し、IBM MQ を使用して送受信されるメッセージに対する暗号保護を提供します。 Advanced Message Security を参照してください。

関連タスク

セキュリティー

セキュリティー要件の計画

IBM MQ.NET 管理対象クライアントの TLS サポート

IBM MQ.NET の完全管理クライアントは、Microsoft.NET SSLStreams キットに基づく Transport Layer Security (TLS) サポートを提供します。 これは、IBM Global Security Kit (GSKit)に基づく他の IBM MQ クライアントとは異なります。

IBM MQ.NET アプリケーションを開発して、管理モードまたは非管理モードで実行できます。

- 管理モードでは、.NET アプリケーションは .NET CLR (共通言語ランタイム) 内で、C MQI 呼び出しなどのクロス・プラットフォーム呼び出しなしで動作します。
- 非管理モードでは、基礎となる MQI 操作のために C MQI が呼び出されます。 基本的には、非管理モード・インターフェースは、C MQI の上にある .NET ラッパー・クラスで構成されます。

管理 IBM MQ.NET クライアントは、Microsoft.NET Framework ライブラリーを使用して、TLS セキュア・ソケット・プロトコルを実装します。 Microsoft の System.NET.Security.SSLStream クラスは、 IBM MQ.NET でセキュリティー (TLS) を実装するために使用されます。

非管理対象 IBM MQ.NET クライアント・モードは、C MQI (および GSKit) に基づく TLS 機能を既にサポートしています。 つまり、TLS 操作は C MQI によって処理されます。 この場合、 GSKit は TLS セキュア・ソケット・プロトコルを実装します。

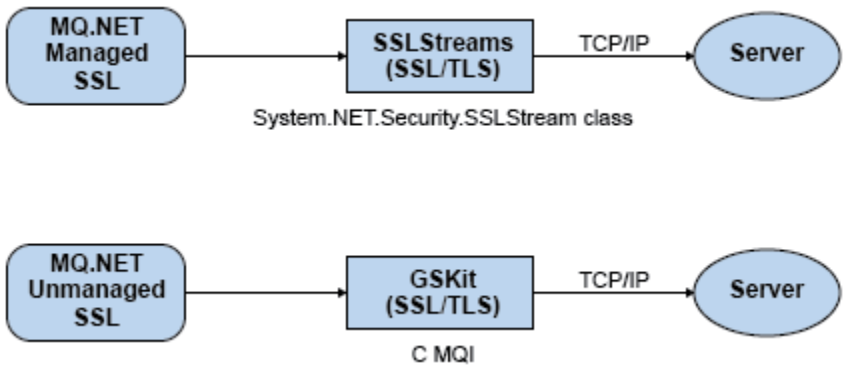


図 *50. IBM MQ.NET* 管理 *TLS* と非管理 *TLS* の比較

次の表に、管理実装と非管理実装の違いについてまとめます。

表 *14.* 管理実装と非管理実装の違い

モード (Mode)	プロトコル	実装	コメント
IBM MQ.NET 管理 SSL	TLS	System.NET.Security.SSLStream クラス SSLStream クラスは、接続済み TCP ソケット上のストリームとして作動します	TLS 1.0 TLS 1.2 (Microsoft.NET Framework v4.5 でのみ)
IBM MQ.NET 非管理 SSL	TLS	GSKIT および C-MQI	TLS セキュア・ソケット・プロトコル

関連概念

.NET のための Secure Sockets Layer (SSL) および Transport Layer Security (TLS) のサポート

IBM MQ MQI clients

IBM MQ MQI client は、キュー・マネージャーが実行されていないシステム上にインストールできる、IBM MQ 製品のコンポーネントです。

IBM MQ MQI クライアント とは、あるシステム上で稼働しているアプリケーションが、別のシステム上で稼働しているキュー・マネージャーに MQI 呼び出しを発行できるようにするコンポーネントのことです。 呼び出しからの出力は クライアントに返送され、さらにクライアントからアプリケーション に戻されます。

IBM MQ MQI client を使用すると、クライアントと同じシステム上で実行されているアプリケーションが、別のシステム上で実行されているキュー・マネージャーに接続することができます。 アプリケーションは、そのキュー・マネージャーに対して MQI 呼び出しを発行できます。 このようなアプリケーションは、IBM MQ MQI client・アプリケーションと呼ばれ、キュー・マネージャーはサーバー・キュー・マネージャーと呼ばれます。

IBM MQ サーバー とは、キューイング・サービスを 1 つ以上のクライアントに提供するキュー・マネージャーのことです。 キューなどのすべての IBM MQ オブジェクトは、キュー・マネージャーのマシン上 (IBM MQ サーバー・マシン) にのみ存在し、クライアント上には存在 しません。 IBM MQ サーバーは、ローカルの IBM MQ アプリケーションもサポートすることができます。

IBM MQ サーバーと通常のキュー・マネージャーとの相違点は、サーバーには、各クライアントとの専用通信リンクが備わっているという点です。 クライアントとサーバーにチャネルを作成する方法の詳細については、分散キューイングの構成を参照してください。

IBM MQ MQI client・アプリケーションとサーバー・キュー・マネージャーは、*MQI* チャネルを使用して相互に通信します。 MQI チャネルは、クライアント・アプリケーションが、キュー・マネージャーに接続するための **MQCONN** または **MQCONNX** 呼び出しを発行した時に開始されます。また、MQI チャネルは、クライアント・アプリケーションが、キュー・マネージャーとの接続を解除する **MQDISC** 呼び出しを発行した時に終了されます。 MQI 呼び出しの入力パラメーターは、MQI チャネル上で 1 つの方向に流れ、出力パラメーターは、それと反対の方向に流れます。

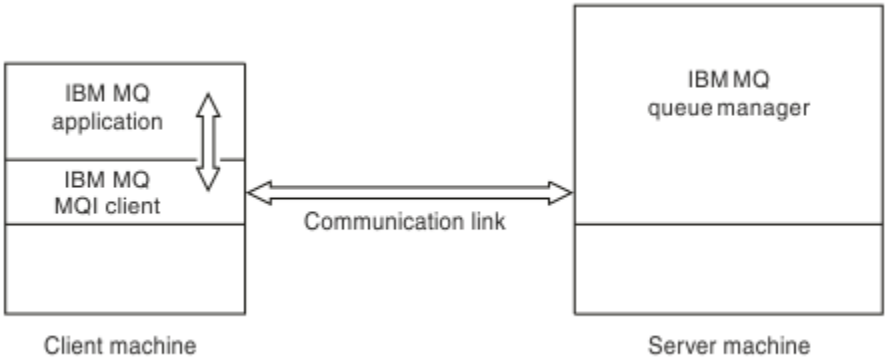


図 *51.* クライアントとサーバー間のリンク

次のプラットフォームを使用できます。 組み合わせは、使用する IBM MQ 製品によって異なります。 この点については、149 ページの『IBM MQ クライアントのプラットフォームのサポート』 で説明されています。

IBM MQ MQI client	**IBM MQ サーバー**
AIX and Linux Windows IBM i	AIX and Linux Windows IBM i z/OS

MQI は、クライアント・プラットフォーム上で稼働するアプリケーションで使用することができます。 キューおよびその他の IBM MQ オブジェクトは、サーバー上にインストールされたキュー・マネージャー上に保存されます。

IBM MQ MQI client 環境で実行するアプリケーションを、最初に、関連したクライアント・ライブラリーにリンクする必要があります。 アプリケーションが MQI 呼び出しを発行すると、IBM MQ MQI client は要求をキュー・マネージャーに送ります。キュー・マネージャーでは要求が処理され、応答が IBM MQ MQI client に返されます。

アプリケーションと IBM MQ MQI client の間のリンクは、実行時に動的に確立されます。

IBM MQ classes for .NET、IBM MQ classes for Java または IBM MQ classes for Java Message Service (JMS) を使って顧客のアプリケーションを開発することもできます。以下のプラットフォームで Java および JMS 顧客を使用することができます。

- IBM i
- AIX
- Linux
- Windows

Java および JMS の使用についてはここでは説明しません。 IBM MQ classes for Java および IBM MQ classes for JMS のインストール、構成および使用の方法に関する詳細については、IBM MQ classes for Java の使用および IBM MQ classes for JMS の使用をご参照ください。

クライアント/サーバー環境での IBM MQ アプリケーション

サーバーにリンクされていれば、クライアント IBM MQ アプリケーションは、ローカル・アプリケーションと同じやり方でたいていの MQI 呼び出しを発行することができます。 クライアント・アプリケーションは、MQCONN 呼び出しを発行して、指定のキュー・マネージャーに接続します。 接続要求から戻された接続ハンドルを指定した追加の MQI 呼び出しは、このキュー・マネージャーによって処理されます。

ユーザーのアプリケーションを、該当するクライアント・ライブラリーにリンクする必要があります。 IBM MQ MQI clients 用のアプリケーションの作成を参照してください。

関連概念

149 ページの『IBM MQ クライアントを使用する理由』
IBM MQ クライアントを使用することは、IBM MQ メッセージングおよびキューイングを効率的に実装する方法です。

151 ページの『拡張トランザクション・クライアントの概要』
IBM MQ 拡張トランザクション・クライアントは、外部トランザクション・マネージャーの制御の元で、別のリソース・マネージャーが管理するリソースを更新できます。

152 ページの『クライアントとサーバーの接続方法』
クライアントは MQCONN または MQCONNX を使ってサーバーに接続し、チャネルを通して通信します。

153 ページの『トランザクションの管理とサポート』
トランザクション管理、および IBM MQ がトランザクションをサポートをする方法に関する概要です。

155 ページの『キュー・マネージャーの機能の拡張』
ユーザー出口、API 出口、またはインストール可能サービスを使用して、キュー・マネージャーの機能を拡張できます。

関連情報

IBM MQ MQI client のセットアップ方法

IBM MQ クライアントを使用する理由

IBM MQ クライアントを使用することは、IBM MQ メッセージングおよびキューイングを効率的に実装する方法です。

MQI とキュー・マネージャーを別々のマシン上 (物理マシン、仮想マシンのどちらでも) で実行するようにアプリケーションを作成することもできます。 MQI とキュー・マネージャーを別のマシンで実行すると、次のような利点があります。

- クライアント・マシン上に IBM MQ を完全に実装する必要がなくなります。
- クライアント・システム側のハードウェア要件が減少します。
- システム管理要件が減少します。
- クライアントで実行する IBM MQ アプリケーションをさまざまなシステム上の複数のキュー・マネージャーに接続できます。
- 異なる伝送プロトコルを使用する代替チャネルが使用できます。

IBM MQ クライアントのプラットフォームのサポート

サポートされるすべてのサーバー・プラットフォーム上の IBM MQ は、複数のプラットフォーム上の IBM MQ MQI clients からのクライアント接続を受け入れます。

サポートされるすべてのサーバー・プラットフォーム上に *Base* 製品およびサーバー としてインストールされた IBM MQ は、以下のプラットフォーム上の IBM MQ MQI clients からの接続を受け入れることができます。

- IBM i
- AIX
- Linux
- Windows

クライアント接続によって、コード化文字セット ID (CCSID) および通信プロトコルが異なる場合があります。

IBM MQ MQI client 上で実行するアプリケーション

クライアント環境では MQI が完全にサポートされています。 これにより、IBM MQ MQI client 上のアプリケーションを MQI ライブラリーではなく MQIC ライブラリーにリンクすることで、ほとんどすべての IBM MQ アプリケーションを IBM MQ MQI client システム上で実行するように構成できます。 ただし、次の例外があります。

- 信号付きの MQGET
- 他のリソース管理プログラムとの同期点の調整を必要とするアプリケーションでは、拡張トランザクション・クライアントを使用する必要があります。

非持続メッセージングのパフォーマンスを良くするために先読みを使用可能にした場合、すべての MQGET オプションが使用可能な訳ではありません。 次の表は、使用できるオプションとそれらを MQGET 呼び出しの間に変更できるかどうかを示します。

表 *15.* 先読みが有効になっている場合に使用できる *MQGET* オプション

値	先読みが有効になっている場合に使用でき、**MQGET** 呼び出し間で変更できる	先読みが有効になっている場合に使用でき、**MQGET** 呼び出し間で変更できない [1]	先読みが有効になっている場合に使用できない **MQGET** オプション [2]
MQGET MD 値	MsgId [3] CorrelId [3]	Encoding CodedCharSetId	
MQGET MQGMO オプション	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 MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_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 MQGMO_SYNCPOINT MQGMO_MARK_SKIP_BACKOUT MQGMO_MSG_UNDER_CURSOR [4] MQGMO_LOCK MQGMO_UNLOCK
MQGMO 値		MsgHandle	

1. これらのオプションが MQGET 呼び出し間で変更された場合、MQRC_OPTIONS_CHANGED 理由コードが戻されます。
2. これらのオプションが最初の MQGET 呼び出しで指定されると、先読みは使用不可になります。 これらのオプションを後続の MQGET 呼び出しで指定すると、理由コード MQRC_OPTIONS_ERROR が戻されます。
3. クライアント・アプリケーション側で以下の点に留意する必要があります。すなわち、MsgId および CorrelId の値が MQGET 呼び出し間で変更された場合、変更前の値によるメッセージがクライアントに送信済みの可能性があり、コンシューム (または自動的にパージ) されるまでクライアントの先読みバッファー内に残るということです。
4. 最初の MQGET 呼び出しは、先読みが有効である場合にメッセージをキューからブラウズするか取得するかを決定します。 アプリケーションがブラウズと取得の組み合わせを使用しようとすると、MQRC_OPTIONS_CHANGED 理由コードが戻されます。
5. MQGMO_MSG_UNDER_CURSOR は先読みでは使用できません。 先読みが有効な場合、メッセージのブラウズまたは取得が可能ですが、ブラウズと取得の組み合わせは指定できません。

IBM MQ MQI client 上で動作するアプリケーションは、同時に複数のキュー・マネージャーに接続できます。また、MQCONN や MQCONNX を呼び出すときに、キュー・マネージャー名にアスタリスク (*) を付けて使用することができます (キュー・マネージャーへの IBM MQ MQI client・アプリケーションの接続の例を参照)。

拡張トランザクション・クライアントの概要

IBM MQ 拡張トランザクション・クライアントは、外部トランザクション・マネージャーの制御の元で、別のリソース・マネージャーが管理するリソースを更新できます。

トランザクション管理の概念についてよくご存じでない場合は、153 ページの『トランザクションの管理とサポート』を参照してください。

XA トランザクション・クライアントは、IBM MQ の一部として提供されるようになったことに注意してください。

クライアント・アプリケーションは、接続先のキュー・マネージャーによって管理される作業単位に加わることができます。 その作業単位内で、クライアント・アプリケーションは、キュー・マネージャーが所有するキューにメッセージを書き込んだり、キューからメッセージを取得したりすることができます。 次に、クライアント・アプリケーションは、**MQCMIT** 呼び出しを使用して作業単位をコミットするか、**MQBACK** 呼び出しを使用して作業単位をバックアウトすることができます。 しかし、クライアント・アプリケーションは同じ作業単位内で、別のリソース・マネージャーのリソース、例えば Db2® データベースの表を更新することはできません。 IBM MQ 拡張トランザクション・クライアントを使用すると、この制限がなくなります。

IBM MQ 拡張トランザクション・クライアントは、いくつかの追加機能を持つ IBM MQ MQI client です。この機能を使用して、クライアント・アプリケーションは、同一の作業単位内で次のタスクを実行できます。

- 接続先のキュー・マネージャーが所有するキューにメッセージを書き込み、そのキューからメッセージを取得する
- IBM MQ キュー・マネージャー以外のリソース・マネージャーのリソースを更新する

この作業単位は、クライアント・アプリケーションと同じシステム上で実行されている外部トランザクション・マネージャーによって管理されなければなりません。 クライアント・アプリケーションの接続先のキュー・マネージャーが作業単位を管理することはできません。 つまり、キュー・マネージャーは、リソース・マネージャーの役目だけをすることができ、トランザクション・マネージャーの役目をすることはできません。 また、クライアント・アプリケーションは、作業単位のコミットまたはバックアウトを行うのに、外部トランザクション・マネージャーが提供するアプリケーション・プログラミング・インターフェース (API) しか使用できません。 したがって、クライアント・アプリケーションは、MQI 呼び出し **MQBEGIN**、**MQCMIT**、 および **MQBACK** を使用できません。

外部トランザクション・マネージャーは、キュー・マネージャーに接続されるクライアント・アプリケーションによって使用されるのと同じ MQI チャネルを使用して、リソース・マネージャーとしてのキュー・マネージャーと情報を交換できます。 しかし、障害後のリカバリー状態では、アプリケーションが実行していないので、トランザクション・マネージャーは、専用の MQI チャネルを使用して、障害の時点でキュー・マネージャーが参加していた未完了の作業単位をリカバリーすることができます。

このセクションでは、拡張トランザクション機能を持たない IBM MQ MQI client は、IBM MQ ベース・クライアントと呼ばれます。 したがって、IBM MQ 拡張トランザクション・クライアントは、拡張トランザクション機能が付加された IBM MQ ベース・クライアントであると見なすことができます。

注 : IBM i IBM i 上の IBM MQ MQI client は、IBM MQ 拡張トランザクション機能をサポートしません。

拡張トランザクション・クライアントのプラットフォームのサポート

Multi

拡張トランザクション・クライアントは、ベース・クライアントをサポートするすべての Multiplatforms で利用可能です。 クライアントは z/OS では利用できません。

拡張トランザクション・クライアントを使用するクライアント・アプリケーションは、以下の IBM MQ 9.0 以降の製品のキュー・マネージャーにのみ接続できます。

- 

IBM MQ for AIX
- 

IBM MQ for IBM i
- 

IBM MQ の Linux
- 

IBM MQ for Windows

z/OS で稼働する拡張トランザクション・クライアントはありませんが、拡張トランザクション・クライアント を使用しているクライアント・アプリケーションが z/OS で稼働するキュー・マネージャーに接続することはできます。

各プラットフォームでは、拡張トランザクション・クライアントのハードウェアおよびソフトウェアの要件は、IBM MQ ベース・クライアントの要件と同じです。 プログラム言語は、IBM MQ ベース・クライアントおよびご使用のトランザクション・マネージャーによってサポートされている場合は、拡張トランザクション・クライアントによってもサポートされます。

すべてのプラットフォームの外部トランザクション・マネージャーについては、 IBM MQ のシステム要件 を参照してください。

クライアントとサーバーの接続方法

クライアントは MQCONN または MQCONNX を使ってサーバーに接続し、チャネルを通して通信します。

IBM MQ クライアント環境で実行されるアプリケーションは、クライアント・マシンとサーバー・マシン間の接続をアクティブに保つ必要があります。

接続は、アプリケーションが MQCONN 呼び出しまたは MQCONNX 呼び出しを発行することによって、確立されます。 クライアントとサーバーは、*MQI* チャネル を介して通信します。ただし、共用会話の使用時には、それぞれの会話どうしが MQI チャネル・インスタンスを共有します。 その呼び出しが成功すると、アプリケーションが MQDISC 呼び出しを発行するまで、MQI チャネル・インスタンスまたは会話は接続されたままになります。 このことは、アプリケーションが接続する必要のあるどのキュー・マネージャーにも該当します。

同一マシン上のクライアントとキュー・マネージャー

また、ご使用のマシンにキュー・マネージャーがインストールされている場合は、 IBM MQ MQI client 環境でアプリケーションを実行することもできます。

この場合、キュー・マネージャーのライブラリー、クライアントのライブラリーのどちらにでも、リンクすることができます。ただし、クライアントのライブラリーにリンクする場合でも、チャネル接続を定義する必要があることに注意してください。 キュー・マネージャーがインストールされているマシンで、クライアント環境を実現できると、アプリケーションの開発段階で役に立ちます。 他のマシンに依存することなく、開発者自身のマシン上でプログラムをテストすることができ、独立した IBM MQ MQI client 環境に移動したときに、これまでどおり順調に稼働させることができます。

異なるプラットフォーム上のクライアント

この例では、サーバー・マシンは異なるプラットフォーム上で 3 つの IBM MQ MQI clients と通信します。

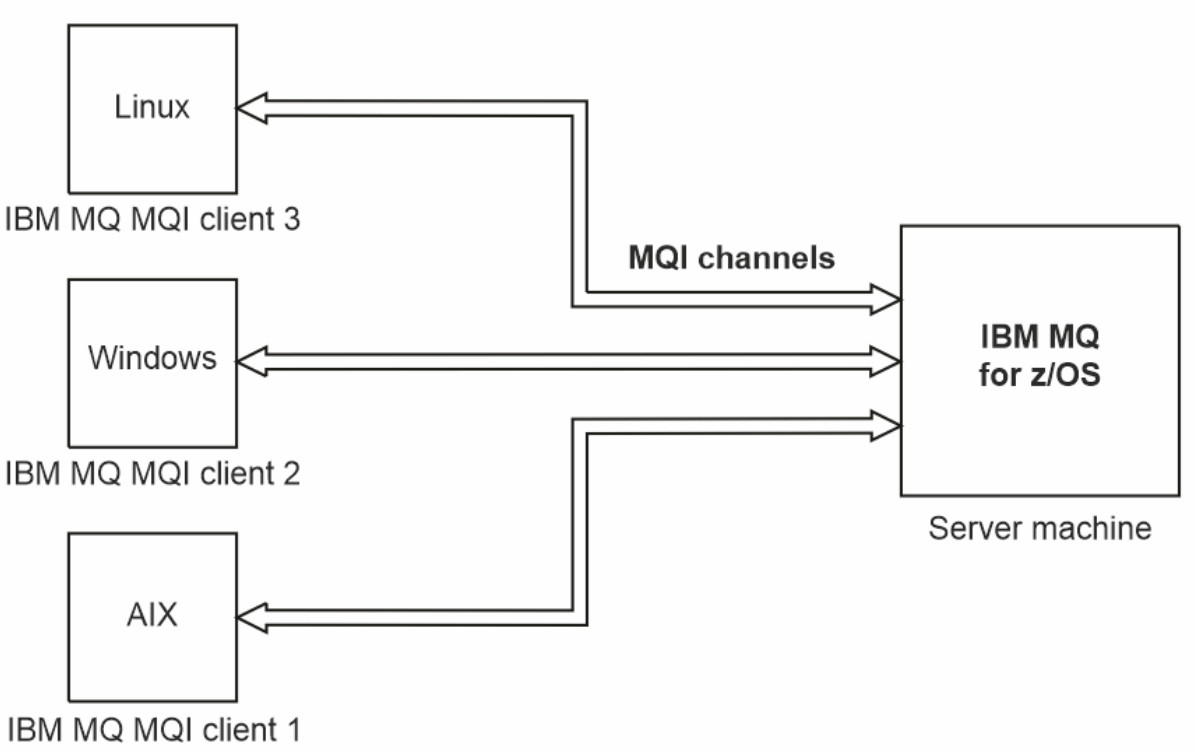


図 *52.* 異なるプラットフォーム上のクライアントに接続された *IBM MQ* サーバー

他のもっと複雑な環境も可能です。 例えば、1 つの IBM MQ クライアントが複数のキュー・マネージャーに接続する環境や、キュー共用グループの一部として接続されている任意の数のキュー・マネージャーに接続する環境も実現できます。

異なるバージョンのクライアントおよびサーバー・ソフトウェアの使用

旧バージョンの IBM MQ 製品を使用する場合は、クライアントの CCSID からのコード変換がサーバーでサポートされることを確認してください。

IBM MQ クライアントは、サポートされるすべてのバージョンのキュー・マネージャーに接続できます。 旧バージョンのキュー・マネージャーに接続する場合は、クライアント上の IBM MQ アプリケーションで、製品のより新しいバージョンの機能および構造体は使用できません。

IBM MQ キュー・マネージャーは、相互にサポートされている最も高いプロトコル・レベルまでネゴシエーションすることにより、異なるバージョンのクライアントとそれ自体と通信することができます。 これは、古いクライアントが新しいキュー・マネージャー・レベルで使用される可能性があることを意味します。 問題診断を容易にし、 IBM によるサポートを有効にするために、クライアントとサーバーの両方が、現在サポートされている IBM MQ のバージョンであることをお勧めします。

詳しくは、 アプリケーションの開発でサポートされているプログラミング言語を参照してください。

トランザクションの管理とサポート

トランザクション管理、および IBM MQ がトランザクションをサポートをする方法に関する概要です。

リソース・マネージャー は、アプリケーションによるアクセスと更新が可能なリソースを所有し管理する、コンピューター・サブシステムです。 以下にリソース・マネージャーの例を示します。

- IBM MQ キュー・マネージャー、キューがリソース
- Db2 データベース、テーブルがリソース

アプリケーションが 1 つ以上のリソース・マネージャーのリソースを更新する場合、所定の更新がすべて 1 つのグループとして 正常に必ず完了するようにする、またはすべて完了しないようにすることが、ビジ

ネス上必要になる場合があります。 こうした必要が生じる理由は、これらの更新の中で正常に完了しているものと、完了していないものがあると、ビジネス・データが不整合のままになるからです。

このように管理されるリソースの更新は、作業単位、またはトランザクション 内で行われます。 アプリケーション・プログラムは、更新のセットを作業単位にグループ化できます。

1 つの作業単位の中で、アプリケーションは、リソース・マネージャーのリソースを更新する要求をリソース・マネージャーに出します。 アプリケーションが、すべての更新をコミットする要求を出すと、この作業単位は終了します。 更新がコミットされるまで、それらの更新は、同じリソースにアクセスする他のアプリケーションから見えません。 または、アプリケーションは、なんらかの理由で作業単位を完了できないと判断した場合、その時点までに要求したすべての更新をバックアウトする要求を出すことができます。この場合、これらの更新はいずれも、他のアプリケーションから見えません。 作業単位としてグループ化される更新は相互に論理的に関連しているため、データ保全性を維持できるように正常に処理されなければなりません。 ある更新が正常に行われても、別の更新が失敗すれば、データ保全性は失われます。

作業単位が正常に完了したことを、コミット されたといいます。 コミットされると、作業単位内のすべての更新内容が永久的になり、これ以降は取り消し不可能になります。 しかし、作業単位が失敗した場合は、すべての更新がバックアウト されます。 このような、データの保全性を維持しながら作業単位をコミットまたはバックアウトするプロセスのことを 同期点調整 といいます。

作業単位内のすべての更新がコミットされるか、バックアウトされる時点は、同期点 と呼ばれます。 作業単位内の更新は、同期点制御内 で行われます。 アプリケーションが、同期点制御外 にある更新を要求する場合、リソース・マネージャーは、進行中の作業単位があっても、その更新をただちにコミットします。この更新を後でバックアウトすることはできません。

作業単位を管理するコンピューター・サブシステムは、トランザクション・マネージャー、またはポイント・コーディネーター と呼ばれます。

ローカル 作業単位とは、IBM MQ キュー・マネージャーのリソースのみが更新対象のリソースとなる作業単位をいいます。 この場合、キュー・マネージャー自体が単一フェーズ・コミットによって同期点を調整します。

グローバル 作業単位では、XA 準拠データベースなどの他のリソース・マネージャーにより管理されているリソースも更新されます。 この場合、必ず 2 フェーズ・コミットが使用され、作業単位がキュー・マネージャー自体によって調整されるか、あるいは IBM TXSeries® や BEA Tuxedo などの XA 準拠トランザクション・マネージャーによって外部から調整されます。

トランザクション・マネージャーは、作業単位内のリソースの更新がすべて正常に完了したか、完了しなかったかを確認します。 アプリケーションが、作業単位のコミットまたはバックアウトを要求するのは、トランザクション・マネージャーに対してです。 例えば、トランザクション・マネージャーの例として CICS や WebSphere Application Server がありますが、どちらも他の機能も持っています。

一部のリソース・マネージャーは、独自のトランザクション管理機能を備えています。 例えば、IBM MQ キュー・マネージャーは、独自のリソースの更新および Db2 表の更新を伴う作業単位を管理できます。 キュー・マネージャーは、この機能を実行するのに別個のトランザクション・マネージャーを必要としません。ただし、ユーザーの要求であれば、トランザクション・マネージャーを使用することはできます。 別個のトランザクション・マネージャーを使用する場合、外部トランザクション・マネージャー と呼ばれます。

外部トランザクション・マネージャーが作業単位を管理するには、トランザクション・マネージャーと、作業単位に加わっているすべてのリソース・マネージャーとの間に、標準インターフェースが存在している必要があります。 このインターフェースにより、トランザクション・マネージャーとリソース・マネージャーは互いに情報を交換することが可能になります。 これらのインターフェースの 1 つが *XA* インターフェース です。XA インターフェースは、複数のトランザクション・マネージャーとリソース・マネージャーによってサポートされる標準インターフェースです。 XA インターフェースは、The Open Group によって *Distributed Transaction Processing: The XA Specification* の中で公開されています。

複数のリソース・マネージャーが 1 つの作業単位に加わっている場合、トランザクション・マネージャーは、システム障害の場合であっても、*2* フェーズ・コミット ・プロトコルを使用して、その作業単位内のすべての更新が正常に完了するか、完了しないかを確認する必要があります。 アプリケーションが、作業単位をコミットする要求をトランザクション・マネージャーに出す場合、トランザクション・マネージャーは次のことを行います。

フェーズ 1 (コミットの準備)
トランザクション・マネージャーは、作業単位に加わっている各リソース・マネージャーに対して、そのリソースの対象の更新についてのすべての情報が、リカバリー可能な状態であることを確認するように求めます。 リソース・マネージャーは通常これを確認するために、情報をログに書き込み、その情報がハード・ディスクに書き込まれるようにします。 トランザクション・マネージャーが、そのリソースの対象の更新についての情報が、リカバリー可能な状態であるという通知を各リソース・マネージャーから受け取ると、フェーズ 1 が完了します。

フェーズ 2 (コミットの実行)
フェーズ 1 が完了すると、トランザクション・マネージャーは、作業単位をコミットするという、取り消すことができない決定を行います。 トランザクション・マネージャーは、その作業単位に加わっている各リソース・マネージャーに対して、そのリソースの更新をコミットするように求めます。 リソース・マネージャーは、この要求を受け取ると、更新をコミットする必要があります。 この段階で更新をバックアウトすることはできません。 トランザクション・マネージャーが、そのリソースの更新をコミットしたという通知を各リソース・マネージャーから受け取ると、フェーズ 2 が完了します。

XA インターフェースは、2 フェーズ・コミット・プロトコルを使用します。

詳細については、トランザクション・サポートのシナリオを参照してください。

IBM MQ は、Microsoft Transaction Server (COM+)のサポートも提供します。Microsoft トランザクション・サーバーの使用 (COM+) は、 COM+ サポートの利点を利用するために IBM MQ をセットアップする方法に関する情報を提供します。

キュー・マネージャーの機能の拡張

ユーザー出口、API 出口、またはインストール可能サービスを使用して、キュー・マネージャーの機能を拡張できます。

ユーザー出口

ユーザー出口は、独自のコードをキュー・マネージャー機能に 挿入するメカニズムを提供します。 サポートされるユーザー出口は、次のとおりです。

チャネル出口
この出口では、チャネルの動作方法を変更できます。 チャネル出口については、 メッセージング・チャネルのためのチャネル出口プログラムで説明されています。

データ変換出口
この出口では、アプリケーション・プログラムに書き込んでデータの形式を変換するための部分ソース・コードを作成できます。 データ変換出口については、データ変換出口の作成で説明されています。

クラスター・ワークロード出口
この出口により実行できる機能は、出口のプロバイダーにより定義されます。 呼び出し定義の情報は、MQ_CLUSTER_WORKLOAD_EXIT - 呼び出しの説明にあります。

API 出口

API 出口を使用すると、MQPUT および MQGET などの IBM MQ API 呼び出しの動作を変更するコードを作成して、これらの呼び出しの直前または直後に、そのコードを挿入することができます。 挿入は自動的に行われます。キュー・マネージャーは登録されているポイントで出口コードを駆動します。 API 出口の詳細は、API 出口の使用/作成方法を参照してください。

インストール可能サービス

インストール可能サービスには、複数のエントリー・ポイントを持つ形式化されたインターフェース (API) があります。

インストール可能サービスは、サービス・コンポーネント と呼ばれるもので実現されます。 IBM MQ で提供されているコンポーネントを使用することも、必要な機能を実行するコンポーネントを独自に作成することもできます。

現在、次のようなインストール可能サービスが用意されています。

許可サービス
これにより、独自のセキュリティー機能を構築することができます。

このサービスを実装するデフォルト・サービス・コンポーネントは、オブジェクト権限マネージャー (OAM) です。 デフォルトでは、OAM はアクティブになっています。つまり、OAM を構成する ための作業は一切必要ありません。 許可サービス・インターフェースを 使用して別のコンポーネントを作成し、OAM を置き換 えたり OAM を補強したりできます。 OAM の詳細については、AIX, Linux, and Windows システムでのセキュリティーのセットアップを参照してください。

ネーム・サービス
これにより、アプリケーションはリモート・キューをローカル・キューであるかのように認識することができ、キューを共用できます。

所有するネーム・サービス・コンポーネントを書き込むことができます。 これは、例えば、IBM MQ でネーム・サービスを使用する予定の場合に行う必要が生じることがあります。 ネーム・サービスを使用するには、ユーザー作成によるコンポーネント、または別のソフトウェア・ベンダーによって提供されたコンポーネントのどちらかが必要です。 デフォルトでは、ネーム・サービスは使用できない状態になっています。

関連概念
ユーザー出口、API 出口、および IBM MQ インストール可能サービス

IBM MQ Java 言語インターフェース

IBM MQ には、 Java アプリケーションで使用するための 3 つのアプリケーション・プログラミング・インターフェース (API) (IBM MQ classes for Jakarta Messaging、 IBM MQ classes for JMS、および IBM MQ classes for Java) が用意されています。

IBM は、オープン・スタンダードをサポートしており、オープン・スタンダードに積極的に参加しています。

- IBM MQ 8.0 以降、この製品は JMS 2.0 標準を実装しています。これにより、新しい簡素化された API と、共有サブスクリプションなどの機能が導入されました。
- V 9.3.0 V 9.3.0 IBM MQ 9.3.0 以降、 Jakarta Messaging 3.0 もサポートされます。
- さらに、 WebSphere Liberty は IBM MQ で JMS 2.0 および Jakarta Messaging 3.0 をサポートします。

IBM MQ 内には、 Java アプリケーションで使用するための以下の 3 つの API があります。

JM 3.0 V 9.3.0 V 9.3.0 **IBM MQ classes for Jakarta Messaging**
IBM MQ classes for Jakarta Messaging は、 IBM MQ 用の Jakarta Messaging インターフェースをメッセージング・システムとして実装する Jakarta Messaging プロバイダーです。 Jakarta Connectors Architecture は、 Jakarta EE 環境で実行されるアプリケーションを IBM MQ や Db2 などのエンタープライズ情報システム (EIS) に接続する標準的な方法を提供します。

JMS 2.0 **IBM MQ classes for JMS**
IBM MQ classes for JMS は、 IBM MQ 用の JMS インターフェースをメッセージング・システムとして実装する JMS プロバイダーです。 Java Platform, Enterprise Edition Connector Architecture (JCA) は、Java EE 環境内で実行されているアプリケーションを、IBM MQ や Db2 などのエンタープライズ情報システム (EIS) に接続する標準的な方法を提供します。

IBM MQ classes for Java
IBM MQ classes for Java を使用すると、 Java 環境で IBM MQ を使用できます。 IBM MQ classes for Java では、Java アプリケーションは IBM MQ に IBM MQ クライアントとして接続するか、または IBM MQ キュー・マネージャーに直接接続することができます。

注 :

- V 9.3.0 V 9.3.0 JMS 2.0 は Jakarta Messaging に置き換えられました。 IBM MQ classes for JMS は引き続き JMS 2.0 標準をサポートしますが、 Java メッセージングに対する将来の機能拡張は Jakarta Messaging でのみ行われるため、 IBM MQ classes for Jakarta Messaging で行われます。 IBM MQ classes for JMS は、既存の JMS 2.0 アプリケーションを保守および拡張する場合にのみ推奨されま

す。 IBM MQ classes for Jakarta Messaging は、新しい開発に推奨されるテクノロジーでなければなりません。

- Stabilized IBM MQ classes for Java は、IBM MQ 8.0 で出荷されたレベルで機能的に固定化されています。 IBM MQ classes for Java を使用する既存のアプリケーションは引き続き完全にサポートされますが、この API は固定化されているため、新機能は追加されず、機能拡張の要求も拒否されます。 完全なサポートとは、欠陥が見つかった場合、IBM MQ システム要件の変更によって必要とされる変更と一緒に修正されることを意味します。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3 以降、IBM MQ classes for Java、IBM MQ classes for JMS、および IBM MQ classes for Jakarta Messaging は、 Java 8 を使用してビルドされています。 これらのインターフェースを使用してアプリケーションを実行するには、これらのレベル以上の Java ランタイム環境を使用する必要があります。

関連概念

Java から IBM MQ へのアクセス-API の選択

V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging を使用する理由

IBM MQ classes for JMS を使用する理由

IBM MQ classes for Java を使用する理由

IBM MQ classes for JMS/Jakarta Messaging

IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging は、IBM MQ で提供されるメッセージング・プロバイダーです。 これらの各プロバイダーは、メッセージング API に対する 2 セットの拡張機能も提供します。 これらのメッセージング・プロバイダーは、 Java Platform, Standard Edition (Java SE) アプリケーションと Java Platform, Enterprise Edition (Java EE) アプリケーションの両方で使用できます。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 では、 Jakarta Messaging 3.0 のサポートが導入されています。 JMS 2.0 は引き続き完全にサポートされます。

JMS および Jakarta Messaging 仕様は、アプリケーションがメッセージング操作を実行するために使用できるインターフェースのセットを定義します。 IBM MQ 8.0 以降、この製品は JMS 標準の JMS 2.0 バージョンをサポートします。 この実装はクラシック API のすべての機能を提供しますが、要求されるインターフェースが少なくなり、より簡単に使用できます。 詳しくは、 160 ページの『JMS および Jakarta Messaging モデル』 および Java.net の JMS 2.0 仕様を参照してください。 JM 3.0 IBM MQ 9.3.0 以降、 Jakarta Messaging もサポートされるようになりました。

jakarta.jms (Jakarta Messaging 3.0) または javax.jms (JMS 2.0) パッケージは、メッセージング・インターフェースの詳細を指定し、メッセージング・プロバイダーは特定のメッセージング製品用にこれらのインターフェースを実装します。 以下に例を示します。

- IBM MQ classes for JMS は、 IBM MQ 用の JMS インターフェースを実装する JMS プロバイダーであり、JMS API に対して以下の 2 セットの拡張機能も提供します。
  - IBM MQ JMS 拡張
  - IBM JMS 拡張
- javax.dims、 jakarta.jms、インターフェース、またはいずれかの JMS 拡張セットを使用して作成された接続ファクトリー、キュー、またはトピック・オブジェクトは、これらの API のいずれかを使用してアドレス指定できます。つまり、任意のインターフェースにキャストできます。 最高レベルでアプリケーションの移植性を維持するには、要件に適した最も汎用的な API を使用してください。

JMS と Jakarta Messaging は多くの共通点を共有しているため、このトピック内の JMS に対するこれ以降の参照は、両方の参照と見なすことができます。 必要に応じて、差異が強調表示されます。

IBM MQ JMS 拡張

IBM MQ classes for JMS は、JMS API に対する拡張機能も提供します。 IBM MQ classes for JMS には、MQConnectionFactory、MQQueue、および MQTopic の各オブジェクトで実装された拡張機能が含まれています。 これらのオブジェクトには IBM MQ に固有のプロパティーやメソッドがあります。 オブジェクト

は管理対象オブジェクトにすることができます。あるいは、アプリケーションはオブジェクトを実行時に動的に作成することができます。 これらの拡張機能は、 IBM MQ JMS 拡張機能と呼ばれます。 本書では、実行時にアプリケーションによって動的に作成されるオブジェクトは、管理対象オブジェクトとは見なされないことに注意してください。

IBM JMS 拡張

IBM MQ JMS 拡張機能に加えて、 IBM MQ classes for JMS には、使用されるプログラミング言語として、JMS API または Java に対する拡張機能のより汎用的なセットが用意されています。 これらの拡張機能は、IBM JMS 拡張機能と呼ばれ、以下のような幅広い目的があります。

- IBM JMS プロバイダー間でより高いレベルの整合性を提供するため。
- 2 つの IBM メッセージング・システム間のブリッジ・アプリケーションの作成を容易にする。
- ある IBM JMS プロバイダーから別のプロバイダーへのアプリケーションの移植を容易にするため。

それらの拡張機能は主に、接続ファクトリーおよび宛先を実行時に動的に作成および構成することに重点が置かれていますが、メッセージングと直接的には関係のない機能 (例えば問題判別のための機能) も提供します。

関連タスク

IBM MQ classes for JMS/Jakarta Messaging の使用

JMS および Jakarta Messaging リソースの構成

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging: 概要

IBM MQ 9.3.0 では、 Jakarta Messaging のサポートが導入されました。 Jakarta Messaging 3.0 の場合、JMS 仕様の制御が Oracle から Java Community Process に移動しました。 ただし、 Oracle は、他の Java テクノロジーで使用される「javax」名の制御を保持します。 したがって、 Jakarta Messaging 3.0 は JMS 2.0 と機能的には同等ですが、命名にはいくつかの違いがあります。 バージョン 3.0 の公式名は、 Java Message Service ではなく Jakarta Messaging であり、パッケージ名と定数名には、 `javax` ではなく `jakarta` という接頭部が付きます。

背景

長年にわたり、 Java プラットフォームには Standard Edition と Enterprise Edition の 2 つの形式があります。

Java Platform, Standard Edition (省略形は Java SE) は、スタンドアロン・コンテキストで実行できるコア言語およびクラス・ライブラリーです。 Java SE のほとんどの Java パッケージには、「java.」で始まる名前が付いています。

Java Platform, Enterprise Edition (Java EE) はこれを拡張し、メッセージング、各種 Bean、トランザクションなどの機能を追加します。 これらのテクノロジーの一部は、 Java SE コンテキストで使用することもできます。 Java EE のほとんどの Java パッケージは、歴史的に「javax」で始まる名前を持っています。ただし、クロスオーバーがいくつかあるため、一部の Java SE パッケージには「javax」があります。 名前の接頭部として使用されます。

Java Message Service (JMS) は、Java Platform, Enterprise Edition の一部です。 Java EE 7 には JMS 2.0 が組み込まれています。

Java EE 7 までは、テクノロジーは Oracle の管理下にありました。

Java EE テクノロジーは、最近、 Oracle のスチュワードシップから、 Eclipse Foundation が監督するコミュニティー・プロセスに移行しました。

"javax" として 名前を新しいプロジェクトに移動できませんでした。新しい名前が採用されました。すべてのパッケージ名とプロパティー名に「jakJakarta」という接頭部が付きます。 将来、 Java Platform, Enterprise Edition は「Jakarta EE」と呼ばれるようになります。 バージョン 8 は概して無視できる暫定バージョンであり、 Jakarta EE 9 は "jakJakarta" が存在するポイントである。 接頭部が有効になります。

IBM MQ コンテキストに適用される主な Jakarta EE テクノロジーは、 Jakarta Messaging 3.0 - Java Message Service (JMS) 2.0 の後継です。 そのため、 Jakarta EE 9 には Jakarta Messaging 3.0 が組み込まれています。

IBM MQ は、 Jakarta EE 9 および Jakarta Messaging 3.0 のサポートを導入する一方で、 Java EE 7 および JMS 2.0 を引き続きサポートします。

提供内容: Java SE

Java Platform, Standard Edition の場合、 IBM MQ classes for JMS (IBM MQ での JMS 2.0 操作をサポートします) に加えて、 IBM MQ 9.3.0 は IBM MQ classes for Jakarta Messaging を提供します。 これらのクラスは、IBM MQ と統合する Jakarta Messaging 3.0 プロバイダーを提供します。これにより、IBM MQ キュー・マネージャーを使用して Jakarta Messaging 操作を容易にすることができます。

これらは、 IBM MQ インストール済み環境の `java/lib` サブディレクトリーに、標準 JAR ファイル `com.ibm.mq.jakarta.client.jar` として提供されています。

Apache Felix や Eclipse Equinox などの OSGi コンテナーで使用するために、 IBM MQ には OSGi バンドルのペアも用意されています。

- `com.ibm.mq.osgi.jms30.clientprereqs_`*`V.R.M.F`*`.jar`
- `com.ibm.mq.osgi.jms30.client_`*`V.R.M.F`*`.jar`

ここで、*V.R.M.F* は、IBM MQ のバージョンを表します (例: 9.3.0.0)。 これらのバンドルは、IBM MQ インストール済み環境の `java/lib/OSGi` サブディレクトリーにあります。

提供内容: Jakarta EE 9

Jakarta EE 9 互換アプリケーション・サーバーで IBM MQ ベースのメッセージングをサポートするために、IBM MQ は Jakarta EE 9 互換リソース・アダプター `wmq.jakarta.jmsra.rar` を提供しています。 これは、 IBM MQ インストール済み環境の `java/lib/jca` サブディレクトリーにあります。

IBM MQ は引き続き、 IBM MQ インストール済み環境の `java/lib/jca` サブディレクトリーに Java EE 7 互換リソース・アダプター `wmq.jmsra.rar` を提供します。

これらの成果物の提出方法

リソース・アダプター用のこれらの JAR および RAR ファイルは、通常の IBM MQ インストール・メディア (「`.rpm`」ファイルなどのプラットフォーム固有のインストール・メディアと、自己解凍型再配布可能クライアント JAR ファイルなどの再配布可能メディアの両方) に既存の成果物とともにパッケージされています。

JMS 2.0 と Jakarta Messaging 3.0 の間の変更点

Jakarta EE 9 および Jakarta Messaging 3.0 では、新機能は導入されていません。 変更されるのは名前のみです。 例えば、 JMS 2.0 で「javax.jms.Connection」を使用する場合は、 Jakarta Messaging 3.0 で「jakarta.jms.Connection」を使用します。

Eclipse Foundation は Jakarta EE プラットフォームを採用するため、このファウンデーションに基づいて構築されます。この命名規則は、将来導入される新機能に使用されます。

IBM MQ classes for JMS と IBM MQ classes for Jakarta Messaging の間の変更点

要約

JMS 2.0 のサポートを提供する IBM MQ classes for JMS は引き続き使用可能であり、主に既存のアプリケーションを保守および拡張するために推奨されます。 これらは完全にサポートされています。

Jakarta Messaging 3.0 のサポートを提供する IBM MQ classes for Jakarta Messaging は、新規開発に推奨されます。

IBM MQ 9.3.0 では、これら 2 つのオファリングは機能的に同等です。 命名のみが異なります。 ただし、 IBM MQ classes for Jakarta Messaging では、 IBM MQ classes for JMS よりも新しいメッセージング機能が出現する可能性が高くなります。

この 2 つのオファリングは相互運用可能です。 IBM MQ classes for JMS によって生成されたメッセージは、 IBM MQ classes for Jakarta Messaging によってコンシュームされる可能性があります。また、その逆も同様です。 ただし、2 つのオファリングが単一のアプリケーション内で共存することはできません。

名前の変更

表 *16.* パッケージ名の変更

IBM MQ classes for JMS パッケージ名	**IBM MQ classes for Jakarta Messaging パッケージ名**
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.*
com.ibm.msg.client.wmq.*	com.ibm.msg.client.jakarta.wmq.*

共通サービス (トレース、ロギング、各国語サポートなど) および JMQI 実装 (ローカルおよびリモート) に関連するパッケージは、IBM MQ classes for JMS と IBM MQ classes for Jakarta Messaging の両方に共通しているため、これらの領域での変更は必要ありません。

プロパティー名も変更されていることに注意してください。 例えば、 IBM MQ classes for Jakarta Messaging で IBM MQ 拡張機能を有効にするプロパティーは、 **com.ibm.mq.jakarta.jms.SupportMQExtensions** です。

IBM MQ classes for JMS または IBM MQ classes for Jakarta Messaging に依存しないプロパティー名 (さまざまな **com.ibm.msg.client.commonservices.trace.*** プロパティーなど) は、両方のオファリングに等しく適用されます。

管理ユーティリティー

crtmqenv および **setmqenv** ユーティリティーは、クラスパスを IBM MQ classes for JMS (`-j 2.0`) 用に構成するか IBM MQ classes for Jakarta Messaging (`-j 3.0`) 用に構成するかを指定するオプションを受け入れるようになりました。また、 **runjms30** と呼ばれる **runjms** ユーティリティーの IBM MQ classes for Jakarta Messaging バリアントおよび類似の名前があります。

dspmqver ユーティリティーは、 Java コンポーネントについて報告するよう要求された場合、その出力に IBM MQ classes for Jakarta Messaging を組み込みます。

JNDI を介して取得されるように IBM MQ classes for Jakarta Messaging オブジェクトを構成する場合、新しい **JMS30Admin** ユーティリティーは IBM MQ classes for JMS 用の **JMSAdmin** ユーティリティーと同等です。

基礎となるオブジェクトは異なるパッケージからのものであることに注意してください。 **JMSAdmin** によって作成された JNDI 定義を IBM MQ classes for Jakarta Messaging で使用することはできません。また、 **JMS30Admin** によって作成された JNDI 定義を IBM MQ classes for JMS で使用することもできません。

注 : IBM MQ Explorer によって提供される IBM MQ classes for Jakarta Messaging オブジェクトはサポートされません。その JNDI 統合は IBM MQ classes for JMS 専用です。

関連概念

IBM MQ classes for Jakarta Messaging を使用する理由

JMS および Jakarta Messaging モデル

JMS および Jakarta Messaging モデルは、 Java アプリケーションがメッセージング操作を実行するために使用できるインターフェースのセットを定義します。 IBM MQ classes for JMS および IBM MQ classes for

Jakarta Messaging は、 Java メッセージング・オブジェクトが IBM MQ の概念にどのように関連するかを定義するメッセージング・プロバイダーです。 JMS および Jakarta Messaging 仕様は、特定のメッセージング・オブジェクトが管理対象オブジェクトであることを想定しています。

IBM MQ 8.0 以降、製品は JMS 標準の JMS 2.0 バージョンをサポートします。これにより、 JMS 1.1 のクラシック API も保持しながら、簡素化された API が導入されました。

IBM MQ 9.3.0 では、 Jakarta Messaging 3.0 のサポートが導入されています。 JMS 2.0 は引き続き完全にサポートされます。 JMS と Jakarta Messaging は多くの共通点を共有しているため、このトピック内の JMS に対するこれ以降の参照は、両方の参照と見なすことができます。 必要に応じて、差異が強調表示されます。

簡易 API

JMS 2.0 では、単純化された API が導入されました。また、 JMS 1.1 からのドメイン固有およびドメイン独立のインターフェースも保持されます。 簡易 API では、メッセージの送受信に必要なオブジェクトの数が減り、次のインターフェースで構成されます。

ConnectionFactory
ConnectionFactory は、接続を作成するために JMS クライアントが使用する管理オブジェクトです。このインターフェースはクラシック API でも使用されます。

JMSContext
このオブジェクトは、クラシック API の接続オブジェクトとセッション・オブジェクトを結合します。JMSContext オブジェクトは、他の JMSContext オブジェクトから作成でき、基礎接続が複製されます。

JMS プロデューサー
JMSProducer は、JMSContext によって作成され、キューまたはトピックにメッセージを送信するために使用されます。 JMSProducer オブジェクトによって、メッセージの送信に必要とされるオブジェクトの作成が発生します。

JMS コンシューマー
JMSConsumer は、JMSContext によって作成され、トピックまたはキューからのメッセージの受信に使用されます。

簡易 API には、様々な効果があります。

- JMSContext オブジェクトは、常に自動的に基礎接続を開始します。
- JMSProducer と JMSConsumer は、メッセージ・オブジェクト全体を取得しなくても、メッセージの getBody メソッドを使用してメッセージ本文を直接処理できるようになりました。
- メッセージのプロパティーは、「本文」、つまりメッセージのコンテンツを送信する前に、メソッド・チェーニングを使用して JMSProducer オブジェクトに設定できます。 JMSProducer は、メッセージの送信に必要なすべてのオブジェクトの作成を処理します。 JMS 2.0 を使用して、次のようにプロパティーを設定して、メッセージを送信できます。

```
context.createProducer().
setProperty("foo", "bar").
setTimeToLive(10000).
setDeliveryMode(NON_PERSISTENT).
setDisableMessageTimestamp(true).
send(dataQueue, body);
```

JMS 2.0 では、メッセージを複数のコンシューマー間で共有できる共有サブスクリプションも導入されました。 すべての JMS 1.1 のサブスクリプションは、非共有サブスクリプションとして処理されます。

クラシック API

次のリストは、クラシック API の主な JMS インターフェースを要約しています。

Destination
Destination は、アプリケーションがメッセージを送信する場所、またはアプリケーションが受信するメッセージの送信元、あるいはその両方です。

ConnectionFactory
ConnectionFactory オブジェクトは、接続の構成プロパティーのセットをカプセル化します。 アプリケーションは、接続ファクトリーを使用して接続を作成します。

接続
Connection オブジェクトは、メッセージング・サーバーに対するアプリケーションのアクティブな接続をカプセル化します。 アプリケーションは、接続を使用してセッションを作成します。

Session
Session は、メッセージを送受信する単一スレッド化されたコンテキストです。 アプリケーションは、Session を使用してメッセージ、メッセージ・プロデューサー、およびメッセージ・コンシューマーを作成します。 セッションは、トランザクション化しても、トランザクション化しなくてもかまいません。

メッセージ
Message オブジェクトは、アプリケーションが送信または受信するメッセージをカプセル化します。

MessageProducer
アプリケーションがメッセージ・プロデューサーを使用して宛先にメッセージを送信します。

MessageConsumer
アプリケーションがメッセージ・コンシューマーを使用して宛先に送信されたメッセージを受信します。

162 ページの図 53 は、これらのオブジェクトとその関係を示します。

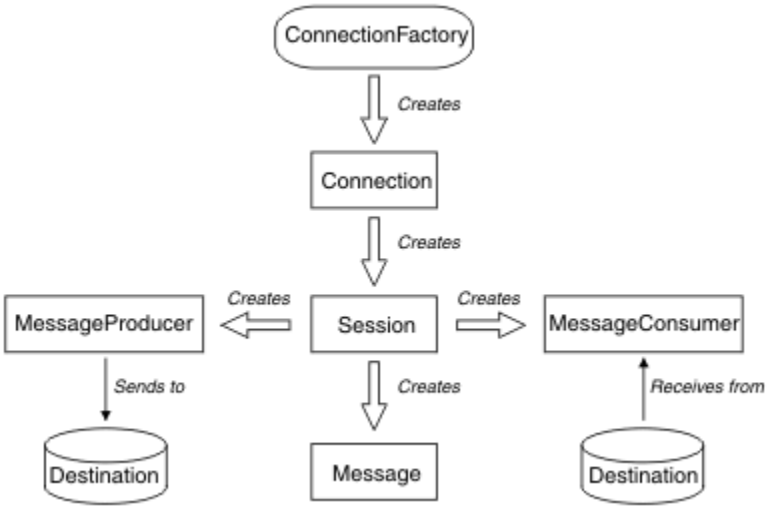


図 *53. JMS* オブジェクトとその関係

図には、メイン・インターフェースの ConnectionFactory、Connection、Session、MessageProducer、MessageConsumer、Message、および Destination が示されています。 アプリケーションは、接続ファクトリーを使用して接続を作成し、接続を使用してセッションを作成します。 アプリケーションは、次にセッションを使用してメッセージ、メッセージ・プロデューサー、およびメッセージ・コンシューマーを作成します。 アプリケーションはメッセージ・プロデューサーを使用してメッセージを宛先に送信し、メッセージ・コンシューマーを使用して宛先に送信されたメッセージを受信します。

Destination、ConnectionFactory、または Connection オブジェクトは、マルチスレッド・アプリケーションの異なるスレッドによって並行して使用できますが、Session、MessageProducer、または MessageConsumer オブジェクトは、異なるスレッドによって並行して使用することはできません。 Session、MessageProducer、または MessageConsumer オブジェクトが並行して使用されないようにする最も簡単な方法は、スレッドごとに別の Session オブジェクトを作成することです。

JMS は、2 つのスタイルのメッセージングをサポートします。

- Point-to-Point メッセージング
- パブリッシュ/サブスクライブ・メッセージング

メッセージングのこれらのスタイルは、メッセージ・ドメイン とも呼ばれ、1 つのアプリケーションで両方のスタイルのメッセージングを結合することができます。 Point-to-Point ドメインの場合、宛先はキューであり、パブリッシュ/サブスクライブ・ドメインの場合、宛先はトピックです。

JMS 1.1 の以前のバージョンの JMS では、Point-to-Point ドメインのプログラミングはインターフェースとメソッドの 1 つのセット使用され、パブリッシュ/サブスクライブ・ドメインのプログラミングは別のセットを使用します。 2 つのセットは似ていますが、別のものです。 JMS 1.1 以降、両方のメッセージ・ドメインをサポートするインターフェースとメソッドの共通のセットを使用できます。 共通のインターフェースは、メッセージ・ドメインごとにドメイン非依存のビューを提供します。 163 ページの表 17 に、JMS のドメイン非依存インターフェースと対応するドメイン固有インターフェースをリストします。

表 *17. JMS* のドメイン非依存インターフェースとその対応するドメイン固有インターフェース

ドメイン非依存インターフェース	Point-to-Point ドメインのドメイン固有インターフェース	パブリッシュ/サブスクライブ・ドメインのドメイン固有インターフェース
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
接続	QueueConnection	TopicConnection
Destination	キュー	トピック
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

JMS 2.0 IBM MQ classes for JMS 2.0 は、以前の JMS 1.1 ドメイン固有インターフェースと、JMS 2.0 の単純化された API の両方をサポートします。 そのため、 IBM MQ classes for JMS 2.0 を使用して、既存のアプリケーションを保守することができます。これには、既存のアプリケーションでの新機能の開発も含まれます。

JM 3.0 IBM MQ classes for Jakarta Messaging 3.0 は、同じインターフェースの Jakarta Messaging バージョンをサポートしており、新しいアプリケーション開発に推奨されます。

IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging では、 JMS オブジェクトは、以下の方法で IBM MQ の概念に関連しています。

- Connection オブジェクトには、接続の作成に使用された接続ファクトリーのプロパティーから派生したプロパティーがあります。 これらのプロパティーは、アプリケーションがキュー・マネージャーに接続する方法を制御します。 これらのプロパティーの例はキュー・マネージャーの名前であり、クライアント・モードのキュー・マネージャーに接続するアプリケーションでは、キュー・マネージャーが動作しているシステムのホスト名または IP アドレスです。
- Session オブジェクトは、IBM MQ 接続ハンドルをカプセル化するため、セッションのトランザクションの範囲を定義します。
- MessageProducer オブジェクトと MessageConsumer オブジェクトは、それぞれ IBM MQ オブジェクト・ハンドルをカプセル化します。

IBM MQ classes for JMS または IBM MQ classes for Jakarta Messaging を使用する場合は、IBM MQ の通常の規則がすべて適用されます。 特に、アプリケーションはリモート・キューにメッセージを送信できますが、アプリケーションが接続しているキュー・マネージャーによって所有されるキューからしかメッセージを受信できないことに注意してください。

JMS 仕様は、ConnectionFactory オブジェクトと Destination オブジェクトが管理オブジェクトであると想定します。 管理者は管理オブジェクトを中央リポジトリーで作成して保守し、JMS アプリケーションは Java Naming and Directory Interface (JNDI) を使用してこれらのオブジェクトを取得します。

IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging では、Destination インターフェースの実装は Queue および Topic の抽象スーパークラスであるため、Destination のインスタンスは Queue

オブジェクトまたは Topic オブジェクトのいずれかになります。 ドメイン非依存インターフェースは、キューまたはトピックを宛先として処理します。 MessageProducer オブジェクトまたは MessageConsumer オブジェクトのメッセージ・ドメインは、宛先がキューまたはトピックのいずれであるかによって決定されます。

したがって、 IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging では、以下のタイプのオブジェクトを管理対象オブジェクトにすることができます。

- ConnectionFactory
- QueueConnectionFactory
- TopicConnectionFactory
- キュー
- トピック
- XAConnectionFactory
- XAQueueConnectionFactory
- XATopicConnectionFactory

IBM MQ classes for JMS/Jakarta Messaging アーキテクチャー

IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging には、階層化アーキテクチャーがあります。 コードの最上位レイヤーは、すべての IBM Java メッセージング・プロバイダーが使用できる共通のレイヤーです。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 では、 Jakarta Messaging 3.0 のサポートが導入されています。 JMS 2.0 は引き続き完全にサポートされます。

IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging には、図 165 ページの図 54 に示す階層化アーキテクチャーがあります。 コードの最上位のレイヤーは、任意の IBM JMS または Jakarta Messaging プロバイダーが使用できる共通レイヤーです。 アプリケーションが JMS メソッドまたは Jakarta Messaging メソッドを呼び出すと、メッセージング・システムに固有でない呼び出しの処理は共通レイヤーによって実行されます。共通レイヤーは、呼び出しに対する一貫性のある応答も提供します。 メッセージング・システムに固有の呼び出しの処理は、より下の層に委任されます。 次の図で、IBM MQ メッセージング・プロバイダーが、他の 2 つのメッセージング・プロバイダー (メッセージング・プロバイダー A とメッセージング・プロバイダー B) と共に、下の層に示されています。

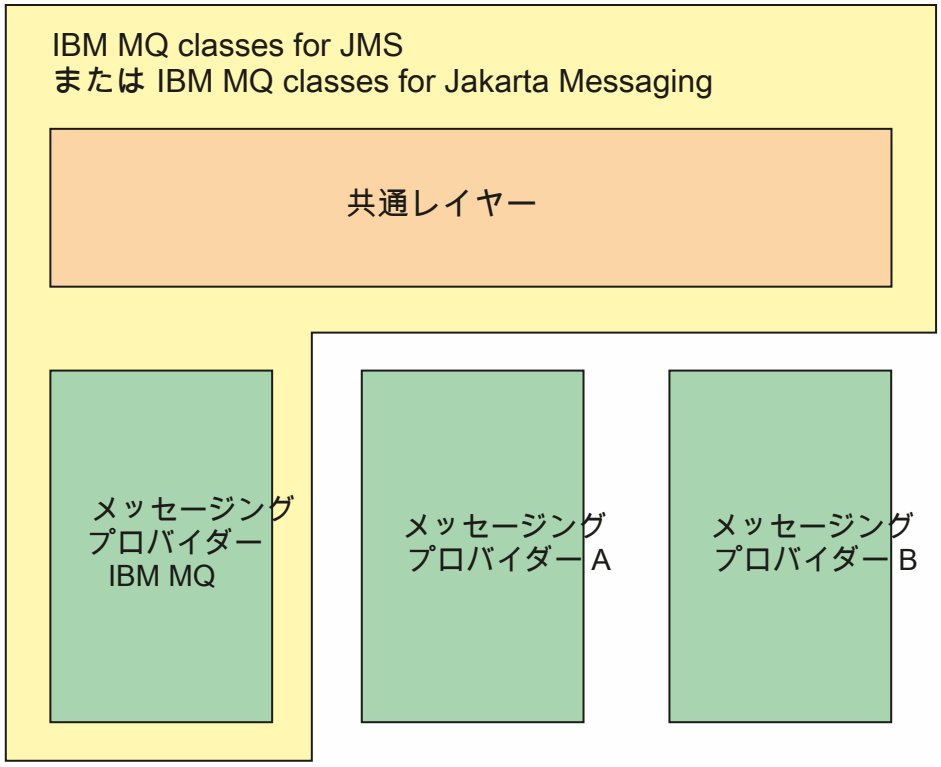


図 *54. IBM JMS および Jakarta Messaging プロバイダーの階層化アーキテクチャー*

階層化アーキテクチャーは、次の目的を達成します。

- さまざまな IBM JMS および Jakarta Messaging プロバイダーの動作の整合性を向上させる
- 2 つの IBM メッセージング・システムの間のブリッジ・アプリケーションの作成を簡単にする
- ある IBM JMS または Jakarta Messaging プロバイダーから別のプロバイダーへのアプリケーションの移植を容易にする

関連タスク

IBM MQ classes for JMS/Jakarta Messaging の使用

管理オブジェクトのサポート

IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging は、管理対象オブジェクトの使用をサポートします。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 以降、 Jakarta Messaging 3.0 は新規アプリケーションの開発用にサポートされています。 IBM MQ 9.3.0 は、既存のアプリケーションに対して JMS 2.0 を引き続きサポートします。 同じアプリケーションで Jakarta Messaging 3.0 API と JMS 2.0 API の両方を使用することはサポートされていません。 詳しくは、 Using IBM MQ classes for JMS/Jakarta Messaging を参照してください。

JMS または IBM MQ classes for Jakarta Messaging アプリケーション内のロジックのフローは、ConnectionFactory オブジェクトと Destination オブジェクトで始まります。 アプリケーションは ConnectionFactory オブジェクトを使用して Connection オブジェクトを作成します。このオブジェクトは、メッセージング・サーバーへのアプリケーションのアクティブな接続を表します。 アプリケーションは Connection オブジェクトを使用して Session オブジェクトを作成します。このオブジェクトは、メッセージを作成およびコンシュームするための単一スレッド・コンテキストです。 さらに、アプリケーションは Session オブジェクトおよび Destination オブジェクトを使用して MessageProducer オブジェクトを作成できます。アプリケーションはこのオブジェクトを使用して、メッセージを指定された宛先に送信します。 宛先はメッセージング・システム内のキューまたはトピックのいずれかであり、Destination オブジェ

クトによってカプセル化されます。 また、アプリケーションは Session オブジェクトおよび Destination オブジェクトを使用して MessageConsumer オブジェクトを作成できます。アプリケーションはこのオブジェクトを使用して、指定された宛先に送信されているメッセージを受信します。

JMS 仕様および Jakarta Messaging 仕様では、 ConnectionFactory オブジェクトと Destination オブジェクトが管理対象オブジェクトであることが想定されています。 管理者は、中央リポジトリーに管理対象オブジェクトを作成して保守します。 JMS または Jakarta Messaging アプリケーションは、 Java Naming Directory Interface (JNDI) を使用してこれらのオブジェクトを取得します。 管理対象オブジェクトのリポジトリーは、単純なファイルから Lightweight Directory Access Protocol (LDAP) ディレクトリーまでの範囲にすることができます。

IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging は、管理対象オブジェクトの使用をサポートします。 アプリケーションは、 IBM MQ 固有の情報をアプリケーション自体にハードコーディングすることなく、IBM MQ を介して公開される IBM MQ classes for JMS または IBM MQ classes for Jakarta Messaging のすべての機能を使用できます。 この方法で、アプリケーションは基盤となる IBM MQ の構成からある程度独立できます。

この独立性を実現するために、アプリケーションは JNDI を使用して管理対象オブジェクトとして保管されている接続ファクトリーおよび宛先を取得し、 `javax.jms` (JMS 2.0) または `jakarta.jms` (Jakarta Messaging 3.0) パッケージで定義されているインターフェースのみを使用してメッセージング操作を実行することができます。

JMS 2.0 JMS 2.0 の場合、管理者は IBM MQ JMS 管理ツール **JMSAdmin** または IBM MQ Explorer を使用して、中央リポジトリーで管理対象オブジェクトを作成および保守することができます。

JM 3.0 Jakarta Messaging 3.0 の場合、 IBM MQ Explorer を使用して JNDI を管理することはできません。 JNDI 管理は、 **JMSAdmin** の Jakarta Messaging 3.0 バリアント (**JMS30Admin**) によってサポートされます。

アプリケーション・サーバーは通常、管理対象オブジェクト用の独自のリポジトリーと、オブジェクトを作成および保守するための独自のツールを提供します。 したがって、Java EE JM 3.0 または Jakarta EE アプリケーションは、 JNDI を使用して、アプリケーション・サーバー・リポジトリーまたは中央リポジトリーのいずれかから管理対象オブジェクトを取り出すことができます。

関連タスク
JMS および Jakarta Messaging リソースの構成

Java EE および Jakarta EE プラットフォームでサポートされる通信タイプ

Java EE および Jakarta EE プラットフォームでは、 IBM MQ classes for JMS および IBM MQ classes for Jakarta Messaging は、アプリケーションのコンポーネントと IBM MQ キュー・マネージャーの間の 2 つのタイプの通信をサポートします。

JM 3.0 V 9.3.0 V 9.3.0 IBM MQ 9.3.0 では、 Jakarta Messaging 3.0 のサポートが導入されています。 JMS 2.0 は引き続き完全にサポートされます。 JMS と Jakarta Messaging は多くの共通点を共有しているため、このトピック内の JMS に対するこれ以降の参照は、両方の参照と見なすことができます。 必要に応じて、差異が強調表示されます。

アプリケーションのコンポーネントと IBM MQ キュー・マネージャーの間の次の 2 つのタイプの通信がサポートされます。

- アウトバウンド通信
- インバウンド通信

アウトバウンド通信

JMS API または Jakarta Messaging API を直接使用して、アプリケーション・コンポーネントはキュー・マネージャーへの接続を作成し、メッセージを送受信します。

例えば、アプリケーション・コンポーネントは、アプリケーション・クライアント、サーブレット、JavaServer Page (JSP)、エンタープライズ Java Bean (EJB)、またはメッセージ駆動型 Bean (MDB) である

ことが可能です。 このタイプの通信では、アプリケーション・サーバー・コンテナーは、接続のプールやスレッド管理など、メッセージング操作をサポートする低レベルの機能のみを提供します。

インバウンド通信

インバウンド通信の場合、宛先に到達するメッセージは MDB に配信されてから、そこでメッセージが処理されます。

Java EE JM 3.0 および Jakarta EE アプリケーションは、MDB を使用してメッセージを非同期に処理します。 MDB は JMS のメッセージ・リスナーとして機能し、メッセージの処理方法を定義する onMessage() メソッドによって実装されます。 MDB は、アプリケーション・サーバーの EJB コンテナーに実装されます。 MDB が構成される正確な方法は、使用しているアプリケーション・サーバーに依存しますが、構成情報で、接続先のキュー・マネージャー、キュー・マネージャーへの接続方法、メッセージをモニターする宛先、および MDB のトランザクションの動作を指定する必要があります。 この情報は EJB コンテナーによって使用されます。 MDB の選択基準を満たすメッセージが指定された宛先に到着すると、EJB コンテナーは IBM MQ classes for JMS または IBM MQ classes for Jakarta Messaging を使用してキュー・マネージャーからメッセージを取得し、onMessage() メソッドを呼び出して MDB にメッセージを配信します。

IBM MQ classes for Java との関係

IBM MQ classes for Java、 IBM MQ classes for Jakarta Messaging 、および IBM MQ classes for JMS は、MQI への共通 Java インターフェースを使用するピアです。

167 ページの図 55 は、 IBM MQ classes for JMS、 IBM MQ classes for Jakarta Messaging 、および IBM MQ classes for Java の間の関係を示しています。

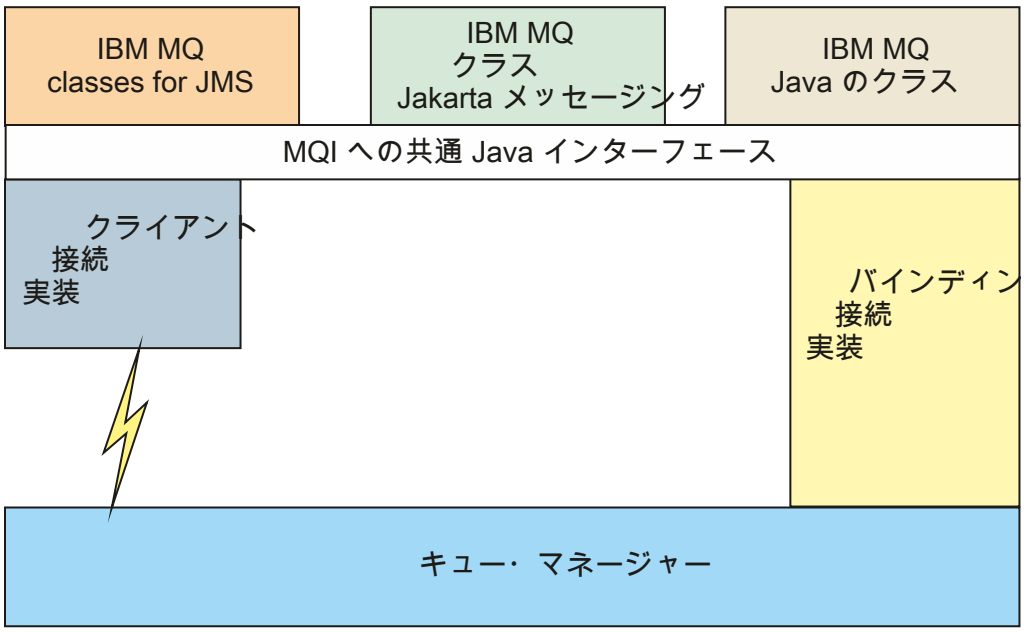


図 *55. IBM MQ classes for JMS*、 *IBM MQ classes for Jakarta Messaging* 、および *IBM MQ classes for Java* の間の関係

一般に、 Java プログラムは、 IBM MQ - IBM MQ classes for Java、 IBM MQ classes for Jakarta Messaging 、または IBM MQ classes for JMS とのインターフェースとして 1 つのインターフェースのみを使用する必要があります。 インターフェースの混合はサポートされていませんが、1 つの例外があります。 IBM WebSphere MQ 7.0 より前のリリースとの互換性を維持するために、Java で作成されたチャネル出口クラスは、チャネル出口クラスが IBM MQ classes for JMS から呼び出される場合でも、 IBM MQ classes for Java インターフェースを使用できます。 ただし、 IBM MQ classes for Java インターフェースを使用することは、アプリケーションが引き続き以下のいずれかに依存していることを意味します。

- JMS 2.0 IBM MQ classes for Java JAR ファイル、 `com.ibm.mq.jar`。 クラス・パスに `com.ibm.mq.jar` が不要な場合は、代わりに `com.ibm.mq.exits` パッケージのインターフェースのセットを使用できます。
- JM 3.0 V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging と相互運用する場合の `com.ibm.mq.jakarta.client.jar` の使用。

関連概念

V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging を使用する理由

JMS 用に IBM MQ クラスを使用する理由

IBM MQ classes for Java を使用する理由

IBM MQ メッセージング・プロバイダー

IBM MQ メッセージング・プロバイダーには、通常モード、制限付き通常モード、マイグレーション・モードという 3 つの操作モードがあります。

IBM MQ メッセージング・プロバイダーには、3 つの操作モードがあります。

- IBM MQ メッセージング・プロバイダーの通常モード
- IBM MQ メッセージング・プロバイダーの制限付き通常モード
- IBM MQ メッセージング・プロバイダーのマイグレーション・モード

IBM MQ メッセージング・プロバイダーの通常モードでは、IBM MQ キュー・マネージャーのすべての機能を使用して JMS が実装されます。 このモードは、 JMS 2.0 JM 3.0 V 9.3.0 V 9.3.0 または Jakarta Messaging 3.0 の API と機能を使用するように最適化されています。

次の場合

- クライアントは、 **ConnectionFactory** 上でプロバイダー・バージョン 6 を指定します。クライアントは、 IBM WebSphere MQ 6.0 で提供されるクライアントと互換性のある方法で動作します。 JMS 1.1 および JMS 2 インターフェースのみがサポートされますが、一部の JMS 2 機能 (共有サブスクリプション、送達遅延、非同期送信など) は使用不可になります。 接続の共有はありません。
- クライアントは、 **ConnectionFactory** 上でプロバイダー・バージョン 7 を指定します。JMS 1.1 インターフェースと JMS 2 インターフェースの両方が完全にサポートされます。
- プロバイダー・バージョンが指定されていません。プロバイダー・バージョン 7 との接続が試行されます。 これが失敗すると、プロバイダー・バージョン 6 でさらに試行が行われます。

IBM MQ Enterprise Transport を使用して IBM Integration Bus に接続する場合は、マイグレーション・モードを使用します。 IBM MQ Real-Time Transport を使用する場合は、接続ファクトリー・オブジェクトでプロパティーを明示的に選択しているため、マイグレーション・モードが自動的に選択されます。 IBM MQ Enterprise Transport を使用した IBM Integration Bus への接続は、JMS **PROVIDERVERSION** プロパティーの構成で説明されているモード選択の一般規則に従います。

関連タスク

JMS リソースの構成

z/OS IBM MQ for z/OS の概念

IBM MQ for z/OS が使用する概念の一部は、z/OS プラットフォームに固有のものです。 例えば、ロギング・メカニズム、ストレージ管理技法、リカバリー単位属性指定、およびキュー共用グループは、IBM MQ for z/OS でのみ提供されます。 このトピックでは、これらの概念について詳しく説明します。

概念には、以下のような、 IBM MQ for z/OS が使用するオブジェクトの概要が含まれます。

- キュー・マネージャー
- チャネル・イニシエーター
- 共用キューとキュー共用グループ

- グループ内キューイング

以下のトピックでは、以下のような、必要なさまざまな手順についても説明します。

- z/OS でのシステム定義
- ストレージ管理
- 回復と再始動
- IBM MQ for z/OS におけるセキュリティーの概念

関連概念

170 ページの『z/OS 上のキュー・マネージャー』
アプリケーション・プログラムで z/OS システム上の IBM MQ を使用できるようにするには、その前に IBM MQ for z/OS 製品をインストールし、キュー・マネージャーを開始する必要があります。 キュー・マネージャーは、IBM MQ によって使用される複数の資源を所有し、管理します。

171 ページの『z/OS 上のチャネル・イニシエーター』
チャネル・イニシエーターは、IBM MQ 分散キューイングを使用可能にする資源を提供し、管理します。 IBM MQ は、メッセージ・チャネル・エージェント (MCA) を使用して、あるキュー・マネージャーから別のキュー・マネージャーにメッセージを送信します。

172 ページの『IBM MQ for z/OS を管理するための用語とタスク』
このトピックでは、IBM MQ for z/OS に固有の用語と作業の概要について説明します。

175 ページの『共用キューとキュー共用グループ』
共用キューおよびキュー共用グループを使用すると、IBM MQ リソースの高可用性を実装することができます。 共有キューおよびキュー共有グループは、IBM MQ for z/OS プラットフォーム上の z/OS に固有な機能です。

221 ページの『グループ内キューイング』
このセクションでは、 z/OS プラットフォームに固有の IBM MQ for z/OS 機能である、グループ内キューイングについて説明します。 この機能を使用できるのは、キュー共用グループに対して定義されているキュー・マネージャーだけです。

234 ページの『z/OS でのストレージ管理』
IBM MQ for z/OS は、永続的および一時的なデータ構造を必要としており、ページ・セットおよびメモリー・バッファーを使用してこのデータを保管します。 これらのトピックでは、IBM MQ がこれらのページ・セットおよびバッファーを使用する方法について詳しく説明します。

239 ページの『IBM MQ for z/OS でのロギング』
IBM MQ は、データの変更と重要なイベントが発生した場合に、そのログ を維持します。 これらのログは、必要に応じてデータを以前の状態にリカバリーするために使用できます。

261 ページの『z/OS での回復と再始動』
このトピックのリンクを使用して、IBM MQ for z/OS の再始動および回復の機能について参照してください。

278 ページの『IBM MQ for z/OS におけるセキュリティーの概念』
このトピックを使用して、IBM MQ のセキュリティーの重要性、システムに十分のセキュリティー設定がなされていない場合の影響について理解してください。

284 ページの『z/OS での可用性』
IBM MQ for z/OS には、高可用性のための多数の機能があります。 このトピックでは、可用性に関するいくつかの考慮事項について説明します。

289 ページの『z/OS での回復単位後処理』
トランザクション・アプリケーションの中には、キュー共用グループ (QSG) 内のキュー・マネージャーに接続されているときに、QMGR ではなく、GROUP 回復単位後処理を使用できるものがあります。その場合、接続時にキュー・マネージャー名の代わりに QSG 名を指定します。 これは QSG 内の同じキュー・マネージャーに再接続する必要がないので、トランザクションの回復をより柔軟かつ堅固に行えます。

関連資料

250 ページの『z/OS でのシステム定義』

IBM MQ for z/OS は、多くのデフォルトのオブジェクト定義を使用しており、それらのデフォルトのオブジェクトを作成するためのサンプル JCL を提供しています。 このトピックでは、これらのデフォルトのオブジェクトおよびサンプル JCL について理解することができます。

288 ページの『IBM MQ for z/OS でのモニターと統計』
IBM MQ for z/OS には、キュー・マネージャーをモニターして、統計を収集するための一連の機能があります。

z/OS 上のキュー・マネージャー

アプリケーション・プログラムで z/OS システム上の IBM MQ を使用できるようにするには、その前に IBM MQ for z/OS 製品をインストールし、キュー・マネージャーを開始する必要があります。 キュー・マネージャーは、IBM MQ によって使用される複数の資源を所有し、管理します。

キュー・マネージャー

キュー・マネージャー は、アプリケーションに対してメッセージング・サービスを提供するプログラムです。 Message Queue Interface (MQI) を使用するアプリケーションは、キューにメッセージを書き込んだり、キューからメッセージを読み取ったりすることができます。 キュー・マネージャーは、メッセージが必ず正しいキューに送信されるか、または別のキュー・マネージャーに経路指定されるようにします。 キュー・マネージャーは、それに対して発行された MQI 呼び出しと、(その元のソースに関係なく) それに対して実行依頼されたコマンドの両方を処理できます。 キュー・マネージャーは、それぞれの呼び出しとコマンドに対して、適切な完了コードを生成します。

キュー・マネージャーによって管理されるリソースには、以下のものがあります。

- IBM MQ オブジェクト定義とメッセージ・データを入れておくページ・セット
- キュー・マネージャーの障害時に、メッセージとオブジェクトを回復するために使用されるログ
- 主記憶域
- 異なるアプリケーション環境 (CICS、IMS、およびバッチ) で IBM MQ API にアクセスすることができる接続
- IBM MQ チャネル・イニシエーターは、IBM MQ システムおよびその他のシステム上の z/OS 間の通信を可能にします。

キュー・マネージャーには名前があり、アプリケーション側はこの名前を使用して接続できます。

170 ページの図 56 はキュー・マネージャーについて説明したもので、 異なるアプリケーション環境への接続およびチャネル・イニシエーターが示されています。

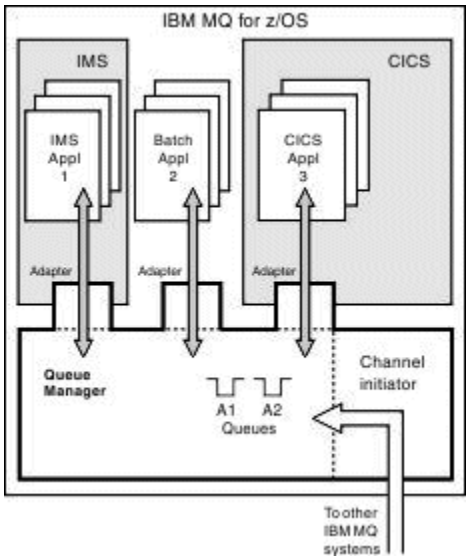


図 *56. IBM MQ for z/OS* の概要

z/OS 上のキュー・マネージャー・サブシステム

z/OS では、 IBM MQ は IPL 時に開始される z/OS サブシステムとして実行されます。 キュー・マネージャーは、ログに関する情報を含み、オブジェクト定義とメッセージ・データを保持する z/OS データ・セット (ページ・セット) を指定する JCL プロシージャーを実行することによって、サブシステム内で始動します。 サブシステムとキュー・マネージャーには、同じ名前 (最大 4 文字) があります。 ユーザーのネットワーク内のすべてのキュー・マネージャーには、固有の名前がなければなりません。 異なるシステム、シスプレックス、またはプラットフォーム上のキュー・マネージャーも、固有の名前が必要です。

z/OS 上のチャネル・イニシエーター

チャネル・イニシエーターは、IBM MQ 分散キューイングを使用可能にする資源を提供し、管理します。 IBM MQ は、メッセージ・チャネル・エージェント (MCA) を使用して、あるキュー・マネージャーから別のキュー・マネージャーにメッセージを送信します。

メッセージをキュー・マネージャー A からキュー・マネージャー B へ送信するには、 キュー・マネージャー A の送信側 MCA が キュー・マネージャー B への通信リンクをセットアップする必要があります。 その通信リンクからメッセージを受信するには、キュー・マネージャー B で受信側 MCA を開始しておく必要があります。 送信側 MCA、通信リンク、および受信側 MCA で構成される、このような一方通行のパスはチャネル と呼ばれます。 送信側 MCA は伝送キューからメッセージを取得し、チャネル経由で受信側 MCA へ送信します。 受信側 MCA は、メッセージを受信して宛先キューに入れます。

IBM MQ for z/OS では、送信側および受信側 MCA のどちらも、チャネル・イニシエーター (チャネル・イニシエーターはムーバー とも呼ばれる) 内で実行されます。 チャネル・イニシエーターは、キュー・マネージャーの制御下で z/OS アドレス・スペースとして機能します。 チャネル・イニシエーターは、1 つのキュー・マネージャーに対して 1 つだけ接続することができ、そのキュー・マネージャーと同じ z/OS イメージ内で機能します。 チャネル・イニシエーターの内部では、多数の MCA プロセスが同時に実行されています。

171 ページの図 57 には、シスプレックス内の 2 つのキュー・マネージャーが示されています。 各キュー・マネージャーには、1 つのチャネル・イニシエーターと 1 つのローカル・キューがあります。 AIX および Windows のキュー・マネージャーによって送信されたメッセージは、ローカル・キューに入れられ、そこからアプリケーションによって取り出されます。 応答メッセージも、同様の経路で戻されます。

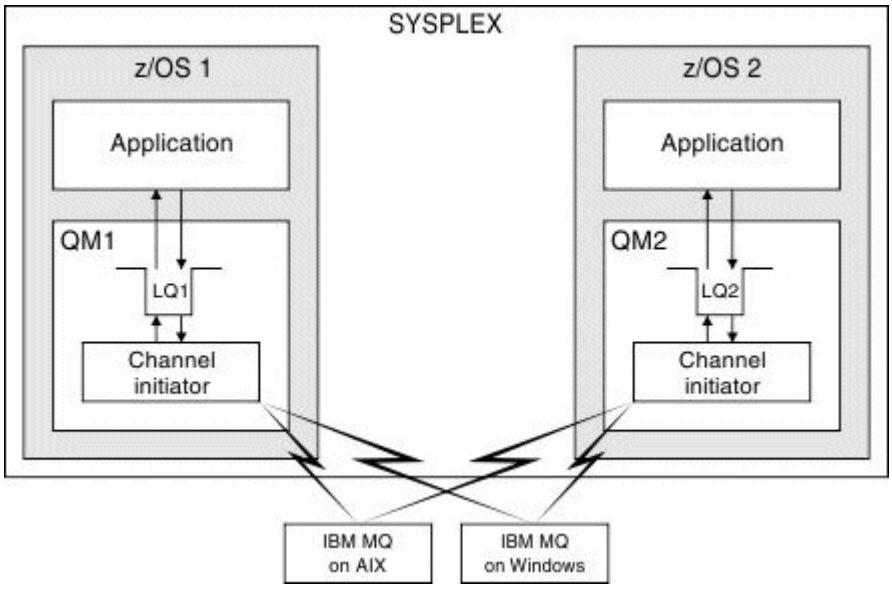


図 57. キュー・マネージャー間の通信

チャネル・イニシエーターには、チャネルの管理に関係した他のプロセスも含まれます。 これらのプロセスには、以下のものが含まれます。

リスナー
これらのプロセスは、TCP などの通信サブシステムでの着信チャネル要求を listen し、着信要求を受信すると、指定した MCA を開始します。

スーパーバイザー
これは、チャネル・イニシエーターのアドレス・スペースを管理します。 例えば、障害後のチャネルの再始動を担当します。

ネーム・サーバー
これは、TCP 名をアドレスに変換するために使用されます。

TLS タスク
これらは、暗号化および暗号化解除を実行し、証明書取り消しリストを検査するために使用されます。

z/OS チャネル・イニシエーター用の SMF レコード

チャネル・イニシエーター (CHINIT) は、タスクおよびチャネルに関する情報を持つ SMF 統計レコードおよびアカウンティング・レコードを作成できます。

CHINIT は、次のタイプの情報を持つ SMF 統計レコードおよびアカウンティング・レコードを作成できます。

- タスク: ディスパッチャー、アダプター、ドメイン・ネーム・サーバー (DNS)、および SSL。 これらのタスクは CHINIT 統計と呼ばれるものを形成します。
- チャネル: DIS CHSTATUS コマンドで使用できるのと同じようなアカウンティング情報を提供します。 これは、チャネル・アカウンティングと呼ばれます。

IBM MQ for Multiplatforms は、PCF メッセージを SYSTEM.ADMIN.STATISTICS.QUEUE。 IBM MQ for Multiplatforms で統計情報が記録される方法について詳しくは、 チャネル統計メッセージ・データ を参照してください。

統計データ

この情報を使用して、以下の情報を見つけられます。

- SSL TCB の数やどれほど CPU がこれらのタスクで使用されているかなど、CHINIT タスクについてさらに情報が必要かどうか。
- これらのタスクの要求の平均時間。
- DNS および SSL タスクについての、インターバル内の最長の要求、該当要求が発生した日時。 また、この日時の情報は、チャネルで発生する可能性のある問題に関連付けられます。

アカウンティング・データ

この情報を使用して、チャネル使用量をモニターし、以下の情報を見つけられます。

- 最高のスループットのチャネル。
- メッセージの送信率、およびデータの送信速度 (MB/秒)。
- 達成されたバッチ・サイズ。 達成されたバッチ・サイズがチャネルに指定されたバッチ・サイズに近い場合、チャネルでメッセージを送信する限界に近い可能性があります。

START TRACE および STOP TRACE コマンドを使用して、アカウンティング・トレースおよび統計トレースの収集を制御します。 STATCHL および STATACLS オプションをチャネルおよびキュー・マネージャーで使用して、チャネルが SMF データを生成するかどうかを制御できます。

z/OS IBM MQ for z/OS を管理するための用語とタスク

このトピックでは、IBM MQ for z/OS に固有の用語と作業の概要について説明します。

IBM MQ for z/OS を管理するために必要な一部の用語と作業は、z/OS プラットフォームに固有です。 以下のリストには、これらの用語と作業の一部が含まれています。

- 共用キュー

- ページ・セットおよびバッファー・プール
- ロギング
- キュー・マネージャー環境の調整
- 再始動とリカバリー
- セキュリティー
- 使用可能
- オブジェクトの取り扱い
- モニターと統計
- アプリケーション環境

共有キュー

キューは、ただ 1 つのキュー・マネージャーによって所有され、アクセス可能な非共有 か、 またはキュー共有グループ によって所有される共有 のいずれかです。 1 つのキュー共有グループは、いくつかのキュー・マネージャーで構成され、単一の z/OS シスプレックス内で稼働します。このため、同じ IBM MQ オブジェクト定義とメッセージ・データへ同時にアクセスすることができます。 1 つのキュー共有グループ内で、共有可能なオブジェクト定義は共有 Db2 データベースに保存されます。 共有キュー・メッセージは 1 つ以上のカップリング・ファシリティー構造体 (CF 構造体) 内に保持されます。 メッセージ・データが大きすぎて構造体に直接保管できない (サイズが 63 KB を超える) 場合や、メッセージがインストール済み環境で定義されたルールによってオフロード対象として選択される大きさである場合、メッセージ制御情報はカップリング・ファシリティー項目に引き続き保管されますが、メッセージ・データは共有メッセージ・データ・セット (SMDS) または共有 Db2 データベースにオフロードされます。 共有メッセージ・データ・セット、共有 Db2 データベース、およびカップリング・ファシリティー構造体は、グループ内のすべてのキュー・マネージャーによって共同で管理されるリソースです。

ページ・セットおよびバッファー・プール

メッセージが非共有キューに入れられると、キュー・マネージャーは、 後で同じキューからメッセージを取得する操作が行われるときに取り出せるように、 そのデータをページ・セットに保管します。 メッセージがそのキューから除去されると、 データが入っていたページ・セットのスペースは、後で解放されて再利用できます。 キューに入れられたメッセージの数が多くなると、ページ・セットの使用スペースも増加し、 キューのメッセージ数が少なくなると、ページ・セットの使用スペースも減少します。

ページ・セットに対するデータの書き込みと、 そこからのデータの読み取りのパフォーマンス・コストを減らすために、 キュー・マネージャーは、更新情報をプロセッサー・ストレージのバッファーに入れます。ページ・セット・アクセスをバッファーに入れるために使用されるストレージの量は、 バッファー・プールと呼ばれる IBM MQ オブジェクトによって制御されます。

ページ・セットおよびバッファー・プールについて詳しくは、 ストレージ管理を参照してください。

ロギング

ページ・セット上に保持されているオブジェクトへの変更、および持続メッセージへの操作が行われると、ログ・レコードとして記録されます。 これらのログ・レコードは、活動ログ というログ・データ・セットに書き込まれます。 活動ログ・データ・セットの名前とサイズは、ブートストラップ・データ・セット (BSDS) というデータ・セットに保持されています。

活動ログ・データ・セットがいっぱいになると、キュー・マネージャーは、 別のログ・データ・セットに切り替えてロギングを続け、 いっぱいになった活動ログ・データ・セットの内容を保存ログ・データ・セットにコピーします。 これらのアクションについての情報 (保存ログ・データ・セットの名前など) は、 ブートストラップ・データ・セットに保持されます。 概念的には、キュー・マネージャーが切り替えながら使用できる、複数の活動ログ・データ・セットがあるということです。 つまり、ある活動ログがいっぱい

になると、そのログ・データは保存ログにオフロードされるため、 活動ログ・データ・セットは再利用できるようになります。

ログおよびブートストラップ・データ・セットについては、 239 ページの『IBM MQ for z/OS でのロギング』を参照してください。

キュー・マネージャー環境の調整

キュー・マネージャーが始動されると、 キュー・マネージャーの運用方法を制御する初期設定パラメーターが読み取られます。 さらに、IBM MQ コマンドを含むデータ・セットが読み取られ、それらのコマンドが実行されます。 通常は、このようなデータ・セットには IBM MQ を実行するのに必要なシステム・オブジェクトの定義が含まれているため、データ・セットを調整することにより、それぞれの運用環境に必要な IBM MQ オブジェクトを定義または初期設定することができます。 これらのデータ・セットが読み取られると、そこで定義されているオブジェクトは、ページ・セットまたは Db2 に保管されます。

パラメーターとシステム・オブジェクトの初期設定の詳細については、250 ページの『z/OS でのシステム定義』を参照してください。

回復と再始動

IBM MQ の運用中は、ページ・セットにまだ書き込まれていない変更が、主記憶域内に保持されている可能性があります。 これらの変更は、キュー・マネージャー内のバックグラウンド・タスクによって、最低使用頻度のページ・セットへ書き込まれます。

キュー・マネージャーが異常終了した場合でも、 永続メッセージ・データがログ・レコードに記録されているため、 キュー・マネージャーを再始動するときの回復フェーズで、 消失したページ・セットの変更内容を回復できます。 このため、IBM MQ では、持続メッセージ・データとオブジェクトの変更内容を、障害発生時点の状態に回復できます。

キュー共有グループのメンバーであるキュー・マネージャーにカップリング・ファシリティー障害が発生した場合は、カップリング・ファシリティー構造体をバックアップしていた場合に限って、そのキューの持続メッセージを回復できます。

リカバリーと再始動の詳細については、261 ページの『z/OS での回復と再始動』を参照してください。

セキュリティー

Security Server (以前の RACF) などの外部セキュリティー・マネージャーを使用すれば、IBM MQ が所有し管理する資源が、許可されていないユーザーによってアクセスされないように保護できます。 チャネル・セキュリティーのために Transport Layer Security (TLS) を使用することもできます。 TLS は IBM MQ 製品の一部として組み込まれています。

IBM MQ セキュリティーの詳細については、278 ページの『IBM MQ for z/OS におけるセキュリティーの概念』を参照してください。

可用性

キュー・マネージャーや通信サブシステムの障害時にシステム使用可能性を向上させるように設計された IBM MQ の機能が、いくつかあります。 これらの機能の詳細については、284 ページの『z/OS での可用性』を参照してください。

オブジェクトの取り扱い

キュー・マネージャーが実行されている場合、IBM MQ オブジェクトは、z/OS コンソール・インターフェースを介して、または TSO の下の ISPF サービスを使用する管理ユーティリティーを介して取り扱うことができます。 どちらのメカニズムを使用しても、IBM MQ オブジェクトを定義、変更、または削除することができます。 また、IBM MQ およびキュー・マネージャーの各種機能の状況を制御および表示することもできます。

これらの機能について詳しくは、 IBM MQ for z/OS で MQSC コマンドおよび PCF コマンドを発行できるソースを参照してください。

また、IBM MQ オブジェクトの操作は、IBM MQ Explorer を使用して行うこともできます。これは、キュー、キュー・マネージャー、およびその他のオブジェクトを操作するための視覚的な方法を提供するグラフィカル・ユーザー・インターフェースです。

モニターと統計

キュー・マネージャーとチャネル・イニシエーターをモニターするために、 いくつかの機能を使用することができます。 さらに、パフォーマンスの評価およびアカウンティングのために、 統計データを収集することもできます。

これらの機能の詳細については、288 ページの『IBM MQ for z/OS でのモニターと統計』を参照してください。

アプリケーション環境

キュー・マネージャーが始動されると、アプリケーションを接続して、IBM MQ API の使用を開始することができます。 これらは、CICS、IMS、バッチ、または WebSphere Application Server のアプリケーションです。 IBM MQ アプリケーションは、 CICS ブリッジおよび IMS ブリッジを使用して、 IBM MQ を認識しない CICS および IMS システム上のアプリケーションにアクセスすることもできます。

これらの機能の詳細については、291 ページの『IBM MQ およびその他の z/OS 製品』を参照してください。

IBM MQ アプリケーションの作成については、以下の資料を参照してください。

- アプリケーションの開発
- C++ の使用
- IBM MQ classes for Java の使用

z/OS 共用キューとキュー共用グループ

共用キューおよびキュー共用グループを使用すると、IBM MQ リソースの高可用性を実装することができます。 共有キューおよびキュー共有グループは、IBM MQ for z/OS プラットフォーム上の z/OS に固有な機能です。

このセクションでは特徴と利点を説明し、いくつかのキュー・マネージャーが、同じキューおよびそれらのキュー上のメッセージをどのように共用するかについて示します。

共用キューとは

共用キューとは、一種のローカル・キューです。 そのキューにあるメッセージは、 シスプレックス内の複数のキュー・マネージャーがアクセスできます。

キュー共用グループ

同じ共用キュー群にアクセスできるキュー・マネージャーは、キュー共用グループ というグループを形成します。

任意のキュー・マネージャーがメッセージにアクセスできる

共用キューには、キュー共用グループ内の任意のキュー・マネージャーがアクセスできます。 つまり、あるキュー・マネージャーで共用キューにメッセージを書き込み、 別のキュー・マネージャーでそのキューから同じメッセージを取り出せます。 これにより、キュー共用グループ内での通信が迅速に行えるため、キュー・マネージャー間のチャネルを活動状態にする必要はなくなります。

IBM MQ は、 Db2 または共有メッセージ・データ・セット (SMDS) へのメッセージのオフロードをサポートします。 任意のサイズのメッセージのオフロードが構成可能です。

176 ページの図 58 には、1 つのキュー共用グループを形成する 3 つのキュー・マネージャーと 1 つのカップリング・ファシリティーが示されています。 3 つのキュー・マネージャーはすべてカップリング・ファシリティーの共用キューにアクセスできます。

アプリケーションは、キュー共用グループ内にある任意のキュー・マネージャーへ接続できます。 キュー共用グループ内のすべてのキュー・マネージャーがどの共用キューにもアクセスできるため、アプリケーションは特定のキュー・マネージャーの可用性に依存しません。 キュー共用グループ内のどのキュー・マネージャーも、キューの操作が可能です。

キュー・マネージャーのうちの 1 つに問題が生じても、 キュー共用グループ内の他のキュー・マネージャーはすべてキューの処理を続行できるため、非常に可用性が高くなります。

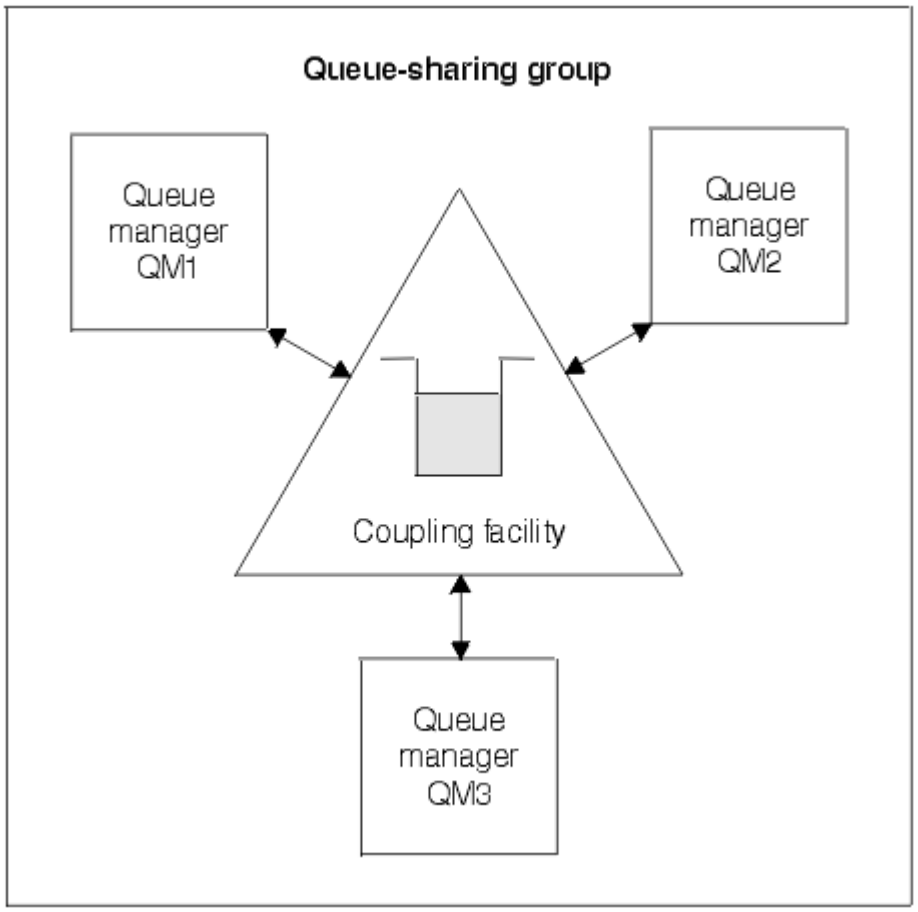


図 *58.* キュー共用グループ

すべてのキュー・マネージャーによって共用されるキュー定義

共用キュー定義は、Db2 データベース表 OBJ_B_QUEUE に保管されます。 そのため、キューは一度定義するだけで十分で、その後はキュー共用グループ のすべてのキュー・マネージャーでアクセスすることができます。 つまり、作成しなければならない定義はほとんどないことになります。

対照的に、非共用キューの定義は、そのキューを所有するキュー・マネージャーの ページ・セット 0 に格納されます (ページ・セットで説明されているとおり)。

同じ名前のキューが、キュー・マネージャー定義のページ・セット で定義されている場合、共用キューを定義することはできません。 同様に、同じ名前の共用キューが存在している場合にも、キュー・マネージャー のページ・セットにローカル・バージョンのキューを定義できません。

キュー共用グループとは

同じ共用キューへアクセスできるキュー・マネージャーのグループは、キュー共用グループと呼ばれます。キュー共用グループの各メンバーには、同じ共用キューのセットに対するアクセス権があります。

キュー共用グループには、最大 4 文字の名前があります。 この名前はネットワーク内で固有であり、かつ、キュー・マネージャー名とは異なるものである必要があります。

177 ページの図 59 には、2 つのキュー・マネージャーを含む キュー共用グループが示されています。 各キュー・マネージャーには、1 つのチャネル・イニシエーターと、 そのローカル・ページ・セットおよびログ・データ・セットがあります。

キュー共用グループの各メンバーは、Db2 システムにも接続していなければなりません。 この Db2 システムはすべて同じ Db2 データ共用グループに含め、キュー・マネージャーが、共用オブジェクト定義を入れておくときに使用する Db2 共用リポジトリーへアクセスできるようにします。 これらは任意のタイプの IBM MQ オブジェクト (例えば、キューおよびチャネル) の定義です。一度だけ定義すれば、グループ内の任意のキュー・マネージャーで使用できます。 これらをグローバル 定義と呼び、専用定義とグローバル定義で説明されています。

複数のキュー共用グループが 1 つの特定データ共用グループを参照することが可能です。 Db2 サブシステムの名前、およびキュー・マネージャーが始動時に IBM MQ システム・パラメーターで使用するデータ共有グループを指定します。

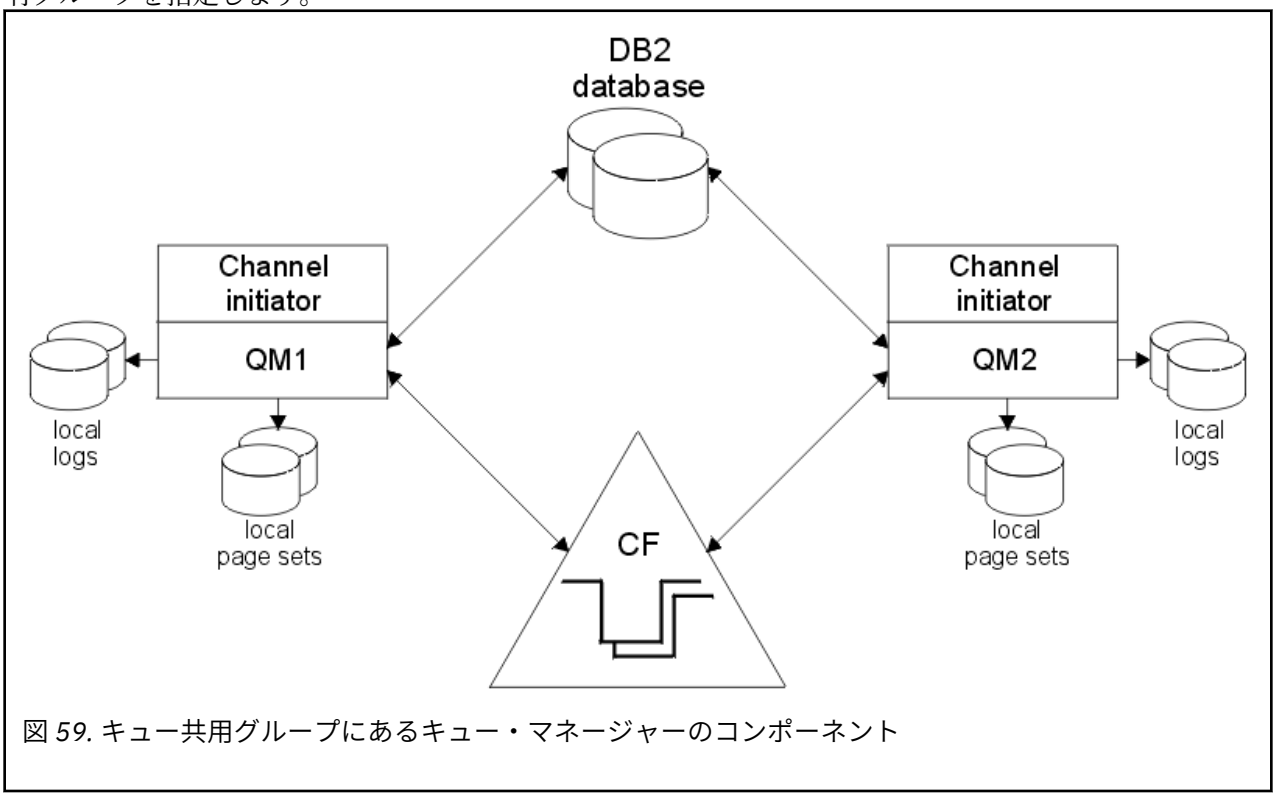


図 *59.* キュー共用グループにあるキュー・マネージャーのコンポーネント

キュー・マネージャーをキュー共用グループへ結合すると、そ のキュー・マネージャーを使用して、そのグループに定義され ている共用オブジェクトへアクセスしたり、グループ内で新し い共用オブジェクトを定義できるようになります。 グループ内で共用キューを定義すると、このキュー・マネージャーで、 そのような共用キューに対するメッセージの読み書きができます。 グループ内のキュー・マネージャーはいずれも、共用キューに保持されているメッセージを取り出すことができます。

MQSC コマンドを一度入力すれば、 各キュー・マネージャーで個々にコマンドを入力しなくても、 キュー共用グループ内のすべてのキュー・マネージャーでそのコマンドを実行できます。 このときに、コマンド有効範囲 属性が使用されます。 この属性については、異なるキュー・マネージャーに対するコマンドの送信に説明されています。

キュー・マネージャーをキュー共用グループのメンバーとして実行する場合、そのキュー・マネージャーに対して私用に定義された IBM MQ オブジェクトと、キュー共用グループのすべてのキュー・マネージャーが使用できるようグローバルに定義された IBM MQ オブジェクトを区別することができます。 このとき

に、キュー共用グループ後処理 属性 が使用されます。 この属性については、専用定義とグローバル定義で説明されています。

グループ内にある IBM MQ オブジェクトへのアクセスを制御する、1 まとまりのセキュリティー・プロファイルを定義できます。 これにより、定義しなければならないプロファイルの数が著しく減少します。

1 つのキュー・マネージャーは、1 つのキュー共用グループにだけ属することができ、 またグループ内のすべてのキュー・マネージャーは同じシスプレックス内になければなりません。 始動時のシステム・パラメーターには、キュー・マネージャーが属するキュー共用グループを指定します。

関連概念

178 ページの『共用キューのメッセージの保管場所』
共用キュー上の各メッセージは、z/OS カップリング・ファシリティー・リスト構造体内のエントリーで表されます。 メッセージ・データが大きすぎて同一エントリー内に収まらない場合は、共用メッセージ・データ・セット (SMDS) か Db2 にオフロードされます。

195 ページの『共用キュー使用の利点』
共用キューを使用すると、IBM MQ applications の拡張が容易になり、可用性が高くなり、ワークロード・バランシングを実行できるようになります。

215 ページの『分散キューイングとキュー共用グループ』
分散キューイングおよびキュー共用グループは、アプリケーション・システムの可用性を高めるために使用できる 2 つの手法です。 このトピックでは、これらの手法の詳細について知ることができます。

219 ページの『共用キューを使用したワークロード分散への影響』
このトピックでは、キュー共用グループにおける共用キューを使用したワークロード分散に影響するいくつかの要素について理解することができます。

関連資料

220 ページの『共用キューおよびキュー共用グループに関する詳細』
このトピックの表を使用して、IBM MQ for z/OS が共有キューおよびキュー共有グループを使用する方法についての詳細を確認してください。

z/OS 共用キューのメッセージの保管場所

共用キュー上の各メッセージは、z/OS カップリング・ファシリティー・リスト構造体内のエントリーで表されます。 メッセージ・データが大きすぎて同一エントリー内に収まらない場合は、共用メッセージ・データ・セット (SMDS) か Db2 にオフロードされます。

システム・クラス・メモリー (SCM) を使用するように CF 構造が構成されていれば、追加の構成をしなくても IBM MQ からその機能を使用できます。

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

共用キュー・メッセージ・ストレージ

共用キューに書き込まれたメッセージは、ページ・セットには格納されません。 また、バッファー・プールを使用することもありません。

共用キューのメッセージは、z/OS カップリング・ファシリティー (CF) 内のリスト構造体上にエントリーを持っています。 同じシスプレックス内のいろいろなキュー・マネージャーが、CF リスト構造体を使用してそれらのメッセージにアクセスできます。

小規模な共用キュー・メッセージのメッセージ・データは、通常はカップリング・ファシリティー・エントリー内に組み込まれます。 大規模メッセージの場合、メッセージ・データは、共用メッセージ・データ・セット (SMDS) 内に保管するか、1 つ以上の 2 進ラージ・オブジェクト (BLOB) として、Db2 データ共用グループによって共用される Db2 表に保管することができます。 63 KB を超えるメッセージ・データは、常に SMDS か Db2 にオフロードされます。 オプションで、これより小さいメッセージも同じ方法でオフロードして、カップリング・ファシリティー構造体のスペースを節約できます。 詳細については、180 ページの『共用メッセージのオフロード・オプションの指定』を参照してください。

1 つの共用キューに書き込まれたメッセージは、MQGET によって検索されるまで、カップリング・ファシリティー構造体の中で参照されます。 カップリング・ファシリティーの操作は、次のことを行うために使用されます。

- 次の検索可能なメッセージを検索する
- 共用キューのコミットされていないメッセージをロックする
- コミットされたメッセージの到着について、関連するキュー・マネージャーに通知する

持続メッセージに対する MQPUT および MQGET 操作が、その操作を実行している キュー・マネージャーのログに記録されます。 このことは、カップリング・ファシリティーの障害時のデータ損失のリスクを 最小にします。

カップリング・ファシリティー

共用キューに入れられたメッセージは、 実際にはカップリング・ファシリティー内で参照されます。 カップリング・ファシリティーは、シスプレックスの z/OS イメージの外側にあり、一般には別の電源機構で稼働するよう構成されています。 したがって、カップリング・ファシリティーはソフトウェア障害に対して強く、 ハードウェア障害や停電に対しても強くなるよう構成できます。 そのため、カップリング・ファシリティーに格納されているメッセージは可用性の高いものです。

IBM MQ で使用されるカップリング・ファシリティーの各リスト構造体は、1 つの特定キュー共用グループ専用になりますが、1 つのカップリング・ファシリティーには複数のキュー共用グループの構造体を含めることが可能です。 別のキュー共用グループにあるキュー・マネージャーは、 データを共用することができません。 カップリング・ファシリティーのリスト構造体へ同時に接続できるキュー・マネージャーの 数は、1 つのキュー共用グループにつき、最大で 32 です。

1 つのカップリング・ファシリティーのリスト構造体には、512 までの共用キューを含めることができます。 構造体に格納されるメッセージ・データの合計量は、構造体の容量によって制限されます。 ただし、**CFLEVEL(5)** では、オフロード・パラメーターを使用して、63 KB 未満のメッセージのデータをオフロードすることができます。これにより、構造に保管できるメッセージの数が増えます。ただし、各メッセージには、少なくとも 1 つのカップリング・ファシリティー項目と、少なくとも 768 バイトのデータ (項目用に 256 バイト、ヘッダーと記述子の 2 つのエレメント用に 512 バイト) が必要です。

リスト構造体のサイズは、次の要因によって制限されます。

- 1 つのカップリング・ファシリティー内に収まること。
- 使用可能なカップリング・ファシリティー・ストレージを IBM MQ および他製品の他の構造体と共用できること。

カップリング・ファシリティーのリスト構造には、 ストレージ・クラス・メモリーを関連付けることができます。 一部の状況では、共有キューと共に使用するときに、このストレージ・クラス・メモリーが役立つことがあります。 詳しくは、197 ページの『共有キューでのストレージ・クラス・メモリーの使用』 を参照してください。

CF 構造体のサイズの計画

CF 構造体のサイズ設定に関する指示が必要な場合は、MP16: IBM MQ for z/OS Capacity planning and tuning サポートパックを使用できます。 CF サイズを支援するために IBM によって提供される Web ベースのツール CFSizer を使用することもできます。

CF 構造体オブジェクト

キュー・マネージャーのカップリング・ファシリティー構造体の使用は、CF 構造体 (CFSTRUCT) IBM MQ オブジェクトで指定されます。

これらの構造体オブジェクトは Db2 に保管されます。

カップリング・ファシリティー構造体に関連した z/OS コマンドまたは定義を使用するときには、キュー共用グループ名の最初の 4 文字が必要です。 しかしながら、IBM MQ CFSTRUCT オブジェクトは常に単一のキュー共用グループ内に存在するので、その名前にはキュー共用グループの名前の最初の 4 文字は組み込まれていません。 例えば、SQ03 で始まるキュー共用グループ内で定義された CFSTRUCT(MYDATA) は、カップリング・ファシリティー・リスト構造体 SQ03MYDATA を使用します。

CF 構造体は、その機能を判別する、以下の CFLEVEL 属性を持っています。

- 1、2 - 63 KB より小さい非持続メッセージに使用可能
- 3 - 63 KB より小さい持続および非持続メッセージに使用可能
- 4 - 100 MB までの持続および非持続メッセージに使用可能
- 5 - 100 MB までの持続および非持続メッセージに使用可能で、選択的に共用メッセージ・データ・セット (SMDS) または Db2 にオフロード可能

注 : IBM MQ を使用する場合、カップリング・ファシリティー・ストラクチャーを暗号化することができます。 詳しくは、 カップリング・ファシリティー・ストラクチャー・データの暗号化 を参照してください。

カップリング・ファシリティーのバックアップとリカバリー

カップリング・ファシリティー・リスト構造体は、IBM MQ コマンド BACKUP CFSTRUCT を使用してバックアップすることができます。 このコマンドは、現在 CF 構造体内にある持続メッセージのコピーを、バックアップを作成するキュー・マネージャーのアクティブ・ログ・データ・セットに書き込み、バックアップ・レコードを Db2 に書き込みます。

カップリング・ファシリティーに障害が発生した場合には、IBM MQ コマンド RECOVER CFSTRUCT を使用することができます。 このコマンドは、Db2 にあるバックアップ・レコードを使用して、CF 構造体のバックアップから持続メッセージを検出し復元します。 最後のバックアップ以降のすべての活動が、 キュー共用グループ内のすべてのキュー・マネージャーのログを使用して再生された後、CF 構造体が 障害前の状態にまで復元されます。

詳細については、 BACKUP CFSTRUCT および RECOVER CFSTRUCT コマンドを参照してください。

関連概念

180 ページの『共用メッセージのオフロード・オプションの指定』
共用キュー・メッセージのメッセージ・データを Db2 表に格納するのか、共用メッセージ・データ・セット (SMDS) に格納するのかを選択できます。 さらに、メッセージのサイズとカップリング・ファシリティー構造体 (CF) の現在の使用状況に基づいて、どのメッセージをオフロードするのかも選択できます。

183 ページの『共用メッセージ・データ・セット (SMDS) 環境の管理』
大規模メッセージをオフロードするための共用メッセージ・データ・セットを選択する場合は、IBM MQ がこれらのデータ・セットを管理するために使用する情報と、この情報を処理するために使用されるコマンドについても知っている必要があります。 このトピックを使用して、共用メッセージ・データ・セットを管理する方法を理解できます。

z/OS 共用メッセージのオフロード・オプションの指定

共用キュー・メッセージのメッセージ・データを Db2 表に格納するのか、共用メッセージ・データ・セット (SMDS) に格納するのかを選択できます。 さらに、メッセージのサイズとカップリング・ファシリティー構造体 (CF) の現在の使用状況に基づいて、どのメッセージをオフロードするのかも選択できます。

共用キューのメッセージ・データは、カップリング・ファシリティーからオフロードして、 Db2 テーブルまたは 共用メッセージ・データ・セット (SMDS) と呼ばれる IBM MQ 管理対象データ・セットのいずれかに保管することができます。

カップリング・ファシリティーの項目サイズである 63 KB より大きいメッセージの場合は、メッセージ・データを SMDS にオフロードすれば、Db2 にオフロードする場合に比べて、パフォーマンスが大幅に改善される可能性があります。

すべての共用キュー・メッセージは、カップリング・ファシリティー構造のリスト項目によって引き続き管理されますが、メッセージ・データを SMDS にオフロードした場合は、一部の制御情報と、メッセージが格納されているディスク・ブロックへの参照のリストだけがカップリング・ファシリティーの項目に入

ることになります。 したがって、このメカニズムを使用すれば、それぞれのメッセージで必要なカップリング・ファシリティー要素のストレージは、メッセージの実際のサイズのごく一部になります。

共用キュー・メッセージの格納場所の選択

SMDS または Db2 共用メッセージ・ストレージの選択は、**CFSTRUCT** 定義の **OFFLOAD(SMDS|DB2)** パラメーターによって制御されます。 **OFFLOAD(SMDS)** はデフォルト値です。

また、このパラメーターでは、**CFSTRUCT** が **CFLEVEL(5)** 以上を使用する必要があります。

OFFLOAD パラメーターは、**CFLEVEL(5)** 以降でのみ有効です。 詳しくは、DEFINE CFSTRUCT を参照してください。

OFFLOAD(DB2) は、主にマイグレーションの目的でサポートされます。

オフロードする共用キュー・メッセージの選択

メッセージ・データは、メッセージ・データのサイズとカップリング・ファシリティー構造体の現在の使用状況に基づいて、SMDS または Db2 にオフロードされます。 この点については 3 つの規則があり、それぞれの規則で、対応するパラメーターのペアを指定します。 これらのパラメーターは、対応するカップリング・ファシリティー構造の使用率しきい値のパーセンテージ (**OFFLDnTH**) およびメッセージ・サイズ制限 (**OFFLDnSZ**) です。

現在実装されている 3 つの規則を指定するために、以下のキーワード・ペアを使用します。

- OFFLD1TH と OFFLD1SZ
- OFFLD2TH と OFFLD2SZ
- OFFLD3TH と OFFLD3SZ

規則のペア	デフォルト値	説明
規則のペア 1	OFFLD1TH(70) と OFFLD1SZ(32K)	カップリング・ファシリティー構造の使用率が 70% を上回る場合に、32 KB を超えるメッセージのデータをオフロードします。
規則のペア 2	OFFLD2TH(80) と OFFLD2SZ(4K)	カップリング・ファシリティー構造の使用率が 80% を上回る場合に、4 KB を超えるメッセージのデータをオフロードします。
規則のペア 3	OFFLD3TH(90) と OFFLD3SZ(0K)	カップリング・ファシリティー構造の使用率が 90% を上回る場合に、0 KB を超えるメッセージ (すべてのメッセージ) のデータをオフロードします。

オフロード規則の OFFLD *x* SZ 値が 64K の場合は、規則が有効ではないことを示します。 このケースでは、別のオフロード規則が有効になっている場合、またはメッセージが 63.75 KB を超えていて構造体に格納するには大きすぎる場合にのみ、メッセージのオフロードが行われます。

メッセージがオフロードされたとしても、カップリング・ファシリティーでは、メッセージ 1 つにつき 0.75 KB のストレージが必要です。

3 つのオフロード規則は、構造体ごとに指定でき、以下の使用目的があります。

- パフォーマンス
  - アプリケーション構造内に十分なスペースがある場合、メッセージ・データをオフロードする必要があるのは、メッセージ・データが大きすぎて構造内に格納できない場合か、メッセージ・サイズのしきい値のうち小さい値を超えているために、必要な構造体スペース量と比較して構造内に格納する価値がパフォーマンス上ない場合に限られます。
  - 特定のメッセージ・サイズしきい値が必要な場合、通常 1 つ目のオフロード規則を使用して指定されます。

- キャパシティー
  - アプリケーション構造内のスペースが非常に少ない場合は、残りのスペースを最大限活用するために、最大量のメッセージ・データをオフロードする必要があります。
  - 通常、3 つ目のオフロード規則は、構造体が満杯間近になっている際に、ほとんどのメッセージをオフロードする必要があり、アプリケーション構造内のエントリーは通常の最小サイズ (約 0.75K バイトが必要) になるようにすることを示すために使用されます。
  - アプリケーション構造のサイズと予想される最大バックログに基づいて、使用量しきい値パラメーターを選択する必要があります。 例えば、予想される最大バックログが 1M のメッセージである場合、このメッセージ数にとって必要な構造体のストレージの量は約 0.75G バイトになります。 したがって、例えば構造体が約 10G バイトの場合は、すべてのメッセージをオフロードするための使用量しきい値を 92% 以下に設定しなければなりません。
  - 構造体のスペースはエレメントとエントリーに分割され、全体のスペースは十分ある場合でも、これらの内一方が他方より先に使い尽くされる可能性があります。 システムは必要な時点で比率を調整する AUTOALTER 機能を提供していますが、この機能はあまり感度が高くないので、実際に使用できるスペースの量が多少減る可能性があります。 したがって、構造体の最大スペースの 90% 以下を使用することを目標にする方が望ましいことがあります。つまり前述の例では、すべてのメッセージをオフロードするための使用量しきい値を 80% 前後に設定する方が望ましいことになります。
- クッションのある遷移
  - カップリング・ファシリティー構造内の残存スペース量が少なくなるとき、パフォーマンス特性が突然大きく変わることは望ましくありません。 使用エントリーと使用エレメントの通常比のしきい値が突然変わることも、カップリング・ファシリティーの管理の点から望ましくありません。
  - 通常、2 つ目のオフロード規則は、パフォーマンスに偏ったオフロード規則と容量に偏ったオフロード規則の間の中間クッションを提供するために使用されます。 これを設定すると、カップリング・ファシリティー構造で使用されるスペースが中間しきい値を超えたときに、オフロード・アクティビティーを大幅に増やすことができます。 その結果、残りのスペースが使い尽くされるまでの時間を延ばせるので、使用レベルの上昇に適合するためにカップリング・ファシリティーの自動変更処理に費やせる時間が長くなります。

カップリング・ファシリティー構造を拡張できない環境で、格納しなければならないメッセージの最低数が決まっている場合は、必要に応じて 3 番目の規則を変更し、すべてのメッセージのデータのオフロードを開始するためのしきい値を適切な値に設定することによって、その決まった数のメッセージを格納するためのスペースを確保できます。

例えば、カップリング・ファシリティー構造のサイズが 4 GB で、格納しなければならないメッセージの数が 100 万件だとすれば、1,000,000 * 0.75 KB のスペース (768 MB) が必要になります。この値は、4 GB の 18.75% に相当します。 この場合は、すべてのメッセージをオフロードするためのしきい値を 90% ではなく 80% 程度に設定する必要があります。 つまり、パラメーターは、OFFLD3TH(80) と OFFLD3SZ(0K) になります。 そうであれば、その他のオフロード・パラメーターも調整する必要があります。

非常に小さいメッセージのオフロードがパフォーマンスにかなりの影響を及ぼしているものの、大きいメッセージの場合と比べれば、その影響は比較的小さい、という状況が明らかになった場合は、その他の規則の使用状況に関するしきい値を小さい値に変更し、大きいメッセージを早期にオフロードするようにして、構造内にオフロード前の小さいメッセージを格納するためのスペースをより多く残すこともできます。

例えば、32KB を超えるメッセージが多数存在するものの、それらのメッセージをオフロードするための経過時間のパフォーマンス (RMF 統計またはアプリケーション・パフォーマンスで確認できる) が、カップリング・ファシリティーにそれらのメッセージを保持しておく場合のパフォーマンスとほぼ同じであれば、最初の規則のしきい値を 0% に設定し、そのようなメッセージをすべてオフロードするように構成できます。 つまり、パラメーターは、OFFLD1TH(0) と OFFLD1SZ(32K) になります。 この場合も、その他のオフロード・パラメーターを調整する必要があります。

特定の中間サイズ (16 KB や 6 KB など) のメッセージが多数存在する場合は、2 番目の規則のメッセージ・サイズ・オプションを変更することによって、大きなメッセージを比較的小さい使用状況しきい値

でオフロードし、かなりの量のスペースを節約する一方で、小さいメッセージをカップリング・ファシリティー内だけに格納しておく、という方法も採用できます。

z/OS 共用メッセージ・データ・セット *(SMDS)* 環境の管理

大規模メッセージをオフロードするための共用メッセージ・データ・セットを選択する場合は、IBM MQ がこれらのデータ・セットを管理するために使用する情報と、この情報を処理するために使用されるコマンドについても知っている必要があります。 このトピックを使用して、共用メッセージ・データ・セットを管理する方法を理解できます。

SMDS オブジェクト

それぞれの共用メッセージ・データ・セットのプロパティーと状況は、キュー共用グループにある任意のキュー・マネージャーを介して更新できる共用 SMDS オブジェクトで追跡されます。

それぞれのカップリング・ファシリティー・アプリケーション構造にアクセスできるキュー・マネージャーごとに、 1 つの共用メッセージ・データ・セットがあります。 共用メッセージ・データ・セットは、SMDS キーワードを使用して指定される所有キュー・マネージャー名と、 CFSTRUCT キーワードを使用して指定されるアプリケーション構造名によって識別されます。

注 : 構造に SMDS データ・セットを定義するときは、キュー・マネージャーごとに 1 つ必要です。

SMDS オブジェクトは、Db2 に保管された対応する CFSTRUCT オブジェクトの拡張部分を形成する配列 (グループ内のキュー・マネージャーごとに 1 つの項目を持つ) に保管されます。

SMDS オブジェクトは、CFSTRUCT オブジェクトの一部として作成または削除されるため、 DEFINE や DELETE を行うコマンドはありませんが、個々の所有キュー・マネージャーの設定を変更するために ALTER を行うコマンドはあります。

SMDS コマンドの詳細については、194 ページの『SMDS 関連コマンド』を参照してください。

SMDSCONN 情報

共用メッセージ・データ・セットが正常な状態にあるとき、 セキュリティー定義や直接アクセス装置の接続に問題があるなどの理由で 1 つ以上のキュー・マネージャーがそれに接続できないことがあります。 そのため、各キュー・マネージャーでは接続状況と、 それぞれの共用メッセージ・データ・セットの可用性情報 (例えば、現在接続できるかどうか、 接続できない場合はなぜかなど) を追跡する必要があります。

SMDSCONN 情報は、共用メッセージ・データ・セットへのキュー・マネージャー接続を表します。 共用メッセージ・データ・セット自体は、その共用メッセージ・データ・セット (共用オブジェクト自体の SMDS キーワードで指定される) を所有するキュー・マネージャーと CFSTRUCT 名の組み合わせによって識別されます。

特定のキュー・マネージャーに向けられたコマンドが参照できるのはその同じキュー・マネージャーの SMDSCONN 情報だけであるため、接続中のキュー・マネージャーを識別するパラメーターはありません。

SMDSCONN 情報項目は所有キュー・マネージャーの主ストレージで維持され、そのキュー・マネージャーの再始動時に再作成されます。 ただし、個々のキュー・マネージャーからの接続が明示的に停止された場合、 この情報はキュー・マネージャーの再始動時に持続するように、対応する CFSTRUCT または SMDS オブジェクトの接続配列にもフラグとして保管されます。

状況と可用性に関する情報

状況情報は、リソースや接続の状態 (例えば、まだ使用されていないか、正常に使用されているか、リカバリーが必要かどうかなど) を示します。 これは通常、STATUS キーワードを使用して記述されます。 可能な値は、オブジェクトのタイプによって異なります。

状況情報は通常、リソースや接続の使用時にエラーが検出された場合などに自動的に更新されます。 ただし、キュー・マネージャーが正確な状況を自動的に判別できないケースなど、 場合によってはコマンドを使用して状況を更新することもできます。

可用性情報は、リソースや接続が使用可能であるかどうかを示し、 通常は主に状況情報によって判別されます。 共用メッセージ・データ・セット・サポートで使用されるリソースや接続のタイプについて、 3 つのレベルの可用性が実装されています。

使用可能
これは、リソースが通常の用途に使用可能であることを意味します。 必ずしもそれが現在使用されていること (これは STATUS 値から判別できます) を意味しません。 データ・セットが再始動処理を必要する場合、 この設定によって所有キュー・マネージャーがそのデータ・セットを開くことが可能になりますが、 他のキュー・マネージャーはデータ・セットが ACTIVE 状態に戻るまで待機する必要があります。

エラーのため使用不可
これは、エラーのためにリソースが自動的に使用不可にされ、 何らかの形態の修復またはリカバリー処理が実行されるまで再び使用可能になることは予測されないことを意味します。 ただし、再び使用可能にしようとする試みは、オペレーター介入なしで許可されます。 そのような試みは、リソースに使用可能のマークを付けるコマンドや、 リカバリー処理が完了したことを示すように状況を変更するコマンドによっても起動できます。

リソースが使用不可にされた理由は、通常は関連する STATUS 値から明らかですが、場合によっては他の理由によってリソースが使用不可にされることもあり、 その場合は理由を示すために別個の REASON 値が提供されます。

オペレーター・コマンドのため使用不可
これは、リソースへのアクセスがコマンドによって明示的に使用不可にされたことを意味します。 このリソースは、それを再び使用可能にするコマンドを使用することによってのみ使用可能にできます。

SMDS の可用性

共用 SMDS オブジェクトでは、 可用性は ACCESS キーワードによって記述され、可能な値は ENABLED、SUSPENDED、および DISABLED です。

可用性は、ACCESS (ENABLED) または ACCESS (DISABLED) を設定するために、グループ内の任意のキュー・マネージャーから関連する共有オブジェクトに対して **RESET SMDS** コマンドを使用して更新できます。

以前の可用性が ACCESS(SUSPENDED) であった場合、 これを ACCESS(ENABLED) に変更すると、共用メッセージ・データ・セットを使用しようとする新しい試みが起動しますが、 以前のエラーが依然として存在している場合、可用性は ACCESS(SUSPENDED) にリセットされます。

SMDSCONN の可用性

ローカル SMDSCONN 情報項目では、 可用性は AVAIL キーワードによって記述され、可能な値は NORMAL、 ERROR、または STOPPED です。 接続を有効または無効にするために、特定のキュー・マネージャーにアドレス指定された **START SMDSCONN** または **STOP SMDSCONN** コマンドを使用して、可用性を更新することができます。

以前の可用性が AVAIL(ERROR) であった場合、 これを AVAIL(NORMAL) に変更すると、共用メッセージ・データ・セットを使用しようとする新しい試みが起動しますが、 以前のエラーが依然として存在している場合、可用性は AVAIL(ERROR) にリセットされます。

共用メッセージ・データ・セットの共用の状況と可用性

各共用メッセージ・データ・セットの可用性は、共用状況情報を使用してグループ内で管理されます。この情報は、TYPE (SMDS) を指定した **DISPLAY CFSTATUS** コマンドを使用して表示できます。 このコマンドでは、構造ごとに 1 つのデータ・セットをアクティブにした各キュー・マネージャーの状況情報が表示されます。 各データ・セットは、以下の状態のいずれかです。

NOTFOUND
これは、対応するデータ・セットがまだアクティブにされていないことを意味します。 この状況は特定のキュー・マネージャーが指定されたときにのみ発生し、 すべてのキュー・マネージャーが選択された場合は、 アクティブにされていないデータ・セットがスキップされます。

NEW
データ・セットは初めてオープンされて初期化されており、アクティブになる準備ができています。

ACTIVE
これは、データ・セットが完全に使用可能であり、 構造用のすべてのアクティブ・キュー・マネージャーによって割り振られ、開かれる必要があることを意味します。

失敗
これは、データ・セットが (リカバリー処理を除いて) まったく使用できず、 すべてのキュー・マネージャーがそれを閉じて割り振り解除しなければならないことを意味します。

INRECOVER
これは、このデータ・セットについてメディア・リカバリー (RECOVER CFSTRUCT を使用) が進行中であることを意味します。

RECOVERED
これは、障害が発生したデータ・セットをアクティブ状態に戻すコマンドが出されたが、 その後で必要な再始動処理がまだ完了していないため、 データ・セットは所有キュー・マネージャーが再始動処理の目的でのみ開けることを示します。

EMPTY
データ・セットにメッセージは含まれていません。 データ・セットは、 所有キュー・マネージャーによって正常に閉じられた場合に、 メッセージが入っていないとこの状態になります。 アプリケーション構造が空になったために (TYPE PURGE を指定した **RECOVER CFSTRUCT** を使用するか、またはリカバリー不能構造の場合のみ、構造の前のインスタンスを削除することによって)、前のデータ・セットの内容を破棄するときに、EMPTY 状態にすることもできます。 所有するキュー・マネージャーによって次回データ・セットがオープンされる際に、スペース・マップが空にリセットされ、状況は ACTIVE に変更されます。 以前のデータ・セットの内容は不要のため、この状態のデータ・セットを新たに割り振られたデータ・セットで置き換えて、例えば、スペース割り振りを変更したり、別のボリュームに移動したりすることができます。

コマンド出力には、 リカバリー・ログ (存在する場合) が使用可能にされた日時と、 データ・セット (現在アクティブでない場合) に障害が発生した日時が含まれます。

共用メッセージ・データ・セットは、 **RESET SMDS** コマンドによって FAILED 状態にすることも、以下のいずれかのタイプのエラーが検出されたときに自動的に FAILED 状態にすることもできます。

- データ・セットを所有キュー・マネージャーが割り振ったり開いたりすることができません。
- データ・セットが任意のキュー・マネージャーによって正常に開かれた後、データ・セット・ヘッダーの検証が失敗しました。
- 所有キュー・マネージャーがデータの読み取りや書き込みを行っている際に永続入出力エラーが発生しました。
- オープン処理と検証が正常に完了したデータ・セットから別のキュー・マネージャーがデータを読み取っている際に、 永続入出力エラーが発生しました。

データ・セットは FAILED または INRECOVER 状態にある場合、 通常の用途に使用することはできません。 そのため、可用性状態が ACCESS(ENABLED) である場合は ACCESS(SUSPENDED) に変更されます。

データ・セットが FAILED 状態にされたがメディア・リカバリーは必要ない場合 (例えば、 データは引き続き有効であるがストレージ・デバイスが一時的にオフラインになった場合)、**RESET SMDS** コマンドを使用して、 状況を RECOVERED 状態に直接変更することを要求できます。

リカバリー処理の完了時に、または **RESET SMDS** コマンドの結果としてデータ・セットが RECOVERED 状態に入った場合、 そのデータ・セットは再始動処理が完了した直後に再び使用可能になります。 データ・セットが ACCESS(SUSPENDED) 状態であった場合は、 自動的に ACCESS(ENABLED) 状態に戻り、 所有キュー・マネージャーが再始動処理を実行することを可能にします。 再始動処理が完了すると、状態は ACTIVE に変更され、 他のすべてのキュー・マネージャーが再びそのデータ・セットに接続可能になります。

共用メッセージ・データ・セットの接続の状況と可用性

キュー・マネージャーは、自身およびグループ内の他のキュー・マネージャーが所有するそれぞれの共用メッセージ・データ・セットへの接続について、 ローカルの状況および可用性情報を維持します。 この情報は、**DISPLAY SMDSCONN** コマンドを使用して表示できます。

他のキュー・マネージャーに属する ACTIVE 状態の共用メッセージ・データ・セットにアクセスできない場合、 キュー・マネージャーは自身の視点から、その接続に使用不可のフラグを付けます。

エラーがデータ・セット自体に問題があることを明確に示している場合、 キュー・マネージャーは自動的に共用状況を変更し、データ・セットが FAILED 状態になったことを示します。 ただし、環境上の問題 (例えば、データ・セットを開くことが許可されていない) によってエラーが発生した場合、キュー・マネージャーはエラー・メッセージを出し、 データ・セットを使用不可のものとして扱いますが、 共用データ・セット状況を変更することはありません。 この環境エラーがデータ・セットに関する何らかの問題 (例えば、一部のキュー・マネージャーがアクセスできないデバイスに割り振られた) によることが判明した場合、オペレーターは RESET SMDS コマンドを STATUS(FAILED) とともに使用して、 データ・セットが必要に応じてリカバーまたは修復されるようにすることができます。

共用メッセージ・データ・セットへの接続を確立できないが、 データ・セットは有効であるように見える場合、所有キュー・マネージャーについて **START SMDSCONN** コマンドを出すことによって、そのデータ・セットを使用する新しい試みを起動できます。

キュー・マネージャーとデータ・セットの間の接続を一時的に終了する操作上の必要があるが、 データ・セット自体は損傷を受けていない場合、 **STOP SMDSCONN** コマンドを使用してデータ・セットを閉じて割り振り解除することができます。 データ・セットが使用中である場合、キュー・マネージャーはそれを通常どおりに閉じます (ただし、そのデータ・セット内のデータの要求は戻りコードとともに拒否されます)。それが所有されるデータ・セットである場合、キュー・マネージャーは CLOSE 処理の間にスペース・マップを保存し、 再始動処理の必要を回避します。

データ・セットを (例えば、移動するために) すべてのキュー・マネージャーから一時的にサービス休止にする必要があるが、損傷していない場合は、オプション CMDSCOPE (*) を指定した関連データ・セットに対して **STOP SMDSCONN** を使用して、そのデータ・セットを最初に使用するキュー・マネージャーを停止することをお勧めします。これにより、データ・セットがサービス開始時に再始動する必要がなくなります。 対照的に、データ・セットに FAILED のマークが付いている場合、 これはキュー・マネージャーが即座にその使用を停止しなければならないことを示します。 つまり、スペース・マップは保存されず、再始動処理によって再作成する必要があります。

キュー・マネージャーが再始動されると、 以前に ACCESS(SUSPENDED) 状態にあった共用メッセージ・データ・セットへのアクセスが再試行されます。

共用メッセージ・データ・セットのリカバリー・ログ

持続共用メッセージは、 メディア・リカバリーの目的でログに記録されます。 つまり、カップリング・ファシリティー構造や共用メッセージ・データ・セットの障害の後、 リカバリー・ログが損なわれていなければ、これらのメッセージをリカバリーすることができます。 持続メッセージは、災害復旧の目的で、別のサイトでリカバリー・ログから再作成することもできます。

メッセージ・データが共用メッセージ・データ・セットに書き込まれるとき、 データ・セットに書き込まれる各ブロックは、カップリング・ファシリティーに書き込まれたように、 別個にログに記録され、その後にメッセージ項目 (データ・マップを含む) が続きます。 リカバリー・プロセスは常にカップリング・ファシリティー構造体をリカバリーしますが、個々の共用メッセージ・データ・セットをリカバリーする必要はありません。ただし、データ・セットの状況が FAILED の場合や、状況が ACTIVE でもデータ・セット・ヘッダー・レコードがもう有効でない場合 (データ・セットが再作成されたことを示す) はリカバリーされます。 データ・セットの状況が ACTIVE でデータ・セット・ヘッダーが引き続き有効な場合や、状況が EMPTY の場合 (障害発生時に保管されていたメッセージがないことを示す)、リカバリー時にそのデータ・セットは選択されません。

共用メッセージ・データ・セットのバックアップ

BACKUP CFSTRUCT が使用されてアプリケーション構造内の共用メッセージのバックアップが作成されるとき、 以前に DB に保管された持続共用メッセージの場合のように、 共用メッセージ・データ・セット内に保管されている持続メッセージのデータが同時にバックアップされます。

共用メッセージ・データ・セットのリカバリー

共用メッセージ・データ・セットは、破損したか失われた場合、 FAILED 状態になる必要があり、これによってキュー・マネージャーは修復が完了するまでその使用を停止します。 これは通常、自動的に実行されますが、 **RESET SMDS** コマンドを STATUS(FAILED) とともに使用して行うこともできます。

共用メッセージ・データ・セットに持続メッセージが含まれていた場合は、RECOVER CFSTRUCT コマンドを使用してリカバリーできます。 このコマンドはまず、最も直近の BACKUP CFSTRUCT コマンドから共用メッセージ・データ・セット用の持続メッセージ・データを復元した後、 そのとき以降にログに記録されたすべての変更を適用します。 データ・セットが最初に活動化された時点以降に **BACKUP CFSTRUCT** コマンドが実行されていない場合、データ・セットは空にリセットされ、活動化以降のすべての変更が適用されます。

CFSTRUCT の内容とすべての共用メッセージ・データ・セットが使用できない場合 (例えば、災害時回復状態の場合)、それらはすべて単一の **RECOVER CFSTRUCT** コマンドで回復することができます。

共用メッセージ・データ・セットが損傷を受けているが、 CFSTRUCT についてリカバリーがアクティブになっていなかった場合や、最新の BACKUP CFSTRUCT が入っているログが入手不可または使用不可である場合、 そのデータ・セットにオフロードされたメッセージはリカバリーすることができません。 このケースでは、 **RECOVER CFSTRUCT** コマンドを パラメーター TYPE(PURGE) とともに使用して、共用メッセージ・データ・セットに空のマークを付け、 そのデータ・セットにデータが保管されたすべてのメッセージを構造から削除します。

RECOVER CFSTRUCT コマンドが発行されると、共用メッセージ・データ・セットの状況が FAILED から INRECOVER に変更されます。 リカバリーが正常に完了した場合、状況は自動的に RECOVERED に変更され、そうでない場合は FAILED に戻ります。

データ・セットが RECOVERED 状態に変更されると、 これは所有キュー・マネージャーに、データ・セットを開いて再始動処理を実行することを試みることが可能になったことを伝えます。

共用メッセージ・データ・セットのリカバリーと同期点

共用メッセージ・データ・セットのリカバリー・プロセスは、 ログの終わりまでにあるすべての完全なログ・レコードについての変更を、同期点にかかわらず再適用します。

変更が同期点の内部で行われた場合、 CFSTRUCT の再始動またはリカバリーの処理を行うと、コミットされていない要求がバックアウトされ、 リカバリーされた変更の一部が実際には使用されないことがありますが、 リカバリーすること自体に害はありません。

コミットされていない MQPUT メッセージが構造に書き込まれたが、 対応するデータがデータ・セットやログに書き込まれていないという可能性もあります (入出力の完了は 同期点の処理の開始時にのみ強制されるため)。 再始動処理によって構造内のメッセージ項目はバックアウトされるため、 これは無害であり、リカバリーされていないデータを参照するという事実も問題になりません。

共用メッセージ・データ・セットの再始動処理

CFSTRUCT へのキュー・マネージャー接続が正常に終了すると、 キュー・マネージャーはそれぞれの共用メッセージ・データ・セット用の空きブロック・スペース・マップを、 そのデータ・セットが閉じられる直前にチェックポイント域に書き出します。 CFSTRUCT または共用メッセージ・データ・セットが次の再始動の前にリカバリー処理を必要としていない限り、 このスペース・マップは、接続の再始動時に再び読み込むことができます。

ただし、キュー・マネージャーが異常終了した場合や、 構造またはデータ・セットがリカバリー処理を必要とする場合は、 構造へのキュー・マネージャー接続が再始動されたときにスペース・マップを動的に再作成するための追加処理が必要です。

データ・セット自体をリカバリーする必要がない場合、キュー・マネージャーの再始動では、 単に構造の現在の内容をスキャンして、現在のキュー・マネージャーが所有するメッセージ・データへの参照を見つける処理が行われます。 また、関連するデータ・ブロックに、スペース・マップで所有されていることを示すマークが付けられます。 他のキュー・マネージャーは、スペース・マップが再作成されている間、引き続きその構造を使用し、再始動するキュー・マネージャーが所有するデータを読み込むことができます。

共用メッセージ・データ・セットのリカバリー後の再始動

共用メッセージ・データ・セットをバックアップからリカバリーしなければならない場合、 そのデータ・セットに保管されていたすべての非持続メッセージは消失し、 データ・セットが TYPE(PURGE) を使用してリカバリーされた場合は、 データ・セットに保管されていたすべてのメッセージが消失します。 リカバリーが完了するまで、 データ・セットには FAILED または INRECOVER のマークが付けられるため、 影響を受けるメッセージのいずれかを別のキュー・マネージャーから読み取ろうと試みると、 そのデータ・セットが一時的に使用不可であることを示すエラー・コードが返されます。

データ・セットがリカバリーされると、 状況は RECOVERED に変更され、 所有キュー・マネージャーが再始動処理でそれを開くことができるようになります。 他のキュー・マネージャーに対してはそのデータ・セットは使用不可のままです。 キュー・マネージャーの再始動では、 スペース・マップを再作成するため、残っているメッセージがないかどうか構造がスキャンされます。 このスキャンでは、データが消失したメッセージがないかどうかも確認され、存在する場合は構造から削除されます (必要な場合は、消失のフラグを立てて 後で削除することもできます)。

データ・セットの状況は、この再始動スキャンが完了したときに自動的に RECOVERED から ACTIVE に変更され、 その時点で他のキュー・マネージャーは再びこのデータ・セットを使用できるようになります。

共用メッセージ・データ・セットの使用状況の情報

DISPLAY USAGE コマンドでは今後、現在開いている共用メッセージ・データ・セットについて、 共用メッセージ・データ・セットのスペースおよびバッファー・プールの使用状況に関する情報も 表示されます。 この情報は、新規のオプション TYPE(SMDS) または既存のオプション TYPE(ALL) が指定されている場合に表示されます。

共用メッセージ・データのパフォーマンスと容量に関する考慮事項

データ・セットの使用量のモニター

DISPLAY USAGE コマンドに **TYPE(SMDS)** オプションを指定すると、所有している共用メッセージ・データ・セットごとに現在の満杯率を表示できます。

SMDS 定義に対してオプション **DSEXPAND(YES)** が有効になっている場合、キュー・マネージャーは通常、共有メッセージ・データ・セットが 90% に達すると自動的に拡張します。 これは、SMDS オプションが **DSEXPAND(YES)** に設定されているか、SMDS オプションが **DSEXPAND(DEFAULT)** に設定されており、CFSTRUCT デフォルト・オプションが **DSEXPAND(YES)** に設定されている場合に適用されます。

データ・セットの作成時に 2 次割り振りサイズが指定されていなかったために、拡張しようとして失敗した場合 (メッセージ IEC070I と理由コード 203 が示される) は、キュー・マネージャーは現行サイズの約 20% のオーバーライド 2 次割り振りを使用して拡張要求を繰り返します。

データ・セットを拡張する際には、拡張処理の一部として新しいデータ・セット・エクステントがフォーマットされます。この処理には数十秒かかることがあり、エクステントが非常に大きい場合は数分かかることさえあります。 新しいスペースはフォーマットの完了後に使用できるようになり、カタログが更新されて、使用頻度の高い新規制御インターバルを示します。

新しいメッセージが立て続けに作成される場合、拡張処理が完了する前に既存のデータ・セットが満杯になる可能性があります。 この場合、スペースを割り振れなかった要求は、拡張の試行が完了して新しいスペースを使用できるようになるまで一時的に中断されます。 拡張が正常に処理されると、要求が自動的に再試行されます。

使用可能なスペースが不足しているか、最大エクステントに既に達しているために拡張の試行が失敗した場合は、失敗の理由を示すメッセージが発行され、影響を受けた SMDS のオーバーライド・オプションが自動的に **DSEXPAND(NO)** に変更されて、それ以上拡張が試行されないようにします。 この場合、データ・セットが満杯の可能性があるというリスクがあり、データ・セットが満杯になるで説明されている追加のアクションが必要になる可能性があります。

アプリケーション構造の使用量のモニター

アプリケーション構造の使用レベルは、アプリケーション構造の絶対パス名 (キュー共有グループ接頭部を含む) を指定した MVS **DISPLAY XCF,STRUCTURE** コマンドを使用して表示できます。 IXC360I 応答メッセージは、エレメントとエントリーの現在の使用量を示します。

構造体の使用量が CFRM ポリシーで指定されている **FULLTHRESHOLD** 値を超えると、システムはメッセージ IXC585E を出し、**ALTER** アクションが指定されている場合は自動的に実行します。このアクションは、エントリーとエレメントの比率を変更するか、構造体のサイズを大きくします。

バッファー・プール・サイズの最適化

共用バッファー・プール内の各バッファーは、最大で論理ブロック・サイズまでの、1 つのメッセージの連続したページ範囲の読み取りや書き込みに使用されます。 メッセージが他のブロックにあふれ出す場合は、その別のブロックのページ範囲ごとに別のバッファーが必要になります。

読み書き操作後のメッセージ・データを含むバッファーはストレージ内に保持され、最長未使用時間 (LRU) キャッシュ・スキームを使用して再利用されるので、少し後に同じデータを再度読み取る要求があっても、ディスクにアクセスする必要はありません。 したがって、共用メッセージが書き込まれた直後に、同じシステム上で実行しているアプリケーションがそのメッセージを再読み取りする際に、大幅な最適化が実現します。 別のキュー・マネージャーが所有するメッセージが選択のためにブラウズされてから取り出される際にも、メッセージをディスクから再読み取りする必要がなくなります。

したがって、アプリケーション構造ごとに必要なバッファーの数は、そのアプリケーション構造に関する大規模メッセージの読み書きを行う並行 API 要求ごとに 1 つと、最近アクセスされたデータを保存して以後の読み取りアクセスを最適化するために使用されるいくつかの追加バッファー数になります。

共用バッファー・プールの場合、バッファーが足りないと、バッファーが即時に使用可能にならなければ API 要求は単に待機します。 しかし、この状態はパフォーマンスが大幅に低下する可能性があるので避ける必要があります。

共用バッファー・プールに対する **DISPLAY USAGE** コマンドからの統計は、現在の統計間隔内にバッファー待機があったかどうかを示します。また、フリー・バッファーの最小数 (または、任意の時点でバッファーを待機したスレッドの最大数を示す負の値)、データを保管したバッファーの数、およびバッファー要求が正常に保管したデータの割合 (%) も示します。 "LRU ヒット数" 読む必要はありません ("LRU ミス数")[1].

- 待機が発生している場合は、バッファーの数を増やす必要があります。
- 未使用のバッファーが多数ある場合は、バッファーの数を減らして、領域内で他の目的に使用できるストレージを増やすこともできます。
- 保存データを含むバッファーが多数あるが、保存データにヒットする読み取りの比率が非常に小さい場合、ストレージを他の目的に使用する方が望ましければ、バッファーの数を減らすことができます。 しかし、待機が発生する可能性があるので、バッファーの数を空きバッファーの最低数より多く減らさないでください。また可能であれば、空きバッファーの最低数が通常ゼロより十分大きくなるような大きさにする必要があります。

共用メッセージ・データ・セットの削除

DELETE CFSTRUCT コマンド (構造体内のすべての共用キューが空でクローズされている場合にのみ許可されます) は、共用メッセージ・データ・セット自体を削除しませんが、このコマンドの完了後に通常の方法で削除することができます。 同じデータ・セットを共用メッセージ・データ・セットとして再使用する場合は、 まず再フォーマットして空の状態にリセットする必要があります。

共用メッセージ・データ・セットの例外状態

ソフトウェアやハードウェアのエラーがなくても、通常の使用中に発生する可能性のある例外状態が複数あります。

[1] (Hits / (Hits+Misses))* 100

データ・セットが満杯になる

データ・セットが満杯になって拡張できない場合や、拡張しようとして失敗した場合は、対応するキュー・マネージャーを使用して対応するアプリケーション構造に大規模メッセージを書き込むアプリケーションは、エラー 2192 の MQRC_STORAGE_MEDIUM_FULL (MQRC_PAGESET_FULL でもある) を受け取ります。

データ・セットが満杯になる原因としては、データを処理することになっていたアプリケーションで障害が発生し、メッセージのバックログが大量に累積した可能性があります。 この場合は、データ・セットをさらに拡張しても一時的な解決策にしかならず、可能な限り早期に処理アプリケーションが再び正しく機能するようにすることが重要です。

使用可能なスペースを増やせる場合は、**ALTER SMDS** コマンドを使用し、**DSEXPAND(YES)** または **DSEXPAND(DEFAULT)** (CFSTRUCT 定義で YES が設定されているか **DSEXPAND** のデフォルトになっていることが前提) を設定して、再試行を起動できます。 ただし、失敗の理由が最大エクステントに達した場合は、新しい拡張の試みはメッセージで拒否され、**DSEXPAND(NO)**が再度設定されます。 この場合、さらに拡張するには再割り振りするしかなく、この場合には次のように一時的に使用不可にすることが含まれます。

データ・セットを移動または再割り振りする必要がある

データ・セットを移動または拡張する必要があるが他の点では通常通り使用できる場合は、一時的に使用できないようにして、移動や再割り振りを可能にすることができます。 データ・セットが使用不可になっている間にそのデータ・セットを使用しようとする API 要求は、理由コード MQRC_DATA_SET_NOT_AVAILABLE を受け取ります。

1. **RESET SMDS** コマンドを使用して、データ・セットに **ACCESS(DISABLED)** のマークを付けます。これにより、現在接続しているすべてのキュー・マネージャーはこのデータ・セットを正常に閉じて割り振り解除します。
2. 必要に応じてデータ・セットの移動または再割り振りを行い、新しく割り振ったデータ・セットに古い内容をコピーします。例えば、アクセス方式サービス (AMS) の **REPRO** コマンドを使用します。

   古いデータをコピーする前に、新しいデータ・セットを事前フォーマットしようとしないでください。事前フォーマットすると、コピーするデータはフォーマットされたデータ・セットの末尾に付加されることになります。
3. **RESET SMDS** コマンドを使用して、再度データ・セットに **ACCESS(ENABLED)** のマークを付け、使用中状態に戻します。

   古い内容が新しいデータ・セットのサイズより小さい場合、新しいデータ・セットを開くときに、残りのスペースは自動的に事前フォーマットされます。

   古い内容が新しいデータ・セットのサイズより大きい場合は、キュー・マネージャーはカップリング・ファシリティー構造体内のメッセージをスキャンし、スペース・マップを再ビルドして、アクティブ・データが失われていないことを確認する必要があります。 新しいエクステント外のデータ・ブロックへの参照が見つかった場合は、データ・セットに **STATUS(FAILED)** のマークが付けられるので、そのデータ・セットを正しいサイズのデータ・セットに置き換えてから、古いデータ・セットを再度コピーするか、**RECOVER CFSTRUCT** を使用して持続メッセージをリカバリーして、修復しなければなりません。

カップリング・ファシリティー構造体のスペースが少ない

カップリング・ファシリティー構造体がスペースを使い尽くしており、メッセージ IXC585E が出ている場合は、このような場合に最大量のデータがオフロードされるようにオフロード規則が設定されているかどうかを確認できます。 設定されていない場合は、**ALTER CFSTRUCT** コマンドを使用してオフロード規則を変更できます。

共用メッセージ・データ・セットのエラー状態

エラーがある場合のみ発生し、通常操作状態では発生しないことが判明している問題が複数あります。

所有データ・セットを開くことができない

共有メッセージ・データ・セットを所有するキュー・マネージャーがそれを割り振ることもオープンすることもできない場合、またはデータ・セット属性がサポートされていない場合、キュー・マネージャーは **ALLOCFAIL** または **OPENFAIL** の適切な **SMDSCONN** 状況値を設定し、**SMDSCONN** 可用性を **AVAIL(ERROR)** に設定します。 また、SMDS の可用性を **ACCESS(SUSPENDED)** に設定します。 エラーが訂正されたら、 **RESET SMDS** コマンドを使用して **ACCESS(ENABLED)** を設定し、再試行をトリガーするか、所有するキュー・マネージャーに対して **START SMDSCONN** コマンドを発行します。

読み取り専用データ・セットを開くことができない

キュー・マネージャーが、別のキュー・マネージャーによって所有されている共有メッセージ・データ・セットを割り振ることもオープンすることもできず、**STATUS(ACTIVE)** としてマークされている場合は、データ・セット自体の問題ではなく、データ・セット (**SMDSCONN** オブジェクトによって表される) への接続に関する特定の問題が原因であると想定されます。

これは、**SMDSCONN** を **STATUS(ALLOCFAIL)** または **STATUS(OPENFAIL)** として適宜マークし、**SMDSCONN** 可用性に AVAIL (ERROR) のマークを付けて、それ以上使用しようとしないようにします。

データ・セット自体の状況に影響を与えずに問題を訂正できた場合は、**START SMDSCONN** コマンドを使用して再試行を起動します。

問題がデータ・セット自体の問題であることが判明した場合は、 **RESET SMDS** コマンドを使用して、リカバリーされるまでデータ・セットに **STATUS(FAILED)** のマークを付けることができます。 データ・セットがリカバリーされると、状況を **STATUS(ACTIVE)** に戻すアクションによって、他のキュー・マネージャーに通知されます。 **SMDSCONN** に **AVAIL(ERROR)** のマークが付けられている場合、自動的に **AVAIL(NORMAL)** に戻され、データ・セットを開くための新しい試行がトリガーされます。

データ・セット・ヘッダーが壊れている

データ・セットが正常にオープンされたが、ヘッダー情報の形式が正しくない場合、キュー・マネージャーはデータ・セットをクローズして割り振り解除し、状況セットを **STATUS(FAILED)** に設定し、可用性を **ACCESS(SUSPENDED)** に設定します。 こうすると、**RECOVER CFSTRUCT** を使用して内容をリカバリーできます。

データ・セットが別の使用での残留データを含んでいて、それ以降に事前フォーマットされていないためにエラーが発生した場合は、データ・セットを事前フォーマットしてから、**RESET SMDS** コマンドを使用して状況を **STATUS(RECOVERED)** に変更します。

それ以外の場合は、データ・セットをリカバリーする必要があります。

データ・セットが予想外に空になっている

キュー・マネージャーが **STATUS(ACTIVE)** としてマークされているデータ・セットを開いたが、そのデータ・セットが初期化されていないか、または新しく事前フォーマットされているが有効であることが検出された場合、キュー・マネージャーは共有メッセージ・データ・セットを閉じて割り振り解除し、状況を **STATUS(FAILED)** に設定し、可用性を **ACCESS(SUSPENDED)** に設定します。

データ・セットに永続入出力エラーがある

OPEN 処理が正常に行われた後に、データ・セットに永続入出力エラーがある場合は、おそらくリカバリーが必要です。 キュー・マネージャーは、現在接続されているすべてのキュー・マネージャーがデータ・セットを閉じて割り振り解除できるように、データ・セットに **STATUS(FAILED)** のマークを付けます。

データ・セットにリカバリー可能な入出力エラーがある

データ・セットに関するハードウェア障害がある場合は、リカバリー可能な入出力エラーになる可能性があります。この種のエラーはキュー・マネージャーに通知されませんが、パフォーマンスを大幅に低下させ、近い将来に永続入出力エラーになるリスクがあることを示します。

この場合、**RESET SMDS** コマンドを使用してデータ・セットに **STATUS(FAILED)** のマークを付けることにより、データ・セットをリカバリーのためにオフラインにすることができます。 すべてのキュー・

マネージャーによってこのデータ・セットが閉じられて割り振り解除されるので、例えば新しいボリュームに移動してから再び使用可能にすることができます。

この方法でデータ・セットを使用不可にするとスペース・マップが保存されないので、データ・セットを再び使用可能にするには、その前にキュー・マネージャー接続の再始動処理で、カップリング・ファシリティー構造体をスキャンしてデータ・セット内のメッセージを見つけ、スペース・マップを再ビルドする必要があります。 あるいは、共用メッセージ・データ・セットが引き続き使用可能な場合は、再び使用できる準備ができるまで、**RESET SMDS** コマンドを使用して当該データ・セットに **ACCESS(DISABLED)** のマークを付けることにより、丁寧な方法で使用不可にすることができます。

データ・セットの内容が正しくない

データ・セットが組み込まれているボリュームをバックアップから復元する必要があったなどの理由で、そのデータ・セットに正しくないデータが含まれていたりデータ・セットが最新でなかったりすることを、キュー・マネージャーは直接検出できません。 しかし、キュー・マネージャーは整合性検査を実行することにより、この種のエラーのためにアプリケーション・プログラムが正しくないメッセージ・データを見ることがほとんどないようにします。

整合性検査の目的で、データ・セット内の各メッセージ・ブロックには、固有のタイム・スタンプを含む、対応するカップリング・ファシリティー・エントリー ID のコピーが接頭部として付けられます。この接頭部は、メッセージ・データがユーザー・プログラムに渡される前に、メッセージ・ブロックが読み取られるたびに検査されます。 メッセージ・ブロックの接頭部がエントリー ID と一致しない (およびこの間にカップリング・ファシリティー・エントリーが削除されなかった) 場合は、メッセージ・ブロックは損傷しており使用できないと見なされます。

損傷したのが持続メッセージだった場合は、データ・セットには **STATUS(FAILED)** のマークが付けられ、**RECOVER CFSTRUCT** コマンドを使用して構造体の内容をリカバリーしなければなりません。 損傷したのが非持続メッセージだった場合は、リカバリーする方法はないので、診断メッセージが出され、対応するカップリング・ファシリティー・メッセージ・エントリーが削除されます。

データ・セットが開かれる際に使用可能な保存済みのスペース・マップがない場合、データ・セット内のデータの参照用にカップリング・ファシリティー構造体をスキャンすることにより再ビルドされます。 このスキャン中に、キュー・マネージャーはいくつかの処置を行います。

1. キュー・マネージャーは、データ・セット内に現在残っているメッセージのうち最新のもの (存在する場合) の場所を判別します。
2. キュー・マネージャーは、その後データ・セットからそのメッセージを読み取って、ブロック接頭語がメッセージ・エントリー ID と一致していることを確認します。

これらの処置により、キュー・マネージャーはデータ・セットがダウン・レベルになっているケースを確実に検出し、データ・セットに FAILED のマークを付けます。 しかし、以前のコピーからデータ・セットが復元され、それ以降新しいメッセージが追加されていない場合や、このコピー以降に追加されたすべてのメッセージが読み取られて削除された場合は、この検査では見逃されます。

データ・セットが通常どおり閉じられた場合にダウン・レベルのデータに対する保護を行うために、キュー・マネージャーはいくつかの処置を行います。

1. キュー・マネージャーはデータ・セットが通常どおり閉じられる際に Db2 内の SMDS オブジェクト内にスペース・マップのタイム・スタンプのコピーを保存します。
2. キュー・マネージャーはその後、データ・セットが再び開かれる際に、スペース・マップのタイム・スタンプが同じであることを確認します。

タイム・スタンプが一致しない場合は、データ・セットのダウン・レベルのコピーが使用されている可能性があることを示しているので、キュー・マネージャーは既存のスペース・マップを無視して再ビルドします。この作業はメッセージ・データが実際に失われていなかった場合のみ正常実行されます。

注 : このような整合性検査により、理論上発生する可能性のあるすべてのケースでダウン・レベルのデータ・セットや損傷のあるデータ・セットが検出されることが保証されるわけではありません。 例えば、メッセージ・ブロックの先頭は有効なものの、残りのデータの一部が上書きされている場合は検出されません。

共用メッセージ・データ・セットのリカバリーのシナリオ

このセクションでは、共用メッセージ・データ・セットのリカバリーのシナリオについて説明します。

データが失われていない場合のデータ・セットのリカバリー

実際にリカバリーを行わなくても、障害が発生したデータ・セットの正しい内容を復元できる場合があります。 一例として、データ・セットに以前の使用時の残留データが含まれていて、再び事前フォーマットされていない場合は、事前フォーマットすると修正できます。 別の例としては、データ・セットを移動した際に、データのコピー・プロセス中にエラーが発生した場合は、再度データを正しくコピーすると修正できます。

このような場合は、 **RESET SMDS** コマンドを使用して **STATUS(RECOVERED)** を設定することにより、訂正したデータ・セットを再び使用可能にすることができます。 現在の可用性が **ACCESS(SUSPENDED)** の場合、これは自動的に **ACCESS(ENABLED)** に戻ります。

所有キュー・マネージャーは、データ・セットがリカバリーされたことを通知されると、構造体の内容をスキャンしてスペース・マップを再構成し、状況を **STATUS(ACTIVE)** に変更します。 その後、他のキュー・マネージャーがデータ・セットの読み取りを再開できます。

TYPE(NORMAL) を指定したデータ・セットのリカバリー

データ・セットの内容が失われたものの、**RECOVER(YES)** を使用してアプリケーション構造が定義されており、該当するリカバリー・ログが使用可能な場合は、**RECOVER CFSTRUCT** コマンドを使用して、構造体内に格納されている持続メッセージをリカバリーできます。リカバリー対象には、共用メッセージ・データ・セットにオフロードされた持続メッセージ・データも含まれます。 このコマンドは、**BACKUP CFSTRUCT** コマンドによってログに記録された情報と、バックアップ時以降に持続メッセージに加えられ、ログに記録されたすべての変更を使用して、現行の状態を復元します。

RECOVER CFSTRUCT コマンドは、カップリング・ファシリティー構造体内のすべての持続メッセージと、Db2 内に格納されているオフロードされたメッセージ・データを常にリカバリーします。 共用メッセージ・データ・セットに保管されているオフロード・データの場合、各データ・セットがリカバリー処理のために選択されるのは、そのデータ・セットが既に **STATUS(FAILED)** としてマークされている場合、またはリカバリー処理によってオープンされたときに予期せず空または無効であることが検出された場合のみです。 アクティブとしてマークが付けられ、妥当性検査に合格した共用メッセージ・データ・セットは、既存のメッセージ・データが既に正しいのでリカバリーする必要がありませんが、リカバリー後に保存済みのスペース・マップを再ビルドする必要があることを示すようにヘッダーが更新されます。

リカバリー処理を実行できるのは、構造体に失敗としてマークが付けられている場合のみです。この場合は、リカバリー処理により構造体の完全な内容を再構成する必要があるからです。 しかし、1 つ以上の共用メッセージ・データ・セットに失敗としてマークが付けられている場合、**RECOVER CFSTRUCT** コマンドはリカバリー処理を進められるようにするため、必要なら自動的に構造体に失敗としてマークを付けます。

キュー共用グループ内のどのキュー・マネージャーからでも、関係するデータ・セットに対して書き込みアクセス権限が付与されていれば、リカバリーを実行できます。

持続メッセージのみバックアップされてログに記録されるので、通常のリカバリー処理で持続メッセージはすべて復元されますが、構造体内の非持続メッセージは失われます。

リカバリーが完了すると、リカバリーのために選択されたすべてのデータ・セットは自動的に **STATUS(RECOVERED)** に変更され、可用性が **ACCESS(SUSPENDED)** であった場合は **ACCESS(ENABLED)** に変更されます。 キュー・マネージャーは、カップリング・ファシリティー内のメッセージをスキャンして各データ・セットのスペース・マップを再作成し、データ・セットに **STATUS(ACTIVE)** のマークを付けて、再び使用できるようにします。

TYPE(PURGE) を指定したデータ・セットのリカバリー

リカバリー可能な構造体の場合、データ・セットの内容が失われたにもかかわらず、何らかの理由 (例えば、リカバリー・ログを使用できなかったり、リカバリーに時間がかかりすぎたりするなど) でリカバリーできなければ、**TYPE(PURGE)** を指定した **RECOVER CFSTRUCT** コマンドを使用して、構造体を

使用可能状態に戻すことができます。 これにより、構造が空の状態にリセットされ、関連するすべてのデータ・セットに **STATUS(EMPTY)** のマークが付けられます。

アプリケーション構造の削除

リカバリー不能アプリケーション構造体が MVS **SETXCF FORCE** コマンドを使用して削除された場合、または構造体障害の結果として削除された場合は、次に構造体が接続されたときに、メッセージ CSQE028I が発行され、構造体がリセットされ、既存のメッセージがすべて破棄され、既存のデータ・セットも自動的に **STATUS(EMPTY)** にリセットされます。 このアクションにより、リカバリー不能な構造体内か関連付けられているデータ・セット内にあるデータが失われた後に、この構造体を再使用できるようになります。

リカバリー可能なアプリケーション構造が削除された場合は、構造体に障害が発生した場合と同じ方法で扱われます。

データ・セットのリカバリーが失敗する

ログ・データ・セットが使用できなくなったり、リカバリーの実行中にキュー・マネージャーが終了したりするなどの理由で **RECOVER CFSTRUCT** を完了できない場合は、リカバリーの開始だけは行われたデータ・セットにヘッダー内でマークが付けられ、リカバリーが一部試行されたことが示されます。データ・セットは **STATUS(FAILED)** 状態のままになります。

この場合、オプションとして、元のリカバリー要求を繰り返すか、代わりに **TYPE(PURGE)** を使用してリカバリーし、既存のデータを破棄します。

実際のリカバリーを行わずにデータ・セットに **STATUS(RECOVERED)** のマークを付けようとすると、次にデータ・セットを開いたときに、キュー・マネージャーは、ヘッダーに不完全なリカバリーが示されていることを認識し、再び **STATUS(FAILED)** のマークを付けます。

オフサイトの災害復旧

オフサイトの災害復旧の場合、CFSTRUCT 定義や関連付けられている SMDS 状況情報を含む Db2 共用オブジェクトとログのみを使用して、持続共用メッセージを再作成できます。

この定義を含む Db2 表をセットアップした後に、アプリケーション構造と共用メッセージ・データ・セットを空としてセットアップできます。 セットアップ後にキュー・マネージャーが接続して、予想外に空であることを検出すると、失敗としてマークを付けます。その後単一の **RECOVER CFSTRUCT** コマンドを使用して、影響を受けたすべての構造体に関するすべての持続メッセージをリカバリーできます。

z/OS *SMDS* 関連コマンド

このトピックでは、共用メッセージ・データ・セットに関連したコマンドについて説明し、アクセスできるようにしてあります。

大きなメッセージ・オフロード (**OFFLOAD** およびオフロード規則) および共用メッセージ・データ・セット (**DSGROUP, DSBLOCK, DSBUFS, DSEXPAND**)) に関連する **CFSTRUCT** オプションを表示および変更します。

- DISPLAY CFSTRUCT
- DEFINE CFSTRUCT
- ALTER CFSTRUCT
- DELETE CFSTRUCT

以下のコマンドは、大規模メッセージ・オフロード **(OFFLDUSE)** に関連した **CFSTRUCT** 状況を表示します。

- DISPLAY CFSTATUS

以下のコマンドは、個々のキュー・マネージャーに関するデータ・セット・オーバーライド・オプション (**DSEXPAND** および **DSBUFS**) を表示したり変更したりします。

- DISPLAY SMDS

- ALTER SMDS

以下のコマンドは、キュー共用グループ内のデータ・セットの状況や可用性を表示したり変更したりします。

- DISPLAY CFSTATUS TYPE(SMDS)
- RESET SMDS

以下のコマンドは、キュー・マネージャーに関する SMDS のデータ・セット・スペースの使用量とバッファーの使用量の情報を表示します。

- DISPLAY USAGE TYPE(SMDS)

以下のコマンドは、個々のキュー・マネージャーからデータ・セットへの接続 (**SMDSCONN**) の状況や可用性を表示したり変更したりします。

- DISPLAY SMDSCONN
- START SMDSCONN
- STOP SMDSCONN

以下のコマンドは、必要な時点で SMDS 内の大規模メッセージ・データを含む共用メッセージのバックアップとリカバリーを行います。

- BACKUP CFSTRUCT
- RECOVER CFSTRUCT

z/OS 共用キュー使用の利点

共用キューを使用すると、IBM MQ applications の拡張が容易になり、可用性が高くなり、ワークロード・バランシングを実行できるようになります。

共用キューの利点

複製サーバーも 1 つの共用キューから作業を取り出すことになる 共用キュー・アーキテクチャーには、次のように有利な特性があります。

- サーバー・アプリケーションの新しいインスタンスを追加する、または (キュー共用グループにある) キュー・マネージャーの新しい z/OS イメージやアプリケーションのコピーを追加することにより拡張が容易であること。
- 可用性が高いこと。
- キュー共用グループ内の各キュー・マネージャーで利用できる処理能力に応じて、プル・ワークロード・バランシングを実行すること。

高可用性のための共用キューの使用

次の例には、共用キューを使用してアプリケーションの可用性を 向上させる方法が示されています。

ネットワーク内で実行されているクライアント・アプリケーションが、 z/OS 上で実行されているサーバー・アプリケーションの要求を行うという IBM MQ シナリオを考えてみます。 クライアント・アプリケーションは、要求メッセージを作成し、 要求キューに入れます。 その後、クライアントは、要求メッセージのメッセージ・ディスクリプターに指定 された応答先キューに送信される、サーバーからの応答を待機します。

IBM MQ は、クライアント・マシンから z/OS 上のサーバーの入力キューへの要求メッセージの転送、およびサーバーからクライアントへの応答の転送を管理します。 サーバーの入力キューを共用キューとして定義すると、そのキュー に入れられるメッセージがあれば、キュー共用グループの任意のキュー・マネージャーで取り出すことができます。 つまり、シスプレックス内の z/OS イメージごとにキュー・マネージャーを構成できるので、それらすべてを同じキュー共用グループに接続することにより、そのいずれかでサーバーの入力キューのメッセージにアクセスすることができます。

いずれかのキュー・マネージャーが異常終了したり、 管理上の理由で停止させたりしなければならない場合でも、 サーバー側入力キューのメッセージを使用できます。 z/OS イメージ全体をオフラインにしても、メッセージを使用することができます。

この共有キュー上のメッセージの可用性を利用するには、 196 ページの図 60 に示すように、シスプレックス内の各 z/OS イメージ上でサーバー・アプリケーションのインスタンスを実行して、より高いサーバー・アプリケーション容量と可用性を提供します。

サーバー・アプリケーションの 1 つのインスタンスは、共用キューから要求メッセージを取り出し、その内容に応じて処理を実行します。そして、クライアントへ返送する結果を IBM MQ メッセージとして生成します。 この応答メッセージは、要求メッセージのメッセージ・ディスクリプターに指定 された応答先キューおよび応答先キュー・マネージャーに送られます。

戻りパスを構成するために使用できるオプションがいくつかあります。 これらのオプションの詳細については、215 ページの『分散キューイングとキュー共用グループ』を参照してください。

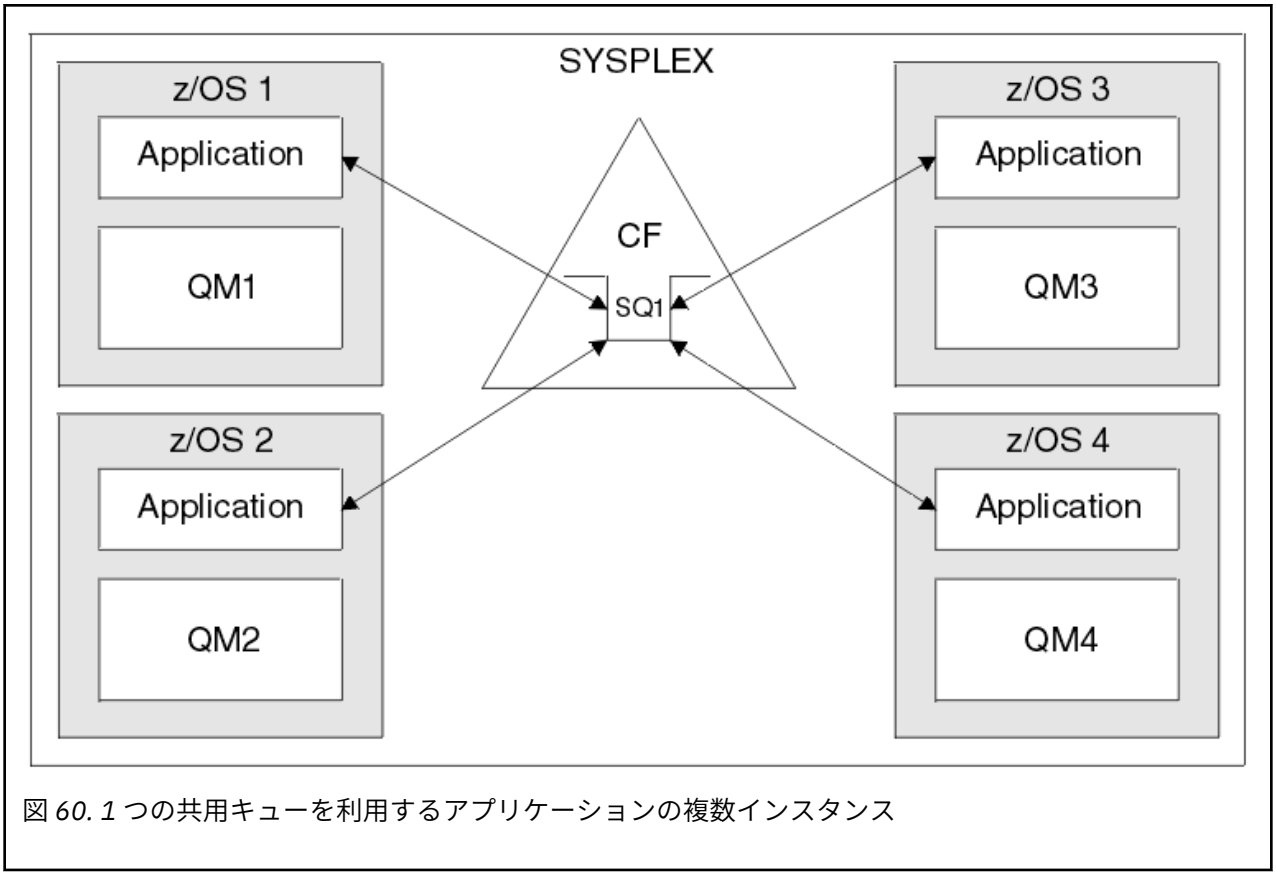


図 *60. 1* つの共用キューを利用するアプリケーションの複数インスタンス

ピア回復

キュー共用グループにあるメッセージの可用性をさらに拡張するために、IBM MQ は、グループ内の他のキュー・マネージャーがカップリング・ファシリティーから異常に切断されていないかどうかを検出し、可能であれば、そのキュー・マネージャーの保留中の作業単位を完了します。 この機能は、ピア回復 と呼ばれます。

アプリケーションがキューから要求メッセージを取り出すと同時にキュー・マネージャーが異常終了したが、 応答メッセージを書き込んでいないか作業単位をコミットしていない状況を考えてください。 キュー共用グループにある別のキュー・マネージャーは、 その障害を検出し、障害のあるキュー・マネージャーで実行していた作業単位をバックアウトします。 すなわち、その要求メッセージは、障害のあるキュー・マネージャーの再始動を待機せずに要求キューに戻されるので、 他のサーバー・インスタンスのいずれかで処理できます。

IBM MQ が自動的に作業単位を解決できない場合、共用部分を手動で解決し、キュー共用グループにある他のキュー・マネージャーにその作業を引き継ぐことができます。

z/OS 共有キューでのストレージ・クラス・メモリーの使用

ストレージ・クラス・メモリー (SCM) を使用することには、IBM MQ for z/OS 共有キューと共に使用する場合に利点があります。

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

z13、zEC12、および zBC12 の各マシンでは、Flash Express カードのインストールが可能です。 これらのカードには、フラッシュ半導体ドライブ (SSD) が組み込まれています。 インストールの後に、カードのフラッシュ・ストレージを 1 つ以上の LPAR (通常は SCM と呼ばれる) に割り振ることができます。

SCM は、入出力の待ち時間とコストという観点から、実ストレージと直接アクセス・ストレージ・デバイス (DASD) との間に位置します。 SCM には可動部分がないので、入出力の待ち時間が DASD よりも大幅に短いという特性があります。

さらに、SCM は実ストレージよりも大幅に安いという特徴もあります。 その結果、比較的低コストで大量のストレージをインストールすることができます。 例えば、一対の Flash Express カードに含まれている使用可能なストレージは 1424 GB です。

これらの特性は、短時間で大量のデータを実ストレージから取り入れる必要があるときに SCM が役立つことを意味しています。 データは、DASD に書き込むよりも短い時間で SCM に書き込むことができるためです。 この最後に述べた特性は、IBM MQ 共有キューを含むカップリング・ファシリティー (CF) リスト構造を使用する際に大変役立つ場合があります。

リスト構造が満杯になる理由

CF 構造体は、定義される際に構造の最大サイズを示す SIZE 属性とともに構成されます。 CF 構造体は実ストレージに常駐しているので、CF 上で定義される構造体の SIZE 属性の合計は CF に割り振られる実ストレージの量よりも小さくなければなりません。

その結果、CF により多くの構造を格納できるように、どの構造の SIZE 値も出来る限り小さい値に維持するというプレッシャーが常時生じることになります。 一方で、目的を達成するために構造を十分に大きくすることは、競合するプレッシャーを生むことになります。 構造があまりに小さいと満杯になってしまって、それを使用するアプリケーションやサブシステムが機能しなくなるためです。

構造を、予期される使用量に基づいて正確にサイズ設定することは重要です。 しかし、作業負荷は時間と共に変化するので、その変動に対応することは簡単ではないため、これは困難な作業となります。

IBM MQ 共有キューは、CF リスト構造を使用してメッセージを保管します。 IBM MQ は、メッセージ構造およびアプリケーション構造が含まれる CF 構造体を呼び出します。

アプリケーション構造は、IBM MQ CFSTRUCT オブジェクトに保管された情報を使用して参照されます。 63 KB より小さいメッセージが共有キューにあるとき、そのメッセージの全体が、単一のリスト・エントリーおよびゼロ個以上のリスト・エレメントとしてアプリケーション構造に保管されます。

IBM MQ 共有キューがリスト構造を使用するので、上記のプレッシャーは共有キューにも影響を与えます。このケースでは、共有キューに保管できるメッセージの最大数は、以下の要素から決まります。

- キュー上のメッセージのサイズ
- 構造の最大サイズ
- 構造で使用可能なエントリーとエレメントの数

最大で 512 の共有キューが同じ構造を使用できるので、エントリーやエレメントの競合が実際に生じて、問題がさらに複雑になることがあります。

IBM MQ キューはアプリケーション間でデータを転送するために使用されるので、アプリケーションがキューにメッセージを入れるときに、 それらのメッセージを取得する必要のあるパートナー・アプリケーションが実行していないという状態が一般的に生じることがあります。

これが起きると、以下の 1 つ以上の状態が生じるまで、キューに含まれるメッセージの数が時間と共に増大します。

- 書き込みアプリケーションがメッセージの書き込みを停止する。
- 取得側のアプリケーションがメッセージの取得を開始する。
- キュー内の既存のメッセージが順次期限切れになり、キューから除去される。
- キューが最大の深さに達する。この場合、MQRC_Q_FULL 理由コードが書き込みアプリケーションに返されます。
- 共有キューを含む構造がその最大サイズに達するか、または構造を含む CF で使用可能なストレージが不足する。 どちらの場合も、MQRC_STORAGE_MEDIUM_FULL 理由コードが書き込みアプリケーションに返されます。

最後の 3 つのケースでは、キューが満杯になっています。 この時点で、メッセージの受け入れ先がないために書き込みアプリケーションで問題が発生します。 書き込みアプリケーションは通常、以下の 1 つ以上の解決策を使用してこの問題を解決します。

- メッセージの書き込みを繰り返し再試行します。オプションで、再試行の間隔を長くします。
- メッセージの書き込み先をデータベースやファイルなど他の場所にします。 後にそのメッセージにアクセスして、通常どおりキューに書き込むことができます。
- 非持続メッセージの場合はそれを破棄します。

ただし、例えば大量の着信メッセージがあるものやファイル・システムにアクセスしないものなど、一部のクラスのアプリケーションでは、上記の解決策は現実的ではありません。 そもそもキューが満杯になることが、まったく、あるいはほとんどないようにする必要があり、これは共有キューの場合に特に当てはまります。

SMDS とオフロード規則

IBM WebSphere MQ 7.1 で導入されたオフロード規則は、アプリケーション構造が満杯になる可能性を小さくする手段となります。

各アプリケーション構造には 3 つの規則が関連付けられていて、それらは以下の 3 対のキーワードを使用して指定されます。

- OFFLD1SZ と OFFLD1TH
- OFFLD2SZ と OFFLD2TH
- OFFLD3SZ と OFFLD3TH

それぞれの規則は、メッセージ・データをアプリケーション構造に関連付けられたストレージ機構にオフロードするために、満たす必要のある条件を指定しています。 現在使用可能なストレージ機構は、以下の 2 つのタイプです。

- Db2
- 仮想記憶アクセス方式 (VSAM) 線形データ・セットのグループ。IBM MQ では、共有メッセージ・データ・セット (SMDS) と呼ばれます。

以下の例は、 DEFINE CFSTRUCT コマンドを使用して LIST1 という名前のアプリケーション構造を作成する MQSC コマンドを示しています。

この構成にはデフォルトのオフロード規則が設定されていて、SMDS がオフロード機構として使用されます。 これにより、構造が 70% 満たされたときには (OFFLD1TH)、32 KB 以上のすべてのメッセージ (OFFLD1SZ) が SMDS にオフロードされます。

同様に、構造が 80% 満たされたときには (OFFLD2TH)、4 KB 以上のすべてのメッセージ (OFFLD2SZ) がオフロードされます。 構造が 90% 満たされたときには (OFFLD3TH)、すべてのメッセージ (OFFLD3SZ) がオフロードされます。

```
DEFINE CFSTRUCT(LIST1)
CFLEVEL(5)
OFFLOAD(SMDS)
```

```
OFFLD1SZ(32K) OFFLD1TH(70)
OFFLD2SZ(4K) OFFLD2TH(80)
OFFLD3SZ(0K) OFFLD3TH(90)
```

オフロードされたメッセージはオフロード・メディアに保管されて、そのメッセージへのポインターが構造内に保管されます。 オフロード規則は、ストレージが不足すると構造に書き込まれるメッセージ・データを減らして構造が満杯になる可能性を小さくしますが、メッセージごとに一部のデータは引き続き構造に書き込まれます。 それは、オフロードされたメッセージへのポインターです。

さらに、オフロード規則にはパフォーマンス・コストが伴います。 メッセージは比較的に短時間で構造に書き込まれるので、上記のコストの大部分は、書き込みの要求を CF に送信するための時間です。 実際の構造への書き込みは高速で、実ストレージの速度と同じです。

SMDS へのメッセージの書き込みは、メッセージ・ポインターの構造への書き込みとメッセージ・データの SMDS への書き込みが含まれるので、より低速になります。 この 2 番目の書き込み操作は DASD の速度で実行され、待ち時間がさらに加わる可能性を含んでいます。 Db2 をオフロード機構として使用する場合には、パフォーマンス・コストがさらに大きくなります。

z/OS ストレージ・クラス・メモリーが *IBM MQ for z/OS* で機能する方法

IBM MQ for z/OS 共用キューでのストレージ・クラス・メモリー (SCM) の使用の概要。

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

CFLEVEL 19 以上のカップリング・ファシリティー (CF) には、SCM を割り振ることができます。 構造体がいっぱいになる可能性 (構造体のフル状態) を減らすため、その CF に定義された構造体を、SCM を利用するように構成することができます。 SCM を使用するように構成された構造体がシステム定義されたポイントを過ぎていっぱいになると、CF はデータを構造体から SCM に移動し始めます。これにより、新しいデータのために構造体にスペースが空けられます。

注 : SCM 自体もいっぱいになる可能性があるため、SCM を構造体に割り振ることは、構造体のフル状態が発生する可能性を低減することにしかなりません。フル状態が発生する可能性は完全にはなくなりません。

その構造体の定義を含むカップリング・ファシリティー・リソース・マネージャー (CFRM) ポリシーに **SCMALGORITHM** と **SCMMAXSIZE** の両方のキーワードを指定することによって、SCM を使用するように構造体を構成します。

これらのキーワードを指定して CFRM ポリシーを適用した後、それらの構成が有効になるように、構造体を再ビルドするか、割り振り解除する必要があります。

SCMALGORITHM キーワード

SCM の入出力速度は実ストレージの速度よりも遅いため、CF は、SCM への書き込みまたは SCM からの読み取りの影響を低減するために、予期される構造体の使用に合わせたアルゴリズムを使用します。

アルゴリズムは、構造体の CFRM ポリシーの **SCMALGORITHM** キーワード (*KEYPRIORITY1* 値を使用) によって構成されます。 *KEYPRIORITY1* 値は、IBM MQ 共用キューによって使用されるリスト構造体でのみ使用する必要があることに注意してください。

KEYPRIORITY1 アルゴリズムは、ほとんどのアプリケーションが共用キューからメッセージを優先順位に従って取得する (つまり、アプリケーションがメッセージを取得するとき、優先順位の最も高い、最も古いメッセージを取得する) ことを想定して機能します。

構造体がシステム定義されたしきい値 (90%) を過ぎていっぱいになり始めると、CF は次に取得される可能性が最も低いメッセージのマイグレーションを非同期的に開始します。 これらは、直近で、キューに書き込まれた優先順位のより低いメッセージです。

構造体から SCM へのメッセージのこの非同期マイグレーションは、「プレステージング」と呼ばれます。

プレステージングによって、SCM との同期的入出力の発生時にアプリケーションがブロックされる可能性が低くなるため、SCM を使用するパフォーマンス・コストが低減されます。

プレステージングに加え、*KEYPRIORITY1* アルゴリズムも、十分な空き領域が使用できるようになると、非同期的に SCM から構造体にメッセージを戻します。 *KEYPRIORITY1* アルゴリズムの場合、これは、構造体が 70% 以下まで下がったことを意味します。

SCM から構造体にメッセージを移動する機能は、「プリフェッチ」と呼ばれます。

プリフェッチは、SCM にプレステージングされているメッセージのうち、CF がメッセージを構造体に非同期的に戻す間待機しなければならなくなるものをアプリケーションが取得しようとする可能性を低減します。

SCMMAXSIZE キーワード

SCMMAXSIZE キーワードは、構造体が使用できる SCM の最大量を定義します。 SCM は必要に応じて CF によって構造体に割り振られるため、使用可能な空 SCM の総量より大きい **SCMMAXSIZE** を指定することができます。 これは「オーバーコミット」と呼ばれます。

重要 : 決して SCM をオーバーコミットしないでください。 オーバーコミットしてしまうと、その SCM に依存するアプリケーションは、それらのアプリケーションが予期する動作が得られなくなります。 例えば、共用キューを使用する IBM MQ アプリケーションは、予期しない MQRC_STORAGE_MEDIUM_FULL 理由コードを受け取る可能性があります。

CF はさまざまなデータ構造体を使用して、SCM の使用を追跡します。 これらのデータ構造体は、CF に割り振られた実ストレージに配置されます。そのため、構造体が使用できる実ストレージの量は減少します。これらのデータ構造体によって使用されるストレージは、「拡張スペース」と呼ばれます。

SCM について構造体を構成する場合、少量の実ストレージ (固定拡張スペースと呼ばれます) を CF から構造体に割り振ります。 構造体が実際には SCM を少しも使用しなかったとしてもこれは割り振られます。構造体からのデータが SCM に格納されると、追加の動的拡張スペースが CF 内のスペアの実ストレージから割り振られます。

データが SCM から除去されると、動的拡張スペースは CF に戻されます。 拡張スペース (固定、動的いずれも) は、構造体に割り振られている実ストレージから取り除かれることはありません。

SCM を使用するように構造体を構成すると、拡張ストレージに加えて、構造体によって使用される制御ストレージの量も増加します。 つまり、SCM について構成されるリスト構造体に格納できる項目とエレメントの量は、SCM が構成されていない同じサイズの構造体よりも少なくなります。

新規または既存の構造体での SCM の影響について把握するには、CFSizer ツールを使用します。

留意すべき最後の重要な点は、動的拡張スペースが使用されていた場合、データを構造体から SCM に移動すると、手動でも自動的にも構造体を変更することはできません。

つまり、構造体に割り振られたストレージの量は、増やしたり、減らしたりできません。構造体によって使用される項目対エレメントの比率は変更できません。 構造体を再び変更可能にするには、SCM に格納されている構造体のデータがないようにし、動的拡張ストレージを使用しないようにする必要があります。

z/OS *SCM* を使用する理由

緊急用ストレージとパフォーマンスの向上は、IBM MQ for z/OS で SCM を使用する 2 つのユース・ケースです。

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

このセクションでは、2 つのシナリオの背後にある理論を紹介します。 これらのシナリオのセットアップ方法の詳細については、以下を参照してください。

- 204 ページの『緊急用ストレージ - 基本構成』
- 210 ページの『パフォーマンスの向上 - 基本構成』

重要 : CF 構造体での SCM の使用は、IBM MQ の特定のバージョンに依存しません。 ただし、緊急用ストレージのシナリオは、SMDS とオフロード規則を必要とするため、IBM WebSphere MQ 7.1 以降でのみ機能します。

緊急用ストレージ

長時間にわたる停止中に MQRC_STORAGE_MEDIUM_FULL 理由コードが IBM MQ アプリケーションに返される可能性を低減させるために、SMDS とメッセージ・オフロードを SCM と組み合わせて使用することができます。

概要

単一の共用キューをアプリケーション構造体で構成します。 書き込み側アプリケーションは、メッセージを共用キューに書き込みます。取得側アプリケーションは、共用キューからメッセージを取得します。

通常の実行中、キュー項目数は 0 に近いことが予想されますが、ビジネス要件には、システムが取得側アプリケーションの 2 時間の停止を許容できる必要があることが示されています。 つまり、共用キューは書き込み側アプリケーションからの 2 時間分のメッセージを格納できる必要があります。

現在、この処理は、デフォルトのオフロード規則と SMDS を使用することによって実現できます。これにより、オフロードに関連するパフォーマンス・コストを削減しつつ、構造体のサイズを最小化できます。

共用キューに送信されるメッセージの割合は、短期間から中期間で 2 倍であることが予想されます。 システムが 2 時間の停止を許容できなければならないという要件は存在しますが、CF には構造体のサイズを 2 倍にするために使用できる十分な実ストレージがありません。

アプリケーション構造体を含む CF は zEC12 マシン上に存在するため、十分な量のメッセージを格納するために必要とされる SCM を構造体に関連付けることができ、2 時間の停止を許容できる可能性があります。

ある一定期間に発生することについて考慮しましょう。

1. 初めに、システムは定常状態にあります。 書き込み側と取得側の両方のアプリケーションが通常どおり実行され、キュー項目数は 0 に近いか、0 になります。 この結果、アプリケーション構造体の大部分は空のままです。
2. ある時に、取得側アプリケーションで予期しない障害が発生し、停止します。 書き込み側アプリケーションはキューにメッセージを書き込み続け、アプリケーション構造体はいっぱいになり始めます。
3. 構造体が 70% まで埋まると、オフロード規則の最初の条件が満たされ、サイズが 32 KB 以上のすべてのメッセージが SMDS にオフロードされます。

   オフロード規則の概要については、198 ページの『SMDS とオフロード規則』を参照してください。
4. メッセージが共用キューに書き込まれ続けると、構造体も埋まり続けます (構造体に格納されているメッセージ・データのため、または構造体に格納されているオフロード・メッセージへのポインターの結果として)。

   構造体が 80% まで埋まると、2 番目のオフロード規則の適用が始まり、4 KB 以上のメッセージが SMDS にオフロードされます。
5. 構造体が 90% を超えて埋まると、すべてのメッセージが SMDS にオフロードされ、メッセージ・ポインターのみが構造体に配置されます。

   この時点で、プレステージング・アルゴリズムの実行が開始され、構造体から SCM へのデータ移動が開始されます。 キューにあるすべてのメッセージの優先順位が同じであるとすると、最新のメッセージがプレステージングされます。

   すべてのメッセージが SMDS にオフロードされているため、SCM に移動されるデータは、実際のメッセージ・データではなく、SMDS 上のメッセージへのポインターになります。

   結果として、構造体とその構造体に関連する SCM および SMDS との組み合わせに格納できるメッセージ数は、非常に大きくなります。

   パフォーマンス : 停止のこの段階では、SMDS に書き込む必要があるため、書き込み側アプリケーションである程度のパフォーマンスの低下が発生する可能性があります。 この場合、パフォーマンスの観点では、SCM の使用は書き込み側アプリケーションに制限を課す要因にはなりません。 SCM は、構造体がいっぱいにならないようにするため、余分のスペースを提供します。

6. 最終的に取得側アプリケーションが再び使用できるようになると、停止状態は解消されます。

   ただし、SCM は構造体によって使用し続けられます。 取得側アプリケーションは、キューのメッセージの読み取りを開始します。その際、古いメッセージから、また、優先順位の高いメッセージから順に取得します。

   構造体がいっぱいになり始める前にこれらのメッセージは書き込まれたため、そのすべてが構造体の実ストレージの部分から出力されます。
7. 構造体が空になり始め、プレステージングをアクティブにするしきい値を下回ると、プレステージングは停止します。
8. オフロード規則が有効になる時点を下回るまで構造体の使用量が下がると、メッセージは、63 KB を超えていない限り SMDS にオフロードされなくなります。

   この時点で、プリフェッチ・アルゴリズムは、SCM から構造体へのデータの移動を開始します。 取得側アプリケーションは SCM のアルゴリズムが予期する順序でメッセージをキューから取得するため、取得側アプリケーションが必要とする前にメッセージは移動されています。

   その結果、取得側アプリケーションは、SCM から同期的に移動されるメッセージを待機する必要はありません。
9. 取得側アプリケーションは、キューの下層に進んでいくに従い、SMDS にオフロードされたメッセージの取得を開始します。
10. 最後に、システムは再び定常状態に戻ります。 SCM や SMDS に格納されているメッセージがなくなると、キュー項目数は 0 に近づきます。

パフォーマンスの向上

このシナリオでは、SMDS の使用に伴うパフォーマンス・コストを掛けずに、SCM を使用して共有キューに格納できるメッセージの数を増やす方法を説明します。

説明

このシナリオでは、書き込み側と取得側のアプリケーションは、アプリケーション構造体に格納される共有キューを使用してやり取りします。

書き込み側アプリケーションは、大量のメッセージを短期間で書き込む際に、一度に大量に実行される傾向があります。 そのため、メッセージをまったく生成しない期間が長く続きます。

取得側アプリケーションは、各メッセージを順番に処理し、複雑な処理をそれぞれのメッセージで実行します。 結果として、書き込み側アプリケーションが実行を開始する時間を除き、ほとんどの時間、キュー項目数は 0 になります。メッセージが取得されるよりも速い速度で書き込まれるようになると、キュー項目数が増え始めます。

書き込み側アプリケーションが停止するまでキュー項目数は増えます。また、取得側アプリケーションは、キューのすべてのメッセージを処理するために時間を十分に取ります。

注 :

1. このシナリオでは、パフォーマンスが重要な要因になります。 キューに送信されるメッセージを常に 63 KB 未満にし、SMDS にオフロードする必要が生じないようにします。
2. アプリケーション構造のサイズは、書き込み側アプリケーションによって単一の「バースト」で配置されるすべてのメッセージを収容するのに十分な大きさに設定されています。
3. 構造体がいっぱいになり始めてもメッセージが SMDS にオフロードされないようにするため、オフロード規則をすべて無効にする必要があります。 これは、SMDS へのメッセージの書き込み、および SMDS からのメッセージの読み取りに関連するパフォーマンス・コストが許容できないと見なされるためです。

時間の経過とともに、書き込み側アプリケーションが一度に大量のメッセージを送信する数は桁違いに増えるに違いありません。 取得側アプリケーションは各メッセージを順番に処理する必要があるため、キューのメッセージ数は構造体がいっぱいになるまで増加します。

この時点で、メッセージを書き込むと、書き込み側アプリケーションは理由コード (MQRC_STORAGE_MEDIUM_FULL) を受け取り、書き込み操作は失敗します。 書き込み側アプリケーショ

ンは、キューにメッセージを書き込むことができない時間をごく短期間しか許容できません。 この期間が長くなると、アプリケーションは終了します。

書き込み側アプリケーションまたは取得側アプリケーションを再作成する時間がない、または利用可能なスキルがないと仮定すると、この問題には次の選択可能な 3 つの解決策があります。

1. アプリケーション構造体のサイズを増やす。
2. キューがいっぱいになり始めたらメッセージを SMDS にオフロードするようアプリケーション構造体にオフロード規則を追加する。
3. SCM を構造体と関連付ける。

最初の解決策は素早く実装できますが、CF では十分な実ストレージを使用できません。

2 番目の解決策も素早く実装することが可能かもしれませんが、SMDS へのオフロードがパフォーマンスに与える影響は、この解決策を使用するには大きすぎると考えられます。

SCM を構造体に関連付けるという、3 番目の解決策では、許容可能なコストとパフォーマンスのバランスが提供されます。

SCM を構造体と関連付けることにより、CF における実ストレージをより効率的に使用できます。これは、取得操作が使用するストレージが大きくなるからです。 ただし、実ストレージの実際の量は、最初の解決策で使用される量より少なくなります。

もう 1 つ考慮すべき点は、SCM のコストです。 ただし、このコストは、実ストレージよりも格段に低く抑えることができます。 これらの要因を合わせると、最初のオプションよりも 3 番目のオプションの方がコストが低くなります。

3 番目の解決策が最初の解決策と同等に機能しない可能性がありますが、CF が使用するプリフェッチ・アルゴリズムとプレステージング・アルゴリズムを組み合わせることによって、パフォーマンスにおける差を許容できる範囲に、場合によっては無視できるほど縮めることができます。

確実に、このパフォーマンスは SMDS を使用してメッセージをオフロードするよりはるかに優れています。

ある一定期間に発生することについて考慮しましょう。

1. 初めに、取得側アプリケーションがアクティブで、共用キューに送信されるメッセージを待機しています。 書き込み側アプリケーションはアクティブではなく、共用キューは空です。
2. しばらくして、書き込み側アプリケーションがアクティブになり、大量のメッセージを共用キューに書き込み始めます。 取得側アプリケーションはメッセージの取得を開始しますが、取得側アプリケーションが書き込み側アプリケーションよりも低速であるため、キュー項目数は急速に増え始めます。

   結果として、アプリケーション構造体はいっぱいになり始めます。
3. 時間が経過しますが、書き込みアプリケーションはまだアクティブです。 アプリケーション構造体は約 90% までいっぱいになります。

   SCM プレステージング・アルゴリズムは構造体から SCM へのメッセージの移動を開始し、構造体のスペースを空けるのはこの時点です。

   取得側アプリケーションは、まずキューから最も古い、優先順位の高いメッセージを取得するため、常に構造体からメッセージを取得しており、SCM から構造体に同期的に移動されるメッセージを待機する必要はありません。
4. 書き込み側アプリケーションはアクティブのままで、メッセージを共用キューに書き込んでいます。 ただし、構造体に収まらないすべてのメッセージを格納するための十分なスペースが SCM に存在するため、アプリケーションは MQRC_STORAGE_MEDIUM_FULL 理由コードを受け取ることはありません。
5. 最終的に、書き込みアプリケーションは、書き込むメッセージがなくなったため、停止します。

   使用中の構造体が 90% 未満に下がったため、プレステージ・アルゴリズムは停止します。取得側アプリケーションは、キューのメッセージを処理し続けます。
6. 取得側アプリケーションが構造体のスペースを空け始めると、プリフェッチ・アルゴリズムが SCM から構造体にメッセージを移動し始めます。

取得側アプリケーションは、プリフェッチ・アルゴリズムが予期する順番でメッセージを処理するため、取得側アプリケーションが SCM から構造体に同期的に移動されるメッセージ・データを待機してブロック状態になることはありません。

7. 最後に、取得側アプリケーションは、共用キューにあるすべてのメッセージを処理し、次のメッセージが使用可能になるまで待機します。 構造体と SCM には、メッセージがなくなります。

z/OS 緊急用ストレージ - 基本構成

IBM MQ で緊急用ストレージの基本的なシナリオをセットアップする方法。

このタスクについて

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

長時間にわたる停止中に MQRC_STORAGE_MEDIUM_FULL 理由コードが IBM MQ アプリケーションに返される可能性を低減させるために、SMDS とメッセージ・オフロードを SCM と組み合わせて使用することができます。

例えば、社内でメッセージをキューに書き込むアプリケーションとキューからメッセージを取得するアプリケーションを使用しているとします。 通常の実行中、キュー項目数は 0 に近いことが予想されますが、ビジネス要件には、メッセージを取得するアプリケーションの 2 時間の停止をシステムが許容できる必要があることが示されています。

つまり、使用される共用キューは書き込み側アプリケーションからの 2 時間分のメッセージを格納できる必要があります。 現在、デフォルトのオフロード規則と SMDS を使用して、これを達成しています。

共用キューへの送信中のメッセージの割合は、短期間から中期間に 2 倍になることが予想されます。 システムが 2 時間の停止を許容できなければならないという要件は存在しますが、CF には構造体のサイズを 2 倍にするために使用できる十分な実ストレージがありません。 アプリケーション構造体を含む CF は zEC12 マシン上に存在するため、十分な量のメッセージを格納するために必要とされる SCM を構造体に関連付けることができ、2 時間の停止を許容できます。

この初期シナリオでは、以下が使用されます。

- キュー共用グループ IBM1。これには、単一のキュー・マネージャー CSQ3 が含まれます。 管理構造体に加え、キュー共用グループは単一のアプリケーション構造体 SCEN1 も定義しています。
- カップリング・ファシリティー (CF) CF01。ここに、SCEN1 アプリケーション構造体が IBM1SCEN1 構造体として格納されます。 この構造体には、1 GB の最大サイズがあります。
- アプリケーション構造体が使用する単一の共用キュー SCEN1.Q。

この構成は、204 ページの図 61 に図示されています。

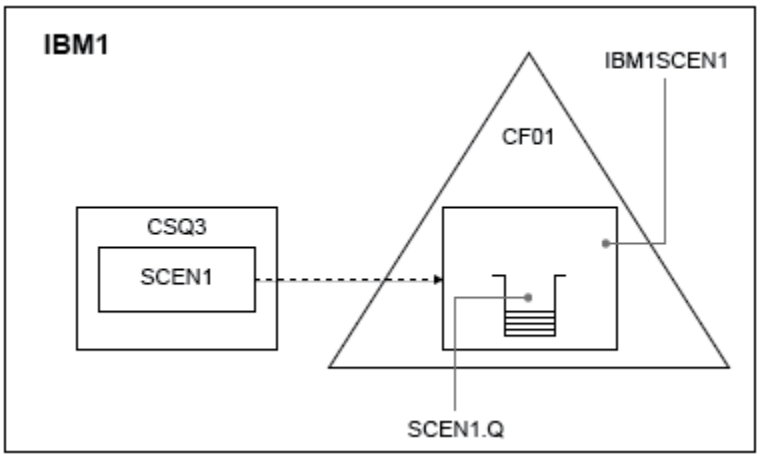


図 61. 基本構成

さらに、キュー・マネージャー CSQ3 が既にキュー共用グループ IBM1 の唯一のメンバーになっていると想定しています。

構造体 IBM1SCEN1 の定義をカップリング・ファシリティー・リソース・マネージャー (CFRM) ポリシーに追加する必要があります。 簡略化するために、構造体は、PREFLIST(CF01) を指定することによって、単一のカップリング・ファシリティー CF01 のみで作成できるように定義します。

重要 : 実動システムで高可用性を実現するには、 IBM MQ によって使用されるすべての構造に対して、少なくとも 2 つの CF を PREFLIST に組み込む必要があります。

手順

1. 次のコマンドを使用して CFRM ポリシーをリフレッシュします。

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN1
```

   構造体 IBM1SCEN1 のサンプル CFRM ポリシー:

```
STRUCTURE
NAME(IBM1SCEN1)
SIZE(1024M)
INITSIZE(512M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

2. 次のコマンドを使用して、構造体が正しく作成されたことを確認します。

```
D XCF,STR,STRNAME=IBM1SCEN1
```

   この時点で、STATUS 行によって構成が割り振られていないことが、キュー共有グループに示されています。

3. CFRM ポリシーで定義した構造体を使用するように、IBM MQ を構成します。

   a. SCEN1 の構造名を指定した DEFINE CFSTRUCT コマンドを使用して、 IBM MQ CFSTRUCT オブジェクトを作成します。

```
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. DISPLAY CFSTRUCT コマンドを使用して構造体を検証します。

   c. 次の MQSC コマンドを使用して、SCEN1 構造体を使用するように SCEN1.Q 共用キューを定義します。

```
DEFINE QLOCAL(SCEN1.Q) QSGDISP(SHARED) CFSTRUCT(SCEN1) MAXDEPTH(999999999)
```

4. IBM MQ Explorer を使用して、1 つのメッセージをキュー SCEN1.Q に書き込み、再びそのメッセージを取り出します。
5. 次のコマンドを発行して、構造体が割り振られていることを確認します。

```
D XCF,STR,STRNAME=IBM1SCEN1
```

コマンドからの出力で、STATUS 行に ALLOCATED が表示されていることを確認します。

タスクの結果

これで基本構成が作成されました。 選択した方法がどのようなものであれ、構成のベースライン・パフォーマンスについて考慮できます。

次のタスク

SMDS と SCM を初期構造体に追加します。

関連概念

197 ページの『共有キューでのストレージ・クラス・メモリーの使用』
ストレージ・クラス・メモリー (SCM) を使用することには、IBM MQ for z/OS 共有キューと共に使用する場合に利点があります。

z/OS *SMDS と SCM の初期構造体への追加*

IBM MQ で緊急用ストレージ用に SMDS と SCM を追加する方法。

このタスクについて

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

タスクのこの部分では、204 ページの『緊急用ストレージ - 基本構成』で説明されている基本構成を使用します。 このシナリオでは、初期構成への共用メッセージ・データ・セット (SMDS) および SCM の追加について説明します。

この最終構成は、206 ページの図 62 に図示されています。

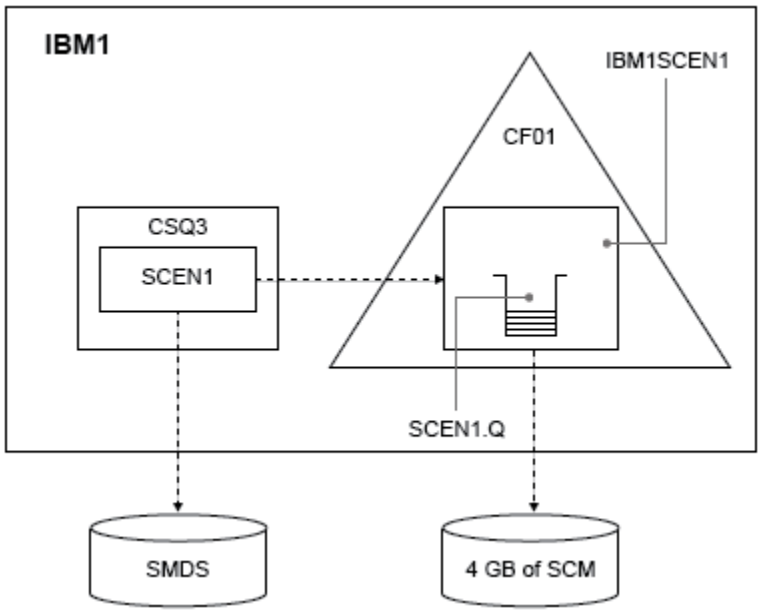


図 62. 緊急用ストレージ用に *SMDS* と *SCM* を追加する構成

手順

1. **CSQ4SMDS** サンプル JCL を以下のように編集して、SCEN1 アプリケーション構造体が使用する SMDS データ・セットを作成します。

```
//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. ALTER CFSTRUCT コマンドを発行して、オフロードに SMDS を使用するように SCEN1 アプリケーション構造体を変更します。また、デフォルトのオフロード規則を実装します。

```
ALTER CFSTRUCT(SCEN1) OFFLOAD(SMDS) OFFLD1SZ(32K) OFFLD2SZ(4K) OFFLD3SZ(0K)
DSGROUP('CSQSMDS.SCEN1.*.SMDS') DSBLOCK(1M)
```

   次の事項に注意してください。

   - SCEN1.Q は SCEN1 アプリケーション構造体の唯一の共用キューであるため、**DSBLOCK** 値は 1M (可能な最大値) に設定されています。 これは、このシナリオで最も効率的な設定になります。
   - 書き込み側アプリケーションによって送信されるメッセージが 30 KB であるため、SMDS へのオフロードは 2 番目のオフロード規則が満たされる (構造体が 80% まで埋まるとき) まで開始されません。

3. テスト・アプリケーションを再度実行します。

   キュー上のメッセージの増大したストレージに注目してください。

4. 次の手順を実行して、4 GB の SCM を構造体 IBM1SCEN1 に追加します。

   a) 次のコマンドを実行して、どのくらいの SCM がインストールされているか、また CF01 に割り振られているかを確認します。

```
D CF,CFNAME=CF01
```

   b) 表示された出力の STORAGE CONFIGURATION セクションの STORAGE-CLASS MEMORY の数値を調べて、使用可能なストレージを確認します。

   c) 以下のように、CFRM ポリシーを SCMMAXSIZE および SCMALGORITHM キーワードで更新します。

```
STRUCTURE
NAME(IBM1SCEN1)
SIZE(1024M)
INITSIZE(512M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
SCMMAXSIZE(4G)
SCMALGORITHM(KEYPRIORITY1)
```

5. 次のコマンドを発行して CFRM ポリシーをアクティブにします。

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=polname
```

6. IBM1SCEN1 構造体を再ビルドします。
   前の変更を実行した際に、構造体が割り振られているため、この手順を実行する必要があります。
   次のコマンドを発行して、構造体を再ビルドします。

```
SETXCF START,REBUILD,STRNM=IBM1SCEN1
```

タスクの結果

これで、SCM は構成に正常に追加されました。

次のタスク

システムのパフォーマンスを最適化します。 詳しくは、208 ページの『ストレージ・クラス・メモリーの使用量の最適化』 を参照してください。

z/OS ストレージ・クラス・メモリーの使用量の最適化

ストレージ・クラス・メモリー (SCM) の使用を向上させる方法。

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

以下のコマンドを実行します。

```
D XCF,STR,STRNAME=IBM1SCEN1
```

以前のテストによって構造体が既にメッセージ・データでいっぱいになった際、再ビルドの一部として、構造体から SCM へのメッセージのプレステージングが含まれていました。 このプロセスは、上記のコマンドを使用することによって開始されました。

このコマンドによって、例えば、次のような出力が生成されます。

```
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
```

このコマンドの出力について以下の点に注目してください。

- STORAGE_CLASS MEMORY は、SCM の 4096 MB の **MAXIMUM** が構造体に追加されたことの確認です。
- プレステージングに使用された STORAGE-CLASS MEMORY の合計を表す ALLOCATED の数字。 SCM が追加される前にはありませんでしたが、現在は構造体に空きスペースがあります。
- SCM 使用量を追跡するために使用された AUGMENTED SPACE の量。
- プレステージング・アルゴリズムが構造体から SCM にデータの移動を開始する時点は、構造体が 90% までいっぱいになったときです。 これは、構成不可の **SCMHIGHTHRESHOLD** プロパティーです。
- プリフェッチ・アルゴリズムが SCM から構造体にデータの移動を開始する時点は、構造体が 70% まで下がったときです。 これは、構成不可の **SCMLOWTHRESHOLD** プロパティーです。

さまざまな方法を試して、SCM の使用を最適化することができます。 次の事項に注意してください。

- メッセージを格納するために SCM が使用された後、SCM からすべてのデータを除去するまで構造体を変更することはできません。

  つまり、この場合、項目対エレメントの比率が、SCM が最初に使用されたときに設定された値で固定されています。 プレステージング・アルゴリズムがデータを SCM に移動し始める前に、構造体が希望する状態にあることを注意深く確認する必要があります。
- SCM を使用する前の現在の構造体のサイズは正しいですか。

  例えば、**INITSIZE** を 512 MB から 1 GB に SIZE を増やしているでしょうか。

  これを行っていない場合、構造体で自動変更を有効にしても、変更を開始する機会が訪れる前に、プレステージング・アルゴリズムが SCM へのデータの移動を開始してしまう可能性があります。 その結果、構造体は実ストレージの 512 MB の使用で固定されてしまいます。
- SCM を使用する前の項目対エレメントの比率は正しいですか。

  このシナリオの目的は、可能な限り多くのメッセージを構造体ストレージで保持しつつ、構造体と SCM に格納できるオフロードされるメッセージ・ポインターの数を全体として増やすことにあります。 これらのメッセージには、SMDS 上のメッセージにアクセスするよりも速くアクセスできます。

  そのため、メッセージを格納するのに適した項目対エレメントの比率で開始し、事前ステージ・アルゴリズムを最初に開始する前にメッセージ・ポインターを格納するのに適した比率に移行する構造体にする必要があります。 この移行は、IBM MQ オフロード規則を使用することによって、部分的に実現できます。

次のコマンドを発行して、オフロード規則を変更します。

```
ALTER CFSTRUCT(SCEN1) OFFLD1SZ(0K)
```

項目対エレメントの比率を最適化するために、いくつかの手順を実行しなければならない場合があります。

以下の表には、緊急用ストレージ・シナリオのさまざまなフェーズでキューに書き込まれるメッセージ数について可能な改善点が示されています。

表 *18.* 緊急用ストレージ・シナリオに関する結果の比較

テストの説明	**メッセージの数**	**キューを埋める時間 (秒)**
基本構成	27,850	3.2
デフォルトのオフロード規則を使用する SMDS	205,000	158
デフォルトのオフロード規則を使用する SCM	828,610	469
調整済みのオフロード規則を使用する SCM	1,135,775	679

表の最後の行には、オフロード規則の調整によって必要な効果が得られたことが示されています。

どの方法でこれらの数値を改善できるかをご使用のシステムで試す必要があります。 例えば、使用可能な SMDS ストレージを使い果たしてしまうかもしれません。 より多くの SMDS ストレージを割り振ることができる場合、キュー上のメッセージ数を大幅に増やすことができます。

z/OS パフォーマンスの向上 - 基本構成

IBM MQ で共用キューを使用してパフォーマンスを向上させる基本的なシナリオをセットアップする方法。

このタスクについて

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

このシナリオでは、SMDS の使用に伴うパフォーマンス・コストを掛けずに、SCM を使用して共有キューに格納できるメッセージの数を増やす方法を説明します。

この初期シナリオは、緊急用ストレージに使用されるシナリオに非常に似ており、以下を使用します。

- キュー共用グループ IBM1。これには、単一のキュー・マネージャー CSQ3 が含まれます。 管理構造体に加え、キュー共用グループは単一のアプリケーション構造体 SCEN2 も定義しています。
- カップリング・ファシリティー (CF) CF01。ここに、SCEN2 アプリケーション構造体が IBM1SCEN2 構造体として格納されます。 この構造体には、2 GB の最大サイズがあります。
- 単一の共用キュー SCEN2.Q。アプリケーション構造体を使用するように構成されます。

この構成は、210 ページの図 63 に図示されています。

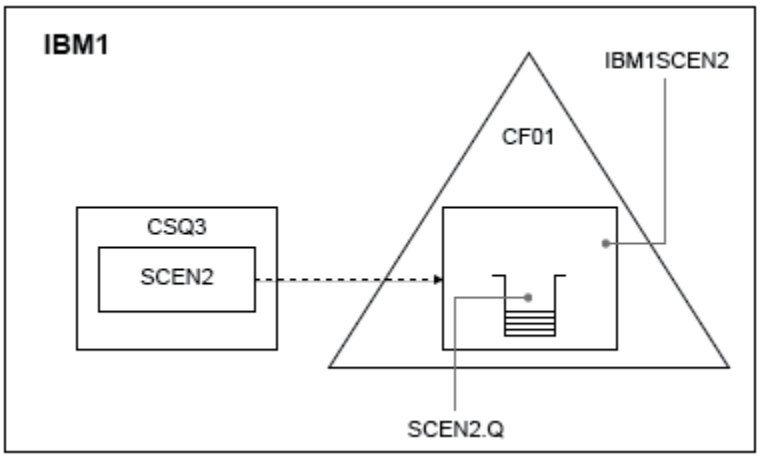


図 *63.* 基本構成

さらに、キュー・マネージャー CSQ3 が既にキュー共用グループ IBM1 の唯一のメンバーになっていると想定しています。

構造体 IBM1SCEN2 の定義をカップリング・ファシリティー・リソース・マネージャー (CFRM) ポリシーに追加する必要があります。 簡略化するために、構造体は、PREFLIST(CF01) を指定することによって、単一のカップリング・ファシリティー CF01 のみで作成できるように定義します。

構造体 IBM1SCEN2 のサンプル CFRM ポリシー:

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
```

```
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

INITSIZE および **SIZE** の両方のキーワードに値 2048M が設定されているため、構造体はサイズ変更できません。

手順

1. 次のコマンドを使用して CFRM ポリシーをリフレッシュします。

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
```

2. 次のコマンドを使用して、構造体が正しく作成されたことを確認します。

```
D XCF,STR,STRNAME=IBM1SCEN2
```

上記のコマンドを発行すると、以下の出力が表示されます。

```
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
```

この時点で、STATUS 行によって構成が割り振られていないことが、キュー共有グループに示されています。

3. CFRM ポリシーで定義した構造体を使用するように、IBM MQ を構成します。
   a. SCEN2 の構造名を指定した DEFINE CFSTRUCT コマンドを使用して、 IBM MQ CFSTRUCT オブジェクトを作成します。

```
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. DISPLAY CFSTRUCT コマンドを使用して構造体を確認します。
   c. 次の MQSC コマンドを使用して、SCEN2 構造体を使用するように SCEN2.Q 共用キューを定義します。

```
DEFINE QLOCAL(SCEN2.Q) QSGDISP(SHARED) CFSTRUCT(SCEN2) MAXDEPTH(999999999)
```

4. IBM MQ エクスプローラーを使用して、1 つのメッセージをキュー SCEN2.Q に書き込み、再びそのメッセージを取り出します。
5. 次のコマンドを発行して、構造体が割り振られていることを確認します。

```
D XCF,STR,STRNAME=IBM1SCEN2
```

コマンドからの出力 (表示されている部分) を確かめ、STATUS 行に ALLOCATED が表示されていることを確認します。

```
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
```

また、SPACE USAGE セクションの以下のフィールドの値にも注目してください。

- ENTRIES
- ELEMENTS
- EMCS
- LOCKS

以下は、値の例です。

```
SPACE USAGE   IN-USE      TOTAL    %
ENTRIES:      344686     345242   99
ELEMENTS:    6548455    6548467   99
EMCS:              2     780318    0
LOCKS:          1024
```

タスクの結果

これで基本構成が作成されました。 選択した方法がどのようなものであれ、構成のベースライン・パフォーマンスについて考慮できます。

次のタスク

基本シナリオをテストする必要があります。 例として、以下の 3 つのアプリケーションを使用できます。これらは、表示されている順序でアプリケーションを開始し、それらを同時に実行できます。

1. PCF アプリケーションを使用して、5 秒おきに SCEN2.Q の現在の項目数 (**CURDEPTH**) の値を要求します。 時間の経過とともにキューの項目数をプロットするために出力を使用できます。
2. 単一スレッドの取得側アプリケーションは、無制限の待機を伴う GET を使用して SCEN2.Q から繰り返しメッセージを取得します。 削除されるメッセージの処理をシミュレートするために、取得側アプリケーションは、削除する 10 メッセージごとに 4 ミリ秒間一時停止します。
3. 単一スレッドの書き込みアプリケーションは、100 万個の 4 KB 非持続メッセージの合計を SCEN2.Q に書き込みます。 このアプリケーションは、書き込む各メッセージの間で一時停止しません。そのため、SCEN2.Q にメッセージを書き込む速さは、取得側アプリケーションが取得できる速さより速くなります。

   結果として、書き込み側アプリケーションが実行されると、SCEN2.Q の項目数は増加します。

構造体 IBM1SCEN2 が失敗し、書き込み側アプリケーションが MQRC_STORAGE_MEDIUM_FULL 理由コードを受け取る場合、書き込み側アプリケーションは次のメッセージのキューへの書き込みを試行する前に 5 秒間スリープします。

一定期間にわたる CURDEPTH アプリケーションの結果をプロットすることができます。 書き込み側アプリケーションが一時停止し、キューが部分的に空になるため、ギザギザの波形をした出力の結果が得られます。

213 ページの『SCM の初期構造体への追加』に移動します。

関連概念

197 ページの『共有キューでのストレージ・クラス・メモリーの使用』

ストレージ・クラス・メモリー (SCM) を使用することには、IBM MQ for z/OS 共有キューと共に使用する場合に利点があります。

z/OS *SCM* の初期構造体への追加

パフォーマンスを向上させるための IBM MQ での SCM の追加方法。

このタスクについて

重要 : IBM z16 は、カップリング・ファシリティー・イメージ用の仮想フラッシュ・メモリー (ストレージ・クラス・メモリー (SCM) とも呼ばれる) の使用をサポートするために、 IBM Z ® の最後の世代になる予定です。 詳しくは、 IBM Z および IBM LinuxONE 4Q 2023 Statements of Direction を参照してください。

代わりに、より大きな構造体を使用するか、SMDS にメッセージをオフロードする必要があります。

タスクのこの部分では、210 ページの『パフォーマンスの向上 - 基本構成』で説明されている基本構成を使用します。 このシナリオでは、初期構成への SCM の追加について説明します。

この最終構成は、213 ページの図 64 に図示されています。

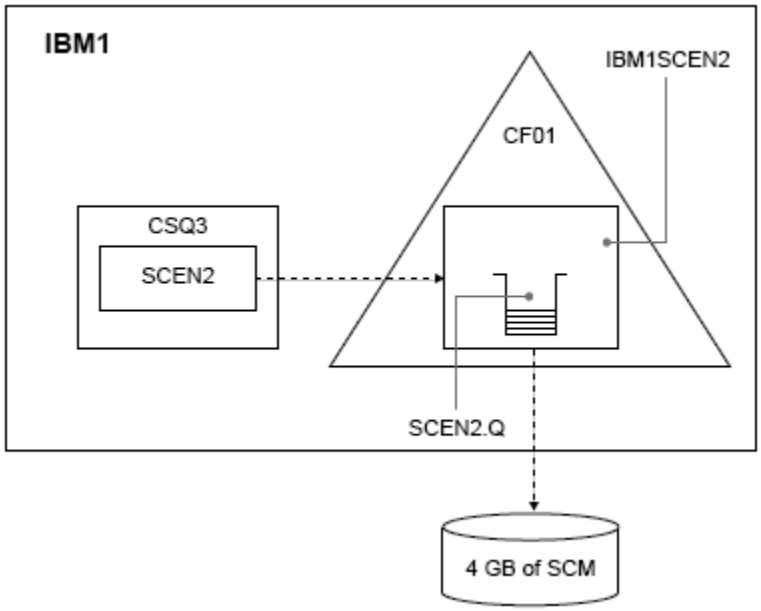


図 *64.* パフォーマンスを向上させるために *SCM* を追加する構成

手順

1. 次の手順を実行して、4 GB の SCM を構造体 IBM1SCEN2 に追加します。
   a) 次のコマンドを実行して、どのくらいの SCM がインストールされているか、また CF01 に割り振られているかを確認します。

```
D CF,CFNAME=CF01
```

b) 表示された出力の STORAGE CONFIGURATION セクションの STORAGE-CLASS MEMORY の数値を調べて、使用可能なストレージを確認します。

c) 以下のように、CFRM ポリシーを SCMMAXSIZE および SCMALGORITHM キーワードで更新します。

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
SCMMAXSIZE(4G)
SCMALGORITHM(KEYPRIORITY1)
```

2. 次のコマンドを発行して CFRM ポリシーをアクティブにします。

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
```

3. IBM1SCEN2 構造体を再ビルドします。

   前の変更を実行した際に、構造体が割り振られているため、この手順を実行する必要があります。

   次のコマンドを発行して、構造体を再ビルドします。

```
SETXCF START,REBUILD,STRNM=IBM1SCEN2
```

4. 次のコマンドを発行して、構造体の新しい構成を確認します。

```
D XCF,STR,STRNAME=IBM1SCEN2
```

   コマンドの出力を確認します (出力の一部を以下に示します)。

```
SPACE USAGE     IN-USE       TOTAL      %
ENTRIES:          33        342684      0
ELEMENTS:         48       6503697      0
EMCS:              2        575600      0
LOCKS:                        1024
```

タスクの結果

SCM を使用するために必要な制御ストレージでの増加分によって実ストレージの使用量がどれほど変化するかを計算します。

- SCM を構造体に追加する前の構造体の合計数は、210 ページの『パフォーマンスの向上 - 基本構成』に示されています。
  - 345,242 項目
  - 6,548,467 エレメント
  - 780,318 EMCS
- SCM を構造体に追加した後の構造体の合計数は、以下のとおりです。
  - 342,684 項目
  - 6,503,697 エレメント
  - 575,600 EMCS

これらの数値を使用して計算すると、SCM の追加後、構造体のサイズは以下のように減少しています。

- 2558 項目
- 44,770 エレメント

- 204,718 EMCS

SCM を管理するために使用される構造体ストレージの量は、2 GB の構造体に 4 GB の SCM を割り振った場合、次のとおりです。

```
(2558 + 44,770 + 204,718) * 256 = 61.5 MB
```

SCM を追跡するために使用される制御ストレージの量が増えるため (構造体のサイズと割り振られる SCM の量の両方とも増加します)、これ以上の SCM を追加しても構造体のサイズはほんのわずかしか減少しない可能性があります。

次のタスク

210 ページの『パフォーマンスの向上 - 基本構成』の最後のセクションで説明されているテストを繰り返します。

一定期間にわたる修正したアプリケーションの結果をプロットすることができます。 このプロットを以前にプロットしたものと比較すると、書き込み側アプリケーションでキューが部分的に空になることを待機する必要がなくなるため、ギザギザの波形でない出力を確認できます。

詳しくは、 MP16: WebSphere MQ for z/OS -Capacity planning & tuning を参照してください。

z/OS 分散キューイングとキュー共用グループ

分散キューイングおよびキュー共用グループは、アプリケーション・システムの可用性を高めるために使用できる 2 つの手法です。 このトピックでは、これらの手法の詳細について知ることができます。

共用キューでのメッセージの高可用性を補足するため、IBM MQ の分散キューイング・コンポーネントには、次のことを実現する補足的な機能が備えられています。

- ネットワークでの可用性をより高くする。
- キュー共用グループへのインバウンド・ネットワーク接続 のために能力を拡大する。

216 ページの図 65 には、分散キューイングと キュー共用グループが示されています。 ここでは、1 つの sysplex に含まれる 2 つのキュー・マネージャー が示されており、どちらも同じキュー共用グループに属しています。 どちらも共用キュー SQ1 にアクセスできます。 (例えば AIX および Windows の) ネットワーク内のキュー・マネージャーでは、いずれかのキュー・マネージャーのチャネル・イニシエーターを使用して、 メッセージをこのキューに書き込むことができます。 両方のキュー・マネージャーの複製アプリケーションが、 そのキューを扱います。

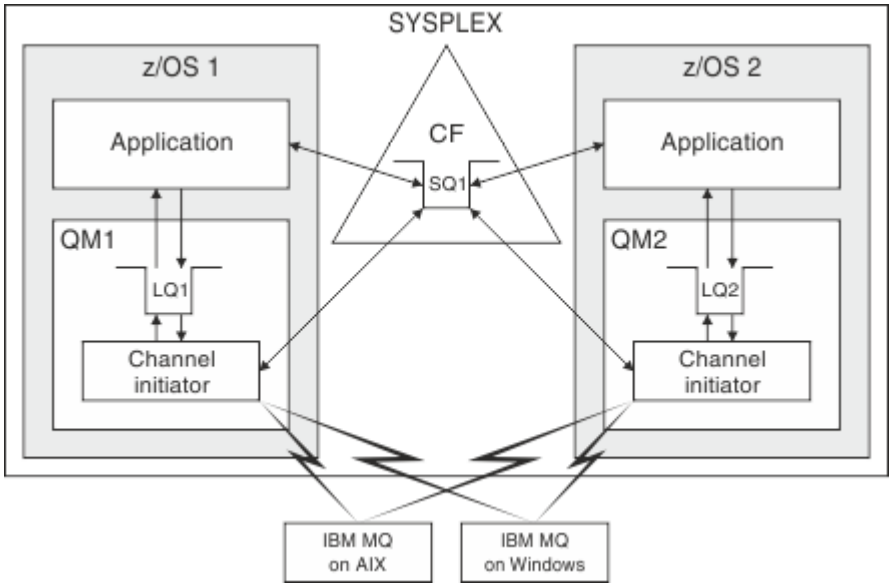


図 *65.* 分散キューイングとキュー共用グループ

関連概念

216 ページの『共用チャネル』
このトピックでは、共用チャネルの概念および IBM MQ for z/OS におけるその使用について知ることができます。

221 ページの『グループ内キューイング』
このセクションでは、 z/OS プラットフォームに固有の IBM MQ for z/OS 機能である、グループ内キューイングについて説明します。 この機能を使用できるのは、キュー共用グループに対して定義されているキュー・マネージャーだけです。

218 ページの『クラスターとキュー共用グループ』
このトピックでは、クラスターでキュー共用グループを使用する方法について理解することができます。

z/OS 共用チャネル

このトピックでは、共用チャネルの概念および IBM MQ for z/OS におけるその使用について知ることができます。

多くのネットワーキング製品には、サーバー障害をネットワークから隠したり、 適格サーバー群の間でインバウンド・ネットワーク要求のバランスを取るためのメカニズムがあります。 ネットワーク製品はインバウンド・ネットワーク接続要求用に汎用ポート を使用可能にし、インバウンド要求は適格サーバーの 1 つに接続することにより満足させられます。

これらのネットワーキング製品には、次のようなものがあります。

- VTAM 汎用資源
- SYSPLEX ディストリビューター

チャネル・イニシエーターは、これらの製品を最大限に利用し、 共用キューの機能を使用します。

共用チャネルには、共有インバウンド・チャネル、および共有アウトバウンド・チャネル の 2 種類があります。

- 共用インバウンド・チャネル
- 共用アウトバウンド・チャネル

チャネルの詳細については、以下を参照してください。

- 共用チャネルのまとめ
- 共用チャネルの状況

共用インバウンド・チャネル

キュー共用グループの各チャネル・イニシエーターは、 汎用ポート で listen する 別のリスナー作業を開始します。 この汎用ポートは、サポートするテクノロジー (VTAM, TCP/IP) のいずれかを使用することにより、ネットワークから利用できるようになります。 汎用ポートへのインバウンド・ネットワーク接続要求は、キュー共用グループ (QSG) に含まれていて、その汎用ポートで listen しているいずれかのリスナーに対して、ネットワーク・テクノロジーにより送られます。

インバウンド接続の接続先となるチャネル・イニシエーター上でチャネルを開始できます。ただし、そのチャネル・イニシエーターには、その名前のチャネルのチャネル定義へのアクセス権が必要です。 チャネル定義は、特定のキュー・マネージャー専用に定義したり、共用リポジトリーに格納してどこからでも利用できるようにしたり (グローバル定義) することができます。 すなわち、チャネル定義をグローバル定義として定義 すれば、そのチャネル定義は、キュー共用グループの 任意のチャネル・イニシエーターで利用できることになります。

汎用ポートを使用してチャネルを開始する場合には、別の違いもあります。 つまり、チャネルが同期化されるのは、キュー共用グループに対してであり、 個々のキュー・マネージャーに対してではありません。 例えば、汎用ポートを使用してチャネルを開始するリモート・キュー・マネージャーについて考えてみましょう。 チャネルを最初に開始する場合、キュー・マネージャー QM1 で開始され、 メッセージが流されます。 チャネルが停止して、キュー・マネージャー QM2 で再開される場合、 同期化はキュー共用グループに対して行われるため、 流されたメッセージの数に関する情報は正しい状態が保たれます。

汎用ポートを使用して開始したインバウンド・チャネルは、 メッセージを任意のキューに書き込むときに使用できます。 リモート・キュー・マネージャーは、ターゲット・キューが共用キューなのか どうかを知りません。 ターゲット・キューが共用キューであれば、リモート・キュー・マネージャーはワークロードのバランスを取りながら、利用できる任意のチャネル・イニシエーターを使用して接続を行い、メッセージは共用キューに書き込まれます。

ターゲット・キューが専用キューであれば、チャネルの現在のインスタンスが接続されているキュー・マネージャーが所有する専用キューにメッセージは書き込まれます。 複製ローカル・キュー として知られるこの環境では、各キュー・マネージャーには専用キューの同じセットが定義されていなければなりません。

キュー共用グループ用の SVRCONN チャネルの構成

キュー共用グループ用の SVRCONN チャネルの最適な構成は、Point-to-Point チャネルの異なるポート番号を使用する CHINIT ごとに専用リスナーをセットアップすることです。 これらのリスナー・ポートは、仮想 IP アドレス (VIPA) を使用するシスプレックス・ディストリビューターなど、新しいワークロード分散メカニズムの「バックエンド」リソースとして使用されます。 外部 VIPA アドレスは、ネットワーク内の CLNTCONN 定義用のターゲット・アドレスとして使用します。 SVRCONN チャネルは QSGDISP(GROUP) を使用して定義できるため、同じ定義を QSG のすべてのキュー・マネージャーで使用できます。 この構成により、共有リスナーの使用が回避されるため、クライアント/サーバー・チャネルには必要ない共有チャネル状態を維持するキュー共有グループのパフォーマンスへの影響が軽減されます。

共用アウトバウンド・チャネル

アウトバウンド・チャネルは、メッセージを共用伝送キューから 取り出すときに、共用チャネルであると見なされます。 共用チャネルである場合、キュー共用グループ・レベルでの 同期化情報が保管されます。 したがって、通信サブシステム、チャネル・イニシエーター、あるいはキュー・マネージャーで障害が発生しても、別のキュー・マネージャー や、キュー共用グループ内のチャネル・イニシエーターのインスタンス でチャネルを再開できることになります。 このような方法で障害が生じたチャネルを再開することは、 ピア・チャネル回復 という共用チャネルの機能です。

共用アウトバウンド・チャネルのワークロード・バランシング

アウトバウンド共用チャネルは、特定のチャネル・イニシエーターで チャネルを開始することを指定していない場合に、キュー共用グループ内に ある任意のチャネル・イニシエーターで開始するのに適したチャネルです。 IBM MQ によって選択されるチャネル・イニシエーターは、次の基準で決定されます。

- 現在必要な通信サブシステムがそのチャネル・イニシエーターで使用可能か。
- Db2 接続がそのチャネル・イニシエーターで使用可能か。
- 現在のところ、どのチャネル・イニシエーターのワークロードが一番低いか。 ワークロードには、活動状態で再試行しているチャネルが含まれます。

共用チャネルのまとめ

共用チャネルと専用チャネルの相違点は、次のとおりです。

専用チャネル
単一のチャネル・イニシエーターに結び付いています。

- アウトバウンド・チャネルは、ローカル伝送キューを使用します。
- インバウンド・チャネルは、ローカル・ポートで開始します。
- 同期化の情報は、SYSTEM.CHANNEL.SYNCQ キューに入れられます。

共用チャネル
ワークロードのバランスが取られていて、可用性が高くなっています。

- アウトバウンド・チャネルは、共用伝送キューを使用します。
- インバウンド・チャネルは、汎用ポートで開始します。
- 同期化の情報は、SYSTEM.QSG.CHANNEL.SYNCQ キューに入れられます。

START CHANNEL コマンドで CHLDISP オプションを使用してチャネルを開始するときに、チャネルを専用にするか共用にするかを指定します。 共用チャネルは、専用チャネルと同じ方法で始動することにより、開始することができます。 しかし、共用チャネルを開始すると、IBM MQ はワークロードのバランスを取り、キュー共用グループ内の最適なチャネル・イニシエーターでチャネルを開始します。 (必要であれば、共用チャネルを特定のチャネル・イニシエーター で開始することを指定できます。)

共用チャネルの状況

キュー共用グループのチャネル・イニシエーターは、Db2 内で共用チャネル状況表を保守します。 ここには、どのチャネル・イニシエーターでどのチャネルが 活動化されているのかが記録されます。 この共用チャネル状況表は、チャネル・イニシエーター または通信システムに障害があるときに使用されます。 その際には、キュー共用グループの別のチャネル・イニシエーターで、 どのチャネルを再開する必要があるのかが示されます。

z/OS クラスターとキュー共用グループ

このトピックでは、クラスターでキュー共用グループを使用する方法について理解することができます。

1 つの定義の中で、クラスターが共用キューを利用できるように 定義することができます。 そのためには、共用キューを定義するときに、クラスター の名前を指定するようにします。

ネットワーク内のユーザーは、その共用キューがキュー共用グループ 内の各キュー・マネージャーによってホスティングされているものと 見なします (ただし、この共用キューは、キュー共用グループによって ホスティングされていると公示されません)。 クライアントは、キュー共用グループの任意のメンバーで セッションを開始し、その同じ共用キューにメッセージを 書き込むことができます。

219 ページの図 66 には、クラスターのメンバーが、キュー共用グループ の任意のメンバーを経由して共用キューにアクセスする方法が示されています。

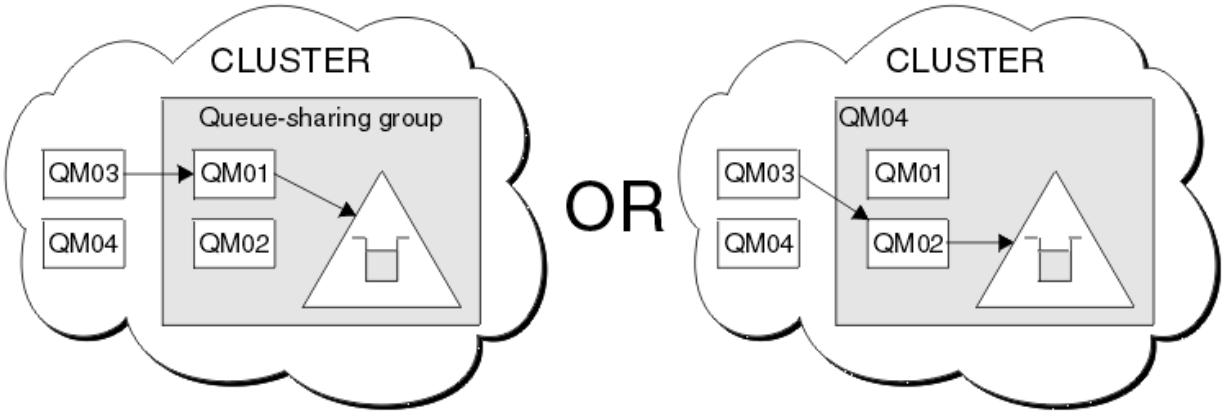


図 66. クラスターの一部であるキュー共用グループ

z/OS 共用キューを使用したワークロード分散への影響

このトピックでは、キュー共用グループにおける共用キューを使用したワークロード分散に影響するいくつかの要素について理解することができます。

IBM MQ は、共用キュー用のワークロード・バランシングを提供していません。 ただし、キュー共用グループ (QSG) 内のワークロード分散は、プル・ベース方式 で影響を受ける場合があります。 1 つのキューを処理する (つまり、共用キューに書き込まれるメッセージを受け取る) キュー・マネージャーの選択は、キュー共用グループ内の各キュー・マネージャーで利用可能な処理能力、およびシスプレックス全体で定義されているワークロード管理目標により影響されます。

ただし、メッセージの MQPUT を実行するキュー・マネージャーも、メッセージを get するキュー・マネージャーの決定に大きく影響する場合があることを認識する必要があります。

通常はローカル・キュー・マネージャーが MQGET を実行する

MQPUT を実行するアプリケーションでは、通常、ローカル・キュー・マネージャーが、アプリケーションが接続されているキュー・マネージャーです。

厳密にどのキュー・マネージャーが、取得側のアプリケーションに代わって MQGET を実行することによりメッセージの MQPUT を処理するかは、以下の考慮事項の影響を受けます。

空の共用キューにメッセージが put されると、通常は、キュー共用グループ内の他のキュー・マネージャーに通知される前に、ローカル・キュー・マネージャーが POST されます。 ローカル・キュー・マネージャーがメッセージを処理する役割である場合、ローカル・キュー・マネージャーは、QSG 内の他のキュー・マネージャーに先立って、カップリング・ファシリティー (CF) からリスト移行通知を受け取ります (リスト移行通知とは、共用キューの状態が空の状態から空ではない状態に変化したことを知らせる通知のことです)。

この場合、以下のようなシナリオが考えられます。

1. 同期点の外での非持続メッセージの MQPUT、および待機 *getter* への高速 *put*。

   キューのローカル・キュー・マネージャーに待機呼び出しを伴う *MQGET* を使用するアプリケーションがある場合、メッセージの MQPUT は、取得側のアプリケーションのバッファーに直接渡され、キューには書き込まれません。 このことは、共用キューにも非共用キューにも当てはまります。 この機能は、待機 *getter* への高速 *put* メカニズムと呼ばれます。 共用キューの場合、キューが空の状態から空ではない状態に遷移することがなかったため、QSG の他のキュー・マネージャーには通知されません。 つまり、例えば、このキュー・マネージャーがこのアプリケーションからのすべての put を処理することができ、キューにメッセージを入れるアプリケーションが他にないと想定される場合、キュー共用グループ内のその他のキュー・マネージャーがこのキューの排出を援助することはありません。 ただし、ローカル・キュー・マネージャー上に待機呼び出しを伴う MQGET がなく、メッセージが共用キューに入れ

られる場合、CF は、リスト移行の通知ルールに従ってキュー共用グループ内のその他のキュー・マネージャーに通知します。

2. 持続メッセージ (同期点内にあるメッセージ) の MQPUT。

   この場合、ローカル・キュー・マネージャー上に待機呼び出し付きの *MQGET* を持つアプリケーションがある場合、メッセージは共用キューに入れられ、CF は、リスト移行の通知ルールに従ってキュー共用グループ内のその他のキュー・マネージャーに通知します。 ただし、ローカル・キュー・マネージャーは、CF からの移行通知を待たず、最初にローカル側の待機呼び出しを伴う *MQGET* を受け入れます。通常は、次に、キュー共用グループ内の他のキュー・マネージャーが CF 通知に応答できるようになる前に、アプリケーションに代わってこのメッセージの get を実行します。 これは、ローカル・キュー・マネージャーがどれほどビジーかに依存しています。 それ以外の場合、空のキューにメッセージが到着したため CF により通知されたすべてのキュー・マネージャーは、最初に get の処理を試みます。 最初に応答するキュー・マネージャーが、新しいメッセージを処理します。

3. 最後に、キューからメッセージが排出されず、CF が、キューの状態が空の状態から空ではない状態に変化したとの通知を送信した場合、接続されたすべてのキュー・マネージャーに、キューの処理を援助する機会が与えられます。 このような場合、ワークロードは *pull* ベース であると言われます。

この設計により、純粋に pull ベースのワークロード分散のパフォーマンス向上が可能になります。 その目的は、CF に保持されているキューの高可用性という利点を活用すること、 かつキュー・マネージャーが、可能なかぎり、メッセージ・ワークロードをできるだけ効率的に処理するために CF を参照する必要なく MQGET を実行できるようにすることです。

上記のパフォーマンスの向上よりワークロード・バランシングに重点を置く必要がある場合は、代わりの方法を採用することができます。 例えば、いずれの取得アプリケーションも、書き込みアプリケーションが接続されているのと同じキュー・マネージャーに接続されないようにします。 この設計を使用すると、すべてのメッセージはキューに入れられ、キューの状態が空の状態から空ではない状態に変化したときに、QSG 内のすべてのキュー・マネージャーが、 そのような遷移を処理するための CF アルゴリズムに従って通知されます。 また、待機 *getter* への高速 *put* メカニズムは、適用されません。

z/OS 共用キューおよびキュー共用グループに関する詳細

このトピックの表を使用して、IBM MQ for z/OS が共有キューおよびキュー共有グループを使用する方法についての詳細を確認してください。

表 *19.* 共用キューおよびキュー共用グループに関する詳細

トピック	**参照先**
キュー共用グループの回復	261 ページの『z/OS での回復と再始動』
キュー共用グループのセキュリティー	278 ページの『IBM MQ for z/OS におけるセキュリティーの概念』
専用およびグローバルなオブジェクト定義 異なるキューへのコマンドの送信 します。	z/OS
カップリング・ファシリティーの計画 環境	カップリング・ファシリティー・リソースの定義
SMDS 環境の計画	共用メッセージ・データ・セット (SMDS) 環境の計画
以下の計画 Db2 環境	Db2 環境の計画
共用キューのセットアップ システム・パラメーター	175 ページの『共用キューとキュー共用グループ』

表 19. 共用キューおよびキュー共用グループに関する詳細 (続き)	
トピック	**参照先**
ユーティリティー・プログラム キューのマイグレーション	z/OS での IBM MQ ユーティリティーのリファレンス
コンソール・メッセージ	IBM MQ for z/OS のメッセージ
MQSC コマンド	MQSC コマンド
IBM MQ クラスター	キュー・マネージャー・クラスターの構成
IBM MQ 分散キューイング チャネル名	分散キュー管理の概要
アプリケーションの作成	アプリケーション設計の概要
MQCONNX 呼び出し	MQCONNX

z/OS グループ内キューイング

このセクションでは、 z/OS プラットフォームに固有の IBM MQ for z/OS 機能である、グループ内キューイングについて説明します。 この機能を使用できるのは、キュー共用グループに対して定義されているキュー・マネージャーだけです。

キュー共用グループの詳細は、175 ページの『共用キューとキュー共用グループ』を参照してください。

グループ内キューイングの概念

チャネルを定義しなくても、キュー共用グループのキュー・マネージャー間で迅速なメッセージ転送を実行することができます。 この場合、共用伝送キューである SYSTEM.QSG.TRANSMIT.QUEUE と呼ばれるシステム・キューが使用されます。 キュー共用グループの各キュー・マネージャーは、グループ内キューイング・エージェントというタスクを開始し、 それぞれのキュー・マネージャーに向けられた、このキューに着信するメッセージを待機します。 そのようなメッセージが検出されると、このキューから取り出されて正しい宛先キューに置かれます。

メッセージを正しい宛先キュー・マネージャーに転送する場合には、 標準的なネーム・レゾリューション規則が使用されますが、 グループ内キューイング (IGQ) を使用していて、ターゲット・キュー・マネージャーがキュー共用グループ内にある場合は、 伝送キューおよびチャネルは使用されず、 SYSTEM.QSG.TRANSMIT.QUEUE が使用されます。

グループ内キューイングは、キュー・マネージャー属性を使用して使用可能に設定します。 グループ内キューイングによって、非持続メッセージは同期点の外へ移動され、 永続メッセージは同期点内に移動されます。 メッセージをターゲット・キューへ送信するときに問題が発生すると、 グループ内キューイングはそれらのメッセージを送達不能キューに書き込もうとします。 送達不能キューがいっぱいになっているか、または定義されていない場合、非持続メッセージは廃棄されますが、持続メッセージはバックアウトされて SYSTEM.QSG.TRANSMIT.QUEUE に戻され、IGQ エージェントがメッセージの送達に成功するまで再試行します。

キュー共用グループにある別のキュー・マネージャーのキューに向けられたメッセージを受信するインバウンド共用チャネルでは、 グループ内キューイングを使用して、メッセージを正確な宛先にホップ することができます。

ターゲット・キューが共用キューの場合、 最初にメッセージがターゲット・キュー・マネージャーに転送されるのではなく、 ローカル・キュー・マネージャーがそのメッセージを直接ターゲット・キューに 書き込むようにしたい場合があります。 キュー・マネージャー属性 SQQMNAME を使用すれば、これを制御す

ることができます。 SQQMNAME の値を USE に設定すると、 **ObjectQMgrName** で指定されたキュー・マネージャーで MQOPEN コマンドが実行されます。

ただし、ターゲット・キューが共用キューであり、SQQMNAME の値を IGNORE に設定し、**ObjectQMgrName** がキュー共用グループ内の別のキュー・マネージャーの値である場合、共用キューはローカル・キュー・マネージャーでオープンされます。 ローカル・キュー・マネージャーがターゲット・キューをオープンできないか、メッセージをキューに書き込むことができない場合、メッセージは IGQ または IBM MQ チャネルのいずれかを介して、指定された **ObjectQMgrName** に転送されます。

グループ内キューイングを使用して、キュー共用グループ内のリモート・キュー・マネージャーに置かれているキューへの小規模なメッセージの送達を高速化および効率化できます。 グループ内キューイングでは、最大 100 MB から伝送キュー・ヘッダーの長さを 引いた 大きなメッセージもサポートされます。

注 : この機能を使用する場合、ユーザーはキュー共用グループ内の各キュー・マネージャー上のキューへの同じアクセス権を 持つ必要があります。

下の図は、グループ内キューイングの典型例を示しています。

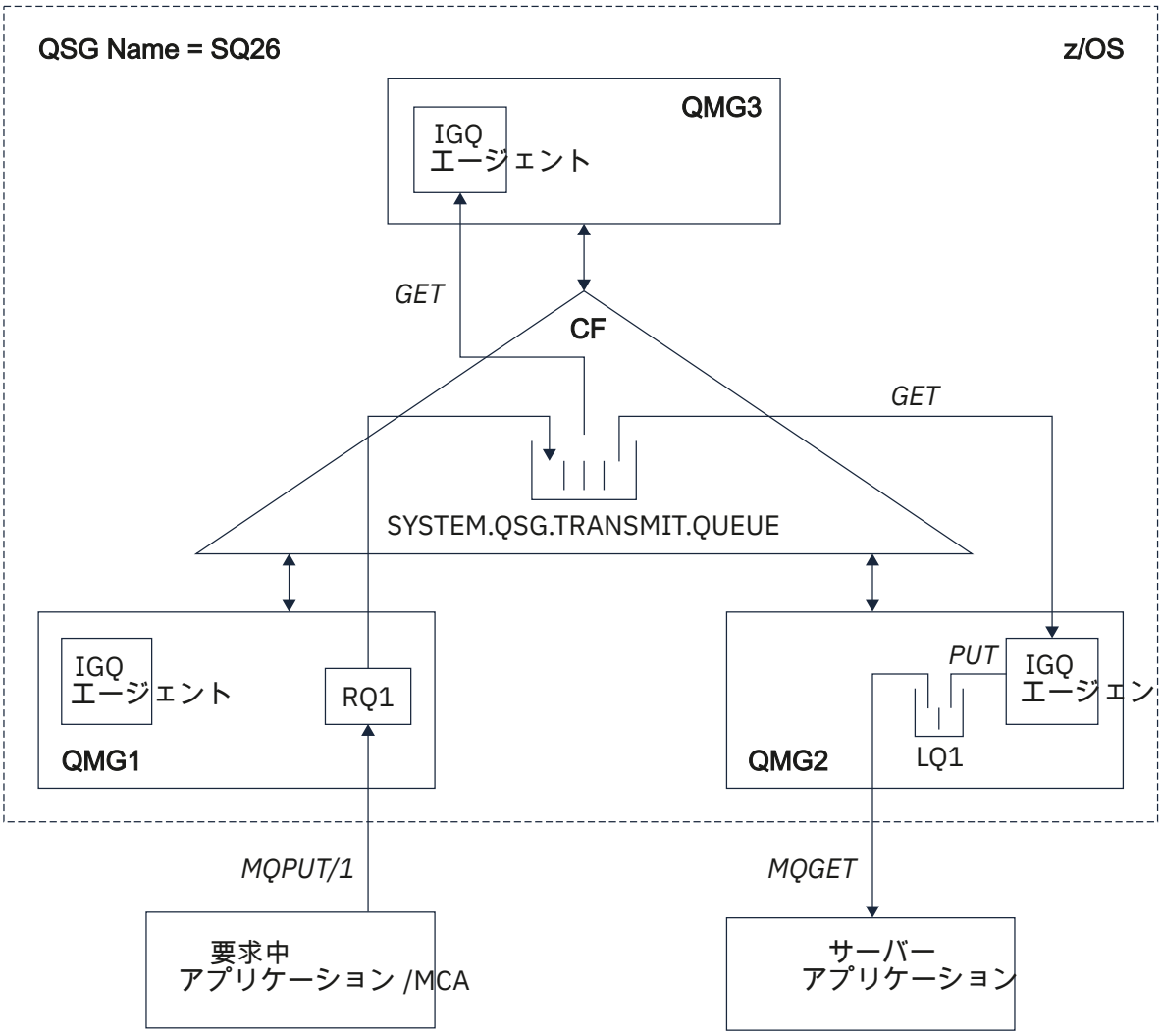


図 67. グループ内キューイングの例

この図は以下を示しています。

- SQ26 という名前のキュー共用グループに対して定義されている 3 つのキュー・マネージャー (QMG1、QMG2、および QMG3) で実行されている IGQ エージェント。
- カップリング・ファシリティー (CF) に定義されている共用伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE。

- キュー・マネージャー QMG1 に定義されているリモート・キュー定義。
- キュー・マネージャー QMG2 に定義されているローカル・キュー。
- キュー・マネージャー QMG1 に接続されている要求アプリケーション (このアプリケーションは、メッセージ・チャネル・エージェント (MCA) の場合があります)。
- キュー・マネージャー QMG2 に接続されているサーバー・アプリケーション。
- SYSTEM.QSG.TRANSMIT.QUEUE に書き込まれている要求メッセージ。

グループ内キューイングとグループ内キューイング・エージェント

キュー・マネージャーの初期化時に IGQ エージェントが始動されます。 アプリケーションが開き、メッセージをリモート・キューに入れると、ローカル・キュー・マネージャーは、メッセージの転送にグループ内キューイングを使用するかどうかを決めます。 ローカル・キュー管理プログラムは、グループ内キューイングを使用することにした場合、メッセージを SYSTEM.QSG.TRANSMIT.QUEUE に入れます。 ターゲットのリモート・キュー・マネージャー上の IGQ エージェントは、メッセージを取り出して宛先キューに入れます。

グループ内キューイングの用語

用語の説明。グループ内キューイング、グループ内キューイングが使用する共用送信キュー、グループ内キューイング・エージェント。

グループ内キューイング

グループ内キューイングを使用すると、チャネルを定義しなくても、キュー共用グループ内のキュー・マネージャーどうしが高速かつ安価でメッセージをやりとりすることができます。

グループ内キューイングで使用される共用伝送キュー

どのキュー共用グループにも、グループ内キューイングに使用するための SYSTEM.QSG.TRANSMIT.QUEUE という名前の共用伝送キューがあります。 グループ内キューイングが使用可能になっている場合に、リモート・キューをオープンすると、ネーム・レゾリューション・パスに SYSTEM.QSG.TRANSMIT.QUEUE が示されます。 アプリケーション (メッセージ・チャネル・エージェント (MCA) を含む) がリモート・キューにメッセージを書き込むと、ローカル・キュー・マネージャーがそのメッセージを高速で転送するべきかどうかを判断し、SYSTEM.QSG.TRANSMIT.QUEUE に入れます。

グループ内キューイング・エージェント

IGQ エージェントは、処理対象のメッセージが SYSTEM.QSG.TRANSMIT.QUEUE に着信するのを待機しているタスクであり、キュー・マネージャーの初期化時に開始されます。 IGQ エージェントは、該当するメッセージをこのキューから取り出し、宛先キューに渡します。

常にキュー・マネージャーごとに IGQ エージェントが開始されます。それは、キュー・マネージャー自身が内部処理でグループ内キューイングを使用するからです。

z/OS グループ内キューイングの利点

グループ内キューイングには、システム定義の削減、システム管理の削減、パフォーマンスの向上、マイグレーションのサポート、キュー共用グループ内のキュー・マネージャー間のマルチホップ時のメッセージのデリバリーなどの利点があります。

グループ内キューイングの利点は、次のとおりです。

システム定義の削減
グループ内キューイングを使用すると、キュー共用グループ内のキュー・マネージャーどうしをつなぐチャネルを定義しなくて済みます。

システム管理の削減
キュー共用グループ内のキュー・マネージャーをつなぐ定義済みチャネルがないので、チャネル管理の必要がありません。

パフォーマンスの向上

ターゲット・キューへのメッセージのデリバリーに 1 つの IGQ エージェントしか必要ない (中間の 2 つの送信側および受信側エージェントは不要) ので、グループ内キューイングを使用したメッセージ・デリバリーは、チャネルを使用したメッセージ・デリバリーよりも安価になります。 グループ内キューイングでは、送信側コンポーネントは必要ないため、受信側コンポーネントしかありません。 このような簡素化が可能なわけは、ローカル・キュー・マネージャーでの書き込み操作の後、(同期点有効範囲内でのメッセージの書き込みの場合) コミットが完了すると、宛先キューに渡すためのメッセージが、宛先キュー・マネージャーの IGQ エージェントの手に入るからです。

移行のサポート

キュー共用グループ外部のアプリケーションは、キュー共用グループ内のいずれか 1 つのキュー・マネージャーに接続していれば、キュー共用グループ内のどのキュー・マネージャーに置かれているキューにでもメッセージを送達することができます。 これは、リモート・キュー・マネージャー上のキュー宛てのメッセージが受信側チャネルに着信すると、グループ内キューイングを使用してそのメッセージを宛先キューに透過的に送ることができるからです。 この機能のおかげで、キュー共用グループ外部のどのシステムも変更せずに、キュー共用グループの中にアプリケーションを配置することができます。

下の図は、典型的な構成を示しています。詳細は、次のとおりです。

- キュー・マネージャー QMG1 に接続された要求側アプリケーションは、キュー・マネージャー QMG3 上のローカル・キューにメッセージを送る必要があります。
- キュー・マネージャー QMG1 はキュー・マネージャー QMG2 にのみ接続されています。
- キュー・マネージャー QMG2 と QMG3 は、以前はチャネルを使用して接続されていましたが、現在はキュー共用グループ SQ26 のメンバーです。

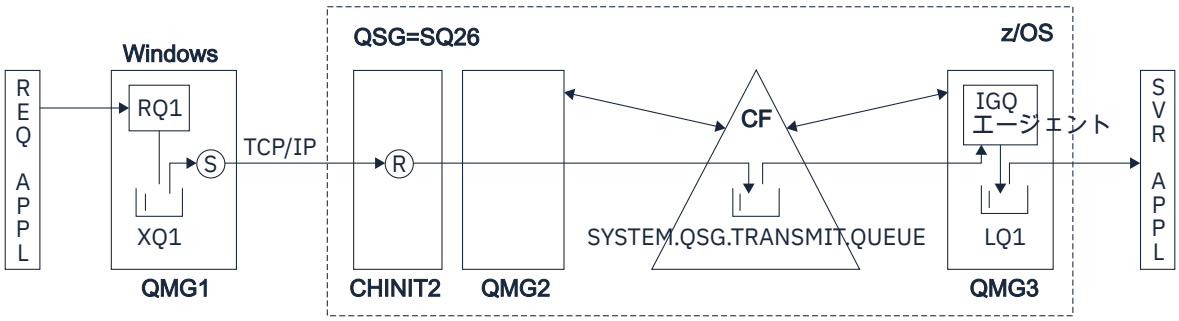


図 *68.* マイグレーション・サポートの例

操作のフローは次のとおりです。

1. 要求側アプリケーションは、リモート・キュー・マネージャー QMG3 のローカル・キュー LQ1 宛てのメッセージを、リモート・キュー定義 RQ1 に書き込みます。
2. Windows NT ワークステーションで実行中のキュー・マネージャー QMG1 は、メッセージを伝送キュー XQ1 に入れます。
3. QM1 の送信側 MCA (S) は、TCP/IP を介してチャネル・イニシエーター CHINIT2 上の受信側 MCA (R) にメッセージを送信します。
4. チャネル・イニシエーター CHINIT2 上の受信側 MCA (R) は、共用伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE にメッセージを入れます。
5. キュー・マネージャー QMG3 の IGQ エージェントは、SYSTEM.QSG.TRANSMIT.QUEUE からメッセージを取り出し、ターゲット・ローカル・キュー LQ1 に入れます。
6. サーバー・アプリケーションは、ターゲット・ローカル・キューからメッセージを取り出して処理します。

キュー共用グループ内のキュー・マネージャー間のマルチホップ時のメッセージのデリバリー

前のマイグレーション・サポート の図は、キュー共用グループ内のキュー・マネージャー間のマルチホップ時のメッセージのデリバリーについても示しています。 キュー共用グループ内のキュー・マネ

ージャーに届くメッセージは、宛先がキュー共用グループ内の別のキュー・マネージャーのキューである場合でも、グループ内キューイングを使用して宛先キュー・マネージャーにある宛先キューに簡単に送信することができます。

z/OS グループ内キューイングの制限

グループ内キューイングには、グループ内キューイングを使用した転送に適したメッセージ、キュー・マネージャーごとのグループ内キューイング・エージェントの数、およびグループ内キューイング・エージェントの開始と停止などの制限があります。

このトピックでは、グループ内キューイングの制限事項について説明します。

グループ内キューイングを使用した転送に適したメッセージ
グループ内キューイングは結合装置 (CF) で定義された共用伝送キューを使用するため、グループ内キューイングは、共用キューでサポートされている最大メッセージ長から伝送キュー・ヘッダーの長さを引いたメッセージの送信に制限されています。

キュー・マネージャーごとのグループ内キューイング・エージェントの数
キュー共用グループ内のキュー・マネージャーごとに 1 つの IGQ エージェントしか開始されません。

グループ内キューイング・エージェントの開始と停止
IGQ エージェントは、キュー・マネージャーの初期化時に開始され、キュー・マネージャーのシャットダウン時に終了されます。 このエージェントは、長期実行で、(異常終了時における) 自己リカバリーのタスクとして設計されています。 SYSTEM.QSG.TRANSMIT.QUEUE の定義にエラーがあった (例えば、該当するキューが読み取り禁止であるなど) 場合、IGQ エージェントは再試行を続けます。 IGQ エージェントでエラーが発生して、キュー・マネージャーがアクティブのままエージェントが正常終了した場合、ALTER QMGR IGQ(ENABLED) コマンドを実行してエージェントを再起動することができます。 このコマンドは、キュー・マネージャーをリサイクルする必要を避けます。

キュー・マネージャーの属性 IGQ の設定 (ENABLED または DISABLED)
キュー・マネージャーの属性 IGQ が ENABLED または DISABLED に設定されている場合、既存のオブジェクト・ハンドルは、理由コード MQRC_OBJECT_CHANGED で無効になることがあります。 詳しくは、イントラグループ・キューイングの準備作業を参照してください。

z/OS グループ内キューイングの概要

このトピックで説明されているように、グループ内キューイングの使用可能化、使用禁止、および使用を行うことができます。

グループ内キューイングの使用可能化
キュー・マネージャーでグループ内キューイングを使用可能にするには、次のようにします。

- SYSTEM.QSG.TRANSMIT.QUEUE という名前の共用伝送キューを定義します。 このキューの定義は、キュー共用グループの SYSTEM オブジェクト用の CSQINP2 サンプルである thlqual.SCSQPROC(CSQ4INSS) 内に置かれています。 グループ内キューイングを正しく作動させるには、thlqual.SCSQPROC(CSQ4INSS) に述べられているとおりの正しい属性を使用してこのキューを定義しなければなりません。
- IGQ エージェントはキュー・マネージャーの初期化で必ず開始されるため、グループ内キューイングは、インバウンド・メッセージ処理では常に使用可能です。 IGQ エージェントは、SYSTEM.QSG.TRANSMIT.QUEUE に置かれたすべてのメッセージを処理します。 ただし、アウトバウンド処理にグループ内キューイングを使用可能にするには、キュー・マネージャー属性 IGQ を ENABLED に設定しなければなりません。

  重要 : キュー・マネージャーの属性 IGQ が ENABLED に設定されている場合、既存のオブジェクト・ハンドルは、理由コード MQRC_OBJECT_CHANGED で無効になることがあります。 詳しくは、233 ページの『グループ内キューイングの特定のプロパティー』 を参照してください。 この理由コードの「プログラマー応答」セクションで説明されているように、アプリケーションはこの状態を処理するようにコーディングする必要があります (詳細については、2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED を参照してください)。

さらに、IGQ は初期化中に始動し、シャットダウン時に終了する長期間実行の自動回復タスクとして設計されています。 詳しくは、225 ページの『グループ内キューイングの制限』を参照してください。

グループ内キューイングの使用禁止

アウトバウンド・メッセージの転送でグループ内キューイングを使用禁止にするには、キュー・マネージャー属性 IGQ を DISABLED に設定します。 グループ内キューイングが特定のキュー・マネージャーで使用不可になっている場合、そのキュー・マネージャーの IGQ エージェントは、アウトバウンド転送でグループ内キューイングが使用可能なキュー・マネージャーを使って、SYSTEM.QSG.TRANSMIT.QUEUE に置かれたインバウンド・メッセージを処理できます。

重要 : キュー・マネージャーの属性 IGQ が ENABLED に設定されている場合、既存のオブジェクト・ハンドルは、理由コード MQRC_OBJECT_CHANGED で無効になることがあります。 詳しくは、233 ページの『グループ内キューイングの特定のプロパティー』 を参照してください。 この理由コードの「プログラマー応答」セクションで説明されているように、アプリケーションはこの状態を処理するようにコーディングする必要があります (詳細については、2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED を参照してください)。

さらに、IGQ は初期化中に始動し、シャットダウン時に終了する長期間実行の自動回復タスクとして設計されています。 詳しくは、225 ページの『グループ内キューイングの制限』を参照してください。

グループ内キューイングの使用

グループ内キューイングを使用可能にすると、いつでもこれを利用できるようになり、キュー・マネージャーは可能であればいつでもこれを使用します。 つまり、リモート・キュー定義、完全修飾リモート・キュー、またはクラスター・リモートにアプリケーションがメッセージを書き込むと、そのメッセージがグループ内キューイングを使用したデリバリーに適しているかどうかがキュー・マネージャーで調べられ、適していれば、SYSTEM.QSG.TRANSMIT.QUEUE に入れられます。 ユーザー・アプリケーションやアプリケーション・キューを変更する必要はありません。なぜなら、キュー・マネージャーは、適格なメッセージにおいて他のいかなる伝送キューよりも SYSTEM.QSG.TRANSMIT.QUEUE を優先させるからです。

z/OS グループ内キューイングの構成

通常のグループ内キューイング構成に加えて、他の構成も可能です。

221 ページの『グループ内キューイングの概念』 は、通常の構成を示しています。

関連概念

226 ページの『グループ内キューイングでの分散キューイング (複数のデリバリー・パス)』
短メッセージを処理するアプリケーションの場合、キュー共用グループにあるキュー・マネージャー間のみのメッセージを送信するグループ内キューイングを構成することが可能かもしれません。

228 ページの『グループ内キューイングでのクラスター化 (複数のデリバリー・パス)』
キュー・マネージャーを、キュー共用グループ化に加えてクラスター化されるように構成することができます。

231 ページの『クラスター化、グループ内キューイング、および分散キューイング』
キュー共用グループに加えてクラスターのメンバーでもあって、しかも送信側/受信側のチャネル・ペアを使用して分散キューイング・マネージャーに接続されるようにキュー・マネージャーを構成することができます。

z/OS グループ内キューイングでの分散キューイング (複数のデリバリー・パス)

短メッセージを処理するアプリケーションの場合、キュー共用グループにあるキュー・マネージャー間のみのメッセージを送信するグループ内キューイングを構成することが可能かもしれません。

チャネル通信でのグループ内キューイングの選択は、CFSTRUCT タイプ・レベルでコントロールできます。(4 または 5 ではなく 3) 最大メッセージ長は SYSTEM.QSQ.TRANSMIT.QUEUE で設定されたとおりです。

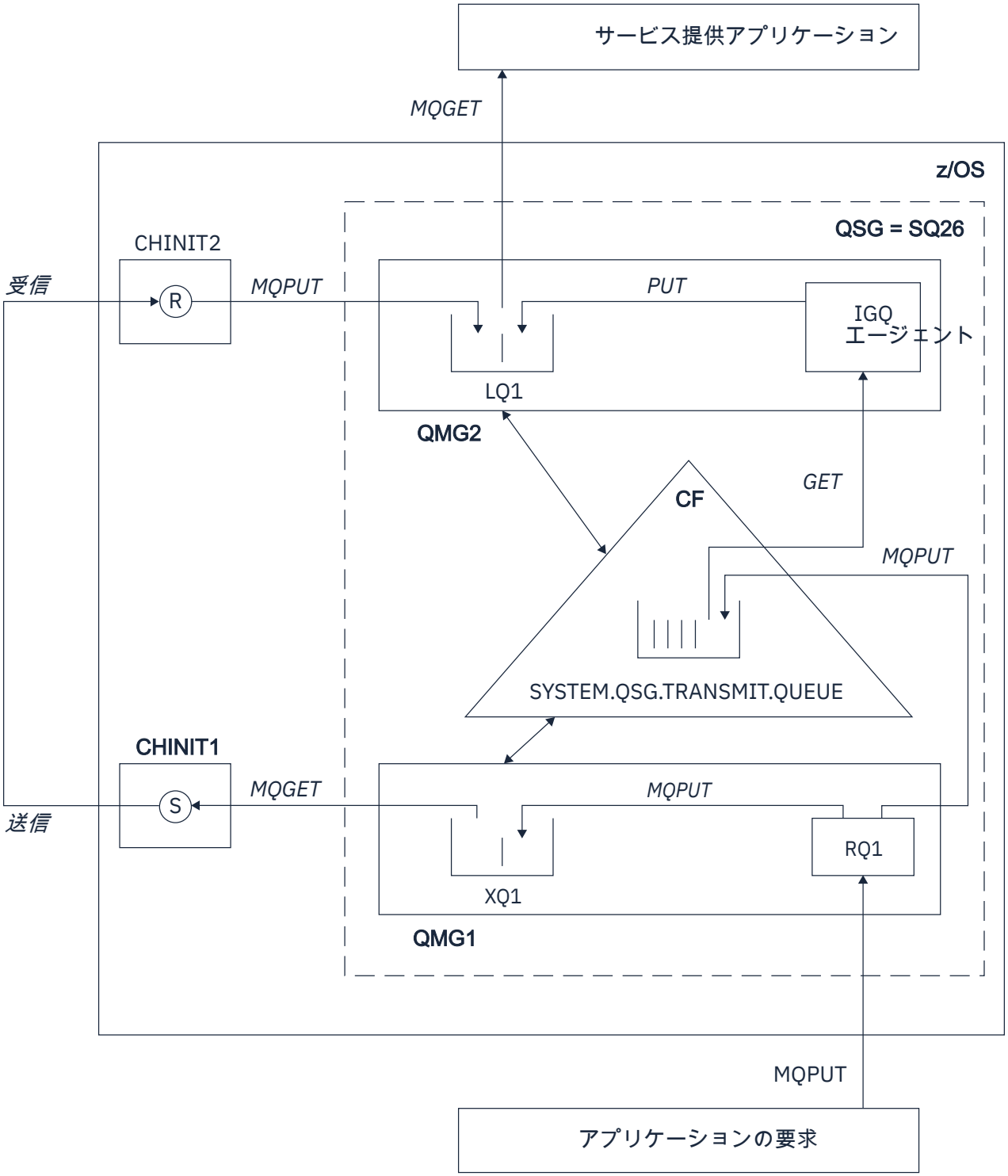


図 69. 構成例

オープン/書き込みの処理

1. 注意する必要があるのは、要求側アプリケーションがリモート・キュー RQ1 をオープンすると、非共用伝送キュー XQ1 と共用伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE の両方のネーム・レゾリューションが行われるということです。
2. 要求アプリケーションがリモート・キューにメッセージを入れる場合、キュー・マネージャーのアウトバウンド転送でグループ内キューイングが有効になっているか、またメッセージがどのような特性を持つかに応じて、メッセージは、伝送キュー XQ1 または伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE に入

れられます。 キュー・マネージャーは、大規模なメッセージはすべて伝送キュー XQ1 に書き込み、小規模なメッセージはすべて伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE に書き込みます。

3. 伝送キュー XQ1 が満杯であるか、または使用できない場合は、大規模なメッセージの書き込み要求は失敗し、それと同期して該当する戻りコードおよび理由コードが戻されます。 それに対して、小規模なメッセージの書き込み要求は持続されて正常に完了し、伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE に入れられます。
4. 伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE が満杯か、または書き込めない場合は、小規模なメッセージの書き込み要求は失敗し、それと同期して該当する戻りコードおよび理由コードが戻されます。 それに対して、大規模なメッセージの書き込み要求は持続されて正常に完了し、伝送キュー XQ1 に入れられます。 この場合、伝送キューへの小規模なメッセージの書き込みは試みられません。

大規模メッセージのフロー

1. 要求側アプリケーションは、大規模メッセージをリモート・キュー RQ1 に書き込みます。
2. キュー・マネージャー QMG1 は、伝送キュー XQ1 にそのメッセージを書き込みます。
3. キュー・マネージャー QMG1 上の送信側 MCA (S) は、伝送キュー XQ1 からそのメッセージを取り出してから、キュー・マネージャー QMG2 に送ります。
4. キュー・マネージャー QMG2 上の受信側 MCA (R) は、メッセージを受け取って宛先キュー LQ1 に入れます。
5. サービス提供側アプリケーションは、キュー LQ1 からそのメッセージを取り出して処理します。

小規模メッセージのフロー

1. 要求側アプリケーションは、小規模メッセージをリモート・キュー RQ1 に書き込みます。
2. キュー・マネージャー QMG1 は、伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE にそのメッセージを書き込みます。
3. キュー・マネージャー QMG2 上の IGQ は、メッセージを取り出して宛先キュー LQ1 に入れます。
4. サービス提供側アプリケーションは、キュー LQ1 からそのメッセージを取り出します。

注意事項

1. 要求側アプリケーションは、メッセージのデリバリーに使用される基礎を成すメカニズムを認識している必要はありません。
2. 小規模メッセージの場合のほうが、より高速なメッセージ・デリバリー・メカニズムを実現できる可能性があります。
3. メッセージ・デリバリーには、複数の経路を使用することができます (つまり、通常のチャネル経路とグループ内キューイング経路)。
4. 通常のチャネル経路よりも早く送れる可能性の高いグループ内キューイング経路のほうが選択されます。 メッセージの特性によっては、メッセージ・デリバリーは 2 つの経路にまたがって分割されることがあります。 したがって、メッセージが順序どおりに送達されるとは限りません (ただし、この送達は、メッセージが通常のチャネル経路のみを使用して送達される場合は可能です)。
5. 経路が選択され、メッセージが伝送キューに置かれた場合、メッセージの送達には選択された経路のみが使用されます。 SYSTEM.QSG.TRANSMIT.QUEUE 上の未処理のメッセージが、伝送キュー XQ1 上に配置変更されることはありません。

z/OS グループ内キューイングでのクラスター化 (複数のデリバリー・パス)

キュー・マネージャーを、キュー共用グループ化に加えてクラスター化されるように構成することができます。

メッセージがクラスター・キューに送信されて、ローカルおよびリモートの宛先キュー・マネージャーがいずれも同じキュー共用グループ内にあるときは、小規模メッセージのデリバリー (SYSTEM.QSG.TRANSMIT.QUEUE を使用して) にはグループ内キューイングが使用されます。また、グループ内キューイングがメッセージのサイズをサポートする場合、大規模メッセージのデリバリーにも使用されます。 また、クラスターには入っているけれどもキュー共用グループには入っていない任意のキュー・

マネージャーへのメッセージのデリバリーでも、SYSTEM.CLUSTER.TRANSMIT.QUEUE が使用されます。下の図は、そのような構成を示しています (ただし、チャネル・イニシエーターは示されていません)。

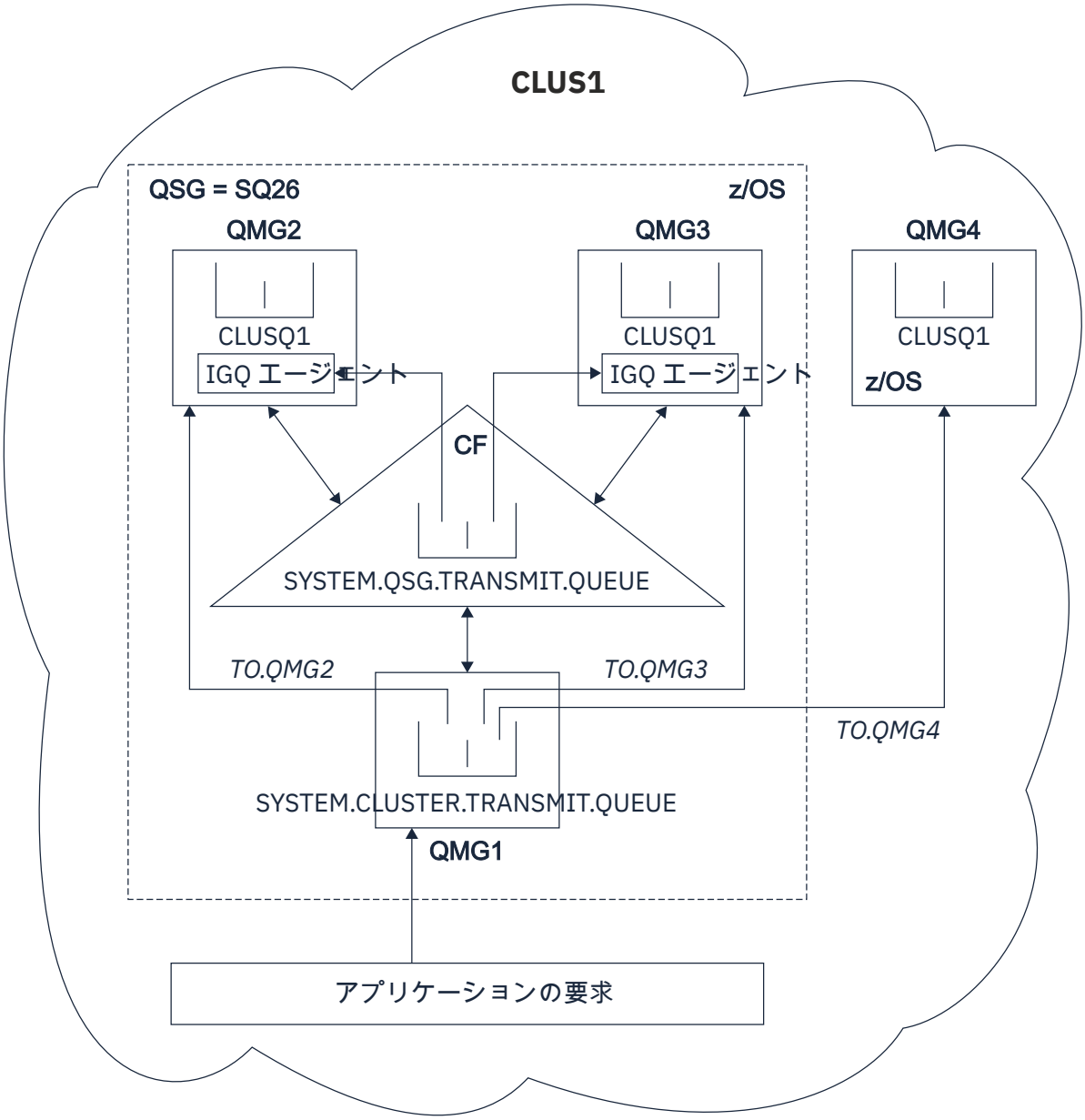


図 *70.* グループ内キューイングでのクラスター化の例

この図は以下を示しています。

- クラスター CLUS1 内に構成された 4 つの z/OS キュー・マネージャー QMG1、QMG2、QMG3、および QMG4。
- キュー共用グループ SQ26 内に構成されたキュー・マネージャー QMG1、QMG2、および QMG3。
- キュー・マネージャー QMG2 および QMG3 で実行されている IGQ エージェント。
- QMG1 内に定義されているローカル SYSTEM.CLUSTER.TRANSMIT.QUEUE。

  注 : 明瞭にするために、他のキュー・マネージャー上の SYSTEM.CLUSTER.TRANSMIT.QUEUE は表示されません。
- 共用 SYSTEM.QSG.TRANSMIT.QUEUE 。これは、CFLEVEL (3) RECOVER (YES) 属性で構成された IBM MQ 構造体内にあります。

- クラスター・チャネル TO.QMG2 (QMG1 を QMG2 に接続)、TO.QMG3 (QMG1 を QMG3 に接続)、および TO.QMG4 (QMG1 を QMG4 に接続)。
- キュー・マネージャー QMG2、QMG3、および QMG4 上でホスト・サービスを利用するクラスター・キュー CLUSQ1。

要求側アプリケーションが MQOO_BIND_NOT_FIXED オプションを使用してクラスター・キューをオープンしたとします。これは、書き込み時にクラスター・キュー用のターゲット・キュー・マネージャーが選択されるようにするためです。

選択されたターゲット・キュー・マネージャーが QMG2 であるとすると、次のようになります。

- 要求側アプリケーションによって書き込まれる大規模メッセージはすべて、次のように処理されます。
  - QMG1 の SYSTEM.CLUSTER.TRANSMIT.QUEUE に書き込まれます。なぜなら、SYSTEM.QSG.TRANSMIT.QUEUE は CFLEVEL(3) 構造体に存在するからです。そのため、メッセージのサイズは 63 KB までしかサポートされません。
  - クラスター・チャネル TO.QMG2 を使用して、QMG2 上のクラスター・キュー CLUSQ1 に転送されます。
- 要求側アプリケーションによって書き込まれる小規模メッセージはすべて、次のように処理されます。
  - 共用伝送キュー SYSTEM.QSG.TRANSMIT.QUEUE に書き込まれます。 このキューは、RECOVER(YES) 属性で構成される構造体に存在するので、持続の小メッセージおよび非持続の小メッセージの両方で使用されます。
  - QMG2 上の IGQ エージェントによって取り出されます。
  - QMG2 上のクラスター・キュー CLUSQ1 に書き込まれます。

選択されたターゲット・キュー・マネージャーが QMG4 であるとすると、次のようになります。

- QMG4 はキュー共用グループ SQ26 のメンバーではないため、要求側アプリケーションによって書き込まれるメッセージはすべて、次のように処理されます。
  - QMG1 上の SYSTEM.CLUSTER.TRANSMIT.QUEUE に書き込まれます。
  - クラスター・チャネル TO.QMG4 を使用して、QMG4 上のクラスター・キュー CLUSQ1 に転送されます。

注意事項

- 要求側アプリケーションは、メッセージのデリバリーに使用される基礎を成すメカニズムを認識している必要はありません。
- キュー共用グループ内のキュー・マネージャー間 (クラスター内の 同一のキュー・マネージャー間であっても) が小さい非持続性メッセージをやりとりする場合のほうが、より早いデリバリー・メカニズムを実現できる可能性が高くなります。
- メッセージ・デリバリーには、複数の経路を使用することができます (つまり、クラスター経路とグループ内キューイング経路)。
- クラスター経路よりも早く送れる可能性の高いグループ内キューイング経路のほうが選択されます。 メッセージの特性によっては、メッセージ・デリバリーは 2 つの経路にまたがって分割されることがあります。 したがってメッセージは、順序どおりに送達されるとは限りません。 重要な点として、この送達は、アプリケーションにより指定された MQOO_BIND_* オプションに関係なく可能です。 グループ内キューイングは、オープンで MQOO_BIND_NOT_FIXED、MQOO_BIND_ON_OPEN、MQOO_BIND_ON_GROUP、または MQOO_BIND_AS_Q_DEF が指定されるかどうかに応じて、クラスタリングと同じ方法でメッセージを配布します。
- 経路が選択され、メッセージが伝送キューに置かれた場合、メッセージの送達には選択された経路のみが使用されます。 SYSTEM.QSG.TRANSMIT.QUEUE 上の未処理のメッセージが、SYSTEM.CLUSTER.TRANSMIT.QUEUE 上に配置変更されることはありません。

z/OS クラスター化、グループ内キューイング、および分散キューイング

キュー共用グループに加えてクラスターのメンバーでもあって、しかも送信側/受信側のチャネル・ペアを使用して分散キューイング・マネージャーに接続されるようにキュー・マネージャーを構成することができます。

この構成は、分散キューイングとグループ内キューイングおよびクラスタリングとグループ内キューイングの組み合わせです。

グループ内キューイングは、226 ページの『グループ内キューイングでの分散キューイング (複数のデリバリー・パス)』で説明されています。

クラスタリングとグループ内キューイングは、228 ページの『グループ内キューイングでのクラスター化 (複数のデリバリー・パス)』で説明されています。

z/OS グループ内キューイング・メッセージ

このセクションでは、SYSTEM.QSG.TRANSMIT.QUEUE に書き込まれるメッセージについて説明します。

メッセージ構造
伝送キューに書き込まれる他のすべてのメッセージと同様、SYSTEM.QSG.TRANSMIT.QUEUE に書き込まれるメッセージには、伝送キュー・ヘッダー (MQXQH) が前に付けられます。

メッセージの持続性

IBM WebSphere MQ 5.3 以上では、共用キューは持続メッセージと非持続メッセージの両方をサポートしています。

IGQ エージェントが非持続性メッセージを処理している間にキュー・マネージャーが終了した場合、または IGQ エージェントがメッセージを処理しているときに異常終了した場合は、処理中の非持続メッセージは失われてしまいます。 アプリケーションは、非持続メッセージのリカバリーが必要であれば、非持続メッセージのリカバリーに対して備えておかなければなりません。

IGQ エージェントから出された非持続メッセージの書き込み要求で予期しない障害が起きた場合、処理中のメッセージは失われます。

メッセージの送達
IGQ エージェントは、同期点有効範囲外のすべての非持続メッセージ、および同期点スコープ範囲内のすべての持続メッセージの検索と送達を行います。 この場合、IGQ エージェントは同期点コーディネーターとして動作します。 そのため、IGQ エージェントは、メッセージ・チャネルで高速の非持続メッセージが処理されるように、非持続メッセージを処理します。 Fast, nonpersistent messages を参照してください。

メッセージのバッチ処理
IGQ エージェントは 50 個のメッセージという固定バッチ・サイズを使用します。 バッチ内で取得されるパーシスタント・メッセージは、50 個間隔でコミットされます。 エージェントは、SYSTEM.QSG.TRANSMIT.QUEUE で検索できるメッセージがなくなると、持続メッセージから成るバッチをコミットします。

メッセージ・サイズ
SYSTEM.QSG.TRANSMIT.QUEUE に書き込めるメッセージの最大サイズは、共用キューでサポートされている最大メッセージ長から、伝送キュー・ヘッダー (MQXQH) の長さを引いたものになります。

デフォルトのメッセージ持続性とデフォルトのメッセージ優先順位
SYSTEM.QSG.TRANSMIT.QUEUE が、オープン時に設定されたキュー・ネーム・レゾリューション・パスにある場合、デフォルトのパーシスタンスおよび優先度 (またはデフォルトのパーシスタンスまたは優先度) で配置されたメッセージには、使用されているデフォルトの優先度とパーシスタンスが指定されたキューの選択に、通常のルールが適用されます (キューの選択のルールについては、「IBM MQ メッセージ」セクションを参照してください)。

関連概念

232 ページの『未配布メッセージ/未処理メッセージ』
このトピックでは、SYSTEM.QSG.TRANSMIT.QUEUE の未配布および未処理のメッセージの処理について説明します。

232 ページの『レポート・メッセージ - グループ内キューイング』

このトピックでは、レポート・メッセージについて説明します。これには、着信の確認、デリバリーの確認、有効期限レポート、および例外レポートが含まれます。

z/OS 未配布メッセージ/未処理メッセージ

このトピックでは、SYSTEM.QSG.TRANSMIT.QUEUE の未配布および未処理のメッセージの処理について説明します。

IGQ エージェントは、宛先キューにメッセージを送達できないと、以下を行います。

- MQRO_DISCARD_MSG レポート・オプションを優先させて (未配布メッセージの MQMD の Report オプション・フィールドでそのように指定されている場合)、未配布メッセージを破棄します。
- メッセージがまだ破棄されていなければ、宛先キュー・マネージャーの送達不能キュー上への未配布メッセージの配置を試みます。 IGQ エージェントは、メッセージの前に送達不能キュー・ヘッダー (MQDLH) を添付します。

送達不能キューが定義されていないか、または送達不能キューに未配布メッセージを書き込めない場合は、次のようになります。

- 未配布メッセージが持続メッセージの場合は、IGQ エージェントは処理中の持続メッセージの現行バッチをバックアウトして、再試行の状態に入ります。 詳細については、233 ページの『グループ内キューイングの特定のプロパティー』を参照してください。
- 未配布メッセージが非持続メッセージの場合は、IGQ エージェントはメッセージを廃棄して、次のメッセージの処理を継続します。

関連した IGQ エージェントがすべてのメッセージを処理し終わるのに間に合わないでキュー共有グループ内のキュー・マネージャーが終了すると、未処理メッセージは、そのキュー・マネージャーが次回始動されるまで SYSTEM.QSG.TRANSMIT.QUEUE に残ります。 その時点で IGQ エージェントは、メッセージを取り出して宛先キューに送達します。

SYSTEM.QSG.TRANSMIT.QUEUE 上のすべてのメッセージが処理される前に Coupling Facility に障害が起きた場合、未処理の非持続メッセージはいずれも失われます。

IBM は、アプリケーションが伝送キューにメッセージを直接書き込まないようにすることをお勧めします。アプリケーションが SYSTEM.QSG.TRANSMIT.QUEUE にメッセージを直接入れた場合、IGQ エージェントはそれらのメッセージを処理できないことがあります。その場合、メッセージは SYSTEM.QSG.TRANSMIT.QUEUE に残されます。 すると、ユーザーは独自の手段を使用して未処理メッセージを処理しなければなりません。

z/OS レポート・メッセージ - グループ内キューイング

このトピックでは、レポート・メッセージについて説明します。これには、着信の確認、デリバリーの確認、有効期限レポート、および例外レポートが含まれます。

着信の確認 (COA)/デリバリーの確認 (COD) のレポート・メッセージ
グループ内キューイングの使用時に COA メッセージと COD メッセージがキュー・マネージャーによって生成されます。

有効期限レポート・メッセージ
有効期限レポート・メッセージがキュー・マネージャーによって生成されます。

例外レポート・メッセージ

未配布メッセージのメッセージ記述子のレポート・オプション・フィールドに MQRO_EXCEPTION_* レポート・オプションが指定されていると、 IGQ エージェントは必要な例外レポートを生成し、指定された応答先キューに入れます。 グループ内キューイングを使用して、宛先の応答先キューに例外レポートを送達することができます。

レポート・メッセージの持続性は、未配布メッセージの持続性と同じです。 IGQ エージェントは、宛先応答先キューの名前を解決できない場合や (宛先応答キューへのその後の転送のための) 伝送キューへの応答メッセージの書き込みに失敗した場合、レポート・メッセージの生成元のキュー・マネージャーの送達不能キューへの例外レポートの書き込みを試みます。 可能でない場合、未配布メッセージは、次のように処理されます。

- 未配布メッセージが持続メッセージの場合は、IGQ エージェントは例外レポートを廃棄し、メッセージの現行バッチをバックアウトして、再試行の状態に入ります。 詳細については、233 ページの『グループ内キューイングの特定のプロパティー』を参照してください。
- 未配布メッセージが非持続メッセージの場合は、IGQ エージェントは例外レポートを廃棄して、SYSTEM.QSG.TRANSMIT.QUEUE 上の次のメッセージの処理を継続します。

z/OS グループ内キューイングのセキュリティー

このトピックでは、グループ内キューイングのセキュリティー構造について説明します。

キュー・マネージャー属性の IGQAUT (IGQ 権限) および IGQUSER (IGQ エージェントのユーザー ID) を設定して、IGQ エージェントが宛先キューをオープンするときに行われるセキュリティー検査のレベルを制御することができます。

グループ内キューイングの権限 (IGQAUT)

IGQAUT 属性を設定して、セキュリティー・タイプの検査を指定することができます。それによって、IGQ エージェントが宛先キューにメッセージを書き込む権限の確立時に使用するユーザー ID を指定します。

IGQAUT 属性は、チャネル定義で使用できる PUTAUT 属性に似ています。

グループ内キューイングのユーザー ID (IGQUSER)

IGQUSER 属性を設定して、IGQ エージェントが宛先キューにメッセージを書き込む権限を確立するときに使用するユーザー ID を指定することができます。

IGQUSER 属性は、チャネル定義で使える MCAUSER 属性に似ています。

z/OS グループ内キューイングの特定のプロパティー

このセクションでは、グループ内キューイングの特定のプロパティー (オブジェクト処理の無効化、グループ内キューイング・エージェントの自己リカバリーおよび再試行機能、およびグループ内キューイング・エージェントと直列化など) について説明します。

オブジェクト処理の無効化 (MQRC_OBJECT_CHANGED)

オブジェクトのオープン後にそのオブジェクトの属性が変更されたことが検出された場合、キュー・マネージャーは次のそのオブジェクトの使用時に MQRC_OBJECT_CHANGED を使用してオブジェクト処理を無効にします。

グループ内キューイングでは、オブジェクト処理の無効化に関して次のような規則が導入されています。

- オープン時にグループ内キューイングが使用可能になっていたためにオープン処理中に解決パス内に SYSTEM.QSG.TRANSMIT.QUEUE が組み込まれたのに、書き込みの時点でそのグループ内キューイングが使用禁止になっていることが検出されると、キュー・マネージャーはそのオブジェクト処理を無効にし、MQRC_OBJECT_CHANGED を使用して書き込み要求を失敗させます。
- オープン時にグループ内キューイングが使用禁止になっていたためにオープン処理中に解決パス内に SYSTEM.QSG.TRANSMIT.QUEUE が組み込まれなかったのに、書き込みの時点でグループ内キューイングが使用可能になっていることが検出されると、キュー・マネージャーはそのオブジェクト処理を無効にし、MQRC_OBJECT_CHANGED を使用して書き込み要求を失敗させます。
- オープン時にグループ内キューイングが使用可能になっていたためにオープン処理中に解決パス内に SYSTEM.QSG.TRANSMIT.QUEUE が組み込まれたのに、書き込みの時点までに SYSTEM.QSG.TRANSMIT.QUEUE 定義が変更されていることが検出されると、キュー・マネージャーはそのオブジェクト処理を無効にし、MQRC_OBJECT_CHANGED を使用して書き込み要求を失敗させます。

グループ内キューイング・エージェントの自己リカバリー

IGQ エージェントが異常終了した場合、メッセージ CSQM067E が発行され、IGQ エージェントは再度開始します。

グループ内キューイング・エージェントのリカバリー機能

IGQ エージェントが SYSTEM.QSG.TRANSMIT.QUEUE にアクセスするときに問題が発生した場合 (例えば、定義されていない、間違った属性で定義されている、Gets で使用禁止になっているなどの理由により)、IGQ エージェントは再試行の状態になります。

IGQ エージェントは、短期および長期の再試行カウントおよび再試行間隔に従います。 変更不能のこのカウントと間隔の値は、次のとおりです。

表 *20.* 短期および長期の再試行カウントおよび再試行間隔の値

定数	**値**
短期再試行カウント	10
短期再試行間隔	60 秒 = 1 分
長期再試行カウント	999,999,999
長期再試行間隔	1200 秒 = 20 分

グループ内キューイング・エージェントと直列化

対等リカバリーがまだ進行中のときに IGQ エージェントが共用キューへのアクセスを直列化しようとすると、失敗する可能性があります。

IGQ エージェントが共用キューまたはキュー上のコミットされていないメッセージを扱っているときに キュー共用グループ内のキュー・マネージャーで障害が起きた場合、IGQ エージェントは終了し、障害のあるキュー・マネージャーの共用キュー対等リカバリーが行われます。 共用キュー対等リカバリーは非同期アクティビティーであるため、共用キュー対等リカバリーが完了する前に、障害のあるキュー・マネージャーおよびそのキュー・マネージャーの IGQ エージェントが再始動する可能性があります。 そのため、コミットされたメッセージが、リカバリーされているメッセージとの順番に関係なく先に処理される場合があります。 メッセージが違う順序で処理されないようにするために、IGQ エージェントは、MQCONNX API 呼び出しを発行することによって、共用キューへのアクセスを直列化します。

対等リカバリーがまだ進行中のときに IGQ エージェントが共用キューへのアクセスを直列化しようとすると、失敗する可能性があります。 エラー・メッセージが出されて、IGQ エージェントは再試行の状態に入ります。 キュー・マネージャー対等リカバリーが完了すると、IGQ エージェントは、例えば次の再試行時に再始動されます。

z/OS z/OS でのストレージ管理

IBM MQ for z/OS は、永続的および一時的なデータ構造を必要としており、ページ・セットおよびメモリー・バッファーを使用してこのデータを保管します。 これらのトピックでは、IBM MQ がこれらのページ・セットおよびバッファーを使用する方法について詳しく説明します。

関連概念

235 ページの『IBM MQ for z/OS のページ・セット』
このトピックでは、IBM MQ for z/OS がページ・セットを使用してメッセージを格納するしくみを説明します。

235 ページの『IBM MQ for z/OS のストレージ・クラス』
ストレージ・クラスは、キュー・マネージャーがキューをページ・セットにマップすることを可能にする IBM MQ for z/OS の概念です。 ストレージ・クラスを使用すると、どのデータ・セットをどのキューが使用するかを制御できます。

237 ページの『IBM MQ for z/OS のバッファーおよびバッファー・プール』
IBM MQ for z/OS は、バッファーおよびバッファー・プールを使用して、一時的にデータをキャッシュに入れます。 このトピックでは、バッファーの編成および使用方法について詳しく説明します。

関連資料

239 ページの『IBM MQ for z/OS のストレージ管理に関する詳細情報の参照先』
このトピックでは、IBM MQ for z/OS のストレージ管理の詳細を参照することができます。

z/OS IBM MQ for z/OS のページ・セット

このトピックでは、IBM MQ for z/OS がページ・セットを使用してメッセージを格納するしくみを説明します。

ページ・セット は、IBM MQ で使用されるように特別にフォーマットされている VSAM 線形データ・セットです。 ページ・セットは、ほとんどのメッセージおよびオブジェクト定義を保管するために使用されます。

ただし、グローバル定義 (Db2 の共用リポジトリーに保管される) と共用キューにあるメッセージは例外です。 これらは、キュー・マネージャーのページ・セットには格納されません。 共用キューの詳細については、「175 ページの『共用キューとキュー共用グループ』」を参照してください。グローバル定義について詳しくは、「プライベート定義およびグローバル定義」を参照してください。

IBM MQ ページ・セットのサイズは、最大で 64 GB です。 各ページ・セットは、ページ・セット ID (PSID)、つまり、00 以上 99 以下の範囲の整数によって識別されます。 各キュー・マネージャーには、独自のページ・セットがなければなりません。

IBM MQ は、ページ・セット 0 (PSID=00) を使用して、オブジェクト定義、およびキュー・マネージャーに関連するその他の重要情報を格納します。 IBM MQ の通常の運用では、ページ・セット 0 がいっぱいになってはならないので、それをメッセージの格納に使用しないでください。

システムのパフォーマンスを向上させるために、有効期間の短いメッセージと長いメッセージを、 別々のページ・セットに置いて区別する必要もあります。

ページ・セットをフォーマットすることが必要です。IBM MQ には、そのための FORMAT ユーティリティーが用意されています。ページ・セットのフォーマット (FORMAT) を参照してください。 ページ・セットを IBM MQ サブシステムに対して定義することも必要です。

ページ・セットがいっぱいになった場合に動的に拡張するように、IBM MQ for z/OS を構成することができます。 IBM MQ は、必要な場合に最大で 123 個の論理エクステントになるまでページ・セットの拡張を続けます (ただし、ディスク・ストレージ・スペースが十分に存在することが前提です)。 線形データ・セットがこのように定義されていれば、エクステントはボリュームをまたがってもかまいません。ただし、IBM MQ では 64 GB を超えてページ・セットを拡張することはできません。

ある IBM MQ キュー・マネージャーのページ・セットを、別の IBM MQ キュー・マネージャーで使用したり、そのキュー・マネージャー名を変更することはできません。 データをあるキュー・マネージャーから別のキュー・マネージャーに転送する場合には、すべてのオブジェクトおよびメッセージを最初のキュー・マネージャーからアンロードし、 別のキュー・マネージャーに再ロードしなければなりません。

V6 より前のリリースを実行するキュー・マネージャーで、4 GB を超えるページ・セットを使用することはできません。 マイグレーション中に、前のリリースの コードに戻さなければならない可能性がある場合は、次の点を考慮します。

- ページ・セット 0 を 4 GB より大きくなるように変更しないでください。
- 4 GB より大きい他のページ・セットは、前のリリースを使用してキュー・マネージャーを再始動したときにオフラインのままになります。

4 GB を超える拡張が可能な既存のページ・セットを移行するための詳細については、『4 GB より大きくするためのページ・セットの定義』を参照してください。

管理者は、実行中のキュー・マネージャーにページ・セットを動的に追加したり、実行中のキュー・マネージャーからページ・セットを削除したりすることができます (ただし、ページ・セット 0 は例外です)。 DEFINE PSID コマンドに DSN キーワードが含まれている場合のみ、キュー・マネージャーの再始動が完了した後で、このコマンドを実行することができます。

z/OS IBM MQ for z/OS のストレージ・クラス

ストレージ・クラスは、キュー・マネージャーがキューをページ・セットにマップすることを可能にする IBM MQ for z/OS の概念です。 ストレージ・クラスを使用すると、どのデータ・セットをどのキューが使用するかを制御できます。

ストレージ・クラスの概要

ストレージ・クラス は、1 つまたは複数のキューを 1 つのページ・セットにマップします。 つまり、特定のキューへのメッセージは、そのキューのページ・セットに保管されることになります。

ストレージ・クラスでは、管理、データ・セット・スペースとロードの管理、またはアプリケーション分離の目的で、 非共用メッセージ・データを格納する位置を制御できます。 IMS ブリッジを使用している場合は、ストレージ・クラスを使用して、 IMS 領域の XCF グループとメンバー名を定義することもできます (293 ページの『IBM MQ と IMS』 を参照)。

共用キューでは、そこに入れられたメッセージはページ・セットに格納されないため、 ストレージ・クラスを使用してページ・セットのマッピングを入手することはありません。

ストレージ・クラスの機能

- ストレージ・クラスは、DEFINE STGCLASS コマンドを使用し、 ページ・セット ID (PSID) を指定することによって定義します。
- キューを定義する場合には、STGCLASS 属性にストレージ・クラスを指定できます。

次の例では、ストレージ・クラス ARC2 全体で、 ローカル・キュー QE5 がページ・セット 21 にマップされています。

```
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE QLOCAL(QE5) STGCLASS(ARC2)
```

つまり、キュー QE5 に書き込まれるメッセージは、ページ・セット 21 に格納されるということです (DASD に書き込めるだけの十分な期間、キューに置かれた場合)。

複数のキューで同じストレージ・クラスを使用でき、また必要な数だけ、ストレージ・クラスを定義できます。 例えば、次のようにして以前の例を拡張し、さらに多くのストレージ・クラスおよびキュー定義を組み込むことができます。

```
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) ...
```

237 ページの図 71 では、ストレージ・クラス ARC1 および MAXI はどちらも、 ページ・セット 05 に関連付けられています。 したがって、キュー QE1、QE2、および QE3 は、 ページ・セット 05 にマップされます。 同様に、ストレージ・クラス ARC2 は、キュー QE4 および QE5 をページ・セット 21 に関連付けています。

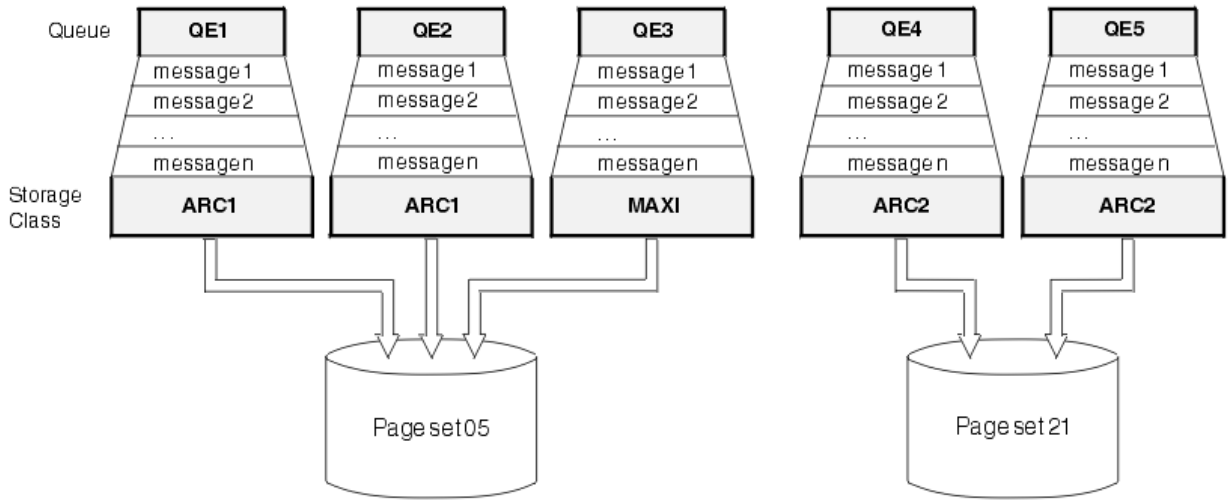


図 71. ストレージ・クラスを経由するページ・セットへのキューのマッピング

ストレージ・クラスを指定せずにキューを定義すると、IBM MQ はデフォルトのストレージ・クラスを使用します。

存在しないストレージ・クラスを指定するキューにメッセージを書き込むと、 アプリケーションはエラーを受け取ります。 この場合、キュー定義を変更して既存のストレージ・クラス名を指定するか、 キューで指定されたストレージ・クラスを作成する必要があります。

ストレージ・クラスを変更できるのは、次の場合に限ります。

- このストレージ・クラスを使用するすべてのキューが空で、 コミットされていない活動がない場合。
- このストレージ・クラスを使用するすべてのキューが閉じられている場合。

z/OS IBM MQ for z/OS のバッファーおよびバッファー・プール

IBM MQ for z/OS は、バッファーおよびバッファー・プールを使用して、一時的にデータをキャッシュに入れます。 このトピックでは、バッファーの編成および使用方法について詳しく説明します。

IBM MQ では、効率のためにキャッシュ方式を使用しており、それによりメッセージ (およびオブジェクト定義) を DASD 内のページ・セットに保管する前に、一時的にバッファーに保管します。 有効期間が短いメッセージ、つまり、受信されるとすぐにキューから取り出されるメッセージは、 バッファーにはそのまま保管しておくことができます。 このキャッシング・アクティビティーは、IBM MQ のコンポーネントであるバッファー管理機能によって制御されます。

バッファーは、バッファー・プール に編成されます。 キュー・マネージャーごとに最大 100 個 (0 から 99 まで) のバッファー・プールを定義できます。

238 ページの図 72 に概説されているオブジェクトおよびメッセージ・タイプ分離、およびご使用のアプリケーションで定義されているデータ分離要件に合った、最小数のバッファー・プールを使用することをお勧めします。 各バッファーの長さは、4 KB です。 バッファー・プールはデフォルトでは 31 ビット・ストレージを使用します。 このモードでは、バッファーの最大数は、キュー・マネージャー・アドレス・スペース内の使用可能な 31 ビット・ストレージの量によって決まります。 ただし、約 70% より多くをバッファーとして使用することはありません。 あるいは、64 ビット・ストレージからバッファー・プール・ストレージを割り振ることもできます (**DEFINE BUFFPOOL** コマンドの LOCATION 属性を使用)。
LOCATION(ABOVE) を使用して 64 ビット・ストレージを使用することには、次の 2 つの利点があります。まず、格段に大きな 64 ビット・ストレージを使用できるため、バッファー・プールを大きくすることができます。 次に、31 ビット・ストレージを他の機能に使用できるようになります。 一般に、バッファーが大きければ大きいほど、 バッファリングはより効率的になり、IBM MQ のパフォーマンスもより向上します。

238 ページの図 72 には、メッセージ、バッファー、 バッファー・プール、およびページ・セットの関連が示されています。 バッファー・プールは、1 つ以上のページ・セットに関連付けられ、 各ページ・セットは、1 つのバッファー・プールに関連付けられています。

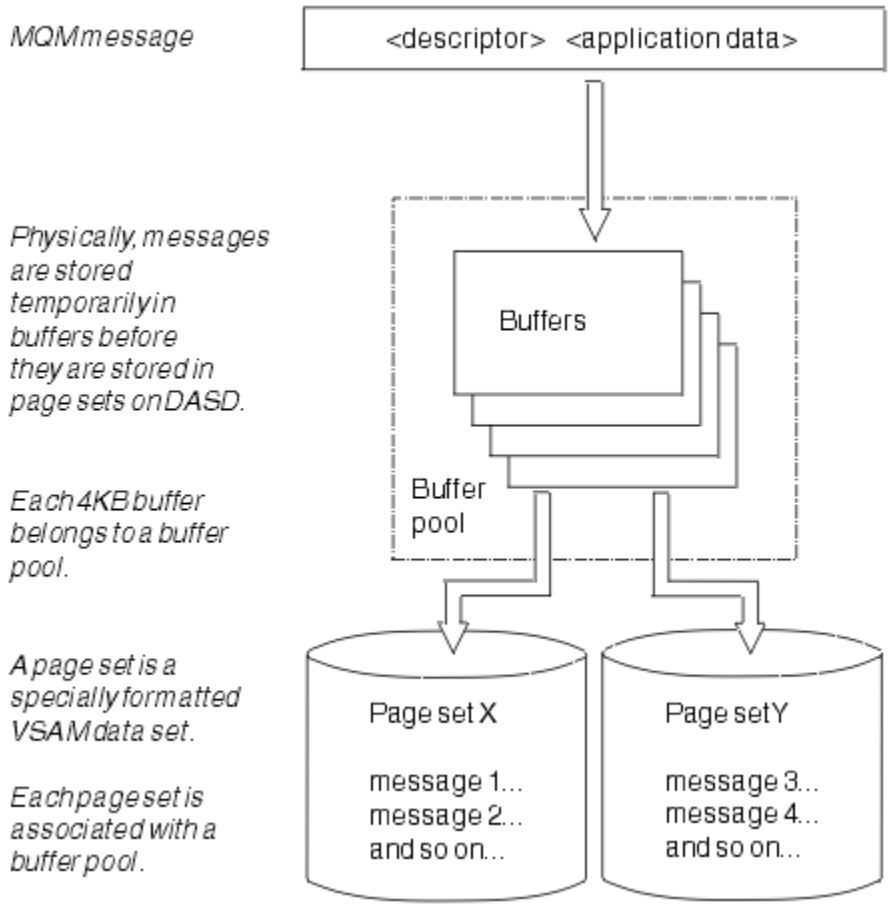


図 72. バッファー、バッファー・プール、およびページ・セット

ALTER BUFFPOOL コマンドを使用して、バッファー・プール・サイズおよび場所を変更するためのコマンドを動的に発行できます。 ページ・セットは、**DEFINE PSID** コマンドを使用して動的に追加したり、**DELETE PSID** コマンドを使用して動的に削除したりできます。

バッファー・プールが小さすぎると、IBM MQ はメッセージ CSQP020E を発行します。 その際、影響を受けたバッファー・プールにバッファーを動的に追加することができます (それには、 他のバッファー・プールからバッファーを除去しなければならない場合があります)。

プール内のバッファーの数を指定する場合は **DEFINE BUFFPOOL** コマンドを使用し、バッファー・プールを動的にサイズ変更するには **ALTER BUFFPOOL** コマンドを使用します。 プール内の現在のバッファーの数は、**DISPLAY USAGE** コマンドを使用して バッファー・プールを使用しているページ・セットを表示することにより動的に決定されます。

パフォーマンス上の理由により、メッセージとオブジェクト定義を、 同じバッファー・プールに入れないようにします。 ページ・セット 0 用に 1 つのバッファー・プール (例えば、ゼロ番) を使用してください。 ここに、オブジェクト定義が保管されます。 同様に、有効期間の短いメッセージと長いメッセージを別々のバッファー・プールに保管し、 別々のページ・セット、および別々のキューに保管するようにしてください。

DEFINE BUFFPOOL コマンドは、新規バッファー・プールを作成するために 再始動の後で使用することはできません。 代わりに、**DEFINE PSID** コマンドが DSN キーワードを使用する場合、現在定義されていないバッファー・プールを明示的に識別することができます。 その新規バッファー・プールが作成されます。

z/OS IBM MQ for z/OS のストレージ管理に関する詳細情報の参照先

このトピックでは、IBM MQ for z/OS のストレージ管理の詳細を参照することができます。

このセクションのトピックに関する詳細情報は、以下の資料から入手できます。

表 *21.* ストレージ管理についての詳細

トピック	参照先
必要なストレージの量	z/OS のストレージ要件とパフォーマンス要件の計画
どのくらいの大きさでページ・セットや バッファ・プールを作成するか	ページ・セットとバッファー・プールの計画
ページ・セットの管理	ページ・セットの管理
MQSC コマンド	MQSC コマンド

z/OS IBM MQ for z/OS でのロギング

IBM MQ は、データの変更と重要なイベントが発生した場合に、そのログ を維持します。 これらのログは、必要に応じてデータを以前の状態にリカバリーするために使用できます。

ブートストラップ・データ・セット (BSDS) は、 ログが入ったデータ・セットについての情報を保管します。

ログには、統計、トレース、およびパフォーマンス評価に関する情報は含まれません。 IBM MQ が収集する統計情報とモニター情報の詳細については、モニターと統計を参照してください。

ロギングの詳細については、以下のトピックを参照してください。

- 239 ページの『IBM MQ for z/OS のログ・ファイル』
- 244 ページの『ログの構造体』
- 244 ページの『IBM MQ for z/OS ログが書き込まれる方法』
- 247 ページの『より大きなログ相対バイト・アドレス』
- 248 ページの『ブートストラップ・データ・セット』

関連タスク

ロギング環境の計画

システム・パラメーター・モジュールを使用したログの設定

z/OS z/OS の管理

関連資料

z/OS IBM MQ for z/OS のメッセージ

z/OS IBM MQ for z/OS のログ・ファイル

ログ・ファイルには、トランザクションのリカバリーに必要な情報が入ります。 アクティブ・ログ・ファイルをアーカイブに保存して、ログ・データを長期間保存することも可能です。

ログ・ファイルについて

IBM MQ は、重要なイベントが発生した場合に、それらすべてを活動ログ に記録します。 ログには、回復に必要な次の情報が含まれます。

- 持続メッセージ
- IBM MQ オブジェクト (キューなど)

- IBM MQ キュー・マネージャー

活動ログは、循環的に使用されるデータ・セットの集合 (310 個まで) で構成されます。

ログの保存を有効にすることによって、活動ログに書き込みが行われるときに、保存データ・セットにコピーが作成されるようにすることができます。 ログの保存により、ログ・データを長期間保持することが可能になります。 ログの保存を使用しない場合、ログは折り返されて、古いデータから上書きされます。 ページ・セットを回復する場合、または CF 構造のデータを回復する場合、ページ・セットまたは構造体のバックアップが取られたときからのログ・データが必要です。 保存ログはディスクまたはテープに作成できます。

保存

活動ログのサイズは固定されているため、IBM MQ はそれぞれのログ・データ・セットの 内容を定期的に保存ログ にコピーします。 通常、保存ログは、直接アクセス・ストレージ・デバイス (DASD) または磁気テープに保管される データ・セットです。 サブシステムまたはトランザクションに障害が発生した場合、IBM MQ は回復のために活動ログを利用し、 必要に応じて保存ログも利用します。

保存ログは、最大で 1000 個の順次データ・セットから構成されます。 各データ・セットは、z/OS 統合カタログ機能 (ICF) を使ってカタログできます。

IBM MQ 回復のために、保存は不可欠です。 回復単位が長時間実行されていた場合、その回復単位のログ・レコードが、保存ログに入っていることがあります。 その場合、回復のためには保存ログのデータが必要とされます。 しかし、保存を使用不可にすると、新しいログ・レコードを含んでいる活動ログが折り返され、それ以前のログ・レコードは上書きされてしまいます。 したがって、IBM MQ は回復単位をバックアウトできない可能性があり、 メッセージは失われるかもしれません。 その後、キュー・マネージャーは異常終了する。

このため、実稼働環境では、**決して保存をオフに切り替えないでください**。 もしオフにすると、システムやトランザクションに障害が起きた場合にデータを失う危険があります。 テスト環境で実行している場合にのみ、保存をオフにすることを考慮できます。 そのようにする必要がある場合は、CSQ6LOGP マクロを使用します。そのマクロの説明については、CSQ6LOGP の使用を参照してください。

IBM MQ では、無計画の長期実行作業単位に関する問題を回避するために、アクティブ・ログのオフロード処理の実行時に長期実行作業単位が検出されると、メッセージ (CSQJ160I または CSQJ161I) が生成されます。

重複ロギング

重複ロギングでは、各ログ・レコードが 2 つのアクティブ・ログ・データ・セットに書き込まれ、再始動時にデータ損失の問題が発生する可能性が最小化されます。

IBM MQ は、単一ロギング または重複ロギング のいずれかで実行するように構成できます。 単一ロギングにすると、ログ・レコードは 1 つの活動ログ・データ・セットに一度だけ書き込まれます。 各活動ログ・データ・セットは、単一エクステントの VSAM 線形データ・セット (LDS) です。 重複ロギングの場合は、それぞれのログ・レコードは 2 つの別個の活動ログ・データ・セットに書き込まれます。 重複ロギングによって、 再始動時にデータが失われる可能性を最小限に抑えることができます。

ログ延期

ログ延期は、一部の作業単位のログ・レコードをログの後の方に書き込みます。 これにより、実行時間の長いまたは長期間未確定の作業単位について、 キュー・マネージャーの再始動時またはバックアウト時に読み取る必要があるログ・データの量が削減されます。

作業単位が長くなると考えられるとき、各ログ・レコードの表示は ログの後の方に書き込まれます。 この手法は、延期 として認識されています。 作業単位全体が処理されると、その作業単位は延期された 状態になります。 延期された作業単位に関連するバックアウトや再始動活動では、元の作業単位のログ・レコードを使用する代わりに、 延期ログ・レコードを使用することができます。

実行時間の長い作業単位を検出する処理は、チェックポイント処理の機能です。 チェックポイント時に、延期が必要かどうかを決定するために、活動状態の各作業単位が検査されます。 作業単位は、作成以降または最後の延期以降に、前の 2 つのチェックポイントを 通過すると、延期に適した作業単位となります。 つまり、1 つの作業単位は複数回にわたって延期される可能性があります。 これは、複数延期 作業単位と呼ばれます。

作業単位は、3 回のチェックポイント処置ごとに延期されます。 ただし、チェックポイント処理は、ログ・スイッチ (つまり、LOGLOAD の超過を引き起こしたログ・レコードの書き込み) と同期せずに実行されます。

一度に実行できるチェックポイント処理は 1 つだけなので、1 つのチェックポイント処理が完了する前に複数のログ・スイッチが発生する可能性があります。

したがって、アクティブ・ログの数が十分にない場合や、アクティブ・ログが小さすぎる場合は、すべてのログが満杯になる前に大きな作業単位の延期が終了しない場合があります。

延期を完了できない場合は、メッセージ CSQR027I が生成されます。

ログのアーカイブがオフになっている状態で、延期が失敗した作業単位をバックアウトしようとすると、理由コード 00D1032A の ABEND 5C6 が発生します。 この問題を回避するには、OFFLOAD=YES を使用する必要があります。

ログの延期の機能は常にアクティブになっており、ログのアーカイブ保存が有効になっているかどうかにかかわりなく実行されます。

注 : ある作業単位のログ・レコードがすべて延期されても、各レコードの内容全体は延期されません。 バックアウトに必要な部分だけ延期されます。 つまり、書き込まれるログ・データの量は最小に保たれて、ページ・セットの障害が発生した場合に延期レコードは使用できません。 実行時間の長い作業単位とは、3 つより多くのキュー・マネージャー・チェックポイントに対して 実行されている作業単位です。

ログの延期の詳細については、『ログの管理』を参照してください。

ログ圧縮

ログ・レコードをログ・データ・セットに書き込んだりそこから読み取ったりする際に圧縮や解凍が行えるように IBM MQ for z/OS を構成することができます。

ログ圧縮を使用すれば、専用キューの持続メッセージに関するログに書き込まれるデータの量を削減できます。 実際に行われる圧縮の量は、メッセージに含まれているデータのタイプによって異なります。 例えば、ラン・レングス・エンコード (RLE) は、バイトの反復インスタンスを圧縮することによって機能します。 これは、構造化データやレコード指向データの場合に効率の良い結果をもたらします。

重要 : 共用キューに書き込まれる持続メッセージは、ログ圧縮の対象ではありません。

達成されるデータ圧縮の量は、システム管理機能 115 (SMF) レコードの「ログ管理プログラム」セクション内のフィールドを使用してモニターできます。 SMF について詳しくは、システム管理機能の使用およびアカウンティング・メッセージと統計メッセージを参照してください。

ログ圧縮を使用すると、システムのプロセッサーの使用率が上がります。 圧縮の使用は、ログ・データ・セットへの書き込みを行う入出力帯域幅によってキュー・マネージャーのスループットが制約されているか、またはログ・データ・セットの保持に必要なディスク・ストレージによる制約を受けている場合のみ検討してください。 共用キューを使用している場合は、キュー共用グループにキュー・マネージャーをさらに追加し、ワークロードの分散先のキュー・マネージャーの数を増やすことで、入出力帯域幅の制約を軽減することができます。

ログ圧縮オプションは、キュー・マネージャーを停止および再始動することなく、必要に応じて使用可能および使用不可にすることができます。 キュー・マネージャーは、現行のログ圧縮設定に関係なく、どの圧縮ログ・レコードも読み取ることができます。

キュー・マネージャーは、ログ圧縮に関して 3 つの設定をサポートしています。

NONE
ログ・データの圧縮を使用しません。 これがデフォルト値です。

RLE
ラン・レングス・エンコード (RLE) を使用してログ・データ圧縮を実行します。

ANY
キュー・マネージャーが、最大の圧縮率でログ・レコード圧縮を行う圧縮アルゴリズムを選択できるようにします。 このオプションを指定すると、RLE 圧縮が行われます。

ログ・レコードの圧縮は、以下のいずれかを使用して制御できます。

- MQSC の SET LOG コマンドと DISPLAY LOG コマンド。『SET LOG』と『DISPLAY LOG』を参照してください。
- PCF インターフェースの Set Log 機能と Inquire Log 機能。『Set log』と『Inquire log』を参照してください。
- システム・パラメーター・モジュールの CSQ6LOGP マクロ。『CSQ6LOGP の使用』を参照してください。

加えて、ログ印刷ユーティリティー CSQ1LOGP もすべてのタイプの圧縮ログ・レコードの解凍をサポートします。

ログ・データ

ログには、最大で 18,000,000,000,000,000,000 ($1.8*10^{19}$) バイトを格納できます。 各バイトは、ログの先頭からのオフセットによってアドレス指定できます。 このオフセットは、相対バイト・アドレス (RBA) と呼ばれます。

RBA は、6 バイトのログ RBA または 8 バイトのログ RBA のどちらを使用しているかに応じて、2^{48} バイトの合計アドレス可能範囲を与える 6 バイト・フィールドまたは 2^{64} バイトの合計アドレス可能範囲を与える 8 バイト・フィールドによって参照されます。

ただし、使用されている範囲が F00000000000 (6 バイト RBA が使用されている場合) または FFFF800000000000 (8 バイトのログ RBA が使用されている場合) を超えることを IBM MQ が検出すると、メッセージ CSQI045、CSQI046、CSQI047、および CSQJ032 が発行され、ログ RBA をリセットするように警告されます。

RBA 値が FFF800000000 (6 バイトのログ RBA が使用されている場合) または FFFFFFC000000000 (8 バイトのログ RBA が使用されている場合) に到達すると、キュー・マネージャーは理由コード 00D10257 で終了します。

使用ログ範囲に関する警告メッセージが出された場合は、キュー・マネージャーの停止を計画する必要があります。停止期間中に、8 バイトのログ RBA を使用するようにキュー・マネージャーを変換するか、ログをリセットできます。 ログをリセットする手順は、キュー・マネージャーのログのリセットに記載されています。

キュー・マネージャーが 6 バイトのログ RBA を使用している場合、キュー・マネージャーのログをリセットするよりも、より大きなログ相対バイト・アドレスの実装で説明されている手順に従って、8 バイトのログ RBA を使用するようキュー・マネージャーを変換することを考慮してください。

ログにはログ・レコード が含まれています。それぞれのログ・レコードは、1 つの単位として処理されるログ・データの集合です。 各ログ・レコードは、そのヘッダーの第 1 バイトの RBA、または ログ・レコード・シーケンス番号 (LRSN) によって識別されます。 RBA または LRSN は、ログ内の特定ポイントで始まるレコードを固有に識別します。

ログ・ポイントを識別するために RBA または LRSN を使用するかどうかは、キュー共用グループを使用しているかどうかによります。 キュー共用環境では、ログ・ポイントを固有に識別するために相対バイト・アドレスを使用することはできません。 これは、複数のキュー・マネージャーが同じキューを同時に更新することができ、 それぞれに独自のログがあるためです。 これを解決するために、ログ・レコード・シーケンス番号はタイム・スタンプ値から取られ、 必ずしもログ内のログ・レコードの物理的な変位を示していません。

各ログ・レコードにはヘッダーがあり、ヘッダーには、ログ・レコードのタイプ、レコードを作成した IBM MQ サブコンポーネント、 および、回復単位レコードの場合は、その回復単位 ID が入っています。

ログ・レコードには 4 つの種類があり、それぞれ以下の見出しで説明されています。

- リカバリー単位ログ・レコード
- チェックポイント・レコード
- ページ・セット制御レコード
- CF 構造体のバックアップ・レコード

リカバリー単位ログ・レコード

ログ・レコードの多くは、IBM MQ キューの変更を記述しています。 このような変更はすべて、回復単位内で行われます。

IBM MQ は、再始動の時間短縮とシステム可用性の改善のために、 *undo/redo* ログ・レコード および補償ログ・レコード という特殊なロギング技法を利用しています。

その効果の 1 つとして、再始動時間が抑えられます。 再始動中に障害が発生してキュー・マネージャーをもう一度再始動する必要が生じた場合、 最初の再始動の障害発生時に完了していたすべての回復活動は、2 度目の再始動中に再適用する必要はありません。 つまり、再始動が続けて発生する場合、 再始動のたびに完了時間が徐々に長くなることはありません。

チェックポイント・レコード

再始動時間を短くするために、IBM MQ は通常の操作中に 定期的なチェックポイントを取ります。 これは以下の場合に行われます。

- 事前定義された数のログ・レコードがすでに書き込まれた場合。 この数は、システム・パラメーター・マクロ CSQ6SYSP の LOGLOAD というチェックポイント頻度オペランドで定義します。『CSQ6SYSP の使用』を参照してください。
- 再始動が正常に終わったとき。
- 通常の終了時。
- IBM MQ がサイクル内の次の活動ログ・データ・セットに移ったとき。

チェックポイントが取られた時点で、IBM MQ は、内部で DISPLAY CONN コマンドを実行し (『DISPLAY CONN』を参照)、現時点で未確定になっている接続のリストを z/OS のコンソール・ログに書き込みます。

ページ・セット制御レコード

これらのレコードは、各チェックポイントで IBM MQ キュー・マネージャーが認識しているページ・セットとバッファー・プールを登録し、チェックポイントの時点のページ・セットのメディア回復の実行に必要なログ範囲についての情報を記録します。

キュー・マネージャーの次回の再始動時に変更内容を回復して、自動的に復元できるように、 ページ・セットおよびバッファー・プールに対する特定の動的変更は、 ページ・セット制御レコードとしても書き込まれます。

CF 構造体のバックアップ・レコード

これらのレコードには、BACKUP CFSTRUCT コマンドでカップリング・ファシリティー・リスト構造体から読み取ったデータを入れます。 まれにしかないことですが、カップリング・ファシリティー構造体に障害が発生した場合に、RECOVER CFSTRUCT コマンドによってこれらのレコードが、回復単位レコードとともに使用されて、 障害発生時の状態まで、カップリング・ファシリティー構造体のメディア回復が実行されます。

関連タスク
より大きなログ相対バイト・アドレスの実装

z/OS ログの構造体

このトピックでは、ログ・レコードを記述するために使用する用語について理解することができます。

各活動ログ・データ・セットは VSAM 線形データ・セット (LDS) でなければなりません。 活動ログ・データ・セットに書き込まれる物理的な出力単位は、4 KB 制御インターバル (CI) です。 各 CI には 1 つの VSAM レコードが含まれます。

物理ログ・レコードと論理ログ・レコード

各 VSAM CI は 1 つの物理 レコードです。 ある特定の時点でログに記録された情報が論理 レコードで、その長さは CI 内の使用可能なスペースとは無関係です。 したがって、1 つの物理レコードには、以下のものが含まれる可能性があります。

- 複数の論理レコード
- 1 つまたは複数の論理レコードと、もう 1 つの論理レコードの一部
- 1 つの論理レコードの一部のみ

「ログ・レコード」は論理 レコードを指し、 それを保管するために必要な物理 レコードの数はさまざまに異なります。

z/OS IBM MQ for z/OS ログが書き込まれる方法

このトピックでは、IBM MQ がログ・ファイル・レコードを処理する方法について理解することができます。

IBM MQ では、それぞれのログ・レコードは活動ログ という DASD データ・セットに書き込まれます。 活動ログがいっぱいになると、IBM MQ はその内容を保存ログ という DASD またはテープ・データ・セットにコピーします。 このプロセスは、オフロード と呼ばれます。

245 ページの図 73 はロギング・プロセスを示しています。 通常、ログ・レコードは以下のサイクルで処理されます。

1. IBM MQ は、データに対する変更と重要なイベントをリカバリー・ログ・レコードに記録します。
2. IBM MQ はリカバリー・ログ・レコードを処理して、必要に応じてそれを複数のセグメントに分けます。
3. ログ・レコードは、VSAM 制御インターバル (CI) としてフォーマット済みの出力ログ・バッファー の中に順番に入ります。 それぞれのログ・レコードは、ゼロから 2^{64}-1 までの範囲の相対バイト・アドレスによって識別されます。
4. いくつかの CI が 1 つの事前定義 DASD 活動ログ・データ・セットに書き込まれます。 活動ログ・データ・セットは順番に使われ、再使用されます。
5. 保存が活動状態の場合、それぞれの活動ログ・データ・セットがいっぱいになるたびに、 その内容は新しい保存ログ・データ・セットに自動的にオフロードされます。

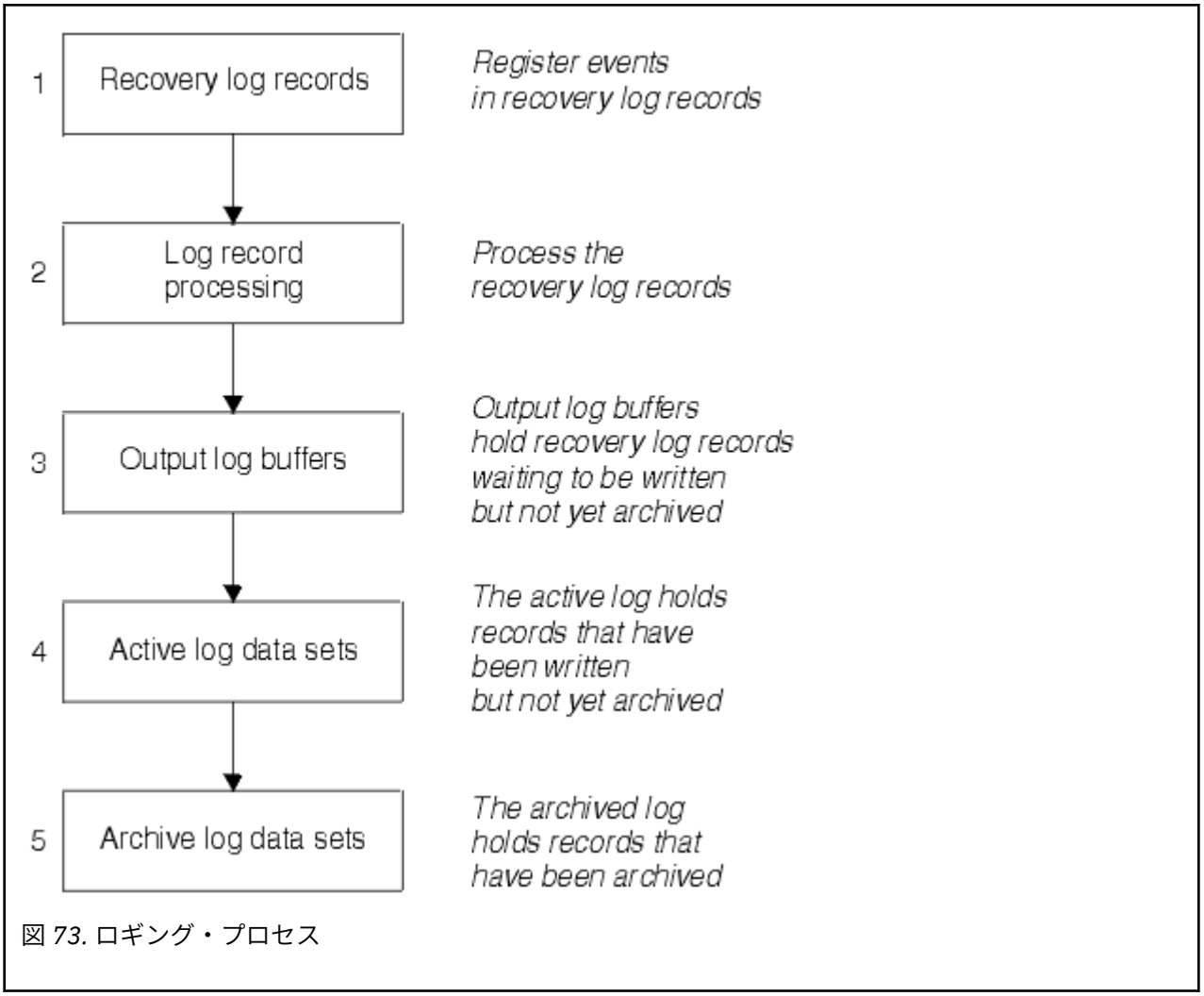


図 73. ロギング・プロセス

活動ログが書き込まれる時点

ストレージ内のログ・バッファーは、以下のいずれかが生じると、 活動ログ・データ・セットに書き込まれます。

- ログ・バッファーがいっぱいになったとき。
- 書き込みの限界値 (CSQ6LOGP マクロで指定) に達したとき。
- コミット点など、特定の重要なイベントが発生したとき、または IBM MQ BACKUP CFSTRUCT コマンドが発行されたとき。

キュー・マネージャーが初期設定されるとき、BSDS で指定されたキュー・マネージャー専用の活動ログ・データ・セットが動的に割り振られて、キュー・マネージャーが終了するまで IBM MQ 専用として継続的に保持されます。

ログ・データ・セットの動的な追加

キュー・マネージャーの実行中に、新しい活動ログ・データ・セットを動的に 定義することができます。この機能は、一時的な問題により、保存で活動ログをオフロードできないときに、 キュー・マネージャーが停止する問題を軽減します。 詳しくは、DEFINE LOG コマンドを参照してください。

注 : 活動ログを再定義または除去するには、 キュー・マネージャーを終了して再始動しなければなりません。

IBM MQ およびストレージ管理サブシステム

IBM MQ パラメーターを使用すると、 IBM MQ アーカイブ・ログ・データ・セットを動的に割り振るときに、ストレージ管理サブシステム (MVS™/DFP SMS) ストレージ・クラスを指定することができます。 ログ・データ・セットの保存を開始するのは IBM MQ ですが、保存データ・セットの割り振りには SMS を使用することができます。

関連資料

246 ページの『IBM MQ for z/OS 保存ログが書き込まれる時点』
このトピックでは、活動ログをコピーしてログを保存するプロセス、およびそのプロセスが発生するタイミングについて理解することができます。

z/OS IBM MQ for z/OS 保存ログが書き込まれる時点

このトピックでは、活動ログをコピーしてログを保存するプロセス、およびそのプロセスが発生するタイミングについて理解することができます。

活動ログを保存ログにコピーするプロセスは、オフロード と呼ばれます。 246 ページの図 74 は、 オフロードとその他のロギング・イベントとの関係を体系的に示しています。

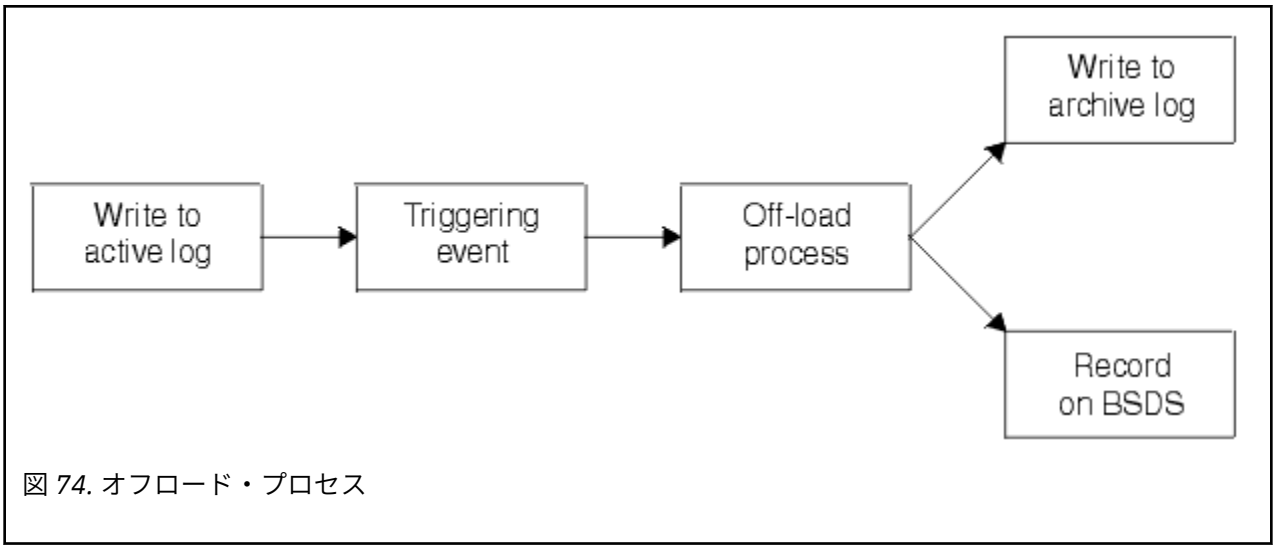


図 74. オフロード・プロセス

オフロード・プロセスのトリガー

活動ログから保存ログへのオフロードは、 いくつかのイベントによって起動されます。 以下に例を示します。

- 活動ログ・データ・セットがいっぱいになったとき。
- MQSC ARCHIVE LOG コマンドを使用したとき。
- 活動ログ・データ・セットへの書き込み中にエラーが発生した場合。

データ・セットは障害発生地点の前で切り捨てられ、 書き込まれなかったレコードが新しいデータ・セットの最初のレコードになります。 ここで、ログ・データ・セットがいっぱいになった場合と同様に、 切り捨てられたデータ・セットのためのオフロードが起動されます。 重複活動ログが存在する場合は、2 つのログを同期化するために、 両方とも切り捨てられます。

使用できる最後の活動ログが 5% までいっぱいになった時点、 およびその後 5% ずつ増加した時点で、 ログの使用率を示すメッセージ CSQJ110E が発行されます。 すべての活動ログが完全にいっぱいになった場合、IBM MQ はオフロードが終わるまで停止し、以下のメッセージが発行されます。

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

オフロード・プロセス

すべての活動ログが完全にいっぱいになった場合、IBM MQ はオフロードを実行し、オフロード完了まで処理を停止します。 活動ログがいっぱいになったあとでオフロードに失敗した場合は、IBM MQ が異常終了します。

活動ログのオフロードの準備ができたら、z/OS コンソール・オペレーターに対して、テープのマウントまたは DASD 装置の準備を要求するメッセージが出されます。 この要求を受け取るかどうかは、ロギング・オプション ARCWTOR の値 (詳しくは、CSQ6ARVP の使用を参照) によって決まります。 オフロードにテープを使用する場合は、ARCWTOR=YES と指定してください。 値が YES であれば、 この要求が WTOR (メッセージ番号 CSQJ008E) の前に発行されて、 保存ログ・データ・セットを取り付けるようオペレーターに指示します。

オペレーターは、このメッセージに必ずしもすぐに応答する必要はありません。 ただし、応答が遅れるとオフロード・プロセスも遅れます。 IBM MQ の活動ログが足りなくなるまでオペレーターが応答を遅らせない限り、IBM MQ のパフォーマンスには影響は出ません。

オペレーターは、オフロードを取り消すよう応答することもできます。 その場合、重複保存データ・セットの最初のコピーを割り振る途中であれば、 次の活動ログ・データ・セットがいっぱいになるまで、オフロードは延期されます。 2 番目のコピーを割り振る途中であれば、 このデータ・セットのみに関して、 単一コピー・モードの保存に切り替わります。

オフロード中の中断とエラー

キュー・マネージャーの停止要求が出されても、 オフロード完了までは要求が実行されません。 オフロードの途中で IBM MQ に障害が起きた場合、キュー・マネージャーが再始動したときにオフロードが再び始まります。

オフロード中のメッセージ

オフロードされたメッセージは、IBM MQ およびオフロード・プロセスによって z/OS コンソールに送信されます。 これらのメッセージを使用して、さまざまなログ・データ・セット内の RBA 範囲を見つけることができます。

z/OS より大きなログ相対バイト・アドレス

この機能は、ログのリセットが必要になるまでの期間を伸ばすことによって、キュー・マネージャーの可用性を向上させます。

キュー・マネージャーが再始動したときに持続メッセージを使用できるように、リカバリー・データがログに書き込まれます。 ログ相対バイト・アドレス (ログ RBA) という用語は、ログの先頭からのオフセットとしてのデータの位置を指して用いられます。

IBM MQ 8.0 より前では、6 バイトのログ RBA で、最大 256 テラバイトのデータまでアドレス指定できました。 書き込まれたログ・レコードがこの数量に達する前に、キュー・マネージャーのログのリセットで説明されている手順に従って、キュー・マネージャーのログをリセットする必要があります。

キュー・マネージャーのログのリセットは短時間でできる処理ではなく、処理の一部としてページ・セットをリセットする必要が生じるために、長時間にわたって停止しなければならなくなる可能性もあります。キュー・マネージャーの使用頻度が高い場合、年に 1 回程度この操作を実施する場合もあります。

IBM MQ 8.0 では、ログ RBA の長さは 8 バイトで、キュー・マネージャーは、ログ RBA のリセットが必要になるまでに、これまでの 64,000 倍以上のデータ (16 エクサバイト) をアドレス指定できるようになりました。 より大きいログ RBA を使用することによって、書き込まれるログ・データのサイズが数バイト大きくなります。

この機能を有効にするタイミング

V 9.3.0 IBM MQ 9.3.0 以降に作成されたキュー・マネージャーでは、この機能が有効になっています。

現在のログ RBA がログ RBA 範囲の終端に近づいている場合、キュー・マネージャーのログをリセットする代わりに、キュー・マネージャーを変換して 8 バイトのログ RBA を使用することを検討してください。 8 バイトのログ RBA を使用するようキュー・マネージャーを変換するときには停止が必要ですが、その所要時間はログをリセットするより短く、ログのリセットが必要になるまでの期間が大幅に長くなります。

キュー・マネージャーの初期設定時に発行されるメッセージ CSQJ034I は、構成されたキュー・マネージャーのログ RBA 範囲の末尾を示し、使用されるログ RBA が 6 バイトか 8 バイトかを判別するために使用できます。

この機能を有効にする方法

8 バイトのログ RBA は、キュー・マネージャーをバージョン 2 形式の BSDS で始動することによって有効にします。 要約すると、これは以下のように実行します。

1. キュー共用グループのすべてのキュー・マネージャーが 8 バイトのログ RBA の有効化の要件を満たすようにする。
2. キュー・マネージャーをクリーンな方式でシャットダウンする。
3. BSDS 変換ユーティリティーを実行してバージョン 2 形式で BSDS のコピーを作成する。
4. 変換後の BSDS を使用してキュー・マネージャーを再始動する。

一度 8 バイトのログ RBA を使用するようにキュー・マネージャーを変換すると、6 バイトのログ RBA を使用するように戻すことはできなくなります。

8 バイトのログ RBA を有効にする方法の詳細な手順については、より大きなログ相対バイト・アドレスの実装を参照してください。

関連タスク

アドレス指定可能な最大ログ範囲を広げる計画

関連資料

BSDS 変換ユーティリティー (CSQJUCNV)

z/OS ブートストラップ・データ・セット

ブートストラップ・データ・セットは、IBM MQ がログ・データ・セットとログ・レコードを参照するためのメカニズムとして必要です。 この情報は、通常の処理時と再始動リカバリー時に必要になります。

ブートストラップ・データ・セットの目的

ブートストラップ・データ・セット (BSDS) とは、IBM MQ の必要とする情報を保持している VSAM キー順データ・セット (KSDS) です。 これには、以下のものが含まれます。

- IBM MQ に認識されているすべての活動および保存ログ・データ・セットのインベントリー。 IBM MQ はこのインベントリーを使って次のことを行います。
  - 活動ログ・データ・セットおよび保存ログ・データ・セットの追跡
  - ログ・レコードを突きとめて、通常処理中にログ読み取り要求を満たすことができるようにする
  - ログ・レコードを突きとめて、再始動処理を取り扱うことができるようにする

  IBM MQ は、保存ログ・データ・セットが定義されるか、または活動ログ・データ・セットが再使用されるたびに、インベントリーに情報を格納します。 インベントリーは活動ログについて、どれがいっぱいであるか、 またどれが再使用できるかを示します。 またインベントリーには、そのデータ・セットにあるログの各部分の相対バイト・アドレス (RBA) が保持されています。
- 最近のすべての IBM MQ アクティビティーの 循環 インベントリー。 これは、キュー・マネージャーを再始動しなければならない場合に必要です。

キュー・マネージャーにエラーが発生して 再始動する必要がある場合には、BSDS が必要とされます。 IBM MQ **には** BSDS が必要です。 再始動時に問題が発生する可能性を最小限に抑えるために、IBM MQ を重複 BSDS で設定し、各 BSDS に同じ情報を記録することができます。 二重の BSDS を使用した実行モードのことを二重モード といいます。 可能であれば、それぞれのコピーを別々のボリュームに配置してください。

これにより、ボリュームが破損または破壊された場合に、 両方とも失われる危険性が小さくなります。 DASD に対する二重の書き込みの代わりに、二重の BSDS を使用してください。

BSDS は IBM MQ のカスタマイズ時にセットアップされ、ログ・インベントリー変更ユーティリティー (CSQJU003) を使用してインベントリーを管理することができます。 このユーティリティーの詳細については、IBM MQ for z/OS の管理を参照してください。 このユーティリティーは、キュー・マネージャー始動プロシージャー の DD ステートメントによって参照されます。

通常、IBM MQ は BSDS の複製コピーを保持しています。 I/O エラーが発生した場合、障害の起きたコピーを割り振り解除して、 単一の BSDS で続行します。 管理者は、二重モードの操作を復元できます。その方法については、IBM MQ for z/OS の管理を参照してください。

IBM MQ がインストールされると、活動ログは最初に BSDS に登録されます。 キュー・マネージャーをいったん終了してから再始動しないと、活動ログを置換できません。

保存ログ・データ・セットは動的に割り振られます。 割り振られた保存ログ・データ・セットは BSDS に登録されます。 保存ログが追加されるにつれて保存ログ・データ・セットのリストは拡張し、 ユーザー定義の項目数に達すると折り返されます。 最大項目数は単一保存ログの場合 1000、重複ロギングの場合は 2000 です。

テープ管理システムを使って、保存ログ・データ・セットを削除することができます (IBM MQ には自動的に削除するメソッドがありません)。 このため、システム管理者が保存ログ・データ・セットを削除した後に、 保存ログ・データ・セット情報が BSDS に長期間残る場合があります。

反対に、保存ログ・データ・セットの最大数を超えたため、 データ・セットが期限切れになるかなり前に BSDS データが 除去される可能性もあります。

次の MQSC コマンドを使用して、ログの範囲、および最も古いログ RBA を保持する 活動ログ・データ・セット名または保存ログ・データ・セット名を判別することができます。 これは、各種タイプのメディア回復またキュー・マネージャー回復に必要です。

```
DISPLAY USAGE TYPE(DATASET)
```

保存ログ・データ・セットの割り振り時にカタログするようにシステム・パラメーター・モジュールが指定している場合、BSDS は、それ以降の割り振りに必要な情報として統合カタログ機能 (ICF) のカタログを示します。 それ以外の場合、各ボリュームの BSDS 項目には、 それ以降の割り振りに必要なボリューム通し番号と装置情報が登録されます。

BSDS のバージョン

BSDS のフォーマットは、バージョンにより異なります。 BSDS のバージョンが進むにつれて、新機能を使用できるようになります。 次の BSDS バージョンが IBM MQ でサポートされています。

バージョン 1
IBM MQ のすべてのリリースでサポートされています。 バージョン 1 BSDS は 6 バイトのログ RBA 値をサポートします。

バージョン 2
IBM MQ 8.0 以降でサポートされます。 バージョン 2 BSDS では 8 バイトのログ RBA 値、およびアクティブ・ログのコピーごとに 310 個までのデータ・セットが使用可能です。

V 9.3.0 IBM MQ 9.3.0 以降で作成されたキュー・マネージャーの場合、デフォルトで使用可能になります。

バージョン 3
IBM MQ 8.0 以降でサポートされます。 BSDS は、アクティブ・ログ・コピーに 31 より大きいデータ・セットが追加されると、バージョン 2 から自動的にバージョン 3 に変換されます。

ログ・マップ印刷ユーティリティー (CSQJU004) を実行すれば、BSDS のバージョンを判別できます。 BSDS をバージョン 1 からバージョン 2 に変換するには、BSDS 変換ユーティリティー (CSQJUCNV) を実行します。

6 バイトのログ RBA と 8 バイトのログ RBA の詳細については、247 ページの『より大きなログ相対バイト・アドレス』を参照してください。

保存ログ・データ・セットと BSDS のコピー

新しい保存ログ・データ・セットが作成されるたびに、BSDS のコピーも作成されます。 保存ログがテープに保管される場合、BSDS は、最初の出力ボリューム上の最初のデータ・セットです。 保存ログが DASD に保管される場合、BSDS は別個のデータ・セットです。

保存ログのデータ・セット名と BSDS コピーの名前はほぼ同じです。 唯一の違いは、保存ログ名の最下位レベルの修飾子は A で始まりますが、BSDS コピーは B で始まることです。例えば、

保存ログ名
CSQ.ARCHLOG1.E00186.T2336229. *A* 0000001

BSDS コピー名
CSQ.ARCHLOG1.E00186.T2336229. *B* 0000001 (B)

BSDS のコピー中に読み取りエラーが発生した場合、 コピーは作成されず、メッセージ CSQJ125E が出されて、 新しい保存ログ・データ・セットのオフロードは BSDS コピーなしで続行されます。

z/OS でのシステム定義

IBM MQ for z/OS は、多くのデフォルトのオブジェクト定義を使用しており、それらのデフォルトのオブジェクトを作成するためのサンプル JCL を提供しています。 このトピックでは、これらのデフォルトのオブジェクトおよびサンプル JCL について理解することができます。

システム・パラメーターの設定

IBM MQ for z/OS では、IBM MQ が実行時に使用するロギング、保存、トレース、および接続環境をシステム・パラメーター・モジュールが制御します。 システム・パラメーターは、 以下の 3 つのアセンブラー・マクロによって指定されます。

CSQ6SYSP
接続環境やトレース環境などの設定のためのシステム・パラメーター。

CSQ6LOGP
ロギング・パラメーター。

CSQ6ARVP
ログ保存パラメーター。

デフォルト・パラメーター・モジュールが IBM MQ for z/OS に付属して提供されています。 使用したくない値がデフォルト値の中に含まれる場合には、IBM MQ 付属のサンプルを利用して独自のパラメーター・モジュールを作成することができます。 サンプルは `thlqual.SCSQPROC(CSQ4ZPRM)` です。

キュー・マネージャーの実行中に、システム・パラメーターの一部を変更できます。 MQSC コマンドの SET SYSTEM、SET LOG、および SET ARCHIVE コマンドを参照してください。

定義の詳細については、以下のトピックを参照してください。

- 251 ページの『IBM MQ for z/OS のシステム・オブジェクトの定義』
- 255 ページの『IBM MQ for z/OS でのキュー・マネージャーの調整』
- 256 ページの『IBM MQ for z/OS に用意されているサンプル定義』

関連概念
サンプル初期設定入力データ・セットのカスタマイズ
IBM MQ for z/OS で MQSC コマンドおよび PCF コマンドを発行できるソース
関連タスク
z/OS の管理
クラスターの構成
IBM MQ のモニター

z/OS IBM MQ for z/OS のシステム・オブジェクトの定義

IBM MQ for z/OS は、パブリッシュ/サブスクライブ・アプリケーション、クラスター、およびチャネル制御とその他のシステム管理機能に対して事前定義されている追加のオブジェクトを必要とします。

IBM MQ for z/OS が必要とするシステム・オブジェクトは、以下のカテゴリーに分類できます。

- パブリッシュ/サブスクライブ・オブジェクト
- システム・デフォルト・オブジェクト
- システム・コマンド・オブジェクト
- システム管理オブジェクト
- チャネル・キュー
- クラスター・キュー
- キュー共用グループ・キュー
- ストレージ・クラス
- システム・オブジェクト送達不能キューの定義
- デフォルト伝送キュー
- 内部キュー
- 255 ページの『チャネル認証キュー』

パブリッシュ/サブスクライブ・オブジェクト

IBM MQ for z/OS でパブリッシュ/サブスクライブ・アプリケーションを使用するためには、いくつかのシステム・オブジェクトをまず定義する必要があります。 これらのオブジェクトの定義に役立つサンプル定義が IBM MQ に付属しています。 サンプルについては、CSQ4INSG を参照してください。

パブリッシュ/サブスクライブ・クラスターを使用するときには、以下のオブジェクトを定義する必要があります。

- SYSTEM.RETAINED.PUB.QUEUE というローカル・キュー。これは、キュー・マネージャーの各保存パブリケーションのコピーを保持するために使用されます。 それぞれの完全トピック名は、このキューに保管されている 1 つの保存パブリケーションと一致します。 アプリケーションが保存パブリケーションを多種多様なトピックで使用する場合、または保存パブリケーション・メッセージが大きなメッセージである場合、このキューに必要なストレージを注意深く計画する必要があります。例えば、このキューに大容量のストレージが必要である場合、このキューを独自のページ・セットに割り当てる必要があります。パフォーマンスを改善するためには、このキューを (付属のサンプル・キュー定義のように) 索引タイプ MSGID で定義してください。
- SYSTEM.DURABLE.SUBSCRIBER.QUEUE というローカル・キュー。これは、永続サブスクリプションの持続コピーをキュー・マネージャーに保持するために使用されます。 パフォーマンスを改善するためには、このキューを (付属のサンプル・キュー定義のように) 索引タイプ CORRELID で定義してください。
- SYSTEM.DURABLE.MODEL.QUEUE というローカル・キュー。これは、管理される永続サブスクリプションのモデルとして使用されます。
- SYSTEM.NDURABLE.MODEL.QUEUE というローカル・キュー。これは、管理される非永続サブスクリプションのモデルとして使用されます。
- SYSTEM.QPUBSUB.QUEUE.NAMELIST という名前リスト。これには、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによってモニターされるキューのリストが含められます。
- SYSTEM.QPUBSUB.SUBPOINT.NAMELIST という名前リスト。これには、キューに入れられたパブリッシュ/サブスクライブ・インターフェースがトピック・オブジェクトをサブスクリプション・ポイントと突き合わせるために使用する、トピック・オブジェクトのリストが含められます。
- SYSTEM.BASE.TOPIC というトピック。これは、属性を解決する基本トピックとして使用されます。
- SYSTEM.BROKER.DEFAULT.STREAM というトピック。これは、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによって使用されるデフォルトのストリームです。

- SYSTEM.BROKER.DEFAULT.SUBPOINT というトピック。これは、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによって使用されるデフォルトの RFH2 サブスクリプション・ポイントです。
- SYSTEM.BROKER.ADMIN.STREAM というトピック。これは、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによって使用される管理ストリームです。
- SYSTEM.DEFAULT.SUB というサブスクリプション。これは、DEFINE SUB コマンドにデフォルト値を提供するために使用されるデフォルトのサブスクリプション・オブジェクトです。

システム・デフォルト・オブジェクト

オブジェクトが定義されるとき、その定義の基礎になる別のオブジェクト名が指定されない場合は、 デフォルト属性を提供するためにシステム・デフォルト・オブジェクトが使われます。

デフォルトのシステム・オブジェクト定義の名前は、「SYSTEM.DEFAULT」または「SYSTEM.DEF」という文字列で始まります。 例えば、システム・デフォルト・ローカル・キューの 名前は SYSTEM.DEFAULT.LOCAL.QUEUE になります。

これらのオブジェクトは、以下の IBM MQ オブジェクトの属性のシステム・デフォルトを定義します。

- ローカル・キュー
- モデル・キュー
- 別名キュー
- リモート・キュー
- Processes
- 名前リスト
- チャネル
- ストレージ・クラス
- 認証情報

共用キューは特別な種類のローカル・キューですから、 共用キューが定義されるときには、 その定義は SYSTEM.DEFAULT.LOCAL.QUEUE に基づいて行われます。 ただし、カップリング・ファシリティー構造体名は定義されていないため、 ユーザーがこの値を提供する必要があることに注意してください。 または、定義されるすべての共用キューが必要な属性を継承するように、 基礎となるデフォルト共用キューをユーザーが独自に定義することもできます。 共用キューは、キュー共用グループの中のただ 1 つのキュー・マネージャーでのみ定義することに注意してください。

システム・コマンド・オブジェクト

システム・コマンド・オブジェクトの名前は、 文字 SYSTEM.COMMAND で始まります。 IBM MQ 操作パネルおよび制御パネルを使って IBM MQ サブシステムへのコマンドを実行する前に、これらのオブジェクトを定義しておく必要があります。

システム・コマンド・オブジェクトには以下の 2 つがあります。

1. システム・コマンド入力キューは、IBM MQ コマンド・プロセッサーによる処理の前にコマンドを格納するローカル・キューです。 名前は SYSTEM.COMMAND.INPUT でなければなりません。 SYSTEM.ADMIN.COMMAND.QUEUE も、IBM MQ for Multiplatforms との互換性のため、および IBM MQ Console と administrative REST API で使用するために定義する必要があります。
2. SYSTEM.COMMAND.REPLY.MODEL は、 システム・コマンド応答キューを定義するモデル・キューです。

IBM MQ Explorer が使用する 2 つの追加オブジェクトがあります。

- SYSTEM.MQEXPLORER.REPLY.MODEL キュー
- SYSTEM.ADMIN.SVRCONN チャネル

SYSTEM.REST.REPLY.QUEUE は、 IBM MQ administrative REST API によって使用される応答キューです。

これら 2 つのシステム・コマンド・オブジェクトが DEFPSIST(NO) 属性を持つようにするために、 通常、コマンドは非持続メッセージを使って送られます。こうすれば、これらを使用するアプリケーション (付属のユーティリティー・プログラムや操作パネルと制御パネルを含む) は、デフォルトで非持続メッセージを受け取ります。 コマンド用に持続メッセージを使用するアプリケーションをお使いの場合は、そのようなコマンドへの応答メッセージを持続させるために、応答キューに DEFTYPE(PERMDYN) 属性を設定してください。

システム管理オブジェクト

システム管理オブジェクトの名前は、文字 SYSTEM.ADMIN で始まります。

以下の 7 つのシステム管理オブジェクトがあります。

- SYSTEM.ADMIN.CHANNEL.EVENT キュー
- SYSTEM.ADMIN.COMMAND.EVENT キュー
- SYSTEM.ADMIN.CONFIG.EVENT キュー
- SYSTEM.ADMIN.PERFM.EVENT キュー
- SYSTEM.ADMIN.QMGR.EVENT キュー
- SYSTEM.ADMIN.TRACE.ROUTE.QUEUE キュー
- SYSTEM.ADMIN.ACTIVITY.QUEUE キュー

チャネル・キュー

分散キューイングを使用するには、以下のオブジェクトを定義する必要があります。

- SYSTEM.CHANNEL.SYNCQ という名前のローカル・キュー。 これは、チャネルの順序番号と作業論理単位 ID (LUWID) の保管に使われます。 チャネルのパフォーマンスを改善するためには、このキューを (付属のサンプル・キュー定義のように) 索引タイプ MSGID で定義してください。
- SYSTEM.CHANNEL.INITQ という名前のローカル・キュー。 これは、チャネル・コマンドに使われます。

これらのキューを共用キューとして定義することはできません。

クラスター・キュー

IBM MQ クラスターを使用するときには、以下のオブジェクトを定義する必要があります。

- SYSTEM.CLUSTER.COMMAND.QUEUE というローカル・キュー。 これは、キュー・マネージャー間でリポジトリー変更を通知するために使われます。 このキューに書き込まれるメッセージには、 リポジトリーのローカル・コピーに適用されるリポジトリー・データ更新情報、 またはリポジトリー・データの要求情報が含まれます。
- SYSTEM.CLUSTER.REPOSITORY.QUEUE というローカル・キュー。 これは、リポジトリーの持続コピーを保持するために使われます。
- SYSTEM.CLUSTER.TRANSMIT.QUEUE というローカル・キュー。 これは、クラスター内のすべての宛先への伝送キューです。 パフォーマンスを改善するためには、このキューを (付属のサンプル・キュー定義のように) 索引タイプ CORRELID で定義してください。

通常、これらのキューには多数のメッセージが含まれます。

これらのキューを共用キューとして定義することはできません。

キュー共用グループ・キュー

共用チャネルおよびグループ内キューイングを使用するには、 以下のオブジェクトを定義する必要があります。

- SYSTEM.QSG.CHANNEL.SYNCQ という名前の共用キュー。 これは、共用チャネルの同期情報を保持するために使われます。
- SYSTEM.QSG.TRANSMIT.QUEUE という名前の共用キュー。 これは、グループ内キューイングの伝送キューとして使われます。 キュー共用グループを実行する場合には、 たとえグループ内キューイングを使用しなくても、 このキューを定義する必要があります。

ストレージ・クラス

以下にあげる 6 つのストレージ・クラスを定義することをお勧めします。 このうち 4 つは IBM MQ で必須であるため、定義しなければなりません。 残りの 2 つのストレージ・クラスはサンプル・キュー定義で使われるため、 これらを定義することをお勧めします。

DEFAULT (必須)
このストレージ・クラスは、パフォーマンス上重要ではなく、 他のいずれのストレージ・クラスにも当てはまらないようなすべてのメッセージ・キュー用に使われます。 また、これは、キュー定義時にストレージ・クラスが指定されない場合に提供されるデフォルト・ストレージ・クラスでもあります。

NODEFINE (必須)
このストレージ・クラスは、キュー定義時に指定されたストレージ・クラスが未定義である場合に使われます。

REMOTE (必須)
このストレージ・クラスは主に、伝送キュー、 つまり短時間しかとどまらないパフォーマンス上重要なメッセージを格納するシステム関連キュー用に使われます。

SYSLNGLV
このストレージ・クラスは、 長期間存続するパフォーマンス上重要なメッセージのために使われます。

SYSTEM (必須)
このストレージ・クラスは、パフォーマンスが重要なシステム関連メッセージ・キュー (SYSTEM.CHANNEL.SYNQ および SYSTEM.CLUSTER.* キュー。

SYSVOLAT
このストレージ・クラスは、短期間しか存続しないパフォーマンス上重要なメッセージのために使われます。

必要に応じて、上記のストレージ・クラスの属性を変更したり、 他のストレージ・クラス定義を追加することができます。

システム・オブジェクト送達不能キューの定義

送達不能キューは、メッセージ宛先が無効である場合に使われます。 IBM MQ は、そのようなキューを送達不能キューというローカル・キューに格納します。 送達不能キューは必須ではありませんが、分散キューイングやいずれかの IBM MQ ブリッジを使用する場合は特に重視しなければなりません。

送達不能キューは、共用キューとして定義**しないでください**。 キュー・マネージャー上のローカル・キューへの put は、送達不能キューへの put となることがあります。 送達不能キューが共有キューである場合は、別のシステムの送達不能キュー・ハンドラーがメッセージを処理して同じ名前でキューに put できますが、それが異なるキュー・マネージャーにあるために、正しくないキューであったりセキュリティー・プロファイルが異なっていたりします。 キューが存在しない場合、再処理は失敗します。

送達不能キューを定義する場合には、 キュー・マネージャーにその名前を伝える必要があります。 そのためには、ALTER QMGR DEADQ(*queue-name*) コマンドを使用します。 詳しくは、キュー・マネージャーの属性の表示および変更を参照してください。

デフォルト伝送キュー

デフォルト伝送キューは、別のキュー・マネージャーへのメッセージ送信に使用できるような適切な伝送キューが他に存在しない場合に使われます。 デフォルト伝送キューを定義する場合は、 そのキューの経路

となるチャネルも定義する必要があります。 チャネルを定義しないと、デフォルト伝送キューに入ったメッセージはリモート・キュー・マネージャーに送信されず、 キューの中に残ったままの状態になります。

デフォルト伝送キューを定義する場合には、 キュー・マネージャーにその名前を伝える必要があります。 これを行うには、ALTER QMGR コマンドを使用します。

内部キュー

- **データ・キューの保留**
  - 内部使用のために定義されたキュー、SYSTEM.PENDING.DATA.QUEUE は、 JMS パブリッシュ/サブスクライブ環境での永続サブスクリプションの使用をサポートします。
- **JMS 2.0 送達遅延のステージング・キュー**
  - JMS 2.0 によって提供される送達遅延機能が使用される場合は、内部ステージング・キュー SYSTEM.DDELAY.LOCAL.QUEUE を定義する必要があります。 このキューは、ゼロ以外の送達遅延を使用して送信されたメッセージを一時的に保管するためにキュー・マネージャーが使用します。 送達遅延が完了すると、メッセージはターゲット宛先に書き込まれます。 このキューの定義のサンプルを、CSQ4INSG にコメント化して示しています。
  - SYSTEM.DDELAY.LOCAL.QUEUE キューを定義する場合は、送達遅延を伴って送信されるメッセージの予測数に関する STGCLASS、MAXMSGL、および MAXDEPTH 属性を設定する必要があります。 また、SYSTEM.DDELAY.LOCAL.QUEUE キューを定義する際には、キュー・マネージャーのみがこのキューにメッセージを書き込めるようにしてください。 このキューへのメッセージ書き込み権限を持つユーザー ID が存在しないように注意する必要があります。

チャネル認証キュー

チャネル認証を内部で使用するためには、SYSTEM.CHLAUTH.DATA.QUEUE キューが必要です。 これらのオブジェクトの定義に役立つサンプル定義が IBM MQ に付属しています。 このサンプルについては CSQ4INSA で説明しています。 これには、いくつかのデフォルト・ルールも定義されています。

z/OS IBM MQ for z/OS でのキュー・マネージャーの調整

基本的なパフォーマンス上の問題を回避するようにキュー・マネージャーが調整されていることを確認するための、いくつかの簡単な手順があります。

キュー・マネージャーのパフォーマンスを改善することのできる、いくつかの方法があります。 それらは ALTER QMGR コマンドによって設定されるキュー・マネージャー属性によって制御されます。 このセクションでは、キュー・マネージャーで許可されるメッセージの最大数を設定することによって、またキュー・マネージャーで「ハウスキーピング」を実行することによって、このことを行う方法についての情報が含まれます。 IBM MQ SupportPac MP16 - IBM MQ for z/OS キャパシティー・プランニング & のチューニング は、パフォーマンスおよび調整に関する詳細を提供しています。

同期点

キュー・マネージャーの役割の 1 つは、アプリケーション内の同期点制御です。 アプリケーションは、任意の数の MQPUT または MQGET 呼び出しがあり、 MQCMIT 呼び出しで終了する作業単位から構成されます。

1 つの MQCMIT の有効範囲内にある MQPUT 呼び出しまたは MQGET 呼び出しの数が増加すると、コミットのパフォーマンス・コストが大幅に増加します。 一般に、アプリケーションは、単一の同期点で多数のメッセージを MQPUT/MQGET ではないように設計する必要があります。

MAXUMSGS Queue Manager の属性を使用することにより、単一の同期点内のメッセージ数を管理して制限することができます。 アプリケーションがこの限界を超えた場合は、MQPUT,MQPUT1 で MQRC_SYNCPOINT_LIMIT_REACHED を受信するか、限界を超える MQGET 呼び出しを受け取ります。 アプリケーションは、必要に応じて MQCMIT または MQBACK を発行する必要があります。

MAXUMSGS のデフォルト値は 10000 です。 下限を適用したい場合は、この値を小さくすることができます。これは、アプリケーションのループを保護するのに役立ちます。 MAXUMSGS を減らす前に、既存のアプリケーションが限界を超えないようにするか、または MQRC_SYNCPOINT_LIMIT_REACHED リターン・コードを許容できることを確認してください。

有効期限切れメッセージ

有効期限が切れたメッセージは、次回の適切な MQGET 呼び出しによって廃棄されます。 ただし、そのような呼び出しがなければ、有効期限切れメッセージが廃棄されず、 一部のキュー、特に MessageId、CorrelId、または GroupId によってメッセージが検索され、 パフォーマンスのために索引が付けられるキューでは、 多数の有効期限切れメッセージが累積されることがあります。 キュー・マネージャーは、期限切れメッセージがないかどうかキューを定期的に走査することができます。 期限切れメッセージがあれば、そのメッセージは削除されます。 この走査を実行する場合、頻度を選択することができます。 これには、次の 2 つの方法があります。

明示的な要求
走査するキューとタイミングを制御することができます。 走査したいキュー (ひとつまたは複数) を指定して、REFRESH QMGR TYPE(EXPIRY) コマンドを発行してください。

定期的なスキャン
EXPRYINT 属性を使用して、キュー・マネージャー・オブジェクトに 廃棄間隔を指定することができます。 キュー・マネージャーは、 各キューの有効期限切れメッセージに関する情報を維持管理し、 有効期限切れメッセージの走査が有効な時期を認識しています。 EXPRYINT 間隔ごとに、有効期限切れメッセージがないかどうかキュー・マネージャーが走査対象のキューを 探索し、対象と判断したキューだけを走査します。 すべてのキューが走査されるわけではありません。 これにより、不要な走査にプロセッサー時間が無駄に費やされることが回避されます。

共用キューは、キュー共用グループ内の 1 つのキュー・マネージャーのみによって走査されます。 通常、再始動する最初のキュー・マネージャーまたは EXPRYINT が設定される 最初のキュー・マネージャーが走査を実行します。

注 : キュー共用グループ内のすべてのキュー・マネージャーに、同じ EXPRYINT 値を 設定する必要があります。

z/OS IBM MQ for z/OS に用意されているサンプル定義

このトピックは、IBM MQ for z/OS に用意されているサンプル JCL とコードのリファレンスとしてご利用ください。

IBM MQ の thlqual.SCSQPROC ライブラリーには、以下のサンプル定義が用意されています。 サンプル定義を使用して、システム・オブジェクトを定義したり、独自のオブジェクトをカスタマイズしたりする操作を実行できます。 初期化入力データ・セットに一部のサンプル定義を組み込みことも可能です (初期化コマンドを参照してください)。

表 *22. IBM MQ* のシステム・オブジェクトのためのサンプル定義

初期化入力データ・セット	**サンプル名**
CSQINP1	CSQ4INP1 CSQ4INPR

表 22. IBM MQ のシステム・オブジェクトのためのサンプル定義 (続き)	
初期化入力データ・セット	**サンプル名**
CSQINP2	CSQ4INSA CSQ4INYS [1] CSQ4INSX CSQ4INSG CSQ4INSR CSQ4INSS CSQ4INSJ CSQ4INSM CSQ4INYG CSQ4INYR CSQ4INYC CSQ4INYD CSQ4INSC
CSQINPT	CSQ4INST CSQ4INYT
その他	CSQ4DISP CSQ4INPX CSQ4IVPQ CSQ4IVPG CSQ4MSTR CSQ4MSRR CSQ4QMIN

注:

1. これらのサンプル定義の順序は重要です。INYS、INSX、INSG の順序が間違っていると、エラーになります。

CSQINP1 サンプル

メッセージのクラスごとに 1 つのページ・セットを使用する場合は、サンプル CSQINP1 データ・セット thlqual.SCSQPROC(CSQ4INP1) を使用します。メッセージのメジャー・クラスで複数のページ・セットを使用する場合は、thlqual.SCSQPROC(CSQ4INPR) を使用します。 このサンプルには、バッファー・プールの定義、バッファー・プールとページ・セットの関連付け、ALTER SECURITY コマンドが含まれています。このサンプルをキュー・マネージャー開始タスク・プロシージャーの CSQINP1 連結に組み込みます。

CSQINP2 サンプル

CSQ4INSG システム・オブジェクト・サンプル

サンプル CSQINP2 データ・セット thlqual.SCSQPROC(CSQ4INSG) には、以下の汎用システム・オブジェクトの定義が含まれています。

- システム・デフォルト・オブジェクト
- システム・コマンド・オブジェクト
- システム管理オブジェクト
- システム用の他のオブジェクト

このサンプルに含まれているオブジェクトを定義する操作は必須ですが、その操作は、サブシステムの始動時に 1 回実行するだけで十分です。 そのための最適な方法は、CSQINP2 データ・セットにこれらの定義を組み込むという方法です。 これらの定義は、キュー・マネージャーのシャットダウンと再始動の境界を越えて保持されます。 オブジェクト名を変更することはできませんが、必要に応じて属性を変更することは可能です。

以下の条件を満たしている場合は、(パブリッシュ・サブスクライブがアクティブでなくても) 1 つのメッセージが SYSTEM.DURABLE.SUBSCRIBER.QUEUE キューに書き込まれます。

- QMGR 属性の PSMODE が DISABLED に設定されている
- サンプル・オブジェクト CSQ4INST のステートメント `DEFINE SUB('SYSTEM.DEFAULT.SUB')` が存在する。

これを回避するには、`DEFINE SUB('SYSTEM.DEFAULT.SUB')` ステートメントを削除またはコメント化します。

JMS 2.0 送達遅延ステージング・キュー、SYSTEM.DDELAY.LOCAL.QUEUE を定義する必要があるのは、JMS 2.0 送達遅延を使用する場合のみです。 デフォルトではこのキューの定義はコメント化されていますが、コメントは必要に応じて外すことができます。

CSQ4INSA システム・オブジェクトおよび認証のサンプル

サンプルの CSQINP2 データ・セット thlqual.SCSQPROC(CSQ4INSA) には、チャネル認証システム・キューの定義が含まれています。 このキューには、チャネル認証レコードが保持されています。 またこのキューには、デフォルトのチャネル認証ルールも含まれています。

キュー・マネージャーで CHLAUTH が有効になっていて、チャネルを実行する必要がある場合、あるいは CHLAUTH レコードを設定または表示する必要がある場合は、このサンプル内のそれらのオブジェクトを定義する必要があります。 それらを定義する必要があるのは、最初にサブシステムを始動したときだけです。 そのための最適な方法は、CSQINP2 データ・セットにこれらの定義を組み込むという方法です。 それらの定義は、キュー・マネージャーがシャットダウンして再始動しても保持されているので、キュー名を変更することはできません。

CSQ4INSS システム・オブジェクト・サンプル

キュー共用グループを使用する場合は、追加のシステム・オブジェクトを定義できます。

サンプル・データ・セット thlqual.SCSQPROC(CSQ4INSS) には、CF 構造で使用するサンプル・コマンドと、共用チャネルやグループ内キューイングで必要なシステム・オブジェクトの定義の集合が含まれています。

このサンプルをそのまま使用することはできません。カスタマイズしてから使用する必要があります。 その後、このメンバーをキュー・マネージャー始動プロシージャーの CSQINP2 DD 連結に含めることができます。 または、必要なコマンドを実行するために、 これを CSQUTIL ユーティリティーの COMMAND 関数の入力として使用することもできます。

グループ・オブジェクトまたは共用オブジェクトを定義する場合は、キュー共用グループに含まれている 1 つのキュー・マネージャーについてのみ、これらの定義を CSQINP2 DD 連結に組み込む必要があります。

CSQ4INSX システム・オブジェクト・サンプル

分散キューイングやクラスタリングを使用する場合は、追加のシステム・オブジェクトを定義する必要があります。

サンプル・データ・セット thlqual.SCSQPROC(CSQ4INSX) には、必要なキュー定義が含まれています。このメンバーをキュー・マネージャー始動プロシージャーの CSQINP2 DD 連結に組み込んだり、CSQUTIL ユーティリティーの COMMAND 機能の入力として使用して必要な DEFINE コマンドを実行したりすることできます。

以下の 2 つのタイプのオブジェクト定義があります。

- SYSTEM.CHANNEL.xx (分散キューイングで必要)

- SYSTEM.CLUSTER.xx (クラスタリングで必要)

CSQ4INSJ システム JMS オブジェクト・サンプル

JMS パブリッシュ/サブスクライブ・ドメインで使用されるキューを定義します。

CSQ4INSM システム・オブジェクト・サンプル

拡張メッセージ・セキュリティーを使用する場合は、追加のシステム・オブジェクトを定義する必要があります。 サンプル・データ・セット thlqual.SCSQPROC(CSQ4INSM) には、必要なキュー定義が含まれています。

CSQ4INSR オブジェクト・サンプル

WebSphere Application Server およびブローカーによって使用されるキューを 定義します。

CSQ4INYD オブジェクト・サンプル

分散キューイングを使用する場合は、独自のキュー、プロセス、チャネルをセットアップする必要があります。

サンプル・データ・セット thlqual.SCSQPROC(CSQ4INYD) には、分散キューイング・オブジェクトをカスタマイズするために使用できるサンプル定義が含まれています。 以下のような内容です。

- 送信側のための定義の集合
- 受信側のための定義の集合
- クライアントを使用するための定義の集合

このサンプルをそのまま使用することはできません。カスタマイズしてから使用する必要があります。 その後、このメンバーをキュー・マネージャー始動プロシージャーの CSQINP2 DD 連結に組み込んだり、CSQUTIL ユーティリティーの COMMAND 機能の入力として使用して必要な DEFINE コマンドを実行したりすることできます。 (キュー・マネージャーを再始動するたびにこれらのオブジェクトを再定義する必要がないため、この方法を推奨します。)

CSQ4INYC オブジェクト・サンプル

クラスタリングを使用する場合は、分散キューイングのチャネル定義とリモート・キュー定義に相当する定義が必要に応じて自動的に作成されます。 ただし、手動のチャネル定義がいくつか必要です。つまり、クラスターのクラスター受信側チャネルとクラスター送信側定義を少なくとも 1 つのクラスター・リポジトリー・キュー・マネージャーに対して定義する必要があります。

サンプル・データ・セット thlqual.SCSQPROC(CSQ4INYC) には、クラスタリング・オブジェクトをカスタマイズするために使用できる以下のサンプル定義が含まれています。

- キュー・マネージャーのための定義
- 受信側チャネルのための定義
- 送信側チャネルのための定義
- クラスター・キューのための定義
- クラスターのリストのための定義

このサンプルをそのまま使用することはできません。カスタマイズしてから使用する必要があります。 その後、このメンバーをキュー・マネージャー始動プロシージャーの CSQINP2 DD 連結に組み込んだり、CSQUTIL ユーティリティーの COMMAND 機能の入力として使用して必要な DEFINE コマンドを実行したりすることできます。 この方法は確かに望ましいといえます。IBM MQ の再始動のたびに、これらのオブジェクトを再定義する必要がなくなるからです。

CSQ4INYG オブジェクト・サンプル

サンプル・データ・セット thlqual.SCSQPROC(CSQ4INYG) には、独自の汎用オブジェクトをカスタマイズするために使用できる以下のサンプル定義が含まれています。

- 送達不能キュー
- デフォルト伝送キュー

- CICS アダプター・オブジェクト

このサンプルをそのまま使用することはできません。カスタマイズしてから使用する必要があります。 その後、このメンバーをキュー・マネージャー始動プロシージャーの CSQINP2 DD 連結に組み込んだり、CSQUTIL ユーティリティーの COMMAND 機能の入力として使用して必要な DEFINE コマンドを実行したりすることできます。 この方法は確かに望ましいといえます。IBM MQ の再始動のたびに、これらのオブジェクトを再定義する必要がなくなるからです。

ここで取り上げるサンプル定義のほかに、システム・オブジェクト定義を独自のリソース定義のベースとして使用することもできます。 例えば、SYSTEM.DEFAULT.LOCAL.QUEUE の作業用コピーを作成し、名前を MY.DEFAULT.LOCAL.QUEUE にします。 次に、このコピーのパラメーターを必要に応じて変更できます。 さらに、任意の方法で DEFINE コマンドを実行できます (ただし、そのタイプのリソースを作成する権限があることが前提になります)。

デフォルト伝送キュー
デフォルト伝送キューを定義するかどうかを決める前に、『デフォルト伝送キュー』の説明を参照してください。

- デフォルト伝送キューを定義することにした場合は、そのキューに対応するチャネルを定義することも必要になります。
- デフォルト伝送キューを定義しないことにした場合は、サンプルの ALTER QMGR コマンドから DEFXMITQ ステートメントを削除してください。

CICS アダプター・オブジェクト
このサンプルでは、CICS01.INITQ という名前の開始キューが定義されています。 このキューは、IBM MQ に用意されている CKTI トランザクションで使用するキューです。 このキューの名前を変更することもできますが、CICS システム初期設定テーブル (SIT) で指定されている名前、または INITPARM ステートメントの SYSIN オーバーライドで指定されている名前と一致する名前にしなければなりません。

CSQ4INYS/CSQ4INYR オブジェクト・サンプル

以下のような用途のストレージ・クラス定義です。

- メッセージのクラスごとに 1 つのページ・セットを使用するため
- メッセージのメジャー・クラスで複数のページ・セットを使用するため

例えば、 SYSTEM.COMMAND.INPUT は STGCLASS ('SYSVOLAT')、および SYSTEM.CLUSTER.TRANSMIT.QUEUE は STGCLASS ('REMOTE') を使用します。 CSQ4INYS では、これらのストレージ・クラスは両方とも同じページ・セットを使用します。 CSQ4INYR では、伝送キューの使用による影響を軽減するために、これらのストレージ・クラスは異なるページ・セットを使用します。

CSQINPT のサンプル

CSQ4INST
サンプル・データ・セット thlqual.SCSQPROC(CSQ4INST) には、システム・デフォルト・サブスクリプションの定義が含まれています。

このサンプル内のオブジェクトを定義する必要がありますが、定義が必要なのは、パブリッシュ/サブスクライブ・エンジンの初回開始時に一度のみです。 そのための最適な方法は、CSQINPT データ・セットにこれらの定義を組み込むという方法です。 この定義は、キュー・マネージャーのシャットダウンと再始動の境界を越えて保持されます。 オブジェクト名を変更することはできませんが、必要に応じて属性を変更することは可能です。

CSQ4INYT
サンプル・データ・セット thlqual.SCSQPROC(CSQ4INYT) には、パブリッシュ/サブスクライブ・エンジンの始動時に実行できるコマンドの集合が含まれています。 このサンプルを実行すると、トピックとサブスクリプションに関する情報が表示されます。

その他

CSQ4DISP 表示サンプル

サンプル・データ・セット thlqual.SCSQPROC(CSQ4DISP) には、キュー・マネージャーで定義されているすべてのリソースを表示するための汎用 DISPLAY コマンドの集合が含まれています。 ストレージ・クラスやトレースなど、すべての IBM MQ オブジェクトおよび定義もこれに含まれます。 これらのコマンドからは、大量の出力が生成される可能性があります。 このサンプルを CSQINP2 データ・セットで使用したり、CSQUTIL ユーティリティーの COMMAND 機能の入力として使用したりすることができます。

CSQ4INPX サンプル

サンプル・データ・セット thlqual.SCSQPROC(CSQ4INPX) には、チャネル・イニシエーターの始動のたびに実行できるコマンドの集合が含まれています。 このサンプルは使用前にカスタマイズする必要があり、カスタマイズしたあとでチャネル・イニシエーター用の CSQINPX データ・セットに組み込むことができます。

CSQ4IVPQ サンプルと CSQ4IVPG サンプル

サンプル・データ・セット thlqual.SCSQPROC(CSQ4IVPQ) と thlqual.SCSQPROC(CSQ4IVPG) には、インストール検査プログラム (IVP) を実行するために必要な DEFINE コマンドの集合が含まれています。

これらのサンプルを CSQINP2 データ・セットに含むことができます。 IVP を正常に実行できたら、キュー・マネージャーの再始動のたびに IVP を繰り返し実行する必要はありません。 したがって、これらのサンプルを CSQINP2 連結の中に永久に入れておく必要はありません。

CSQ4MSTR と CSQ4MSRR のサンプル

これらは、キュー・マネージャーのための開始済みタスク・プロシージャーのサンプル thlqual.SCSQPROC(CSQ4MSTR) および thlqual.SCSQPROC(CSQ4MSRR) です。

CSQ4MSRR は、 CSQINP2 連結で CSQ4INYR を使用して、重要なキューが異なるページ・セットに分散されるようにします。

コメントを解除して、新しく作成したキュー・マネージャーで CSQMINI カードを必要に応じて使用できます。

CSQ4QMIN サンプル

サンプルの QMINI データ・セット、thlqual.SCSQPROC(CSQ4QMIN)。

QMINI データ・セットおよび **TransportSecurity** スタンザの詳細については、 QMINI データ・セット を参照してください。

z/OS での回復と再始動

このトピックのリンクを使用して、IBM MQ for z/OS の再始動および回復の機能について参照してください。

IBM MQ for z/OS には、再始動および回復のための堅固な機能があります。 停止後にキュー・マネージャーが回復する方法、および再始動時に発生する処理については、以下のサブトピックを参照してください。

- 262 ページの『IBM MQ for z/OS でのデータ変更の方法』
- 263 ページの『IBM MQ for z/OS での整合性維持の方法』
- 265 ページの『IBM MQ for z/OS で終了時に行われる処理』
- 267 ページの『IBM MQ for z/OS で再始動および回復時に行われる処理』
- 269 ページの『未確定の回復単位の解決方法』
- 271 ページの『共用キューの回復』

関連概念

z/OS IBM MQ for z/OS のリカバリー処置

関連タスク

バックアップおよび回復の計画

z/OS z/OS の管理

関連資料

z/OS IBM MQ for z/OS のメッセージ

z/OS IBM MQ for z/OS でのデータ変更の方法

IBM MQ for z/OS は他のサブシステムと対話して、すべてのデータの整合性を保つ必要があります。 このトピックには、回復単位 とは何か、またバックアウト において回復単位がどのように使用されるかに関する情報が含まれます。

回復単位

回復単位 とは、単一のキュー・マネージャーが 1 つのアプリケーション・プログラムに関して行う、ある整合点から別の整合点に向けて IBM MQ データを変更する処理です。 整合点 (同期点 またはコミット点 ともいう) とは、アプリケーション・プログラムが利用するすべての回復可能データが整合する時点です。

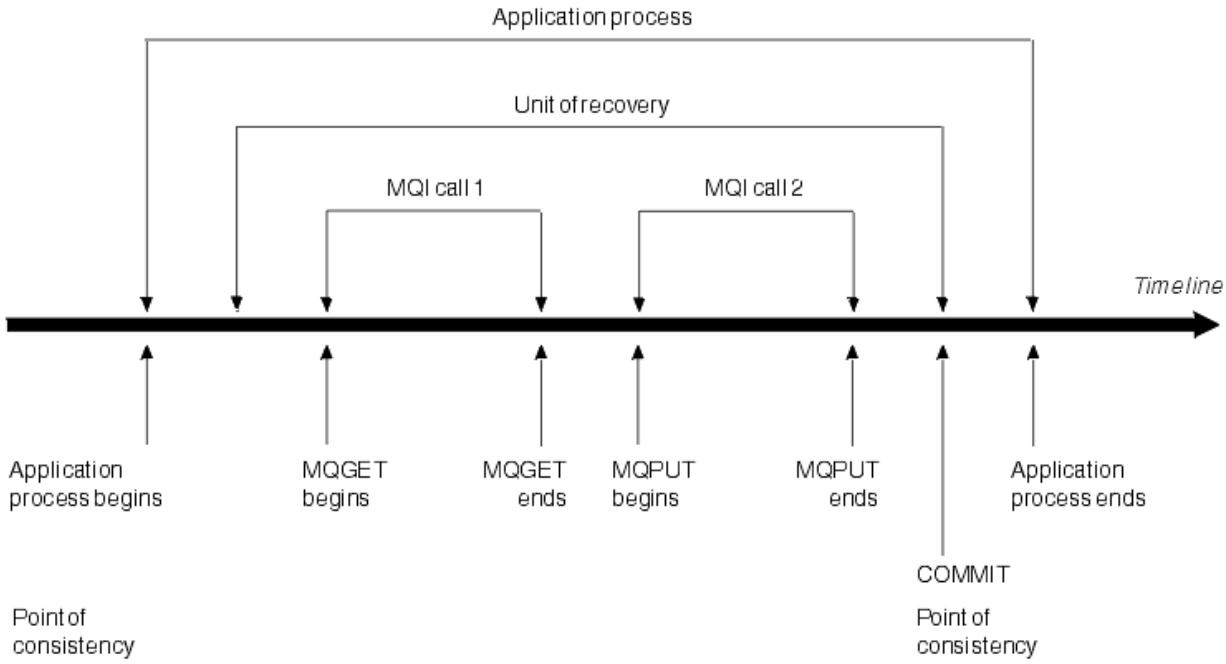


図 75. アプリケーション・プログラム内の回復単位

プログラムの始動後、または前の整合点の後に、データが初めて変更された時点で回復単位が始まります。回復単位は、次の整合点で終わります。 262 ページの図 75 は、回復単位、整合点、 およびアプリケーション・プログラムの関係を表しています。 この例では、アプリケーション・プログラムは MQI 呼び出し 1 および 2 を介してキューに変更を加えます。 アプリケーション・プログラムには、複数のリカバリー単位または 1 つのリカバリー単位を含めることができます。 いずれの場合も、すべての回復単位はコミット点で終了します。

例えば、銀行決済では 1 つの口座から別の口座に資金が移転されます。 まず、プログラムは最初の口座 (口座 A) から金額を差し引きます。 次に、その金額を 2 番目の口座 (B) に追加します。 口座 A から金額を引いた時点では、2 つの口座は整合しないため、IBM MQ はコミットすることができません。 金額が口座 B に追加された時に、2 つの口座で整合性が保たれます。 2 つのステップが完了したとき、プログラムは整合点の発生をコミットによって通知することができ、他のアプリケーション・プログラムは変更内容を認識できるようになります。

アプリケーション・プログラムが正常終了したとき、 自動的に整合点が発生します。 CICS や IMS プログラムの一部のプログラム要求も、例えば、EXEC CICS SYNCPOINT など、整合点を発生させます。

作業のバックアウト

回復単位内でエラーが発生した場合、IBM MQ はデータ変更内容を削除し、そのデータを回復単位の開始時の状態に戻すことができます。つまり、IBM MQ は作業をバックアウトします。 この流れが 263 ページの図 76 に示されています。

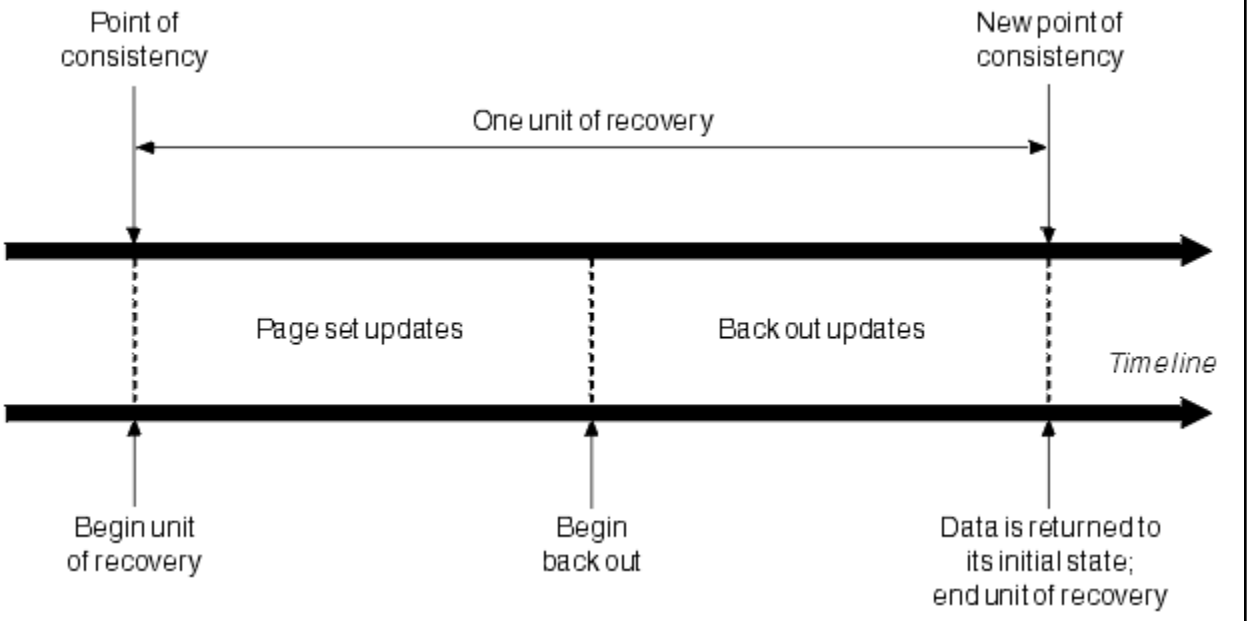


図 76. 回復単位のバックアウト

z/OS IBM MQ for z/OS での整合性維持の方法

IBM MQ for z/OS のデータは、バッチ、CICS、IMS、または TSO との間で整合性を保つ必要があります。いずれかの場所でデータが変更された場合は、他の場所にあるデータもそれに応じて変更されなければなりません。

一方のシステムがデータ変更内容をコミットする前に、 他方のシステムがこれに対応する変更を行うことができるかどうかを 確認しなければなりません。 このため、システムは互いに通信する必要があります。

2 フェーズ・コミット の間 (例えば、 CICS の下で)、1 つのサブシステムがプロセスを調整します。 そのサブシステムはコーディネーター と呼ばれ、 他方のサブシステムは参加者 と呼ばれます。 CICS または IMS は常に IBM MQ との対話のコーディネーターであり、IBM MQ は常に参加者になります。 バッチまたは TSO 環境では、IBM MQ は z/OS RRS が調整する 2 フェーズ・コミット・プロトコルに参加することができます。

(例えば TSO やバッチによる) 単一フェーズ・コミット の場合、IBM MQ は常に対話のコーディネーターとなり、コミット・プロセスを完全に制御します。

WebSphere Application Server 環境では、JMS セッション・オブジェクトのセマンティクスが、単一フェーズ・コミット調整と 2 フェーズ・コミット調整のどちらを使用するかを判別します。

CICS または IMS との整合性

IBM MQ と CICS または IMS との接続では、以下の同期点プロトコルがサポートされます。

- 2 フェーズ・コミット – 複数の資源マネージャーに所有されている資源を更新するトランザクションで使用。

  これは、標準的な分散同期点プロトコルです。 単一フェーズ・コミットに比べると、 ロギングやメッセージ・フローがより大量に発生します。
- 単一フェーズ・コミット – 単一の資源マネージャー (IBM MQ) に所有されている資源を更新するトランザクションで使用。

  このプロトコルは、ロギングおよびメッセージ・フローに関しては最適です。
- 同期点のバイパス – IBM MQ が関係しているものの、同期点をとるキュー・マネージャーでは何も行われないトランザクション (例えば、キューのブラウズ) で使用。

いずれの場合も、CICS または IMS は同期点マネージャーとして機能します。

IBM MQ または CICS との通信に IMS が使用する 2 フェーズ・コミットのステージは、以下のとおりです。

1. フェーズ 1 では、それぞれのシステムが十分な回復情報をログに 記録し終わってコミットできる状態にあるかどうかを、 システムごとに独立して判別します。

   このフェーズの終わりに、2 つのシステムは互いに通信します。 両者の状況が一致したら、次のフェーズに移ります。
2. フェーズ 2 では、変更が永続的になります。 どちらかのシステムがフェーズ 2 の途中で異常終了した場合、 再始動の回復プロセスによって操作が完了します。

2 フェーズ・コミット・プロセスを示した図

264 ページの図 77 は、2 フェーズ・コミット・プロセスを示しています。 CICS または IMS のコーディネーターのイベントは、下側の行にある IBM MQ のイベントの上に表示されます。

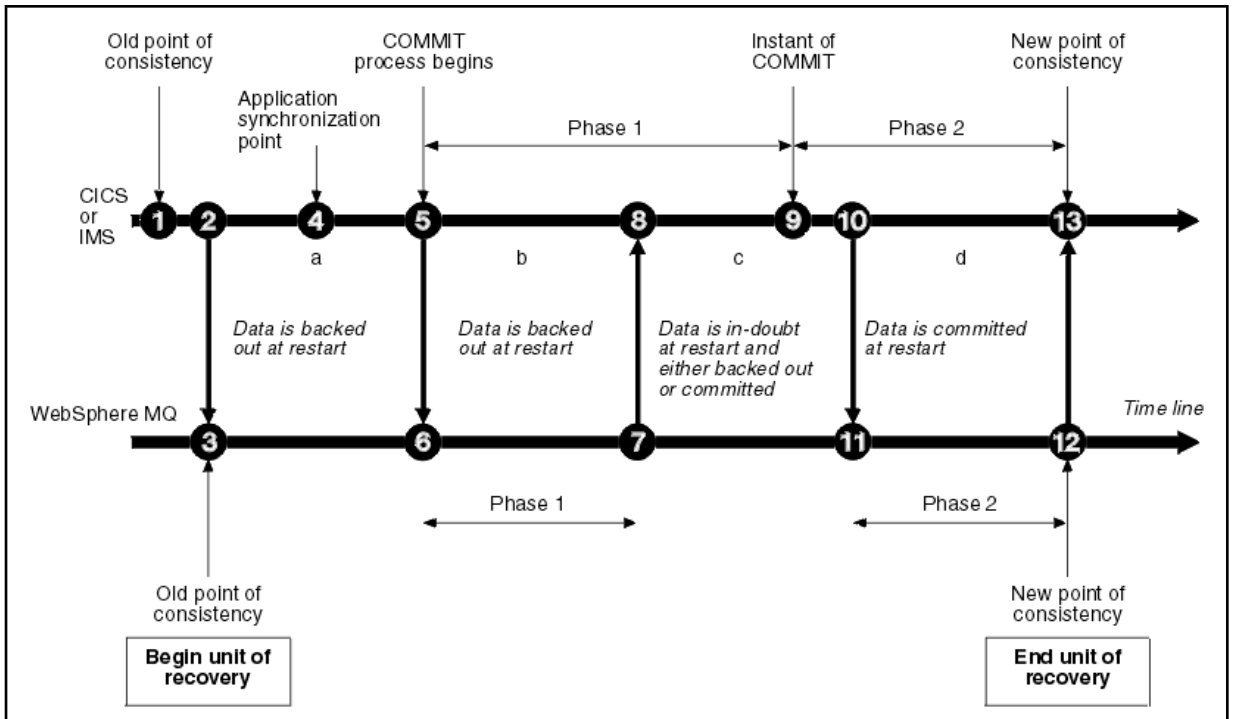


図 77. 2 フェーズ・コミット・プロセス

以下のセクションの番号は、図で示される番号に対応しています。

1. コーディネーター内のデータが、整合点に達します。
2. コーディネーター内のアプリケーション・プログラムが IBM MQ を呼び出して、メッセージを追加することによってキューを更新するよう要求します。
3. これによって、IBM MQ で回復単位が始まります。

4. コーディネーターでは処理が続けられて、 アプリケーション同期点に達します。
5. コーディネーターはコミット処理を開始します。 CICS プログラムは SYNCPOINT コマンドまたはアプリケーション正常終了を使って、コミットを開始します。 IMS プログラムは CHKP 呼び出し、SYNC 呼び出し、IOPCB に対する GET UNIQUE 呼び出し、またはアプリケーション正常終了を使ってコミットを開始することができます。 こうしてコミットのフェーズ 1 が始まります。
6. コーディネーターがフェーズ 1 の処理を開始すると、IBM MQ もフェーズ 1 を開始します。
7. IBM MQ はフェーズ 1 を正常に完了し、それをログに書き込んで、コーディネーターに通知します。
8. コーディネーターは通知を受け取ります。
9. コーディネーターは、自身のフェーズ 1 処理を正常に終了します。 両方のサブシステムともフェーズ 1 が終わってエラーから回復できる 状態になったため、 両者はデータ変更内容をコミットできる点で一致しました。 コーディネーターは、コミットの瞬間 (つまり、取り消し不能な形で 2 つのサブシステムが変更を決定したこと) をログに記録します。

   次に、コーディネーターはフェーズ 2 の処理、つまり実際のコミットを開始します。
10. コーディネーターは IBM MQ に通知して、フェーズ 2 を開始するよう要求します。
11. IBM MQ はフェーズ 2 の開始をログに記録します。
12. フェーズ 2 が正常に完了し、これが IBM MQ の新しい整合点になります。 続いて IBM MQ は、自らのフェーズ 2 処理が終わったことをコーディネーターに通知します。
13. コーディネーターは自らのフェーズ 2 処理を終了します。 これで 2 つのサブシステムによって制御されるデータは整合し、 他のアプリケーションから使用できるようになりました。

異常終了後の整合性保持の方法

異常終了後にキュー・マネージャーが再始動されるとき、キュー・マネージャーは、 異常終了時点で活動状態であった回復単位をコミットするか、 それともバックアウトするかを決定しなければなりません。 一部の回復単位については、IBM MQ は決定を行うのに十分な情報を持っています。 その他の回復単位については、WebSphere MQ は十分な情報を持っていないため、 接続が再確立されたときにコーディネーターから情報を入手する必要があります。

264 ページの図 77 は、2 つのフェーズ内に 4 つの期間 a、b、c、d があることを示しています。 回復単位の状況は終了が発生する期間により左右されます。 状況は次のいずれかになります。

未完了 (In flight)
キュー・マネージャーはフェーズ 1 の終了前 (期間 a または b) に終了しました。再始動のとき、IBM MQ は更新内容をバックアウトします。

未確定
キュー・マネージャーは、 自らのフェーズ 1 が終わってフェーズ 2 が始まる前 (期間 c) に終了しました。 コミット (図の 9 の時点) の前にエラーが発生したか、 それともコミット後に発生したかについては、 コーディネーターのみが把握しています。 コミット前にエラーが発生した場合、IBM MQ は変更内容をバックアウトする必要があります。コミット後に発生した場合、IBM MQ は変更内容をコミットする必要があります。 再始動時に、IBM MQ はこの回復単位を処理する前に、コーディネーターに対して情報を要求します。

コミット中 (In commit)
キュー・マネージャーは、自らのフェーズ 2 処理を開始した後 (期間 d) に 終了しました。変更内容はコミットされます。

バックアウト中 (In backout)
回復単位のバックアウトが始まったものの、それが完了する前にキュー・マネージャーが終了しました (この状況は図には示されていません)。再始動時に、IBM MQ は変更内容のバックアウトを続けます。

z/OS IBM MQ for z/OS で終了時に行われる処理

通常、キュー・マネージャーは STOP QMGR コマンドに応答して終了します。 他の何らかの理由でキュー・マネージャーが停止した場合は、異常終了です。

キュー・マネージャーの終了中、IBM MQ は内部的に次のコマンドを発行します。

```
DISPLAY CONN(*) TYPE(CONN) ALL WHERE (APPLTYPE NE SYSTEMAL)
```

これにより、キュー・マネージャーのシャットダウンの完了の妨げになっている可能性のあるスレッドが分かります。

SYSTEMAL は SYSTEM または CHINIT のどちらかの APPLTYPES と一致するので、SYSTEMAL と一致しないアプリケーション・タイプをフィルタリングする DISPLAY CONN コマンドは、通常シャットダウンを妨げている可能性のあるスレッドに関する情報をジョブ・ログに返します。

正常終了

正常終了では、IBM MQ はすべての活動を適切な順序で停止します。 IBM MQ は、静止モード (QUIESCE)、強制モード (FORCE)、または再始動モード (RESTART) で停止することができます。 それぞれの結果が 266 ページの表 23 に示されています。

表 *23. QUIESCE*、*FORCE*、および *RESTART* を使った終了

スレッド・タイプ	**QUIESCE**	**FORCE**	**RESTART**
活動状態のスレッド	完了するまで実行	バックアウト	バックアウト
新しいスレッド	開始可能	許可されない	許可されない
新しい接続	許可されない	許可されない	許可されない

バッチ・アプリケーションに対しては、 アプリケーションへの接続中に終了が発生するかどうかが通知されます。

CICS では、現在のスレッドは単に回復単位の終わりまで実行されます。 CICS の場合、キュー・マネージャーを静止モードで停止すると CICS アダプターも停止されます。このため、活動状態のタスクに複数の回復単位が含まれていれば、タスクは必ずしも完了するまで実行されるとは限りません。

キュー・マネージャーを強制モードまたは再始動モードで停止する場合、 新しいスレッドは割り振られず、接続中のスレッドの作業はロールバックされます。 これらのモードを使用すると、2 つのコミット処理フェーズ間にある スレッドに関して未確定の回復単位が生じる可能性があります。 これらは、IBM MQ が制御 CICS、 IMS、または RRS サブシステムと再接続されるときに解決されます。

いずれのモードでキュー・マネージャーを停止する場合も、 以下のようなステップが実行されます。

1. 接続が終了します。
2. IBM MQ はコマンドの受け入れを終了します。
3. IBM MQ は、ページ・セットの更新がすべて完了したことを確認します。
4. IBM MQ が内部的に発行する DISPLAY USAGE コマンドによって、再始動 RBA が z/OS コンソール・ログに記録されます。
5. シャットダウン・チェックポイントがとられて、BSDS が更新されます。

静止モードを指定して終了すると、未確定の回復単位には影響が及びません。 未確定の回復単位は、すべて未確定のままになります。

異常終了

異常終了によって、例えば以下のように、 データが不整合状態のままになる可能性があります。

- 回復単位が整合点に達する前に中断された。
- コミット済みデータがページ・セットにまだ書き込まれていない。
- コミットされていないデータがページ・セットにすでに書き込まれている。
- アプリケーション・プログラムが、コミット・プロセスの フェーズ 1 とフェーズ 2 の間で中断され、回復単位が未確定のままになっている。

IBM MQ は、異常終了により生じたデータ不整合を、再始動および回復のときにすべて解決します。

z/OS IBM MQ for z/OS で再始動および回復時に行われる処理

IBM MQ は再始動時に何を回復するかを決定するために、リカバリー・ログとブートストラップ・データ・セット (BSDS) を利用します。 BSDS は活動ログ・データ・セットと保存ログ・データ・セット、およびログ内の最も新しい IBM MQ チェックポイントを識別します。

再始動および回復の概要

IBM MQ が初期化された後、キュー・マネージャー再始動プロセスが以下のようにして実行されます。

- ログの初期設定
- 現在の状況の再構築
- 順方向ログ回復
- 逆方向ログ回復
- キュー索引の再構築

回復が完了すると、次のような状態になります。

- コミット済みの変更内容が、データに反映されます。
- 未確定の活動がデータに反映されます。 ただし、未確定に関する決定を IBM MQ が認識してそれに従って処理を行うまでは、データがロックされて使用できません。
- 未完了または取り消し中に中断された場合は、 変更内容がキューから取り除かれます。 メッセージは整合性のある状態になり、使用することができます。
- 新しいチェックポイントがとられます。
- 持続メッセージが入っている索引キュー用の新しい索引が作成されます (268 ページの『キュー索引の再作成』を参照してください)。

二重 BSDS を使用している場合、IBM MQ は両方の BSDS のタイム・スタンプの整合性を検査します。

- BSDS の両方のコピーとも現行のものであれば、IBM MQ は 2 つのタイム・スタンプが等しいかどうかを検査します。 等しくない場合、IBM MQ はメッセージ CSQJ120E を出して終了します。 こうした状況が発生する可能性があるのは、2 つの BSDS が別々の DASD ボリュームに保管されていて、 キュー・マネージャー停止時にそのうち 1 つのボリュームが復元されたような場合です。 IBM MQ は再始動にこの状態を検出します。
- BSDS の 1 つが割り振り解除され、 単一の BSDS を使ってロギングが続行された場合には、 問題が発生する可能性があります。 BSDS の両方のコピーが単一ボリューム上に維持されていて、そのボリュームがリストアされた場合、または BSDS の両方のコピーが別々にリストアされた場合、IBM MQ はリストアを検出しない可能性があります。 そのような場合、BSDS に記録されていないログ・レコードをシステムは認識できません。

バッチ・アプリケーションが接続を 要求した 後に 再始動が行われた場合は、 バッチ・アプリケーションにはそれが通知されません。

回復のために必要なログ範囲について

再始動中に読み取りが必要なログ・データの範囲は、多くの要因によって 決まります。

- 異常終了時には、通常、システム内に多くの不完全な作業単位があります。 前述したように、再始動処理によって、システムは整合性のとれた状態になります。 これには、未完了の作業単位のバックアウト、または未確定の作業単位の ロックの回復が含まれる場合があります。 作業単位の回復には、未完了、 バックアウト中、または未確定の作業単位について、すべての作業単位ログ・レコードが 使用できる必要があります。 IBM MQ は、古い作業単位を「延期する」ため、より狭い範囲のログ・データを使用して、作業単位の回復を実行することができます。
- 異常終了時には、通常、バッファー・プールのキャッシュにのみ保持される、 多くの持続的な更新があります。 これらの更新は、まだディスクには書き込まれていません。 これらの変更内容は、 ログから読み取られて、ページ・セットに保持されているデータに 再適用される必要があります。 チェックポイン

ト内のページ・セット回復 RBA には、 ページ・セットを整合性のある状態に更新するために必要な最小のログ RBA が記述されています。

- 古いページ・セットをシステムに導入した (例えば、メディア障害から 回復するためにページ・セットのバックアップを行った) 場合は、バックアップを行った時点からのすべての変更内容をログから読み取る必要があります。 これらの変更内容は、回復中のページ・セット内に保持されているデータに再適用されます。 ページ・セットのページ 0 に保持されているページ・セット回復 RBA には、 ページ・セットのメディア回復に必要な最小のログ RBA が記述されています。
- 共用キュー上の持続メッセージを使用する場合は、持続メッセージを保持している CFSTRUCT を回復するために、ある範囲のログ・データが必要です。 CFSTRUCT 回復を 実行するために必要になる最も古いログ・データは、古い CFSTRUCT BACKUP を 行った時点の近辺のものです。

正常実行時は、DISPLAY USAGE TYPE(DATASET) コマンドを使用して、これらの要因に関連付けられたリカバリー・ログの範囲を表示することができます (もちろん、古いページ・セットを再導入したので、情報を提供することはできません)。 異常終了した場合にキュー・マネージャーの再始動を長引かせる可能性があるすべての問題を回避するには、DISPLAY USAGE TYPE(DATASET) から出力される値を定期的にモニターします。

さらに、キュー・マネージャーは、以下の要因に関連する通知メッセージを出します。

- CSQJ160I および CSQJ161I は、実行時間の長い作業単位について警告します。
- CSQR026I および CSQR027I は、これらの実行時間の長い作業単位が正常に 延期されたかどうかに関する情報を提供します。
- CSQE040I および CSQE041E は、構造体のバックアップが古くなっていること、 およびその結果 RECOVER CFSTRUCT 操作に長時間かかることを警告します。

実行時間の長い作業単位があるアプリケーションの判別

実行時間の長い作業単位があるアプリケーションを判別することが可能です。 そのためには、DISPLAY CONN コマンドを使用します。

DISPLAY CONN コマンドは、キュー・マネージャーに接続されたすべてのアプリケーションに関する接続情報と、 実行時間の長い作業単位を現在所有するアプリケーションの判別に役立つ追加情報を戻します。 DISPLAY CONN コマンドから戻される情報は DISPLAY QSTATUS コマンドから戻される情報に似ていますが、主な違いとして DISPLAY CONN は特定のオブジェクトに関連する接続の詳細ではなく、オブジェクトに関する情報と特定の接続に関するトランザクション情報を戻します。

接続されたアプリケーションごとに、DISPLAY CONN コマンドは以下の情報を戻します。

- 接続 ID と PID を含む基本情報。
- その接続のトランザクション情報。 トランザクション作成時 (つまり、同期点で最初の MQGET/PUT が行われた時点) の 日時、およびトランザクションが最初にログに書き込まれた日時が含まれます。
- 依然として実行時間の長い作業単位があるアプリケーションを示すログ時間情報。
- 接続が現在オープンしているすべてのオブジェクトのリスト。 オブジェクトごとの詳細は、接続 ID をキーにして 別個のメッセージとして戻されます。 キューやキュー・マネージャーなど、さまざまなオブジェクト・タイプがあるため、 オブジェクトとともに表示される情報は、そのオブジェクト・タイプに固有なものとなります。

キュー索引の再作成

メッセージが順次的に検索されないキューにおける MQGET 操作を高速化するには、IBM MQ がメッセージの索引、またはキュー内のすべてのメッセージの相関 ID またはグループ ID を保持するよう指定することができます。

キュー・マネージャーが再始動されるとき、 各キューごとにこれらの索引が再作成されます。 ただし、これは持続メッセージにのみ当てはまります。 非永続メッセージは、再始動時に削除されます。 索引キューに多数の持続メッセージが含まれる場合には、 キュー・マネージャーの再始動に時間がかかる可能性があります。

CSQ6SYSP マクロの QINDXBLD パラメーターを使用することによって、キュー・マネージャー始動に対して非同期に索引が再作成されることを選択することができます。 QINDXBLD=NOWAIT を設定すると、IBM MQ は索引の再作成を待たずに再始動します。

z/OS 未確定の回復単位の解決方法

IBM MQ は、別のリソース・マネージャーとの接続を失ったときには、通常、再始動時にすべての不整合オブジェクトを回復しようとします。

IBM MQ が CICS、IMS、または RRS との接続を失った場合には、通常、再始動時にすべての不整合オブジェクトを回復しようとします。 調整を行うシステムから、未確定の回復単位の解決に必要な情報を得る必要があります。 次のセクションでは、異なる環境における解決プロセスについて説明します。

- 未確定の回復単位の CICS からの解決方法
- 未確定の回復単位の IMS からの解決方法
- 未確定の回復単位の RRS からの解決方法
- GROUP 回復単位後処理を持つ未確定の回復単位を解決する方法

未確定の回復単位の CICS からの解決方法

状況によっては、CICS が IBM MQ プロセスを実行して未確定のリカバリー単位を解決できないことがあります。 この場合、IBM MQ は、以下のメッセージの 1 つを送ります。

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E

それに続いてメッセージ CSQC408I が送られます。

これらのメッセージの詳しい意味については、IBM MQ for z/OS のメッセージ、完了コード、および理由コード の資料を参照してください。

未確定の回復単位を解決しても CICS 資源には影響ありません。 CICS はリカバリー調整を制御し、再始動時に、コミットの開始を示すログ・レコードがあったかどうかに応じて、各装置を自動的にコミットまたはバックアウトします。 未確定オブジェクトの存在は、CICS が再接続されている間は、IBM MQ リソースをロックしません。

CICS アダプターの機能の 1 つは、CICS と IBM MQ との間でデータを同期することです。 CICS への接続中にキュー・マネージャーが異常終了した場合、IBM MQ に認識させずに CICS が作業をコミットまたはバックアウトする可能性があります。 キュー・マネージャーが再始動したとき、その作業は未確定 と呼ばれる状態です。

IBM MQ は、 CICS への接続が再始動または再接続されるまで、これらの未確定リカバリー単位を解決できません (つまり、 IBM MQ リソースに対して行われた変更をコミットまたはバックアウトできません)。

未確定の回復単位を解決するプロセスは、CICS アダプターの始動時に始まります。 解決プロセスの最初に、アダプターは未確定回復単位のリストを要求します。 続いて、

- アダプターは、この接続 ID について未確定のリカバリー単位のリストを IBM MQ から受信し、それらを解決のために CICS に渡します。
- CICS は、このリストの項目と、それ自体のログの項目を比較します。 CICS は、それぞれの未確定回復単位ごとに、どんな処置が行われたかを自身のリストから判別します。

解決されたすべての単位について、IBM MQ は必要に応じてキューを更新し、対応するロックを解放します。 再始動後に、解決されない単位が残る可能性があります。 IBM MQ for z/OS の管理で説明されている方法で解決してください。

未確定の回復単位の IMS からの解決方法

IMS の未確定の回復単位を解決しても、DL/I 資源には影響ありません。 IMS はリカバリー調整を制御し、再始動時に、不完全な DL/I 作業を自動的にコミットまたはバックアウトします。 オンライン領域 (ノンファスト・パス) をコミットするかバックアウトするかは、IMS ログ・レコード・タイプ X'3730' および X'3801' がそれぞれ存在するかどうかで決定されます。 未確定回復単位が存在しても、DL/I レコードが IBM MQ との接続まで必ずしもロックされるとは限りません。

キュー・マネージャーの再始動中に、IBM MQ は未確定回復単位のリストを作成します。 IMS は、自身の残余回復項目 (RRE) のリストを作成します。 すべての項目が解決されるまで、RRE は IMS チェックポイントでログに記録されます。

IMS 領域から IBM MQ への再接続中に、IMS は IBM MQ が未確定と見なした作業単位をコミットするか、それともバックアウトするかを IBM MQ に対して示します。

未確定の単位が解決されるとき、

1. IBM MQ がある項目のコミットを決定したものの、IMS がその項目のバックアウトを決定したことが判明した場合、IBM MQ はメッセージ CSQQ010E を出します。 IBM MQ は、 IBM MQ と IMS の間のこのタイプのすべての不整合に対してこのメッセージを出します。
2. IBM MQ にいずれかの未確定単位が残っていれば、アダプターはメッセージ CSQQ008I を出します。

解決されたすべての単位について、IBM MQ は必要に応じてキューを更新し、対応するロックを解放します。

IBM MQ は、解決されなかった未確定作業のロックを保持します。 このため、重要なロックが保持された場合には、 システムのバックログが発生する可能性があります。 接続は活動状態のままであるため、IMS RRE を解決することができます。 IBM MQ for z/OS の管理で説明されている方法で、未確定のスレッドを回復してください。

すべての未確定作業は、(例えば IMS コールド・スタートのような) ソフトウェア上またはオペレーティング上の問題がない限り、解決されるはずです。 以下の 2 つの状況では、IMS 制御領域によって未確定作業が解決されます。

1. IBM MQ への接続の開始時 (同期的に解決される)。
2. プログラムの異常終了時 (非同期的に解決される)。

未確定の回復単位の RRS からの解決方法

RRS アダプターの機能の 1 つは、IBM MQ と RRS 関連の他の資源マネージャーとの間でデータの同期を保つことです。 IBM MQ がコミット・フェーズ 1 を完了した後、RRS (コミットのコーディネーター) の決定を待っている間に障害が発生した場合には、その回復単位は未確定状態になります。

RRS と IBM MQ の間で通信が再確立されたとき、コミット開始点を示すログ・レコードが存在するかどうかに応じて、RRS はそれぞれの回復単位を自動的にコミットまたはバックアウトします。 IBM MQ は、RRS との接続が再確立されるまでは、このような未確定の回復単位を解決できません (つまり、IBM MQ 資源の変更内容のコミットやバックアウトを実行できません)。

特定の状況のもとでは、RRS が未確定の回復単位を解決できない可能性があります。 この場合、IBM MQ は、以下のメッセージの 1 つを z/OS コンソールに送ります。

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

これらのメッセージの詳しい意味については、IBM MQ for z/OS のメッセージ、完了コード、および理由コード の資料を参照してください。

解決されたすべての回復単位について、IBM MQ は必要に応じてキューを更新し、対応するロックを解放します。 再始動後に、解決されない回復単位が残る可能性があります。 IBM MQ for z/OS の管理で説明されている方法で解決してください。

GROUP 回復単位後処理を持つ未確定の回復単位を解決する方法

GROUP 回復単位後処理を持つ未確定トランザクションは、GROUPUR キュー・マネージャー属性が有効であるキュー共有グループ (QSG) 内の任意のキュー・マネージャーでトランザクション・コーディネーターにより解決できます。 トランザクション・コーディネーターは再接続すると、通常は未解決で未確定のトランザクションのリストを要求し、ログからの情報を使って解決します。

GROUP 回復単位後処理で接続されたトランザクション・コーディネーターが未確定トランザクションのリストを要求すると、返されるリストはキュー共有グループ全体に存在する GROUP 回復単位後処理を持つ未確定トランザクションすべてからなります。 このリストは、それらの未確定トランザクションがどのキュー・マネージャーで始動されたかに関わらず作成されます。 このような要求を処理するキュー・マネージャーは、SYSTEM.QSG.UR.RESOLUTION.QUEUE を使用してキュー共用グループ内の他のすべての活動状態のキュー・マネージャーと通信してこのリストを作成します。 キュー・マネージャーはそれから非活動のキュー・マネージャーがあればそのログを最後のチェックポイントから読み取り、活動状態にあれば報告したはずの追加の未確定トランザクションを識別します。

トランザクション・コーディネーターが未確定トランザクションの解決を要求する場合、そのトランザクションの接続先のキュー・マネージャーはそのトランザクションが自分から開始されたものかどうかを識別し、もしそうであれば QMGR 回復単位後処理を持つトランザクションと同じ方法で解決します。 トランザクションが QSG 内の別の活動状態のキュー・マネージャーが開始したものである場合、解決を完了する要求が SYSTEM.QSG.UR.RESOLUTION.QUEUE を使用してそのキュー・マネージャーに経路指定されます。 トランザクションが QSG 内の非活動のキュー・マネージャーで開始された場合、共用キュー作業があればそれは即時に解決され、残りの専用キュー作業があればそれを解決する要求は SYSTEM.QSG.UR.RESOLUTION.QUEUE に入れられます。 非活動のキュー・マネージャーは、始動時に新しい作業を受け付ける前に、この要求を処理します。 このシナリオにおいて、元のキュー・マネージャーのログは、再始動されて要求を処理するまで、この回復単位が未確定であることを反映したままになります。

z/OS 共用キューの回復

このトピックでは、キュー共用グループ環境のさまざまなコンポーネントの IBM MQ によるリカバリーと回復力について理解することができます。

トランザクションの回復

アプリケーションが MQBACK を呼び出した場合、または (例えば EXEC CICS ROLLBACK または IMS 異常終了のために) 異常終了した場合には、キュー・マネージャーに保管されたスレッド・レベル情報を使って、未完了の作業単位がロールバックされます。 共用キューの同期点内での MQPUT 操作および MQGET 操作は、 非共用キューの更新と同じ方法でロールバックされます。

ピア回復

キュー・マネージャーに障害が発生した場合、 現在の接続先のカップリング・ファシリティー構造体から異常切断します。 (例えば物理リンクの障害や、カップリング・ファシリティーまたはその一部の電源切れによって) z/OS インスタンスとカップリング・ファシリティーとの接続に障害が起きた場合、それに関連

するキュー・マネージャーとカップリング・ファシリティーの間で接続が異常終了したと検出されます。その構造体にまだ接続されているキュー共用グループ内の他のすべてのキュー・マネージャーは、異常切断を検出して、その構造体の問題のキュー・マネージャーのためにピア回復 を開始しようとします。 このうち、実際にピア回復の開始に成功するのはただ 1 つのキュー・マネージャーのみですが、 その他のすべてのキュー・マネージャーも互いに協力して、 問題の起きたキュー・マネージャーが所有していた 作業単位を回復しようとします。

構造体にどのピアも接続されていない状態でキュー・マネージャーに障害が起きた場合、 別のキュー・マネージャーがその構造体に接続した時点、 または障害が起きたキュー・マネージャー自体が再始動した時点で回復が実行されます。

ピア回復 (ピア・レベル回復 (PLR) と呼ばれることも多い) は、構造体ごとに実行されますが、1 つのキュー・マネージャーが同時に複数の構造体の回復に関与する場合もあります。 ただし、障害発生時にどのキュー・マネージャーが 複数の構造体に接続されていたかによって、 複数の構造体の回復を共同で行うピアの組み合わせは異なる場合があります。

障害の起きたキュー・マネージャーが再始動するとき、障害発生時に接続していた構造体に再び接続して、ピア回復によって回復されなかった未解決の作業単位をすべて回復します。

ピア回復は、複数のフェーズがあるプロセスです。 最初のフェーズでは、未完了フェーズを過ぎて進行していた 作業単位が回復されます。これには、 コミット中の作業単位のメッセージをコミットする処理、および未確定作業単位のメッセージをロックする処理が含まれます。 2 番目のフェーズでは、障害が起きたキュー・マネージャーの中に 関連のある活動状態のスレッドが存在していたキューを検査し、 未完了作業単位のまだコミットされていないメッセージをロールバックして、 障害が起きたキュー・マネージャーの共用キューにあった 活動状態のハンドルに関する情報をリセットします。 つまり、問題のキュー・マネージャーが共用キューを入力専用でオープンしたことを示すすべての標識を IBM MQ はリセットして、他の活動キュー・マネージャーがそのキューを入力用にオープンできるようにします。

共用キューの定義

共用キューの属性を表すキュー・オブジェクトは、キュー共用グループが使用する共用 Db2 リポジトリーに保管されています。 IBM MQ オブジェクトを保持するために使用される Db2 表のバックアップとリカバリーのための適切な手順が用意されていることを確認してください。 また、IBM MQ CSQUTIL ユーティリティーを使用して、キュー・マネージャーに適用する MQSC コマンドを作成し、 Db2 に保管されている共有キュー定義やグループ定義などの IBM MQ オブジェクトを再定義することもできます。

ロギング

共用キューのメッセージをキュー・マネージャー・ログに記録することができるので、 キュー共用グループが永続メッセージをサポートすることができます。

カップリング・ファシリティーおよび構造体の障害

カップリング・ファシリティー (CF) 構造体について報告できる障害は、構造体の障害と接続の損失の 2 種類です。 データ共用のためのシスプレックス・サービス (XES) は、IBM MQ に、CF 構造体の障害、または構造体障害イベントによる CF 障害を通知します。 XES が接続損失のイベントを作成する場合、それは必ずしも構造体に問題があることを示しているのではなく、構造体との通信に使用できる接続がない可能性があります。 すべてのキュー・マネージャーが構造体の接続損失イベントを受け取る訳ではありません。それは、CF への接続の構成によって異なります。 接続損失イベントは、オペレーター・コマンド (例えば、VARY PATH OFFLINE または CONFIG CHP OFFLINE など) のために受け取る場合もあります。

IBM MQ によって使用される CF 構造体は、システム管理の二重化を使用するように構成できます。 これはつまり、単一の障害の場合は、システム管理フェイルオーバー処理は構造体の障害または接続損失を隠し、キュー・マネージャーに障害が通知されないということです。 二重構造の両方のインスタンスまたは接続に障害がある場合、キュー・マネージャーは適切なイベントを受け取り、単一構造の障害イベントと同じ方法で処理します。 キュー・マネージャーがイベントを処理する方法の詳細については、シナリオを参照してください。

CF または構造体障害が起きる可能性はきわめて低いものですが、実際に障害が発生した場合には、関連するアプリケーション構造体に保管されていた非持続メッセージはすべて失われます。 持続メッセージは RECOVER CFSTRUCT コマンドを使用して回復することができます。 回復可能なアプリケーション構造体に障害が起きた場合には、構造体を回復するまで、この構造体に対するアプリケーション活動は行えません。

BACKUP CFSTRUCT コマンドを使用して頻繁にバックアップを実行して、 適切な時間内で CF 構造体を確実に回復できるようにしてください。 キュー共用グループ内のすべてのキュー・マネージャーがバックアップを実行するようにも選択でき、1 つのキュー・マネージャーを専用にして、 そのキュー・マネージャーがすべてのバックアップを実行するようにも選択できます。 バックアップの実行プロセスを自動化して、バックアップが確実に定期的に実行されるようにしてください。

それぞれのバックアップは、 バックアップを実行するキュー・マネージャーの活動ログ・データ・セットに書き込まれます。 共用キュー Db2 リポジトリーは、バックアップされる CF 構造体の名前、バックアップを実行するキュー・マネージャーの名前、そのキュー・マネージャーのログにおける今回のバックアップ用の RBA 範囲、およびバックアップ時刻を記録します。

管理構造体には、アプリケーション構造体の障害時に共用キューにあった不完全な作業単位に関する情報が入っているため、管理構造体は RECOVER CFSTRUCT の処理の間、使用可能でなければなりません。 管理構造体に障害が起きた場合には、RECOVER CFSTRUCT コマンドを発行できるようにするには、キュー共用グループ内のすべてのキュー・マネージャーは管理構造体の項目を再作成する必要があります。

キュー・マネージャーは、終了しないで管理構造体の項目を自動的に再作成します。 障害発生時にキュー・マネージャーが稼働していなかった場合は、その管理構造体の項目は、同じキュー共用グループに属する、同等以上のレベルで稼働する別のキュー・マネージャーが再作成することができます。

アプリケーション構造体を回復するには、リカバリーを実行したいキュー・マネージャーに対して RECOVER CFSTRUCT コマンドを発行します。 単一の CF 構造体を回復することもできますし、いくつかの CF 構造体を同時に回復することもできます。 回復のためには、同じキュー共用グループに属するどのキュー・マネージャーでも使用できます。バックアップを実行したキュー・マネージャーや、障害のある構造体に接続したことのあるキュー・マネージャーである必要はありません。

RECOVER CFSTRUCT コマンドは、Db2 リポジトリー情報によって位置指定されるバックアップを使用して (そのため、リカバリーが実行されるキュー・マネージャーで Db2 が利用可能でなければなりません)、バックアップ・データを障害ポイントに回復します。

RECOVER CFSTRUCT コマンドは、CF 構造体にマップする共用キューに対して、バックアップ開始時から障害発生時までの間に MQPUT または MQGET を実行した、キュー共用グループ内のすべてのキュー・マネージャーから得たログ・レコードを適用することによってこれを行います。 その結果行われるログのマージでは、バックアップ以後に関係したキュー・マネージャーによって書き込まれたすべてのログ・データが読み取られるため、かなりの量のログ・データが読み取られる可能性があります。 特にバックアップ内に大量のメッセージがある場合は、頻繁に (例えば、1 時間ごとに) バックアップするよう強くお勧めします。

構造体の障害のシナリオ

シナリオ

CF 構造体に関する障害が報告される場合、接続されたキュー・マネージャーによって行われる処置は、以下に依存しています。

- z/OS のシステム間拡張サービスの構成要素によって IBM MQ に報告されたエラーの型。
- 構造体タイプ (アプリケーションまたは管理)
- キュー・マネージャー・レベル
- IBM MQ CFStruct オブジェクトの CFLEVEL (2、3、4、または 5)。 これは CFCC マイクロコードの CFLEVEL ではありません)
- CFLEVEL(5) の IBM MQ CFSTRUCT オブジェクトの RECAUTO 属性

以下のシナリオでは、管理構造体の障害が報告されたときの処理について説明します。

- 管理構造体に関する構造体障害イベントを受け取る場合、 キュー・マネージャーは終了することなく、構造体は自動的に再割り振りされ、再作成されます。 キュー・マネージャーが構造体への接続を試行すると、XES によってその構造体の新しいインスタンスが割り振られます。

  キュー・マネージャーが構造体の新しいインスタンスに接続されている場合、キュー・マネージャーはそれ自体の項目を構造体に書き込みます。 この処理は、キュー・マネージャーによって行われ、XES による再作成処理には含まれません。

  障害発生時にキュー・マネージャーが稼働していなかった場合や、管理構造体の該当部分のリカバリーが完了する前にキュー・マネージャーが終了した場合は、その管理構造体の項目は、同じキュー共用グループに属する別のキュー・マネージャーが再作成することができます。

  キュー・マネージャーの管理構造体の項目を再作成できるのは、同等以上のレベルで稼働する別のキュー・マネージャーに限られます。 キュー・マネージャーの管理構造体の項目を、そのキュー共用グループに属する別のキュー・マネージャーが再作成できない場合は、対象のキュー・マネージャーを再始動して、構造体の該当部分の再作成を完了できるようにしてください。

  すべてのキュー・マネージャーの管理構造体の項目が再作成されるまで、一部のアクションは中断されます。 中断されるアクションには、以下のアクションがあります。

  – 共用キューのオープンとクローズ。
  – 回復単位のコミットまたはバックアウト。
  – キュー・マネージャーに接続中またはキュー・マネージャーから切断中の逐次化アプリケーション。
  – アプリケーション構造体のバックアップまたはリカバリー。

  既にキュー・マネージャーに接続されている逐次化アプリケーションは、 処理を継続できます。 MQCNO_SERIALIZE_CONN_TAG_QSG または MQCNO_RESTRICT_CONN_TAG_QSG パラメーターで接続しようとしている逐次化アプリケーションは、MQRC_CONN_TAG_NOT_USABLE 戻りコードを受け取ります。

  キュー・マネージャーの管理構造体項目が再作成されると、 中断されたアクションは再開されます。

以下のシナリオでは、アプリケーション構造体の障害が報告されたときの処理について説明します。

- アプリケーション構造体に関する構造体障害イベントを受け取り、CFLEVEL が 1 または 2 である場合、キュー・マネージャーは終了します。 キュー・マネージャーを再始動する。 その構造体への再接続を試行する最初のキュー・マネージャーによって、XES が構造体の新しいインスタンスを割り振ります。
- アプリケーション構造体に関する構造体障害イベントを受け取り、CFLEVEL が 3、4 または 5 である場合、その構造体に接続しているキュー・マネージャーは実行を続けます。 障害のある構造体内のキューを使用しないアプリケーションは、通常の処理を続行できます。

  一方、障害のある構造体内のキューで操作を実行しようとするアプリケーションは、その構造体の再作成が正常に完了するまで、MQRC_CF_STRUC_FAILED エラーを受け取ります。再作成が正常に完了すると、アプリケーションはそのキューを再び開けるようになります。

  RECAUTO(YES) が定義されている CFLEVEL(5) のアプリケーション構造体では、構造体の再作成が自動的に開始されます。 そうでない場合は、RECOVER CFSTRUCT コマンドの発行時に構造体が再作成されます。

カップリング・ファシリティーの接続障害に対する回復力

カップリング・ファシリティーの接続障害に対する回復力とは?

カップリング・ファシリティーの接続障害に対する回復力とは、キュー共用グループ内のキュー・マネージャーが、終了せずにカップリング・ファシリティー構造体に対する接続が失われることを許容する機能を指します。 この機能は、可能な限り早期に共用キューにアクセスできるようにするために、接続状態がより良好な別のカップリング・ファシリティー内での構造体の再作成も試みます。

接続の一部損失とは?

IBM MQ では、接続の一部損失を、シスプレックス内の 1 つ以上のシステムで、そのシステムのアクセス対象の構造体が割り振られているカップリング・ファシリティーに対する接続が失われているものの、シスプレックス内の少なくとも 1 つのシステムでその同じカップリング・ファシリティーに対する接続が維持されている状態と定義しています。

接続の全損失とは?

IBM MQ では、接続の全損失を、シスプレックス内のどのシステムにも、カップリング・ファシリティーと、カップリング・ファシリティー内に割り振られている構造体に対する接続がない状態と定義しています。

この機能を使用可能にする理由

カップリング・ファシリティーの接続障害に対する回復力により、キュー・マネージャーが 1 つ以上のカップリング・ファシリティー構造体に対する接続を失った後でも非共用キューを引き続き使用できるので、IBM MQ の可用性が向上します。 また、カップリング・ファシリティー構造体に対する接続を失ったキュー・マネージャーは、別の使用可能なカップリング・ファシリティー内に構造体を自動的に再作成しようとするので、キュー共用グループ内の共用キューの可用性が向上します。

この機能を使用可能にする際の考慮事項

終了せずにカップリング・ファシリティー構造体に対する接続損失を許容するキュー・マネージャーは、使用可能な代替カップリング・ファシリティーがない場合には、一定の期間カップリング・ファシリティー構造体に再接続できない可能性があります。 接続を失った構造体上で定義されている共用キューは、この構造体に対する接続が復元されるまで、使用不可のままになります。 この状態では、共用キューの作業を実行するためにキュー共用グループのメンバーに接続するアプリケーションが、アクセスする必要がある共用キューを使用できないことがあります。 この状態を避けるには、カップリング・ファシリティー構造体に対する接続が失われた時点で終了するようにキュー・マネージャーを構成することをお勧めします。 終了した場合、アプリケーションはキュー共用グループの別のメンバーのうち、そのアプリケーションにとって必要な共用キューが定義されているカップリング・ファシリティー構造体に対する接続があるものに強制的に接続されます。

カップリング・ファシリティーの接続障害に対する回復力の管理

この機能を使用可能にする方法

カップリング・ファシリティー接続に対する回復力を使用可能にするには、次の作業を実行する必要があります。

1. システム管理再作成をサポートするように CFRM 結合データ・セットがフォーマットされていることを確認します。 これにより、キュー・マネージャーは、システム管理再作成を開始して、使用可能なカップリング・ファシリティー内に構造体を再作成できます。 **DISPLAY XCF,COUPLE,TYPE=CFRM** コマンドを使用して、CFRM 結合データ・セットのフォーマットを判別します。 システム管理再作成をサポートするには、次のコマンドを指定して CFRM 結合データ・セットをフォーマットする必要があります。

   ```
   "ITEM NAME(SMREBLD) NUMBER(1)"
   ```

   CFRM 結合データ・セットのフォーマットについて詳しくは、「z/OS MVS シスプレックスのセットアップ」の資料を参照してください。
2. すべての IBM MQ カップリング・ファシリティー構造体について、代替カップリング・ファシリティーが使用可能で、CFRM 設定リスト内にあることを確認します。 このようになっている場合、キュー・マネージャーは使用可能な代替カップリング・ファシリティー内に構造体を再作成し、可能な限り早期に構造体へのアクセスを復元しようとすることができます。

   IBM MQ 構造体は CFRM ポリシーにおいて ENFORCEORDER(NO) で定義する必要があります。そのように定義すれば、IBM MQ が構造体を再割り振りする必要がある場合、XCF は構成内で最適な CF を選択できます。

   構造の優先リストについて詳しくは、「z/OS MVS シスプレックスのセットアップ」の資料を参照してください。

3. 接続損失を許容する必要があるすべてのアプリケーション・カップリング・ファシリティー構造体を CFLEVEL(5) に変更します。 これは接続損失を許容できる最小レベルです。
4. **QMGR CFCONLOS** 属性および **CFSTRUCT CFCONLOS** 属性に必要な値を判別し、それらを適宜変更してください。 **QMGR CFCONLOS** 属性は管理構造体に対する接続損失を許容するかどうかを制御し、**CFSTRUCT CFCONLOS** 属性は各アプリケーション・カップリング・ファシリティー構造体が接続損失を許容するかどうかを制御します。 これらの属性をデフォルト値のままにすると、いずれかのカップリング・ファシリティー構造体に対する接続が失われた後にキュー・マネージャーは終了します。
5. アプリケーション・カップリング・ファシリティー構造体ごとの **CFSTRUCT RECAUTO** 属性に必要な値を判別して、それぞれ変更します。 この属性は、接続の全損失の後に、ログに記録されたデータを使用して、カップリング・ファシリティー構造体を自動的にリカバリーする必要があるかどうかを制御します。 この属性をデフォルト値のままにすると、接続の全損失の後にアプリケーション構造に関する自動リカバリーは実行されません。

シナリオ 1 - 管理構造体に対する接続損失

キュー・マネージャーは、終了せずに管理構造体への接続が失われることを許容できます。

管理構造体に対する接続損失を許容するように構成されているキュー・マネージャーのいずれかで、管理構造体に対する接続が失われると、キュー共用グループ内のメンバーはすべて管理構造体から切断されます。 その後、キュー共用グループ内のアクティブなキュー・マネージャーはすべて管理構造体に再接続しようとします。その結果、シスプレックス内のすべてのシステムに対する接続状態が最良のカップリング・ファシリティー内に構造体が再割り振りされ、管理構造体データが再作成されます。

注 : これは、アクティブなキュー・マネージャーがあるすべてのシステムに対する接続状態が最良のカップリング・ファシリティーになるとは限りません。

管理構造体用の CFRM 設定リスト内のカップリング・ファシリティーがいずれも使用できないなどの理由で、キュー・マネージャーが管理構造体に再接続できない場合は、キュー・マネージャーが管理構造体への再接続を正常に実行して管理構造体データを再作成するまで、一部の共用キュー操作が使用不可のままになります。 システム上で適切なカップリング・ファシリティーが使用可能になると、自動的に再接続されます。

カップリング・ファシリティーに対する接続が失われるか、構造体の割り振りに使用できる適切なカップリング・ファシリティーがない結果として、キュー・マネージャーの始動時に管理構造体への接続に失敗することは許容されません。 その後、キュー共用グループ内のアクティブなキュー・マネージャーはすべて管理構造体に再接続しようとします。その結果、別のカップリング・ファシリティーが使用可能であればそのカップリング・ファシリティーに構造体が再割り振りされ、管理構造体データが再作成されます。

シナリオ 2 - アプリケーション構造に対する接続損失

CFLEVEL(5) 以上のアプリケーション構造に対する接続損失は、キュー・マネージャーを終了しなくても許容できます。 **CFLEVEL(4)** 以下のアプリケーション構造や、接続損失を許容するように構成されていない **CFLEVEL(5)** 構造に接続しているキュー・マネージャーは、構造体に対する接続が失われると、理由コード 00C510AB で異常終了します。

接続損失を許容するように構成されているアプリケーション構造に対する接続が失われると、構造体に対する接続を失ったキュー・マネージャーはすべて切断されます。 その後のキュー・マネージャーの動作は、接続の一部損失か全損失かによって異なります。

アプリケーション構造に対する接続の一部損失

接続の一部損失と判別されると、構造体に対する接続を失ったキュー・マネージャーは、より良い接続状態の別のカップリング・ファシリティーに構造体を移動するために、システム管理再作成を開始しようとします。 この再作成が正常に実行されると、構造体内の持続メッセージと非持続メッセージの両方ともこの別のカップリング・ファシリティーにコピーされ、構造体上のキューへのアクセスが復元されます。 接続を失っていないキュー・マネージャーは構造体に接続されたままですが、システム管理の再作成プロセスの間、その構造体にアクセスする操作は遅延されます。

より良い接続状態の別のカップリング・ファシリティーにアプリケーション構造を再作成できない場合や、別のカップリング・ファシリティー内に構造体を再作成した後でも一部のキュー・マネージャーが

接続していない場合は、カップリング・ファシリティーに対する接続が復元されるまで、構造体に対する接続がないキュー・マネージャーは、構造体上で定義されたキューを使用できないままになります。構造体が使用可能になると、キュー・マネージャーは構造体に自動的に再接続し、構造体上で定義されている共用キューへのアクセスが復元されます。

アプリケーション構造に対する接続の全損失

シスプレックス内のすべての MVS システムが、アプリケーション構造が割り振られているカップリング・ファシリティーへの接続を失った場合は、z/OS は構造体への再接続が試行されるつど、カップリング・ファシリティーから構造体を割り振り解除します。 アプリケーションが共用キューを開こうとしたり、新しいカップリング・ファシリティー・リソースが使用可能になったという通知をシステムから受け取ったりするなどの理由で、キュー・マネージャーが構造体への再接続を試行する可能性があります。 したがって、アプリケーション構造への接続の全損失の後に、影響を受けている構造体内の非持続メッセージがすべて失われるおそれがあります。

RECAUTO(YES) を使用して定義されていれば、リカバリー可能アプリケーション構造は接続の全損失後に自動的にリカバリーされます。 構造体の割り振りに使用できる代替カップリング・ファシリティーがある場合や、そのようなカップリング・ファシリティーが使用可能になった時点で、リカバリーはほぼ即時に開始されます。 構造が **RECAUTO(YES)**で定義されていない場合は、 **RECOVER CFSTRUCT** コマンドを発行してリカバリーを開始できます。 この場合は、構造体内の持続メッセージはすべてリカバリーされますが、非持続メッセージはすべて失われます。 このプロセスにはキュー・マネージャー・ログの読み取りが関係するので、完了するまで多少時間がかかります。したがって、構造体のバックアップを定期的に取って、構造体上の共用キューへのアクセスが復元されるまでに要する時間を短縮することをお勧めします。

アプリケーションが構造体上で定義されている共用キューを開こうとしたり、新しいカップリング・ファシリティー・リソースが使用可能になったという通知をシステムから受け取ったりすると、即時にキュー・マネージャーはリカバリー不能アプリケーション構造に再接続しようとします。 構造体の割り振りに使用できる適切なカップリング・ファシリティーがある場合は、新しい構造体が割り振られ、構造体上で定義されている共用キューへのアクセスが復元されます。 リカバリー不能構造体内で定義されているキューには持続メッセージを入れられないので、共用キュー上のメッセージはすべて失われます。

操作上の動作

特定のカップリング・ファシリティー構造体に対する接続損失を許容するように構成されている IBM WebSphere MQ 7.1 またはそれ以降のキュー・マネージャーが接続を失うと、キュー共用グループのメンバーは障害から自動的にリカバリーして構造体に再接続しようとします。 接続状態がより良好な別のカップリング・ファシリティーが使用可能な場合、このアクティビティーには、このカップリング・ファシリティー内に構造体を再割り振りすることが含まれます。 しかし、接続損失からリカバリーするには、依然としてオペレーター介入が必要になることがあります。

通常、以下のオペレーター・アクションが必要になります。

1. 接続を失うことになった失敗の原因の解決。
2. シスプレックス内のすべてのシステムで、IBM MQ 構造体を割り振れるカップリング・ファシリティーが使用可能になっていることの確認。

接続損失イベントの後で別のカップリング・ファシリティーに自動的に再割り振りされた構造体を、キュー共用グループ内のすべてのキュー・マネージャーに対する最良の接続があるカップリング・ファシリティーに移動できます。 必要な場合は、 z/OS 「MVS」 システム・コマンド解説書に記載されているように、システム管理再作成コマンド **SETXCF START,REBUILD** を開始することによってこれを行うことができます。

アプリケーション構造に対する接続の一部損失の場合は、構造体に対する接続を失ったキュー・マネージャーは、システム管理再作成を開始しようとします。 構造体に現在接続しているアクティブ・キュー・マネージャーすべてに対する接続が別のカップリング・ファシリティーにある場合、このプロセスは、このカップリング・ファシリティーに構造体を割り振るだけになります。 したがって、キュー共用グループ内の大多数のキュー・マネージャーがアプリケーション構造への接続を失った場合、引き続き元の構造体に接続しているキュー・マネージャーがあるために、別のカップリング・ファシリティー内に構造体を再作

成できない可能性があります。 この状態では、元の構造体に接続しているキュー・マネージャーをシャットダウンして構造体を再作成できるようにするか、**RESET CFSTRUCT ACTION(FAIL)** コマンドを出して構造体を失敗させることができます。 **RECOVER CFSTRUCT** コマンドを出して、該当する構造体に対してリカバリーを開始できます。

注 : 構造体を失敗させてからリカバリーすると、構造体上の非持続メッセージはすべて失われます。

z/OS IBM MQ for z/OS におけるセキュリティーの概念

このトピックを使用して、IBM MQ のセキュリティーの重要性、システムに十分のセキュリティー設定がなされていない場合の影響について理解してください。

IBM MQ 資源を保護する必要がある理由

IBM MQ は、潜在的に価値のある情報の転送を処理します。 セキュリティーを適用することにより、IBM MQ が所有して管理する資源が無許可アクセスから確実に保護されます。 このようなアクセスにより、情報の紛失や開示が発生することがあります。

次のすべての資源が、無許可のユーザーやプロセスによってアクセスされたり変更されたりしないようにしてください。

- IBM MQ への接続
- キュー、プロセス、および名前リストなどの IBM MQ オブジェクト
- IBM MQ 伝送リンク、つまり IBM MQ チャネル
- IBM MQ システム制御コマンド
- IBM MQ メッセージ
- メッセージに関連したコンテキスト情報

必要なセキュリティーを提供するため、IBM MQ は z/OS システム許可機能 (SAF) を使用して、許可要求を外部セキュリティー・マネージャー (ESM)、例えば、Security Server (以前の RACF) に経路指定します。 IBM MQ は、独自のセキュリティー検査を行いません。 分散キューイングかクライアントを使用する場合は、追加のセキュリティー手段が必要になることがあります。 IBM MQ には、そのためのチャネル認証レコード、チャネル出口、MCAUSER チャネル属性、TLS が備えられています。

オブジェクトへのアクセスを認めるかどうかの判断は ESM により行われ、IBM MQ は、その決定に従います。 ESM が判断できないときは、IBM MQ はそのオブジェクトへのアクセスを認めません。

IBM MQ リソースを保護しない場合に起きる事柄

セキュリティー対策を講じない場合、すべての ユーザーがすべての 資源のアクセスや変更を行えるようになることを十分想定しなければなりません。 この対象には、ローカル・ユーザーだけでなく、 分散キューイングやクライアントを使用するリモート・システムのユーザーも含まれます。 この種の場所のログオン・セキュリティー制御は z/OS より緩やかだからです。

セキュリティー検査を使用可能にするには、以下の処理を行わなければなりません。

- ESM (Security Server など) をインストールしてアクティブ化する。
- Security Server 以外の ESM を使用する場合は、MQADMIN クラスを定義する。
- MQADMIN クラスをアクティブ化する。

大/小文字混合の資源名を使用することが企業にとって有益かどうかを考慮する必要があります。 ESM プロファイルで大/小文字混合の資源名を使用する場合、MXADMIN クラスを定義してアクティブ化する必要があります。

z/OS データ・セット暗号化

データ・セット暗号化 (DSE) は z/OS データ・セットを暗号化する機能を備えているため、そのデータ・セットに含まれているデータは、特定の許可が付与されたユーザー ID しか表示したり変更したりすることはできません。 これにより、ファイル・システム内に格納されたデータの暗号化が可能となり、正当なビジ

ネス・ニーズおよびデータ・セット自体を管理する権限を持つユーザーに対する機密情報の不適切な開示を防ぐことができます。

IBM MQ for z/OS 9.1.4 より前の IBM MQ for z/OS は、IBM MQ メッセージの基本永続性メカニズムを提供するアクティブ・ログ、ページ・セット、および共用メッセージ・データ・セット (SMDS) での DSE の使用をサポートしていません。

代わりに、Advanced Message Security は IBM MQ メッセージングのエンドツーエンドの暗号化解決を提供し、それは全体的な IBM MQ ネットワーク、静止時および実行時の IBM MQ プロセス内におけるフライト状態でデータの暗号化を包含します。

IBM MQ サブシステムで使用される他の VSAM および順次データ・セットは DSE を使用して暗号化できます。 以下に例を示します。

- ブートストラップ・データ・セット (BSDS)
- 始動時に CSQINPx DDNAME を使用して読み取られるシステム構成 (MQSC) コマンドを保持する順次ファイル
- IBM MQ アーカイブ・ログ。これは、多くの場合、監査目的で IBM MQ ログ・データの長期アーカイブに使用されます。

DSE を使用した暗号化は、データ・セット・キー・ラベルで定義されたデータ・クラスを割り振ることにより実行できます。 詳しくは、『ログ・アーカイブ・ストレージの計画』を参照してください。

IBM MQ for z/OS 9.1.4 以降では、IBM MQ for z/OS は、以前のリリースで提供されていたサポートに加えて、アクティブ・ログおよびページ・セットでの DSE の使用をサポートしています。

IBM MQ for z/OS は、共用メッセージ・データ・セット (SMDS) での DSE の使用をサポートしていません。

データ・セット暗号化による IBM MQ for z/OS での保存データの機密性 のセクションを参照してください。 for more information.

関連概念

セキュリティーの概念

チャネル認証レコード

IBM MQ 上で z/OS オブジェクトを処理する権限

暗号セキュリティー・プロトコル: TLS

関連タスク

z/OS でのセキュリティーのセットアップ

リンク・レベル・セキュリティーとアプリケーション・レベル・セキュリティーの比較

関連資料

IBM MQ for z/OS のメッセージ

z/OS IBM MQ for z/OS におけるセキュリティー管理とセキュリティー・オプション

IBM MQ サブシステム全体に対してセキュリティーをオンにするかどうかを指定することができます。さらに、キュー・マネージャーまたはキュー共用グループのレベルでセキュリティー検査を実行するかどうかを指定することもできます。 また、API 資源のセキュリティーに関連して検査されるユーザー ID の数も制御できます。

サブシステムのセキュリティー

サブシステム・セキュリティーでは、 キュー・マネージャー全体でセキュリティー検査を行うかどうか指定することにより制御します。 セキュリティー検査 (テスト・システムなどの) が必要ない場合や、IBM MQ に接続できるすべての資源 (クライアントやチャネルを含む) に対するセキュリティーだけでよい場合は、キュー・マネージャーかキュー共用グループのセキュリティー検査をオフにして、必要以上のセキュリティー検査が行われないようにすることができます。

この方法で行えることは、セキュリティーを完全にオフにすることと、 他のセキュリティー検査を実行するかどうかを決めることだけです。 つまり、キュー・マネージャーかキュー共用グループの検査をオフにすると、他の IBM MQ 検査も行われません。逆にオンにしておくと、IBM MQ で他の IBM MQ 資源のセキュリティー要件が検査されます。

コマンドなどの特定の資源の集合に対して、セキュリティーのオン/オフを切り替えることもできます。

キュー・マネージャー・レベルまたはキュー共用グループ・レベルの検査

キュー・マネージャー・レベルかキュー共用グループ・レベルでセキュリティーを実装できます。 キュー共用グループ・レベルでセキュリティーを実装すると、 グループ中のすべてのキュー・マネージャーの間で同一のプロファイルが共用されます。 したがって、定義して保守するプロファイルの数が少なくて済み、セキュリティー管理が楽になります。 また、キュー共用グループに新規キュー・マネージャーを追加するときに、 既存のセキュリティー・プロファイルを継承するので、追加が容易になります。

インストール要件によっては、両方を組み合わせて実装することもできます。例えば、マイグレーションの際や、キュー共用グループ中の特定のキュー・マネージャーに他のキュー・マネージャーとは違うレベルのセキュリティーが必要な場合などが該当します。

キュー共用グループ・レベルのセキュリティー

キュー共用グループ・レベルのセキュリティー検査は、 キュー共用グループ全体に実行されます。 この方法を使用すると、 定義する必要のあるセキュリティー・プロファイルの数が少なくて済むので、セキュリティー管理を単純化できます。 ユーザー ID が特定の資源を使用するための許可はキュー共用グループ・レベルで行われ、 ユーザー ID がどのキュー・マネージャーを使用して資源にアクセスするかはまた別問題です。

例えば、ユーザー ID SERVER でサーバー・アプリケーションを実行している場合に、SERVER.REQUEST というキューにアクセスし、シスプレックス中の個々の z/OS イメージ上で SERVER のインスタンスを実行するとします。 個々のキュー・マネージャーで個別に SERVER が SERVER.REQUEST をオープンすることを 許可する (キュー・マネージャー・レベルのセキュリティー) のではなく、 キュー共用グループ・レベルだけでアクセス許可を済ませることができます。

キュー共用グループ・レベルのセキュリティー・プロファイルを使用して、 ローカルでも共用でも、すべてのタイプの資源を保護できます。

キュー・マネージャー・レベルのセキュリティー

キュー・マネージャー・レベルのセキュリティー・プロファイルを使用して、 ローカルでも共用でも、すべてのタイプの資源を保護できます。

両方のレベルの組み合わせ

キュー・マネージャーとキュー共用グループの両方のレベルのセキュリティーを組み合わせて使用できます。

キュー共用グループ・レベルのセキュリティー設定を、そのグループのメンバーである特定のキュー・マネージャーに関してオーバーライドできます。 したがって、個々のキュー・マネージャーに、 同じグループ中の他のキュー・マネージャーとは違うレベルのセキュリティー検査を実行できます。

詳細については、『キュー共用グループ・レベル・セキュリティーまたはキュー・マネージャー・レベル・セキュリティーを制御するためのプロファイル』を参照してください。

検査されるユーザー ID の数の制御

RESLEVEL は Security Server プロファイルの 1 つで、IBM MQ 資源セキュリティーについて検査されるユーザー ID の数を制御します。 通常は、ユーザーが IBM MQ 資源にアクセスしようとすると、Security Server により関係のある 1 つまたは複数のユーザー ID が検査され、その資源に対するアクセスが許可されているかどうかが調べられます。 RESLEVEL プロファイルを定義すると、 ユーザー ID がゼロ、1 つ、または適切な場合は 2 つ検査されるよう制御できます。

この種の制御は、接続別に行われ、接続している間中持続します。

RESLEVEL プロファイルはそれぞれのキュー・マネージャーごとに 1 つあります。 ユーザー ID がこのプロファイルに対するアクセス権を持つと、 この制御が実装されます。

大/小文字混合または大文字の IBM MQ RACF クラス

大/小文字混合の RACF プロファイル・サポートを使用できるようになり、大/小文字混合の資源名を使用し、さらに IBM MQ RACF プロファイルを定義してそれらを保護することが可能になりました。

以下のいずれかを選択できます。

- これまでのリリースと同様、大文字だけの IBM MQ RACF クラスを引き続き使用する。
- 新しく導入された大/小文字混合の IBM MQ RACF クラスを使用する。

大/小文字混合の RACF プロファイルを使用しない場合でも、IBM MQ for z/OS で大/小文字混合のリソース名を使用できます。 ただし、これらのリソース名は、大文字の IBM MQ クラスの総称 RACF プロファイルによってのみ保護できます。 大/小文字混合の IBM MQ RACF プロファイル・サポートを使用する場合は、大/小文字混合の IBM MQ クラスで IBM MQ RACF プロファイルを定義することにより、よりきめ細かいレベルの保護を提供できます。

z/OS IBM MQ for z/OS で保護できるリソース

キュー・マネージャーの開始時か、オペレーター・コマンドによって指示された時点で、 保護したい資源が IBM MQ for z/OS により判別されます。

個々のキュー・マネージャーに実行されるセキュリティー検査の内容を制御できます。 例えば、実動キュー・マネージャーでは複数のセキュリティー検査を実装し、 テスト・キュー・マネージャーでは 1 つも実装しないようにすることができます。

接続のセキュリティー

接続セキュリティー検査は、アプリケーション・プログラムがキュー・マネージャーへの接続を試みたときに行われます。 検査は、MQCONN 要求または MQCONNX 要求を発行することによって行われるか、チャネル・イニシエーター、CICS、または IMS アダプターが接続要求を発行したときに実行されます。

キュー・マネージャー・レベルのセキュリティーを使用している場合、特定のキュー・マネージャーの接続セキュリティー検査をオフにすることができます。 ただし、オフにすると、すべてのユーザーがそのキュー・マネージャーに接続できるようになります。

CICS アダプターの場合、 接続セキュリティー検査で使用されるのは、CICS アドレス・スペースのユーザー ID のみで、 個々の CICS 端末のユーザー ID は使用されません。 IMS・アダプターの場合、IMS 制御領域または従属領域が IBM MQ に接続されると、 IMS アドレス・スペース・ユーザー ID が検査されます。 チャネル・イニシエーターの場合、 チャネル・イニシエーターのアドレス・スペースで使用されるユーザー ID が検査されます。

キュー・マネージャー・レベルかキュー共用グループ・レベルのどちらか一方で、 接続のセキュリティー検査をオン/オフにすることができます。

コマンドのセキュリティー

コマンド・セキュリティー検査は、ユーザーが IBM MQ for z/OS で MQSC コマンドおよび PCF コマンドを発行できるソースに記載されているいずれかのソースから MQSC コマンドを発行すると実行されます。 282 ページの『コマンド資源のセキュリティー』で説明されているように、 コマンドで指定されている資源に関する検査を別個に行うことができます。

コマンドの検査をオフにすると、 コマンドの発行側にそのコマンドの発行権限があるかどうか検査されません。

MQSC コマンドをコンソールから入力する場合は、そのコンソールに z/OS SYS コンソール権限属性がなければなりません。 コマンドを CSQINP1 または CSQINP2 データ・セットから発行するか、またはキュー・

マネージャーにより内部的に発行されると、 すべてのセキュリティー検査から免除されます。 一方 CSQINPX のコマンドは、チャネル・イニシエーターのアドレス・スペースのユーザー ID を使用します。 通常のデータ・セット保護を使用して、 誰がこれらのデータ・セットを更新できるかを制御する必要があります。

キュー・マネージャー・レベルかキュー共用グループ・レベルのどちらか一方で、 コマンドのセキュリティー検査をオン/オフにすることができます。

コマンド資源のセキュリティー

ローカル・キューの定義など、一部の MQSC コマンドは、IBM MQ 資源の操作に関係しています。 コマンド資源のセキュリティーが活動状態の場合は、 資源に関係するコマンドを発行するたびに、 ユーザーがその資源の定義に変更を加えることを許可されているかどうかが IBM MQ により検査されます。

コマンド資源のセキュリティーを使用すると、命名規格を実施するのに役立ちます。 例えば、給与計算の管理者は、名前の先頭が "PAYROLL" のキューだけを削除したり定義したりできます。 コマンド資源のセキュリティーが非活動状態の場合は、 コマンドによって操作される資源のセキュリティー検査は行われません。 コマンド資源のセキュリティーとコマンド・セキュリティーを混同しないでください。 これらのセキュリティーは互いに無関係です。

コマンド資源のセキュリティー検査をオフにしても、 コマンドに関係していない処理の際に限って行われる資源検査には影響しません。

キュー・マネージャー・レベルかキュー共用グループ・レベルのどちらか一方で、 コマンド資源のセキュリティー検査をオン/オフにすることができます。

チャネル・セキュリティーに関する考慮事項

チャネル・セキュリティー

チャネルを使用する場合、 使用しようとしている通信プロトコルに応じたセキュリティー機能を使用できます。 TCP を使用する場合は、TLS は使用することができますが、この通信プロトコル用のセキュリティー機能は備えられていません。 APPC を使用する場合は、 送信側 MCA からネットワークを介して検証先 MCA へユーザー ID 情報を流すことができます。

どちらのプロトコルの場合も、どのユーザー ID をセキュリティー検査したいのか、 およびいくつのユーザー ID をセキュリティー検査したいのかを指定できます。 この場合も、使用しようとしているプロトコル、チャネル定義時の指定内容、 およびチャネル・イニシエーターの RESLEVEL 設定値に応じた選択項目を使用できます。

使用可能なチャネル・セキュリティー・タイプについて詳しくは、チャネル認証レコードおよびセキュリティー出口の概要を参照してください。

関連資料

282 ページの『IBM MQ for z/OS での API リソース・セキュリティー』
アプリケーションで MQOPEN 呼び出しか MQPUT1 呼び出しが行われてオブジェクトがオープンされる際に、資源が検査されます。 オブジェクトをオープンするのに必要なアクセス権は、キューのオープン時に指定したオープン・オプションに応じて異なります。

z/OS *IBM MQ for z/OS での API リソース・セキュリティー*

アプリケーションで MQOPEN 呼び出しか MQPUT1 呼び出しが行われてオブジェクトがオープンされる際に、資源が検査されます。 オブジェクトをオープンするのに必要なアクセス権は、キューのオープン時に指定したオープン・オプションに応じて異なります。

API 資源のセキュリティーは、さらに以下の検査に分けられます。

- キュー
- Process
- 名前リスト

- 代替ユーザー
- コンテキスト

キュー・マネージャーのオブジェクトのオープン時や、 ストレージ・クラス・オブジェクトのアクセス時には、セキュリティー検査は実行されません。

キュー

キューのセキュリティー検査により、だれがキューをオープンできるか、 またどのオプションを指定してキューをオープンできるかを制御できます。 例えば、ユーザーが PAYROLL.INCREASE.SALARY というキューをオープンして、 そのキュー上のメッセージをブラウズできる (MQOO_BROWSE オプションを使用) が、 そのキューからメッセージを除去することはできない (MQOO_INPUT_* オプションの 1 つを使用) ようにすることができます。 キューの検査をオフにすると、どの有効なオープン・オプション (MQOPEN 呼び出しか MQPUT1 呼び出しの有効な MQOO_* オプション) を指定しても、 すべてのユーザーがどのキューでもオープンできます。

キュー・マネージャー・レベルかキュー共用グループ・レベルのどちらか一方で、 キューのセキュリティー検査をオン/オフにすることができます。

プロセス

プロセスのセキュリティー検査は、 ユーザーがプロセス定義オブジェクトをオープンする際に実行されます。 プロセスの検査をオフにすると、すべてのユーザーがどのプロセスでもオープンできます。

キュー・マネージャー・レベルかキュー共用グループ・レベルのどちらか一方で、 プロセスのセキュリティー検査をオン/オフにすることができます。

名前リスト

名前リストのセキュリティー検査は、ユーザーが名前リストをオープンする際に実行されます。 名前リストの検査をオフにすると、すべてのユーザーがどの名前リストでもオープンできます。

キュー・マネージャー・レベルかキュー共用グループ・レベルのどちらか一方で、 名前リストのセキュリティー検査をオン/オフにすることができます。

代替ユーザー

代替ユーザーのセキュリティーにより、特定のユーザー ID が、別のユーザー ID の権限を使用して IBM MQ オブジェクトをオープンできるかどうかを制御できます。

以下に例を示します。

- ユーザー ID PAYSERV の下で実行するサーバー・プログラムで、 ユーザー ID USER1 がキューに挿入した要求メッセージがそのキューから取り出される。
- サーバー・プログラムは、要求メッセージを読み取ると、要求を処理し、要求メッセージで指定されている応答先キューに応答を書き戻します。
- 応答先キューのオープンを許可するのに、独自のユーザー ID (PAYSERV) を使用する代わりに、 サーバーで他の特定のユーザー ID (この例では USER1) を指定できる。 この例では、代替ユーザーのセキュリティーにより、 ユーザー ID PAYSERV が応答先キューをオープンする際に 代替ユーザー ID としてユーザー ID USER1 を指定できるかどうかが制御されます。

代替ユーザー ID は、 オブジェクト・ディスクリプター (MQOD) の *AlternateUserId* フィールドで指定します。

すべての IBM MQ オブジェクト (プロセスや名前リストなど) に対して代替ユーザー ID を使用できます。 代替ユーザー ID は、他の資源マネージャーによって使用されるユーザー ID (CICS セキュリティー用や z/OS データ・セット・セキュリティー用の ID) には影響しません。

代替ユーザーのセキュリティーが活動状態でない場合は、 すべてのユーザーが他のどのユーザー ID でも代替ユーザー ID として使用できます。

キュー・マネージャー・レベルかキュー共用グループ・レベルのどちらか一方で、 代替ユーザーのセキュリティー検査をオン/オフにすることができます。

Context

コンテキストとは、特定のメッセージに適用できる情報のことで、 メッセージの一部であるメッセージ・ディスクリプター (MQMD) 中に含まれます。 コンテキスト情報は、以下の 2 つのセクションから構成されます。

ID セクション
初めてメッセージをキューに挿入したアプリケーションのユーザー。 以下のフィールドから成ります。

- *UserIdentifier*
- *AccountingToken*
- *ApplIdentityData*

起点セクション
キュー上の現在の格納場所にメッセージを挿入したアプリケーション。 以下のフィールドから成ります。

- *PutApplType*
- *PutApplName*
- *PutDate*
- *PutTime*
- *ApplOriginData*

MQPUT 呼び出しか MQPUT1 呼び出しが行われる際に、 アプリケーションでコンテキスト・データを指定できます。 このデータは、アプリケーションが生成するか、別のメッセージから渡されるか、 またはデフォルトでキュー・マネージャーが生成します。 例えば、サーバー・プログラムでコンテキスト・データを使用して、要求側の ID を検査できます。 つまり、このメッセージが正しいアプリケーションから送信されたかどうかを検査できます。 普通 *UserIdentifier* フィールドは、 代替ユーザーのユーザー ID を判別するのに使用します。

コンテキストのセキュリティーを使用すると、 MQOPEN または MQPUT 呼び出し時にユーザーがコンテキスト・オプションを指定できるかどうかを制御できます。 コンテキストのオプションの詳細については、メッセージ・コンテキストに関連する MQOPEN オプションを参照してください。 コンテキストに関連するメッセージ記述子フィールドの説明については、MQMD – メッセージ記述子を参照してください。

コンテキストのセキュリティー検査をオフにすると、 すべてのユーザーが、キューのセキュリティーによって許可されているどのコンテキスト・オプションでも使用できます。

キュー・レベル、キュー・マネージャー・レベル、またはキュー共用グループ・レベルのいずれかで、コンテキストのセキュリティー検査をオン/オフにすることができます。

z/OS での可用性

IBM MQ for z/OS には、高可用性のための多数の機能があります。 このトピックでは、可用性に関するいくつかの考慮事項について説明します。

IBM MQ には、キュー・マネージャーまたはチャネル・イニシエーターで障害が発生した場合のシステム使用可能性を向上させることができるいくつかの機能があります。 これらの機能の詳細については、以下のセクションを参照してください。

- シスプレックスの考慮事項
- 共用キュー
- 共用チャネル
- IBM MQ ネットワークの可用性

- z/OS 自動再始動マネージャー (ARM) の使用
- z/OS 拡張回復機能 (XRF) の使用
- キュー共用グループの回復に対する z/OS GROUPUR 属性の使用
- 可用性に関する詳細が記載されている資料

シスプレックスの考慮事項

シスプレックス では、多数の z/OS オペレーティング・システム・イメージがカップリング・ファシリティーを使用して連絡することにより、単一のシステム・イメージとして協調して稼働します。 IBM MQ は、可用性を向上するためにシスプレックス環境の機能を使用することができます。

キュー・マネージャーと特定の z/OS イメージ間の類縁性を取り除くことによって、キュー・マネージャーは、イメージ障害が起こったときに別の z/OS イメージ上で再始動できるようになります。 以下の条件が満たされる場合、再始動メカニズムは手動でも、ARM を使用しても、 またはシステム・オートメーションを使用してもかまいません。

- すべてのページ・セット、ログ、ブートストラップ・データ・セット、コード・ライブラリー、 およびキュー・マネージャー構成データ・セットが共用ボリューム上に定義されている。
- サブシステム定義には、シスプレックス・スコープとシスプレックス内に固有の名前がある。
- IPL 時にすべての z/OS イメージにインストールされる 早期コード のレベルは、同じレベルです。
- TCP 仮想 IP アドレス (VIPA) がシスプレックス内の各 TCP スタック上で使用できる。デフォルトのホスト名でなく VIPA を使用するように、IBM MQ TCP リスナーとインバウンド接続が構成されている。

シスプレックスでの TCP の使用について詳しくは、「*TCP/IP in a sysplex*」(SG24-5235) (IBM Redbooks® 資料) を参照してください。

さらに、シスプレックス内の異なるオペレーティング・システム・イメージ上で稼働する複数のキュー・マネージャーが、 キュー共用グループとして動作するように構成することができます。 この場合、共用キューと共用チャネルを活用でき、高可用性およびワークロード・バランシングに有効です。

共有キュー

キュー共用グループ環境では、 アプリケーションからキュー共用グループ中のどのキュー・マネージャーにも接続できます。 キュー共用グループ中のすべてのキュー・マネージャーが同じ共用キューの集合にアクセスできるので、 アプリケーションが特定のキュー・マネージャーの可用性に左右されることはありません。 キュー共用グループ中のすべてのキュー・マネージャーがどのキューにもサービスを行えます。 したがって、キュー・マネージャーの 1 つが停止した場合の可用性が向上しています。 なぜなら、キュー共用グループ中の他のキュー・マネージャーはすべて、キューの処理を引き続き行えるからです。 共用キューの高可用性に関する情報は、195 ページの『共用キュー使用の利点』を参照してください。

キュー共用グループにあるメッセージの可用性をさらに拡張するために、IBM MQ は、グループ内の他のキュー・マネージャーがカップリング・ファシリティーから異常に切断されていないかどうかを検出し、可能であれば、そのキュー・マネージャーの保留中の作業単位を完了します。 これはピア回復 と呼ばれていて、271 ページの『ピア回復』で説明されています。

ピア回復により、 障害時に未確定だった作業単位を回復することはできません。 自動再始動管理プログラム (ARM) を使用して、障害に関係したすべてのシステム (例えば、CICS、Db2、および IBM MQ) を再始動すると、すべてのシステムを同一の新しいプロセッサー上で確実に再始動できます。 したがって、これらのシステムを再同期でき、 未確定の作業単位の回復を速く行えます。 これについては、286 ページの『z/OS 自動再始動管理プログラム (ARM) の使用』で説明されています。

共用チャネル

キュー共用グループ環境の場合、IBM MQ にはネットワークの高可用性を実現する機能が備えられています。 チャネル・イニシエーターを使用すると、ネットワーキング製品を使用して、適格なサーバーの集合の間でネットワーク要求の平衡を取り、サーバーの障害をネットワーク (VTAM 汎用資源など) から隠すこ

とができます。 IBM MQ では汎用ポートを使用してインバウンド要求が行われるので、接続要求をキュー共用グループ中の使用可能なチャネル・イニシエーターに経路指定できます。 これについては、216 ページの『共用チャネル』で説明されています。

共用アウトバウンド・チャネルでは、 共用伝送キューから送信されたメッセージが取得されます。 共用チャネルの状況に関する情報は、 キュー共用グループ全体のレベルで 1 つの場所に保留されます。 したがって、チャネル・イニシエーター、キュー・マネージャー、 または通信サブシステムに障害が起こった場合に、 キュー共用グループ中の別のチャネル・イニシエーターでチャネルを自動的に再始動できます。 これはピア・チャネル回復 と呼ばれ、共有アウトバウンド・チャネルで説明されています。

IBM MQ ネットワークの可用性

IBM MQ メッセージは、IBM MQ ネットワーク内のキュー・マネージャー間で、チャネルを使用して運ばれます。 キュー・マネージャーのネットワーク可用性、およびネットワークの問題を検出して再接続する IBM MQ チャネルの能力を向上させるために、ユーザーはさまざまなレベルで構成を変更することができます。

TCP/IP チャネルでは TCP *Keepalive* が使用可能です。 これによって、TCP は定期的にセッション間でパケットを送信し、ネットワーク障害を検出できるようになります。 KAINT チャネル属性が、チャネル用にこれらのパケットの頻度を決定します。

AdoptMCA は、 ネットワーク障害のために受信処理できないチャネルを強制終了し、新規の接続要求に置き換えられるようにします。 AdoptMCA は、MQSC ユーティリティーを持つ ADOPTMCA キュー・マネージャーのプロパティー、または Programmable Command Format インターフェースを持つ AdoptNewMCAType プロパティーを使用して制御します。

ReceiveTimeout は、 チャネルが永続的にネットワーク受信呼び出しできなくなることを回避します。 RCVTIME および RCVTMIN チャネル・イニシエーター・パラメーターが、そのハートビート・インターバル機能として、チャネル用の受信タイムアウト特性を決定します。 詳しくは、キュー・マネージャー・パラメーターを参照してください。

z/OS 自動再始動管理プログラム (ARM) の使用

IBM MQ for z/OS は、z/OS 自動再始動管理プログラム (ARM) と組み合わせて使用できます。 キュー・マネージャーまたはチャネル・イニシエーターに障害が起きた場合、ARM は同じ z/OS イメージ上でそれを再始動します。 z/OS に障害が起きると、関連したサブシステムとアプリケーションのグループ全体も実行できなくなります。 ARM では、同じシスプレックス内の別の z/OS イメージ上で、障害が起きたすべてのシステムを定義済みの順序で自動的に再始動できます。 これを、システム間再始動と呼びます。

ARM を使用すると、共用キュー環境で未確定トランザクションの回復を速く行えます。 キュー共用グループを使用していない場合も、高可用性を実現できます。

z/OS の障害時には、ARM を使用して、シスプレックス内の別の z/OS イメージでキュー・マネージャーを再始動することができます。

自動再始動を使用可能にするには、以下の処理を行わなければなりません。

1. ARM 結合データ・セットをセットアップする。
2. z/OS で実行する自動再始動アクションを *ARM* ポリシー に定義します。
3. ARM ポリシーを開始する。

別の z/OS イメージにあるキュー・マネージャーを自動的に再始動するには、キュー・マネージャーの再始動を行う可能性がある個々の z/OS イメージの中で、該当のシスプレックス全体にわたって固有な 4 文字のサブシステム名を使って、個々のキュー・マネージャーをすべて定義しておく必要があります。

IBM MQ での ARM の使用については、IBM MQ ネットワークでの ARM の使用で説明されています。

z/OS 拡張回復機能 (XRF) の使用

IBM MQ は、拡張回復機能 (XRF) 環境で使用できます。 すべての IBM MQ 所有のデータ・セット (実行可能コード、BSDS、ログ、およびページ・セット) は、活動プロセッサーと代替 XRF プロセッサーとの間で共用される DASD 上にある必要があります。

回復のために XRF を使用する場合は、活動プロセッサー上のキュー・マネージャーを停止し、それを代替プロセッサーで開始する必要があります。 CICS の場合、CICS により提供されるコマンド・リスト表 (CLT) を使用して行うか、またはシステム・オペレーターが手操作で行うことができます。 IMS の場合、XRF は手操作であり、調整側 IMS システムがプロセッサー切り替えを完了したあとに行わなければなりません。

キュー・マネージャーが代替プロセッサーに切り替える前に、IBM MQ ユーティリティーを完了または終了しなければなりません。 これは中断が起こる可能性を意味するので、XRF 回復計画を立てる際にはその点を十分に考慮してください。

活動プロセッサーのキュー・マネージャーが終了する前に、 代替プロセッサーでキュー・マネージャーが開始されないように注意してください。 開始される時期が早すぎると、データ、カタログ、 およびログにおける整合性に関して重大な問題が発生することがあります。 グローバル資源の逐次化 (GRS) を使用すれば、2 つのシステムでの IBM MQ の同時使用を防ぐことができるため、整合性問題の回避に役立ちます。 BSDS は保護資源として組み込まなければならず、また活動および代替 XRF プロセッサーは GRS リングの中に含めなければなりません。

キュー共用グループの回復に対する z/OS GROUPUR 属性の使用

キュー共用グループ (QSG) では、このトピックで説明される追加のトランザクション機能を使用できます。 GROUPUR 属性を使用すると、必要になる可能性がある未確定のトランザクションの回復を、XA クライアント・アプリケーションが QSG の任意のメンバー上で実行できるようになります。

XA クライアント・アプリケーションがシスプレックスによってキュー共用グループ (QSG) に接続する場合、接続先のキュー・マネージャーを指定することができません。 キュー共用グループ内のキュー・マネージャーが GROUPUR 属性を使用することで、必要になる可能性がある未確定のトランザクションの回復を、QSG の任意のメンバー上で実行できるようになります。 アプリケーションが最初に接続したキュー・マネージャーが使用できないとしても、トランザクションの回復を行うことができます。

この機能では、XA クライアント・アプリケーションが特定の QSG のメンバーに依存しないので、キュー・マネージャーの可用性が拡大します。 トランザクション・アプリケーションはキュー共用グループを、すべての IBM MQ 機能を提供し、キュー・マネージャーの Single Point of Failure がない 1 つのエンティティーと見なします。

トランザクション・アプリケーション側からは、この機能は分かりません。

可用性に関する詳細が記載されている資料

これらのトピックに関する詳細情報は、以下の資料から入手できます。

表 *24.* 可用性に関する詳細が記載されている資料

トピック	**参照先**
キュー共有グループ	175 ページの『共用キューとキュー共用グループ』
システム・パラメーター	システム・パラメーターの構成
自動再始動管理プログラムの使用 ユーティリティー・プログラム	IBM MQ ネットワークでの ARM の使用
MQSC コマンド	MQSC コマンド

z/OS IBM MQ for z/OS でのモニターと統計

IBM MQ for z/OS には、キュー・マネージャーをモニターして、統計を収集するための一連の機能があります。

IBM MQ には、システムのモニターを行い統計を収集する機能が備えられています。 これらの機能の詳細については、以下のセクションを参照してください。

- 288 ページの『オンライン・モニター』
- 288 ページの『IBM MQ トレース』
- 288 ページの『イベント』

オンライン・モニター

IBM MQ には、IBM MQ オブジェクトの状況をモニターするために、以下のコマンドが用意されています。

- DISPLAY CHSTATUS は指定されたチャネルの状況を表示する。
- DISPLAY QSTATUS は指定されたキューの状況を表示する。
- DISPLAY CONN は指定された接続の状況を表示する。

これらのコマンドの詳細については、MQSC コマンドを参照してください。

IBM MQ トレース

IBM MQ にはトレース機能が備えられており、キュー・マネージャーの実行時にこれを使用して以下の情報を集めることができます。

パフォーマンス統計
統計トレースにより、以下の情報を集めて、 パフォーマンスのモニターに利用し、システムを調整することができます。

- さまざまな MQI 要求のカウント (メッセージ・マネージャー統計)
- さまざまなオブジェクト要求のカウント (データ・マネージャー統計)
- Db2 の使用状況に関する情報 (Db2 マネージャー統計)
- カップリング・ファシリティーの使用状況に関する情報 (カップリング・ファシリティー・マネージャー統計)
- SMDS の使用に関する情報 (共有メッセージ・データ・セット統計量)
- バッファー・プールの使用状況に関する情報 (バッファー・マネージャー統計)
- ロギングに関する情報 (ログ・マネージャー統計)
- ストレージの使用状況に関する情報 (ストレージ・マネージャー統計)
- ロック要求に関する情報 (ロック・マネージャー統計)

アカウンティング・データ

- アカウンティング・トレースにより、MQI 呼び出しの処理に要したプロセッサー時間に関する情報と、 特定のユーザーが行った MQPUT 要求と MQGET 要求の数に関する情報を集めることができます。
- IBM MQ は、IBM MQ を使用したそれぞれのタスクに関する情報も集めることができます。 このデータは、スレッド・レベルのアカウンティング・レコードとして集められます。 スレッドごとに、IBM MQ は、そのスレッドによって使用されたそれぞれのキューに関する情報も集めます。

トレースで生成されたデータは、システム管理機能 (SMF) または汎用トレース機能 (GTF) に送信されます。

イベント

IBM MQ イベントにより、エラー、警告、およびキュー・マネージャー中で起きた他の重要なイベントに関する情報を得られます。 これらのイベントを独自のシステム管理アプリケーションに組み込むと、複数の IBM MQ アプリケーションの、多数のキュー・マネージャーにわたる活動をモニターできます。 特に、1 つのキュー・マネージャーから、 システム中のすべてのキュー・マネージャーをモニターできます。

ユーザー作成の報告機構を使用して、 イベントをオペレーターに表示する機能をサポートしている管理アプリケーションに、 イベントを報告できます。 さらに、イベントを使用すると、アプリケーションは他の管理ネットワーク (例えば NetView®) のエージェントとして動作して、報告をモニターし、適切なアラートを作成できます。

関連タスク

IBM MQ トレースの使用

IBM MQ イベントの使用

z/OS での回復単位後処理

トランザクション・アプリケーションの中には、キュー共用グループ (QSG) 内のキュー・マネージャーに接続されているときに、QMGR ではなく、GROUP 回復単位後処理を使用できるものがあります。その場合、接続時にキュー・マネージャー名の代わりに QSG 名を指定します。 これは QSG 内の同じキュー・マネージャーに再接続する必要がないので、トランザクションの回復をより柔軟かつ堅固に行えます。

キュー共有グループ名を使って接続されたアプリケーションによって開始されるトランザクションにも GROUP 回復単位後処理が付帯します。

トランザクション・アプリケーションが GROUP 回復単位後処理を使って接続すると、必然的にキュー共用グループにも接続されます。特定のキュー・マネージャーとのアフィニティーはありません。 2 フェーズ・コミットのトランザクションが開始し、コミット・プロセスのフェーズ 1 が完了した場合 (つまり未確定の状態)、そのトランザクションは、QSG 内のどのキュー・マネージャーに接続されたときでも、照会され、解決することができます。 回復のシナリオにおいて、これはトランザクション・コーディネーターが同じキュー・マネージャーに再接続する必要がないことを意味します。そのキュー・マネージャーはその時点で使用不可になっている可能性があります。

QMGR 回復単位後処理を使って接続するアプリケーションには、接続先のキュー・マネージャーとの直接的なアフィニティーが付与されます。 回復のシナリオにおいてトランザクション・コーディネーターは、キュー・マネージャーがキュー共用グループに属しているかどうかに関係なく、未確定のトランザクションを解決するために同じキュー・マネージャーに再接続する必要があります。

アプリケーションがキュー共用グループ名を指定し、GROUP 回復単位後処理を使って QSG 内のキュー・マネージャーに接続すると、キュー共有グループは必然的に別個のリソース・マネージャーになります。これは、アプリケーションが同じ回復単位後処理を使って再接続する場合のみ未確定のトランザクションがそのアプリケーションに対して可視になることを意味します。 GROUP 回復単位後処理を使って接続されたアプリケーションは、QMGR 回復単位後処理の未確定のトランザクションを認識できません。その逆についても同じことが言えます。

関連概念

289 ページの『GROUP 回復単位の使用可能化』
キュー共用グループは、GROUP 回復単位のサポートを構成し、使用可能にすることができます。

290 ページの『アプリケーション・サポート』
このページは、GROUP 回復単位後処理を使って接続できるアプリケーションを判別するために使用します。

z/OS GROUP 回復単位の使用可能化

キュー共用グループは、GROUP 回復単位のサポートを構成し、使用可能にすることができます。

QSG 内のキュー・マネージャーで GROUP 回復単位を使用可能にするには、GROUPUR キュー・マネージャー属性を使用可能にします。 この概念について詳しくは、このトピックの続く部分を読む前に 289 ページの『z/OS での回復単位後処理』を参照してください。

GROUPUR キュー・マネージャー属性を使用可能にすると、キュー・マネージャーは GROUP 回復単位後処理を使用する新しい接続を受け入れます。 この属性を使用不可にすると、この後処理を使用する新しい接

続は受け入れられません。ただし、既に接続されているアプリケーションは、切断されるまで影響を受けません。

アプリケーションが GROUP 回復単位後処理を使って接続し、未確定のトランザクションを照会するか、キュー共用グループ (QSG) 内の他の場所で開始されたトランザクションを解決しようとする際、接続先のキュー・マネージャーは、要求を処理するためにキュー共有グループの他のメンバーとの通信が行えなければなりません。 これを行うために、SYSTEM.QSG.UR.RESOLUTION.QUEUE という共用キューが使用されます。 このキューは、CSQSYSAPPL という回復可能なアプリケーション構造体上に存在している必要があります。 この構造体が回復可能でなければならないのは、解決要求を処理する際に持続メッセージがこのキューに保管されるからです。

GROUP 回復単位を使用可能にするにはまず、カップリング・ファシリティー構造体と共用キューを定義しておく必要があります。 CSQ4INSS サンプルの定義を使用できます。 キューが定義されるか、始動時に検出されると、キュー共用グループ内の各キュー・マネージャーは、着信要求を受け取れるようにキューを開きます。 キューが正しく定義されなかったためにキューを削除または移動する場合は、キュー・オブジェクトを更新して MQGET 要求を禁止することによって、そのキューに関してキュー・マネージャーがオープン・ハンドルを閉じることを要求できます。 必要な修正を加えた後、もう一度アプリケーションがキューからメッセージを受け取れるように許可すると、各キュー・マネージャーに対し、キューを再び開くように指示が出されます。 どのハンドルがキューで開いているか識別するには、DISPLAY QSTATUS コマンドを使用します。

このセットアップが完了した後、GROUP 回復単位を使用可能にする設定を、GROUP 回復単位後処理を使ったトランザクション・アプリケーションからの接続を受け入れるキュー・マネージャーごとに行うことができます。 キュー共用グループ内のすべてのキュー・マネージャーでこれを行う必要はありませんが、キュー共有グループのサブセットでのみこの機能を使用可能にする場合は、この機能を使用可能にしたキュー・マネージャーだけにアプリケーションが接続を試みるようにする必要があります。 詳細については、290 ページの『アプリケーション・サポート』を参照してください。

GROUPUR キュー・マネージャー属性を使用可能に設定しようとすると、いくつかの構成検査が実行されます。 キュー・マネージャーは以下を検査します。

- キュー共有グループに属すること。
- CSQ4INSS の定義に従って、SYSTEM.QSG.UR.RESOLUTION.QUEUE という共有キューが定義されていること。
- SYSTEM.QSG.UR.RESOLUTION.QUEUE が CSQSYSAPPL という回復可能 CF 構造体上に存在すること。

以上の検査のいずれかに合格しないと、GROUPUR 属性は使用不可のままになり、メッセージ・コードが返されます。

キュー・マネージャー属性が使用可能になっている場合、キュー・マネージャーの開始時にもこれらの構成検査が実行されます。 開始時にいずれかの検査が失敗すると、GROUP 回復単位は使用不可にされ、キュー・マネージャーは失敗した検査を示すメッセージを発行します。 必要な修正アクションを実行した後、キュー・マネージャー属性を再び使用可能にする必要があります。

z/OS アプリケーション・サポート

このページは、GROUP 回復単位後処理を使って接続できるアプリケーションを判別するために使用します。

GROUP 回復単位後処理のサポートは、IBM MQ for z/OS がそのトランザクション・コーディネーターではなくリソース・マネージャーとなる、特定のタイプのトランザクション・アプリケーションに限られます。現在サポートされているトランザクション・アプリケーションとは、

- IBM MQ 拡張トランザクション・クライアント・アプリケーション
- WebSphere Application Server などのアプリケーション・サーバーで実行される IBM MQ classes for JMS アプリケーション。
- CICS またはそれ以降で CICS Transaction Server 4.2 アプリケーションが実行中で、CICS MQCONN のリソースの定義が RESYNCMEMBER(GROUPRESYNC) で構成されている場合。

関連概念

291 ページの『IBM MQ 拡張トランザクション・クライアント・アプリケーション』

このページは、IBM MQ 拡張トランザクション・クライアント・アプリケーションが GROUP 回復単位後処理を使用できる方法を判別するために使用します。

291 ページの『CICS アプリケーション』
このページは、CICS による GROUP 回復単位後処理の使用法を判別するために使用します。

z/OS *IBM MQ* 拡張トランザクション・クライアント・アプリケーション

このページは、IBM MQ 拡張トランザクション・クライアント・アプリケーションが GROUP 回復単位後処理を使用できる方法を判別するために使用します。

IBM MQ 拡張トランザクション・クライアント・アプリケーションの一例として、JMS を使用して WebSphere Application Server で実行されるアプリケーションがあります。ローカル・バインディングではなく TCP/IP を介して IBM MQ に接続します。 この種のクライアント・アプリケーションは、TCP/IP などのネットワーク接続を介して IBM MQ for z/OS に接続します。 この種のアプリケーションでは、xa_open 呼び出しに渡される xa_info ストリングの QMNAME パラメーターに指定される値により、QMGR と GROUP のどちらの回復単位後処理を使用するかが指定されます。 xa_open について詳しくは、xa_open ストリングの形式および xa_open の追加のエラー処理を参照してください。 JMS アプリケーションの場合、これは特定のキュー・マネージャーの名前の代わりにキュー共用グループ (QSG) の名前を ConnectionFactory に指定することによって行われます。

XA クライアント・アプリケーションで GROUP 回復単位後処理を使用するには、特定のキュー・マネージャーではなく、キュー共有グループ内の GROUPUR 属性が使用可能にされたキュー・マネージャーにクライアント・アプリケーションが経路指定されるように TCP/IP の設定を構成する必要があります。 これを行うために使用できる動的仮想 IP アドレステクノロジーの 1 つが、z/OS SysPlex ディストリビューターです。 詳しくは、z/OS Communications Server および z/OS 基本スキル: 動的仮想アドレッシングを参照してください。 キュー共用グループのキュー・マネージャーのサブセットで GROUP 回復単位を使用可能にする場合は、それが使用可能にされないキュー・マネージャーにクライアント・アプリケーションを経路指定できないようにする必要があります。

クライアント・アプリケーションは、共用チャネルを使ってキュー共用グループに接続する必要はありません。

z/OS *CICS* アプリケーション

このページは、CICS による GROUP 回復単位後処理の使用法を判別するために使用します。

CICS 4.2 以降では、MQCONN リソース定義でグループ再同期オプション RESYNCMEMBER (GROUPRESYNC) が提供されます。 このオプションを使用して構成された CICS は、CICS 領域と同じ LPAR で実行されるキュー共用グループ内の適切なキュー・マネージャーに接続できます。 CICS GROUPRESYNC オプションをサポートするには、キュー・マネージャーが MQ V7.1 以降で実行されており、GROUPUR サポートで有効である必要があります。

GROUPRESYNC を使用して MQ に接続された CICS 領域内でトランザクションを実行すると、GROUP 回復単位後処理を使用して作業単位が作成されます。

RESYNCMEMBER (GROUPRESYNC) を使用すると、キュー・マネージャーの再始動を待たずに、必要に応じて未確定トランザクションを解決して、同じ LPAR 上で実行されている代替適格キュー・マネージャーに CICS 領域が即時に接続できるため、キュー・マネージャーの障害後のリカバリーを高速化することができます。

RESYNCMEMBER (GROUPRESYNC) により、 CICS の再始動オプションの柔軟性も向上します。 GROUPRESYNC および MQ 共用キューを使用するように MQ 接続が構成された CICS 領域は、同じキュー共用グループのメンバーとして実行されているキュー・マネージャーがあるいずれの LPAR でも再始動することができます。

z/OS IBM MQ およびその他の z/OS 製品

このトピックでは、IBM MQ が他の z/OS 製品と連携する仕組みについて理解することができます。

関連概念

292 ページの『IBM MQ と CICS』
IBM MQ 9.0.0 以降でサポートされるすべての CICS バージョンは、CICS 提供バージョンのアダプターおよびブリッジを使用します。

298 ページの『IBM MQ for z/OS と WebSphere Application Server』
このトピックでは、 WebSphere Application Server による IBM MQ for z/OS の使用について説明します。

関連資料

293 ページの『IBM MQ と IMS』
このトピックでは、どのように IBM MQ が IMS と共に作動するかについて知ることができます。 IMS アダプターを使用すると、キュー・マネージャーを IMS に接続できるようになり、IMS アプリケーションが MQI を使用できるようになります。

297 ページの『IBM MQ および z/OS バッチ・アダプター、TSO アダプター、および RRS アダプター』
このトピックでは、IBM MQ が z/OS バッチ、TSO、および RRS アダプターを処理する方法について理解することができます。

z/OS IBM MQ と CICS

IBM MQ 9.0.0 以降でサポートされるすべての CICS バージョンは、CICS 提供バージョンのアダプターおよびブリッジを使用します。

IBM MQ CICS アダプターおよび IBM MQ CICS bridge コンポーネントの構成について詳しくは、CICS 資料の「 IBM MQ への接続の構成 」セクションを参照してください。

関連タスク

IBM MQ と CICS の使用

z/OS CICS グループ接続

CICS グループ接続は、CICS 領域が、同じ LPAR 上の IBM MQ キュー共用グループのアクティブ・メンバーに接続できるようにします。この場合、個々のキュー・マネージャーを指定する必要はありません。 この場合も、CICS は、一度に 1 つのキュー・マネージャーに接続します。

CICS グループ接続をサポートするには、LPAR 上に少なくとも 2 つのキュー・マネージャーが必要です。グループ接続を使用する場合、特定のキュー・マネージャーをアクティブにする必要がないため、より高い可用性が得られます。 CICS は、LPAR 上のキュー共用グループ内のどのキュー・マネージャーにも接続できます。

詳しくは、MQCONN リソースの CICS 資料を参照してください。

CICS は、キュー・マネージャーであるかのように渡される MQNAME への接続を試みます。

- キュー・マネージャーが存在し、アクティブの場合、接続が機能します。
- 接続が失敗する場合、CICS は、同じ LPAR でアクティブになっているキュー・マネージャーを確認するために、グループ内のキュー・マネージャーの状況を照会します。
- 複数のキュー・マネージャーがアクティブである場合、 CICS は RESYNCMEMBER (YES) および UOW 状況を検査して、 CICS が特定のメンバーに接続する必要があるか、または接続する必要があるかを判別します。アクティブでない場合は待機します。
- 特定のメンバーに接続する必要がない場合、CICS は任意のキュー・マネージャーを選択します (その際、ランダム化アルゴリズムを使用します)。
- CICS は、選択されたキュー・マネージャーへの接続を試みます。
- 試行が失敗する場合、戻りコードに応じて、CICS は次のメンバーを選択します。その後、再び選択ループに入ります。
- キュー・マネージャーがアクティブでない場合、CICS は、キュー・マネージャーのリストへの複数の接続を発行し、最初のキュー・マネージャーが使用可能になるまで ECBLIST で待ちます。

関連概念

293 ページの『CICS のグループ回復単位 (GROUPUR)』
CICS の IBM MQ GROUPUR は、キュー共用グループ (QSG) 内の未確定作業単位のピア・リカバリーを提供します。 キュー共用グループ内の IBM MQ キュー・マネージャーは、キュー共用グループ内の別のキュー・マネージャーの代わりに未確定の作業単位を解決できます。 このことは、CICS がグループ接続を介して QSG 内の別のキュー・マネージャーに再接続した場合に、以前の IBM MQ 接続からの未確定トランザクションを解決できることを意味します。

関連情報
IBM MQ キュー共用グループのサポート

z/OS CICS のグループ回復単位 (GROUPUR)

CICS の IBM MQ GROUPUR は、キュー共用グループ (QSG) 内の未確定作業単位のピア・リカバリーを提供します。 キュー共用グループ内の IBM MQ キュー・マネージャーは、キュー共用グループ内の別のキュー・マネージャーの代わりに未確定の作業単位を解決できます。 このことは、CICS がグループ接続を介して QSG 内の別のキュー・マネージャーに再接続した場合に、以前の IBM MQ 接続からの未確定トランザクションを解決できることを意味します。

CICS 領域がキュー・マネージャーと一緒に作業しているときにキュー・マネージャーが異常終了する場合、未確定トランザクションはすべて回復されます。 これにより、CICS 領域が、一緒に作業しているキュー・マネージャーを待って、再始動し、未確定の作業を解決する必要がなくなります。 これは、LPAR に少なくとも 2 つのキュー・マネージャーが必要であることを意味します。キュー・マネージャーの異常終了が発生した場合に最初の CICS が別のキュー・マネージャーに接続できるようにするためです。

CICS MQCONN 定義での新しい RESYNCMEMBER (GROUPRESYNC) 設定:

- IBM MQ グループ接続機能およびピア回復を使用します。
- GROUPUR 属性を有効にしたキュー・マネージャーが必要です。
- 既存の CICS MQCONN RESYNCMEMBER 設定を引き続きサポートします (YES および NO)。
  - 既存の CICS グループ接続機能を使用し、ピア回復は使用しません。
  - RESYNCMEMBER 設定の変更は、次回 CICS が IBM MQ に接続するときに有効になります。

関連概念
289 ページの『GROUP 回復単位の使用可能化』
キュー共用グループは、GROUP 回復単位のサポートを構成し、使用可能にすることができます。

z/OS IBM MQ と IMS

このトピックでは、どのように IBM MQ が IMS と共に作動するかについて知ることができます。 IMS アダプターを使用すると、キュー・マネージャーを IMS に接続できるようになり、IMS アプリケーションが MQI を使用できるようになります。

オプションの追加の IBM MQ - IMS ブリッジを使用すると、MQI を使用しない IMS アプリケーションをアプリケーションで実行できます。 したがって、レガシー・アプリケーションを作成し直さなくても IBM MQ と一緒に使用できます。

これらのコンポーネントの詳細については、以下のサブトピックを参照してください。

関連概念
IBM MQ for z/OS 上の IMS および IMS ブリッジ・アプリケーション
関連タスク
IMS アダプターのセットアップ
IMS ブリッジのセットアップ
IMS アダプターの操作
関連資料
MQIIH - IMS 情報ヘッダー

z/OS IMS アダプター

IMS アダプターは、IMS アプリケーション・プログラムと IBM MQ サブシステムの間のインターフェースです。

IBM MQ アダプターを使用すると、さまざまなアプリケーション環境で、メッセージ・キューイング・ネットワークを介してメッセージの送受信を行えます。 IMS アダプターは、IMS アプリケーション・プログラムと IBM MQ サブシステムの間のインターフェースです。 このアダプターを使用すると、IMS アプリケーション・プログラムで MQI を使用できます。

IMS アダプターは、 IMS が提供する 外部サブシステム接続機能 (ESAF) を使用して、 IBM MQ へのアクセス要求を受け取って解釈します。 普通は、オペレーターが介入しなくても IMS は IBM MQ に自動的に接続されます。

IMS アダプターは、以下のモードまたは状態で実行されているプログラムに、IBM MQ 資源へのアクセス権を付与します。

- タスク (TCB) モード
- 問題プログラム状態
- 非仮想記憶間モード
- 非アクセス登録モード

このアダプターには、アプリケーション・タスク制御ブロック (TCB) から IBM MQ への接続スレッドが備えられています。

アダプターは、同期点コーディネーターとして機能する IMS を使用して IBM MQ が所有するリソースに対して行われた変更に対して、2 フェーズ・コミット・プロトコルをサポートします。 IMS が同期点コーディネーターでない会話 (例えば、APPC で保護されている (SYNCLVL=SYNCPT) 会話) は、IMS アダプターではサポートされません。

このアダプターには、トリガー・モニター・トランザクション (CSQQTRMN) も備えられています。 これについては、294 ページの『IMS トリガー・モニター』 で説明されています。

IBM MQ と IMS 拡張回復機能 (XRF) を一緒に使用すると、IMS エラーからの回復に役立ちます。

注 : IMS 15.2 以降、拡張回復機能 (XRF) はサポートされなくなりました。 詳しくは、 IMS の資料を参照してください。

アダプターの使用

アプリケーション・プログラムと IMS アダプターは同じアドレス・スペースで実行されます。 キュー・マネージャーは別個に独自のアドレス・スペースで実行されます。

1 つまたは複数の MQI 呼び出しを発行する個々のプログラムを、適切な IMS 言語インターフェース・モジュールにリンク・エディットさせなければなりません。また、動的 MQI 呼び出しを使用していない場合は、IBM MQ に備えられている API スタブ・プログラム CSQQSTUB にもリンク・エディットさせなければなりません。 アプリケーションで MQI 呼び出しが発行される際に、スタブから IMS 外部サブシステム・インターフェースを介してアダプターに制御権が移動されます。このインターフェースによりメッセージ・キュー・マネージャーからの要求の処理が管理されます。

IMS によるシステムの管理と操作

許可された IMS 端末オペレーターは、 IMS コマンドを発行して、 IBM MQ への接続を制御およびモニターすることができます。 しかしながら、IMS 端末オペレーターには IBM MQ アドレス・スペース上の制御権はありません。 例えば、オペレーターは IMS アドレス・スペースから IBM MQ をシャットダウンすることはできません。

制約事項

以下の IBM MQ API 呼び出しは、IMS アダプターを使用するアプリケーションではサポートされていません。

- MQCB
- MQCB_FUNCTION
- MQCTL

IMS トリガー・モニター

IMS トリガー・モニター (**CSQQTRMN**) IBM MQ イベントが発生したとき (例えば、メッセージが特定のキューに書き込まれたとき) に IMS トランザクションを開始する IBM MQ 提供の IMS アプリケーションです。

作業の方法

メッセージがアプリケーション・メッセージ・キューに挿入された時点で、 トリガー条件が満たされると、トリガーが生成されます。 次いでキュー・マネージャーにより、 トリガー・メッセージ と呼ばれるメッセージ (ユーザー定義データを含む) が、 このメッセージ・キュー用に指定された開始キューに書き込まれます。 IMS 環境では、CSQQTRMN のインスタンスを開始して、開始キューをモニターしてトリガー・メッセージが挿入され次第取り出すことができます。 通常 CSQQTRMN は、INSERT (ISRT) により IMS メッセージ・キューに別の IMS トランザクションをスケジュールします。 開始された IMS アプリケーションにより、アプリケーション・メッセージ・キューからメッセージが読み取られて処理されます。 CSQQTRMN は非メッセージ BMP として稼働しなければなりません。

個々の CSQQTRMN のコピーは、1 つの開始キューにサービスを提供します。 トリガー・モニターを開始すると、IBM MQ または IMS が終了するまで実行されます。

CSQQTRMN の APPLCTN マクロで SCHDTYP=PARALLEL を指定しなければなりません。

トリガー・モニターはバッチ向きの BMP なので、トリガー・モニターにより開始される IMS トランザクションには以下のものが含まれます。

- IOPCB の LTERM フィールドがブランク
- IOPCB の Userid フィールドがトリガー・モニター BMP の PSB 名

ターゲットの IMS トランザクションが Security Server (以前の RACF) によって保護されている場合、CSQQTRMN を Security Server に対するユーザー ID として定義する必要が生じることがあります。

z/OS IBM MQ-IMS ブリッジ

IBM MQ - IMS ブリッジは、IBM MQ for z/OS のアプリケーションから IBM MQ システム上のアプリケーションへの直接アクセスを可能にする IMS のコンポーネントです。

IBM MQ-IMS ブリッジは、暗黙的な *MQI* サポート を使用可能にします。 つまり、3270 接続の端末で制御されている既存のアプリケーションの書き換え、再コンパイル、または再リンクを一切行わずに、IBM MQ メッセージで制御されるように修正することができます。 このブリッジは、IMS オープン・トランザクション管理アクセス (OTMA) クライアントです。

ブリッジ・アプリケーションでは、 IMS アプリケーション内に IBM MQ 呼び出しはありません。 このアプリケーションは、IOPCB に対して GET UNIQUE (GU) を使用して入力を 取得し、IOPCB に対して ISRT を使用して出力を送信します。 IBM MQ アプリケーションでは、メッセージ・データ中に IMS ヘッダー (MQIIH 構造体) が使用されており、非プログラム式端末主導の場合と同じように実行できるようになっています。 複数セグメントのメッセージを処理する IMS アプリケーションを使用している場合は、すべてのセグメントが 1 つの IBM MQ メッセージに含まれている必要があるため注意してください。

296 ページの図 78 に IMS ブリッジが示されています。

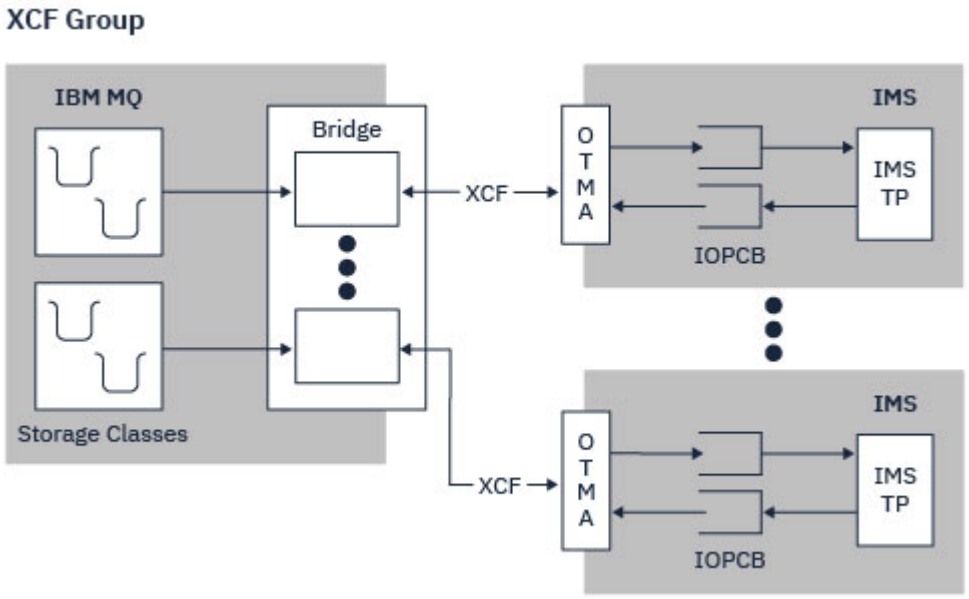


図 *78. IBM MQ-IMS* ブリッジ

1 つのキュー・マネージャーが 1 つまたは複数の IMS システムに接続でき、複数のキュー・マネージャーが 1 つの IMS システムに接続できます。 制約事項が 1 つだけありますが、 それはすべて同じ XCF グループに属していなければならず、 同じシスプレックスになければならないというものです。

IMS ブリッジのセットアップと、同じキュー・マネージャーへの IMS 接続の追加については、 IMS ブリッジのセットアップ を参照してください。

OTMA について

IMS OTMA 機能は、 IMS 上で実行されるトランザクション・ベースのコネクションレス・クライアント/サーバー・プロトコルです。 これは、z/OS システム間カップリング・ファシリティー (XCF)を介して IMS TM アプリケーションにアクセスするホスト・ベースの通信サーバーのインターフェースとして機能します。

大規模ネットワークの場合や多数のセッションがある場合、OTMA を使用すると、クライアントは IMS に接続して、IMS との間で効率の良い対話を行うことができます。 OTMA は、z/OS シスプレックス環境でインプリメントされます。 したがって、OTMA のドメインは、XCF のドメインに限定されます。

OTMA リソース・モニター

x '3C' OTMA プロトコル・メッセージ (IMS v10 以上で使用可能) のサポートは、 IBM MQ for z/OS の IBM MQ - IMS ブリッジに含まれています。 これらのメッセージは、ヘルス状況をレポートするために、IMS によって OTMA クライアントに送信されます。

IMS パートナーが、送信されるトランザクション要求のボリュームを処理できない場合は、フラッディング警告が発生したと IBM MQ に通知します。 それに応えて IBM MQ は、ブリッジ経由で要求を送信する速度を低下させます。

IMS がまだトランザクション要求を処理できずに完全なフラッディング状態になった場合は、IMS パートナーへのすべての TPIPE が中断されます。 IMS パートナーから、フラッディングまたはフラッディング警告状態が緩和されたことが通知されると、IBM MQ は該当する場合には中断されたすべての TPIPE を再開し、トランザクション要求の送信速度を、最大速度に達するまで徐々に高めます。 コンソール・メッセージは、 IMS パートナーの状況の変化に応じて IBM MQ によって発行されます。

IMS v10 パートナーが使用されている場合、PTF UK45082 が適用されていることを確認する必要があります。

IBM MQ からの IMS トランザクションのサブミット

ブリッジを使用する IMS トランザクションを実行依頼するために、通常どおりアプリケーションでメッセージが IBM MQ キューに挿入されます。 このメッセージには IMS トランザクション・データが含まれています。このデータに IMS ヘッダー (MQIIH 構造体) を含めたり、このデータを使用して IBM MQ-IMS ブリッジでメッセージ中のデータに関する前提事項を作成したりできます。

次に、IBM MQ は、IMS キューにメッセージを書き込みます (最初に、IBM MQ でキューに入れられているため、同期点を使用してデータ保全性を確保することができます)。 IBM MQ キューのストレージ・クラスにより、そのキューが *OTMA* キュー (つまり IBM MQ-IMS ブリッジへのメッセージ伝送に使用されるキュー) であるかどうかと、メッセージ・データの送信先の特定の IMS パートナーが判別されます。

リモート・キュー・マネージャーは、 IBM MQ for z/OS 上のこれらの OTMA キューに書き込むことによって、 IMS トランザクションを開始することもできます。

IMS システムから戻されるデータは、メッセージ・ディスクリプター構造体 (MQMD) に指定されている IBM MQ 応答先キューに直接書き込まれます。 (このキューは、MQMD の **ReplyToQMgr** フィールドで指定されているキュー・マネージャーに対する伝送キューの場合があります。)

関連概念

IBM MQ for z/OS 上の IMS および IMS ブリッジ・アプリケーション

関連タスク

IMS ブリッジのカスタマイズ

関連資料

293 ページの『IBM MQ と IMS』

このトピックでは、どのように IBM MQ が IMS と共に作動するかについて知ることができます。 IMS アダプターを使用すると、キュー・マネージャーを IMS に接続できるようになり、IMS アプリケーションが MQI を使用できるようになります。

z/OS IBM MQ および z/OS バッチ・アダプター、TSO アダプター、および RRS アダプター

このトピックでは、IBM MQ が z/OS バッチ、TSO、および RRS アダプターを処理する方法について理解することができます。

バッチ・アダプターの紹介

バッチ /TSO アダプターは、 IBM MQ と、JES、TSO、または z/OS UNIX System Services の下で実行される z/OS アプリケーション・プログラムとの間のインターフェースです。 このアダプターを使用すると、z/OS アプリケーション・プログラムが MQI を使用できるようになります。

アダプターは、次のモードまたは状態で実行されているプログラムに、IBM MQ 資源へのアクセス権を付与します。

- タスク (TCB) モード
- 障害状態または監視プログラム状態
- 非仮想記憶間モード
- 非アクセス登録モード

アプリケーション・プログラムと IBM MQ との接続は、タスク・レベルで行われます。 アダプターは、アプリケーションのタスク制御ブロック (TCB) から IBM MQ への接続スレッドを提供します。

バッチ/TSO アダプターは、IBM MQ の所有する資源に対して変更が行われたときのための、単一フェーズ・コミット・プロトコルをサポートします。 複数フェーズ・コミット・プロトコルはサポートしません。 RRS アダプターにより、IBM MQ アプリケーションは、z/OS リソース・リカバリー・サービス (RRS) により調整される、他の RRS イネーブル製品との 2 フェーズ・コミット・プロトコルに参加することができます。

このアダプターは z/OS STIMERM サービスを使用して、非同期イベントを毎秒、スケジュールします。 このイベントは、バッチ・アプリケーションのタスクの待機とは無関係の、 割り込み要求ブロック (IRB) を

実行します。 この IRB は、IBM MQ 終了 ECB が通知されているかどうかを確認します。 終了 ECB が通知されている場合、IRB は IBM MQ 内のイベント (例えば、信号または待機) で待機しているアプリケーション ECB を通知します。

バッチ/TSO アダプター

IBM MQ バッチ /TSO アダプターは、z/OS バッチおよび TSO アプリケーションの IBM MQ サポートを提供します。 z/OS バッチまたは TSO 下で実行しているすべてのアプリケーション・プログラムは、API スタブ・プログラム CSQBSTUB とリンク・エディットされている必要があります。 スタブは、すべての MQI 呼び出しへのアクセスをアプリケーションに提供します。 アプリケーションに対して、単一フェーズ・コミットおよびバックアウトを 使用するには、MQI 呼び出し **MQCMIT** および **MQBACK** を発行します。

RRS アダプター

リソース・リカバリー・サービス (RRS) は、複数の z/OS 製品をまたがる 2 フェーズ・コミットを調整するためのシステム全体のサービスを提供する z/OS のサブコンポーネントです。 IBM MQ バッチ /TSO RRS アダプター (RRS アダプター) は、これらのサービスを使用する z/OS バッチおよび TSO アプリケーションの IBM MQ サポートを提供します。 RRS アダプターにより、IBM MQ は RRS 調整の完全な参加プログラムとなることができます。 アプリケーションは RRS をサポートする他の製品 (例えば Db2) との 2 フェーズ・コミット処理に参加できます。

RRS アダプターは、次の 2 つのスタブを提供します。RRS を使用すべきアプリケーション・プログラムは、これらの 2 つのうち 1 つのスタブとリンク・エディットされる必要があります。

CSQBRSTB
このスタブを使えば、アプリケーションに対して、2 フェーズ・コミットおよびバックアウトを使用することができます。その場合、MQI 呼び出し **MQCMIT** および **MQBACK** の代わりに、RRS 呼び出し可能リソース・リカバリー・サービスを使用します。

また、ライブラリー SYS1.CSSLIB にあるモジュール ATRSCSS を、 ご使用のアプリケーションにリンク・エディットする必要もあります。 MQI 呼び出し **MQCMIT** および **MQBACK** を使用すると、 リターン・コード MQRC_ENVIRONMENT_ERROR を受け取ります。

CSQBRRSI
このスタブを使えば、MQI 呼び出し **MQCMIT** および **MQBACK** を使用することができます。IBM MQ は実際にはこれらの呼び出しを、**SRRCMIT** および **SRRBACK** RRS 呼び出しとしてインプリメントします。

RRS アダプターを使用するアプリケーション・プログラムを作成する方法については、RRS バッチ・アダプターを参照してください。

z/OS バッチ、TSO、および RRS アダプターの詳細について記載されている資料

このセクションのトピックに関する詳細情報は、以下の資料から入手できます。

表 *25. z/OS* バッチと *IBM MQ* を同時に使用する方法に関する詳細が記載されている資料

トピック	参照先
バッチ・アダプターの設定	作業 19: バッチ、TSO、および RRS アダプターをセットアップする
RRS 呼び出し可能資源回復サービス	*MVS* プログラミング: 高水準言語向け呼び出し可能サービス

z/OS IBM MQ for z/OS と WebSphere Application Server

このトピックでは、 WebSphere Application Server による IBM MQ for z/OS の使用について説明します。

WebSphere Application Server で実行される Java で作成されたアプリケーションは、 Java Message Service (JMS) 仕様を使用してメッセージングを実行できます。 この環境での Point-to-Point メッセージングは、IBM MQ for z/OS キュー・マネージャーを使用して実現できます。

IBM MQ for z/OS キュー・マネージャーを使用してメッセージングを実現することの利点は、JMS アプリケーションを接続することで IBM MQ ネットワークの機能をフルに活用できることです。 例えば、IMS ブリッジを使用したり、他のプラットフォームで稼働中のキュー・マネージャーとメッセージを交換したりできます。

WebSphere Application Server とキュー・マネージャー間の接続

詳しくは、 IBM MQ と WebSphere Application Server の併用 を参照してください。

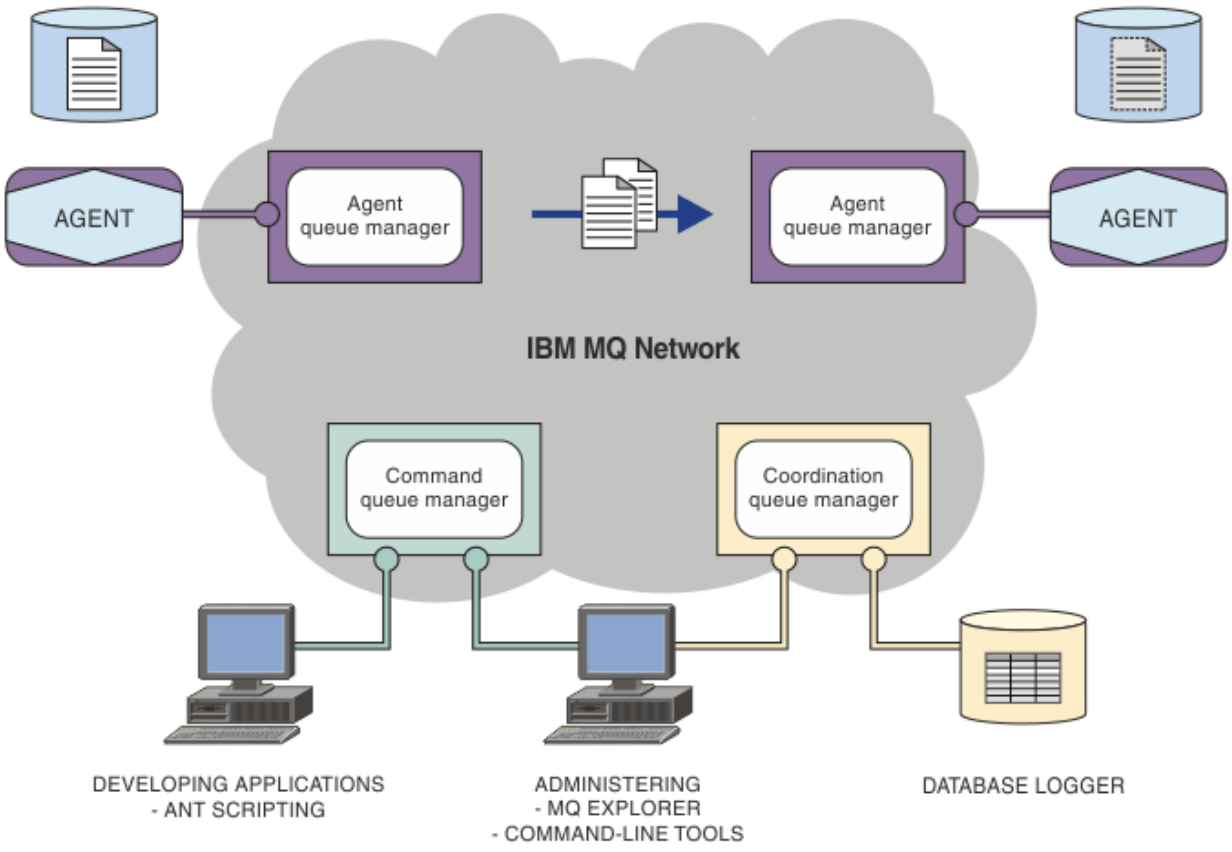
JMS アプリケーションからの IBM MQ 機能の使用

デフォルトでは、IBM MQ キューに保持される JMS メッセージは、JMS メッセージ・ヘッダー情報の一部を保持するために MQRFH2 ヘッダーを使用します。 多数のレガシー IBM MQ アプリケーションは、これらのヘッダー付きメッセージを処理できず、独自の特性ヘッダー (例えば、CICS ブリッジ用 MQCIH、または IBM MQ Workflow アプリケーション用 MQWIH) が必要です。 これらの特別な考慮事項の詳細については、JMS メッセージの IBM MQ メッセージへのマッピングを参照してください。

Managed File Transfer

Managed File Transfer は、ファイルのサイズや使用するオペレーティング・システムにかかわりなく、システム間のファイル転送を管理下に置いて実行できます。監査も可能です。

Managed File Transfer を使用すれば、ファイル転送を管理し、確認し、保護するために、カスタマイズしたスケーラブルな自動化ソリューションを構築できます。 Managed File Transfer によって、コストのかかる冗長性を除去し、保守コストを削減し、既存の IT 投資を最大限に活用することが可能になります。



この図は、単純な Managed File Transfer トポロジーを示しています。 2 つのエージェントがあり、それぞれが IBM MQ ネットワーク内の独自のエージェント・キュー・マネージャーに接続しています。 図の一方にあるエージェントから IBM MQ ネットワークを経由して、図のもう一方にあるエージェントにファイルを転送します。 さらに、IBM MQ ネットワークには、調整キュー・マネージャーとコマンド・キュー・マネージャーもあります。 アプリケーションおよびツールは、これらのキュー・マネージャーに接続して、Managed File Transfer ネットワーク内の IBM MQ アクティビティーを構成、管理、操作、およびログに記録します。

Managed File Transfer には、オペレーティング・システムと全体的なセットアップに応じて、4 種類のインストール・オプションがあります。 これらのオプションは、Managed File Transfer Agent、Managed File Transfer Logger、Managed File Transfer Service、または Managed File Transfer Tools です。 詳しくは、Managed File Transfer 製品のオプションを参照してください。

Managed File Transfer を使用して、次のタスクを実行できます。

- 管理対象ファイル転送を作成します。
  - Windows Linux Linux または Windows プラットフォーム上の IBM MQ Explorer から新規ファイル転送を作成します。
  - サポートされているすべてのオペレーティング・システムで、コマンド・ラインから新しいファイル転送を作成できます。
  - ファイル転送機能を Apache Ant ツールに組み込みます。
  - エージェント・コマンド・キューにメッセージを PUT することによって、Managed File Transfer を制御するアプリケーションを作成します。
  - ファイル転送は、後の時点で実行されるようにスケジュールに入れます。 また、スケジュール済みファイル転送を、一定の範囲のファイル・システム・イベント (例えば、新規ファイルの作成など) に基づいてトリガーすることもできます。
  - 例えばディレクトリーなどのリソースを継続的にモニターして、そのリソースの内容が事前定義の条件に一致した場合にタスクを開始します。 このタスクは、ファイル転送、Ant スクリプト、または JCL ジョブにすることができます。
  - IBM MQ キューとの間のファイル転送が可能です。
  - FTP サーバー、FTPS サーバー、または SFTP サーバーとの間でファイル転送が可能です。
  - Connect:Direct® ノードとの間のファイル転送が可能です。
  - テキスト・ファイルとバイナリー・ファイルの両方の転送が可能です。 テキスト・ファイルの場合、ソース・システムと宛先システムの間でコード・ページと行の終わり規則が自動的に変換されます。
  - 転送は、SSL (Secure Socket Layer) ベース接続の業界規格を使用して保護できます。
- 転送の進行状況を表示することや、ネットワーク内のすべての転送に関する情報をログに記録することが可能です。
  - Windows Linux Linux または Windows プラットフォーム上の IBM MQ Explorer から、進行中の転送の状況を表示します。
  - Windows Linux Linux または Windows プラットフォームで IBM MQ Explorer を使用して、完了した転送の状況を確認します。
  - Managed File Transfer のデータベース・ロガー機能を使用して、Db2 または Oracle データベースにログ・メッセージを保存します。

Managed File Transfer は、IBM MQ の基盤の上に構築されている製品であり、アプリケーション間の 1 回限りのメッセージ配信を確実に実行できるようになっています。 IBM MQ のさまざまなフィーチャーを活用することができます。 例えば、チャネル圧縮を使用して、IBM MQ チャネルを介してエージェント間で送信するデータを圧縮し、SSL チャネルを使用して、エージェント間で送信するデータを保護することができます。 ファイルは安全に転送され、ファイル転送を行う媒体となるインフラストラクチャーで発生した障害に対処する機能があります。 ネットワーク障害が発生した場合、接続が復元されたときに、ファイル転送は中止された位置から再開します。

ファイル転送を既存の IBM MQ ネットワークと統合することにより、2 つの別個のインフラストラクチャーを保守して必要なリソースを浪費するということを避けられます。 まだ IBM MQ のお客様でない場合

は、Managed File Transfer をサポートする IBM MQ ネットワークを作成することにより、将来の SOA 実装のためのバックボーンを構築します。 既にお客様が IBM MQ 顧客であれば、Managed File Transfer は IBM MQ や IBM MQ Internet Pass-Thru などの既存の IBM Integration Bus インフラストラクチャーを利用できます。

IBM MQ 高可用性ソリューションを利用して、 Managed File Transfer 構成の回復力を向上させることができます。 エージェントが複製データ・キュー・マネージャー (RDQM) を使用する場合は、浮動 IP アドレス機能を使用するように構成する必要があります。 これは、エージェントが同じ IP アドレスを使用して、現在実行中の 3 つの RDQM インスタンスのいずれかと通信し、フェイルオーバー時に自動的に再接続することを意味します (RDQM 高可用性 および 浮動 IP アドレスの作成と削除を参照してください)。 複数インスタンス・キュー・マネージャー・ソリューションを使用する場合、アプリケーションは、各インスタンスとの通信に異なる IP アドレスを使用します。これは、フェイルオーバー時にクライアント再接続によって処理されます (複数インスタンス・キュー・マネージャー および チャネルとクライアントの再接続を参照)。

Managed File Transfer は、以下のように、他の多くの IBM 製品との統合が可能です。

IBM Integration Bus

Managed File Transfer によって転送されたファイルを IBM Integration Bus フローの一部として処理できます。 詳しくは、Working with MFT from IBM Integration Bus を参照してください。

IBM Sterling Connect:Direct

Managed File Transfer Connect:Direct ブリッジを使用して、既存の Connect:Direct ネットワークとの間でファイルを転送します。 詳しくは、Connect:Direct ブリッジを参照してください。

IBM Tivoli® Composite Application Manager

IBM Tivoli Composite Application Manager には、調整キュー・マネージャーにパブリッシュされた情報をモニターするために使用できるエージェントが用意されています。

関連概念

Managed File Transfer 製品のオプション

302 ページの『MFT トポロジーの概要』
Managed File Transfer エージェントが IBM MQ ネットワーク内の調整キュー・マネージャーとどのように接続されるかについての概要。

301 ページの『MFT と IBM MQ の連動について』
Managed File Transfer は、さまざまな方法で IBM MQ と対話します。

MFT と IBM MQ の連動について

Managed File Transfer は、さまざまな方法で IBM MQ と対話します。

- Managed File Transfer は、各ファイルを 1 つ以上のメッセージに分割し、それらメッセージを IBM MQ ネットワークを介して送信することにより、エージェント・プロセス間でファイルを転送します。
- エージェント・プロセスは、IBM MQ ログに対する影響を最小化するために、非永続メッセージを使用してファイル・データを移動します。 エージェント・プロセスは、相互にやり取りすることにより、ファイル・データが含まれるメッセージのフローを調整します。 このようにして、ファイル・データが含まれているメッセージが IBM MQ 伝送キューに蓄積される状況が回避され、いずれかの非永続メッセージが送信されなかった場合にファイル・データが確実に再送信されるようになります。
- Managed File Transfer エージェントは、いくつかの IBM MQ キューを使用します。 詳しくは、MFT システム・キューおよびシステム・トピックを参照してください。
- これらのキューの一部は内部使用に限られていますが、エージェントは、読み取り先の特定のキューに送信される特殊形式のコマンド・メッセージの形で要求を受け入れることができます。 コマンド行コマンドおよび IBM MQ Explorer・プラグインの両方は、IBM MQ メッセージをエージェントに送信し、対象となるアクションを実行するようにエージェントに指示します。 このような方法でエージェントと対話する IBM MQ アプリケーションを作成できます。 詳しくは、エージェント・コマンド・キューへのメッセージの書き込みによる MFT の制御を参照してください。
- Managed File Transfer エージェントは、その状態と、転送の進行状況と結果に関する情報を、調整キュー・マネージャー として指定されている MQ キュー・マネージャーに送信します。 この情報は、調整キ

ュー・マネージャーによりパブリッシュされ、転送の進行状況のモニターまたは発生した転送の記録を行うアプリケーションによってサブスクライブできます。 コマンド行コマンドおよび IBM MQ Explorer・プラグインの両方で、パブリッシュされた情報を利用できます。 この情報を使用する IBM MQ アプリケーションを作成できます。 情報がパブリッシュされるトピックについて詳しくは、 SYSTEM.FTE トピック。

- Managed File Transfer のキー・コンポーネントは、IBM MQ キュー・マネージャーの機能を利用してメッセージのストア・アンド・フォワード処理を行います。 これは、故障が発生した場合、インフラストラクチャーの中で影響を受けていない部分はファイルの転送を続行できることを意味します。 このことは調整キュー・マネージャーにも当てはまります。ストア・アンド・フォワードと永続サブスクリプションの組み合わせにより、調整キュー・マネージャーは、使用不可状態になっても行われたファイル転送に関する主要な情報を失うことなく対処できます。

MFT トポロジーの概要

Managed File Transfer エージェントが IBM MQ ネットワーク内の調整キュー・マネージャーとどのように接続されるかについての概要。

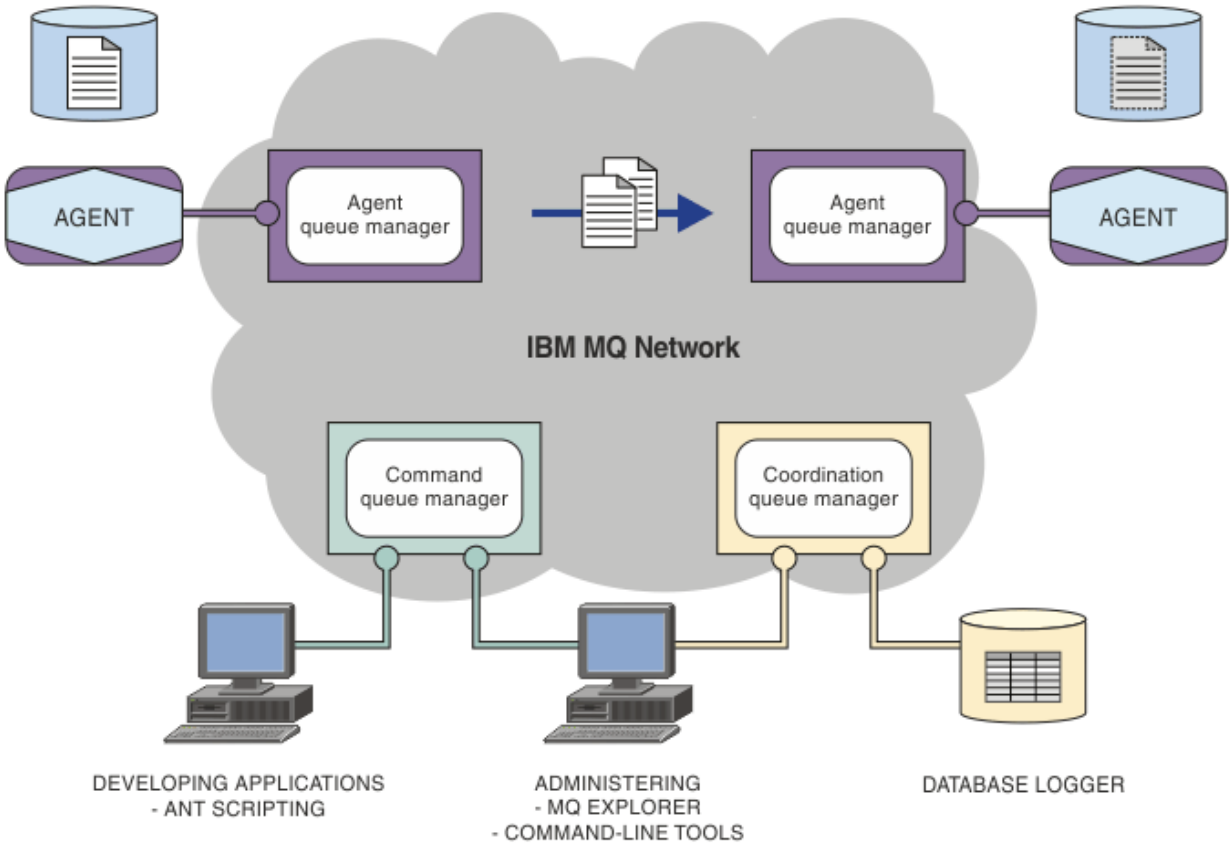
Managed File Transfer エージェントは、転送されるファイルを送受信します。 エージェントはそれぞれ、関連付けられているキュー・マネージャーに対して独自の一連のキューを持ちます。 エージェントはバインディング・モードまたはクライアント・モードのいずれかでキュー・マネージャーに接続されます。 また、エージェントはそのキュー・マネージャーとして調整キュー・マネージャーを使用することもできます。

調整キュー・マネージャーは、監査およびファイル転送情報をブロードキャストします。 調整キュー・マネージャーは、エージェント、転送状況、および転送監査の情報を収集するためのシングル・ポイントの役割を果たします。 調整キュー・マネージャーが使用可能な状態になっていることは、転送を実行するための必要条件ではありません。 調整キュー・マネージャーが一時的に使用できなくなった場合でも、転送処理は、通常どおり続行されます。 監査メッセージと状況メッセージは、調整キュー・マネージャーが使用できるようになって、通常の処理が可能になるまで、エージェント・キュー・マネージャーに格納されます。

エージェントは調整キュー・マネージャーに登録され、詳細をそのキュー・マネージャーにパブリッシュします。 このエージェント情報は、IBM MQ Explorer からの転送を開始できるようにするために Managed File Transfer プラグインによって使用されます。 各種のコマンドも、エージェントの情報と状況を表示するために、調整キュー・マネージャーで収集されるエージェント情報を使用します。

転送状況と転送監査の情報は、調整キュー・マネージャーでパブリッシュされます。 Managed File Transfer プラグインはその転送状況と転送監査の情報を使用して、IBM MQ Explorer から転送の進行状況をモニターします。 監査能力を確保するために、調整キュー・マネージャーに格納される転送監査の情報を保存することもできます。

IBM MQ ネットワークに接続する場合にはこのコマンド・キュー・マネージャーが使用され、このコマンド・キュー・マネージャーが Managed File Transfer のコマンドを発行するときに接続されるキュー・マネージャーとなります。



関連概念

299 ページの『Managed File Transfer』
Managed File Transfer は、ファイルのサイズや使用するオペレーティング・システムにかかわりなく、システム間のファイル転送を管理下に置いて実行できます。監査も可能です。

301 ページの『MFT と IBM MQ の連動について』
Managed File Transfer は、さまざまな方法で IBM MQ と対話します。

Managed File Transfer シナリオ

MFT REST API の概要

REST API は、転送のリスト表示やファイル転送エージェントの詳細など、特定の Managed File Transfer コマンドをサポートします。

IBM MQ 9.1.0 以降、REST API には、現在のすべての Managed File Transfer 転送をリストするためのオプションと、Managed File Transfer エージェントの状況を照会するためのオプションが含まれています。 詳しくは、「REST API MFT の使用を開始する」を参照してください。

IBM MQ Internet Pass-Thru

IBM MQ Internet Pass-Thru (MQIPT) は IBM MQ のオプション・コンポーネントで、インターネットを介してリモート・サイト間のメッセージング・ソリューションを実装するために使用できます。

IBM MQ 9.2.0 から、MQIPT は IBM MQ のオプション・コンポーネントです。 IBM MQ 9.3.x の MQIPT インストール・ファイルを入手するには、 IBM Fix Central (IBM MQ の場合)にアクセスしてください。 IBM MQ 9.2.0 より前のバージョンでは、MQIPT は サポート・パックとして提供されていました。

IBM MQ 9.3 で MQIPT を使用するために IBM MQ 9.3 を実行する必要はありません。 MQIPT を使用して、サポートされているバージョンの IBM MQ を接続することができます。また、MQIPT と同じバージョンに他の IBM MQ コンポーネントをインストールする必要はありません。

IBM MQ のライセンスを購入した場合は、必要な数の MQIPT のコピーをインストールできます。 MQIPT インストール済み環境は、購入した IBM MQ ライセンスを消費するものとしてカウントされません。 IBM MQ のライセンス交付について詳しくは、IBM MQ のライセンス情報を参照してください。

注 : この資料は、 IBM MQ 9.3 の MQIPT に関連しています。 IBM Documentation の MQIPT サポート・パック (バージョン 2.1) の資料については、 IBM MQ 9.0 資料の MQIPT (SupportPac MS81) を参照してください。

注 : MQIPT 2.1 以前を使用している場合は、MQIPT サポート・パックのサポート終了日が 2020 年 9 月 30 日 30th ため、 MQIPT for IBM MQ 9.3 にアップグレードすることをお勧めします。

IBM MQ Internet Pass-Thru は、2 つの IBM MQ キュー・マネージャー間、または IBM MQ クライアントと IBM MQ キュー・マネージャー間の IBM MQ メッセージ・フローを受け取って転送することができる、スタンドアロンのサービスとして実行されます。

クライアントとサーバーが同じ物理ネットワーク上にない場合、MQIPT によってこのような接続が使用可能になります。

MQIPT の 1 つ以上のインスタンスを、2 つの IBM MQ キュー・マネージャー間の通信パス、または IBM MQ クライアントと IBM MQ キュー・マネージャーの間の通信パスに配置することができます。 MQIPT のインスタンスによって、2 つの IBM MQ システムが、2 つのシステム間に直接 TCP/IP 接続がなくてもメッセージを交換することができます。 このことは、ファイアウォール構成で 2 つのシステム間の直接 TCP/IP 接続が禁止されている場合に便利です。

MQIPT は 1 つ以上の TCP/IP ポートで着信接続を listen しますが、この接続では通常の IBM MQ メッセージ、HTTP トンネリングによる IBM MQ メッセージ、または Transport Layer Security (TLS) や Secure Sockets Layer (SSL) を使用して暗号化されたメッセージのいずれかを伝送することができます。 MQIPT は複数の同時接続を処理することができます。

最初に TCP/IP 接続要求を行う IBM MQ チャネルは呼び出し側 と呼ばれ、接続先のチャネルは応答側 と呼ばれます。また、最終的な接続先となるキュー・マネージャーは宛先キュー・マネージャー と呼ばれます。

MQIPT は、ソースから宛先にデータを転送する際、そのデータをメモリー内に保持します。 ディスク上にデータが保存されることはありません (ただし、オペレーティング・システムによってディスクにページングされるメモリーは例外です)。 MQIPT が明示的にディスクにアクセスするのは、構成ファイルを読み取り、接続ログとトレース・レコードを書き込む場合のみです。

IBM MQ チャネル・タイプの全範囲を、1 つ以上の MQIPT インスタンスを介して行うことができます。 通信パスに MQIPT が存在しても、接続されている IBM MQ コンポーネントの機能特性が影響を受けることはありませんが、メッセージ転送のパフォーマンスには何らかの影響がある可能性があります。

MQIPT は、IBM MQ に説明されるように、IBM Integration Bus および 307 ページの『MQIPT の考えられる構成』 と連結して使用できます。

MQIPT をインストールするには、MQIPT のインストールを参照してください。

関連タスク

IBM MQ Internet Pass-Thru の構成

IBM MQ Internet Pass-Thru の管理および構成

関連資料

IBM MQ Internet Pass-Thru 構成リファレンス

MQIPT の用途

IBM MQ Internet Pass-Thru (MQIPT) は以下のようなさまざまな用途に使用することができます。

MQIPT をチャネル・コンセントレーターとして使用する

この用途で MQIPT を使用することによって、ファイアウォールで、複数の個別のホストとの間のチャネルをすべて MQIPT ホストとの間のチャネルであるかのように認識できます。 これにより、ファイアウォールのフィルタリング・ルールの定義と管理が容易になります。

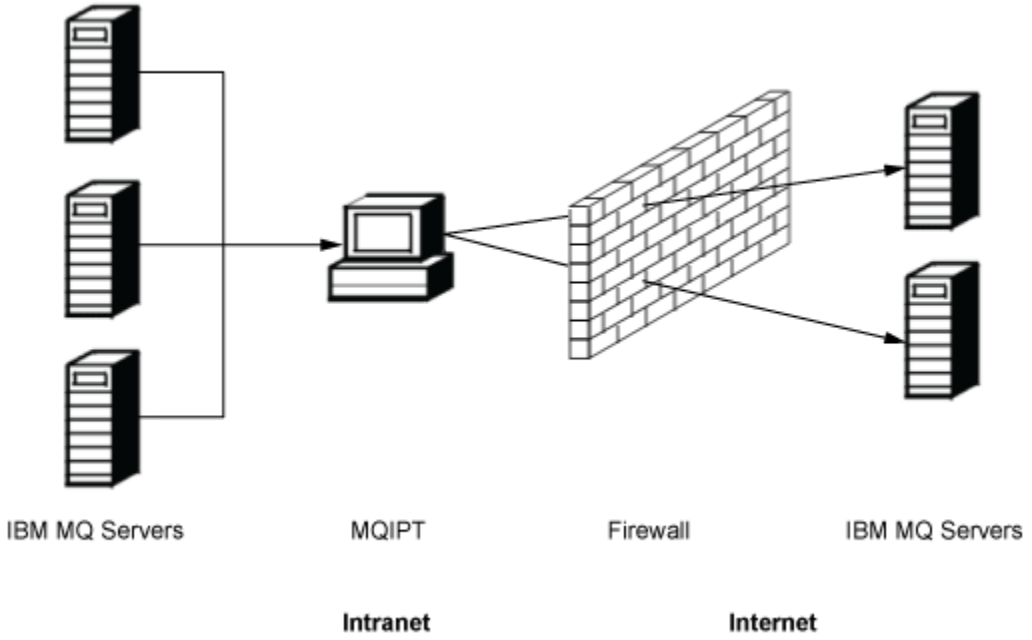


図 *79.* チャネル・コンセントレーターとしての *MQIPT* の例

MQIPT を DMZ に配置して単一アクセス・ポイントを提供する

信頼された既知のインターネット・プロトコル (IP) アドレスを持つコンピューター上で、DMZ ファイアウォール内に MQIPT が配置されている場合 (ローカル・エリア・ネットワークを保護するためのファイアウォール構成)、MQIPT を使用して、着信 IBM MQ チャネル接続を listen し、それを信頼されたイントラネットに転送することができます。内側のファイアウォールでは、この信頼されたコンピューターがインバウンド接続を確立できるよう許可する必要があります。 この構成では、MQIPT により、外部のアクセス要求が、信頼されたイントラネット内のコンピューターの本当の IP アドレスを受信することができません。 このようにして、MQIPT は、単一アクセス・ポイントを提供します。 必要に応じて、TLS 接続を受け入れ、別個の TLS 接続を使用してデータを宛先に転送するように MQIPT を構成できます。これにより、DMZ 内の TLS セッションが終了します。

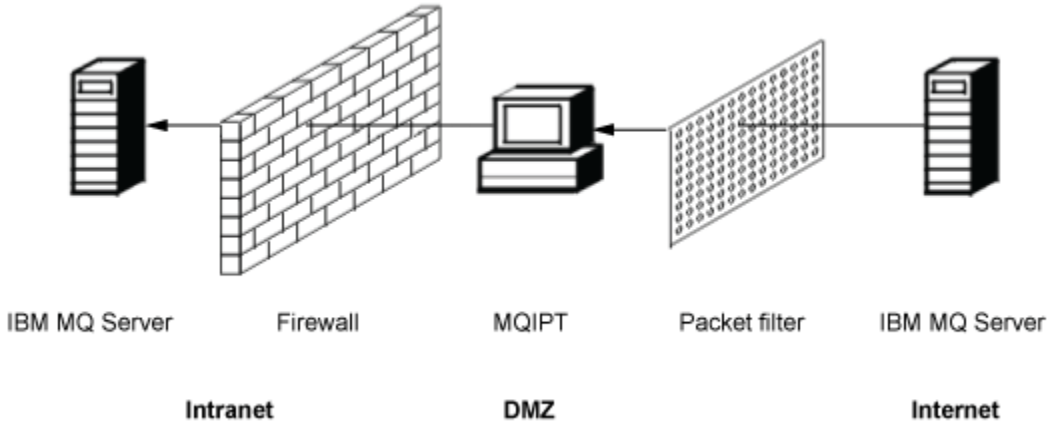


図 *80. DMZ* ファイアウォール内の *MQIPT* の例

HTTP トンネリングによって通信できる MQIPT

直列でデプロイされている 2 つの MQIPT インスタンスは、HTTP を使用して通信することができます。 HTTP トンネリング機能によって、既存の HTTP プロキシーを使用することによりファイアウォールを介して要求を伝送することが可能です。 最初の MQIPT は HTTP に IBM MQ プロトコルを挿入し、2 番目のものはその HTTP ラッパーから IBM MQ プロトコル を抽出して、それを宛先キュー・マネージャーに送ります。

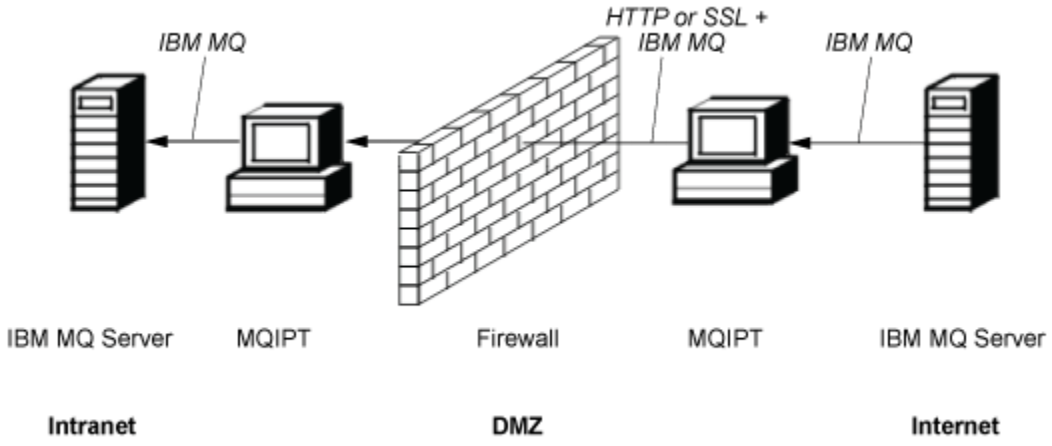


図 *81. MQIPT* および *HTTP* トンネリングの例

MQIPT でメッセージを暗号化する

前の例のように MQIPT が構成されている場合、要求は、暗号化した後にファイアウォールを介して送信することができます。 1 つ目の MQIPT がデータを暗号化し、2 つ目が SSL/TLS を使用して復号した後、宛先キュー・マネージャーに送信します。

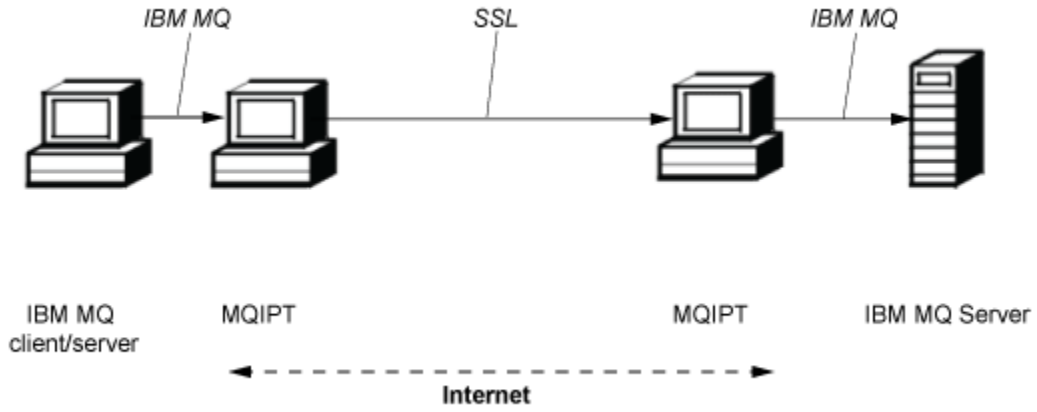


図 *82. MQIPT* および *SSL/TLS* の例

MQIPT の動作

最も単純な構成では、MQIPT は IBM MQ の プロトコル・フォワーダーとして機能します。 TCP/IP ポートを listen し、IBM MQ チャネルからの 接続要求を受け入れます。

正しい形式の要求を受信すると、MQIPT は、 自身と宛先 IBM MQ キュー・マネージャーとの間で TCP/IP 接続を 確立します。 次に、着信接続から受信したすべてのプロトコル・パケットを 宛先キュー・マネージャーに渡してから、宛先キュー・マネージャーからのプロトコル・パケットを元の着信接続に戻します。

両側とも中継の存在について直接認識していないため、IBM MQ プロトコル (クライアント/サーバーまたはキュー・マネージャー対キュー・マネージャー) に変更を 加える必要はありません。 IBM MQ クライアントまたはサーバーのコードの新規バージョンは 必要ありません。

MQIPT を使用するには、宛先キュー・マネージャーの ホスト名とポートではなく、MQIPT のホスト名とポートを使用するように、 呼び出し元チャネルを構成する必要があります。 これは、 IBM MQ チャネルの **CONNAME** プロパティーで定義されます。 MQIPT は 着信データを読み取り、単純にそれを宛先キュー・マネージャーに渡します。 クライアント/サーバー・チャネル内の ユーザー ID やパスワードなどの他の構成フィールドも、同様に宛先キュー・マネージャーに渡されます。

複数のキュー・マネージャー

MQIPT を使用して、複数の宛先キュー・マネージャーへのアクセスを許可できます。 このためには、どのキュー・マネージャーに接続するかを MQIPT に通知するためのメカニズムが 必要です。MQIPT は、着信 TCP/IP ポート番号を使用して、 接続するキュー・マネージャーを判別します。

したがって、複数の TCP/IP ポートを listen するように MQIPT を構成できます。 各リスニング・ポートは、MQIPT の経路 を介して 宛先キュー・マネージャーにマップされます。 リスニング TCP/IP ポートを宛先キュー・マネージャーのホスト名とポートに関連付ける経路を、最大 100 まで定義できます。 これは、宛先キュー・マネージャーのホスト名 (IP アドレス) は、発生元のチャネルからは見えないことを意味します。 各経路は、リスニング・ポートと宛先間の複数の接続を処理できます。各接続は独立して機能します。

MQIPT 構成ファイル

MQIPT は `mqipt.conf` という構成ファイルを使用します。 このファイルには、すべての経路およびその関連プロパティーの定義が 含まれています。 `mqipt.conf` について詳しくは、「IBM MQ Internet Pass-Thru の管理および構成」を参照してください。

MQIPT を起動すると、構成ファイルにリストされている各経路が 開始されます。 各経路の状況を示すメッセージがシステム・コンソールに 書き込まれます。 経路についてメッセージ MQCPI078 が表示される場合、その経路で 接続要求を受け入れる準備ができたことを示しています。

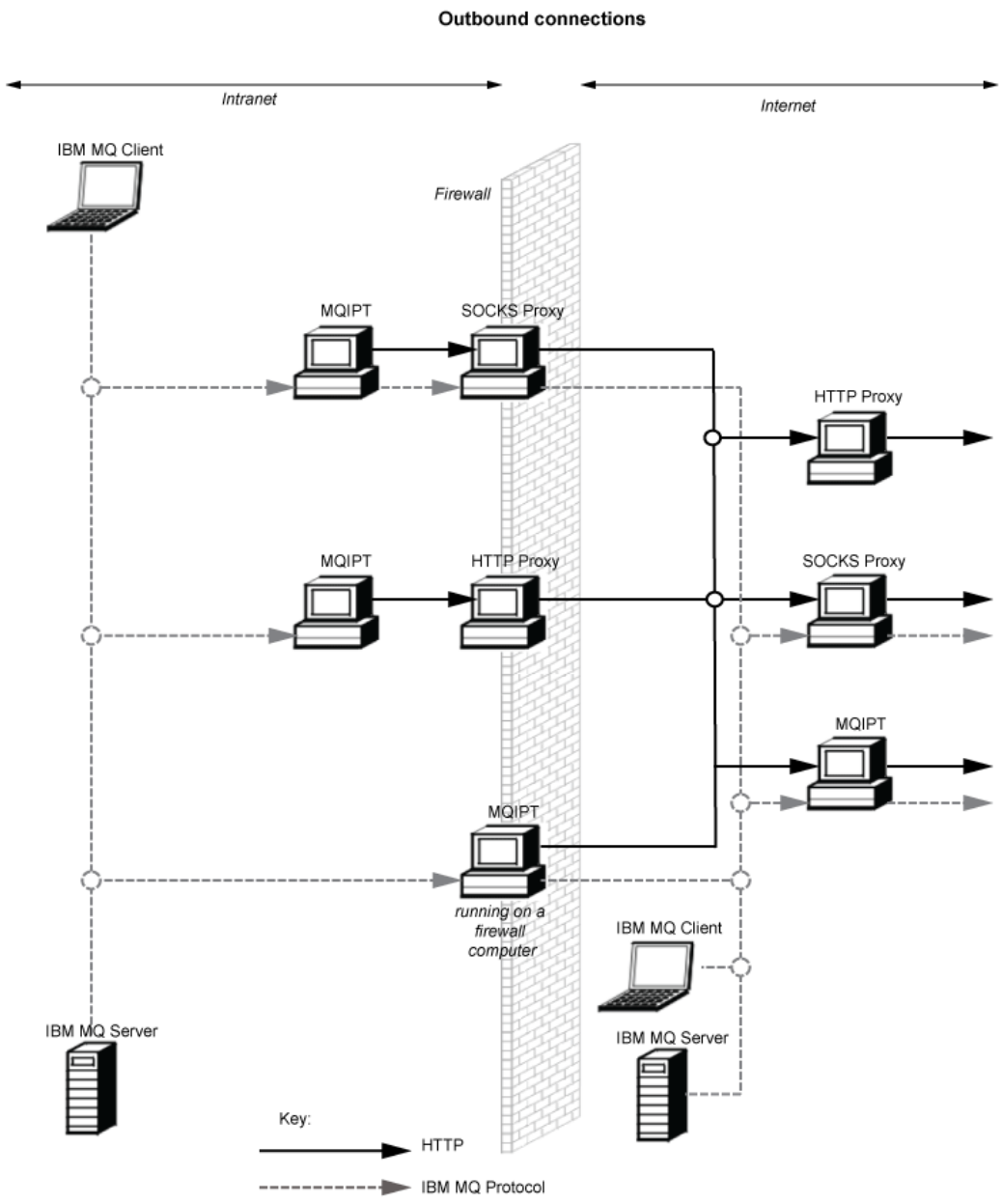
MQIPT の考えられる構成

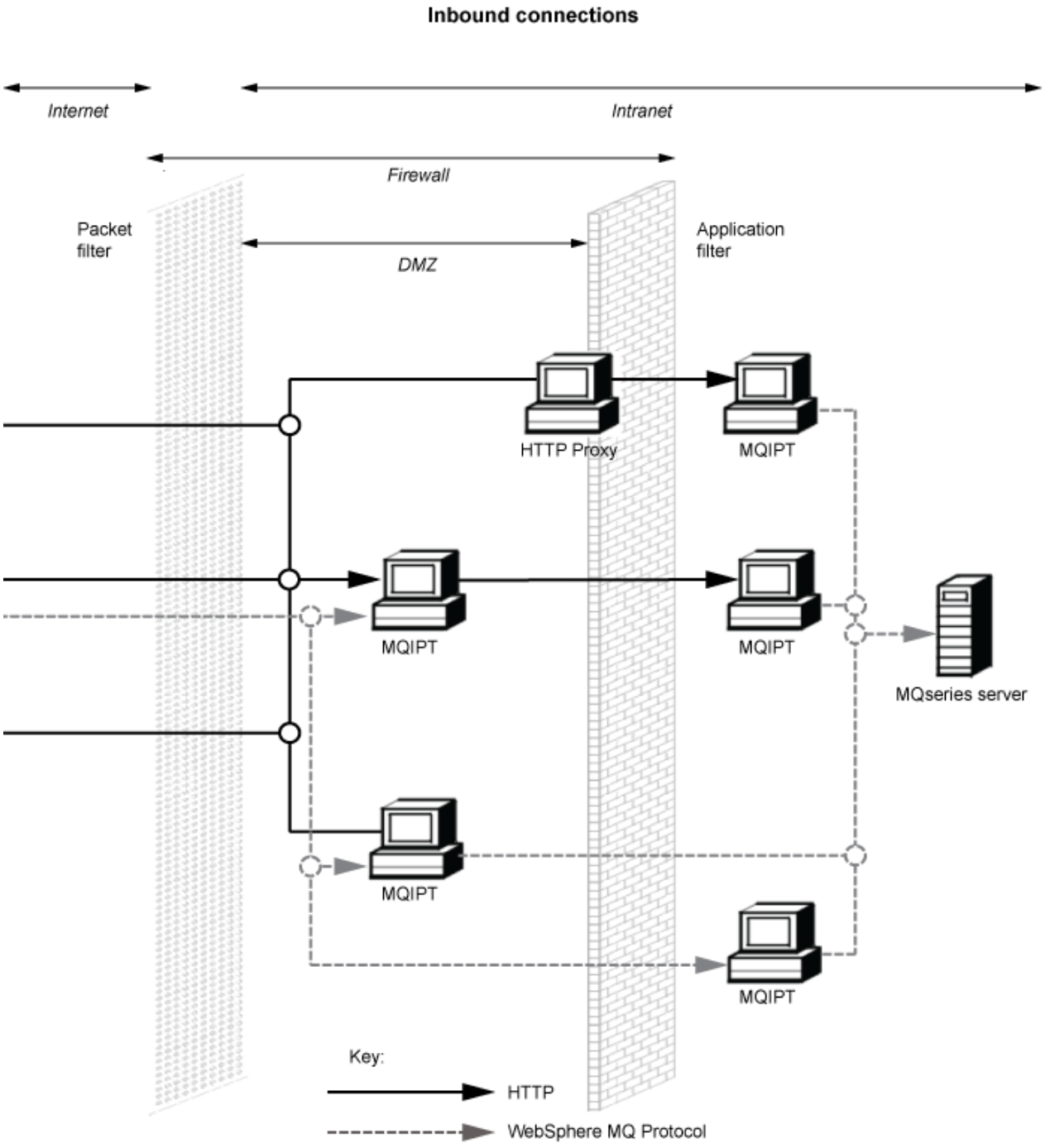
MQIPT は IBM MQ および IBM Integration Bus と組み合わせて使用できます。

以下のマルチパートの図は、IBM MQ トポロジー内の MQIPT で可能な構成の多くを示しています。 ここには、MQIPT がメッセージを送信できるさまざまな方法が示されています。 イントラネット、ファイアウォールの内側、およびファイアウォールの外側のインターネット上にあるクライアントとサーバーが、メッセージを MQIPT、HTTP プロキシー、または SOCKS プロキシーに渡して転送する様子が示されています。

インバウンド・ファイアウォールを介してサーバーにメッセージが渡される前に、DMZ の MQIPT プロキシーまたは HTTP プロキシーがメッセージを受信します。

ファイアウォールのイントラネット側の HTTP プロキシー、SOCKS プロキシー、および MQIPT コンピューターは、インターネット上で複数のコンピューターが相互にチェーニングされる可能性があることを示していることに注意してください。 例えば、MQIPT コンピューターは、ターゲットに達する前に、1 つ以上の SOCKS または HTTP プロキシー・コンピューター、さらに MQIPT コンピューターを介して通信する可能性があります。


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



互換性のある構成

IBM MQ クライアントまたはキュー・マネージャーが MQIPT と通信する、互換性のある接続シナリオ。 同一または 2 つ目の MQIPT 経路が、宛先キュー・マネージャーとの通信に使用されます。

単一の MQIPT 経路を使用した互換性のある構成

単一の MQIPT 経路を使用して、IBM MQ と通信することができます。

310 ページの表 26 の列には、以下の情報が含まれます。

1. IBM MQ と MQIPT の経路の間で使用されるプロトコル。 接続は IBM MQ クライアントまたはキュー・マネージャーのいずれかから作成でき、IBM MQ Formats and Protocols (FAP) または SSL/TLS プロトコルのいずれかを使用できます。
2. MQIPT 経路が作動するモード。 MQIPT と IBM MQ との間のインターネット経由での通信形式は、MQIPT 経路の構成によって決まります。 表で SSL が言及されている場合は TLS も使用することができます。
3. MQIPT 経路と宛先キュー・マネージャーの間で使用されるプロトコル。

表 *26. MQIPT* の単一インスタンスでの有効な構成

1 秒. IBM MQ ソース・プロトコル	**2. MQIPT 経路のモード**	**3. IBM MQ 宛先プロトコル**
FAP	FAP プロキシー (デフォルト)	FAP
	FAP サーバーおよび SSL クライアント	SSL/TLS
SSL/TLS	SSL プロキシー	SSL/TLS
	SSL サーバーおよび FAP クライアント	FAP
	SSL サーバーおよび SSL クライアント	SSL/TLS

複数の MQIPT 経路を使用した互換性のある構成

1 つ以上の MQIPT インスタンスで複数の経路を使用して、IBM MQ と通信するよう選択することができます。

311 ページの表 27 の列には、以下の情報が含まれます。

1. IBM MQ と 1 つ目の MQIPT 経路の間で使用されるプロトコル。 接続は IBM MQ クライアントまたはキュー・マネージャーのいずれかから作成でき、IBM MQ Formats and Protocols (FAP) または SSL/TLS プロトコルのいずれかを使用できます。
2. 1 つ目の MQIPT 経路が作動するモード。 MQIPT と IBM MQ との間のインターネット経由での通信形式は、MQIPT 経路の構成によって決まります。 表で SSL が言及されている場合は TLS も使用することができます。
3. 2 つ目の MQIPT 経路が作動するモード。
4. 2 つ目の MQIPT 経路と宛先キュー・マネージャーの間で使用されるプロトコル。

表 27. *MQIPT* の複数インスタンスでの有効な構成

1 秒. IBM MQ ソース・プロトコル	2. 1 つ目の MQIPT 経路のモード	3. 2 つ目の MQIPT 経路のモード	4. IBM MQ 宛先プロトコル
FAP (デフォルト)	FAP プロキシー (デフォルト)	FAP プロキシー (デフォルト)	FAP
	FAP サーバーおよび SSL クライアント	SSL プロキシー	SSL/TLS
		SSL サーバーおよび FAP クライアント	FAP
		SSL サーバーおよび SSL クライアント	SSL/TLS
	HTTP クライアント	HTTP サーバーおよび SSL クライアント	SSL/TLS
	HTTPS クライアント	HTTPS サーバーおよび SSL クライアント	SSL/TLS
	HTTP クライアント	HTTP サーバー	FAP
	HTTPS クライアント	HTTPS サーバー	FAP
SSL/TLS	SSL プロキシー	SSL プロキシー	SSL/TLS
		SSL サーバーおよび FAP クライアント	FAP
		SSL サーバーおよび SSL クライアント	SSL/TLS
	HTTP クライアント	HTTP サーバー	FAP
	HTTPS クライアント	HTTPS サーバー	SSL/TLS
	HTTP クライアント	HTTP サーバーおよび SSL クライアント	FAP
	HTTPS クライアント	HTTPS サーバーおよび SSL クライアント	SSL/TLS

サポートされるチャネルの構成

IBM MQ のすべてのチャネル・タイプがサポートされますが、 構成は TCP/IP 接続に制限されます。 IBM MQ クライアントや キュー・マネージャーには、MQIPT は宛先キュー・マネージャーのように 見えます。 チャネル構成で宛先ホストとポート番号が必要な場合は、MQIPT の ホスト名とリスナー・ポート番号が指定されます。

クライアント/サーバー・チャネル
MQIPT は着信クライアント接続要求を listen し、HTTP トンネリング か SSL/TLS を使用するか、または標準の IBM MQ プロトコル・パケット として、その要求を転送します。 MQIPT が HTTP トンネリング または SSL/TLS を使用している場合、接続上で要求を 2 番目 の MQIPT に転送します。 HTTP トンネリングを使用していない場合は、 接続上で宛先キュー・マネージャーのように見える宛先に要求を転送します (これはさらに 先の MQIPT である場合があります)。 宛先キュー・マネージャーが クライアント接続を受け入れると、クライアントとサーバー間でパケットがリレーされます。

クラスター送信側/受信側チャネル
クラスター送信側チャネルから着信要求を受信すると、MQIPT は キュー・マネージャーが SOCKS 対応であると見なし、真の宛先アドレスが SOCKS ハンドシェーク・プロセス中に 取得されます。 要求は、クライアント接続チャネルの場合と同様に、次の MQIPT または 宛先キュー・マネージャーに転送されます。 これには、自動定義のクラスター送信側チャネルも含まれます。

送信側/受信側
送信側チャネルから着信要求を 受信すると、MQIPT は、クライアント接続チャネルの場合と同様に、要求を次の MQIPT または宛先キュー・マネージャーに転送します。 宛先キュー・マネージャーは 着信要求を検証し、適切な場合は受信側チャネルを開始します。 送信側と受信側チャネル間の すべての通信 (セキュリティー・フローなど) がリレーされます。

要求側/サーバー
この組み合わせは、前の構成と同様の方法で 処理されます。 接続要求の検証は、宛先キュー・マネージャーのサーバー・チャネルで実行されます。

要求側/送信側
2 つのキュー・マネージャーで相互に直接接続を 確立することが許可されていないが、両方とも MQIPT への接続は許可されており、 そこからの接続を受け入れることができる場合は、「コールバック」構成が有用です。

サーバー/要求側およびサーバー/受信側
これらは、 Sender/Receiver 構成を処理する場合と同じ方法で MQIPT によって処理されます。

チャネルの終了と障害の状態

MQIPT は IBM MQ チャネルのクローズ (正常または異常) を検出すると、そのチャネル・クローズを伝搬します。 MQIPT を使用して経路をクローズすると、その経路を通過するすべてのチャネルがクローズされます

MQIPT は、オプションのアイドル・タイムアウト機能を備えています。 MQIPT は、チャネルがタイムアウトを超過して一定期間アイドル状態になっていることを検出すると、該当する 2 つの接続の即時シャットダウンを実行します。

チャネルの両端の IBM MQ システムは、これらの異常シャットダウン状態をネットワーク障害、またはパートナーによるチャネルの終了と見なします。 その後、MQIPT を使用していない場合と同じように、チャネルを再始動してリカバリーできます (障害がプロトコル未確定期間中に発生した場合)。

メッセージの安全性

IBM MQ 分散キュー管理によって、 メッセージが確実に適切な方法で送達されます。 これは、チャネルの両側に MQIPT が存在する場合にも当てはまります。 MQIPT は、メッセージ・データを保管することや、適切なメッセージの送達を確実にする同期点プロシージャーに関与することはしません。

高速の非持続 IBM MQ メッセージを使用している場合、IBM MQ メッセージの転送中に MQIPT 経路に障害が発生したり、経路が再始動したりすると、メッセージが失われる可能性があります。 経路を再始動する前に、 MQIPT 経路を使用するすべての IBM MQ チャネルが非アクティブであることを確認してください。

IBM MQ でのメッセージの安全性について詳しくは、メッセージの安全性を参照してください。

複数インスタンス・キュー・マネージャーおよび高可用性

MQIPT は、高可用性環境で複数インスタンス・キュー・マネージャーとともに使用することができます。

MQIPT には持続状態がないため、MQIPT を別のシステムにフェイルオーバーすることには利点がありません。 代わりに、さまざまなシステム上で実行される同一の MQIPT 構成ファイルを持つ複数インスタンスの mqipt.conf を使用します。 MQIPT の各インスタンスのアベイラビリティーをモニターし、必要な場合は再始動します (同じシステム上で)。 これにより、接続の経路指定に使用することができる同一の MQIPT インスタンスのセットが提供されます。 IBM MQ が接続を MQIPT に経路指定できるようにし、MQIPT がこれらの接続を宛先キュー・マネージャーに転送できるようにする必要があります。

アウトバウンド IBM MQ チャネルは、以下の例に示すさまざまな方法で、使用可能な MQIPT インスタンスにダイレクトすることができます。

- WebSphere Edge Components 製品の IBM Network Dispatcher などのロード・バランサーまたは高可用性ルーターを使用します。
- コンマ区切りリストを使用して、IBM MQ チャネル定義で複数の接続名を指定します。 これにより IBM MQ は、使用可能な MQIPT インスタンスが見つかるまで、各 MQIPT アドレスへの接続を順番に試行します。

また、MQIPT からの接続を宛先キュー・マネージャーにダイレクトする必要があります。 高可用性構成で IP アドレスが宛先キュー・マネージャーを使用してフェイルオーバーするようになっている場合は、特別な MQIPT 構成は必要ありません。**Destination** 経路プロパティーに宛先 IP アドレスを指定し、フェイルオーバー操作でキュー・マネージャーを使用して IP アドレスを移動できるようにします。

ただし、フェイルオーバー後にキュー・マネージャーの IP アドレスが変更された場合は、MQIPT が接続を正しい宛先に転送できるように調整する必要があります。 これは、以下のいずれかの方法で行うことができます。

- どの IP アドレスとポート番号がアクセス可能かをチェックするルーティング出口を作成し、各接続の経路の宛先を指定変更します。 MQIPT には、いくつかのサンプルのルーティング出口が用意されています。これらをこの目的に合うように調整することができます。
- 高可用性ロード・バランサーを使用して、接続をリダイレクトします。
- キュー・マネージャーが実行されている可能性がある各 IP アドレスとポートに 1 つずつ、複数の MQIPT 経路を定義します。 次に、例えば、アウトバウンド・チャネルの接続名のコンマ区切りリストに経路 IP アドレスとポート番号をすべてリストすることによって、IBM MQ 接続をさまざまな MQIPT 経路にダイレクトします。

また、ネットワーク・パス上にあるすべてのエンドツーエンド・コンポーネントを調整することも重要です。

1. 使用不可のシステムへの接続の試行は、再接続が試行が使用可能な最初の宛先に進めるように即座に失敗する必要があります。

   MQIPT SSL 経路については、使用不可の宛先の場合に接続が即座に失敗するように **SSLClientConnectTimeout** 経路プロパティーを調整します。 IBM MQ チューニング・パラメーターの詳細については、IBM MQ の資料を参照してください。 また、オペレーティング・システム向けの TCP/IP の調整の詳細については、ご使用のオペレーティング・システムの資料を参照してください。 すべての場合において、失敗した接続の試行は、迅速にネットワーク障害 (TCP リセット・パケットなど) を戻すか、または過度の遅延なしにタイムアウトになる必要があります。
2. 障害が発生したシステムへのアクティブな接続は、新規接続を確立できるように即座に切断される必要があります。

   また、接続でアクティブに MQIPT が使用されているときにフェイルオーバーすることの影響を考慮する必要があります。 フェイルオーバー中はネットワーク接続が切断される可能性があります。 クライアント・アプリケーションの場合は、IBM MQ 自動クライアント再接続機能を使用して、切断された接続を再確立することができます。 メッセージ・チャネルの場合は、チャネルが即座に再接続するように、短期再試行間隔を指定することができます。 自動クライアント再接続およびメッセージ・チャネルの再試行構成の詳細については、IBM MQ の資料を参照してください。

V 9.3.5 IBM MQ Console および REST API

IBM MQ Console および REST API を使用して、 IBM MQ を管理し、 HTTP を使用してメッセージング操作を実行することができます。

- IBM MQ Console を使用して、Web ブラウザーから基本的な管理タスクを実行できます。 詳しくは、IBM MQ Console による管理を参照してください。
- administrative REST API を使用して、キュー・マネージャーやキューなどの IBM MQ オブジェクト、Managed File Transfer エージェントおよび転送を管理できます。 詳しくは、REST API による管理を参照してください。
- messaging REST API を使用して、単純な Point-to-Point メッセージングおよびパブリッシュ・メッセージングを実行できます。 詳しくは、 REST API を使用したメッセージングを参照してください。

インストールのオプション

IBM MQ Console および REST API は、mqweb という WebSphere Liberty サーバーで実行されます。 IBM MQ 9.3.5 以降、mqweb サーバーは、 IBM MQ インストール済み環境のオプション・コンポーネントとしてインストールすることも、スタンドアロンの IBM MQ Web Server インストール済み環境としてインストールすることもできます。

Linux V 9.3.5 スタンドアロン IBM MQ Web Server インストール済み環境

IBM MQ 9.3.5 以降、mqweb サーバーは IBM MQ Web Server のスタンドアロン・インストール済み環境で実行できます。 スタンドアロンの IBM MQ Web Server インストール済み環境では、IBM MQ インストール済み環境とは別のシステムに mqweb サーバーをインストールして実行することができます。 スタンドアロン IBM MQ Web Server をインストールすると、mqweb サーバーを実行するために選択するシステムとシステムの数に関して柔軟性が向上します。 必要に応じて、mqweb サーバーの複数のインスタンスを異なるマシン上で実行して、必要なスケーラビリティーと可用性を提供することができます。

IBM MQ ライセンスを購入した場合は、スタンドアロン IBM MQ Web Server に必要な数だけコピーをインストールできます。 IBM MQ Web Server インストール済み環境は、購入した IBM MQ ライセンスを消費するものとしてカウントされません。 IBM MQ のライセンス交付について詳しくは、IBM MQ のライセンス情報を参照してください。

スタンドアロン IBM MQ Web Server インストール済み環境では、以下の制約事項が適用されます。

- IBM MQ Console は、リモート・キュー・マネージャーの管理にのみ使用できます。
- messaging REST API は、リモート・キュー・マネージャーでのみ使用できます。
- administrative REST API は使用できません。

スタンドアロン IBM MQ Web Server は、 Linux プラットフォームでのみサポートされます。

スタンドアロン IBM MQ Web Server のインストールについて詳しくは、 スタンドアロン IBM MQ Web Server のインストールを参照してください。

IBM MQ インストール済み環境のオプション・コンポーネント

IBM MQ Console および REST API コンポーネントを IBM MQ インストールの一部としてインストールすることを選択できます。

IBM MQ Console および REST API のすべての機能は、mqweb サーバーが IBM MQ インストール済み環境で実行されている場合に使用可能です。

- IBM MQ Console を使用して、ローカルおよびリモートのキュー・マネージャーを管理できます。
- messaging REST API は、ローカルおよびリモートのキュー・マネージャーで使用できます。
- administrative REST API を使用して、ローカルおよびリモートのキュー・マネージャーを管理できます。

IBM MQ Console および REST API コンポーネントを使用するには、 IBM MQ インストールの一部として以下のコンポーネントをインストールします。

- AIX で、`mqm.web.rte` ファイルセットをインストールします。
- IBM i の場合は、WEB コンポーネントをインストールします。
- Linux で、`MQSeriesWeb` コンポーネントをインストールします。
- Windows で、`Web Administration` フィールドをインストールします。
- z/OS で、`IBM MQ for z/OS UNIX System Services Web Components` フィールドをインストールします。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

IBM 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。 本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510
東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
日本アイ・ビー・エム株式会社
法務・知的財産
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
〒 103-8510
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 INTERNATIONAL BUSINESS MACHINES CORPORATION は、法律上の瑕疵担保責任、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 "" 国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものでは ありません。 それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他の ライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で 決定されたものです。 そのため、他の操作環境で得られた結果は、異なる可能性があります。 一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。 さらに、一部の測定値が、推定値である可能性があります。 実際の結果は、異なる可能性があります。 お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、 もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。 したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、 名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報 (提供されている場合) は、このプログラムで使用するアプリケーション・ソフトウェアの作成を支援することを目的としています。

本書には、プログラムを作成するユーザーが WebSphere MQ のサービスを使用するためのプログラミング・インターフェースに関する情報が記載されています。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。 診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

重要 : この診断、修正、およびチューニング情報は、変更される可能性があるため、プログラミング・インターフェースとして使用しないでください。

商標

IBM、 IBM ロゴ、 ibm.com®は、世界の多くの国で登録された IBM Corporation の商標です。 現時点での IBM の商標リストについては、 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml をご覧ください。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds 氏の米国およびその他の国における登録商標です。

この製品には、 Eclipse Project (https://www.eclipse.org/) により開発されたソフトウェアが含まれています。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

部品番号:

(1P) P/N: