7.5

IBM WebSphere MQ の計画

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、163 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® WebSphere® MQ バージョン 7 リリース 5、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。

© Copyright International Business Machines Corporation 2007 年, 2024.

目次

計画

IBM WebSphere MQ 環境の計画時には、構成する IBM WebSphere MQ アーキテクチャー、リソース要件、およびロギングとバックアップ機能の必要性について考慮する必要があります。 このトピックにあるリンクを使用して、IBM WebSphere MQ が実行される環境を計画します。

IBM WebSphere MQ 環境を計画する前に、基本的な IBM WebSphere MQ の概念について習得しておく必要があります。 「技術概説」にあるトピックを参照してください。

関連概念
可用性、リカバリー、および再始動
関連タスク
移行
インストール
構成
WebSphere MQ の管理
メッセージの消失を確実に回避する (ログ)

IBM MQ および IBM MQ Appliance の GDPR 対応に関するオンプレミス考慮事項

対象 PID:

- 5724-H72 IBM MQ
- 5655-AV9 IBM MQ Advanced for z/OS®
- 5655-AV1 IBM MQ Advanced for z/OS、Value Unit Edition
- 5655-AM9 IBM MQ Advanced Message Security for z/OS
- 5725-Z09 IBM MQ Appliance M2001
- 5725-S14 IBM MQ Appliance M2000
- 5655-MQ9 IBM MQ for z/OS
- 5655-VU9 IBM MQ for z/OS バリュー・ユニット・エディション
- 5639-L92 IBM MQ Internet Pass-Thru
- 5655-MF9 IBM MQ Managed File Transfer for z/OS
- 5655-ADV IBM WebSphere MQ Advanced for z/OS
- 5655-AMS IBM WebSphere MQ Advanced Message Security for z/OS
- 5724-R10 IBM WebSphere MQ File Transfer Edition for Multiplatforms
- 5724-A39 IBM WebSphere MQ for HP NonStop サーバー
- 5724-A38 IBM WebSphere MQ for HP OpenVMS
- 5655-W97 IBM WebSphere MQ for z/OS
- 5655-VU8 IBM WebSphere MQ for z/OS バリュー・ユニット・エディション
- 5655-VUE IBM WebSphere MQ for z/OS バリュー・ユニット・エディション
- 5725-C79 IBM WebSphere MQ Hypervisor Edition for Red Hat Enterprise Linux® for x86
- 5725-F22 IBM WebSphere MQ AIX®
- 5655-MFT IBM WebSphere MQ Managed File Transfer for z/OS

© Copyright IBM Corp. 2007 年, 2024

注意:

この資料は、お客様の GDPR 対応の準備を支援することを目的としています。 本資料では、組織が GDPR に対応するために考慮する必要がある、構成可能な IBM MQ の機能と、この製品の使用のさまざまな側面について説明します。 お客様が機能を選択および構成できる方法が多岐にわたっており、また製品を単体で、あるいはサード・パーティーのアプリケーションおよびシステムとともにさまざまな方法で使用できるため、この情報はすべてを網羅したリストではありません。

お客様は、欧州連合 (EU) の一般データ保護規則を含む、さまざまな法律および規制への準拠を保証する責任があります。 お客様のビジネスに影響を及ぼす可能性のある関連法令の特定およびそれらの解釈、ならびにかかる関連法令を遵守するためにお客様が講ずるべき必要措置に関する助言は、お客様の責任により適格な弁護士から得るものとします。

本書に記載の製品、サービス、および他の機能が、すべてのお客様の状況に適しているとは限らず、使用する際に制約を受ける場合があります。 IBM は、法律、会計または監査に関する助言を提供することはしませんし、IBM のサービスまたは製品が、お客様のあらゆる法令遵守の裏付けとなる表明または保証もいたしません。

目次

GDPR

一般データ保護規則 (GDPR) は、欧州連合 (EU) によって採択され、2018 年 5 月 25 日から適用されています。

GDPR が重要である理由

GDPR により、個人に関する個人データの処理に関する、より強固なデータ保護規制の枠組みが確立されます。 GDPR により以下のことがもたらされます。

- 個人の新たな権利および強化された権利
- 個人データの定義の拡大
- データを処理する人の新たな義務
- 不遵守に対して高額の制裁金の可能性
- データ漏えいの届け出の義務付け

GDPR について詳しくは、以下のサイトを参照してください。

- EU GDPR 情報ポータル
- ibm.com/GDPR の Web サイト

製品の構成 - GDPR 対応のための考慮事項

以下の各セクションでは、お客様の組織において GDPR 対応の準備を行うための IBM MQ の構成に関する考慮事項を紹介します。

データ・ライフサイクル

IBM MQ は、アプリケーション間でアプリケーション提供のデータを非同期に交換できるようにするための、トランザクション型のメッセージ指向ミドルウェア製品です。 IBM MQ では、アプリケーションを接続するために、さまざまなメッセージング API、プロトコル、およびブリッジをサポートしています。 そのため、IBM MQ は様々な形態のデータの交換に使用される可能性があり、その一部が GDPR の対象になる可能性があります。 IBM MQ とデータ交換を行う可能性のあるサード・パーティー製品もいくつかあります。 その一部は IBM 所有ですが、その他の多くは他のテクノロジー・サプライヤーから提供されている製品です。 Software Product Compatibility Reports Web サイトに、関連するソフトウェアのリストが記載されています。 サード・パーティー製品の GDPR 対応に関する考慮事項については、その製品の資料を調べてください。 IBM MQ 管理者は、キュー、トピック、およびサブスクリプションの定義によって、IBM MQ がそれを通過するデータと相互作用する方法を制御します。

IBM MQ を流れるデータのタイプにはどのようなものがありますか?

IBM MQ は、アプリケーション・データの非同期メッセージング・サービスを提供するため、アプリケーションのデプロイメントによってユースケースが異なり、この問いに対する 1 つの明確な答えはありません。アプリケーション・メッセージ・データは、キュー・ファイル (z/OS のページ・セットまたはカップリング・ファシリティー)、ログ、およびアーカイブに保持されています。またメッセージ自体に GDPR によって管理されるデータが含まれている場合があります。 アプリケーション提供のメッセージ・データは、エラー・ログ、トレース・ファイル、および FFST など、問題判別のために収集されたファイルにも含まれている可能性もあります。 z/OS では、アプリケーション提供のメッセージ・データは、アドレス・スペースやカップリング・ファシリティーのダンプにも含まれる可能性があります。

IBM MQ を使用して交換される可能性のある代表的な個人データの例として、以下のようなものがあります。

- お客様の雇用者の個人データ (例えば、IBM MQ を使用してお客様の給与計算システムまたは HR システムを接続する場合があります)
- お客様自身の顧客の個人データ (例えば、お客様が IBM MQ を使用して顧客に関係するデータをアプリケーション間で交換する場合があります。CRM システムで見込み客情報を取得したりデータを格納したりする場合などです)。
- お客様自身の顧客の機密性の高い個人データ (例えば、個人データの交換を必要とする業界特有の状況で IBM MQ を使用する場合があります。臨床アプリケーションを統合するときの HL7 ベースの医療記録などです)。

アプリケーション提供のメッセージ・データの他にも、IBM MQ は以下のタイプのデータを処理します。

- 認証資格情報 (ユーザー名とパスワード、API 鍵など)
- 技術的に識別可能な個人情報 (デバイス ID、使用ベースの ID、IP アドレスなど - 個人にリンクされている場合)

IBM とのオンラインによる連絡のために使用される個人データ

IBM MQ のお客様は、IBM MQ の問題について IBM に連絡するために、オンラインによるさまざまな方法でコメント/フィードバック/要求を送信することができ、主に以下のようなものがあります。

- IBM Developer の IBM MQ 領域内のページにあるパブリック・コメント領域
- IBM MQ IBM Documentation の製品情報のページ上にあるパブリック・コメント領域
- IBM サポート・フォーラム内のパブリック・コメント
- IBM Developer の IBM RFE コミュニティー内のパブリック・コメント

通常、クライアント名と E メール・アドレスのみが使用され、コンタクトの対象となる個人の応答を使用可能にし、個人データの使用は IBM オンライン・プライバシー・ステートメントに準拠します。

データ収集

IBM MQ を使用して個人データを収集できます。 IBM MQ の使用および GDPR の要求を満たす必要性を評価する場合、ご使用の環境で IBM を通過する個人データのタイプを考慮する必要があります。 次のような側面を考慮することができます。

- データはどのようにキュー・マネージャーに到着するか。 (どのプロトコルか。 データは暗号化されているか。 データは署名されているか。)
- データはどのようにキュー・マネージャーから送信されるか。 (どのプロトコルか。 データは暗号化されているか。 データは署名されているか。)
- データはキュー・マネージャーを通過するときにどのように格納されるか。 (メッセージが非持続の場合でも、メッセージング・アプリケーションはメッセージ・データをステートフル・メディアに書き込む可能性がある。 この製品を通過するアプリケーション・メッセージ・データの特定の側面が、メッセージングのフィーチャーによってどのように公開される可能性があるかを認識しているか?)
- IBM MQ がサード・パーティー・アプリケーションにアクセスするために必要なときに、資格情報をどのように収集して保管するか。

IBM MQ は、LDAP など、認証を必要とする他のシステムおよびサービスと通信する必要がある場合があります。 必要なときに、IBM MQ はそのような通信で利用するために認証データ (ユーザー ID、パスワード) を構成して保管します。 可能な限り、IBM MQ 認証に個人の資格情報を使用しないようにする必要があります。 認証データ用に使用されるストレージの保護を検討してください。 (下記の「データ・ストレージ」を参照)

データ・ストレージ

メッセージ・データがキュー・マネージャーを通過するとき、IBM MQ はそのデータ (おそらくその複数のコピー) をステートフル・メディアに直接保存します。 IBM MQ ユーザーは、メッセージ・データが保存状態である間はそれを保護するよう検討してください。

以下の項目は、IBM MQ がアプリケーション提供のデータを保持する領域を取り上げています。これらは、GDPR への準拠を確実にする場合にユーザーが考慮したいと考える項目です。

- アプリケーション・メッセージ・キュー:

  IBM MQ は、アプリケーション間での非同期データ交換を可能にするメッセージ・キューを提供します。キューに格納された非持続メッセージおよび持続メッセージは、ステートフル・メディアに書き込まれます。
- ファイル転送エージェント・キュー:

  IBM MQ はメッセージ・キューを使用してファイル・データの信頼性のある転送を調整します。個人データと転送記録を入れたファイルは、これらのキューに格納されます。
- 伝送キュー

  メッセージをキュー・マネージャー間で確実に転送するために、メッセージは一時的に伝送キューに格納されます。
- 送達不能キュー:

  メッセージは、宛先キューに書き込むことができずに送達不能キューに格納されることがあります (構成済みの送達不能キューがキュー・マネージャーにある場合)。
- バックアウト・キュー:

  JMS および XMS のメッセージング・インターフェースは、他の有効なメッセージを処理できるように、いくつかのバックアウトが発生すると有害メッセージをバックアウト・キューに移動できる機能を提供します。
- AMS エラー・キュー:

  IBM MQ Advanced Message Security は、セキュリティー・ポリシーに準拠していないメッセージを SYSTEM.PROTECTION.ERROR.QUEUE エラー・キューは、送達不能キューイングと同様の方法で作成されます。
- 保存パブリケーション:

IBM MQ は、サブスクライブ側のアプリケーションが前のパブリケーションを再呼び出しできるようにするために、保存パブリケーション・フィーチャーを提供します。

詳しくは、以下を参照してください。

- ロギング: メッセージが失われないようにするための機能
- MFT エージェント・キュー設定
- 伝送キューの定義
- 送達不能キューの使用
- IBM MQ classes for JMS での有害メッセージの処理
- AMS エラー処理
- 保存パブリケーション

以下の項目は、IBM MQ がアプリケーション提供のデータを間接的に保持する領域を取り上げています。これらも、GDPR への準拠を保証するためにユーザーが考慮できる項目です。

- 経路トレース・メッセージング:

  IBM MQ は、アプリケーション間でメッセージが取る経路を記録する経路トレース機能を提供します。生成されるイベント・メッセージには、IP アドレスなどの技術的に識別可能な個人情報が含まれる場合があります。
- アプリケーション・アクティビティー・トレース:

  IBM MQ は、アプリケーションとチャネルのメッセージング API アクティビティーを記録するアプリケーション・アクティビティー・トレースを提供します。アプリケーション・アクティビティー・トレースでは、アプリケーション提供のメッセージ・データの内容をイベント・メッセージに記録することができます。
- サービス・トレース:

  IBM MQ は、メッセージ・データが流れる内部コード・パスを記録するサービス・トレース・フィーチャーを提供します。 これらのフィーチャーの一部として、IBM MQ では、アプリケーション提供のメッセージ・データの内容を、ディスクに保管されているトレース・ファイルに記録できます。
- キュー・マネージャー・イベント:

  IBM MQ は、権限イベント、コマンド・イベント、構成イベントなどの、個人データを含む可能性のあるイベント・メッセージを生成することがあります。

詳しくは、以下を参照してください。

- 経路トレース・メッセージング
- トレースの使用法
- イベント・モニター
- キュー・マネージャー・イベント

アプリケーション提供のメッセージ・データのコピーへのアクセスを保護するには、以下のアクションを考慮してください。

- ファイル・システム内の IBM MQ データへの特権ユーザー・アクセスを制限します。例えば、UNIX プラットフォームでの 'mqm' グループのユーザー・メンバーシップを制限します。
- 専用キューおよびアクセス制御によって、IBM MQ データへのアプリケーションのアクセスを制限します。 必要に応じて、アプリケーション間でのキューなどのリソースの不要な共有を避け、キューおよびトピック・リソースに対してきめ細かくアクセス制御を設定します。
- IBM MQ Advanced Message Security を使用して、メッセージ・データのエンドツーエンドの署名または暗号化 (あるいはその両方) を行います。
- ファイル・レベルまたはボリューム・レベルの暗号化を使用して、トレース・ログの格納に使用するディレクトリーのコンテンツを保護します。
- サービス・トレースを IBM にアップロードした後、個人データが入っている可能性があるコンテンツについて懸念がある場合は、サービス・トレース・ファイルおよび FFST データを削除できます。

詳しくは、以下を参照してください。

- 特権ユーザー
- ファイル・システム・サポートの計画 (Multiplatforms)

IBM MQ 管理者は、資格情報 (ユーザー名とパスワード、API キーなど) を使用してキュー・マネージャーを構成できます。 3rd パーティー・サービス (LDAP、 IBM Cloud® Product Insights、 Salesforce など) このデータは、通常、ファイル・システム権限を使用して保護されたキュー・マネージャーのデータ・ディレクトリーに格納されます。

IBM MQ キュー・マネージャーが作成されるときには、IBM MQ が構成ファイルを読み取って資格情報を使用してこれらのシステムに接続できるように、グループ・ベースのアクセス制御を使用してデータ・ディレクトリーがセットアップされます。 IBM MQ 管理者は特権ユーザーと見なされ、このグループのメンバーであるため、これらのファイルへの読み取り権限があります。 一部のファイルは難読化されていますが、暗号化はされていません。 そのため、資格情報へのアクセスを完全に保護するには、以下のアクションを考慮する必要があります。

- IBM MQ データへの特権ユーザーのアクセスを制限します。例えば、UNIX プラットフォームでの「mqm」グループのメンバーシップを制限します。
- ファイル・レベルまたはボリューム・レベルの暗号化を使用して、キュー・マネージャー・データ・ディレクトリーのコンテンツを保護します。
- 実動構成ディレクトリーのバックアップを暗号化し、適切なアクセス制御を設定して保管します。
- セキュリティー・イベント、コマンド・イベント、および構成イベントでの、認証失敗、アクセス制御、および構成変更に対して監査証跡を提供することを検討してください。

詳しくは、以下を参照してください。

- IBM MQ の保護

データ・アクセス

IBM MQ キュー・マネージャー・データは、以下の製品インターフェースからアクセスできます。 リモート接続によってアクセスするように設計されているものと、ローカル接続によってアクセスするように設計されているものがあります。

- IBM MQ コンソール [リモートのみ]
- IBM MQ REST API [リモートのみ]
- MQI [ローカルとリモート]
- JMS [ローカルとリモート]
- XMS [ローカルとリモート]
- IBM MQ Telemetry (MQTT) [リモートのみ]
- IBM MQ Light (AMQP) [リモートのみ]
- IBM MQ IMS ブリッジ [ローカルのみ]
- IBM MQ CICS ブリッジ [ローカルのみ]
- IBM MQ bridge for HTTP [リモートのみ]
- IBM MQ MFT プロトコル・ブリッジ [リモートのみ]
- IBM MQ Connect:Direct ブリッジ [リモートのみ]
- IBM MQ Bridge to Salesforce [リモートのみ]
- IBM MQ Bridge to Blockchain [リモートのみ]
- IBM MQ MQAI [ローカルおよびリモート]
- IBM MQ PCF コマンド [ローカルおよびリモート]
- IBM MQ MQSC コマンド [ローカルおよびリモート]
- IBM MQ エクスプローラー [ローカルおよびリモート]

インターフェースは、ユーザーが IBM MQ キュー・マネージャーおよびそこに格納されたメッセージに変更を加えられるように設計されています。 管理操作およびメッセージング操作は、要求が行われたときに、関与する以下の 3 つのステージが存在するように保護されます。

- 認証
- ロール・マッピング
- 認証

認証:

メッセージ操作または管理操作がローカル接続から要求された場合、この接続のソースは、同じシステム上の実行中のプロセスです。 プロセスを実行するユーザーは、オペレーティング・システムが提供する認証ステップを通過している必要があります。 接続を行ったプロセスの所有者のユーザー名が ID として表明されます。 これは例えば、アプリケーションを開始したシェルを実行しているユーザーの名前などです。 ローカル接続に使用できる認証の形式として、次のものが挙げられます。

1. 表明されたユーザー名 (ローカル OS)
2. オプションのユーザー名とパスワード (OS、LDAP、またはカスタムのサード・パーティー・リポジトリー)

管理アクションがリモート接続から要求された場合、IBM MQ との通信はネットワーク・インターフェースを介して行われます。 ネットワーク接続を介した認証では、以下の形式の ID を提示できます。

1. 表明されたユーザー名 (リモート OS 由来のもの)
2. ユーザー名とパスワード (OS、LDAP、またはカスタムのサード・パーティー・リポジトリー)
3. ソース・ネットワーク・アドレス (IP アドレスなど)
4. X.509 デジタル証明書 (相互 SSL/TLS 認証)
5. セキュリティー・トークン (LTPA2 トークンなど)
6. その他のカスタム・セキュリティー (サード・パーティーの出口が提供する機能)

ロール・マッピング:

ロール・マッピング・ステージでは、認証ステージで提供された資格情報を代替ユーザー ID にマップできます。 マップされたユーザー ID が処理を許可された場合 (管理ユーザーがチャネル認証ルールによってブロックされる場合もあります)、マップされたユーザー ID は、IBM MQ リソースに対してアクティビティーを許可するときに最終ステージに持ち越されます。

authorization:

IBM MQ では、キュー、トピック、その他のキュー・マネージャー・オブジェクトなどのさまざまなメッセージング・リソースに対して、さまざまなユーザーにさまざまな権限を付与することができます。

ロギング・アクティビティー:

IBM MQ の一部のユーザーは、MQ リソースへのアクセスの監査レコードを作成する必要がある場合があります。 望ましい監査ログの例としては、変更を要求したユーザーに加えて変更に関する情報を記載した構成変更が考えられます。

この要件を実装するために以下の情報ソースを利用できます。

1. IBM MQ キュー・マネージャーは、admin コマンドが正常に実行されたときにコマンド・イベントを生成するように構成できます。
2. IBM MQ キュー・マネージャーは、キュー・マネージャー・リソースが作成、変更、または削除されたときに構成イベントを生成するように構成できます。
3. IBM MQ キュー・マネージャーは、リソースの許可検査が不合格になったときに権限イベントを生成するように構成できます。
4. 許可検査が不合格になったことを示すエラー・メッセージは、キュー・マネージャーのエラー・ログに書き込まれます。
5. IBM MQ Web コンソールは、認証、許可検査が不合格になったとき、またはキュー・マネージャーが作成、開始、停止、または削除されたときに、監査メッセージをログに書き込みます。

このようなソリューションを検討する場合、IBM MQ ユーザーは、以下の点について考慮する必要があります。

- イベント・メッセージは非持続的なので、キュー・マネージャーが再始動すると、情報は失われます。いずれのイベント・モニターも、入手可能なあらゆるメッセージを常にコンシュームしてその内容を永続メディアに転送するように構成する必要があります。
- IBM MQ 特権ユーザーは、イベントの無効化、ログのクリア、またはキュー・マネージャーの削除を行うために十分な特権を持っています。

IBM MQ データへのアクセスの保護、および監査証跡の提供について詳しくは、以下のトピックを参照してください。

- IBM MQ セキュリティー・メカニズム
- 構成イベント
- コマンド・イベント
- エラー・ログ

データ処理

公開鍵インフラストラクチャー (PKI) を使用した暗号化:

接続が TLS を使用するように指定することで、IBM MQ へのネットワーク接続を保護できます。TLS は、接続の開始側の相互認証も提供できます。

トランスポート・メカニズムによって提供される PKI セキュリティー機能を使用することが、IBM MQ でのデータ処理を保護するための最初のステップとなります。 しかし、追加のセキュリティー・フィーチャーを有効にしないと、コンシューム側のアプリケーションの動作は、メッセージの発信元や転送中に変更されたかどうかを検証せずに、配信されたメッセージをすべて処理するだけになってしまいます。

Advanced Message Security (AMS) 機能を使用するようにライセンス交付されている IBM MQ ユーザーは、セキュリティー・ポリシーを定義および構成することによって、メッセージに含まれる個人データをアプリケーションで処理する方法を制御できます。 セキュリティー・ポリシーを使用すると、アプリケーション間のメッセージ・データにデジタル署名または暗号化 (あるいはその両方) を適用できます。

メッセージが本物であることを保証するために、メッセージをコンシュームするときにセキュリティー・ポリシーを使用してデジタル署名を要求および検証することができます。 AMS 暗号化は、読み取り可能な形式のメッセージ・データを、エンコード・バージョンに変換する方式を提供します。このエンコード・バージョンは、別のアプリケーションが意図されたメッセージ受信者であり、かつ正しい暗号化解除鍵にアクセスできる場合にのみ、このアプリケーションでデコードできます。

SSL および証明書を使用してネットワーク接続を保護する方法について詳しくは、IBM MQ V9 製品資料の以下のトピックを参照してください。

- IBM MQ の TLS セキュリティーの構成
- AMS の概要

データ削除

IBM MQ には、この製品に提供されたデータを削除するためのコマンドおよびユーザー・インターフェース・アクションが用意されています。 これによって、IBM MQ のユーザーは、特定の個人に関連するデータを削除することが求められる場合に、そのことを行うのが容易になります。

- GDPR クライアント・データの削除に準拠するために考慮する必要がある IBM MQ の動作の領域
  - 次のようにしてアプリケーション・キューに保管されたメッセージ・データを削除する。
    - メッセージング API またはツールを使用して、またはメッセージの有効期限を使用して、個々のメッセージを除去する。
    - 対象メッセージを、非持続メッセージ・クラスが正常の状態であるキューに保持された非持続メッセージとして指定し、キュー・マネージャーを再始動する。
    - 管理者がキューをクリアする。

    - キューを削除する。
  - 次のようにしてトピックに保管された保存パブリケーション・データを削除する。
    - メッセージを非持続メッセージとして指定し、キュー・マネージャーを再始動する。
    - 保存データを新規データに置き換えるか、メッセージ有効期限を使用する。
    - 管理者がトピック・ストリングをクリアする。
  - キュー・マネージャー全体を削除することによって、キュー・マネージャーに保管されたデータを削除する。
  - トレース・ディレクトリー内のファイルを削除することによって、サービス・トレース・コマンドによって保管されたデータを削除する。
  - エラー・ディレクトリー内のファイルを削除することによって保管された FFST データを削除する。
  - アドレス・スペースとカップリング・ファシリティー・ダンプ (z/OS 上) を削除する。
  - アーカイブ、バックアップ、またはそのようなデータのその他のコピーを削除する。
- GDPR アカウント・データの削除に準拠するために考慮する必要がある IBM MQ の動作の領域
  - キュー・マネージャーとサード・パーティー・サービスに接続するために IBM MQ に保管されたアカウント・データと設定を削除するために以下を削除する (アーカイブ、バックアップ、それらの複製コピーを含む)。
    - 資格情報を格納するキュー・マネージャー認証情報オブジェクト。
    - ユーザー ID を参照するキュー・マネージャー権限レコード。
    - 特定の IP アドレス、証明書 DN、またはユーザー ID をマップまたはブロックするキュー・マネージャー・チャネル認証規則。
    - キュー・マネージャーおよびファイル・サーバーでの認証用に、IBM MQ Managed File Transfer エージェント、ロガー、および MQ Explorer MFT プラグインが使用する資格情報ファイル。
    - SSL/TLS 接続または IBM MQ Advanced Message Security (AMS) で使用する可能性がある、個人を表すかその個人についての情報を含んだ、鍵ストア由来の X.509 デジタル証明書。
    - IBM MQ Appliance の個人ユーザー・アカウント (システム・ログ・ファイル内のそれらのアカウントへの参照を含む)。
    - IBM MQ エクスプローラーのワークスペース・メタデータおよび Eclipse 設定。
    - IBM MQ エクスプローラーのパスワード・ストア。「パスワード設定」で指定します。
    - IBM MQ コンソールおよび mqweb サーバーの構成ファイル。
    - Salesforce 接続データ構成ファイル。
    - ブロックチェーン接続データ構成ファイル。
    - IBM Cloud qm.ini および APIKeyFile の ReportingService スタンザにある Product Insights 接続データ。

詳しくは、以下を参照してください。

- IBM MQ Bridge to Salesforce の構成
- ブロックチェーンで使用するための IBM MQ の構成
- MFT と IBM MQ の接続認証
- ProtocolBridgeCredentials.xml ファイルを使用してファイル・サーバーの資格情報をマップする
- IBM MQ Console ユーザーおよび役割の構成

データのモニタリング

IBM MQ は、ユーザーがアプリケーションとキュー・マネージャーの実行状態をよりよく理解するために活用できるさまざまなモニター・フィーチャーを提供します。

さらに IBM MQ は、キュー・マネージャーのエラー・ログの管理に役立つさまざまなフィーチャーも提供します。

詳しくは、以下を参照してください。

- IBM MQ ネットワークのモニター
- 診断メッセージ・サービス
- QMErrorLog サービス

IBM MQ は、 IBM MQ ユーザーがキュー・マネージャーの始動情報と使用情報を表示できるように、ユーザーが IBM Cloud Product Insights サービスに情報をパブリッシュできるようにする機能を提供します。

詳しくは、以下を参照してください。

- IBM Cloud の IBM Cloud Product Insights サービスを使用するための IBM MQ の構成

個人データの使用を制限するための機能

本書に要約されている機能を使用すると、IBM MQ によって、エンド・ユーザーが自分の個人データの使用を制限できるようになります。

IBM MQ メッセージ・キューは、データベースとは異なるので、永続データ・ストアとしては使用しないでください。これは特に、GDPR の対象となるアプリケーション・データを処理する場合に当てはまります。

検索照会によってデータを検出できるデータベースとは異なり、メッセージのキュー、メッセージ、および相関 ID が分かっていないと、メッセージ・データを見つけるのが困難な場合があります。

個人データが含まれているメッセージを容易に特定して見つけることができる場合は、標準の IBM MQ メッセージング・フィーチャーを使用してメッセージ・データにアクセスしたり変更したりできます。

ファイル処理

1. IBM MQ Managed File Transfer は、転送されたファイルに対してマルウェア・スキャンを実行しません。 ファイルは現状のまま転送され、整合性検査が実行されて、転送中にファイル・データが変更されていないことが確認されます。 転送状況のパブリケーションの一部として、ソースと宛先のチェックサムがパブリッシュされます。 エンド・ユーザーは、MFT がファイルをリモート・エンドポイントに転送する前、および MFT がファイルをリモート・エンドポイントに送信した後に、環境に応じてマルウェア・スキャンを実装することをお勧めします。
2. IBM MQ Managed File Transfer は、MIME タイプやファイル拡張子に基づくアクションを実行しません。MFT はファイルを読み取り、入力ファイルから読み取られたとおり正確にバイトを転送します。

IBM WebSphere MQ アーキテクチャーの設計

Point-to-Point メッセージング・スタイルおよびパブリッシュ/サブスクライブ・メッセージング・スタイル用に IBM WebSphere MQ がサポートするさまざまなアーキテクチャーについて説明します。

IBM WebSphere MQ アーキテクチャーを計画する前に、基本的な IBM WebSphere MQ の概念について習得しておく必要があります。 「IBM WebSphere MQ 技術概説」にあるトピックを参照してください。

IBM WebSphere MQ アーキテクチャーは、単一のキュー・マネージャーを使用した単純なアーキテクチャーから、より複雑な相互接続キュー・マネージャーのネットワークまで多岐にわたります。 複数のキュー・マネージャーの接続には、分散キューイング技法が使用されます。 単一のキュー・マネージャーのアーキテクチャーおよび複数のキュー・マネージャーのアーキテクチャーの計画について詳しくは、以下のトピックを参照してください。

- 15 ページの『1 つのキュー・マネージャーに基づくアーキテクチャー』
- 16 ページの『複数のキュー・マネージャーに基づくアーキテクチャー』
- 17 ページの『ネットワークおよびネットワーク計画』
- WebSphere MQ 分散メッセージング技法

論理的に関連する複数のキュー・マネージャーが必要であり、データおよびアプリケーションを共用する必要がある場合、それらのキュー・マネージャーをクラスター内でグループ化できます。 クラスターを使用すると、キュー・マネージャーは、追加のチャネル定義やリモート・キュー定義をセットアップせずに

相互に通信できるため、構成および管理が単純化されます。 クラスターの使用法について詳しくは、クラスターが機能する仕組みを参照してください。

関連概念

5 ページの『計画』
IBM WebSphere MQ 環境の計画時には、構成する IBM WebSphere MQ アーキテクチャー、リソース要件、およびロギングとバックアップ機能の必要性について考慮する必要があります。 このトピックにあるリンクを使用して、IBM WebSphere MQ が実行される環境を計画します。

関連タスク

構成

1 つのキュー・マネージャーに基づくアーキテクチャー

IBM WebSphere MQ の最もシンプルなアーキテクチャーは、キュー・マネージャーを 1 つだけ構成して使用するというものです。

IBM WebSphere MQ のアーキテクチャーを計画する前に、IBM WebSphere MQ の基本的な概念をよく理解することが必要です。IBM WebSphere MQ の概要を参照してください。

キュー・マネージャーを 1 つだけ使用したアーキテクチャーとしては、以下の各セクションで取り上げるようなアーキテクチャーが考えられます。

- 15 ページの『1 つのキュー・マネージャーで複数のローカル・アプリケーションがサービスにアクセスするアーキテクチャー』
- 15 ページの『1 つのキュー・マネージャーで複数のリモート・アプリケーションがクライアントとしてサービスにアクセスするアーキテクチャー』
- 15 ページの『1 つのキュー・マネージャーでパブリッシュ/サブスクライブを構成するアーキテクチャー』

1 つのキュー・マネージャーで複数のローカル・アプリケーションがサービスにアクセスするアーキテクチャー

1 つのキュー・マネージャーに基づく最初のアーキテクチャーは、サービスにアクセスするアプリケーションとサービスを提供するアプリケーションを同じシステムで実行するというものです。 IBM WebSphere MQ キュー・マネージャーは、サービスを要求するアプリケーションとサービスを提供するアプリケーションの間の非同期相互通信を提供します。 この場合は、いずれかのアプリケーションが長期にわたってオフラインになっても、アプリケーション間の通信を継続できます。

1 つのキュー・マネージャーで複数のリモート・アプリケーションがクライアントとしてサービスにアクセスするアーキテクチャー

1 つのキュー・マネージャーに基づく 2 番目のアーキテクチャーは、サービスを提供するアプリケーションからアプリケーションをリモート実行するというものです。 つまり、サービスが存在するシステムとは異なるシステムでリモート・アプリケーションを実行します。 それらのアプリケーションは、クライアントとして 1 つのキュー・マネージャーに接続します。 この場合は、1 つのキュー・マネージャーで複数のシステムにサービスに対するアクセスを提供することになります。

このアーキテクチャーの場合は、アプリケーションの操作のためにネットワーク接続を有効にしておく必要がある、という制約があります。 ネットワーク接続を経由したアプリケーションとキュー・マネージャーの対話は、同期モードになります。

1 つのキュー・マネージャーでパブリッシュ/サブスクライブを構成するアーキテクチャー

1 つのキュー・マネージャーを使用するさらに別のアーキテクチャーは、パブリッシュ/サブスクライブ構成を使用するというものです。 パブリッシュ/サブスクライブ・メッセージングでは、情報のプロバイダーとコンシューマーを分離できます。 これまでに取り上げたアーキテクチャーの Point-to-Point スタイルのメッセージングとは、この点が異なります。前述のアーキテクチャーでは、アプリケーションにおいて、

ターゲット・アプリケーション (メッセージの書き込み先のキュー名など) についての情報が必要になります。 IBM WebSphere MQ のパブリッシュ/サブスクライブ構成を使用する場合、送信側のアプリケーションは、情報のサブジェクトに基づいて指定されたトピックにメッセージをパブリッシュします。 その後、IBM WebSphere MQ がメッセージの配布を処理します。 つまり、サブスクリプションによってそのサブジェクトを興味の対象として登録しているアプリケーションにメッセージを配布します。 受信側のアプリケーションも、メッセージを受信するために、そのソースについて何かの情報を知っておく必要はありません。 パブリッシュ/サブスクライブ・メッセージングの詳細については、WebSphere MQ パブリッシュ/サブスクライブ・メッセージングの紹介を参照してください。 キュー・マネージャーを 1 つだけ使用したパブリッシュ/サブスクライブ・メッセージングの例については、単一キュー・マネージャー・パブリッシュ/サブスクライブ構成の例を参照してください。

関連概念

14 ページの『IBM WebSphere MQ アーキテクチャーの設計』
Point-to-Point メッセージング・スタイルおよびパブリッシュ/サブスクライブ・メッセージング・スタイル用に IBM WebSphere MQ がサポートするさまざまなアーキテクチャーについて説明します。

関連情報

WebSphere MQ の概要

キュー・マネージャーの作成および管理

複数のキュー・マネージャーに基づくアーキテクチャー

分散メッセージ・キューイングの手法を使用して、複数のキュー・マネージャーの構成と使用を含む IBM WebSphere MQ アーキテクチャーを作成できます。

IBM WebSphere MQ のアーキテクチャーを計画する前に、IBM WebSphere MQ の基本的な概念をよく理解することが必要です。IBM WebSphere MQ の概要を参照してください。

追加のキュー・マネージャーを加えることにより、サービスを提供するアプリケーションを変更せずに IBM WebSphere MQ アーキテクチャーを変更することができます。

キュー・マネージャーと同じマシン上でアプリケーションをホストしてから、別のシステム上の別のキュー・マネージャー上でホストされているサービスとの非同期通信を行うことができます。 または、サービスにアクセスしているアプリケーションをクライアントとしてキュー・マネージャーに接続してから、別のキュー・マネージャー上のサービスに非同期アクセスすることもできます。

さまざまなキュー・マネージャーとそのキューを接続する経路は、分散キューイングの手法を使用して定義します。 アーキテクチャー内のキュー・マネージャーは、チャネルを使用して接続されます。 チャネルを使用すると、キュー・マネージャーの構成に応じて、キュー・マネージャー間でメッセージが一方向に自動的に移動します。

IBM WebSphere MQ ネットワークの計画の概要については、17 ページの『ネットワークおよびネットワーク計画』を参照してください。

IBM WebSphere MQ アーキテクチャーのチャネルを計画する方法については、WebSphere MQ 分散メッセージング技法を参照してください。

分散キュー管理を使用すると、キュー・マネージャー間の通信を作成してモニターできます。 分散キュー管理について詳しくは、分散キュー管理の概要を参照してください。

関連概念

WebSphere MQ の概要

14 ページの『IBM WebSphere MQ アーキテクチャーの設計』
Point-to-Point メッセージング・スタイルおよびパブリッシュ/サブスクライブ・メッセージング・スタイル用に IBM WebSphere MQ がサポートするさまざまなアーキテクチャーについて説明します。

関連タスク

キュー・マネージャーの作成および管理

ネットワークおよびネットワーク計画

WebSphere MQ は、キュー・マネージャーおよびチャネルを使用して、 アプリケーション間で、ネットワークを介したデータの送受信を行います。 ネットワークを介して各システムを接続するフレームワークを作成するためには、 ネットワーク計画において要件を定義する必要があります。

チャネルは、システムと、通信する必要がある他のシステムとの間に 作成できます。 直接接続していないシステムに接続するために、マルチ・ホップ・チャネルを作成することができます。 各シナリオで説明されたメッセージ・チャネル接続は、17 ページの図 1 でネットワーク・ダイアグラムとして示されています。

チャネルと伝送キューの名前

伝送キューには任意の名前を付ける ことができます。 ただし、混乱を避けるためには、適宜、 宛先キュー・マネージャーの名前または別名と同じ名前を 付けるようにします。 こうすると、伝送キューに、その伝送キューで使用する経路が関連付けられるため、 中間 (マルチ・ホップ) のキュー・マネージャーを介して作成された 並列経路の概要が明確になります。

チャネル名については、あまり分かりやすくはなりません。 例えば、17 ページの図 1 で示された QM2 のチャネル名は、着信チャネルと発信チャネルとで異なっていなければなりません。 この場合にも、すべてのチャネル名には伝送キューの名前を付けることができますが、これらの名前を修飾して固有なものにしなければなりません。

例えば、QM2 には、QM1 から接続されている QM3 チャネルがあり、この QM3 チャネルは QM3 に接続されています。 これらの名前を固有なものにするには、最初のチャネルには 'QM3_from_QM1' という名前を付け、2 番目のチャネルには 'QM3_from_QM2' という名前を付けることができます。 このようにすると、チャネル名の最初の部分にその伝送キューの名前が示され、 名前の 2 番目の部分には方向および隣接キュー・マネージャーの名前が示されます。

17 ページの図 1 の場合に推奨されるチャネル名の例が、17 ページの表 1 に示してあります。

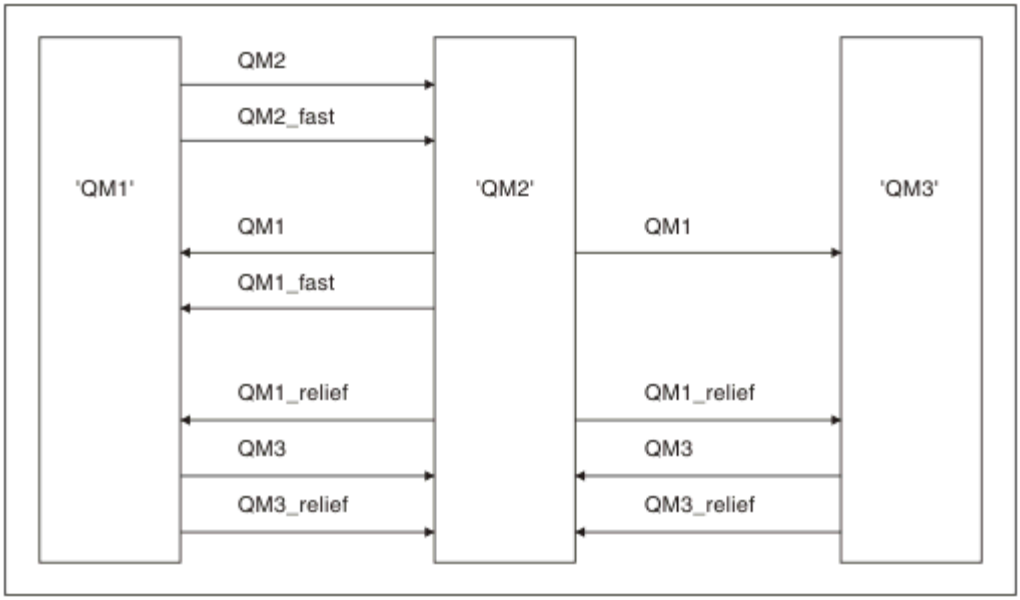


図 *1.* すべてのチャネルを表すネットワーク・ダイアグラム

表 *1.* チャネル名の例

経路名	チャネルのホストとなるキュー・マネージャー	伝送キュー名	推奨されるチャネル名
QM1	QM1 & QM2	QM1 (QM2 内)	QM1.from.QM2

表 1. チャネル名の例 (続き)			
経路名	**チャネルのホストとなるキュー・マネージャー**	**伝送キュー名**	**推奨されるチャネル名**
QM1	QM2 & QM3	QM1 (QM3 内)	QM1.from.QM3
QM1_fast	QM1 & QM2	QM1_fast (QM2 内)	QM1_fast.from.QM2
QM1_relief	QM1 & QM2	QM1_relief (QM2 内)	QM1_relief.from.QM2
QM1_relief	QM2 & QM3	QM1_relief (QM3 内)	QM1_relief.from.QM3
QM2	QM1 & QM2	QM2 (QM1 内)	QM2.from.QM1
QM2_fast	QM1 & QM2	QM2_fast (QM1 内)	QM2_fast.from.QM1
QM3	QM1 & QM2	QM3 (QM1 内)	QM3.from.QM1
QM3	QM2 & QM3	QM3 (QM2 内)	QM3.from.QM2
QM3_relief	QM1 & QM2	QM3_relief (QM1 内)	QM3_relief.from.QM1
QM3_relief	QM2 & QM3	QM3_relief (QM2 内)	QM3_relief.from.QM2

注 :

1. WebSphere MQ for z/OS では、キュー・マネージャー名は 4 文字までに制約されています。
2. ネットワーク内のすべてのチャネルに固有の名前を付けてください。 17 ページの表 1 に示すように、発信元および宛先のキュー・マネージャー名をチャネル名に含める方法を推奨します。

ネットワーク計画者

ネットワークの作成にあたっては、より高レベルのネットワーク計画者 の役割が前提となります。ネットワーク計画者が立てた計画は、チームの別のメンバーによって実現されます。

広範囲に使用されるアプリケーションの場合には、 19 ページの図 2 に示すように、 メッセージ・トラフィックを集中させるローカル・アクセス・サイトを設けて、 各ローカル・アクセス・サイト間で広帯域リンクを使用すれば、コストをより低く抑えることができます。

この例では、 2 つのメイン・システムといくつかのサテライト・システムが あります。 実際の構成は業務に関する考慮事項によって異なります。 2 つのキュー・マネージャー・コンセントレーターがキュー・マネージャーの間に配置されています。 各 QM コンセントレーターには、次のように、ローカル・キュー・マネージャーへのメッセージ・チャネルがあります。

- QM コンセントレーター 1 には、3 つのローカル・キュー・マネージャー QM1、QM2、QM3 のそれぞれに通じるメッセージ・チャネルがあります。 これらのキュー・マネージャーを使用するアプリケーションは、QM コンセントレーターを使用して相互に通信できます。
- QM コンセントレーター 2 には、3 つのローカル・キュー・マネージャー QM4、QM5、QM6 のそれぞれに通じるメッセージ・チャネルがあります。 これらのキュー・マネージャーを使用するアプリケーションは、QM コンセントレーターを使用して相互に通信できます。
- QM コンセントレーター間にはメッセージ・チャネルがあり、あるキュー・マネージャーのロケーションにあるアプリケーションが別のキュー･マネージャーのロケーションにある任意のアプリケーションとメッセージを交換できるようになっています。

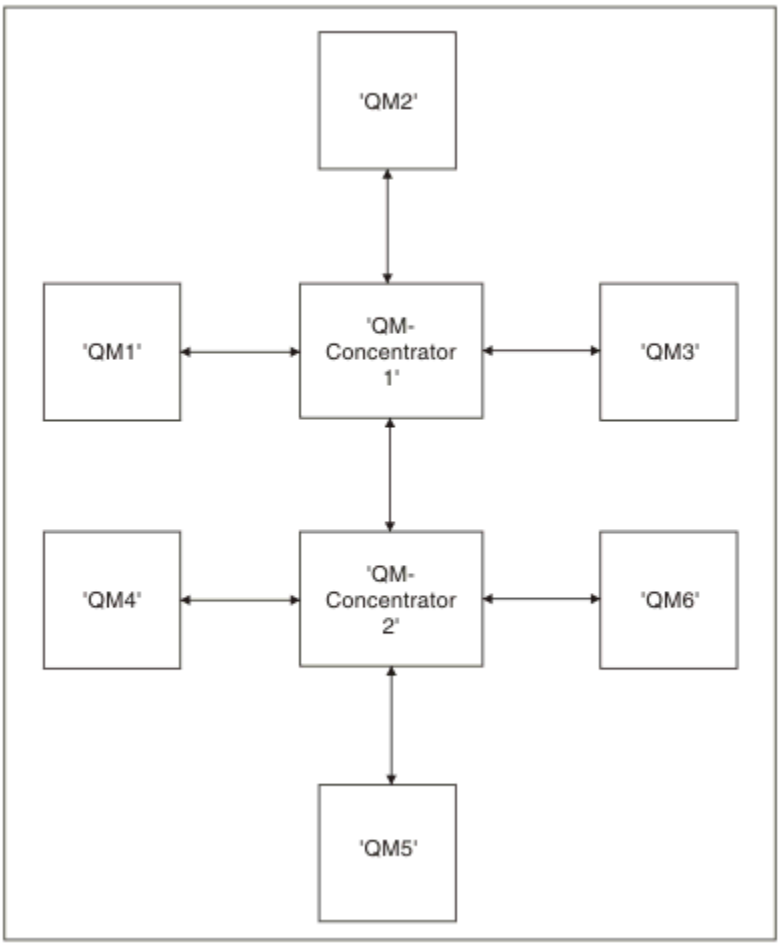


図 *2. QM* コンセントレーターを表すネットワーク・ダイアグラム

クラスタリング

このトピックでは、IBM WebSphere MQ クラスターを計画および管理するための指針について記載しています。 この情報は、テストおよびお客様からのフィードバックに基づく指針を示すものです。

以下の情報は、ユーザーが IBM WebSphere MQ クラスターの基礎を理解していることを前提にしています。 この情報は、 "すべてに当てはまる 1 つのサイズ" のソリューションとして意図されたものではなく、一般的な問題に対する一般的なアプローチを共有することを目的としています。

クラスターによって提供されるキュー・マネージャーを相互接続するためのメカニズムにより、システムのセットアップに必要な初期構成と、必要とされる継続的な管理を簡単に行えます。 構成が大規模であるほど、メリットは大きくなります。

システムのクラスター化を計画する際には、システムが正常に機能するよう、またシステムで必要とされる可用性と応答性のレベルを確保するよう注意を払う必要があります (特に、大規模または複雑なクラスター・システムの場合)。

クラスターのセットアップを成功させるには、適切な計画を立てることと IBM WebSphere MQ に関する基礎知識 (適切なアプリケーション管理やネットワーク設計など) を十分に理解することが必要です。 相互通信の概念およびクラスターが機能する仕組みにある情報に必ず精通してください。

クラスターとはどのようなもので、クラスターを使う理由は何か

クラスター化には、次の 2 つの主な利点があります。

- クラスターにより、チャネル、送信キュー、およびリモート・キューを構成するために通常は多数のオブジェクト定義を必要とする IBM WebSphere MQ ネットワークの管理が単純化されます。 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。

関連情報
クラスター化: ベスト・プラクティス

Point-to-Point メッセージング

IBM WebSphere MQ で最も単純なメッセージング形式は、Point-to-Point メッセージングです。

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。

IBM WebSphere MQ で使用できる代替メッセージング・スタイルは、パブリッシュ/サブスクライブ・メッセージングです。 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。 パブリッシュ/サブスクライブ・メッセージングの詳細については、WebSphere MQ パブリッシュ/サブスクライブ・メッセージングの紹介を参照してください。

関連情報
アプリケーションの開発
WebSphere MQ メッセージ

IBM WebSphere MQ パブリッシュ/サブスクライブ・メッセージングの紹介

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 WebSphere 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 WebSphere 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。 送信側のアプリケーションは、IBM WebSphere MQ メッセージを書き込み (必要な情報を含む)、情報のサブジェクトを示すトピックにそれを割り当てるだけで、その情報の配布処理は IBM WebSphere MQ が行います。 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。

20 ページの図 3 は、最も簡単なパブリッシュ/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。 サブスクライバーからキュー・マネージャーへサブスクリプションが送信され、パブリッシャーからキュー・マネージャーへパブリケーションが送信され、キュー・マネージャーからサブスクライバーへパブリケーションが転送されます。

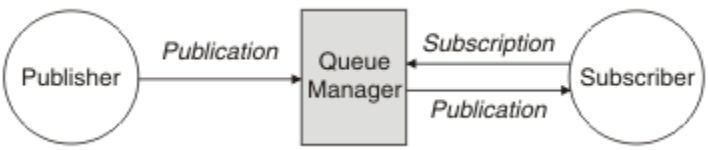


図 3. 簡単なパブリッシュ/サブスクライブの構成

標準的なパブリッシュ/サブスクライブ・システムには、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、さらにたいていは複数のキュー・マネージャーがあります。 1 つのアプリケーションがパブリッシャーとサブスクライバーの両方を兼ねることもあります。

パブリッシュ/サブスクライブ・コンポーネントの概要

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、WebSphere 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、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、さらにたいていは複数のキュー・マネージャーがあります。 1 つのアプリケーションがパブリッシャーとサブスクライバーの両方を兼ねることもあります。

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 WebSphere 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。

WebSphere 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 WebSphere 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。

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

22 ページの図 4 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。

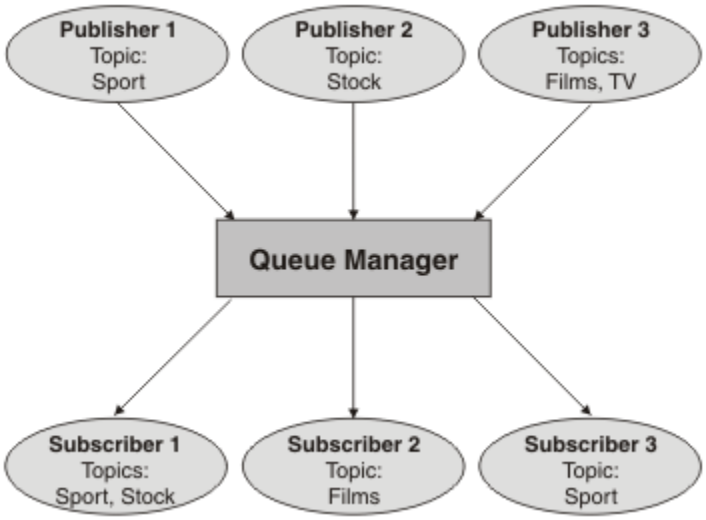


図 *4.* 単一キュー・マネージャーのパブリッシュ/サブスクライブの例

パブリッシャーとパブリケーション

WebSphere 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 WebSphere 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。

WebSphere 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。

同期点の下にあるパブリケーション

IBM WebSphere 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

WebSphere 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。

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。 管理対象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。 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のハンドル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になります。 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。 SUBTYPE ALL または ADMIN を指定した DISPLAY SBSTATUS コマンドを使用した場合でも、サブスクリプション名が表示されます。

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。

トピックの永続サブスクリプションを行えるかどうかは、**DURSUB** トピック属性を使用して制御できます。

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。

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。

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 WebSphere 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。 28 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、29 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、35 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。 管理トピック・オブジェクトについては、37 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。 SYSTEM.BASE.TOPIC。 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か、あるいはその両方かを選択できます。 両方の場合、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、33 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。 WebSphere 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、WebSphere Message Broker 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。

バージョン 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、29 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。28 ページの『トピック・ストリングの例』 は、 '#' の適切な使用法を示しています。

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
```

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。 詳しくは、 DEFINE SUB を参照してください。

注 : WebSphere MQ バージョン 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、WebSphere 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

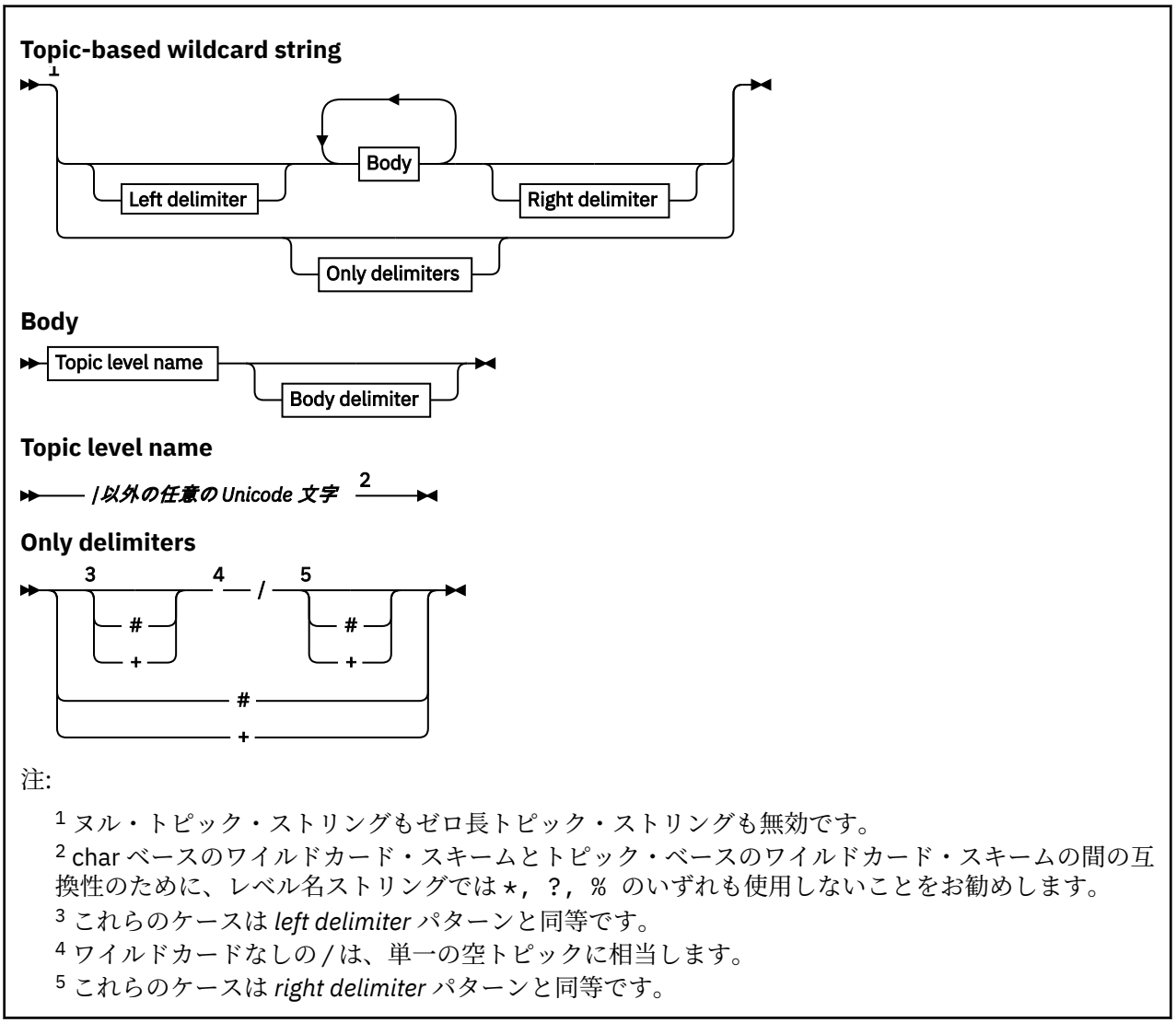
- マルチレベル・ワイルドカードは、サブスクリプションで使用されます。 パブリケーションで使用すると、リテラルとして扱われます。
- マルチレベル・ワイルドカード文字 '#' は、トピック内のレベルをいくつでも一致させる場合に使用します。 例えば、トピック・ツリー例を使用すると、'USA/Alaska/#' をサブスクライブした場合は、'USA/Alaska' トピックと 'USA/Alaska/Juneau' トピックに関するメッセージを受け取ります。
- マルチレベル・ワイルドカードはゼロ個以上のレベルを表すことができます。 したがって、'USA/#' は 'USA' 単独とも一致します。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、31 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、31 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、31 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。

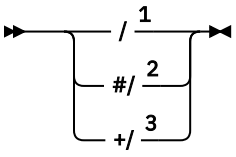
3 これらのケースは *left delimiter* パターンと同等です。

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。

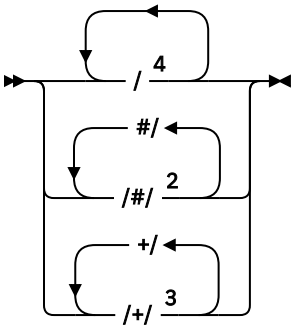
5 これらのケースは *right delimiter* パターンと同等です。

[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。

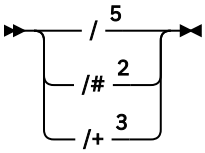
Left delimiter



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。

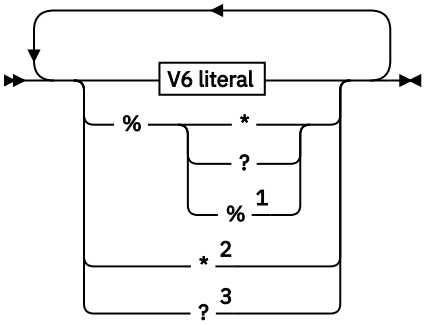
'*x*/*/*y*' はトピック・ベースのスキーマの '*x*/#/*y*' と同等で、レベル '*x* と *y*'の間のトピック階層内のすべてのトピックを選択します。ここでの '*x*' と '*y*' は、ワイルドカードで戻されたレベル・セット内にないトピック名です。

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

**? 以外の任意の 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。 28 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。 両方を指定すると、 **TOPICOBJ**/**TOPICSTR** の形式で単一のトピック・ストリングを形成するために連結されます。ここで、 **TOPICOBJ** 構成のトピック・ストリングは常に最初になり、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。

33 ページの表 2 は、トピック・ストリングの連結例を示しています。

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。
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。

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	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、 35 ページの『トピック・ツリー』のトピック名全体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。

35 ページの図 5 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。

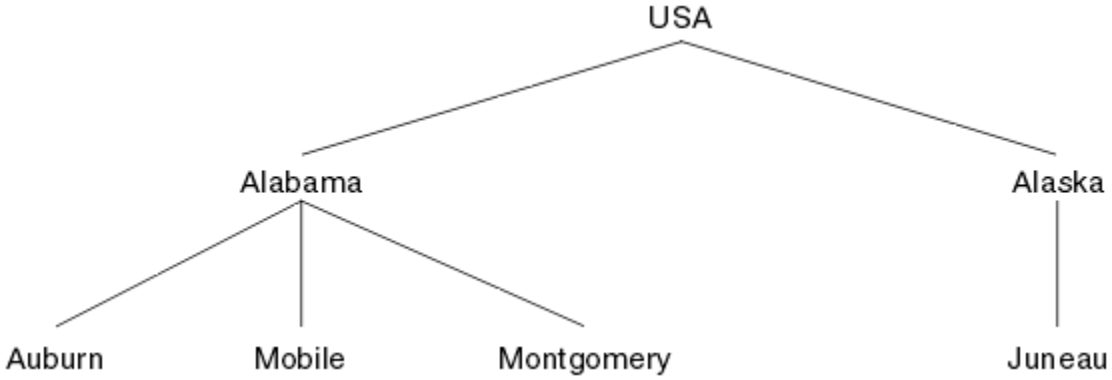


図 *5.* トピック・ツリーの例

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。

35 ページの図 5 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。

トピック・ツリー内の不要なトピック数の削減
トピック・ツリー内の不要なトピックの数を減らすことによって、パブリッシュ/サブスクライブ・システムのパフォーマンスは向上します。 どのようなトピックが不要なトピックであるか、および不要なトピックを削除する方法について説明します。

パフォーマンスに悪影響を与えることなく、多数のトピックを作成することができます。 ただし、パブリッシュ/サブスクライブを使用する方法によっては、トピック・ツリーが継続的に拡張されるものがあります。 非常に多くのトピックが、一度作成されたきり、二度と使用されることがありません。 トピック数が増加していくと、パフォーマンス上の問題が発生する場合があります。

多数の不要なトピックが作成され続けるような設計を避けるには、どうすればよいでしょうか。 キュー・マネージャーがトピック・ツリーから不要なトピックを削除できるようにするには、どうすればよいでしょうか。

キュー・マネージャーは、トピックが 30 分間使用されなければ、それを不要なトピックと見なします。 キュー・マネージャーは、使用されないトピックをトピック・ツリーから削除します。 キュー・マネージャー属性 **TREELIFE** を変更することによって、この 30 分という時間を変更することができます。 キュー・マネージャーに対して、トピックが使用されていないことが示されていることを確認し、キュー・マネージャーが不要なトピックを削除できるようにします。 36 ページの『使用されないトピックとは何か』のセクションでは、使用されないトピックとは何かが説明されています。

アプリケーション (特に、実行時間の長いアプリケーション) を設計するプログラマーは、リソースの使用 (プログラムが必要とするリソース量、無制限の要求の有無、リソース・リークの有無) について検討します。 トピックは、パブリッシュ/サブスクライブ・プログラムが使用するリソースです。 プログラムが使用するその他のリソースと同様に、トピックの使用法を綿密に調べてください。

使用されないトピックとは何か

使用されないトピックとは何かを定義する前に、具体的に何がトピックと見なされるのかについて説明します。

USA/Alabama/Auburn などのトピック・ストリングは、トピックに変換されて、トピック・ツリーに追加されます。 追加のトピック・ノードと、それに対応するトピックが、必要に応じてツリー内に作成されます。 トピック・ストリング USA/Alabama/Auburn は、3 つのトピックを含むツリーに変換されます。

- USA
- USA/Alabama
- USA/Alabama/Auburn

トピック・ツリー内のすべてのトピックを表示するには、 **runmqsc** コマンド DISPLAY TPSTATUS('#') TYPE(TOPIC) を使用します。

トピック・ツリー内の使用されていないトピックには以下の特性があります。

トピック・オブジェクトと関連付けられていない

管理トピック・オブジェクトには、それをトピックと関連付けるトピック・ストリングがあります。 トピック・オブジェクト Alabama を定義するときに、それと関連付けるトピック USA/Alabama が存在しない場合、トピックはトピック・ストリングから作成されます。 トピックが存在する場合、トピック・オブジェクトとトピックは、トピック・ストリングを使用して互いに関連付けられます。

保存パブリケーションを持たない

保存パブリケーションを持つトピックは、パブリッシャーがオプション MQPMO_RETAIN を指定してメッセージをトピックに書き込んだ結果、生成されます。

runmqsc コマンド DISPLAY TPSTATUS('USA/Alabama') RETAINED を使用して、 USA/Alabama に保存パブリケーションがあるかどうかを確認します。 応答は YES または NO です。

runmqsc コマンド CLEAR TOPICSTR('USA/Alabama') CLTRTYPE(RETAINED) を使用して、 USA/Alabama から保存パブリケーションを除去します。

子トピックを持たない

USA/Alabama/Auburn は、子トピックを持たないトピックです。 USA/Alabama/Auburn は、USA/Alabama の直接の子トピックです。

runmqsc コマンド DISPLAY TPSTATUS('USA/Alabama/+') を使用して、 USA/Alabama の直接の子を表示します。

ノードに対するアクティブなパブリッシャーがない

ノードに対するアクティブなパブリッシャーは、出力用にトピックをオープンにしているアプリケーションです。

例えば、アプリケーションは、オープン・オプション MQOO_OUTPUT を指定して、**Alabama** という名前のトピック・オブジェクトを開きます。

USA/Alabama およびそのすべての子に対してアクティブなパブリッシャーを表示するには、**runmqsc** コマンド DISPLAY TPSTATUS('USA/Alabama/#') TYPE(PUB) ACTCONN を使用します。

ノードに対するアクティブなサブスクライバーがない

アクティブなサブスクライバーは、永続サブスクリプション、またはトピックへのサブスクリプションを MQSUB に登録したが、それをクローズしていないアプリケーションのいずれかです。

USA/Alabama に対するアクティブなサブスクリプションを表示するには、 **runmqsc** コマンド DISPLAY TPSTATUS('USA/Alabama') TYPE(SUB) ACTCONN を使用します。

USA/Alabama とそのすべての子に対するアクティブなサブスクリプションを表示するには、**runmqsc** コマンド DISPLAY TPSTATUS('USA/Alabama/#') TYPE(SUB) ACTCONN を使用します。

トピック・ツリー内のトピック数の削減

要約すると、トピック・ツリー内のトピック数を減らす方法は数多くあります。

TREELIFE の変更

使用されないトピックの存続時間は、デフォルトで 30 分間です。 使用されないトピックの存続時間を短くすることができます。

例えば、**runmqsc** コマンド ALTER QMGR TREELIFE(900)は、未使用のトピックの存続時間を 30 分から 15 分に短縮します。

キュー・マネージャーの再始動 (例外的)

キュー・マネージャーを再始動すると、トピック・ツリーは、トピック・オブジェクト、保存パブリケーションを含むノード、および永続サブスクリプションから再初期設定されます。 パブリッシャーおよびサブスクライバー・プログラムの操作によって作成されたトピックは除去されます。

定期的に **runmqsc** コマンド DISPLAY TPSTATUS('#') TYPE(TOPIC) を使用して、すべてのトピックをリストし、その数が増加しているかどうかを確認します。

不要なトピックの増加がパフォーマンス上の問題の原因となっていた場合、最後の手段としてキュー・マネージャーを再始動します。

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。

38 ページの図 6 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 6. 視覚化したトピック・ツリー

38 ページの図 7 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 7. 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 トピックに管理トピック・オブジェクトはなく、 SYSTEM.BASE.TOPIC から属性を継承します。

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。 以下の例では、管理トピック・オブジェクトを定義して、サッカー・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 に設定します。

```
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。

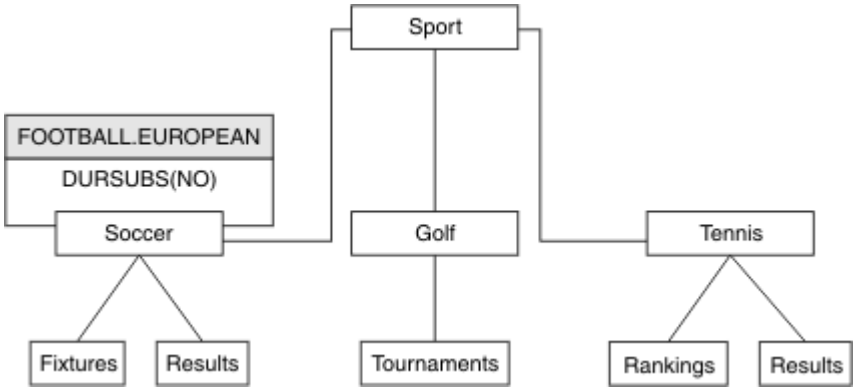


図 8. 「*Sport/Soccer*」トピックに管理トピック・オブジェクトを関連付けたトピック・ツリーを視覚化した図

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。 39 ページの図 9 は、前の例を拡張したものです。

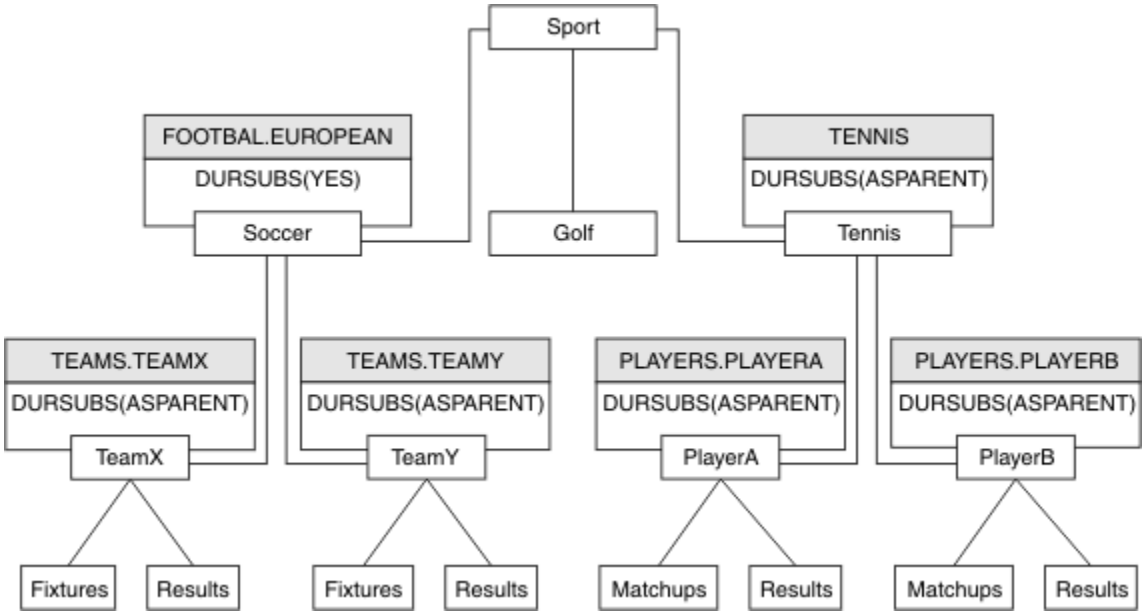


図 9. 複数の管理トピック・オブジェクトを定義したトピック・ツリー

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。 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・オブジェクトでは、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

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するには、MQSC **Topic** WILDCARD プロパティーまたは同等の PCF トピック `WildcardOperation` プロパティーを使用します。 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。

44 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターの作成』 の例では、 40 ページの図 10 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。

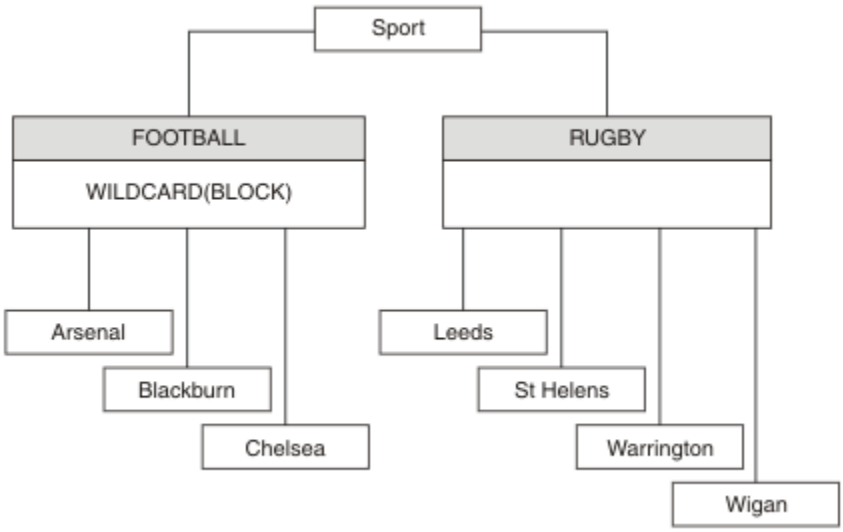


図 *10. WILDCARD* プロパティー *BLOCK* を使用するトピック・ツリー

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。 `WILDCARD` プロパティー値 `PASSTHRU` を `Sport` ツリー内のノードに設定できます。 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。 46 ページの図 14 を参照してください。 47 ページの図 17 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 *11. QMA* へのパブリッシュ

結果を 41 ページの表 3 に示します。 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	Notes®
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	-	WILDCARD(BLOCK) on Sports/Football は、 Arsenal でワイルドカード・サブスクリプションを禁止します。
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:

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・オブジェクトと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。 40 ページの図 10 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。

この例のクラスター・セットアップでは、 44 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターの作成』 QMB には QMA と同じサブスクリプション・セットがありますが、 QMB は QMA にパブリッシ

ュされたパブリッシャーの後にパブリケーションを受信しませんでした。 41 ページの図 11 を参照してください。 Sports/Football トピックと Sports/Rugby 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。 プロキシー・サブスクリプションは QMB から QMA に伝搬されません。 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 です。これにはトピック・オブジェクトがないため、 Sports/#/Leeds はローカル・トピック・オブジェクトである SYSTEM.BASE.TOPIC を継承します。

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 (41 ページの図 11 を参照してください)。

42 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。 43 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	記録 (Notes)
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	-	WILDCARD(BLOCK) on Sports/Football は、 Arsenal でワイルドカード・サブスクリプションを禁止します。
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。

43 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 は両方ともクラスター・トピックであるため、予期されるように、2 つの新しいサブスクリプションはパブリケーションを受け取ります。 QMB が Sports/

Football/Arsenal および Sports/Rugby/Leeds のプロキシー・サブスクリプションを QMA に転送し、がパブリケーションを QMB に送信しました。

予期しない結果は、以前に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/# および Sports/#/Leeds が、パブリケーションを受信するようになったことです。 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	記録 (Notes)
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	-	WILDCARD(BLOCK) on Sports/Football は、 Arsenal でワイルドカード・サブスクリプションを禁止します。
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

WebSphere MQ バージョン 6 のストリームは、WebSphere MQ バージョン 7 によってトピックにマップされます。 47 ページの『ストリームおよびトピック』を参照してください。 バージョン 7 の **strmqbrk** によって実行されるデフォルトのマッピングでは、 ストリーム Sports 内のすべてのトピックはトピック Sports にマップされます。 ストリーム Business 内のすべてのトピックは、トピック Business にマップされます。

WebSphere MQ バージョン 6 では、 Sports ストリーム内の ★ へのサブスクリプションは、Sports ツリー内のすべてのパブリケーションを受信し、 Business ツリー内のパブリケーションは受信しません。 バージョン 7 では、このサブスクリプションは Sports ツリー内のすべてのパブリケーション と Business ツリー内のすべてのパブリケーションを受信します。 この動作をブロックするために、ストリームがバージョン 7 にマイグレーションされると、**strmqbrk** は WILDCARD プロパティーを設定します。 このプロパティーは、ストリームから移行される各トップレベル・トピックについて、値 BLOCK に設定されます。

Sports および Business の WILDCARD プロパティーは、Sports および Business という名前のバージョン 6 ストリームからの変換により、値 BLOCK に設定されます。

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、* へのサブスクリプションはパブリケーションを受信しません。「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。

パブリッシュ/サブスクライブ・ブローカーを WebSphere MQ バージョン 7 に移行しても、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・アプリケーションの動作は変わりません。 **Publish**、**Register Publisher**、または **Subscriber** コマンドの **StreamName** プロパティーは、ストリームのマイグレーション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

WebSphere Message Broker サブスクリプション・ポイントは、 WebSphere MQ バージョン 7 によってトピックに マップされます (49 ページの『サブスクリプション・ポイントおよびトピック』を参照)。 バージョン 7 で **migmqbrk** によって実行されるデフォルトのマッピングでは、 サブスクリプション・ポイント Sports 内のすべてのトピックはトピック Sports にマップされます。 サブスクリプション・ポイント Business 内のすべてのトピックは、トピック Business にマップされます。

WebSphere Message Broker バージョン 6 において、 Sports サブスクリプション・ポイント内の * へのサブスクリプションは、 Sports ツリー内のすべてのパブリケーションを受信し、 Business ツリー内のパブリケーションは受信しません。 バージョン 7 の同じサブスクリプションは、 Sports ツリー内のすべてのパブリケーションと、Business ツリー内のすべてのパブリケーションを受け取ります。 この動作をブロックするため、サブスクリプション・ポイントがバージョン 7 に移行される際、 **migmqbrk** で WILDCARD プロパティーが設定されます。 このプロパティーは、サブスクリプション・ポイントから移行される各トップレベル・トピックについて、値 BLOCK に設定されます。 Sports と Business の WILDCARD プロパティーは、 Sports および Business と呼ばれる WebSphere Message Broker サブスクリプション・ポイントからの変換時に、 値 BLOCK に設定されます。

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。

パブリッシュ/サブスクライブ・ブローカーを WebSphere MQ バージョン 7 に移行しても、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・アプリケーションの動作は変わりません。 **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マイグレーション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・クラスターの作成

以下のステップでは、4 つのキュー・マネージャー (2 つの完全リポジトリー、 CL1A と CL1B 、および 2 つの部分リポジトリー、 QMA と QMB) を持つクラスター CL1 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
```

40 ページの図 10 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 *12.* 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、40 ページの図 10 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、この例でさまざまなサブスクリプションをテストするために役立つプロパティーです。 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 *13.* 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 *14.* 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 *15.* 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 *16.* 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 *17.* 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。 IBM WebSphere MQ Version 6.0 では、ストリームはキューとして実装され、ストリームをサポートするブローカーごとに定義されます。 それぞれのキューは同じ名前 (ストリームの名前) になります。 IBM WebSphere MQ Version 7.0 以降、ストリームは、管理上異なるトピック ID にマップできる最上位の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。 デフォルトのストリームにパブリッシュされたトピックは、キュー型パブリッシュ/サブスクライブが有効な状態で実行されている、接続済みのすべての Version 6.0 ブローカー、および Version 7.0 以降のすべてのキュー・マネージャーから、すぐに使用することができます。 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。

定義したトピックは、Version 6.0 のパブリッシュ/サブスクライブ・ブローカー、および IBM WebSphere MQ の後継バージョンで実行されているパブリッシャーおよびサブスクライバーで、特別な構成を行うことなく使用できます。

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

ソリューション設計者は、キューに入れられたパブリッシュ/サブスクライブ・プログラミング・モデルを使用して、すべてのスポーツ・パブリケーションを Sport という名前のストリームに入れることを決定する場合があります。 Version 6.0 では通常、モデル・キュー SYSTEM.BROKER.MODEL.STREAM を使用する他のブローカー上にストリームが自動的に複製されます。 しかし、キュー型パブリッシュ/サブスクライブが有効な状態で Version 7.0 以降で実行されるキュー・マネージャーで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。

キュー・マネージャーを Version 6.0 からマイグレーションする場合、コマンド **strmqbrk** を実行すると、Version 6.0 で指定されたストリームがトピックへマイグレーションされます。 ストリーム Sport は、トピック Sport にマップされます。 ただし、これは z/OS には適用されません。

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 を指定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。

strmqbrk によって作成されたトピック Soccer/Result は、トピック・ストリング Sport を使用して、トピック Sport の子として定義されます。 Soccer/Results へのサブスクリプションは Sport/Soccer/Results へのサブスクリプションとして実現されるため、 Sport ストリームへのパブリケーションは、トピック・スペース内の別の場所にマップされ、別のストリーム (Business など) へのパブリケーションにマップされます。

場合によっては、**strmqbrk** により実行される自動移行が解決策にはならず、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 3 つの理由があります。

1. バージョン 6 のキュー・マネージャー上で引き続きパブリッシュ/サブスクライブ・アプリケーションを維持して、後のバージョンのキュー・マネージャーで作動する新しく作成されたパブリッシュ/サブスクライブ・アプリケーションと相互運用する場合。
2. 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。
3. 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、パブリッシュまたはサブスクライブの場合) 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、空でないトピック・ストリングを使用して再作成された場合、そ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

キュー型パブリッシュ/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で、Version 7.0 以降で模擬的に使用されます。 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ために、そのストリームと関連するトピック用に、別のトピック・ストリングを選択することができます。 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。 Version 7.0 では、トピック・ストリングが衝突した場合、ストリームを使用してトピック・スペースを分離することができました。 Version 7.0 以降は、接頭部付加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。

例

49 ページの図 18 では、トピック 'Sport' にトピック・ストリング 'xyz' があるため、バージョン 7 のキュー・マネージャーのトピック・スペースでは、ストリーム 'Sport' からのパブリケーションの接頭部としてストリング 'xyz' が付けられます。 バージョン 7 でトピック 'Sport' にパブリッシュまたはサブスクライブを行うと、トピック・ストリングに接頭部 'xyz' が付加されます。 パブリケーションがバージョン 6 のサブスクライバーに流れると、パブリケーションから接頭部 'xyz' が削除され、そのパブリケーションが 'Sport' ストリーム内に置かれます。 逆に、パブリケーションがバージョン 6 からバージョン 7 に、 'Sport' ストリームから 'Sport' トピックに流れる場合、接頭部 'xyz' がトピック・ストリングに追加されます。

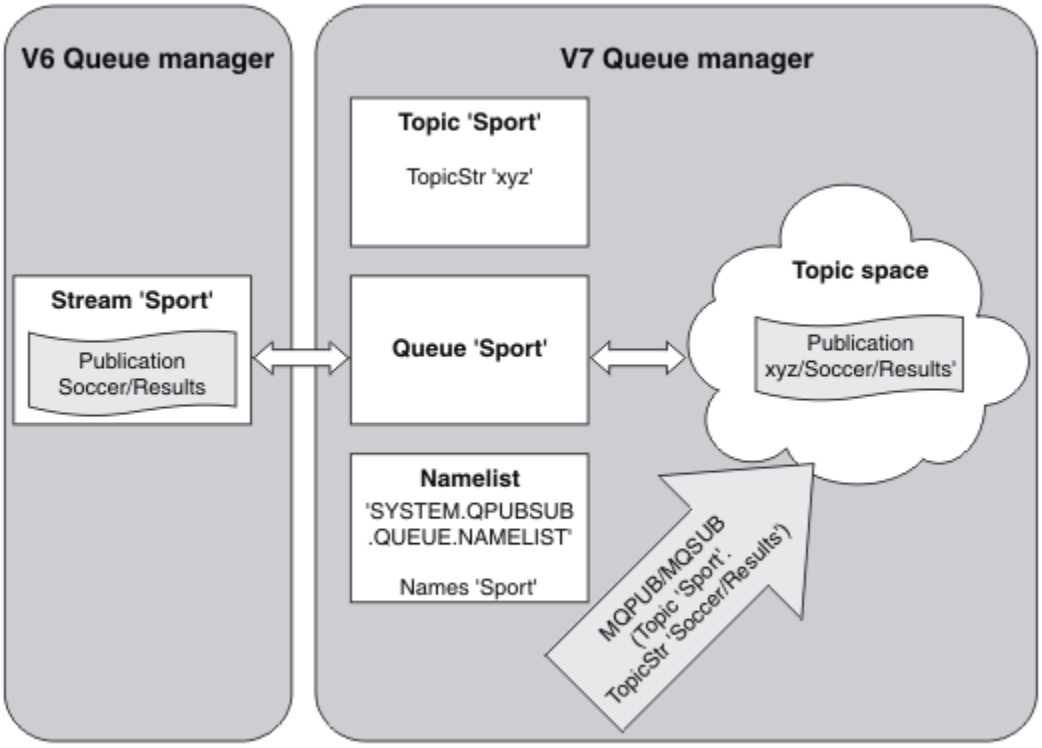


図 *18.* バージョン *6* のストリームとバージョン *7* のトピックの共存

サブスクリプション・ポイントおよびトピック

WebSphere MQ Event Broker および Message Broker では、Publication ノードの特定のセットにパブリケーションを要求する際に、サブスクリプション・ポイントが使用されます。 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。

WebSphere MQ Event Broker V6.0 から WebSphere MQ V7.0.1 へのマイグレーション・プロシージャー **migmbbrk** は、名前付きサブスクリプション・ポイントをトピックとトピック・オブジェクトに変換します。 サブスクリプション・ポイントは、保存パブリケーションがある場合、つまり登録済みサブスクライバーがある場合は、自動的にマイグレーションされます。 **migmbbrk** は、名前付きサブスクリプション・ポイントからトピック・オブジェクトを作成します。 サブスクリプション・ポイントの名前は、トピック・オブジェクトの名前とトピック・ストリングそのものになります。 トピック・オブジェクトが SYSTEM.QPUBSUB.SUBPOINT.NAMELIST に追加されます。

同じ名前のトピック・オブジェクトが存在する場合、**migmbbrk** は次の 2 つの処理のいずれかを行います。

1. トピック・オブジェクトのトピック・ストリングが異なる場合、またはサブスクリプション・ポイント名がオブジェクト名より長い場合、**migmbbrk** は生成された名前を持つトピック・オブジェクトを作成します。
2. トピック・オブジェクトのトピック・ストリングが同じである場合、**migmbbrk** は既存オブジェクトを名前リストに追加します。

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。

WebSphere MQ Event Broker におけるサブスクリプション・ポイント

WebSphere MQ Event および Message Broker メッセージ・フローでは、サブスクライバーへのメッセージのフィルタリングと送信に Publication ノードが使用されます。 パブリッシャーは通常、Publication ノードにサブスクリプション・ポイントを設定しません。 サブスクライバーも、トピックの特定のセットへのインタレストを登録し、通常はサブスクリプション・ポイントを指定しません。

サブスクリプション・ポイントは、どの Publication ノードがサブスクリプションにメッセージを転送するかを選択するための手段です。 サブスクライバーは、トピックのセットへのインタレストをサブスクリプション・ポイントの名前で限定します。

Publication ノードのサブスクリプション・ポイント名を設定するには、その「**Subscription point**」プロパティーに名前を割り当てます。

サブスクリプション・ポイント・プロパティーは、トピックのパブリケーションを同じトピックのサブスクライバーに転送するかどうかを制御します。 名前付きサブスクリプション・ポイントが設定された Publication ノードからのパブリケーションは、同じサブスクリプション・ポイントのサブスクライバーにのみ転送されます。 名前付きサブスクリプション・ポイントが設定されていない Publication ノードからのパブリケーションは、デフォルトでは、サブスクリプション・ポイントの名前が付けられていないサブスクライバーにのみ転送されます。

名前付きサブスクリプション・ポイントのあるノードは、**SubPoint** プロパティーが設定された Publish コマンド・メッセージを MQRFH2 形式で送信します。 名前付きサブスクリプション・ポイントのサブスクリプションは、MQRFH2 Register subscriber コマンド・メッセージで **SubPoint** プロパティーを設定する必要があります。

WebSphere MQ におけるサブスクリプション・ポイント

WebSphere MQ は、WebSphere 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・ポイントが含まれないコマンド・メッセージ内のトピックは、WebSphere MQ トピック・ツリーのルートに変更なしでマップされ、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・ポイントが含まれるコマンド・メッセージは、SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 内のトピック・オブジェクトのリストを使用して処理します。 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。 トピックはそのプロパティーを、SYSTEM.QPUBSUB.SUBPOINT.NAMELIST で検出された関連トピック・オブジェクトから継承します。

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・オブジェクトで設定されていないプロパティーは、SYSTEM.BASE.TOPIC から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。

新しい WebSphere MQ V7 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。 一致するトピック・オブジェクトのいずれかから継承するトピックを使用する場合は、一致するサブスクリプション・ポイントを使用する、キューに入れられたアプリケーションと相互動作します。

例

集合内の既存の WebSphere MQ Event Broker パブリッシュ/サブスクライブ・アプリケーションが、サブスクリプション・ポイントを使用して株価を異なる通貨で発行します。 IBM 株式のドル・スポット株価は、サブスクリプション・ポイント USD とトピック NYSE/IBM/SPOT を使用してパブリッシュされます。 ポンド株価は、同じトピックとサブスクリプション・ポイント GBP を使用してパブリッシュされます。

WebSphere MQ のマイグレーション・プロシージャーは、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' および 'USD' と共に作成します。

WebSphere MQ 上で動作するようにマイグレーションされた、トピック NYSE/IBM/SPOT の既存のパブリッシャー (サブスクリプション・ポイント USD を使用) は、トピック USD/NYSE/IBM/SPOT の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 USD を使用する、NYSE/IBM/SPOT の既存のサブスクライバーは、USD/NYSE/IBM/SPOT のサブスクリプションを作成します。

MQSUB を呼び出すことによって、バージョン 7 パブリッシュ/サブスクライブ・プログラムでドル・スポット株価をサブスクライブします。「C」コード・フラグメントで示すように、USD トピック・オブジェクトとトピック・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);
```

集合内の WebSphere MQ Event Broker アプリケーションがサブスクリプション・ポイント USD および GBP を常に使用していたかどうかを考慮してください。 使用していた場合は、USD および GBP トピック・オブジェクトを、クラスター・トピック・ホスト上のクラスター・トピックとして 1 回だけ作成します。 クラスター内のすべてのキュー・マネージャーで SYSTEM.BASE.TOPIC をクラスター・トピックに変更するために、マイグレーション手順のステップ ../com.ibm.mq.mig.doc/q007670_.dita#q007670_/clusterstep を実行する必要はありません。 代わりに、以下のステップを実行します。

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・マネージャーの USD および GBP トピック・オブジェクトのすべてのコピーを削除します。
3. クラスター内のすべてのキュー・マネージャーの SYSTEM.QPUBSUB.SUBPOINT.NAMELIST で、USD および GBP を定義しておきます。

分散パブリッシュ/サブスクライブ

このセクションには、キュー・マネージャー間でのパブリッシュ/サブスクライブ・メッセージングの実行方法、またキュー・マネージャー、クラスター、および階層の接続に使用できる 2 つの異なるキュー・マネージャー・トポロジーについての説明が含まれています。

キュー・マネージャーは、WebSphere MQ パブリッシュ/サブスクライブ・システム内の他のキュー・マネージャーと通信できるため、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て、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できます。 これは 52 ページの図 19 に図示されています。

52 ページの図 19 は、2 つのキュー・マネージャーのあるパブリッシュ/サブスクライブ・システムを示しています。

- 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。

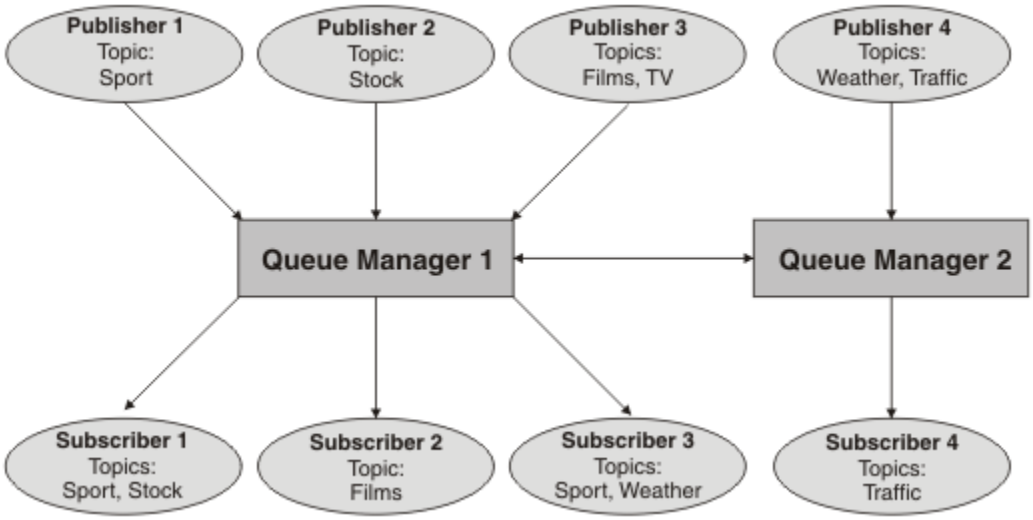


図 *19.* キュー・マネージャーが *2* つあるパブリッシュ/サブスクライブの例

分散パブリッシュ/サブスクライブの仕組み

WebSphere MQ パブリッシュ/サブスクライブはプロキシー・サブスクリプションを使用して、リモート・キュー・マネージャーにパブリッシュされるメッセージをサブスクライバーが受け取れるようにします。

分散パブリッシュ/サブスクライブは、分散キューイングと同じコンポーネントを使用して、キュー・マネージャーのネットワーク、さらには、それらのキュー・マネージャーに接続しているアプリケーションに接続します。 キュー・マネージャーと、キュー・マネージャー間の接続に関係するコンポーネントの間のメッセージングとの詳細については、「相互通信」という資料を参照してください。

分散パブリッシュ/サブスクライブ・システムにおいて、サブスクライバーは標準のサブスクリプション操作以外を行う必要はありません。 キュー・マネージャーでサブスクリプションが行われると、キュー・マネージャーは、接続されているキュー・マネージャーへサブスクリプションを伝搬するプロセスを管理します。 プロキシー・サブスクリプションはネットワーク内のすべてのキュー・マネージャーに流れます。 それらは、オリジナルのサブスクリプションが作成されたキュー・マネージャーへパブリケーションを経路指定して戻すために作成されます。53 ページの図 20 を参照してください。

パブリケーションがリモート・キュー・マネージャーへ伝搬されるのは、そのトピックへのサブスクリプションがそのリモート・キュー・マネージャー上に存在する場合だけです。

キュー・マネージャーは、そこで作成されるすべてのサブスクリプションを、それがローカル・アプリケーションからかリモート・キュー・マネージャーからかに関係なく統合します。 サブスクリプションが既に存在していない限り、キュー・マネージャーはサブスクリプションのトピックに対するプロキシー・サブスクリプションを隣接するキュー・マネージャーとの間で作成します。53 ページの図 21 を参照してください。

アプリケーションが情報をパブリッシュすると、受け取る側のキュー・マネージャーは、リモート・キュー・マネージャー上で有効なサブスクリプションを持つすべてのアプリケーションへそれを転送します。 1 つ以上の中間キュー・マネージャーを介してそれを転送する場合もあります。54 ページの図 22 を参照してください。

サブスクライバー 1 は、Asia キュー・マネージャー上で特定のトピックのサブスクリプションを登録します (1)。 このトピックのサブスクリプションは、ネットワーク内の他のすべてのキュー・マネージャーへ転送されます (2、3、4)。

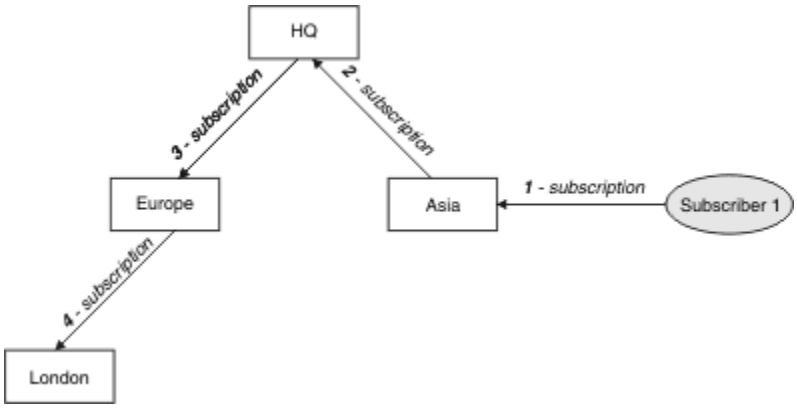


図 *20.* キュー・マネージャー・ネットワークを介したサブスクリプションの伝搬

サブスクライバー 2 は、HQ キュー・マネージャーで 53 ページの図 20 と同じトピックへのサブスクリプションを登録します (5)。 このトピックに対するサブスクリプションは Asia キュー・マネージャーに転送されるので、ネットワーク上の他の場所にもそれらのサブスクリプションが存在することが分かります (6)。 サブスクリプションは Europe キュー・マネージャーには転送されません。これは、このトピックに対するサブスクリプションが既に登録されているためです。53 ページの図 20 のステップ 3 を参照してください。

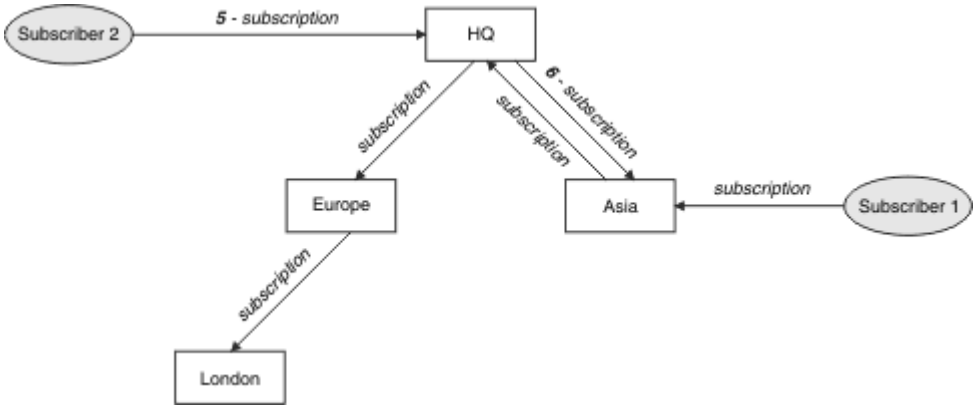


図 *21.* 複数のサブスクリプション

パブリッシャーが、53 ページの図 21 と同じトピックに関するパブリケーションを Europe キュー・マネージャーに送信します (7)。 このトピックに対するサブスクリプションが HQ から Europe に対して存在するため、パブリケーションが HQ キュー・マネージャーへ転送されます (8)。 しかし、London から Europe に対してサブスクリプションが存在しない (Europe から London に対してのみ) ため、London キュー・マネージャーにはパブリケーションは転送されません。 HQ キュー・マネージャーはパブリケーションをサブスクライバー 2 および Asia キュー・マネージャーへ直接送信します (9)。 パブリケーションが Asia からサブスクライバー 1 へ転送されます (10)。

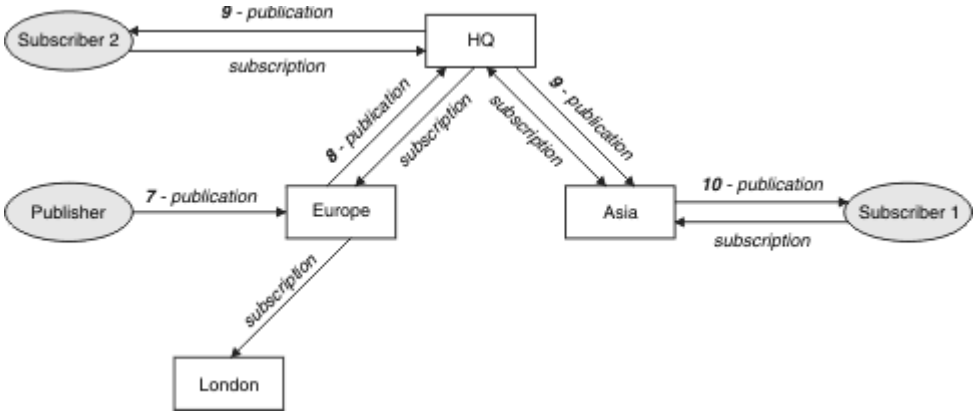


図 *22.* キュー・マネージャー・ネットワークを介したパブリケーションの伝搬

キュー・マネージャーは、パブリケーションまたはサブスクリプションを別のキュー・マネージャーへ送信する際に、メッセージにそれ自体のユーザー ID を設定します。 パブリッシュ/サブスクライブ階層を使用しており、メッセージ内のユーザー ID の権限を使用してメッセージを書き込むように着信チャネルがセットアップされている場合、送信側のキュー・マネージャーのユーザー ID に権限を与えなければなりません。112 ページの『キュー・マネージャー階層でのデフォルトのユーザー ID の使用』を参照してください。 パブリッシュ/サブスクライブ・クラスターを使用している場合、権限を与える処理はクラスターによって扱われます。

パブリッシュ/サブスクライブのキュー・マネージャーの相互接続という性質上、プロキシー・サブスクリプションがネットワーク内のすべてのノードに伝搬するのに時間がかかります。 リモート・パブリケーションでは、必ずしも即時にサブスクライブが開始されるわけではありません。 55 ページの『ルーティング・メカニズムの詳細』で説明されているように、**Topic** 属性 PROXYSUB で値 FORCE を設定することにより、このサブスクリプションの遅延を避けることができます。

サブスクリプション操作が完了するのは、直接接続されている各キュー・マネージャーの適切な伝送キューにプロキシー・サブスクリプションが書き込まれた時点です。 サブスクリプション操作で、トポロジーの残りの部分へのプロキシー・サブスクリプションの伝搬を待機することはありません。

プロキシー・サブスクリプションは、それを作成したキュー・マネージャー名と関連付けられています。 階層内に同じ名前のキュー・マネージャーが存在する場合、パブリケーションが到達できなくなってしまう場合があります。 この問題を避けるために、Point-to-Point メッセージングの場合と同様に、キュー・マネージャーに固有の名前を付けてください。これは特に、それらが WebSphere MQ ネットワーク内で直接または間接的に接続されている場合に必要です。

分散パブリッシュ/サブスクライブ・ネットワーク内では、パブリケーションおよびサブスクリプションの有効範囲を使用して、パブリケーションおよびサブスクリプションのフローを制御することができます (必要であれば、制限することもできます)。

プロキシー・サブスクリプションの集約およびパブリケーションの集約

パブリッシュ/サブスクライブ・キュー・マネージャー間で受け渡しされるメッセージの量を最小限に抑えるために、分散パブリッシュ/サブスクライブ・パブリケーションおよびプロキシー・サブスクリプションは集約されます。

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。 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 (52 ページの『分散パブリッシュ/サブスクライブの仕組み』を参照)。

キュー・マネージャーをまとめて、パブリッシュ/サブスクライブ階層またはパブリッシュ/サブスクライブ・クラスターに接続できます。 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 (65 ページの『パブリッシュ/サブスクライブ・トポロジー』を参照)。

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。

Topic 属性 PUBSCOPE および SUBSCOPE を使用して、接続されているキュー・マネージャー間でのプロキシー・サブスクリプションおよびパブリケーションのフローを制限できます。 また、**Topic** 属性 WILDCARD を BLOCK に設定することにより、ワイルドカードを含むプロキシー・サブスクリプションのフローを制限することもできます (57 ページの『ワイルドカードの規則』を参照)。

プロキシー・サブスクリプションは、サブスクリプションの作成とは非同期の状態で、キュー・マネージャーの間で流れます。 トピック (またはサブスクライブされるトピックの親) に対して **Topic** 属性 PROXYSUB を FORCE に設定することにより、接続されているすべてのキュー・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のを待機するための待ち時間を削減できます (55 ページの『ルーティング・メカニズムの詳細』を参照)。

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。 個々のプロキシー・サブスクリプションは、 FORCE に設定されているトピック属性 **PROXYSUB** を使用して統合できます。 ルーティング・メカニズムおよびクラスター・トピックのパフォーマンスの詳細については、55 ページの『ルーティング・メカニズムの詳細』を参照してください。

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。

ルーティング・メカニズムの詳細
全対象パブリッシュ は、個々のプロキシー・サブスクリプション転送の代わりになるルーティング・メカニズムです。 個々のプロキシー・サブスクリプション転送は、トピック・ストリングのサブスクリプションに対応するパブリケーションのみがリモート・メッセージング・サーバーに送信されることを意味します。 全対象パブリッシュ (つまり、ブロードキャスト) は、メッセージング・サーバーにパブリッシュされるすべてのパブリケーションを、分散パブリッシュ/サブスクライブ・ネットワーク内の他のすべてのメッ

セージング・サーバーに転送することで機能します。 その後、受信側のメッセージング・サーバーは、ローカル・サブスクリプションに対応するパブリケーションを送信します。

各メカニズムにはそれぞれ利点がありますが、制限もあります。

個々のプロキシー・サブスクリプション転送
このメカニズムの場合、キュー・マネージャーのサブスクリプションに対応するパブリケーションのみが送信されるため、キュー・マネージャー間のパブリケーション・トラフィックが最小になります。

ただし、以下の制限があります。

- サブスクライブされる個々のトピック・ストリングの場合、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・トポロジー内の他のすべてのキュー・マネージャーに送信されます。 このメッセージング・オーバーヘッドは、作成または削除するサブスクリプション (キュー・マネージャーの再始動後のすべての非永続サブスクリプションなど) が何千個もある場合、またはサブスクリプションのセットが急速に変化しており、それぞれ別のトピック・ストリングにサブスクライブされる場合には、非常に大きくなることがあります。
- プロキシー・サブスクリプションは非同期メッセージングを使用して、他のキュー・マネージャーに流れます。したがって、サブスクリプションの作成からプロキシー・サブスクリプションの作成、送信、および他のキュー・マネージャーによる処理まで間に遅延が発生します。 この間隔の中でこれらのキュー・マネージャーでパブリッシュされるメッセージは、リモート・サブスクリプションには送信されません。

全対象パブリッシュ
このメカニズムの場合は次のようになります。

- システムにおけるトピック・ストリングごとのプロキシー・サブスクリプション・オーバーヘッドは存在しません。つまり、短時間でサブスクリプションの作成、削除、または変更が行われても、ネットワーク・ロードおよび処理が増えることはありません。
- サブスクリプションの作成からキュー・マネージャーに流れるパブリケーションまでの間に遅延はありません。これらは常にすべてのキュー・マネージャーに流れるからです。 したがって、パブリケーションが新しく作成されたリモート・サブスクリプションに送信されない時間帯はありません。

ただし、以下の制限があります。

- すべてのパブリケーションがパブリッシュ/サブスクライブ・トポロジー内のすべてのキュー・マネージャーに送信されます。そのため、各キュー・マネージャーでパブリケーションに対応するサブスクリプションが存在せずに、ネットワーク・トラフィックが過度に増加する可能性があります。

全対象パブリッシュ・メカニズムを使用すると良い状況としては、クラスターまたは階層内のかなりの割合のキュー・マネージャーからパブリケーションのサブスクライブが出されることが想定される場合、またはサブスクリプションの変更の頻度が原因でプロキシー・サブスクリプション・オーバーヘッドが大きくなりすぎる場合があります。 この処理方法をこれらの事例で使用すると、すべてのキュー・マネージャーにパブリケーションが送信されてメッセージング・トラフィックが増加するような事例と比較して、効果的に処理できる可能性があります。対応するサブスクリプションが存在するキュー・マネージャーに送信されるからです。

上位トピック・オブジェクトの **PROXYSUB** 属性を FORCE に設定することにより、 IBM WebSphere MQ 分散パブリッシュ/サブスクライブ・トポロジーで全対象パブリッシュ・メカニズムを有効にすることができます。

個々のプロキシー・サブスクリプションを無効にする場合の詳細については、79 ページの『個々のプロキシー・サブスクリプションの無効化』を参照してください。

この強制プロキシー・サブスクリプションがトポロジー全体に伝搬されると、新しいサブスクリプションは、他の接続先のキュー・マネージャーからあらゆるパブリケーションをただちに (待ち時間なしで) 受け取るようになります。

このようなシステムを構成する場合は注意が必要です。 **PROXYSUB** が FORCE に設定されているトピックの下位にあるトピック・オブジェクトは、**PROXYSUB** が FORCE に設定されているノードとは異なるクラスターまたは階層ストリームに配置することはできません。 同様に、下位トピック・オブジェクトでは **WILDCARD** 属性を BLOCK に設定することはできません。 いずれの場合も、パブリッシュされたメッセージがキュー・マネージャー間を正しく流れなくなる可能性があります。

PROXYSUB が FORCE に設定されている場合でも、サブスクライブされる個々のトピック・ストリングのプロキシー・サブスクリプションは引き続き伝搬されます。 サブスクリプションの数が多く、頻度が高くてシステムに対するオーバーヘッドが大きすぎる場合には、キュー・マネージャーのすべてのトピックでサブスクリプションを無効にすることができます。 個々のプロキシー・サブスクリプションを無効にする場合の詳細については、79 ページの『個々のプロキシー・サブスクリプションの無効化』を参照してください。

マルチキャストおよびサブスクリプションの待ち時間

サブスクリプションの待ち時間および PROXYSUB(FORCE) オプションを使用して、プロキシー・サブスクリプションを保守できます。

例えば、すべてのサブスクライバーが切断された後、QM_B から QM_A へのプロキシー・サブスクリプションが廃棄されるという問題が生じる可能性があります。 キュー・マネージャーへのユニキャスト接続が終了してもマルチキャスト・トラフィックを継続する必要がある場合、こうした状態は望ましくありません。 最後のサブスクライバーの終了時にプロキシー・サブスクリプションがすぐ廃棄されないように、WebSphere MQ のマルチキャストでは、新しいサブスクライバーが接続した場合、各プロキシー・サブスクリプションに数分の待ち時間を追加することにより、プロキシー・サブスクリプションが短時間維持されます。

また、トピックで PROXYSUB(FORCE) オプションを使用して、未解決のプロキシー・サブスクリプションが常に未解決であるようにできます。 サブスクリプションがアクティブである時間のほとんどで、キューの間で流れるメッセージが、1 つ以上のサブスクライバーによって必要とされていることを確認する必要があります。 PROXYSUB(FORCE) が設定されている場合、プロキシー・サブスクリプションは、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るプロキシー・サブスクリプションの前に送信される場合があり、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でも取り消されません。

サブスクリプションが引き続き廃棄される場合は、メッセージの転送が継続されるように対等通信を使用できます。詳しくは、マルチキャストの高可用性を参照してください。

ワイルドカードの規則
プロキシー・サブスクリプションでは、ワイルドカードはトピック・ワイルドカードを使用するように変換されます。

ワイルドカードのサブスクリプションを受け取る場合、そのワイルドカードには、WebSphere MQ バージョン 6.0 で使用する文字を使用できます。 これは、WebSphere Message Broker バージョン 6.0 および WebSphere MQ バージョン 7.0 で使用するトピックにすることもできます。

- 文字ワイルドカードでは、* を使用して任意の文字を表します (/ を含む)。
- トピック・ワイルドカードでは、# を使用して、/ 文字にはさまれているトピック・スペースの一部を表します。

WebSphere MQ バージョン 7.0 では、すべてのプロキシー・サブスクリプションはトピック・ワイルドカードを使用するように変換されます。 文字ワイルドカードが検出されると、その文字ワイルドカードは、直近の / の後で # 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

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するには、MQSC **Topic** WILDCARD プロパティーまたは同等の PCF トピック `WildcardOperation` プロパティーを使用します。 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。

44 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターの作成』 の例では、 40 ページの図 10 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。

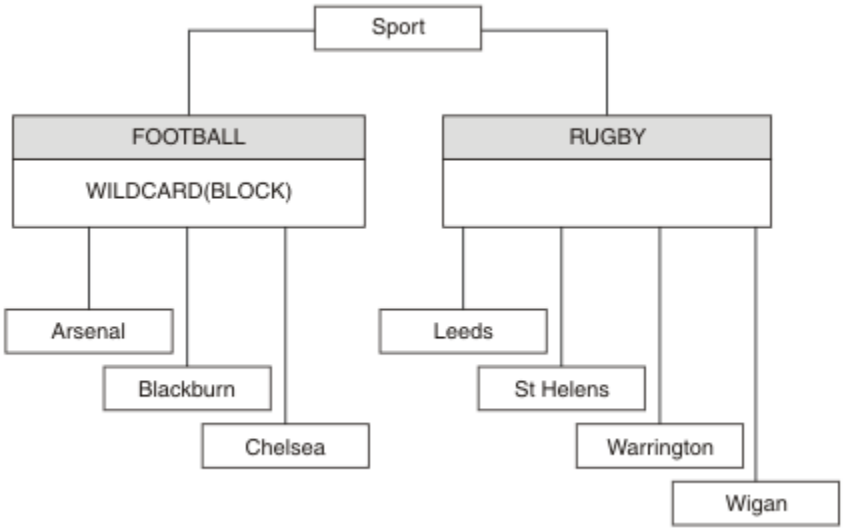


図 *23. WILDCARD* プロパティー *BLOCK* を使用するトピック・ツリー

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。 WILDCARD プロパティー値 PASSTHRU を Sport ツリー内のノードに設定できます。 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。 46 ページの図 14 を参照してください。 47 ページの図 17 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

結果を 41 ページの表 3 に示します。 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	記録 (Notes)
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	-	WILDCARD(BLOCK) on Sports/Football は、 Arsenal でワイルドカード・サブスクリプションを禁止します。
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 :

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・オブジェクトと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。 40 ページの図 10 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。

この例のクラスター・セットアップでは、 44 ページの『例: Sport パブリッシュ/サブスクライブ・クラスターの作成』 QMB には QMA と同じサブスクリプション・セットがありますが、 QMB は QMA にパブリッシュされたパブリッシャーの後にパブリケーションを受信しませんでした。 41 ページの図 11 を参照してください。 Sports/Football トピックと Sports/Rugby 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。 プロキシー・サブスクリプションは QMB から QMA に伝搬されません。 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 です。これにはトピック・オブジェクトがないため、 Sports/#/Leeds はローカル・トピック・オブジェクトである SYSTEM.BASE.TOPIC を継承します。

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 (41 ページの図 11 を参照してください)。

42 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。 43 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	記録 (Notes)
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	-	WILDCARD(BLOCK) on Sports/Football は、 Arsenal でワイルドカード・サブスクリプションを禁止します。
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。

43 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 は両方ともクラスター・トピックであるため、予期されるように、2 つの新しいサブスクリプションはパブリケーションを受け取ります。 QMB が Sports/Football/Arsenal および Sports/Rugby/Leeds のプロキシー・サブスクリプションを QMA に転送し、がパブリケーションを QMB に送信しました。

予期しない結果は、以前に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/# および Sports/#/Leeds が、パブリケーションを受信するようになったことです。 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**	**記録 (Notes)**
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	-	WILDCARD(BLOCK) on Sports/Football は、 Arsenal でワイルドカード・サブスクリプションを禁止します。
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

WebSphere MQ バージョン 6 のストリームは、WebSphere MQ バージョン 7 によってトピックにマップされます。 47 ページの『ストリームおよびトピック』を参照してください。 バージョン 7 の **strmqbrk** によって実行されるデフォルトのマッピングでは、 ストリーム Sports 内のすべてのトピックはトピック Sports にマップされます。 ストリーム Business 内のすべてのトピックは、トピック Business にマップされます。

WebSphere MQ バージョン 6 では、 Sports ストリーム内の * へのサブスクリプションは、Sports ツリー内のすべてのパブリケーションを受信し、 Business ツリー内のパブリケーションは受信しません。 バージョン 7 では、このサブスクリプションは Sports ツリー内のすべてのパブリケーション と Business ツリー内のすべてのパブリケーションを受信します。 この動作をブロックするために、ストリームがバージョン 7 にマイグレーションされると、**strmqbrk** は WILDCARD プロパティーを設定します。 このプロパティーは、ストリームから移行される各トップレベル・トピックについて、値 BLOCK に設定されます。 Sports および Business の WILDCARD プロパティーは、Sports および Business という名前のバージョン 6 ストリームからの変換により、値 BLOCK に設定されます。

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、* へのサブスクリプションはパブリケーションを受信しません。「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。

パブリッシュ/サブスクライブ・ブローカーを WebSphere MQ バージョン 7 に移行しても、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・アプリケーションの動作は変わりません。 **Publish**、**Register Publisher**、または **Subscriber** コマンドの **StreamName** プロパティーは、ストリームのマイグレーション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

WebSphere Message Broker サブスクリプション・ポイントは、 WebSphere MQ バージョン 7 によってトピックに マップされます (49 ページの『サブスクリプション・ポイントおよびトピック』を参照)。 バージョン 7 で **migmqbrk** によって実行されるデフォルトのマッピングでは、 サブスクリプション・ポイント Sports 内のすべてのトピックはトピック Sports にマップされます。 サブスクリプション・ポイント Business 内のすべてのトピックは、トピック Business にマップされます。

WebSphere Message Broker バージョン 6 において、 Sports サブスクリプション・ポイント内の * へのサブスクリプションは、 Sports ツリー内のすべてのパブリケーションを受信し、 Business ツリー内のパブリケーションは受信しません。 バージョン 7 の同じサブスクリプションは、 Sports ツリー内のすべてのパブリケーションと、Business ツリー内のすべてのパブリケーションを受け取ります。 この動作をブロックするため、サブスクリプション・ポイントがバージョン 7 に移行される際、 **migmqbrk** で WILDCARD プロパティーが設定されます。 このプロパティーは、サブスクリプション・ポイントから移行される各トップレベル・トピックについて、値 BLOCK に設定されます。 Sports と Business の WILDCARD プロパティーは、 Sports および Business と呼ばれる WebSphere Message Broker サブスクリプション・ポイントからの変換時に、 値 BLOCK に設定されます。

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。

パブリッシュ/サブスクライブ・ブローカーを WebSphere MQ バージョン 7 に移行しても、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・アプリケーションの動作は変わりません。 **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マイグレーション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・クラスターの作成

以下のステップでは、4 つのキュー・マネージャー (2 つの完全リポジトリー、 CL1A と CL1B 、および 2 つの部分リポジトリー、 QMA と QMB) を持つクラスター CL1 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
```

40 ページの図 10 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、40 ページの図 10 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、この例でさまざまなサブスクリプションをテストするために役立つプロパティーです。 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*

パブリッシュ/サブスクライブ・トポロジー

パブリッシュ/サブスクライブ・トポロジー は、パブリッシュ/サブスクライブ・アプリケーションをサポートする、複数のキュー・マネージャーとそれらのキュー・マネージャー間の接続で構成されています。

パブリッシュ/サブスクライブ・アプリケーションは、相互に接続されたキュー・マネージャーのネットワークで構成することができます。 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・マネージャーを相互に接続することによって、アプリケーションはネットワーク内のいずれかのキュー・マネージャーを使用して、パブリケーションを受け取ることができます。

これには、以下の利点があります。

- クライアント・アプリケーションは、遠くにあるキュー・マネージャーではなく近くにあるキュー・マネージャーと通信できるので、応答時間が短くなります。
- 複数のキュー・マネージャーを使用することにより、より多くのサブスクライバーをサポートできます。

クラスターと階層という 2 つの異なる方法によって、パブリッシュ/サブスクライブ・メッセージングを行うキュー・マネージャーを配置できます。 単純なクラスターおよび単純な階層の例については、65 ページの図 31 および 66 ページの図 32 を参照してください。 これら 2 つのトポロジーについて、また、最適なトポロジーを決定する方法について詳しくは、製品資料のこのセクションの情報を参照してください。

階層内でクラスターを相互に結合させることによって、両方のトポロジーを組み合わせて使用できます。

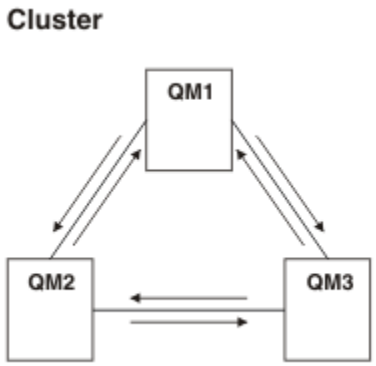


図 *31.* 単純なパブリッシュ/サブスクライブ・クラスター

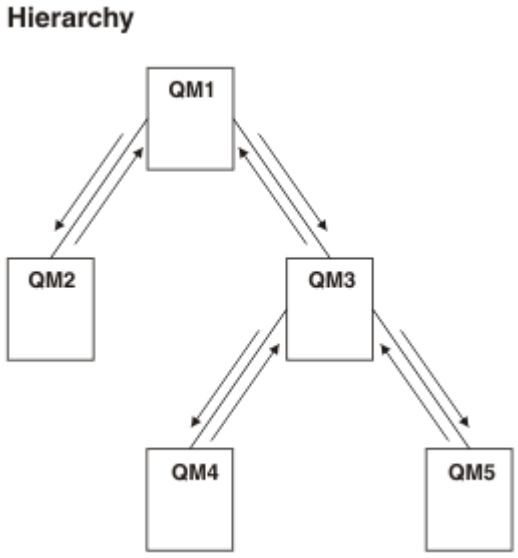
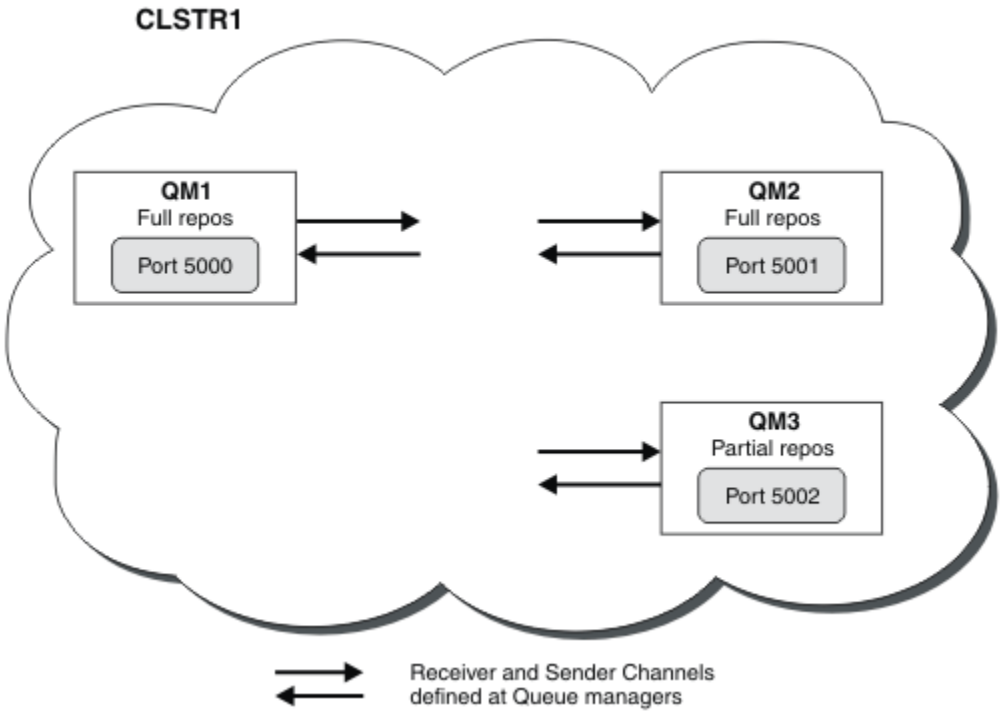


図 *32.* 単純なパブリッシュ/サブスクライブの階層

パブリッシュ/サブスクライブ・クラスターのセットアップ*:* シナリオ *1*
2 つのキュー・マネージャーを完全リポジトリーとしてクラスターに追加し、それらの間のチャネルを定義します。

このタスクについて

次の図は、3 つのキュー・マネージャー (QM1 、 QM2、および QM3) を示しています。



QM1 および QM2 はクラスター内の完全リポジトリーであり、 QM3 は部分リポジトリーです。

シナリオ 1 は、QM1 と QM2 を完全リポジトリーとしてクラスター DEMO に追加します。

シナリオ 2 は、QM3 を部分リポジトリーとしてクラスター DEMO に追加します。

これらの作業を行うために、少なくとも 1 つのコマンド・ウィンドウが必要です。

手順

1. QM1 と QM2 を DEMO クラスターの完全リポジトリーとして設定します。

```
alter QMGR REPOS(DEMO)
```

2. QM1 のリスナーを定義して開始します。

```
define listener(QM1_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5000)
start listener(QM1_LS)
```

3. QM2 のリスナーを定義して開始します。

```
define listener(QM2_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5001)
start listener(QM2_LS)
```

4. QM1 の受信側チャネルを定義します。

```
DEFINE CHANNEL(DEMO.QM1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')
CLUSTER(DEMO) DESCR('TCP Cluster-receiver channel for queue manager QM1')
```

5. QM1 から QM2 への送信側チャネルを定義します。

```
DEFINE CHANNEL(DEMO.QM2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5001)')
CLUSTER(DEMO) DESCR('TCP Cluster-sender channel from QM1 to  queue manager QM2')
```

6. QM2 の受信側チャネルを定義します。

```
DEFINE CHANNEL(DEMO.QM2) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5001)')
CLUSTER(DEMO) DESCR('TCP Cluster-receiver channel for queue manager QM2')
```

7. QM2 から QM1 への送信側チャネルを定義します。

```
DEFINE CHANNEL(DEMO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')
CLUSTER(DEMO) DESCR('TCP Cluster-sender channel from qm2 to qm1')
```

8. QM1 でクラスター・トピック scores を定義します。

```
define topic(scores) TOPICSTR(/football) CLUSTER(DEMO)
```

9. 以下のコマンドでセットアップを検証します。

```
display topic(scores) type(all) clusinfo
display clusqmgr(*)
display chstatus(*)
```

10. 2 つのコマンド・ウィンドウを使用してセットアップをテストします。
    a. 次のコマンドを一方のコマンド・ウィンドウに入力します。

```
/opt/mqm/samp/bin/amqspub /FOOTBALL/scores QM1
```

    b. 次のコマンドを他方のコマンド・ウィンドウに入力します。

```
/opt/mqm/samp/bin/amqssub /FOOTBALL/scores QM2
```

関連タスク

WebSphere MQ クラスターの管理
新規クラスターのセットアップ

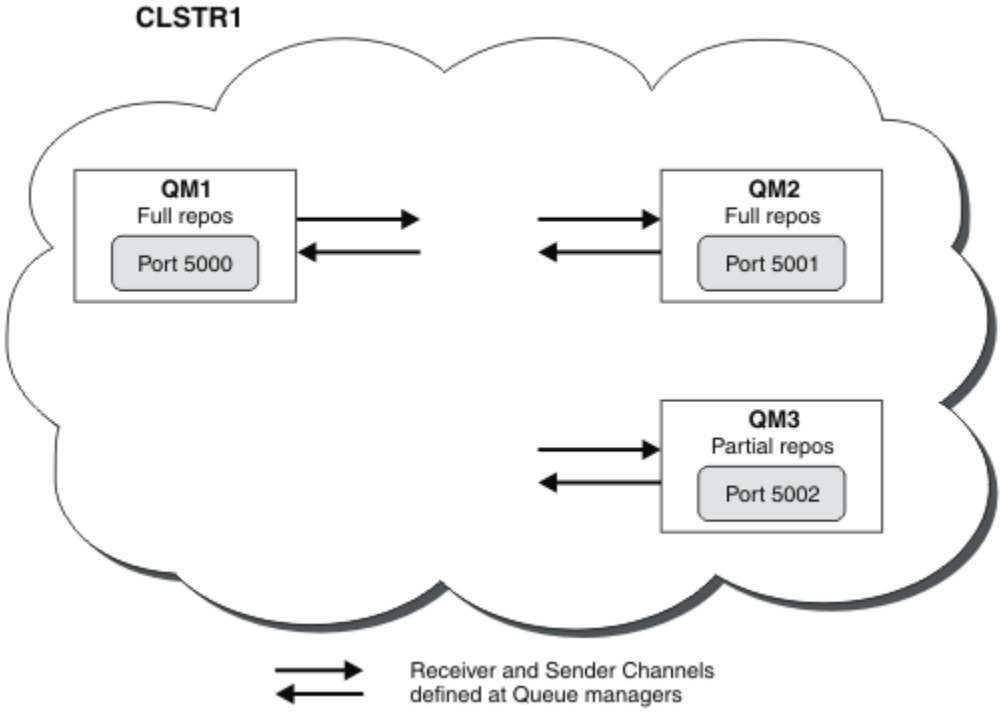
パブリッシュ/サブスクライブ・クラスターのセットアップ: シナリオ 2
3 つ目のキュー・マネージャーを部分リポジトリーとしてクラスターに追加します。

始める前に

この作業を終える前に、66 ページの『パブリッシュ/サブスクライブ・クラスターのセットアップ: シナリオ 1』 の作業を完了しておく必要があります。

このタスクについて

以下のダイアグラムでは、QM1、QM2、および QM3 という 3 つのキュー・マネージャーが使われています。



QM1 と QM2 はクラスター内の完全リポジトリーで、QM3 は部分リポジトリーです。

シナリオ 1 は、QM1 と QM2 を完全リポジトリーとしてクラスター DEMO に追加します。

シナリオ 2 は、QM3 を部分リポジトリーとしてクラスター DEMO に追加します。

これらの作業を行うために、少なくとも 1 つのコマンド・ウィンドウが必要です。

手順

1. QM3 のリスナーを定義して開始します。

```
define listener(QM3_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5002)
start listener(QM3_LS)
```

2. QM3 の受信側チャネルを定義します。

```
DEFINE CHANNEL(DEMO.QM3) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5002)') CLUSTER
(DEMO) DESCR('TCP Cluster-receiver channel for queue manager QM3')
```

3. QM3 から QM1 への送信側チャネルを定義します。

```
DEFINE CHANNEL(DEMO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')
CLUSTER(DEMO) DESCR('TCP Cluster-sender channel from qm3 to qm1')
```

4. 以下のコマンドでセットアップを検証します。

```
display topic(scores) type(all) clusinfo
display clusqmgr(*)
display chstatus(*)
```

5. 2 つのコマンド・ウィンドウを使用してセットアップをテストします。

a. 次のコマンドを一方のコマンド・ウィンドウに入力します。

```
/opt/mqm/samp/bin/amqspub /FOOTBALL/scores QM2
```

b. 次のコマンドを他方のコマンド・ウィンドウに入力します。

```
/opt/mqm/samp/bin/amqssub /FOOTBALL/scores QM3
```

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・クラスターは、相互接続されたキュー・マネージャーの標準の IBM WebSphere MQ クラスターであり、パブリッシュ元のアプリケーションから、クラスター内の任意のキュー・マネージャー上に存在するサブスクリプションに、パブリケーションを自動的に移動します。

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 WebSphere 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。 IBM WebSphere MQ クラスターの計画と構成の方法については、クラスターが機能する仕組みを参照してください。

パブリッシュ/サブスクライブ・クラスターは、クラスター・トピック・オブジェクトの定義時に、クラスター内のキュー・マネージャーによって構成されたトピックに **CLUSTER** 属性を設定することで、作成されます。 トピック定義はクラスターのすべてのメンバーに伝搬されます。 クラスター内の任意のキュー・マネージャー上のトピック、およびトピック・ツリー内で当該トピック下にある任意のトピック・ストリングに対し、パブリッシュとサブスクライブを行うことができます。 パブリケーションは、クラスター内の他のキュー・マネージャーに接続されたサブスクライバーに、自動的に伝搬されます。

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。 この構造は、すべてのサブスクリプションが階層全体にわたって伝搬されるパブリッシュ/サブスクライブ階層とは異なります。 いずれの場合も、サブスクリプションおよびパブリケーションの有効範囲を使用して、その制御性を向上させることができます。

パブリッシュ/サブスクライブ・トポロジーでクラスターを使用すると、以下の利点があります。

- 同一クラスター内の特定のキュー・マネージャー上のサブスクリプション宛のメッセージは、そのキュー・マネージャーに直接移送されるため、中間キュー・マネージャーを経由して移動する必要がありません。 これにより、階層トポロジーに比べてパフォーマンスが向上し、キュー・マネージャー間パブリッシュ/サブスクライブ・トラフィックが最適化されます。
- すべてのキュー・マネージャーは相互に直接接続されるため、このトポロジーには単一障害点が存在しません。 あるキュー・マネージャーが使用不可になっても、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は、使用可能なキュー・マネージャー上のパブリッシャーからメッセージを引き続き受け取ることができます。
- 複数の個別クラスターが存在するシステムでは (クラスターが地理的に分散している場合など)、クラスターを階層状に接続することができます。 各クラスターの単一キュー・マネージャーを結合することで、この接続を作成し、ネットワークを通過するパブリケーションとサブスクリプションのフローを可能にします (103 ページの『複数のクラスターのトピック・スペースの結合』を参照)。 あるクラスターから別のクラスターへ流れるパブリケーションを制御することもできます (104 ページの『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』を参照)。
- サブスクライブ・アプリケーションはそれに最も近いキュー・マネージャーに接続できるので、それ自身のパフォーマンスが向上します。 キュー・マネージャーは、クライアントのサブスクリプション登録に一致するすべてのメッセージを、クラスター内のすべてのキュー・マネージャーから受信します。

  このキュー・マネージャーに対して要求される他のサービスについても、クライアント・アプリケーションのパフォーマンスが向上します。 クライアント・アプリケーションは、パブリッシュ/サブスクライブと Point-to-Point メッセージングの両方を使用できます。
- キュー・マネージャーをクラスターにさらに追加してワークロードを共有することで、各キュー・マネージャーのクライアントおよびサブスクリプションの数を減らすことができます。 パブリケーションは、新規キュー・マネージャー上のクライアントに自動的に配布されます。 一部の使用パターンでは、このプロセスにより、パブリッシュ/サブスクライブ・クラスター・トポロジーが非常にスケーラブルなものになります。

パブリッシュ/サブスクライブでクラスターを使用する場合の考慮事項

- パブリッシュ/サブスクライブ・クラスター内のすべてのキュー・マネージャーは、クラスター内の他のすべてのキュー・マネージャーを自動的に認識します。 この処理は、あるキュー・マネージャーが対象とするキュー・マネージャーのみが認識される、Point-to-Point クラスターの場合とは異なります。
- クラスター・トピックに対する 1 つ以上のサブスクリプションをホストするパブリッシュ/サブスクライブ・クラスター内のキュー・マネージャーは、クラスター内の他のすべてのキュー・マネージャーに対してクラスター送信側チャネルを自動的に作成します。 またキュー・マネージャーは、受信側のキュー・マネージャーがクラスター・トピックに関するメッセージをパブリッシュしていなくても、クラスター内の各キュー・マネージャーに対してサブスクリプションの情報を送信します。
- クラスター・トピック下のトピック・ストリングに対するキュー・マネージャー上の最初のサブスクリプションでは、クラスター内の他のすべてのキュー・マネージャーにメッセージが送信されます。 同様に、削除対象のトピック・ストリングに対する最後のサブスクリプションでも、メッセージが送信されます。クラスター・トピック下で使用される個々のトピック・ストリングが増加すると、キュー・マネージャー間の通信も増加します。

注意 :

このトピックで前述した理由により、大規模な IBM WebSphere MQ クラスター (つまり、多数のキュー・マネージャーが存在するクラスター) にクラスター・トピックを導入すると、クラスター内の各キュー・マネージャーで追加の負荷が即座に発生し、パフォーマンスが低下する場合があります。詳しくは、77 ページの『クラスター・トピックのパフォーマンス』を参照してください。

キュー・マネージャーのクラスター (特に既存のクラスター) にパブリッシュ/サブスクライブを導入する際は、慎重に計画し、これらのパフォーマンス低下に対処する必要があります。

クラスターがパブリッシュ/サブスクライブのパフォーマンス低下に対処できない場合、**PSCLUS** パラメーターを使用して、キュー・マネージャー内のクラスター・パブリッシュ/サブスクライブ機能を無効にすることができます。 **PSCLUS** パラメーターは主に、パブリッシュ/サブスクライブ・クラスターの作成時に、クラスター・トピックを不注意でまたは間違って定義したために発生する可能性がある、重大な問題を停止するためのものです。 この機能を無効にする方法の詳細については、76 ページの『クラスター内でのクラスター化されたパブリッシュ/サブスクライブの禁止』を参照してください。

パブリッシュ/サブスクライブのクラスター化: ベスト・プラクティス
このトピックでは、 IBM WebSphere MQ パブリッシュ/サブスクライブ・クラスターの計画と管理について説明します。 この情報は、テストおよびお客様からのフィードバックに基づいています。

以下の情報は、ユーザーが IBM WebSphere MQ クラスター、パブリッシュ/サブスクライブの基礎を理解していること、および 51 ページの『分散パブリッシュ/サブスクライブ』のトピックを十分に理解していることを前提にしています。 この情報は「フリー・サイズ」ソリューション、すなわちすべてに適合できるソリューションとして意図されたものではなく、共通の問題に対しては共通のアプローチを共有しようとするものです。

パブリッシュ/サブスクライブ・クラスター

クラスターでは、必要に応じて、クラスター内のキュー・マネージャーの間で「任意対任意」の直接接続を使用します。 クラスターを Point-to-Point メッセージングに使用する場合、キュー・マネージャーに接続するアプリケーションがキュー・マネージャーの使用を要求するときに、クラスター内の各キュー・マネージャーには、他のクラスター・リソース (クラスター内の他のキュー・マネージャーやクラスター・キューなど) に関する情報のみが与えられます。つまり、キュー・マネージャーは、知る必要がある情報だけを知らされて機能します。

パブリッシュ/サブスクライブ・クラスターとは、通常の CLUSSDR チャネル定義と CLUSRCVR チャネル定義を持つ、キュー・マネージャーのクラスターです。 ただし、パブリッシュ/サブスクライブ・クラスターには、そのクラスター内の少なくとも 1 つのキュー・マネージャーで定義された 1 つ以上の TOPIC オブジェクトも含まれています。そのトピック・オブジェクトによってクラスター名が特定されます。

このクラスター内にトピック・オブジェクトが定義されていると、クラスター内のあるキュー・マネージャーに接続されているアプリケーションは、そのトピックまたはトピック・ツリー内の、そのトピックの下の任意のノードにサブスクライブし、クラスター内の他のキュー・マネージャーからそのトピックに関

するパブリケーションを受信することができます。 このプロセスを実現する方法は、サブスクリプションが存在するキュー・マネージャーを特定する、クラスター内の他のすべてのキュー・マネージャーでプロキシー・サブスクリプションを作成することです。 当該のトピックへのパブリケーションがそれぞれのキュー・マネージャーで発生すると、それらのキュー・マネージャーは、そのパブリケーションをクラスター内の該当する他のメンバーに転送することを認識し、そこから、パブリケーションが個々のアプリケーション・サブスクリプションに配信されます。

この配信を実現するためには、トピックがクラスターに追加されると同時に、クラスター内のすべてのキュー・マネージャーが、そのクラスター内の他のすべてのキュー・マネージャーの ID を知る必要があります。 この情報は、クラスターの完全リポジトリー・キュー・マネージャー経由で伝搬されます。 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、クラスター内の他のキュー・マネージャーのうち、同じトピックへのサブスクリプションをホストしていることが認識されているものへのみ送信されます。 このプロセスを実現するには、アプリケーションがクラスター・トピックへのサブスクリプションを作成するときには、該当するキュー・マネージャーがクラスター内の他のすべてのキュー・マネージャーと直接通信し、クラスター送信側チャネル経由でプロキシー・サブスクリプションを伝搬しなければなりません。

このプロセスは、クラスターを Point-to-Point の配信に使用するときに、知る必要のある限られた情報で接続する場合とは大幅に異なります。 したがって、パブリッシュ/サブスクライブ・クラスターの要件は、(クラスター・トピックを使用しない) Point-to-Point クラスターの要件とは異なります。

クラスター・トピックを使用すると、キュー・マネージャー間のパブリッシュ/サブスクライブ・ドメインを簡単に拡張できるようになりますが、その機構と影響が理解されておらず、パブリッシュ/ サブスクライブに使用されているクラスターについて考慮されていない場合には、問題につながる可能性があります。 以下に説明するベスト・プラクティスは、この理解と準備を支援することを目的としています。

要約すると、クラスター化されたパブリッシュ/サブスクライブのパフォーマンスへの影響は、大規模なクラスターに悪影響を与える可能性があるため、既存のクラスターでパブリッシュ/サブスクライブを使用しようとする前に、慎重に検討して理解する必要があります。 例えば、クラスター・トピック・オブジェクトを簡単に作成することもできます。 パブリッシュ/サブスクライブ・アクティビティーを専門とする小規模な新規クラスターから始め、そこからクラスターを拡大していく方が得策です。

パブリッシュ/サブスクライブ・トポロジーの設計

前述したように、クラスター内でパブリッシュ/サブスクライブを使用する際には、容量およびパフォーマンスに関する考慮事項があります。 したがって、ベスト・プラクティスとなるのは、キュー・マネージャー全体でのパブリッシュ/サブスクライブの必要性について慎重に検討すること、そしてパブリッシュ/サブスクライブを必要な数のキュー・マネージャーに制限することです。 トピックのセットに対してパブリッシュおよびサブスクライブする必要があるキュー・マネージャーの最小セットを特定したら、それらのキュー・マネージャーのみを含み、他のキュー・マネージャーは含まないクラスターを作成することができます。

このことは、すでに Point-to-Point メッセージングに有効に機能している確立済みのクラスターでは特に当てはまります。 そのため、既存の大規模クラスターをパブリッシュ/サブスクライブ・クラスターにする場合は、最初にパブリッシュ/サブスクライブ作業用に別のクラスターを作成し、現行のクラスターではなく、そこでアプリケーションを試すという方法をお勧めします。 1 つまたは複数の Point-to-Point クラスター内の既存のキュー・マネージャーを引き続き使用することも可能ですが、それらのキュー・マネージャーのサブセットを新しいパブリッシュ/サブスクライブ・クラスターのメンバーにする必要があります。 ただし、その新しいクラスターでは、既存のクラスターの完全リポジトリーから追加負荷を分離するため、完全リポジトリーとして構成した別個の複数のキュー・マネージャーが必要です。

サイズまたは現行の負荷を理由に、ある特定のクラスターをパブリッシュ/サブスクライブに使用しないことにした場合は、クラスター内の任意のキュー・マネージャーにクラスター・トピックを作成して、クラスターが予期せずにパブリッシュ/サブスクライブ・クラスターに変換されるのを防止することをお勧めします。 この設計を実現するには、**PSCLUS** キュー・マネージャー・プロパティーを使用します。詳しくは、クラスターでのクラスター化されたパブリッシュ/サブスクライブの禁止を参照してください。

クラスターに追加するトピックを慎重に選択することも重要です。 トピック・ツリー内でトピックが上位にあるほど、その影響は広範囲にわたります。 そのため、トピックの追加による動作を考慮せずに、トピックのルート・ノードをクラスターに追加することはお勧めしません。 グローバル・トピックは、可能な

限り明白にしてください。例えば、トピック・ストリングで /global や /cluster といった高位修飾子を使用します。

システムのサイズ変更の方法

パブリッシュ/サブスクライブ・クラスター・モデルは Point-to-Point メッセージングとは異なるため、パブリッシュ/サブスクライブ・クラスターには多数のチャネルが必要です。 各キュー・マネージャーが、そのクラスター内の他のすべてのキュー・マネージャーと対話する必要があります。 Point-to-Point モデルは「オプトイン」モデルですが、パブリッシュ/サブスクライブ・クラスターのサブスクリプション・ファンアウトには無差別という性質があります。 したがって、完全リポジトリー・キュー・マネージャーと、パブリッシュ/サブスクライブ・クラスター内のローカル・サブスクリプションをホストする任意のキュー・マネージャーに、クラスターのすべてのメンバーへのチャネルを同時に確立できるだけの容量が必要です。

パブリッシュ/サブスクライブ・クラスター内のすべてのキュー・マネージャーでこの容量が確保されるようにすることが最善策ですが、サブスクリプションを決してホストしないことがわかっているキュー・マネージャーは他のすべてのキュー・マネージャーとチャネルを確立する必要はないことが認知されているため、このレベルの容量は必要ありません。

ただし、そのようなキュー・マネージャーで誤ってサブスクリプションが作成されたり、クラスター内の他のキュー・マネージャーとの手動による再同期が試行されたりすると、すべてのチャネルが同時に開始されることになるので、注意が必要です。 詳しくは、72 ページの『プロキシー・サブスクリプションの再同期』を参照してください。

クラスター化されたパブリッシュ/サブスクライブにより、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を、別のキュー・マネージャー上のサブスクリプションに配信することが可能になります。 一方、Point-to-Point メッセージングの場合には、キュー・マネージャー間のメッセージの転送コストがパフォーマンスに悪影響を及ぼす可能性があります。 そのため、可能な限り、トピックへのサブスクリプションの作成は、メッセージがパブリッシュされたキュー・マネージャーと同じキュー・マネージャーで行うようにする必要があります。

その他の考慮事項には、プロキシー・サブスクリプションを伝搬するシステムのパフォーマンスへの影響もあります。 通常、キュー・マネージャーは、特定のクラスター・トピック・ストリングについて初めてサブスクリプションが作成されると、(構成されたトピック・オブジェクトだけでなく) プロキシー・サブスクリプション・メッセージをクラスター内の他のすべてのキュー・マネージャーに送信します。 パブリッシュ/サブスクライブ・ソリューションを構成する固有のサブスクライブ対象のトピック・ストリングが数多くある場合、あるいはトピックのサブスクライブやアンサブスクライブが頻繁に行われる場合には、クラスター内のすべてのキュー・マネージャーの間で大量のプロキシー・サブスクリプション・トラフィックが生成されて、システムのパフォーマンス全体に悪影響を及ぼす可能性があります。 プロキシー・サブスクリプションのオーバーヘッドを抑える方法については、77 ページの『クラスター・トピックのパフォーマンス』を参照してください。

プロキシー・サブスクリプションの再同期

通常の環境では、キュー・マネージャーが自動的に、システム内のプロキシー・サブスクリプションがクラスター内の各キュー・マネージャー上のサブスクリプションを正しく反映するようにします。

ただし、必要が生じた場合は、キュー・マネージャーのローカル・サブスクリプションと、クラスター全体に伝搬されているプロキシー・サブスクリプションを手動で再同期するために、REFRESH QMGR TYPE(PROXYSUB) コマンドを使用できます。

注 : 再同期を行うと、コマンドが発行されたキュー・マネージャーから、クラスター上に付加的なプロキシー・サブスクリプション負荷が突発的に発生します。 このため、 IBM WebSphere MQ サービス、 IBM WebSphere MQ 資料、またはエラー・ロギングで指示されない限り、これを使用しないでください。

再同期が必要になるのは、例えば、チャネルが停止してメッセージの一部を送信のためにキューに配置できないなどの理由により、キュー・マネージャーが自らのプロキシー・サブスクリプションを正しく伝搬できない場合や、オペレーター・エラーの結果として SYSTEM.CLUSTER.TRANSMIT.QUEUE キューからメッセージが誤って削除された場合などです。 この状況では、まず元の問題を修正してから (例えば、チャネルを再始動する)、キュー・マネージャーで **REFRESH QMGR TYPE(PROXYSUB)** コマンドを発行します。 ただし、プロキシー・サブスクリプションが存在しないために失われたパブリケーションは、影響を受けたサブスクリプションには復旧されません。 この欠点を考慮する必要があります。

再同期を行うには、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・マネージャーに対するチャネルを始動する必要があります。 このため、リフレッシュを行うキュー・マネージャーには、クラスター内の他のすべてのキュー・マネージャーとの通信を処理するのに十分の能力がなければなりません。

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、キュー・マネージャーごとに別個のトピック・スペースが作成されます。

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。

キュー・マネージャーごとに、そのキュー・マネージャーが認識するローカル・トピック定義およびクラスター・トピック定義から、1 つのトピック名前空間が構成されます。 クラスター・トピックとして解決されるトピックに対してアプリケーションがサブスクライブすると、IBM WebSphere MQ はプロキシー・サブスクリプションを作成し、サブスクリプションが作成されたキュー・マネージャーからクラスター内の他のすべてのメンバーに直接そのプロキシー・サブスクリプションを送信します。 クラスター・トピック自体とは異なり、プロキシー・サブスクリプションは完全リポジトリー・キュー・マネージャーを経由しません。

トピックに関してパブリッシュされたメッセージは、パブリッシャーが接続されたキュー・マネージャーに既知のすべてのサブスクリプションに送信されます。 これらのサブスクリプションのいずれかがプロキシー・サブスクリプションである場合、パブリッシュされたメッセージのコピーが、プロキシー・サブスクリプションを生成したキュー・マネージャーに送信されます。 それから、受信側キュー・マネージャーがメッセージのコピーをすべてのローカル・サブスクリプションに送信します。 このプロセスにより、クラスター・トピックのサブスクライバーは、クラスター内のいずれかのキュー・マネージャーに接続しているパブリッシャーからパブリケーションを受け取るようになり、また、パブリッシュされた最小数のメッセージがクラスターを介して伝搬されるようになります。

クラスター化されたトピックとローカル・トピック・オブジェクトがある場合は、ローカル・トピックが優先されます。 詳しくは、74 ページの『複数のクラスター・トピック定義』を参照してください。

クラスター・トピックの表示に使用するコマンドについて詳しくは、以下の関連リンクを参照してください。

ワイルドカード・サブスクリプション

プロキシー・サブスクリプションが作成されるのは、クラスター・トピック・オブジェクトまたはその下位で解決するトピック・ストリングに対してローカル・サブスクリプションが作成されるときです。 ワイルドカード・サブスクリプションがいずれかのクラスター・トピックよりも高いトピック階層で作成された場合、一致するクラスター・トピックのプロキシー・サブスクリプションがクラスター全体に送信されないため、キュー・マネージャーはクラスターの他のメンバーからのパブリケーションを受け取りません。ただし、ローカル・キュー・マネージャーからのパブリケーションは受け取ります。

一方、別のアプリケーションがクラスター・トピックまたはその下位のトピックに解決されるトピック・ストリングにサブスクライブすると、プロキシー・サブスクリプションが生成され、パブリケーションがこのキュー・マネージャーに伝搬されます。 オリジナルが到着すると、上位のワイルドカード・サブスクリプションがそれらのパブリケーションの正当な宛先であると見なされ、コピーを受け取ります。

この動作は、同じトピックに関してローカルにパブリッシュされるメッセージとは異なります。 この動作が不要な場合は、クラスター・トピックで **WILDCARD(BLOCK)** を設定すると、オリジナルのワイルドカード・サブスクリプションは、正当なサブスクリプションとは見なさなくなるので、クラスター・トピックまたはそのサブトピックに関するパブリケーションを (ローカル、またはクラスター内の別の場所から) 受け取りません。

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
DISPLAY TOPIC
DISPLAY TPSTATUS
DISPLAY SUB

クラスター・トピックの属性
パブリッシュ/サブスクライブ・クラスターを設計し管理する際には、クラスター・トピックの属性について十分に理解しておく必要があります。

トピック・オブジェクトには、マルチ・キュー・マネージャーのパブリッシュ/サブスクライブ・トポロジーに適用される多数の属性があります。 IBM WebSphere MQ クラスターを使用してそのようなトポロジーを作成している場合、これらの属性は以下のような振る舞いをします。

PROXYSUB

- **PROXYSUB** は、プロキシー・サブスクリプションの作成時期を制御する属性です。 これらの属性をデフォルト値 FIRSTUSE から変更する必要がある理由について詳しくは、55 ページの『ルーティング・メカニズムの詳細』を参照してください。
- クラスター・トピックの他の属性と同じように、**PROXYSUB** 属性は、トピックが定義されたキュー・マネージャーだけでなく、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。 ひいては、クラスター内のすべてのキュー・マネージャーで他のすべてのキュー・マネージャーに対するワイルドカード・プロキシー・サブスクリプションが作成されることになります。 このプロセスの結果、すべてのキュー・マネージャーで他のすべてのキュー・マネージャーへのクラスター送信側チャネルが作成され、パブリッシュされたあらゆるメッセージがすべてのキュー・マネージャーに送信されるようになります。

PUBSCOBE および SUBSCOPE
PUBSCOPE および **SUBSCOPE** により、このキュー・マネージャーがパブリケーションを、トポロジー (パブリッシュ/サブスクライブ・クラスターまたは階層) 内のキュー・マネージャーに伝搬するか、あるいはその有効範囲をそのローカル・キュー・マネージャーのみに制限するかが決まります。 MQPMO_SCOPE_QMGR / MQSO_SCOPE_QMGR を使用すると、これに相当するジョブをプログラムで実行できます。

- **PUBSCOPE PUBSCOPE(QMGR)** を指定してクラスター・トピック・オブジェクトを定義した場合、定義がクラスターと共有されますが、そのトピックに基づくパブリケーションの有効範囲はローカルのみであり、それらはクラスター内の他のキュー・マネージャーへ送信されません。
- **SUBSCOPE SUBSCOPE(QMGR)** を指定してクラスター・トピック・オブジェクトを定義した場合、その定義はクラスターと共有されますが、そのトピックに基づくサブスクリプションの有効範囲はローカルのみであり、プロキシー・サブスクリプションはクラスター内の他のキュー・マネージャーに送信されません。

これらの 2 つの属性は普通、特定のトピックに関してクラスターの他のメンバーと対話しないように、キュー・マネージャーを分離させるために一緒に使用されます。 そのキュー・マネージャーは、それらのトピックに関するパブリケーションを、クラスターの他のメンバーへパブリッシュせず、またそれらのメンバーから受け取ることもありません。 トピック・オブジェクトがサブトピックに定義されている場合、この状態ではパブリケーションもサブスクリプションも妨げられることはありません。

トピックのローカル定義で **SUBSCOPE** を QMGR に設定しても、クラスター内の他のキュー・マネージャーは、**SUBSCOPE(ALL)** が設定されているクラスター・バージョンのトピックを使用している場合には、プロキシー・サブスクリプションをキュー・マネージャーに伝搬できなくなることはありません。 ただし、ローカル定義でも **PUBSCOPE** が QMGR に設定されていると、キュー・マネージャーからパブリケーションがそれらのプロキシー・サブスクリプションに送信されなくなります。

複数のクラスター・トピック定義
ローカル・トピック定義は、リモートで定義された同じ名前のクラスター・トピック定義をオーバーライドします。 また、クラスター内の異なるキュー・マネージャーで同じクラスター・トピックの複数の定義

を作成することもできます。 ただし、それらのどちらのシナリオでも、若干の注意が必要であり、このトピックではその理由を説明します。

クラスター・キューと同様に、クラスター内で同じクラスター・トピック・オブジェクトに複数の定義があると、それぞれに関して異なるプロパティーが定義される可能性があります。 クラスター内の各キュー・マネージャーでどのバージョンのトピック定義が認識されるかを判別するのは容易ではなく、したがって、予期される動作を特定するのは困難です。

1 つのトピック・ストリングに対して複数のクラスター・トピック定義が異なる属性を持つか、複数のクラスターに存在する場合は、メッセージ (AMQ5465 & AMQ5466) がエラー・ログに書き込まれ、最近受信したクラスター・トピック定義が使用されます。

クラスター・トピック・ホストのキュー・マネージャーは、トピック定義を削除してはいけません。また、クラスター・トピックがクラスターのすべてのメンバーに認識され続けるように、クラスター内に残留します。 クラスター・トピック定義は、完全リポジトリーのキュー・マネージャー、および部分クラスター・リポジトリーの他のすべてのキュー・マネージャーによってキャッシュされているため、このキュー・マネージャーが常時使用可能である必要はありません。 このキャッシュにより、ホスト・キュー・マネージャーが使用不可な場合でも 60 日以上使用可能になります。 この件について詳しくは、81 ページの『パブリッシュ/サブスクライブ・クラスターのキュー・マネージャーの重要な役割』を参照してください。

ローカルでのクラスター・トピック定義のオーバーライド

クラスター内の特定のキュー・マネージャーでクラスター・トピックの動作をオーバーライドすることが必要になる場合もあります。 そのようなオーバーライドを実現するには、同じトピック・ストリングをもつクラスター・トピック・オブジェクトをオーバーライドするようにローカル・トピック・オブジェクトを定義します。それから、それを使用して、ローカルで接続されているサブスクライバーにのみパブリッシュされるようにします。

キュー・マネージャーでクラスター・トピックをオーバーライドするようにトピックのローカル定義が作成されている場合でも、キュー・マネージャーは、依然として、クラスター・トピック定義を使用してクラスターの他のメンバーからプロキシー・サブスクリプションを受け取ります。 デフォルトでは、ローカルにパブリッシュされたメッセージが引き続きリモート・キュー・マネージャーに送信され、プロキシー・サブスクリプションが受け入れられるようになります。 このような配置が不要な場合は、キュー・マネージャーに接続されているパブリッシャー・アプリケーションがローカル・サブスクライバーにのみパブリッシュするように、ローカル・トピック・オブジェクトで **PUBSCOPE**(QMGR) を指定してください。

クラスター・トピック定義の変更

クラスター・トピック定義を変更する必要がある場合、定義したのと同じキュー・マネージャー (クラスター・トピック・ホスト) で変更します。 クラスター内の別のキュー・マネージャーで同じクラスター・トピックの定義を作成しないでください。 トピックを再び定義すると、同じクラスター・トピックに 2 つのクラスター・トピック・ホストが存在することになります。

クラスター・トピックを複数回定義すると競合する定義が作成されてしまう場合があり、別々のキュー・マネージャーが別々のときに別々の定義を使用する可能性があります。

クラスター内の別のキュー・マネージャーへのクラスター・トピック定義の移動

クラスター・トピック定義をクラスター内のあるキュー・マネージャーから別のキュー・マネージャーに移動することが必要になる場合があります (例えば、キュー・マネージャーをクラスターで使用廃止にする場合など)。 パブリケーションのフローを中断せずに、クラスター内の別のキュー・マネージャーへクラスター・トピック定義を移動させるには、以下の手順に従う必要があります。 この例では、定義を QM1 から QM2 に移動します。

1. QM2 で、QM1 の定義と同じ属性を指定してクラスター・トピック定義の複製を作成します。
2. その新しい定義が完全リポジトリー・キュー・マネージャーによってクラスター全体に伝搬されるまで待ちます。 伝搬は、**(DISPLAY CLUSTER)** コマンドを使用して各クラスター・メンバー上のクラスター・トピックを表示し、 QM2 から発信された定義を確認することによって判別できます。
3. QM1 のクラスター・トピック定義を削除します。

元の定義が QM1 から削除された後は、プロパティーの競合を発生させることなく、必要に応じて QM2 で定義を変更することができます。

障害が発生したキュー・マネージャーでのクラスター・トピック定義の置換

前のシナリオでは、QM1 がしばらく使用できない場合に定義を QM1 から削除できない場合がありました。このシナリオでは、両方の定義が存在している状態で稼働することが許容されます。

後でクラスター・トピック定義を変更することが要件となった場合は、QM2 の定義は QM1 の定義よりも新しいので優先されることがわかっているため、QM2 のバージョンを変更することが可能です。 ただし、この期間、競合するクラスター・トピック定義が存在するので、キュー・マネージャーのエラー・ログにエラーが書き込まれます。 QM1 が再始動可能な場合は、QM1 から重複しているクラスター・トピック定義を削除して、できるだけ早くエラーを解消してください。

あるいは、QM1 が決してクラスターに戻されない場合 (例えば、壊滅的なハードウェア障害が発生した結果、予期せずに使用廃止になった場合など) は、RESET CLUSTER コマンドを使用して強制的にキュー・マネージャーを排除することができます。 **RESET CLUSTER** コマンドを実行すると、自動的に、ターゲット・キュー・マネージャーでホストされているすべてのトピック・オブジェクトが削除されます。

クラスター内でのクラスター化されたパブリッシュ/サブスクライブの禁止

キュー・マネージャーのクラスター (特に既存のクラスター) へのパブリッシュ/サブスクライブの導入は、パフォーマンスの低下に対応できるように注意深く計画する必要があります。

大規模な IBM WebSphere MQ クラスター (多数のキュー・マネージャーを含むもの) にクラスター・トピックを導入すると、クラスター内の各キュー・マネージャーへの負荷が直ちに増えることがあり、場合によってはパフォーマンスが低下する結果になることもあります。 そのため、パブリッシュ/サブスクライブの導入は注意深く計画する必要があります。 詳しくは、77 ページの『クラスター・トピックのパフォーマンス』を参照してください。

クラスターがパブリッシュ/サブスクライブのオーバーヘッドに対応できなかったことが分かっている場合は、キュー・マネージャー属性 **PSCLUS** を DISABLED に設定して、クラスター化されたパブリッシュ/サブスクライブ機能をキュー・マネージャーで使用不可にすることもできます。

PSCLUS を DISABLED に設定すると、キュー・マネージャーの機能の 3 つの面が変更されます。

- このキュー・マネージャーの管理者は、トピック・オブジェクトをクラスター化されたものとして定義できなくなります。
- 他のキュー・マネージャーから着信するトピック定義やプロキシー・サブスクリプションは拒否されます (構成が正しくないことを管理者に通知する警告メッセージがログに記録されます)。
- 完全リポジトリーがトピック定義を受け取ると、すべてのキュー・マネージャーに関する情報が他のすべての部分リポジトリーと自動的に共有されるということがなくなります。

PSCLUS はクラスター内の個々のキュー・マネージャーのパラメーターではあっても、クラスター内のキュー・マネージャーのサブセットでパブリッシュ/サブスクライブを選択的に無効にすることは意図されていません。 とりわけ、この方法ではプロキシー・サブスクリプションとトピック定義が表示されては拒否されるということが続き、エラー・メッセージが頻繁に表示されることになります。 このオプションを使用する場合は、クラスター内のすべてのキュー・マネージャーを一貫して使用不可に設定するのが理想的です。 キュー・マネージャーが 1 つ以上のパブリッシュ/サブスクライブ・クラスターだけでなく、1 つ以上の従来型のクラスターにも参加する場合は、そのキュー・マネージャー上で **PSCLUS** を ENABLED に設定する必要があります。 完全リポジトリーでの使用不可化については、以下の情報を参照してください。

重要な点として、クラスター内のすべての完全リポジトリー・キュー・マネージャーで **PSCLUS** を DISABLED に設定すると、正しく構成されていない部分リポジトリー上にクラスター化されたトピック定義があっても、それがクラスター内の他のキュー・マネージャーに影響を及ぼすことを未然に防ぐことができます。 このようなシナリオでは、完全リポジトリー・キュー・マネージャーのエラー・ログに不整合が報告されます。

従来の Point-to-Point クラスターとパブリッシュ/サブスクライブ・クラスターがオーバーラップする場合に大切なのは、クラスターごとに個別の完全リポジトリー・セットを使用することです。 このように調整

することによって、トピック定義と「すべてのキュー・マネージャー」情報がパブリッシュ/サブスクライブ・クラスター内でのみ流れるようになります。

このパラメーターの使用に関して、構成に矛盾が生じないようにするための注意点がいくつかあります。ENABLED から DISABLED に変更すると、このキュー・マネージャーがメンバーになっているどのクラスターにも、クラスター・トピック・オブジェクトを配置することはできなくなります。 このようなトピックがあるなら (リモートに定義されたトピックを含む)、この機能を無効にする前に削除しておく必要があります。

PSCLUS の詳細については、ALTER QMGR (PSCLUS) を参照してください。

クラスター・トピックのパフォーマンス
クラスター・トピックのパフォーマンス特性は、クラスター・キューのパフォーマンス特性とは異なるため、それらについては特別な配慮が必要です。十分に配慮せずに使用すると、大規模なクラスターまたはバランスの取れていないクラスターでパフォーマンス上の問題が発生する原因となる可能性があります。

パブリッシュ/サブスクライブのパフォーマンスへの影響の軽減

クラスター内のキュー・マネージャーでのワークロードには 2 つのソースがあります。1 つはアプリケーション・プログラムのメッセージの直接的な処理、もう 1 つはクラスターを管理するために必要なメッセージおよびチャネルの処理です。 典型的な Point-to-Point クラスターでは、クラスター・システム・ワークロードは、主に、必要に応じてクラスターのメンバーによって明示的に要求される情報に限定されます (80 ページの『パブリッシュ/サブスクライブ・クラスターのパフォーマンス特性』の対照例を参照)。 したがって、非常に大規模なクラスター (多数のキュー・マネージャーを含むクラスターなど) 以外の場合は、キュー・マネージャーのパフォーマンスを考慮する際に、パフォーマンスに対するクラスターの管理の影響の大部分を無視できます。

パブリッシュ/サブスクライブ・クラスターでは、すべてのクラスター・キュー・マネージャーがパブリッシュ/サブスクライブ・メッセージングにアクティブに参加しているかどうかに関わらず、クラスター・トピックやプロキシー・サブスクリプションなどの情報がクラスターのすべてのメンバーにプッシュされます。 このプロセスによって、システムに多大な負荷が生じる可能性があります。 したがって、時間とサイズの両方の点で、キュー・マネージャーのパフォーマンスに対するクラスター管理の影響を考慮する必要があります。

クラスターのパフォーマンスに対する、パブリッシュ/サブスクライブ・クラスター管理の影響を軽減するには、以下の 2 つの推奨事項を考慮してください。

1. クラスター、トピック、およびサブスクリプションの更新を、一日の内のオフピーク時に行う。
2. 単にそのクラスターが既に存在しているというだけの理由で、既存の大規模なクラスターにパブリッシュ/サブスクライブ・トピックを追加することを考えている場合は、パブリッシュ/サブスクライブに関与するキュー・マネージャーから構成されるより小規模なサブセットを定義して、それを「オーバーラップ」クラスターにすることができないか検討してください。 そのようにすると、このクラスターはクラスター・トピックが定義されるクラスターになります。 いくつかのキュー・マネージャーが 2 つのクラスターに置かれることになりますが、パブリッシュ/サブスクライブの全体的な影響は以下のように軽減されます。
   a. パブリッシュ/サブスクライブ・クラスターのサイズが小さくなります。
   b. パブリッシュ/サブスクライブ・クラスターにないキュー・マネージャーが受ける、クラスター管理トラフィックの影響がかなり小さくなります。

プロデューサーおよびコンシューマーのバランシング

非同期メッセージング・パフォーマンスの重要な概念はバランスを取ること です。 メッセージ・コンシューマーとメッセージ・プロデューサーとのバランスが取れていない場合、生じうる危険として、コンシュームされていないメッセージのバックログが増大し、複数のアプリケーションのパフォーマンスに深刻な影響を及ぼす場合があります。

Point-to-Point メッセージング・トポロジーでは、メッセージ・コンシューマーとメッセージ・プロデューサーの間の関係は理解しやすいといえます。 メッセージのプロダクションとコンシュームの見積もり (キ

ューごと、チャネルごと) を取得できます。 バランスが取れていない場合、ボトルネックを容易に識別して修正できます。

パブリッシュ/サブスクライブ・トポロジーで、パブリッシャーとサブスクライバーのバランスが取れているかどうかを分析することは、それよりも難しくなります。 クラスター・トピックに解決する各サブスクリプションから始めて、そのトピックに対するパブリッシャーを持つキュー・マネージャーを考慮します。 各キュー・マネージャーから各サブスクライバーに流れるパブリケーションの数を計算します。

クラスター内のリモート・キュー・マネージャーでのサブスクリプションと (プロキシー・サブスクリプションに基づき) 一致するパブリケーションのそれぞれは、SYSTEM.CLUSTER.TRANSMIT.QUEUE に書き込まれます。 そのパブリケーションのプロキシー・サブスクリプションが複数のリモート・キュー・マネージャーにある場合、それぞれに異なるクラスター送信側チャネルをターゲットとした複数のメッセージのコピーが伝送キューに書き込まれます。

これらのパブリケーションのターゲットは、リモート・キュー・マネージャー上の SYSTEM.INTER.QMGR.PUBS です。 各キュー・マネージャーは、そのキューに到着したメッセージを処理し、そのキュー・マネージャーにある正しいサブスクリプションにメッセージを配信します。

そのため、ボトルネックが発生する可能性のある以下のポイントでの負荷をモニターする必要があります。

- 個々のサブスクリプション・キュー自体:
  - このボトルネックは、サブスクライブ側アプリケーションによるコンシュームが、パブリケーションのパブリッシュより遅いスピードで行われていることを意味しています。
- SYSTEM.INTER.QMGR.PUBS キュー:
  - キュー・マネージャーが、ローカル・サブスクリプションに配信できるよりも速いスピードで、1 つまたは複数のリモート・キュー・マネージャーからパブリケーションを受け取っています。
- パブリッシュ側キュー・マネージャーとサブスクライブ側キュー・マネージャーの間のクラスター・チャネル、およびパブリッシュ側キュー・マネージャー上のクラスター伝送キュー (デフォルトでは SYSTEM.CLUSTER.TRANSMIT.QUEUE):
  - 1 つ以上のクラスター・チャネルが稼働していないか、または、チャネルがリモート・キュー・マネージャーに配信できるよりも速いスピードでメッセージがローカル・キュー・マネージャーにパブリッシュされています。
- パブリッシング・アプリケーションがキューに入れられたパブリッシュ/サブスクライブ・インターフェースを使用している場合、 SYSTEM.QPUBSUB.QUEUE.NAMELIST にリストされている SYSTEM.BROKER.DEFAULT.STREAM キューとその他のストリーム・キュー、および SYSTEM.QPUBSUB.SUBPOINT.NAMELIST にリストされている SYSTEM.BROKER.DEFAULT.SUBPOINT キューとその他のサブポイント・キューも考慮する必要があります。
  - ローカル・キュー・マネージャーが処理できるよりも速いスピードでメッセージがローカルのパブリッシュ側アプリケーションによって送信されています。

サブスクリプション・パフォーマンスの考慮事項

前に説明したように、クラスター・トピックとして解決されるトピック・ストリングに対するサブスクリプションがキュー・マネージャーで行われた場合、そのキュー・マネージャーは、クラスター内の他のすべてのキュー・マネージャーにそのトピックのプロキシー・サブスクリプションが確実に配置されるようにする必要があります。 この結果を達成するためには、キュー・マネージャーがプロキシー・サブスクリプション・メッセージを作成し、そのメッセージをクラスター内の他のすべてのキュー・マネージャーに送信しなければなりません。

デフォルト構成を使用して、クラスター・トピックに対するサブスクリプションの作成によって新しいプロキシー・サブスクリプションが送信されることがないのは、ローカル・キュー・マネージャーに既にまったく同じトピック・ストリングに対するサブスクリプションがある場合のみです。 この場合、追加のプロキシー・サブスクリプションは必要ありません。到着するパブリケーションは、トピック・ストリングに対する元のサブスクリプションに配信されるだけでなく、一致するすべてのサブスクリプションにも配信されるためです。

デフォルトの代わりとなる構成については、79 ページの『個々のプロキシー・サブスクリプションの無効化』を参照してください。

サブスクリプション・セレクターは考慮されないので、2 つのサブスクリプションは、対象のトピック・ストリングが同じでも、セレクターが異なるため、依然としてプロキシー・サブスクリプションを共有します。 また、そのような状況は、パブリケーションがサブスクリプションのセレクターに一致していなくても、トピック・ストリングに一致するパブリケーションがサブスクライバー・キュー・マネージャーに伝搬されることを意味している可能性があります。

トピック・ストリングに対する最後のサブスクリプションがキュー・マネージャーから削除されると、プロキシー・サブスクリプション・メッセージに相当するメッセージが作成され、すべてのキュー・マネージャーに送信されます。 このプロセスにより、プロキシー・サブスクリプションはリモート・キュー・マネージャーから削除されます。

以上の理由から、クラスターのサイズ、そしてさまざまなトピック・ストリングに対するサブスクリプションの頻度は、クラスター自体に大きな負荷を与える可能性があるため、クラスターおよびパブリッシュ/サブスクライブ・アプリケーションで使用するトピックを計画する際には、これらの事項を考慮しなければなりません。

プロキシー・サブスクリプション・トラフィックによるシステムの負荷を考慮する際には、77 ページの『プロデューサーおよびコンシューマーのバランシング』セクションにリストされているキューの他、以下のキューについてもモニターしてください。

- サブスクライバー・キュー・マネージャー上の SYSTEM.INTER.QMGR.FANREQ キュー
- クラスター内の他のすべてのキュー・マネージャー上の SYSTEM.INTER.QMGR.CONTROL キュー

上記のキューに大量のメッセージ・バックログがある場合、サブスクリプションの変更率がシステムにとって大きすぎるか、またはキュー・マネージャーがクラスター内で正常に機能していないことを意味します。 パブリッシュ/サブスクライブ・サポートが使用不可になっていることが原因か (ALTER QMGR の **PSMODE** を参照)、あるいはさらに詳しい調査が必要な問題が発生しているのかという点で、キュー・マネージャーのエラー・ログを調べてください。

プロキシー・サブスクリプション・トラフィックの削減

プロキシー・サブスクリプション・オーバーヘッドが高い場合は、それを低くするための手順を実行する必要があります。 これは、一般的なトピック統合を行うか、キュー・マネージャー間パブリケーションをブロードキャスト・モデルに変更することによって実行できます。

パブリッシュ/サブスクライブに関する一般的な推奨事項は、システムのリソースに関する全体的な負荷が軽減されるようにトピック・ストリングをまとめることができるかどうかを確認するため、トピック・ストリングの使用状況を評価することです。 多数の別個の一時的トピック・ストリングを使用すると、パブリッシャーまたはサブスクリプションが接続しているシステム内の各キュー・マネージャーで何らかのレベルの管理オーバーヘッドが発生します。 トピック・ストリングの数を減らし、一時的な性質をもつトピック・ストリングの使用を避けることで、トピック・ストリングに対するパブリッシャーおよびサブスクリプションが削減され、システムへの影響が軽減されます。

プロキシー・サブスクリプション・トラフィックを削減する方法の 1 つは、同じキュー・マネージャーの同じトピック・ストリングにサブスクリプションを配置することです。 この方法を使用すると、複数のキュー・マネージャーでプロキシー・サブスクリプションを送信する (それぞれ、同じトピック・ストリングに関する独自のサブスクリプション・セットを送信する) のではなく、同じキュー・マネージャーで単一のプロキシー・サブスクリプションを送信できるようになります。 また、この手法により、クラスター全体のパブリケーションの経路指定も最適化されます。

個々のプロキシー・サブスクリプションの無効化

状況によっては (例えば、クラスター全体にサブスクライブされている個別のトピック・ストリングのセットが大きく、頻繁に変更される場合など)、サブスクリプション伝搬モデルからパブリケーション・ブロードキャスト・モデルに変更する方が望ましいことがあります。 この推奨モデルでは、任意のクラスター・トピックに関するあらゆるパブリケーションが自動的にクラスター内のあらゆるキュー・マネージャーに送信されます。それらのキュー・マネージャーにサブスクリプションが存在するかどうかは考慮されません。

そのため、受信側キュー・マネージャーは、ローカル・サブスクリプションが存在すれば、それらのサブスクリプションにメッセージを配信することができ、存在しなければ、メッセージを破棄することができます。 このモデルでは、個々のプロキシー・サブスクリプションを作成し、サブスクリプションが存在するかどうかに基づいて削除する必要はありません。 このモードで稼働した場合、すべてのパブリケーションがすべてのキュー・マネージャーに送信されるので、パブリッシュされるメッセージのリソース負荷が増加する可能性があります。 したがって、クラスター内のキュー・マネージャーには、そのような追加負荷を処理するキャパシティーが必要です。

ブロードキャスト・モデルを有効にするには、以下の構成ステップを実行します。

1. ローカル・サブスクリプションに一致するプロキシー・サブスクリプションをクラスター・トピックに対して送信しないようにサブスクリプションをホストしているすべてのキュー・マネージャーを構成します。 この構成では、クラスターでクラスター・トピックを定義したり、サブスクリプションを作成したりする前に、各キュー・マネージャーの qm.ini ファイルで以下のチューニング・パラメーターを設定する必要があります。

   ```
   TuningParameters:
       pscProxySubFlags=1
   ```

2. チューニング・パラメーターを設定したら、すべてのキュー・マネージャーを再始動します。
3. キュー・マネージャーが再始動したら、クラスター・トピックを定義することができます。 各クラスター・トピックで **PROXYSUB** を FORCE に設定します。

動作を元に戻す方法

79 ページの『個々のプロキシー・サブスクリプションの無効化』で説明した動作モードを元に戻すには、以下のステップを実行します。

1. すべてのキュー・マネージャーの qm.ini ファイルからチューニング・パラメーターを削除します。
2. すべてのキュー・マネージャーを再始動します。
3. サブスクリプションをホストしているすべてのキュー・マネージャーで **REFRESH QMGR TYPE(PROXYSUB)** コマンドを発行します。
4. クラスター・トピックの **PROXYSUB** を FIRSTUSE に設定します。

注意 : この動作を有効にする場合も、元に戻す場合も、すべてのステップを記述されている順序どおりに完了させないと、パブリケーションからサブスクリプションへの正しいフローが提供されない可能性があります。

注: PROXYSUB を (FORCE に) 設定したときの影響

このトピックで既に説明したように、**PROXYSUB(FORCE)** トピック属性はプロキシー・サブスクリプション・トラフィックを削減するのに役立ちますが、注意して使用する必要があります。 **PROXYSUB(FORCE)** 属性は、トピックが定義されたキュー・マネージャーだけでなく、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。 ひいては、クラスター内のすべてのキュー・マネージャーで他のすべてのキュー・マネージャーに対するワイルドカード・プロキシー・サブスクリプションが作成されることになります。 このプロセスの結果、すべてのキュー・マネージャーで他のすべてのキュー・マネージャーへのクラスター送信側チャネルが作成され、パブリッシュされたあらゆるメッセージがすべてのキュー・マネージャーに送信されるようになります。

このプロパティーを大規模なクラスターまたはトラフィック量の多いクラスターで設定すると、システム・リソースに負荷が加わる可能性があります。

パブリッシュ/サブスクライブ・クラスターのパフォーマンス特性

パブリッシュ/サブスクライブ・クラスターの属性の変更 (クラスターへのキュー・マネージャー、トピック、またはサブスクリプションの追加など) が、クラスター内で実行されているアプリケーションのパフォーマンスにどのような影響を与えるかを考慮することは重要です。

2 つの管理作業の観点から、Point-to-Point クラスターとパブリッシュ/サブスクライブ・クラスターを比較します。

最初に、Point-to-Point クラスターです。

1. 新しいクラスター・キューが定義されると、宛先情報が完全リポジトリー・キュー・マネージャーにプッシュされ、他のクラスター・メンバーには、それらのメンバーが最初にそのクラスター・キューを参照したとき (例えば、アプリケーションがそのクラスター・キューを開こうとしたときなど) にのみ送信されます。 この情報は、キューにアクセスするたびに情報をリモートから取得する必要を取り除くために、キュー・マネージャーによってキャッシュに入れられます。
2. クラスターへのキュー・マネージャーの追加は、他のキュー・マネージャーの負荷には直接影響しません。 新しいキュー・マネージャーに関する情報は完全リポジトリーにプッシュされますが、クラスター内の他のキュー・マネージャーから新しいキュー・マネージャーへのチャネルが作成され開始されるのは、新しいキュー・マネージャーとのトラフィックが流れ始めてからです。

つまり、Point-to-Point クラスターでのキュー・マネージャーの負荷は、アプリケーション・プログラム用に処理されるメッセージ・トラフィックと関係があり、クラスターのサイズとは直接的な関係はありません。

次に、パブリッシュ/サブスクライブ・クラスターです。

1. 新しいクラスター・トピックが定義されると、その情報は、完全リポジトリー・キュー・マネージャーにプッシュされ、即座にそこからクラスターのすべてのメンバーへと直接プッシュされます。その結果、完全リポジトリーからクラスターの各メンバーへのチャネルが開始されます (まだ開始されていない場合)。
2. 新しいトピック・ストリングに関するクラスター・トピックに対するサブスクリプションが作成されると、その情報は、即座にキュー・マネージャーからクラスターの他のすべてのメンバーへと直接プッシュされます。その結果、そのキュー・マネージャーからクラスターの各メンバーへのチャネルが開始されます (まだ開始されていない場合)。
3. 新しいキュー・マネージャーが既存のクラスターに加わると、すべてのクラスター・トピックに関する情報が完全リポジトリー・キュー・マネージャーからそのキュー・マネージャーにプッシュされます。すると、その新しいキュー・マネージャーは、クラスター内のクラスター・トピックに対するすべてのサブスクリプションのナレッジをクラスターのすべてのメンバーのものと同期させます。その結果、新しいキュー・マネージャーからクラスターの各メンバーへのチャネルが作成され、開始されます。

つまり、クラスター内のいずれのキュー・マネージャーでも、クラスター管理の負荷は、そのクラスター内のキュー・マネージャー、クラスター・トピック、およびプロキシー・サブスクリプションの数に応じて増加します。各キュー・マネージャーでそれらのクラスター・トピックがローカルに使用されるかどうかは関係ありません。

パブリッシュ/サブスクライブ・クラスターのキュー・マネージャーの重要な役割
Point-to-Point クラスターと同様に、パブリッシュ/サブスクライブ・クラスターのキュー・マネージャーには、完全リポジトリー・キュー・マネージャーとクラスター・トピック・ホストという 2 つの重要な役割があります。

フル・リポジトリー
完全リポジトリーのキュー・マネージャーには、オブジェクト定義をクラスターの他のメンバーにプッシュするという役割があります。パブリッシュ/サブスクライブ・クラスターの場合、クラスター・トピック・オブジェクト定義をクラスターの他のメンバーにプッシュします。

クラスター・トピック・ホスト
クラスター・トピック・ホストは、クラスター・トピック・オブジェクトが定義されているキュー・マネージャーです。 クラスター・トピック・オブジェクトは、パブリッシュ/サブスクライブ・クラスターの任意のキュー・マネージャーで定義できます。 クラスター・トピック・オブジェクトは完全リポジトリー・キュー・マネージャーにプッシュされ、そこからクラスター内の他のすべてのキュー・マネージャーへプッシュされます。そこでキャッシュに入れられ、クラスター内のすべてのキュー・マネージャー上で実行されているパブリッシャーおよびサブスクライバーが使用できるようにされます。

可用性および管理

クラスター内でのクラスター・トピック定義の可用性を最大限にするために、クラスター内に 2 つの完全リポジトリーを定義してください。

キュー型メッセージング・クラスターの場合と同じく、多くのコンピューターの中で高可用性コンピューターが 2 つだけ含まれているパブリッシュ/サブスクライブ・クラスターでは、それら 2 つの高可用性コンピューターを完全リポジトリーとして定義するのが賢明な方法です。

キュー・クラスターでは、クラスター内の複数のキュー・マネージャーで同じクラスター・キューを定義する ことにより、クラスター・キューの可用性およびスループットを向上させることができます。 その結果、クラスター全体でのメッセージのワークロード・バランシングが実現されます。 一方、パブリッシュ/サブスクライブ・クラスターでは、クラスター・トピックはクラスター内のすべてのキュー・マネージャーで使用可能 ですが、パブリッシュ/サブスクライブ・トラフィックのワークロード・バランシングは行われません。 その代わりに、個々のサブスクリプションおよびパブリッシャーが別々のキュー・マネージャーに分散されるので、パブリッシュ/サブスクライブの負荷も分散されます。 クラスター・トピックを定義したキュー・マネージャーが使用不可になると、他のキュー・マネージャーが、そのトピックのパブリッシュ/サブスクライブ要求の処理を継続します。

ただし、クラスター・トピック・オブジェクトを定義したキュー・マネージャーが再び使用可能になることがない場合、他のキュー・マネージャー上のキャッシュに入れられたトピック・オブジェクトは最終的に削除され、そのトピックは使用不可になります。 このプロセスは、トピック定義が使用不可になってから少なくとも 60 日後 (トピック定義が最後にリフレッシュされた時点に依存) に行われます。

クラスター・トピック・オブジェクトを定義したキュー・マネージャーの回復期間は 60 日あるため、クラスター・トピック・ホストの可用性を高めるために特別な措置を高じる必要はほとんどありません。 60 日間という期間は、技術上の問題に対処するのに十分な期間です。60 日間を超えてしまう理由として考えられるのは、管理上のエラーのみです。 この可能性を低減させるために、 クラスター・トピック・ホストが使用不可である場合は、クラスターのすべてのメンバーが、 キャッシュに入れられたクラスター・トピック・オブジェクトがリフレッシュされていないことを示すエラー・ログ・メッセージを 1 時間ごとに書き込みます。 クラスター・トピック・オブジェクトが定義されているキュー・マネージャーが稼働していることを確認することで、 このメッセージに応答する必要があります。

他のキュー・マネージャーで同じクラスター・トピック・オブジェクトを定義するという手法を採用することもできます。 それぞれの定義により、追加のクラスター・トピック・オブジェクトがクラスター内の他のキュー・マネージャー (他のクラスター・トピック・ホストを含む) にプッシュされます。 これで、クラスター・トピック・ホストが 60 日間使用不可になっても、他のホストから削除されるのは、そのホストからキャッシュされたクラスター・トピック・オブジェクトだけです。 他のホストからキャッシュされたクラスター・トピック・オブジェクトはそのまま残ります。 クラスター内の特定トピックに対するすべての定義は同一であることが求められます。同一ではない場合、キュー・マネージャーによって使用されているトピック定義の確定が困難になります。 どのホストにあるものであれ最新のコピーが常に、使用されるクラスター・トピック・オブジェクトです。

複数のクラスター・トピック定義によって保護が強化されることと、管理がもっと複雑になることとを比較検討してください。複雑さが増すと、人的なエラーが生じやすくなります。

クラスター・キューをホストする場合とは異なり、クラスター・トピック定義のホスト・キュー・マネージャーであることが、追加のアプリケーション・メッセージ・トラフィックをもたらすことはありません。このトラフィックは、サブスクリプションの作成およびメッセージのパブリッシュが行われるキュー・マネージャーに限られます。 このどちらも行わないキュー・マネージャーでクラスター・トピックをホストすることは可能です。 このような状況は、クラスター・トピックをクラスターの完全リポジトリー・キュー・マネージャーでホストすることが、必須ではないものの妥当である場合が多いということを意味しています。なぜならば、これらのキュー・マネージャーは、高いレベルの可用性が実現されるようにプロビジョンされ、クラスター・トピックをより厳しく管理制御できるものと思われるからです。 このような配置を使用することで、定義や、キュー・マネージャーまでも、誤って変更または削除される可能性が低くなります。

クラスターのオーバーラップのサポートとパブリッシュ/サブスクライブ

IBM WebSphere MQ クラスターでは、1 つのキュー・マネージャーを複数のクラスターのメンバーにすることができます。 この配置は、オーバーラップするクラスターとして知られています。 キュー・マネージャーでクラスターがオーバーラップしている場合、キューに対するパブリッシュ/サブスクライブ・クラスター内のクラスター・トピックの動作はさまざまです。 オーバーラップするクラスターと一緒にクラスター・パブリッシュ/サブスクライブを使用する場合には、そのような動作について明確に理解する必要があります。

キューの場合とは異なり、トピック定義を複数のクラスターに関連付けることはできません。 したがって、クラスター内で作成されたプロキシー・サブスクリプションの有効範囲は、クラスター・トピックが定義されている単一のクラスターに限定されます。 ただし、各キュー・マネージャーには、それらのキュー・マネージャーがメンバーになっているクラスターのすべてのローカル・トピックおよびあらゆる既知のクラスター・トピックが含まれている単一のトピック・ツリーがあります。 そのため、パブリッシュ/サブスクライブの動作を把握することが困難なシステムが構築される可能性があります。

複数のパブリッシュ/サブスクライブ・クラスターの統合

Point-to-Point メッセージに関し、1 つのキュー・マネージャーを複数のクラスターのメンバーにするのは、2 つのクラスター間でクラスター・ゲートウェイを作成するためです。 この件について詳しくは、 クラスターのオーバーラップ を参照してください。 このクラスター・ゲートウェイを使用すると、あるクラスターから発信される Point-to-Point メッセージを別のクラスターでの照会に対して経路指定することができます。 パブリッシュ/サブスクライブ・クラスターは、従来のキュー・マネージャーからオーバーラップ機能を継承します。 ただし、あるクラスターから別のクラスターにパブリケーションおよびサブスクリプションを経路指定するためにこのメカニズムを使用することはできません。

その代わりに、あるクラスター内のキュー・マネージャーから別のクラスター内のキュー・マネージャーにパブリケーションやサブスクリプションを渡すため、パブリッシュ/サブスクライブ階層を使用して、それらのキュー・マネージャーをリンクする必要があります。 この配置を実現するには、一方のクラスター内のキュー・マネージャーと他方のクラスター内のキュー・マネージャーとの間に親子としての階層関係を明示的に作成します。 この関係により、クラスター間でのすべてのプロキシー・サブスクリプションのフローが可能になり、したがって、一致するパブリケーションのフローも可能になります。 この関係について詳しくは、85 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。

クラスター間のパブリケーションとサブスクリプションのフローを制限する方法の 1 つは、どちらのクラスターにも含まれていないゲートウェイ・キュー・マネージャーを使用することです (104 ページの『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』を参照)。

オーバーラップするクラスター、単一のトピック・ツリー

各キュー・マネージャーには、ローカル・トピックおよび既知のクラスター・トピックが含まれている単一のトピック・ツリーがあります。 オーバーラップするクラスターが、どちらもパブリッシュ/サブスクライブを使用する場合には、さらに考慮しなければならないことがあります。それは、それぞれのクラスター内のキュー・マネージャーが、同じ名前のクラスター・トピックを定義する可能性、あるいは同じトピック・ストリングを持つクラスター・トピックに異なる名前を定義する可能性があることです。 両方のクラスターのメンバーとなっているキュー・マネージャーでは、クラスターごとに異なる複数のクラスター・トピック定義を通知されると、競合が発生します。 キュー・マネージャーは問題を報告するものの、最新のクラスター・トピック定義のみを使用して動作し続けます。 そのため、動作は非決定的となり、依存できなくなります。

このことから、オーバーラップするクラスターでクラスター・パブリッシュ/サブスクライブを使用している場合には、トピック定義の名前空間がすべてのクラスターにまたがるものとなるようにすることを検討する必要があり、しかるべく、トピック・オブジェクトの名前を指定し、トピック・ストリングを構成します。 これにより、オーバーラップに含まれるキュー・マネージャーを使用して、予測可能な方法で両方のクラスターにパブリッシュ/サブスクライブすることができます。

84 ページの図 33 で、T_B および T_C はオーバーラップしないトピック定義です。 クラスターのオーバーラップに含まれる QM3 に接続するパブリッシャーは、それぞれのクラスターで両方のトピックにパブリッシュできます。 オーバーラップに含まれる QM3 に接続するサブスクライバーは、両方のクラスターのトピックにサブスクライブできます。

84 ページの図 33 を説明する別の方法として、プロキシー・サブスクリプションを考えます。 アプリケーションがキュー・マネージャー QM3 に接続され、トピック・オブジェクト T_B (CLUSTER 1 にのみ存在) に解決されるトピックにサブスクライブすると、プロキシー・サブスクリプションがキュー・マネージャー QM3 からキュー・マネージャー QM1 および QM2 にのみ送信されます。 キュー・マネージャー QM3 に接続するアプリケーションは、CLUSTER 2 のみに存在するトピック・オブジェクト T_C に解決されるトピックにサブスクライブします。 このサブスクリプションの結果、プロキシー・サブスクリプションがキュー・マネージャー QM3 からキュー・マネージャー QM4 と QM5 に送信されます。

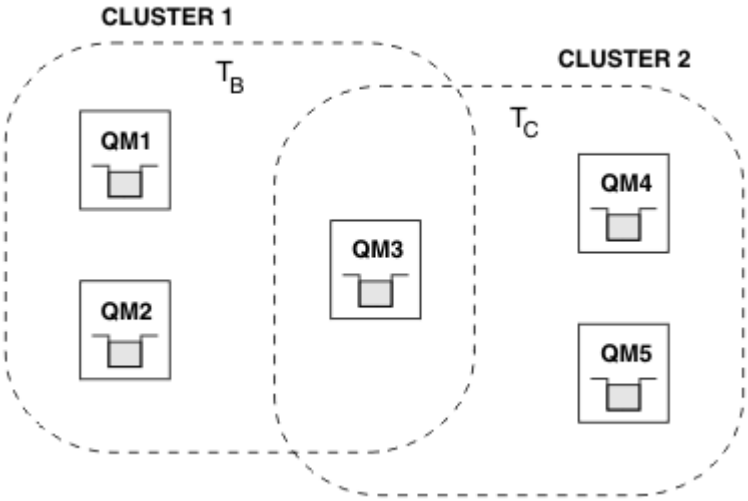


図 *33.* オーバーラップするクラスター*: 2* つのクラスターがそれぞれ別のトピックにサブスクライブする場合

オーバーラップに含まれないキュー・マネージャーのパブリッシャーとサブスクライバーは、それぞれのクラスター内のトピックに対してのみパブリッシュおよびサブスクライブすることができます。例えば、QM2 のトピック・ストリングのサブスクライバーは、トピックがクラスター化されているかどうかに関係なく、QM5 からパブリッシュされる同じトピック・ストリングに対してパブリッシュされるメッセージは受け取りません。 この配置を実現するには、パブリッシュ/サブスクライブ階層が必要です。

オーバーラップするクラスター、ワイルドカード・サブスクリプション

このトピックの前セクションで説明した理由により、複数のクラスターのメンバーであるキュー・マネージャーでワイルドカードを使用してトピックにサブスクライブする場合には注意する必要があります。

前述の例では、2 つのトピック・オブジェクトが以下のように構成されているものと想定されています。

- T_B: トピック名 'Football'、クラスター 'CLUSTER1'. トピック・ストリング '/Sport/Football'
- T_C: トピック名 'Tennis'、クラスター 'CLUSTER2'. トピック・ストリング '/Sport/Tennis'

このシナリオでは、2 つのクラスター・トピックが明確に分離されており、トピック名に関しても、トピック・ストリングに関しても、オーバーラップは一切ありません。

QM3 に接続されたアプリケーションは、 '/Sport/Football' へのサブスクリプションと '/Sport/Tennis'へのサブスクリプションを作成できます。 したがって、2 つのクラスターのどちらからもパブリケーションを受け取ります。 ただし、37 ページの『管理トピック・オブジェクト』で説明されているように、'/Sport/Football' と '/Sport/Tennis'の両方でパブリケーションを受信する目的で '/Sport/#'にサブスクライブする場合、このモデルはいずれのクラスターでもクラスター・トピックとして認識されないため、プロキシー・サブスクリプションは作成されません。 そのため、これらのクラスターの他のキュー・マネージャーからのパブリケーションが見逃されることになります。

既に説明したように、 CLUSTER 1 と CLUSTER 2 の両方で '/Sport/#' のクラスター・トピックを作成することは無効です。これらのクラスター・トピックが競合し、これを示す情報メッセージがエラー・ログに書き込まれるためです。 ただし、そのようなトピックを 1 つのクラスター (CLUSTER 1 など) にのみ作成することは「許可」されます。 現在は、 QM3 で '/Sport/#' にサブスクリプションを行うと、プロキシー・サブスクリプションが CLUSTER 1 のキュー・マネージャーにのみ送信されるため、 QM4 または QM5 から '/Sport/Tennis' へのパブリケーションの受信は引き続き失敗します。

このシナリオにおける唯一の解決策は、引き続き、2 つのサブスクリプションを個別に作成することです。

パブリッシュ/サブスクライブ・クラスターの *REFRESH CLUSTER* についての考慮事項
REFRESH CLUSTER コマンドを発行すると、キュー・マネージャーで、ローカルに保持されているクラスター情報 (クラスター・トピックおよびそれらが関連付けられているプロキシー・サブスクリプションを含む) が、当面の間、破棄されることになります。

REFRESH CLUSTER コマンドが発行されてから、クラスター・パブリッシュ/サブスクライブに関する必要な全情報をキュー・マネージャーが再び取得する時点までに要する時間は、クラスターの規模、可用性、および完全リポジトリー・キュー・マネージャーの応答性によって異なります。

リフレッシュ処理の間、パブリッシュ/サブスクライブ・クラスター内のパブリッシュ/サブスクライブ・トラフィックで障害が発生します。 大規模クラスターでは、**REFRESH CLUSTER** コマンドを使用すると、処理中のクラスターが中断される可能性があります。その後、クラスター・オブジェクトが 27 日間隔で対象のキュー・マネージャーすべてに状況の更新を自動的に送信する際にも同様のことが起こり得ます。 大規模クラスターでのリフレッシュはクラスターのパフォーマンスと可用性に影響を与える可能性があるを参照してください。 そのような理由で、**REFRESH CLUSTER** コマンドは、IBM サポート・センターで指示された場合にのみ、パブリッシュ/サブスクライブ・クラスターで使用してください。

クラスターへの悪影響は以下の症状として表面化することがあります。

- あるキュー・マネージャーのクラスター・トピックに対するサブスクリプションで、クラスター内の他のキュー・マネージャーに接続されているパブリッシャーからのパブリケーションを受け取っていません。
- あるキュー・マネージャーのクラスター・トピックにパブリッシュされたメッセージが、他のキュー・マネージャーのサブスクリプションに伝搬されません。
- この期間に作成されたキュー・マネージャーのクラスター・トピックに対するサブスクリプションで、クラスターの他のメンバーにプロキシー・サブスクリプションが一貫して送信されません。
- この期間に削除されたキュー・マネージャーのクラスター・トピックに対するサブスクリプションで、クラスターの他のメンバーからプロキシー・サブスクリプションが一貫して削除されません。
- メッセージ送信が 10 秒以上一時停止します。
- **MQPUT** の失敗。例えば、MQRC_PUBLICATION_FAILURE です。
- パブリケーションが MQRC_UNKNOWN_REMOTE_Q_MGR の理由で送達不能キューに置かれます。

このような理由から、**REFRESH CLUSTER** コマンドを発行する前にパブリッシュ/サブスクライブ・アプリケーションを静止させる必要があります。

REFRESH CLUSTER の使用上の注意 および クラスタリング: REFRESH CLUSTER の使用のベスト・プラクティスも参照してください。

パブリッシュ/サブスクライブ・クラスター内のキュー・マネージャーに対して **REFRESH CLUSTER** コマンドを発行した後、すべてのクラスター・キュー・マネージャーおよびクラスター・トピックが正常にリフレッシュされるまで待ってから、72 ページの『プロキシー・サブスクリプションの再同期』の説明に従ってプロキシー・サブスクリプションを再同期します。 この配置では、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・マネージャーに対してクラスター送信側チャネルを開始する必要があります。すべてのプロキシー・サブスクリプションが正しく再同期されてから、パブリッシュ/サブスクライブ・アプリケーションを再始動してください。

REFRESH CLUSTER コマンドが完了するのに長い時間がかかっている場合は、
SYSTEM.CLUSTER.COMMAND.QUEUE の CURDEPTH を調べてコマンドをモニターしてください。

関連概念
REFRESH CLUSTER の実行中に発生するアプリケーションの問題
クラスター化: REFRESH CLUSTER の使用に関するベスト・プラクティス
関連資料
MQSC コマンドのリファレンス: REFRESH CLUSTER

パブリッシュ/サブスクライブの階層
複数のキュー・マネージャーを 1 つの階層内にグループとしてまとめることができます。階層には、直接接続された 1 つ以上のキュー・マネージャーが含まれます。 キュー・マネージャーは、接続時の親子関係によって一緒に接続されます。 2 つのキュー・マネージャーが初めて接続されるときに、子キュー・マネージャーが親キュー・マネージャーに接続されます。

階層内で接続されている親キュー・マネージャーと子キュー・マネージャーの間には、階層から両者を切断しない限り、機能的な違いはありません。

注 : IBM WebSphere MQ の階層接続では、キュー・マネージャーの属性 PSMODE を ENABLED に設定する必要があります。

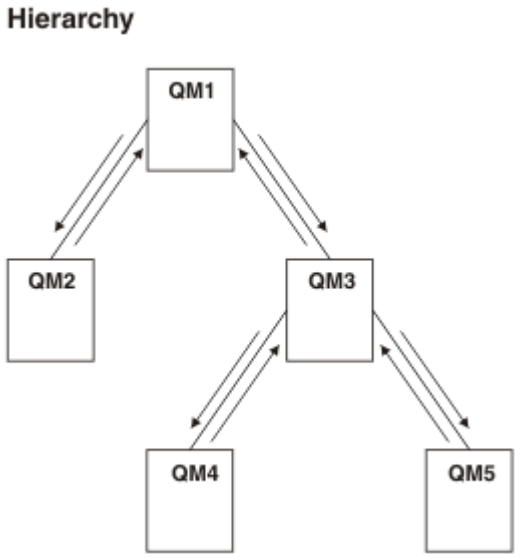


図 *34.* 単純なパブリッシュ/サブスクライブの階層

ブローカー階層へのキュー・マネージャーの接続
ローカル・キュー・マネージャーを親キュー・マネージャーに接続して、ブローカー階層を変更することができます。

始める前に

1. キュー型パブリッシュ/サブスクライブ・モードを有効にします。 キューに入れられたパブリッシュ/サブスクライブの開始を参照してください。
2. この変更は、 IBM WebSphere MQ 接続を使用して親キュー・マネージャーに伝搬されます。 この接続を確立する方法は 2 とおりです。
   - キュー・マネージャーを IBM WebSphere MQ クラスターに接続します。 クラスターへのキュー・マネージャーの追加 を参照してください。
   - 親キュー・マネージャーと同じ名前の伝送キュー、つまりキュー・マネージャー別名を使用して、Point-to-Point のチャネル接続を確立する。 Point-to-Point チャネル接続の確立方法について詳しくは、 WebSphere MQ 分散メッセージング技法を参照してください。

このタスクについて

`ALTER QMGR PARENT (PARENT_NAME)` runmqsc コマンドを使用して、子を親に接続します。

分散パブリッシュ/サブスクライブは、キュー・マネージャー・クラスターおよびクラスター化されたトピック定義を使用して実装されます。 IBM WebSphere MQ Version 6.0 および WebSphere Message Broker Version 6.1 および WebSphere Event Broker Version 6.1 以前とのインターオペラビリティーのために、キューに入れられたパブリッシュ/サブスクライブ・モードが有効になっている限り、 Version 7.1 以降のキュー・マネージャーをブローカー階層に接続することもできます。

手順

ALTER QMGR PARENT(PARENT)

例

最初の例は、QM2 を QM1 の子として接続し、QM2 にその接続について照会する方法を示しています。

```
C:>runmqsc QM2
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM2
alter qmgr parent(QM1)
     1 : alter qmgr parent(QM1)
AMQ8005: WebSphere MQ queue manager changed.
display pubsub all
     2 : display pubsub all
AMQ8723: Display pub/sub status details.
   QMNAME(QM2)                             TYPE(LOCAL)
   STATUS(ACTIVE)
AMQ8723: Display pub/sub status details.
   QMNAME(QM1)                             TYPE(PARENT)
   STATUS(ACTIVE)
```

次の例は、QM1 にその接続について照会した結果を示しています。

```
C:\Documents and Settings\Admin>runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.
display pubsub all
     2 : display pubsub all
AMQ8723: Display pub/sub status details.
   QMNAME(QM1)                             TYPE(LOCAL)
   STATUS(ACTIVE)
AMQ8723: Display pub/sub status details.
   QMNAME(QM2)                             TYPE(CHILD)
   STATUS(ACTIVE)
```

次のタスク

接続されたキュー・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が使用できる 1 つのブローカーまたはキュー・マネージャーに、トピックを定義することができます。 詳しくは、管理トピックの定義を参照してください。

関連概念

ストリームおよびトピック

WebSphere MQ パブリッシュ/サブスクライブ・メッセージングの概要

関連資料

DISPLAY PUBSUB

ブローカー階層からのキュー・マネージャーの切断

ブローカー階層内の親キュー・マネージャーから子キュー・マネージャーを切断します。

このタスクについて

ALTER QMGR コマンドを使用して、キュー・マネージャーをブローカー階層から切断します。 キュー・マネージャーは、いつでも任意の順序で切断することができます。

親の更新に対応する要求は、キュー・マネージャー間の接続が稼働中に送信されます。

手順

```
ALTER QMGR PARENT('')
```

例

```
C:\Documents and Settings\Admin>runmqsc QM2
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM2.
    1 : alter qmgr parent('')
AMQ8005: WebSphere MQ queue manager changed.
    2 : display pubsub type(child)
AMQ8147: WebSphere MQ object  not found.
display pubsub type(parent)
```

```
    3 : display pubsub type(parent)
AMQ8147: WebSphere MQ object  not found.
```

次のタスク

不要になったすべてのストリーム、キュー、および手動で定義したチャネルをすべて削除できます。

パブリッシュ/サブスクライブ階層の例*:* シナリオ *1*

キュー・マネージャー名の別名が指定されている Point-to-Point チャネルを使用して、パブリッシュ/サブスクライブ階層トポロジーをセットアップします。

このタスクについて

これらのシナリオでは、パブリッシュ/サブスクライブ階層を異なる方法でセットアップして、キュー・マネージャー間の接続を確立します。 これらすべてのシナリオで、1 つの親キュー・マネージャー QM1 および 2 つの子キュー・マネージャー QM2 と QM3 を使用します。

プロセスを簡単にたどれるように、シナリオ 1 は小さいセクションに分割されています。

シナリオ *1 (*パート *1):* キュー・マネージャーの作成

手順

1. 以下のコマンドを使用して、3 つのキュー・マネージャー QM1、QM2、および QM3 を作成して開始します。

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

2. 3 つのすべてのキュー・マネージャーで以下のコマンドを使用して、キュー・マネージャー・パブリッシュ/サブスクライブ・モードを有効にします。

```
ALTER QMGR PSMODE(ENABLED)
```

シナリオ *1 (*パート *2): Point-to-Point* チャネル接続

このタスクについて

親キュー・マネージャーと同じ名前のキュー・マネージャー別名を使用して、キュー・マネージャー間の Point-to-Point チャネル接続を確立します。

手順

1. QM2 で、QM1 に対する伝送キューおよびキュー・マネージャー別名を定義します。 QM1 に対する送信側チャネル、および QM1 で QM2 用に作成された送信側チャネルに対応する受信側チャネルを定義します。

```
DEFINE QLOCAL(QM1.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM1) RNAME('') RQMNAME(QM1) XMITQ(QM1.XMITQ)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(RCVR) TRPTYPE(TCP)
```

2. QM3 で、QM1 に対する伝送キューおよびキュー・マネージャー別名を定義します。 QM1 に対する送信側チャネル、および QM1 で QM3 用に作成された送信側チャネルに対応する受信側チャネルを定義します。

```
DEFINE QLOCAL(QM1.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM1) RNAME('') RQMNAME(QM1) XMITQ(QM1.XMITQ)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(RCVR) TRPTYPE(TCP)
```

3. QM1 で、QM2 および QM3 に対する伝送キューおよびキュー・マネージャー別名を定義します。 QM2 および QM3 に対する送信側チャネル、ならびに QM2 および QM3 で QM1 用に作成された送信側チャネルに対応する受信側チャネルを定義します。

```
DEFINE QLOCAL(QM2.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM2) RNAME('') RQMNAME(QM2) XMITQ(QM2.XMITQ)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(SDR) CONNAME('localhost(7777)') XMITQ(QM2.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)

DEFINE QLOCAL(QM3.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM3) RNAME('') RQMNAME(QM3) XMITQ(QM3.XMITQ)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(SDR) CONNAME('localhost(8888)') XMITQ(QM3.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)
```

4. キュー・マネージャーで該当のリスナーを開始します。

```
runmqlsr -m QM1 -t TCP -p 9999 &
runmqlsr -m QM2 -t TCP -p 7777 &
runmqlsr -m QM3 -t TCP -p 8888 &
```

5. 以下のチャネルを開始します。
   a. QM1 の場合:

   ```
   START CHANNEL('QM1.TO.QM2')

   START CHANNEL('QM1.TO.QM3')
   ```

   b. QM2 の場合:

   ```
   START CHANNEL('QM2.TO.QM1')
   ```

   c. QM3 の場合:

   ```
   START CHANNEL('QM3.TO.QM1')
   ```

6. すべてのチャネルが開始されたことを確認します。

```
DISPLAY CHSTATUS('QM1.TO.QM2')

DISPLAY CHSTATUS('QM1.TO.QM3')

DISPLAY CHSTATUS('QM2.TO.QM1')

DISPLAY CHSTATUS('QM3.TO.QM1')
```

シナリオ *1 (*パート *3):* キュー・マネージャーの接続およびトピックの定義

このタスクについて

子キュー・マネージャー QM2 と QM3 を親キュー・マネージャー QM1 に接続します。

手順

1. QM2 および QM3 で、親キュー・マネージャーを QM1 に設定します。

```
ALTER QMGR PARENT (QM1)
```

2. すべてのキュー・マネージャーで次のコマンドを実行して、子キュー・マネージャーが親キュー・マネージャーに接続していることを確認します。

```
DISPLAY PUBSUB TYPE(ALL)
```

3. トピック・オブジェクトを定義します。

```
define topic(FOOTBALL) TOPICSTR('Sport/Soccer')
```

シナリオ *1 (*パート *4):* トピックのパブリッシュとサブスクライブ

このタスクについて

アプリケーション amqspub.exe と amqssub.exe を使用して、トピックをパブリッシュおよびサブスクライブします。

手順

1. 1 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqspub Sport/Soccer QM2
```

2. 2 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqssub Sport/Soccer QM1
```

3. 3 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqssub Sport/Soccer QM3
```

タスクの結果

2 番目と 3 番目のコマンド・ウィンドウのアプリケーション amqssub.exe が、1 番目のコマンド・ウィンドウでパブリッシュされたメッセージを受け取ります。

関連タスク

90 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 2』
リモート・キュー・マネージャーと同じ伝送キュー名が指定されている Point-to-Point チャネルを使用して、パブリッシュ/サブスクライブ階層トポロジーをセットアップします。

93 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 3』
クラスター・チャネルを使用して、キュー・マネージャーを階層トポロジーに追加します。

パブリッシュ/サブスクライブ階層の例*:* シナリオ *2*
リモート・キュー・マネージャーと同じ伝送キュー名が指定されている Point-to-Point チャネルを使用して、パブリッシュ/サブスクライブ階層トポロジーをセットアップします。

このタスクについて

これらのシナリオでは、パブリッシュ/サブスクライブ階層を異なる方法でセットアップして、キュー・マネージャー間の接続を確立します。 これらすべてのシナリオで、1 つの親キュー・マネージャー QM1 および 2 つの子キュー・マネージャー QM2 と QM3 を使用します。

プロセスを簡単にたどれるように、シナリオ 2 は、小さいセクションに分割されています。 このシナリオでは、88 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 1』 のシナリオ 1 のパート 1、パート 3、およびパート 4 を再び使用します。

シナリオ *2 (*パート *1):* キュー・マネージャーの作成および *PSMODE* の設定

手順

1. 以下のコマンドを使用して、3 つのキュー・マネージャー QM1、QM2、および QM3 を作成して開始します。

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

2. 3 つのすべてのキュー・マネージャーで以下のコマンドを使用して、キュー・マネージャー・パブリッシュ/サブスクライブ・モードを有効にします。

```
ALTER QMGR PSMODE(ENABLED)
```

シナリオ *2 (*パート *2): Point-to-Point* チャネル接続

このタスクについて

親キュー・マネージャーと同じ名前の伝送キューを使用して、キュー・マネージャー間の Point-to-Point チャネル接続を確立します。

手順

1. QM2 で、QM1 に対する伝送キューを定義します。 QM1 に対する送信側チャネル、および QM1 で QM2 用に作成された送信側チャネルに対応する受信側チャネルを定義します。

```
DEFINE QLOCAL(QM1) USAGE(XMITQ)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1) TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(RCVR) TRPTYPE(TCP)
```

2. QM3 で、QM1 に対する伝送キューを定義します。 QM1 に対する送信側チャネル、および QM1 で QM3 用に作成された送信側チャネルに対応する受信側チャネルを定義します。

```
DEFINE QLOCAL(QM1) USAGE(XMITQ)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1) TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(RCVR) TRPTYPE(TCP)
```

3. QM1 で、QM2 および QM3 に対する伝送キューを定義します。 QM2 および QM3 に対する送信側チャネル、ならびに QM2 および QM3 で QM1 用に作成された送信側チャネルに対応する受信側チャネルを定義します。

```
DEFINE QLOCAL(QM2) USAGE(XMITQ)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(SDR) CONNAME('localhost(7777)') XMITQ(QM2) TRPTYPE(TCP)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)

DEFINE QLOCAL(QM3) USAGE(XMITQ)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(SDR) CONNAME('localhost(8888)') XMITQ(QM3) TRPTYPE(TCP)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)
```

4. キュー・マネージャーで該当のリスナーを開始します。

```
runmqlsr -m QM1 -t TCP -p 9999 &
runmqlsr -m QM2 -t TCP -p 7777 &
runmqlsr -m QM3 -t TCP -p 8888 &
```

5. 以下のチャネルを開始します。
   a. QM1 の場合:

```
START CHANNEL('QM1.TO.QM2')
START CHANNEL('QM1.TO.QM3')
```

   b. QM2 の場合:

```
START CHANNEL('QM2.TO.QM1')
```

   c. QM3 の場合:

```
START CHANNEL('QM3.TO.QM1')
```

6. すべてのチャネルが開始されたことを確認します。

```
DISPLAY CHSTATUS('QM1.TO.QM2')
DISPLAY CHSTATUS('QM1.TO.QM3')
DISPLAY CHSTATUS('QM2.TO.QM1')
DISPLAY CHSTATUS('QM3.TO.QM1')
```

シナリオ *2 (*パート *3):* キュー・マネージャーの接続およびトピックの定義

このタスクについて

子キュー・マネージャー QM2 と QM3 を親キュー・マネージャー QM1 に接続します。

手順

1. QM2 および QM3 で、親キュー・マネージャーを QM1 に設定します。

```
ALTER QMGR PARENT (QM1)
```

2. すべてのキュー・マネージャーで次のコマンドを実行して、子キュー・マネージャーが親キュー・マネージャーに接続していることを確認します。

```
DISPLAY PUBSUB TYPE(ALL)
```

3. トピック・オブジェクトを定義します。

```
define topic(FOOTBALL) TOPICSTR('Sport/Soccer')
```

シナリオ *2 (*パート *4):* トピックのパブリッシュとサブスクライブ

このタスクについて

アプリケーション amqspub.exe と amqssub.exe を使用して、トピックをパブリッシュおよびサブスクライブします。

手順

1. 1 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqspub Sport/Soccer QM2
```

2. 2 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqssub Sport/Soccer QM1
```

3. 3 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqssub Sport/Soccer QM3
```

タスクの結果

2 番目と 3 番目のコマンド・ウィンドウのアプリケーション amqssub.exe が、1 番目のコマンド・ウィンドウでパブリッシュされたメッセージを受け取ります。

関連タスク

88 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 1』
キュー・マネージャー名の別名が指定されている Point-to-Point チャネルを使用して、パブリッシュ/サブスクライブ階層トポロジーをセットアップします。

93 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 3』
クラスター・チャネルを使用して、キュー・マネージャーを階層トポロジーに追加します。

パブリッシュ/サブスクライブ階層の例*: シナリオ 3*
クラスター・チャネルを使用して、キュー・マネージャーを階層トポロジーに追加します。

このタスクについて

これらのシナリオでは、パブリッシュ/サブスクライブ階層を異なる方法でセットアップして、キュー・マネージャー間の接続を確立します。 これらすべてのシナリオで、1 つの親キュー・マネージャー QM1 および 2 つの子キュー・マネージャー QM2 と QM3 を使用します。

プロセスを簡単にたどれるように、シナリオ 3 は、小さいセクションに分割されています。 このシナリオでは、88 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 1』 のシナリオ 1 のパート 1、パート 3、およびパート 4 を再び使用します。

このシナリオでは、QM1 と QM2 を完全リポジトリーとし、QM3 を部分リポジトリーとするクラスター DEMO を作成します。 キュー・マネージャー QM1 はキュー・マネージャー QM2 と QM3 の親です。

シナリオ *2 (*パート *1):* キュー・マネージャーの作成および *PSMODE* の設定

手順

1. 以下のコマンドを使用して、3 つのキュー・マネージャー QM1、QM2、および QM3 を作成して開始します。

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

2. 3 つのすべてのキュー・マネージャーで以下のコマンドを使用して、キュー・マネージャー・パブリッシュ/サブスクライブ・モードを有効にします。

```
ALTER QMGR PSMODE(ENABLED)
```

シナリオ *2 (*パート *2): Point-to-Point* チャネル接続

このタスクについて

クラスター内のキュー・マネージャー間で Point-to-Point チャネル接続を確立します。

手順

1. QM1 および QM2 で、**REPOS** パラメーターをクラスター DEMO の名前に設定します。

```
ALTER QMGR REPOS(DEMO)
```

2. キュー・マネージャーで該当のリスナーを開始します。

```
runmqlsr -m QM1 -t TCP -p 9999 &
runmqlsr -m QM2 -t TCP -p 7777 &
runmqlsr -m QM3 -t TCP -p 8888 &
```

3. それぞれのキュー・マネージャーでクラスター受信側チャネルを定義します。

   a. QM1 の場合:

```
DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(9999)')
CLUSTER(DEMO)
```

   b. QM2 の場合:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(7777)')
CLUSTER(DEMO)
```

   c. QM3 の場合:

```
DEFINE CHANNEL(TO.QM3) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(8888)')
CLUSTER(DEMO)
```

4. クラスター内の各キュー・マネージャーで完全リポジトリーに対するクラスター送信側チャネルを定義します。

   a. QM1 の場合:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(7777)')
CLUSTER(DEMO)
```

   b. QM2 の場合:

```
DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(9999)')
CLUSTER(DEMO)
```

   c. QM3 は、QM1 と QM2 のいずれかの完全リポジトリーに対するクラスター送信側チャネルをもつことができます。 この例では、QM1 に対するチャネルを定義します。

```
DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(9999)')
CLUSTER(DEMO)
```

シナリオ *2 (*パート *3):* キュー・マネージャーの接続およびトピックの定義

このタスクについて

子キュー・マネージャー QM2 と QM3 を親キュー・マネージャー QM1 に接続します。

手順

1. QM2 および QM3 で、親キュー・マネージャーを QM1 に設定します。

```
ALTER QMGR PARENT (QM1)
```

2. すべてのキュー・マネージャーで次のコマンドを実行して、子キュー・マネージャーが親キュー・マネージャーに接続していることを確認します。

```
DISPLAY PUBSUB TYPE(ALL)
```

3. トピック・オブジェクトを定義します。

```
define topic(FOOTBALL) TOPICSTR('Sport/Soccer')
```

シナリオ *2 (*パート *4):* トピックのパブリッシュとサブスクライブ

このタスクについて

アプリケーション amqspub.exe と amqssub.exe を使用して、トピックをパブリッシュおよびサブスクライブします。

手順

1. 1 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqspub Sport/Soccer QM2
```

2. 2 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqssub Sport/Soccer QM1
```

3. 3 番目のコマンド・ウィンドウで次のコマンドを実行します。

```
amqssub Sport/Soccer QM3
```

タスクの結果

2 番目と 3 番目のコマンド・ウィンドウのアプリケーション amqssub.exe が、1 番目のコマンド・ウィンドウでパブリッシュされたメッセージを受け取ります。

関連タスク

88 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 1』
キュー・マネージャー名の別名が指定されている Point-to-Point チャネルを使用して、パブリッシュ/サブスクライブ階層トポロジーをセットアップします。

90 ページの『パブリッシュ/サブスクライブ階層の例: シナリオ 2』
リモート・キュー・マネージャーと同じ伝送キュー名が指定されている Point-to-Point チャネルを使用して、パブリッシュ/サブスクライブ階層トポロジーをセットアップします。

パブリケーションとサブスクリプションのフローの制御

分散パブリッシュ/サブスクライブ・トポロジーにまとめて接続されるキュー・マネージャーは、共通のフェデレーテッド・トピック・スペースを共有します。 それぞれのパブリケーションおよびサブスクリプションをローカルまたはグローバルとして選択することにより、トポロジー内のパブリケーションおよびサブスクリプションのフローを制御できます。

ローカル・パブリケーションおよびサブスクリプションは、パブリッシャーまたはサブスクライバーの接続先であるキュー・マネージャーを超えて伝搬されることはありません。

クラスターまたは階層の中で複数のキュー・マネージャーをまとめて接続することによって、作成されるトピック・スペースの範囲を制御できます。 パブリッシュ/サブスクライブ・クラスターでは、トピック・オブジェクトは「クラスター化」される必要があります。 そうしない場合、すべてのエレメントがローカルのままとなり、パブリケーションまたはサブスクリプションは無効になります。

1 つのサブスクリプションが複数の異なるパブリケーションのトピック・ストリングに一致する場合、そのサブスクリプションは複数の異なるトピック・オブジェクトに解決される可能性があります。 これらをオーバーラップ・トピックといいます。 一致する特定のパブリケーションに関連付けられたトピック・オブジェクトは、トピック属性を提供し、例えば、サブスクライバーがパブリケーションを受け取るかどうかなどを決定します。

パブリケーション有効範囲
パブリケーションの有効範囲は、キュー・マネージャーがパブリケーションをリモート・キュー・マネージャーに転送するかどうかを制御します。 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。

PUBSCOPE トピック属性は、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
パブリケーションは、ローカル・サブスクライバー、およびリモート・キュー・マネージャーに接続されたサブスクライバーに送信されます。 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
親の **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。

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。

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
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・パブリケーションとリモート・パブリケーションを受け取ります。

ASPARENT
親の **SUBSCOPE** 設定を使用します。

サブスクリプション作成時に MQSO_SCOPE_QMGR サブスクリプション・オプションを指定することにより、個々のサブスクライバーは **SUBSCOPE** 設定の ALL をオーバーライドできます。 サブスクリプションはトピックの **SUBSCOPE** 設定の ALL をオーバーライドできます。

注 : 個々のサブスクライバーが制限できるのは、トピック **SUBSCOPE** のみです。 個々のサブスクリプションの **SUBSCOPE** を ALL に設定した場合、一致するトピックの **SUBSCOPE** 設定がサブスクリプションに適用されます。

パブリケーションとサブスクリプションの有効範囲の結合

WebSphere MQ バージョン 7 以降では、パブリケーションおよびサブスクリプションの有効範囲は独立して機能し、キュー・マネージャー間のパブリケーションのフローを決定します。

パブリケーションは、パブリッシュ/サブスクライブ・トポロジー内で接続されているすべてのキュー・マネージャーに流れるようにすることも、ローカル・キュー・マネージャーのみに流れるようにすることも

できます。 プロキシー・サブスクリプションに関しても同様です。 どのパブリケーションがサブスクリプションに一致するかは、それら 2 つのフローの組み合わせによって制御されます。

パブリケーションとサブスクリプションは両方とも、有効範囲を QMGR または ALL に設定できます。 パブリッシャーとサブスクライバーが両方とも同じキュー・マネージャーに接続されている場合、有効範囲の設定が、サブスクライバーがパブリッシャーからどのパブリケーションを受け取るかに影響を及ぼすことはありません。

パブリッシャーとサブスクライバーが異なるキュー・マネージャーに接続されている場合、リモート・パブリケーションを受け取るには、両方の設定を ALL にする必要があります。

例えば、パブリッシャーが異なるキュー・マネージャーに接続されているとします。 サブスクライバーがあらゆるパブリッシャーからパブリケーションを受け取るようにするには、サブスクリプションの有効範囲を ALL に設定します。 その後、パブリッシャーごとに、そのパブリケーションの有効範囲をそのパブリッシャーにローカルのサブスクライバーに制限するかどうかを決定できます。

サブスクライバーが異なるキュー・マネージャーに接続されているとします。 あるパブリッシャーからのパブリケーションをすべてのサブスクライバーに送信するようにする場合、パブリケーション有効範囲を ALL に設定します。 サブスクライバーが同じキュー・マネージャーに接続されているパブリッシャーだけからパブリケーションを受け取るようにする場合、サブスクリプション有効範囲を QMGR に設定します。

バージョン 6 以前では、パブリケーションおよびサブスクリプション有効範囲は、どのパブリケーションが流れるかを制御するだけではありませんでした。 さらに、パブリケーションの有効範囲は、サブスクリプションの有効範囲と一致している必要がありました。

例: フットボールの試合結果サービス

フットボール・リーグのメンバーのチームであるとします。 各チームは、キュー・マネージャーをパブリッシュ/サブスクライブ・クラスター内の他のすべてのチームと接続させています。

チームは、トピック Football/result/*Home team name/Away team name* を使用して、ホーム・グラウンドで行われたすべてのゲーム結果を公開します。 イタリック体のストリングはトピック名の変数で、パブリケーションは試合の結果です。

また、各クラブはトピック・ストリング Football/myteam/*Home team name/Away team name* を使用して、そのクラブだけの結果をリパブリッシュします。

両方のトピックがクラスター全体にパブリッシュされます。

リーグは以下のサブスクリプションをセットアップして、どのチームのファンも 3 つの関心のある方法で試合結果をサブスクライブできるようにします。

SUBSCOPE(QMGR) を指定してクラスター・トピックをセットアップできることに注意してください。 トピック定義はクラスターの各メンバーに伝搬されますが、サブスクリプションの有効範囲はローカル・キュー・マネージャーのみです。 そのため、各キュー・マネージャーのサブスクライバーは、同じサブスクリプションからのさまざまなパブリケーションを受け取ります。

すべての試合結果を受け取る

```
DEFINE TOPIC(A) TOPICSTR('Football/result/') CLUSTER SUBSCOPE(ALL)
```

ホームの試合結果すべてを受け取る

```
DEFINE TOPIC(B) TOPICSTR('Football/result/') CLUSTER SUBSCOPE(QMGR)
```

サブスクリプションの有効範囲は QMGR なので、ホーム・グラウンドでパブリッシュされた試合結果のみが一致します。

自分のチームの試合結果すべてを受け取る

```
DEFINE TOPIC(C) TOPICSTR('Football/myteam/') CLUSTER SUBSCOPE(QMGR)
```

サブスクリプションの有効範囲は QMGR なので、ローカルでリパブリッシュされた、ローカルのチームの結果のみが一致します。

トピック・スペース
トピック・スペースは、サブスクライブ対象となる複数のトピックから成るセットです。 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・マネージャーに接続されたサブスクライバーごとにトピック・スペースが存在し、そこには、接続されているキュー・マネージャー上で定義されたトピックが含まれる可能性があります。

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際、トピックが最初に管理的に作成されます。または、アプリケーションがパブリケーションやサブスクリプションを動的に作成する際に、トピックが動的に作成されます。

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のメカニズムを使用すると、さまざまなキュー・マネージャーで定義されたトピックが共通のトピック・スペースにまとめて入れられます。

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。

さらに、クラスターのすべてのメンバーの間でクラスター・トピックに関するプロキシー・サブスクリプションが伝搬されます。 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。 プロキシー・サブスクリプションは、クラスター内のすべてのメンバーに転送されます。 複雑な状況について詳しくは、107 ページの『トピックのオーバーラップ』 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。

クラスター・トピック・オブジェクトを作成すると、二重の効果があります。 サブスクリプションがクラスター・トピックに解決される場合、トピックへのプロキシー・サブスクリプションがクラスターの他のメンバーに送られます。 さらに、トピック・オブジェクトのコピーがクラスターの他のメンバーに送られます。 クラスター・トピック・オブジェクトを転送すると、トピックの管理が単純化されるという効果があります。 通常、クラスター・トピック・オブジェクトはクラスター内の 1 つのキュー・マネージャー (これをクラスター・トピック・ホストという) に定義されます。

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。

- 99 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』.
- 100 ページの『ケース 2. バージョン 7 のパブリッシュ/サブスクライブ階層』.
- 100 ページの『ケース 3. バージョン 6 のパブリッシュ/サブスクライブ階層およびストリーム』.

各トピックで、以下のタスクはトピック・スペースを結合する方法を説明しています。

- 101 ページの『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』.
- 102 ページの『既存のバージョン 6 トピック・スペースへのバージョン 7 キュー・マネージャーの追加』.
- 103 ページの『複数のクラスターのトピック・スペースの結合』.

- 104 ページの『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』
- 106 ページの『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。

クラスター内の別の場所で定義されたクラスター・トピックの定義が競合する場合、最も新しい定義が優先的に選ばれ、競合が解決されます。 ある時点で 1 つのクラスター・トピックが重複定義されている場合、さまざまなキュー・マネージャーの間でクラスター・トピック定義が異なる可能性があります。

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・トピックのパブリケーションまたはサブスクリプションがクラスター内の別のキュー・マネージャーに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。

クラスター・トピック・ホストで、トピック・ストリング USA/Alabama を使用してクラスター・トピック・オブジェクト Alabama を定義するとします。 結果は次のとおりです。

1. クラスター・トピック・ホストのトピック・スペースには、クラスター・トピック・オブジェクト Alabama およびトピック USA/Alabama が含まれるようになります。
2. クラスター・トピック・オブジェクト Alabama はクラスター内のすべてのキュー・マネージャーに複製され、各キュー・マネージャーのトピック・スペースに結合されます。 クラスター内の各キュー・マネージャーの動作は、トピック・オブジェクト Alabama がキュー・マネージャーに存在するかどうかに応じて異なります。
   - Alabama が新しいトピック・オブジェクトである場合、キュー・マネージャーはクラスター・トピック・オブジェクト Alabama およびトピック USA/Alabama を自身のトピック・スペースに追加します。
   - Alabama がローカル定義である場合、クラスター・トピック・オブジェクト Alabama が追加されます。 ローカル定義が削除されない限り、リモートに定義されたクラスター・トピック・オブジェクトは無視されます。 キュー・マネージャーは両方の定義を保持します。
   - Alabama が別の場所で定義されたより古いクラスター・トピック・オブジェクトである場合は、より新しいクラスター・トピック・オブジェクトで置換されます。
3. アプリケーションまたは管理者は、Alabama トピック・オブジェクトを参照することによって USA/Alabama へのサブスクリプションを作成できます。
4. アプリケーションは、クラスター内の任意の場所で、トピック・ストリング USA/Alabama を直接使用して、トピック・オブジェクト Alabama の属性を継承するサブスクリプションを作成できます。 トピック・オブジェクト Alabama は、USA/Alabama で始まる任意のトピック・ストリングから形成されるサブスクリプションによって継承されます。

   別のいずれかのキュー・マネージャー上にトピック・オブジェクト Alabama の定義がもう 1 つ存在する場合、それはクラスター・トピック・ホスト上の定義よりも優先されます。 ローカル・オブジェクトにはクラスター属性がある場合とない場合とがあります。 クラスター属性は同じクラスターを参照す

ることも、別のクラスターを参照することもあります。 このような多重定義が起こらないようにしてください。 多重定義により、動作のばらつきが生じます。

5. トピック・オブジェクト Alabama が **PUBSCOPE** 属性 ALL を持っている場合、Alabama に解決されるサブスクリプションはクラスター内の他のすべてのキュー・マネージャーに送られます。

   クラスター内の別のキュー・マネージャーに接続されたサブスクライバーにパブリッシャーからパブリケーションが送られることを防ぐには、Alabama の **PUBSCOPE** 属性を QMGR に設定してください。

   Alabama トピック・オブジェクトはクラスター内の各キュー・マネージャーに複製されるので、**PUBSCOPE** 属性および **PUBSCOPE** 属性はクラスター内のすべてのキュー・マネージャーに適用されます。

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。

ただし、別のトピック・ストリングを使用して、1 つのトピック・オブジェクトの重複定義をクラスター内の別のキュー・マネージャー上に作成することが可能です。 1 つのキュー・マネージャー上でクラスター・トピック・ホストを管理することにより、常に重複を避けるようにしてください。 この重要点について詳しくは、74 ページの『複数のクラスター・トピック定義』を参照してください。 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ります。

ケース 2. バージョン 7 のパブリッシュ/サブスクライブ階層

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。

バージョン 7 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、および接続されているキュー・マネージャー上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・トピックとクラスター・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 およびトピック・ストリング USA/Alabama が含まれるようになります。
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・マネージャーで、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。

ケース 3. バージョン 6 のパブリッシュ/サブスクライブ階層およびストリーム

バージョン 7 より前では、トピック・スペースが複数の個別のストリームに分割され、各ストリームにすべてのキュー・マネージャーに存在するデフォルト・ストリームが含まれていました。 パブリケーションは、異なるストリーム間で送信できません。 名前付きストリームが使用される場合、異なるキュー・マネージャー上のトピック・スペースが異なる可能性があります。 トピックは、デフォルト・ストリーム内のトピック、およびさまざまな名前付きストリーム内のトピックに分割されます。

注 : 各名前付きストリームが、個別のトピック・スペースを形成します。 接続されたトポロジーを形成するには、接続されるキュー・マネージャー上にそれぞれの名前付きストリームが存在する必要があります。 ストリーム X が QMA および QMC に定義され、QMB には定義されていないとします。 QMA が QMB の親であり、QMB が QMC の親である場合には、QMA と QMC の間ではストリーム X のトピックが送信されません。

オプション **PUBSCOPE** および **SUBSCOPE** の両方を QMGR または ALL に設定した場合、トピックのパブリッシャーとサブスクライバーは、ローカルにコンシュームされるパブリケーションのみを交換するか、グローバルにコンシュームされるパブリケーションのみを交換する必要があります。

バージョン 7 以降、パブリッシュ/サブスクライブ API を使ってストリームを使用することはできません。 キュー型パブリッシュ/サブスクライブをバージョン 7 キュー・マネージャーで使用すると、ストリームがさまざまなトピック・オブジェクトにマップされて、ストリームの機能をシミュレートできます。 ストリーム内のすべてのトピックのルート・トピックであるトピック・オブジェクトを作成することにより、ストリームがシミュレートされます。 キュー・マネージャーは、各ツリーの対応するルート・トピックとストリームとの間でパブリケーションおよびサブスクリプションをマップします。

トピック・スペースの結合
キュー・マネージャーのトピック・スペースを、パブリッシュ/サブスクライブ・クラスターまたは階層内の他のキュー・マネージャーと結合します。 パブリッシュ/サブスクライブ・クラスターを結合し、パブリッシュ/サブスクライブ・クラスターと階層を結合します。

CLUSTER、**PUBSCOPE**、および **SUBSCOPE** 属性、パブリッシュ/サブスクライブ・クラスター、およびパブリッシュ/サブスクライブ階層といったビルディング・ブロックを使用することにより、さまざまなパブリッシュ/サブスクライブ・トピック・スペースを作成できます。

以下のシナリオでは、1 つのキュー・マネージャーからパブリッシュ/ サブスクライブ・クラスターに拡大する例に始まり、さまざまなパブリッシュ/サブスクライブ・トポロジーについて説明します。

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
複数のキュー・マネージャー上で実行されるように、パブリッシュ/サブスクライブ・システムを拡大します。 各パブリッシャーおよびサブスクライバーに単一で同一のトピック・スペースを提供するために、パブリッシュ/サブスクライブ・クラスターを使用します。

始める前に
単一のバージョン 7 キュー・マネージャーにパブリッシュ/サブスクライブ・システムを実装しています。

SYSTEM.BASE.TOPIC の属性の継承に依存するのではなく、常に、独自のルート・トピックを含むトピック・スペースを作成してください。 パブリッシュ/サブスクライブ・システムをクラスターに拡大する場合、クラスター・トピック・ホストで、ルート・トピックをクラスター・トピックとして定義でき、そうするとすべてのトピックがクラスター全体で共有されます。

このタスクについて
より多くのパブリッシャーおよびサブスクライバーをサポートするためにシステムを拡大し、すべてのトピックをクラスター全体で可視にしようとしています。

手順
1. パブリッシュ/サブスクライブ・システムで使用するクラスターを作成します。
   従来のクラスターが既に存在する場合、パフォーマンス上の理由で、新しいパブリッシュ/サブスクライブ・システム用に新しいクラスターをセットアップする方が良いでしょう。 両方のクラスターのクラスター・リポジトリーに、同じサーバーを使用できます。
2. クラスター・トピック・ホストにするキュー・マネージャーを 1 つ (おそらく、リポジトリーのいずれか) 選択します。
3. パブリッシュ/サブスクライブ・クラスター全体で可視にするすべてのトピックが、管理トピック・オブジェクトに解決されるようにします。
   CLUSTER 属性を設定して、パブリッシュ/サブスクライブ・クラスターの名前を指定します。

次のタスク

パブリッシャーおよびサブスクライバー・アプリケーションをクラスター内の任意のキュー・マネージャーに接続します。

CLUSTER 属性を持つ管理トピック・オブジェクトを作成します。 これらのトピックもクラスター全体に伝搬されます。 パブリッシャーおよびサブスクライバー・プログラムは、クラスター内の異なるキュー・マネージャーに接続しても動作が変わらないように、管理トピックを使用します。

SYSTEM.BASE.TOPIC がすべてのキュー・マネージャー上でクラスター・トピックのように機能する必要がある場合、それをすべてのキュー・マネージャー上で変更する必要があります。

既存のバージョン 6 トピック・スペースへのバージョン 7 キュー・マネージャーの追加

同じトピック・スペースを共有して、バージョン 7 キュー・マネージャーと相互運用するために、既存のバージョン 6 パブリッシュ/サブスクライブ・システムを拡張します。

始める前に

バージョン 6 パブリッシュ/サブスクライブ・システムが既に存在しています。

新しいサーバーに WebSphere MQ バージョン 7 をインストールしており、キュー・マネージャーを構成しました。

このタスクについて

バージョン 7 キュー・マネージャーで作業するために、既存のバージョン 6 パブリッシュ/サブスクライブ・システムを拡張しようとしています。

キュー・パブリッシュ/サブスクライブ・インターフェースを使用する、バージョン 6 パブリッシュ/サブスクライブ・システムの開発を固定することに決めました。 バージョン 7 MQI を使用して、システムに拡張を追加しようとしています。 現時点では、キュー・パブリッシュ/サブスクライブ・アプリケーションを作成し直すという計画はありません。

将来的には、バージョン 6 キュー・マネージャーをバージョン 7 にアップグレードすることを考えています。 現時点では、バージョン 7 キュー・マネージャー上で既存のキュー・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実行するつもりです。

手順

1. 送信側と受信側のチャネルのセットを 1 つ作成し、両方向でバージョン 7 キュー・マネージャーをバージョン 6 キュー・マネージャーの 1 つと接続します。
2. ターゲット・キュー・マネージャーの名前を指定して、2 つの伝送キューを作成します。 何らかの理由で、伝送キュー名としてターゲット・キュー・マネージャーの名前を使用できない場合は、キュー・マネージャーの別名を使用します。
3. 送信側チャネルをトリガーするように、伝送キューを構成します。
4. バージョン 6 パブリッシュ/サブスクライブ・システムがストリームを使用する場合は、 ストリームの追加 の説明に従って、ストリームをバージョン 7 キュー・マネージャーに追加します。
5. バージョン 7 キュー・マネージャーの **PSMODE** が ENABLE に設定されていることを検査します。
6. その **PARENT** 属性を変更して、バージョン 6 キュー・マネージャーのいずれかを参照するようにします。
7. キュー・マネージャー間の親子関係の状況が両方向でアクティブであることを検査します。

次のタスク

この作業を完了すると、バージョン 6 とバージョン 7 の両方のキュー・マネージャーが同じトピック・スペースを共有します。 例えば、以下の作業をすべて行えます。

- バージョン 6 キュー・マネージャーとバージョン 7 キュー・マネージャーの間でパブリケーションおよびサブスクリプションを交換する。

- 既存のバージョン 6 パブリッシュ/サブスクライブ・プログラムをバージョン 7 キュー・マネージャーで実行する。
- バージョン 6 キュー・マネージャーとバージョン 7 キュー・マネージャーのどちらでも、トピック・スペースを表示および変更する。
- バージョン 7 パブリッシュ/サブスクライブ・アプリケーションを作成し、バージョン 7 キュー・マネージャーで実行する。
- バージョン 7 アプリケーションで新しいパブリケーションおよびサブスクリプションを作成し、それらをバージョン 6 アプリケーションと交換する。

複数のクラスターのトピック・スペースの結合
複数のクラスターにまたがるトピック・スペースを作成します。 1 つのクラスターでトピックにパブリッシュし、別のクラスターでそれに対してサブスクライブします。

始める前に

既存のパブリッシュ/サブスクライブ・クラスターがあり、いくつかのクラスター・トピックをすべてのクラスターに伝搬させようとしています。

このタスクについて

あるクラスターから別のクラスターへパブリケーションを伝搬するには、それらのクラスターをまとめて 1 つの階層に結合する必要があります。103 ページの図 35 を参照してください。 階層型の接続は、接続されているキュー・マネージャー間でサブスクリプションおよびパブリケーションを伝搬します。クラスターは各クラスター内でクラスター・トピックを伝搬しますが、クラスター間では伝搬しません。

これらの 2 つのメカニズムの組み合わせにより、すべてのクラスター間でクラスター・トピックが伝搬されます。 クラスターごとに、クラスター・トピック定義を繰り返す必要があります。

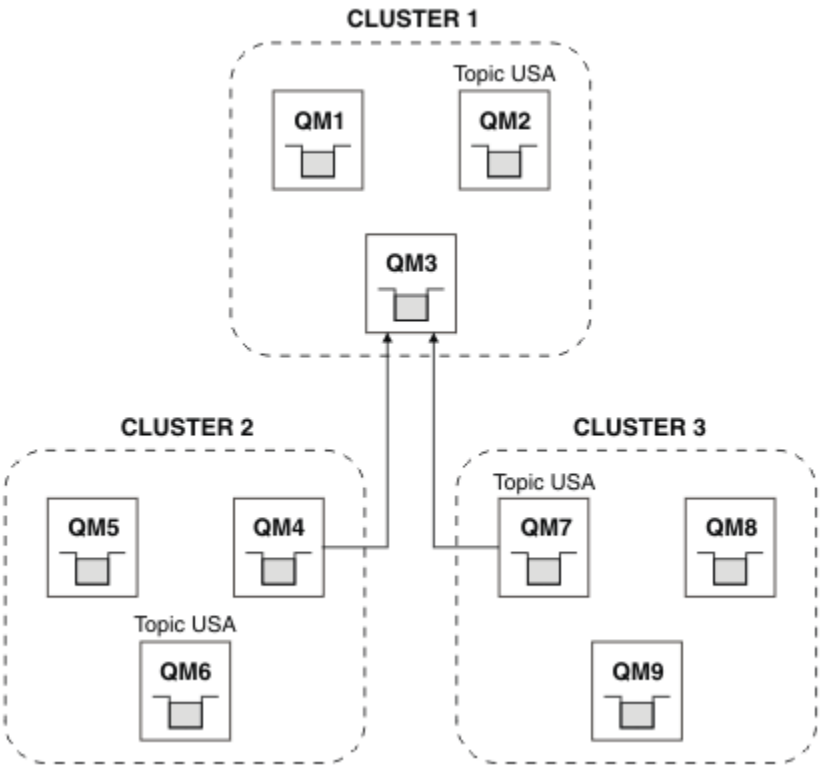


図 *35.* 階層を使用したクラスターの接続

以下のステップにより、クラスターが階層に接続されます。

手順

1. 送信側と受信側のチャネルのセットを 2 つ作成し、QM3 と QM4、QM3 と QM7 を両方向で接続します。 従来の送信側と受信側のチャネルおよび伝送キュー (クラスターではなく) を使用して、階層に接続しなければなりません。
2. ターゲット・キュー・マネージャーの名前を指定して、3 つの伝送キューを作成します。 何らかの理由で、伝送キュー名としてターゲット・キュー・マネージャーの名前を使用できない場合は、キュー・マネージャーの別名を使用します。
3. 送信側チャネルをトリガーするように、伝送キューを構成します。
4. QM3、QM4、および QM7 の **PSMODE** が ENABLE に設定されていることを検査します。
5. QM4 および QM7 の **PARENT** 属性を QM3 に変更します。
6. キュー・マネージャー間の親子関係の状況が両方向でアクティブであることを検査します。
7. クラスター 1、2、および 3 の 3 つのクラスター・トピック・ホストそれぞれに、属性 **CLUSTER**('CLUSTER 1')、**CLUSTER**('CLUSTER 2')、および **CLUSTER**('CLUSTER 3') を指定して管理トピック USA を作成します。 クラスター・トピック・ホストは、階層的に接続されたキュー・マネージャーである必要はありません。

次のタスク

これで、103 ページの図 35 のクラスター・トピック USA に対してパブリッシュまたはサブスクライブを行えるようになりました。 パブリケーション/サブスクリプションは、3 つのすべてのクラスター内のパブリッシャーおよびサブスクライバーに流れます。

例えば、他のクラスター内では USA をクラスター・トピックとして作成しなかったとします。 USA が QM7 だけで定義されている場合、USA に対するパブリケーションおよびサブスクリプションは、QM7、QM8、QM9、および QM3 の間で交換されます。 QM7、QM8、QM9 で実行されているパブリッシャーおよびサブスクライバーは、管理トピック USA の属性を継承します。 QM3 でのパブリッシャーおよびサブスクライバーは、QM3 上の SYSTEM.BASE.TOPIC の属性を継承します。

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

いくつかのトピック・スペースを特定のクラスターに分離して、他のトピック・スペースを結合し、接続されているすべてのクラスター内でそれらにアクセスできるようにします。

始める前に

103 ページの『複数のクラスターのトピック・スペースの結合』のトピックを調べてください。 ブリッジとしてさらにキュー・マネージャーを追加しなくても、このトピックの手順で十分に必要を満たせる場合があります。

このタスクについて

すべてのクラスター間で共有されないクラスター・トピックを分離させることにより、103 ページの『複数のクラスターのトピック・スペースの結合』 の 103 ページの図 35 で示されているトポロジーを向上させることができる可能性があります。 どのクラスターにも含まれていないブリッジング・キュー・マネージャーを作成することにより、クラスターを分離します (105 ページの図 36 を参照)。 ブリッジング・キュー・マネージャーを使用して、どのパブリケーションおよびサブスクリプションが、あるクラスターから別のクラスターへ流れるようにするかをフィルター処理します。

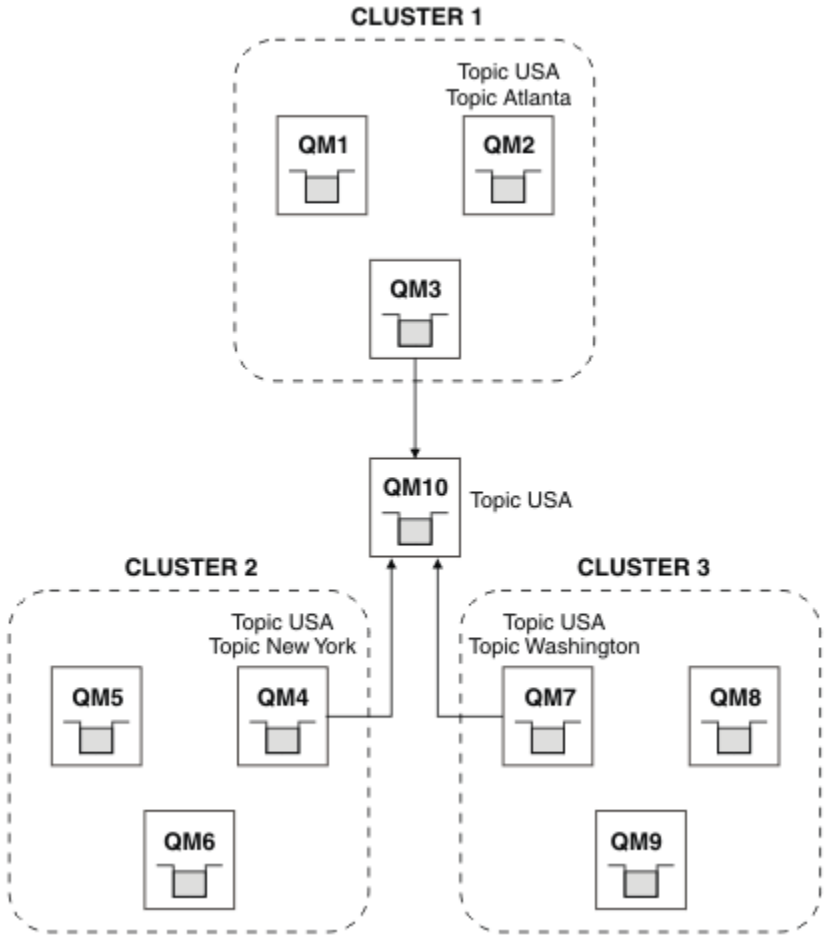


図 *36.* ブリッジされるクラスター

ブリッジを使用すると、ブリッジを超えて他のクラスターで公開しないクラスター・トピックを分離できます。 105 ページの図 36 では、 USA はすべてのクラスターで共有されるクラスター・トピックであり、 Atlanta、 New York 、および Washington は、それぞれ 1 つのクラスターでのみ共有されるクラスター・トピックです。

以下の手順を使用して、構成をモデル化しまてください。

手順

1. すべてのキュー・マネージャーで **SUBSCOPE**(QMGR) および **PUBSCOPE**(QMGR) が設定されるように、すべての SYSTEM.BASE.TOPIC トピック・オブジェクトを変更します。

   クラスター・トピックのルート・トピックで明示的に **SUBSCOPE**(ALL) および **PUBSCOPE**(ALL) を設定しない限り、他のキュー・マネージャーにトピック (クラスター・トピックも) は伝搬されません。
2. 属性 **CLUSTER**(*clustername*)、**SUBSCOPE**(ALL)、および **PUBSCOPE**(ALL) を使用して、各クラスターで共有するトピックを 3 つのクラスター・トピック・ホスト上に定義します。

   いくつかのクラスター・トピックをすべてのクラスター間で共有するには、それぞれのクラスターで同じトピックを定義します。 クラスター属性として、各クラスターのクラスター名を使用します。
3. すべてのクラスター間で共有するクラスター・トピックの場合、属性 **SUBSCOPE**(ALL) および **PUBSCOPE**(ALL) を指定して、ブリッジ・キュー・マネージャー (QM10) で再びトピックを定義します。

例

105 ページの図 36 の例では、USA から継承するトピックのみが 3 つのクラスターすべての間で伝搬されます。

次のタスク

SUBSCOPE(ALL) および **PUBSCOPE**(ALL) を指定してブリッジ・キュー・マネージャーで定義されたトピック用のサブスクリプションは、クラスター間で伝搬されます。

属性 **CLUSTER**(*clustername*)、**SUBSCOPE**(ALL)、および **PUBSCOPE**(ALL) を指定して各クラスター内で定義されているトピックのサブスクリプションが、各クラスター内で伝搬されます。

他のサブスクリプションはすべて、キュー・マネージャーにとってローカルです。

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

オーバーラップされたクラスターを使用して、複数のクラスター内のトピックに対するパブリッシュおよびサブスクライブを行います。 クラスター内でトピック・スペースがオーバーラップしない限り、この手法を使用できます。

始める前に

従来のクラスターを複数作成して、クラスター間の交差部分にいくつかのキュー・マネージャーを配置します。

このタスクについて

さまざまな理由で、クラスターをオーバーラップさせることを選択する場合があります。

1. 高可用性サーバーまたはキュー・マネージャーの数が限られています。 クラスター・リポジトリーおよびクラスター・トピック・ホストはすべてそれらにデプロイすることにしています。
2. ゲートウェイ・キュー・マネージャーを使用して接続される従来のキュー・マネージャー・クラスターが既に存在します。 パブリッシュ/サブスクライブ・アプリケーションを同じクラスター・トポロジーにデプロイする予定です。
3. 自己完結型のパブリッシュ/サブスクライブ・アプリケーションがいくつか存在します。 パフォーマンス上の理由で、パブリッシュ/サブスクライブ・クラスターを小さいままにし、従来のクラスターから分離させる方が適切です。 それらのアプリケーションを別のクラスターにデプロイすることにしています。 ただし、モニタリング・アプリケーションのライセンス交付を受けたコピーが 1 つだけであるため、1 つのキュー・マネージャー上ですべてのパブリッシュ/サブスクライブ・アプリケーションをモニターする必要があります。 このキュー・マネージャーは、すべてのクラスター内のクラスター・トピックに対するパブリケーションにアクセスできなければなりません。

オーバーラップしないトピック・スペースでトピックが定義されるようにすることにより、オーバーラップしているパブリッシュ/サブスクライブ・クラスターにそのトピックをデプロイすることができます (107 ページの図 37 を参照)。 トピック・スペースがオーバーラップしている場合、オーバーラップするクラスターにデプロイすると問題が発生します。

パブリッシュ/サブスクライブ・クラスターがオーバーラップするため、オーバーラップ内のキュー・マネージャーを使用して、任意のトピック・スペースにパブリッシュおよびサブスクライブすることができます。

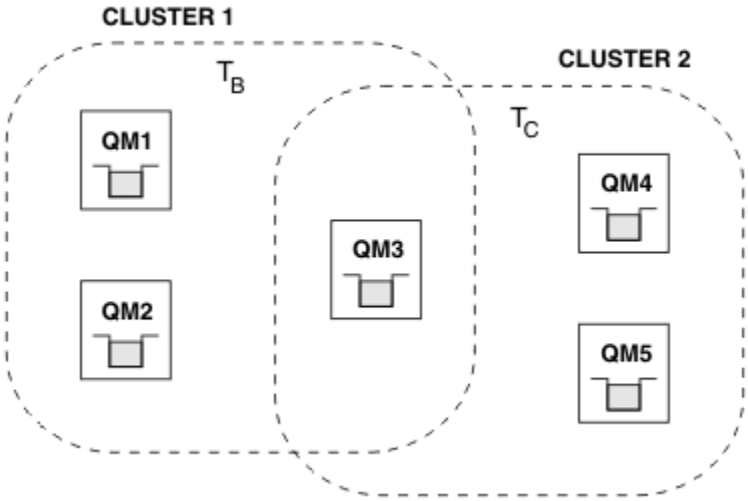


図 37. オーバーラップするクラスター、オーバーラップしないトピック・スペース

手順

トピック・スペースがオーバーラップしないようにする方法を作成します。

例えば、トピック・スペースごとに固有のルート・トピックを定義します。 そのルート・トピックをクラスター・トピックにします。

a) `DEFINE TOPIC(B) TOPICSTR('B') CLUSTER('CLUSTER 1') ...`

b) `DEFINE TOPIC(C) TOPICSTR('C') CLUSTER('CLUSTER 2') ...`

例

107 ページの図 37 で、QM3 に接続されているパブリッシャーおよびサブスクライバーは、T_B または T_C に対してパブリッシュまたはサブスクライブできます。

次のタスク

両方のクラスター内のトピックを使用するパブリッシャーおよびサブスクライバーを、オーバーラップ内のキュー・マネージャーに接続します。

特定のクラスター内のトピックのみを使用する必要があるパブリッシャーおよびサブスクライバーを、オーバーラップ内にないキュー・マネージャーに接続します。

トピックのオーバーラップ
分散パブリッシュ/サブスクライブ・トポロジー、パブリケーション、およびサブスクリプション・トピック・ストリングによっては、パブリケーションを異なるトピック・オブジェクトと関連付けられる場合に、トピックのオーバーラップが生じます。

トピックを複数のトピック・オブジェクトに解決できる場合、トピック間のオーバーラップを考慮する必要があります。

クラスター内のローカル・トピック
トピックは、クラスター内の任意のキュー・マネージャーで定義できます。 トピックがローカルで定義されている場合、他の場所で定義されている、同じトピック・ストリングに解決されるクラスター・トピックよりも優先されます。

クラスター内のクラスター・トピック
トピックは、クラスター内の任意のキュー・マネージャーで定義できます。 トピックがクラスター化されている場合は、クラスターの他のメンバーに複製されます。 トピックがクラスター内の別のキュー・マネージャー上のクラスター・トピックとして定義されている場合は、エラーになります。 既存

のクラスター定義があるキュー・マネージャーのエラー・ログに、エラー・メッセージが書き込まれます。

原則として、クラスター・トピックの定義が 1 つだけ存在するように、クラスター・トピックはクラスター内の 1 つのキュー・マネージャー (クラスター・トピック・ホスト) のみで定義してください。

クラスター・トピックを再定義すると、変更が各キュー・マネージャーに達するまで時間がかかります。 最終的に、最新の定義が、非クラスター・トピック・ホストへ複製された以前のクラスター・トピック定義をオーバーライドします。

異なる属性を指定して、クラスター内の複数のキュー・マネージャー上でクラスター・トピックを定義しても、最新の定義が以前のローカル定義をオーバーライドすることはありません。

複数のトピック・ストリングへ解決されるワイルドカード・サブスクリプション

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、トピック・スペース内のさまざまなトピックがサブスクリプションと一致する可能性があり、サブスクリプションはさまざまなトピック・オブジェクトに解決される場合があります。

例えば、クラスター SPORTS 内の以下のトピック定義について考慮してください。

```
DEFINE TOPIC(A) TOPICSTR('Football/result/#') SUBSCOPE(QMGR) CLUSTER(SPORTS)
DEFINE TOPIC(B) TOPICSTR('Football/#') SUBSCOPE(ALL) CLUSTER(SPORTS)
DEFINE TOPIC(C) TOPICSTR('Football/result/Newport/Cardiff') PUBSCOPE(ALL) SUBSCOPE(ALL)
CLUSTER(SPORTS)
DEFINE TOPIC(D) TOPICSTR('Football/matches/Newport/Cardiff') PUBSCOPE(ALL) SUBSCOPE(QMGR)
CLUSTER(SPORTS)
```

そのクラスター内に QM1 と QM2 という 2 つのキュー・マネージャーが存在するとします。 トピック C および D が QM1 でパブリッシュされます。

QM2 上のサブスクライバーが何を受け取るか考えてみてください (それらのサブスクリプションがグループ化されていない場合)。

- トピック A へのサブスクリプションでは、何も受け取りません。
  - SUBSCOPE(QMGR) であり、パブリケーションは他のキュー・マネージャー上です。
- トピック B へのサブスクリプションでは、両方のパブリケーションを受け取ります。
  - 両方のケースで SUBSCOPE(ALL) および PUBSCOPE(ALL) です。
- トピック C へのサブスクリプションでは、1 つのパブリケーションを受け取ります。
  - SUBSCOPE(ALL) および PUBSCOPE(ALL) であり、トピック C のパブリケーションに一致します。
- トピック D へのサブスクリプションでは、何も受け取りません。
  - SUBSCOPE(QMGR) であり、パブリケーションは他のキュー・マネージャー上です。

QM2 上のサブスクライバーが何を受け取るか考えてみてください (それらのサブスクリプションがグループ化されている場合)。

- サブスクライバーはトピック C での 1 つのパブリケーションを受け取ります。
  - SUBSCOPE(QMGR) が指定されているトピック A での一致するサブスクリプションは、SUBSCOPE(ALL) が指定されているトピック C での一致するサブスクリプションによってオーバーライドされます。 より具体的なサブスクリプションが優先され、パブリケーションが受け取られます。
  - サブスクリプションがグループ化されており、C の方がより具体的であるため、トピック C での一致するサブスクリプションが優先され、トピック B での一致するサブスクリプションは拒否されます。 重複するパブリケーションは廃棄されます。
- サブスクライバーはトピック D でパブリケーションを受け取りません。
  - SUBSCOPE(ALL) が指定されているトピック B での一致するサブスクリプションは、SUBSCOPE(QMGR) が指定されているトピック D での一致するサブスクリプションによってオーバ

ーライドされます。 より具体的なサブスクリプションが優先され、パブリケーションが廃棄されます。

ループ検出の仕組み

分散パブリッシュ/サブスクライブ・ネットワークでは、パブリケーションやプロキシー・サブスクリプションのループを作らないことが重要になります。ループができると、接続しているサブスクライバーが同じオリジナル・パブリケーションのコピーを複数受け取ることになり、ネットワークがあふれてしまいます。

プロキシー・サブスクリプションの集約システム (54 ページの『プロキシー・サブスクリプションの集約およびパブリケーションの集約』を参照) を使用しても、ループをすべて回避できるわけではありません。ただし、プロキシー・サブスクリプションの永久ループは回避できます。 パブリケーションが伝搬するかどうかは、プロキシー・サブスクリプションが存在するかどうかによって決まるので、パブリケーションについては永久ループが生じる可能性があります。 Websphere MQ V7.0 では、パブリケーションの永久ループを回避するために、以下の技法を使用しています。

パブリケーションがパブリッシュ/サブスクライブ・トポロジーの中を移動すると、それぞれのキュー・マネージャーがメッセージ・ヘッダーに固有の指紋を追加します。 パブリッシュ/サブスクライブ・キュー・マネージャーは、別のパブリッシュ/サブスクライブ・キュー・マネージャーからパブリケーションを受け取るたびに、メッセージ・ヘッダーで保持されているそれらの指紋をチェックします。 自分の指紋が既に存在した場合、キュー・マネージャーはパブリケーションがループ内を完全に一周してきたと見なし、そのメッセージを廃棄して、エラー・ログに項目を追加します。

注 : ループの中では、パブリケーションが両方向に伝搬するので、ループ内の各キュー・マネージャーは、発信側のキュー・マネージャーがループのパブリケーションを廃棄するまで、両方のパブリケーションを受け取ることになります。 その結果、サブスクライブ側のアプリケーションは、ループが破棄されるまで、パブリケーションの重複コピーを受け取ります。

ループ検出の指紋の形式

ループ検出の指紋は、V7.0 プロトコルの一部として RFH2 ヘッダーまたはフローに挿入されます。 RFH2 プログラマーは、ヘッダーを理解し、指紋情報をそのまま渡す必要があります。 WebSphere MessageBroker は、指紋情報を含まない RFH1 ヘッダーを使用します。

```
<ibm>
  <Rfp>uuid1</Rfp>
  <Rfp>uuid2</Rfp>
  <Rfp>uuid3</Rfp>
  . . .
</ibm>
```

<ibm> は、アクセスされた各キュー・マネージャーの固有のユーザー ID (UUID) を入れる、ルーティング用の指紋のリストを格納するためのフォルダーの名前です。

キュー・マネージャーは、メッセージをパブリッシュするたびに、<Rfp> (Routing fingerprint: ルーティング用の指紋) タグを使用して、自分の UUID を <ibm> フォルダーに追加します。 パブリケーションを受け取るたびに、 WebSphere MQ はメッセージ・プロパティー API を使用して < Rfp> タグを反復処理し、その特定の UUID 値が存在するかどうかを確認します。 キュー・パブリッシュ/サブスクライブ・インターフェースを使用する際に、WebSphere MQ の WebSphere Platform Messaging コンポーネントがチャネルおよび RFH2 サブスクリプションを介して Websphere Message Broker に接続する方法のため、WebSphere MQ はその経路でパブリケーションを受け取ったときにも指紋を作成します。

この目的は、アプリケーションが RFH2 を必要としないのに、指紋情報を追加したからというだけの理由で、アプリケーションに何らかの RFH2 を送信することがないようにすることです。

RFH2 がメッセージ・プロパティーに変換されるたびに、< ibm> フォルダーも変換する必要があります。これにより、Websphere MQ V7.0 API を使用したアプリケーションに渡されるか配信される RFH2 から指紋情報が削除されます。

指紋情報を含んでいるメッセージを RFH1 サブスクライバーに送信したり、Websphere Message Broker V6.0 に渡したりするときにはいつでも、指紋情報が RFH1 に変換されます。

Websphere Message Broker V6.0 がそのメッセージを SIB などの RFH2 サブスクライバーに渡すときには、指紋情報を変換して RFH2 形式に戻す必要があります。

JMS アプリケーションは指紋情報を参照しません。JMS インターフェースは、RFH2 から指紋情報を抽出しないので、JMS アプリケーションにその情報を渡さないからです。

Rfp メッセージ・プロパティーは、 propDesc.CopyOptions = MQCOPY_FORWARD and MQCOPY_PUBLISH で作成されます。 これは、メッセージを受け取ってその同じメッセージをリパブリッシュするアプリケーションに影響します。 つまり、このようなアプリケーションは、PutMsgOpts.Action = MQACTP_FORWARD を使用して指紋のルーティング・チェーンを続行できますが、それ自体の指紋をチェーンから削除するには、適切にコーディングする必要があります。 デフォルトでは、アプリケーションは PutMsgOpts.Action = MQACTP_NEW を使用し、新しいチェーンを開始します。

分散パブリッシュ/サブスクライブ・トポロジーでの保存パブリケーション

分散パブリッシュ/サブスクライブ・トポロジーで保存パブリケーションを使用する場合のベスト・プラクティスは、トポロジー内の 1 つのキュー・マネージャーにある同じトピックで、保存パブリケーションのみをパブリッシュすることです。

それ以外の場合、同じトピックに関して、別々のキュー・マネージャーで別々の保存パブリケーションがアクティブになり、予期しない動作が発生することがあります。 複数のプロキシー・サブスクリプションが配布されると、複数の保存パブリケーションを受け取ることになる場合があります。

キュー・マネージャー間におけるパブリッシュ/サブスクライブのセキュリティー

パブリッシュ/サブスクライブの内部メッセージ (プロキシー・サブスクリプションやパブリケーションなど) は、通常のチャネル・セキュリティー規則に基づいてパブリッシュ/サブスクライブのシステム・キューに書き込まれます。 このトピックでは、いくつかの図を交えながら、それらのメッセージの送信に必要な各種のプロセスとユーザー ID について特に説明します。

ローカル・アクセス制御

パブリケーションとサブスクリプションのためのトピックに対するアクセス権限を制御するには、ローカル・セキュリティー定義と規則を使用します (パブリッシュ/サブスクライブのセキュリティーを参照)。 z/OS では、アクセス制御を設定するためにローカル・トピック・オブジェクトは必要ありません。 その他のプラットフォームでも、アクセス制御のためのローカル・トピックは必要ありません。 管理者は、クラスター・トピック・オブジェクトがクラスターに依然として含まれているかどうかにかかわりなく、クラスター・トピック・オブジェクトにアクセス制御を適用することを選択できます。

システム管理者は、自身のローカル・システムのアクセス制御を担当します。 システム管理者は、階層またはクラスター集合の他のメンバーの管理者が、それぞれ責任を持ってアクセス制御ポリシーを実行していることを信頼する必要があります。 アクセス制御は個々のマシンごとに定義するので、細かいレベルでの制御が必要になると、作業が煩雑になるおそれがあります。 アクセス制御を実施する必要がない場合もありますが、トピック・ツリー内の上位オブジェクトでアクセス制御を定義することができます。 トピック名前空間のサブディビジョンごとに細かくアクセス制御を定義することもできます。

プロキシー・サブスクリプションの作成

他の組織のキュー・マネージャーがこちら側のキュー・マネージャーに接続する場合は、通常のチャネル認証手段によって、その組織を信頼できるかどうかが確認されます。 その組織が信頼でき、分散パブリッシュ/サブスクライブの実行も許可されると、権限検査が行われます。 権限検査は、チャネルが分散パブリッシュ/サブスクライブ・キューにメッセージを書き込む時点で行われます。 例えば、SYSTEM.INTER.QMGR.CONTROL キューにメッセージを書き込む時点で行われます。 キューの権限検査の対象になるユーザー ID は、受信側チャネルの PUTAUT 値によって決まります。 例えば、その値とプラットフォームによっても異なりますが、チャネルのユーザー ID、MCAUSER、メッセージ・コンテキストなどです。 チャネルのセキュリティーについて詳しくは、チャネル・セキュリティーを参照してください。

プロキシー・サブスクリプションは、リモート・キュー・マネージャーの分散パブリッシュ/サブスクライブ・エージェントのユーザー ID で作成されます。 例えば、111 ページの図 38 では QM2 がそれに該当します。 そのユーザー ID はシステムで定義されており、ドメインの競合がないため、ユーザーにはローカル・トピック・オブジェクト・プロファイルに対するアクセス権限がすぐに付与されます。

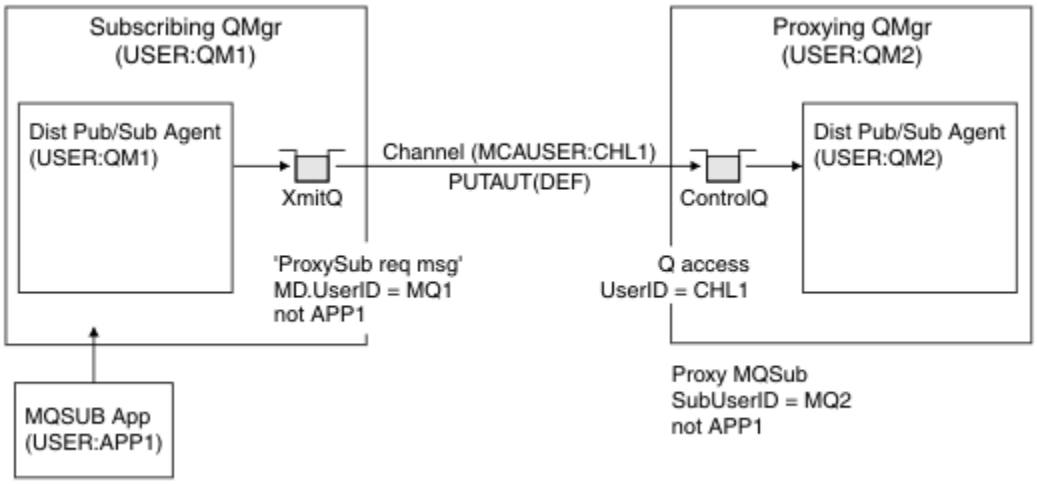


図 *38.* プロキシー・サブスクリプションのセキュリティー *(*サブスクリプションの作成*)*

リモート・パブリケーションを送り返す操作

パブリッシュ側のキュー・マネージャーでパブリケーションが作成されるときには、任意のプロキシー・サブスクリプションにパブリケーションのコピーが作成されます。 サブスクリプションを作成したユーザー ID (111 ページの図 39 では QM2) のコンテキストが、コピーされたパブリケーションのコンテキストに入ります。 プロキシー・サブスクリプションは、リモート・キューである宛先キューと共に作成されるため、パブリケーション・メッセージは伝送キューに解決されます。

他の組織のキュー・マネージャー QM2 が別のキュー・マネージャー QM1 に接続する場合、通常のチャネル認証手段によって、その組織を信頼できるかどうかが確認されます。 その組織が信頼でき、分散パブリッシュ/サブスクライブを実行することが許可されると、チャネルが分散パブリッシュ/サブスクライブ・パブリケーション・キュー SYSTEM.INTER.QMGR.PUBS にパブリケーション・メッセージを書き込む時点で、権限検査が行われます。 キューの権限検査の対象になるユーザー ID は、受信側チャネルの PUTAUT 値によって決まります (例えば、その値とプラットフォームによっても異なりますが、チャネルのユーザー ID、MCAUSER、メッセージ・コンテキストなどになります)。 チャネルのセキュリティーについて詳しくは、チャネル・セキュリティーを参照してください。

パブリケーション・メッセージがサブスクライブ側のキュー・マネージャーに達すると、そのキュー・マネージャーの権限で、そのトピックに対するもう 1 つの MQPUT が実行され、各ローカル・サブスクライバーがメッセージを受け取るたびに、メッセージのコンテキストが各ローカル・サブスクライバーのコンテキストに置き換えられます。

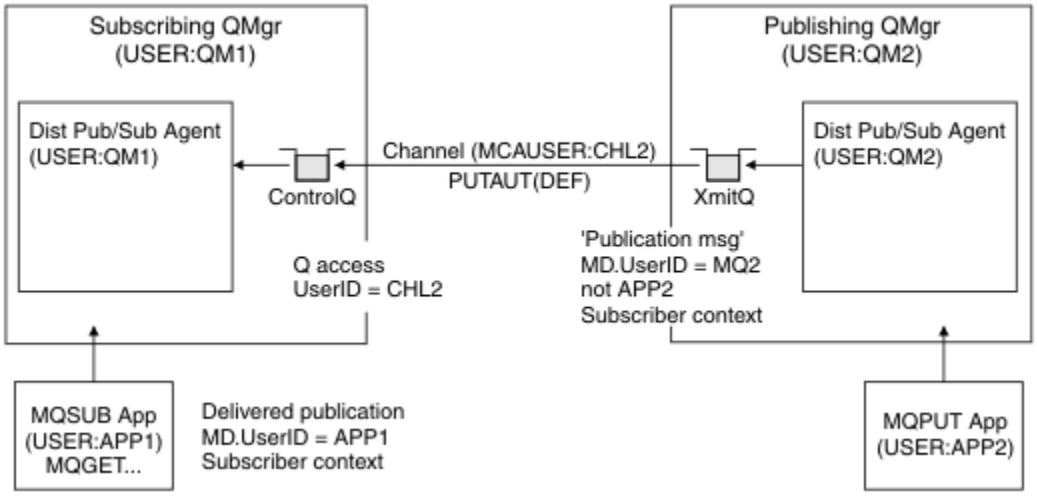


図 *39.* プロキシー・サブスクリプションのセキュリティー *(*パブリケーションの転送*)*

セキュリティーを重視していないシステムでは、ほとんどの場合、分散パブリッシュ/サブスクライブ・プロセスを mqm グループのユーザー ID で実行しており、チャネルの MCAUSER パラメーターはブランク (デフォルト) になっていて、メッセージは必要に応じてさまざまなシステム・キューに送信されます。 そのようなセキュリティーで保護されていないシステムでは、分散パブリッシュ/サブスクライブの PoC (概念検証) を簡単にセットアップできます。

セキュリティーを重視するシステムでは、それらの内部メッセージも、チャネルを通過する他のあらゆるメッセージと同じセキュリティー制御の対象になります。

チャネルのセットアップで非ブランクの MCAUSER を指定し、その MCAUSER をチェックするように PUTAUT 値を指定した場合は、対象の MCAUSER に SYSTEM.INTER.QMGR.* キューに対するアクセス権限を与える必要があります。 複数の異なるリモート・キュー・マネージャーがあり、それぞれが異なる MCAUSER ID でチャネルを実行する場合は、そのすべてのユーザー ID に SYSTEM.INTER.QMGR.* キューに対するアクセス権限を与える必要があります。 例えば、1 つのキュー・マネージャーで複数の階層接続を構成する場合に、異なる MCAUSER ID でチャネルが実行されることがあります。

チャネルのセットアップで、メッセージのコンテキストを使用するように PUTAUT 値を指定した場合は、内部メッセージの中で指定されているユーザー ID に基づいて、SYSTEM.INTER.QMGR.* キューに対するアクセス権限が検査されます。 それらのメッセージはすべて、内部メッセージまたはパブリケーション・メッセージを送信するキュー・マネージャーから、分散パブリッシュ/サブスクライブ・エージェントのユーザー ID と共に書き込まれるので (111 ページの図 39 を参照)、そのような方法で分散パブリッシュ/サブスクライブのセキュリティーをセットアップする場合は、さまざまなシステム・キューに対するアクセス権限を与えるユーザー ID のセットがそれほど大きくなりません (リモート・キュー・マネージャーごとに 1 つです)。 それでも、チャネル・コンテキスト・セキュリティーに伴う問題はすべて残ります。つまり、さまざまなユーザー ID ドメインの問題や、メッセージの中で指定されているユーザー ID が受信側のシステムで定義されていない場合がある、といった問題です。 それでも、必要であれば、このような実行方法は完全に有効です。

分散パブリッシュ/サブスクライブのための内部キュー・マネージャー・メッセージングはすべて、通常のチャネル・セキュリティーで実行されます。

トピック・レベルでのパブリケーションとプロキシー・サブスクリプションの制限については、 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。

キュー・マネージャー階層でのデフォルトのユーザー ID の使用

異なるプラットフォームで実行されているキュー・マネージャーの階層があり、デフォルトのユーザー ID を使用している場合、そのデフォルトのユーザー ID はプラットフォームによって異なり、ターゲット・プラットフォームで認識されない可能性があることに注意してください。 結果として、一方のプラットフォーム上で実行されているキュー・マネージャーは、もう一方のプラットフォーム上のキュー・マネージャーから受信するメッセージを、理由コード MQRC_NOT_AUTHORIZED で拒否します。

メッセージが拒否されないようにするには、もう一方のプラットフォーム上で使用されるデフォルトのユーザー ID に対して、少なくとも以下の権限を追加する必要があります。

- SYSTEM.BROKER における *PUT *GET 権限 キュー
- SYSTEM.BROKER における *PUB *SUB 権限 トピック
- SYSTEM.BROKER.CONTROL.QUEUE キューにおける *ADMCRT *ADMDLT *ADMCHG 権限

デフォルトのユーザー ID は次のとおりです。

プラットフォーム	デフォルト・ユーザー ID
Windows	MUSR_MQADMIN **注:** MUSR_MQADMIN は、最初のインストールのみに対するデフォルトのユーザー ID です。 以降のインストールでは、IBM WebSphere MQ の準備ウィザードによって、MUSR_MQADMINx という名前のユーザー・アカウントが作成されます。ここで x は、まだ存在しないユーザー ID を示す、次に使用可能な番号です。

プラットフォーム	デフォルト・ユーザー ID
UNIX and Linux システム	mqm
IBM i	QMQM
z/OS	チャネル・イニシエーター・アドレス・スペースのユーザー ID

Windows、UNIX、Linux、および z/OS プラットフォーム上のキュー・マネージャーにおいて、IBM i 上のキュー・マネージャーに階層的に接続する場合、「qmqm」ユーザー ID を作成しアクセス権限を付与します。

IBM i および z/OS プラットフォーム上のキュー・マネージャーにおいて、Windows、UNIX、または Linux 上のキュー・マネージャーに階層的に接続する場合、「mqm」ユーザー ID を作成してアクセス権限を付与します。

Windows、UNIX、Linux、および IBM i プラットフォーム上のキュー・マネージャーにおいて、z/OS 上のキュー・マネージャーに階層的に接続する場合、z/OS チャネル・イニシエーター・アドレス・スペース・ユーザー ID を作成してユーザー・アクセス権限を付与します。

ユーザー ID には大/小文字の区別があります。 発信側のキュー・マネージャー (IBM i、Windows、UNIX、または Linux システムの場合) では、ユーザー ID をすべて強制的に大文字に変換します。 受信側のキュー・マネージャー (Windows、UNIX または Linux システムの場合) では、ユーザー ID をすべて強制的に小文字に変換します。 そのため、UNIX and Linux システムでは常に小文字の形式でユーザー ID を作成する必要があります。 メッセージ出口がインストールされている場合には、ユーザー ID が強制的に大文字または小文字に変換されることはありません。 メッセージ出口がユーザー ID を処理する方法をよく理解する必要があります。

ユーザー ID の変換での潜在的な問題を回避するには、以下のようにします。

- UNIX、Linux および Windows システムでは、ユーザー ID が小文字で指定されていることを確認します。
- IBM i および z/OS では、ユーザー ID が大文字で指定されていることを確認します。

分散パブリッシュ/サブスクライブのシステム・キュー

キュー・マネージャーは、パブリッシュ/サブスクライブ・メッセージング用に 4 つのシステム・キューを使用します。 それらのキューの存在について意識する必要があるのは、問題判別またはキャパシティー・プランニングを行う場合のみです。

表 *9.* 分散プラットフォームでのシステム・キューのパブリッシュ/サブスクライブ

システム・キュー	目的
SYSTEM.INTER.QMGR.CONTROL	WebSphere MQ 分散パブリッシュ/サブスクライブの制御キュー
SYSTEM.INTER.QMGR.FANREQ	WebSphere MQ 分散パブリッシュ/サブスクライブの内部プロキシー・サブスクリプション多分岐プロセスの入力キュー
SYSTEM.INTER.QMGR.PUBS	WebSphere MQ 分散パブリッシュ/サブスクライブのパブリケーション
SYSTEM.HIERARCHY.STATE	WebSphere MQ 分散パブリッシュ/サブスクライブ階層関係の状態

パブリッシュ/サブスクライブ・システム・キューの属性を 113 ページの表 10 に示します。

表 *10.* パブリッシュ/サブスクライブ・システム・キューの属性

属性	デフォルト値
DEFPSIST	Yes
DEFSOPT	EXC

表 *10.* パブリッシュ/サブスクライブ・システム・キューの属性 *(続き)*	
属性	**デフォルト値**
MAXMSGL	AIX、 HP-UX、 Linux、 IBM i、Solaris、および Windows の各プラットフォームの場合: ALTER QMGR コマンドの MAXMSGL パラメーターの値
MAXDEPTH	999999999
SHARE	なし
STGCLASS	この属性は、z/OS プラットフォームのみで使用されます。

パブリッシュ/サブスクライブ・システム・キューのエラー
分散パブリッシュ/サブスクライブ・キュー・マネージャーのキューが使用不可の場合、エラーが発生することがあります。

多分岐要求キュー SYSTEM.INTER.QMGR.FANREQ が使用不可になっていると、直接接続されているキュー・マネージャーにプロキシー・サブスクリプションを送信しなければならない場合に、MQSUB API は理由コードを受け取り、エラー・メッセージがエラー・ログに書き込まれます。

階層関係状態キュー SYSTEM.HIERARCHY.STATE が使用不可になっていると、エラー・メッセージがエラー・ログに書き込まれ、パブリッシュ/サブスクライブ・エンジンが COMPAT モードになります。

その他の SYSTEM.INTER.QMGR キューが使用不可になっていると、エラー・メッセージがエラー・ログに書き込まれ、機能が無効になっていなくても、パブリッシュ/サブスクライブ・メッセージがリモート・キュー・マネージャーのキューで作成される可能性があります。

親キュー・マネージャー、子キュー・マネージャー、またはパブリッシュ/サブスクライブ・クラスター・キュー・マネージャーに対する伝送キューが使用不可になっていると、以下のようになります。

1. MQPUT API が理由コードを受け取り、パブリケーションは送信されません。
2. 受け取った内部キュー・マネージャー・パブリケーションは入力キューにバックアウトされ、後で再試行されます。バックアウトしきい値に達すると、送達不能キューに書き込まれます。
3. プロキシー・サブスクリプションは多分岐要求キューにバックアウトされ、後で再試行されます。バックアウトしきい値に達すると、送達不能キューに書き込まれます。その場合、プロキシー・サブスクリプションは、接続されているどのキュー・マネージャーにも送信されません。
4. 階層関係プロトコル・メッセージは失敗し、PUBSUB コマンドの接続状況が ERROR になります。

WebSphere MQ 送達不能キュー・ハンドラーを使用した、未配布メッセージの処理

送達不能キューの内容、このキューにメッセージを配置する方法、 およびその管理方法を示します。

送達不能キュー (DLQ) とは、宛先キューに配布できないメッセージが入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。 ネットワーク内のすべてのキュー・マネージャーが、関連した DLQ を持つ必要があります。

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。

キュー・マネージャーまたはメッセージ・チャネル・エージェントが DLQ に書き込むメッセージには、常に MQDLH があります。メッセージを DLQ に書き込むアプリケーションは MQDLH を提供している必要があります。 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っています。

すべての WebSphere MQ 環境には、DLQ 上のメッセージを定期的に処理するルーチンが必要です。
WebSphere MQ は、送達不能キュー・ハンドラー (DLQ ハンドラー) と呼ばれるデフォルト・ルーチンを提供しています。DLQ ハンドラーは、`runmqdlq` コマンドを使用して呼び出します。

DLQ 上のメッセージを処理する命令は、ユーザー作成ルール・テーブル を介して DLQ ハンドラーに提供されます。 つまり、DLQ ハンドラーは DLQ 上のメッセージとルール・テーブルの項目の突き合わせを行います。DLQ メッセージがルール・テーブルの項目と一致すると、DLQ ハンドラーはその項目に関連付けられたアクションを実行します。

DLQ ハンドラーの起動

DLQ ハンドラーは、runmqdlq コマンドを使用して呼び出します。 処理する DLQ の名前および使用するキュー・マネージャーの名前を指定するには、次の 2 つの方法があります。

2 つの方法は次のとおりです。

- runmqdlq へのパラメーターとしてコマンド・プロンプトから指定する。 以下に例を示します。

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

- 規則テーブルで指定する。 以下に例を示します。

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

この例は、キュー・マネージャー ABC1.QUEUE.MANAGER が所有する ABC1.DEAD.LETTER.QUEUE という DLQ に適用されます。

DLQ またはキュー・マネージャーを例に示すように指定しなかった場合は、インストール先のデフォルト・キュー・マネージャーと共に、そのキュー・マネージャーの DLQ が使用されます。

runmqdlq コマンドは、stdin から入力内容を取ります。ユーザーはルール・テーブルから stdin をリダイレクトして、ルール・テーブルを runmqdlq に関連付けます。

DLQ ハンドラーを実行するためには、DLQ 自体、および DLQ 上のメッセージの転送先となるあらゆるメッセージ・キューの両方へのアクセスが許可されていることが必要です。 DLQ ハンドラーがメッセージ・コンテキスト中のユーザー ID の権限を使用してキューにメッセージを書き込むことが可能になっている場合には、DLQ ハンドラーを実行するユーザーは他のユーザーの ID を借用する許可を持っている必要があります。

runmqdlq コマンドの詳細については、runmqdlq を参照してください。

サンプル *DLQ* ハンドラー *amqsdlq*

runmqdlq コマンドを使用して呼び出される DLQ ハンドラーの他に、WebSphere MQ は、runmqdlq によって提供されるものと同様の機能があるサンプル DLQ ハンドラー amqsdlq のソースを提供します。

この amqsdlq をカスタマイズして、要件に適合した DLQ ハンドラーを与えることができます。 例えば、送達不能ヘッダーのないメッセージを処理できる DLQ ハンドラーが必要となる場合があります。 (デフォルト DLQ ハンドラーとサンプルの amqsdlq は、両方共、DLQ 上のメッセージのうち送達不能ヘッダー MQDLH で始まるもののみを処理するようになっています。 MQDLH で始まらないメッセージはエラーとして識別され、DLQ 上にいつまでも残ることになります。)

MQ_INSTALLATION_PATH は、WebSphere MQ がインストールされている上位ディレクトリーを表します。

WebSphere MQ for Windows では、amqsdlq のソースは、次のディレクトリーに提供されています。

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

また、コンパイル済みバージョンは次のディレクトリー内にあります。

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

WebSphere MQ for UNIX and Linux システムでは、amqsdlq のソースは、次のディレクトリーに提供されています。

MQ_INSTALLATION_PATH/samp/dlq

また、コンパイル済みバージョンは次のディレクトリー内にあります。

MQ_INSTALLATION_PATH/samp/bin

DLQ ハンドラーの規則テーブル

DLQ ハンドラーのルール・テーブルは、DLQ に到着したメッセージを DLQ ハンドラーがど のように処理するかを定義するものです。

規則テーブルの項目には、次の 2 つのタイプがあります。

- テーブルの最初の項目は制御データ で、この項目はオプションです。
- 表中の他のすべての項目は、DLQ ハンドラーが従う規則 です。 各規則は、メッセージを突き合わせるパターン (一連のメッセージ特性) と、指定したパターンと DLQ 上のメッセージが一致したときに行われるアクション で構成されます。 規則テーブルには、規則が少なくとも 1 つ必要です。

規則テーブルの各項目は、1 つ以上のキーワードから構成されます。

制御データ

このセクションでは、DLQ ハンドラーの規則テーブルの制御データ項目に入れることができるキーワードについて説明します。

注 :

- 縦線 (|) によって、代替の値を区切っています。値のうちの 1 つのみを指定できます。
- キーワードはすべてオプションです。

INPUTQ (*QueueName*| '__')
処理対象の DLQ の名前です。

1. INPUTQ 値を runmqdlq コマンドのパラメーターとして指定すると、ルール・テーブル内の INPUTQ 値がそれで指定変更されます。
2. runmqdlq コマンドのパラメーターとして INPUTQ 値を指定しないで、ルール・テーブルの中に値を**指定する**と、ルール・テーブルの INPUTQ 値が使用されます。
3. DLQ を指定しなかった場合、またはルール・テーブルの中で INPUTQ(' ') を指定した場合は、runmqdlq コマンドにパラメーターとして指定した名前を持つキュー・マネージャーに属する DLQ の名前が使用されます。
4. INPUTQ 値を runmqdlq コマンドのパラメーターとしても、ルール・テーブルの中の値としても指定しなかった場合は、ルール・テーブル内の INPUTQM キーワードで指定したキュー・マネージャーが所有する DLQ が使用されます。

INPUTQM (*QueueManager* 名| '__')
INPUTQ キーワードで指定した DLQ を所有するキュー・マネージャーの名前。

1. INPUTQM 値を runmqdlq コマンドのパラメーターとして指定すると、ルール・テーブル内の INPUTQM 値がそれで指定変更されます。
2. INPUTQM 値を runmqdlq コマンドのパラメーターとして指定しなかった場合は、ルール・テーブル内の INPUTQM 値が使用されます。
3. キュー・マネージャーを指定しない場合、または INPUTQM(' ') をルール・テーブルの中に指定した場合は、インストール・システムのデフォルト・キュー・マネージャーが使用されます。

RETRYINT (間隔|_60)
最初の試行で処理できなかった DLQ 上のメッセージについて、試行の反復が要求されている場合に DLQ ハンドラーが再処理する間隔 (秒数)。 デフォルトでは、再試行間隔は 60 秒です。

WAIT (YES|NO|*nnn*)
DLQ ハンドラーが処理できるメッセージがこれ以上ないことを DLQ ハンドラーが検出したとき、DLQ にメッセージが新たに到着するまで DLQ ハンドラーが待機するかどうかを指定します。

YES
DLQ ハンドラーは際限なく待機します。

NO
DLQ ハンドラーは、DLQ が空になるか、あるいは処理できるメッセージがなくなったことを検出すると終了します。

nnn
キューが空であるか、または DLQ ハンドラーが処理できるメッセージがキューにないことを DLQ ハンドラーが検出した後、メッセージが新たに到着するのを DLQ ハンドラーが *nnn* 秒間だけ待機するようにします。

使用頻度の高い DLQ については WAIT (YES)、使用頻度の低い DLQ については WAIT (NO) または WAIT (*nnn*) を指定します。 DLQ ハンドラーが終了するようにした場合は、トリガー操作によって DLQ ハンドラーを再び呼び出してください。 トリガー操作について詳しくは、 トリガーによる WebSphere MQ アプリケーションの開始 を参照してください。

ルール・テーブルに制御データを組み込む代わりに、`runmqdlq` コマンドの入力パラメーターとして DLQ とそのキュー・マネージャーの名前を指定することもできます。 ルール・テーブルに値を指定し、さらに `runmqdlq` コマンドの入力データとしても値を指定した場合は、`runmqdlq` コマンドに指定された値が優先されます。

ルール・テーブルに制御データ項目を組み込む場合、その項目はテーブル内の**最初の**項目でなければなりません。

規則 (パターンおよびアクション)

パターン・マッチング・キーワード (DLQ 上のメッセージを突き合わせるキーワード)、およびアクション・キーワード (一致するメッセージを DLQ ハンドラーが処理する方法を決定するキーワード) についての説明。 規則の例も提供されています。

パターン照合キーワード

DLQ 上のメッセージを突き合わせる値を指定するために使用する パターン・マッチング・キーワードは、次のとおりです。 (すべてのパターン・マッチング・キーワードは、オプションです)

APPLIDAT (*ApplIdentity データ*|_*)
DLQ 上のメッセージのメッセージ記述子 MQMD に指定した *ApplIdentityData* の値。

APPLNAME (*PutAppl 名*|_*)
DLQ 上のメッセージのメッセージ記述子 MQMD の「 *PutApplName* 」フィールドに指定されている、`MQPUT` または `MQPUT1` 呼び出しを発行したアプリケーションの名前。

APPLTYPE (*PutAppl タイプ*|_*)
DLQ 上のメッセージのメッセージ記述子 MQMD に指定した *PutApplType* 値。

DESTQ (*QueueName*|_*)
メッセージの送り先のメッセージ・キューの名前。

DESTQM (*QueueManager 名前*|_*)
メッセージの宛先であるメッセージ・キューのキュー・マネージャーの名前。

FEEDBACK (フィードバック|_*)
MsgType 値が MQFB_REPORT の場合、*Feedback* はレポートの性質を記述します。

シンボル名を使用できます。 例えば、シンボル名 MQFB_COA を使用して、DLQ 上のメッセージのうち、ターゲット・キューへの到着の確認が必要なものを識別することができます。

FORMAT (フォーマット|_*)
メッセージ・データの形式を記述するためにメッセージの送信側が使用する名前。

MSGTYPE (*MsgType*|_*)
DLQ 内のメッセージのメッセージ・タイプ。

シンボル名を使用できます。 例えば、シンボル名 MQMT_REQUEST を使用して、DLQ 上のメッセージのうち応答が必要なものを識別することができます。

PERSIST (永続性|_*)
メッセージの永続値。 (この永続値によって、キュー・マネージャーの再始動後もメッセージが保存されるかどうかが決まります。)

シンボル名を使用できます。 例えば、シンボル名 MQPER_PERSISTENT を使用して、DLQ 上のメッセージのうち持続するものを識別することができます。

REASON (*ReasonCode*|_*)
メッセージが DLQ に書き込まれた理由を説明する理由コード。

シンボル名を使用できます。 例えば、シンボル名 MQRC_Q_FULL を使用して、宛先キューが満杯であったために DLQ に書き込まれたメッセージを識別することができます。

REPLYQ (*QueueName*|_*)
DLQ 上のメッセージのメッセージ記述子 MQMD に指定した応答先キューの名前。

REPLYQM (*QueueManager* 名前|_*)
DLQ 上のメッセージのメッセージ記述子 MQMD に指定した応答先キューのキュー・マネージャーの名前。

USERID (*UserIdentifier*|_*)
DLQ 上のメッセージのメッセージ記述子 MQMD に指定された、DLQ 上のメッセージを発信したユーザーのユーザー ID。

アクション・キーワード

一致するメッセージの処理方法の記述に使用されるアクション・キーワードは、次のとおりです。

ACTION (DISCARD|IGNORE|RETRY|FWD)
このルールに定義したパターンに一致する DLQ 上のメッセージに関して実行されるアクション。

DISCARD
メッセージは DLQ から削除されます。

IGNORE
メッセージは DLQ 上に残されます。

再試行
メッセージをターゲット・キューに入れる最初の試行が失敗した場合に、再試行します。 RETRY キーワードは、1 つのアクションをインプリメントするために行われる試行回数を設定します。 試行相互間の間隔は、制御データの RETRYINT キーワードで制御されます。

FWD
FWDQ キーワードで指定されたキューにメッセージが転送されます。

ACTION キーワードは必ず指定する必要があります。

FWDQ (*QueueName*| & DESTQ | & REPLYQ)
ACTION (FWD) を要求したときのメッセージの転送先となるメッセージ・キューの名前。

QueueName
メッセージ・キューの名前。 FWDQ(' ') は無効です。

&DESTQ
MQDLH 構造体の「*DestQName*」フィールドからキュー名を取得します。

&REPLYQ
メッセージ記述子 MQMD の「*ReplyToQ*」フィールドからキュー名を取得します。

FWDQ (& REPLYQ) を指定したルールがブランクの *ReplyToQ* フィールドを持つメッセージと一致する場合のエラー・メッセージを回避するには、メッセージ・パターンに REPLYQ (? *) を指定します。

FWDQM (*QueueManager* 名| & DESTQM | & REPLYQM | '__')
メッセージの転送先となるキューのキュー・マネージャー。

QueueManagerName
ACTION (FWD) を要求したときのメッセージの転送先となるキューのキュー・マネージャーの名前。

&DESTQM
MQDLH 構造体の「*DestQMgrName*」フィールドからキュー・マネージャー名を取得します。

&REPLYQM
メッセージ記述子 MQMD の「*ReplyToQMgr*」フィールドからキュー・マネージャー名を取得します。

' '
FWDQM(' ') がデフォルト値です。 この値は、ローカル・キュー・マネージャーを識別します。

HEADER (YES|NO)
ACTION (FWD) が要求されたメッセージに MQDLH を残すかどうかを指定します。 デフォルトでは、MQDLH はメッセージに残ります。 HEADER キーワードは、FWD 以外のアクションには無効です。

PUTAUT (DEF|CTX)
DLQ ハンドラーがメッセージを書き込む際の権限。

DEF
メッセージは DLQ ハンドラー自体の権限で書き込まれます。

CTX
メッセージはメッセージ・コンテキストの中のユーザー ID の権限で書き込まれます。 PUTAUT (CTX) を指定する場合、他のユーザーの ID を使用することが許可されている必要があります。

RETRY (*RetryCount*|1)
1 から 999 999 999 までの範囲の数値で、(制御データの RETRYINT キーワードに指定されている間隔で) アクションを試行する回数。 DLQ ハンドラーが特定の規則を実行するために行う試行回数は、DLQ ハンドラーの現行インスタンスに特有のものであり、再始動後には持ちこされません。 DLQ ハンドラーが再始動すると、あるルールに適用された試行のカウントはゼロにリセットされます。

規則の例

次に、DLQ ハンドラー規則テーブルの規則の一例を示します。

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

この規則は、MQPUT および MQPUT1 の使用が禁止されたため DLQ に書き込まれた持続メッセージをその宛先キューに送達する試みを 3 回行うように、DLQ ハンドラーに指示しています。

ルールに使用できるすべてのキーワードについては、このセクションであらためて説明します。 次の事項に注意してください。

- キーワードのデフォルト値 (ある場合) には、下線が引いてあります。 ほとんどのキーワードで、デフォルト値は、すべての値と一致する * (アスタリスク) です。
- 縦線 (|) によって、代替の値を区切っています。値のうちの 1 つのみを指定できます。
- ACTION を除いて、どのキーワードも指定は任意です。

規則テーブルの規則

DLQ ハンドラーのルール・テーブルの構文、構造、 および内容は、以下の規則に従う必要があります。

ルール・テーブルは、以下の規則に従う必要があります。

- 規則テーブルには少なくとも 1 つの規則が必要です。
- キーワードは、任意の順序で組み込むことができます。
- キーワードは、どのルールにも 1 回のみ指定できます。
- キーワードには大文字小文字の区別はありません。
- 1 つ以上のブランクまたはコンマでキーワードとパラメーター値を他のキーワードと区切る必要があります。
- ルールの始めまたは終わり、およびキーワード、句読点、値の間には、ブランクをいくつ入れても構いません。

- 各規則ごとに改行する必要があります。
- Windows システムでは、テーブルの最後のルールは、 復帰/改行文字で終わる必要があります。 これを行うには、 テーブルの最終行がブランク行になるように、 ルールの最後で Enter キーを押します。
- 移植性のために、行の有効長は 72 文字を超えないようにする必要があります。
- 次行の最初の非ブランク文字にルールが継続するよう指示するには、行の最後の非ブランク文字として正符号 (+) を使用します。 次行の先頭にルールが継続するよう指示するには、行の最後の非ブランク文字として負符号 (-) を使用します。 連結文字がキーワードおよびパラメーターの内部に現れても構いません。

  以下に例を示します。

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

  と指定すると 'ABCD' となり、

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

  これは 'ABC D' となります。
- 注釈行は、アスタリスク (*) で始まり、規則テーブルのどの位置にでも含めることができます。
- ブランク行は無視されます。
- DLQ ハンドラーのルール・テーブルの各項目は、1 つ以上のキーワードと、それらに関連付けられたパラメーターからなります。パラメーターは、次の構文規則に従う必要があります。
  - 各パラメーター値は、有効な文字を 1 つ以上含んでいる必要があります。 引用符で囲んだ値の区切り用の引用符は、無効とみなされます。 例えば、次のパラメーターは有効です。

FORMAT('ABC')	有効文字数は 3
FORMAT(ABC)	有効文字数は 3
FORMAT('A')	有効文字数は 1
FORMAT(A)	有効文字数は 1
FORMAT(' ')	有効文字数は 1

    次のパラメーターは、有効文字を含んでいないので無効です。

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

  - ワイルドカード文字はサポートされています。 後書きブランク以外の 1 文字の代わりに疑問符 (?) を使用し、また 0 個以上の隣接した文字の代わりにアスタリスク (*) を使用することができます。 アスタリスク (*) および疑問符 (?) は、パラメーター値の中では **常に** ワイルドカード文字と解釈されます。
  - 次のキーワードのパラメーターの中には、ワイルドカード文字を含めることはできません。ACTION、HEADER、RETRY、FWDQ、FWDQM、および PUTAUT。
  - パラメーター値の中の後書きブランク、および DLQ 上のメッセージ内のそれに対応するフィールドの中の後書きブランクは、ワイルドカード突き合わせの実行時には無効です。 ただし、引用符で囲まれたストリング内の先行および組み込みブランクは、ワイルドカード照合時に有効です。
  - 数値パラメーターには、疑問符 (?) のワイルドカード文字を含めることはできません。 1 個の数値パラメーター全体の代わりにアスタリスク (*) を使用できますが、数値パラメーターの一部分として含めることはできません。 例えば、次の数値パラメーターは有効です。

MSGTYPE(2)	応答メッセージのみが対象

MSGTYPE(*)	あらゆるメッセージ・タイプが対象
MSGTYPE('*')	あらゆるメッセージ・タイプが対象

しかし、MSGTYPE('2*') の場合は、アスタリスク (*) を数値パラメーターの一部として使用しているため、無効です。

- 数値パラメーターは 0 から 999 999 999 の範囲内でなければなりません。 パラメーター値がこの範囲内であるなら、キーワードが関連するフィールドで現在無効であっても、パラメーター値は受け入れられます。 数値パラメーターには、シンボル名を使用することができます。
- キーワードが関連する MQDLH または MQMD 内のフィールドよりも ストリング値が短い場合、そのストリング値は、フィールドの長さになるまでブランクが埋め込まれます。 ストリング値 (アスタリスクを除外して) がフィールドより長い場合は、エラーの診断が下されます。 例えば、次のストリング値は、8 文字のフィールドに関してすべて有効です。

'ABCDEFGH'	8 文字
'A*C*E*G*I'	アスタリスクを除く 5 文字
'*A*C*E*G*I*K*M*O*'	アスタリスクを除く 8 文字

- ブランク、小文字、または特殊文字 (ピリオド (.)、スラッシュ (/)、下線 (_)、およびパーセント記号 (%) を除く) が使用されているストリングは、単一引用符で囲みます。 引用符で囲まれていない小文字は大文字に変換されます。 ストリングが引用符を含む場合は、その引用符の始めと終わりの両方を示すために、2 個の単一引用符を使用します。 ストリングの長さを計算するとき、二重引用符はすべて 1 文字としてカウントされます。

ルール・テーブルの処理方法

DLQ ハンドラーは、パターンが DLQ 内のメッセージと一致している規則を規則テーブルから探します。

検索は、規則テーブルの最初の規則から始まって、テーブル中を順番に進みます。 DLQ ハンドラーは一致するパターンを持つルールを見つけると、そのルールの処理を実行します。 DLQ ハンドラーは、そのルールを適用するたびに、ルールの再試行カウントを 1 つずつ増分します。 最初の試行が失敗すると、DLQ ハンドラーは、なされた試行数が RETRY キーワードに指定された数と一致するまで試行を繰り返します。 試行がすべて失敗すると、DLQ ハンドラーは、ルール・テーブルの中の次に一致するルールを検索します。

このプロセスは、アクションが正常に実行されるまで、一致するルールについて順番に繰り返されます。 一致する規則がそれぞれ RETRY キーワードで指定されている回数だけ試行され、その試行がすべて失敗した場合は、ACTION (IGNORE) であると見なされます。 一致する規則が見つからないときにも、ACTION (IGNORE) であると見なされます。

注 :

1. 一致する規則のパターンは、接頭部が MQDLH の DLQ 上のメッセージについてのみ検索されます。 接頭部が MQDLH 以外のメッセージは、エラーとして定期的に報告され、DLQ 上にいつまでも残ります。
2. すべてのパターン・キーワードを、デフォルトにして、ルールがアクションのみで構成されるようにすることができます。 ただし、そのキューにおいて、MQDLH が付いているメッセージのうち、テーブル内のその他のルールに従ってまだ処理されていないすべてのメッセージに、そのアクションのみのルールが適用されることに注意してください。
3. ルール・テーブルは、DLQ ハンドラーの開始時に検査され、その時点でエラーのフラグが付けられます。 ルール・テーブルにはいつでも変更を加えることができますが、DLQ ハンドラーが再始動されないと、その変更は有効になりません。
4. DLQ ハンドラーは、メッセージ、MQDLH、メッセージ記述子のいずれの内容も変更しません。 DLQ ハンドラーは、常にメッセージ・オプション MQPMO_PASS_ALL_CONTEXT を使用して、メッセージを他のキューに書き込みます。
5. ルール・テーブルで構文エラーが連続して発生しても認識されないことがあります。それは、ルール・テーブルは妥当性検査中に繰り返し発生するエラーを排除するように設計されているためです。
6. DLQ ハンドラーは MQOO_INPUT_AS_Q_DEF オプションで DLQ を開きます。

7. DLQ ハンドラーの複数インスタンスは、同一の規則テーブルを使用して、同一キューについて並行して実行されることがあります。 ただし、一般に DLQ と DLQ ハンドラー間には 1 対 1 の関係があります。

すべての *DLQ* メッセージを確実に処理する

DLQ ハンドラーは、すでに参照されたが除去されていない DLQ 上の メッセージをすべて記録しています。

DLQ からメッセージの小さいサブセットを 抽出するためのフィルターとして DLQ ハンドラーを使用する場合にも、DLQ ハンドラーは、DLQ 上にある未処理のメッセージの記録を保持し続けます。 また、DLQ が先入れ先出し (FIFO) として定義されても、DLQ に到着する新規メッセージが参照されることを DLQ ハンドラーは保証できません。 キューが空ではないとき、DLQ は定期的に再スキャンされて、すべてのメッセージが検査されます。

以上の点から、DLQ にはできるだけ少数のメッセージを入れるようにしてください。廃棄したり他のキューに転送したりできない (その理由が何であろうと) メッセージをキュー上に累積させると、DLQ ハンドラーのワークロードが増大し、DLQ 自体が満杯になる可能性があります。

DLQ ハンドラーが DLQ を空にできるように適切な処置をとることができます。 例えば、ACTION (IGNORE) は、DLQ 上のメッセージを放置するので、使用しないようにしてください (テーブルの中の他の規則によって明示的に処理されないメッセージには、ACTION (IGNORE) が適用されることに注意してください)。 その代わり、無視するメッセージに関して、別のキューにそのメッセージを移動するアクションを実行してください。 以下に例を示します。

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

また、テーブルの最後の規則は、テーブルの中のそれまでのルールから漏れたメッセージをまとめて扱えるものにします。 例えば、テーブルの中の最後のルールは、次のような形にすることができます。

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

こうすると、テーブルの中の最後のルールが適用されることになったメッセージは、キュー REALLY.DEAD.QUEUE に転送されます。このキューで、そのメッセージを手動によって処理できます。 このような規則がないと、メッセージはいつまでも DLQ に残ることになります。

DLQ ハンドラー規則テーブルの例

1 つの制御データ項目といくつかの規則を含む、runmqdlq コマンドの規則テーブルの例。

```
*****************************************************************************
*        An example rules table for the runmqdlq command                    *
*****************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* runmqdlq, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to runmqdlq,
* use the DLQ defined for the local queue manager.
*
inputqm(' ')  inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.
* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)
```

```
* The AAAA corporation are always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
  ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never do things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
  action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
  ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)
```

```
* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it should be able
* to cope with the message being lost, so we can afford to
* discard the message.   PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)
* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We don't have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

複数のインストール

UNIX, Linux, and Windows では、1 つのシステムで複数の IBM WebSphere MQ のコピーを持つことができます。

IBM WebSphere MQ の各コピーのインストール場所を選択することはできますが、各コピーは別のインストール場所になければなりません。 単一のマシン上で、一度に最大 128 の IBM WebSphere MQ インストールを共存させることができます。 インストールの 1 つは、IBM WebSphere MQ Version 7.0.1 (フィックスパック 6 以上) にすることができます。 次の選択が可能になります。

- マシン上で IBM WebSphere MQ の単一インストールを保守および管理することで簡易性を維持する。
- 複数の IBM WebSphere MQ インストール済み環境を使用可能にすることで提供される柔軟性を活用します。

複数の IBM WebSphere MQ のコピーをインストールする前に、以下の各点について決定してください。

- システムに IBM WebSphere MQ Version 7.0.1 のコピーを含めるか。

  システムに IBM WebSphere MQ Version 7.0.1 (フィックスパック 6 以上) をインストールすると以下のさまざまな制限が生じるので、考慮が必要です。

  - UNIX and Linux システムでは、IBM WebSphere MQ Version 7.0.1 がデフォルトの場所にインストールされている必要があります。
  - IBM WebSphere MQ Version 7.0.1 は、システム上の最初のインストールでなければなりません。 バージョン 7.1 以降をインストールした後に IBM WebSphere MQ Version 7.0.1 をインストールすることはできません。 バージョン 7.0.1 をアンインストールした場合、後のバージョンの WebSphere MQ がインストールされている間は、再インストールできません。
  - IBM WebSphere MQ Version 7.0.1 は、自動的にプライマリー・インストールになります。 IBM WebSphere MQ Version 7.0.1 がインストールされている間は、別のインストールをプライマリー・インストールとして選択することはできません。
- IBM WebSphere MQ の各コピーをどこにインストールするか。

  バージョン 7.1 以上のインストールでは、インストール場所を選択できます。 詳しくは、インストール場所の選択を参照してください。
- プライマリー・インストールが必要ですか。

  プライマリー・インストールとは、システム全体が関係する場所が参照するインストールです。 詳しくは、125 ページの『プライマリー・インストールの選択』 を参照してください。
- アプリケーション間の接続方法。

  アプリケーションが適切な IBM WebSphere MQ ライブラリーを見つける方法について考慮する必要があります。 詳しくは、複数のインストール環境でのアプリケーションの接続、および複数のインストール環境での .NET アプリケーションの接続を参照してください。
- 既存の出口に変更が必要か。

  IBM WebSphere MQ をデフォルトの場所にインストールしない場合、出口を更新する必要があります。 詳しくは、 出口とインストール可能サービスの作成とコンパイルを参照してください。
- どのインストールにどのキュー・マネージャーを関連付けるか。

  各キュー・マネージャーは、特定のインストールに関連付けられています。 キュー・マネージャーが関連付けられているインストールは、そのキュー・マネージャーに制限を課します。 つまり、そのキュー・マネージャーは、そのインストールからのコマンドでしか管理できないのです。 詳しくは、キュー・マネージャーとインストールの関連付けを参照してください。
- 各インストールを操作するために、どのように環境をセットアップするか。

  1 つのシステム上に複数のインストールが存在する場合、特定のインストールを操作するための方法、およびそのインストールからコマンドを発行する方法について考慮する必要があります。 コマンドへの絶対パスを指定する方法と、**setmqenv** または **crtmqenv** コマンドを使用して環境変数を設定する方法があります。 環境変数を設定する方法では、当該インストールのコマンドへのパスを省略することができます。 詳細については、 setmqenv および crtmqenv を参照してください。

これらの質問に回答したら、 IBM WebSphere MQ のインストールに記載されているステップを使用して IBM WebSphere MQ をインストールできます。

IBM WebSphere MQ の既存のインストール済み環境があり、複数インストール機能を使用して IBM WebSphere MQ の 1 つのバージョンから別のバージョンにマイグレーションする場合は、 UNIX、 Linux、および Windows での複数インストール済み環境のキュー・マネージャーの共存 を参照してください。

IBM Message Service Client for .NET サポート・パックと複数インストール

複数バージョンをサポートするには、 IBM WebSphere MQ 製品とともに *Java* メッセージング、*.NET* メッセージング、および *Web* サービス・フィーチャーをインストールする必要があります。 このフィーチャーには、*IBM Message Service Client for .NET* サポート・パック (IA9H) に含まれているすべての機能が含まれています。 システムにサポート・パックがインストールされていると、複数バージョンはサポートされません。 IBM WebSphere MQ をインストールする前に、サポート・パックをアンインストールする必要があります。 .NET フィーチャーをインストールする方法について詳しくは、『WebSphere MQ classes for .NET のインストール』を参照してください。

関連概念

UNIX、Linux、および Windows のバージョン 7.0.1 からバージョン 7.5 への横並びマイグレーション

UNIX、Linux、および Windows のバージョン 7.0.1 からバージョン 7.5 への段階的マイグレーション

関連タスク

複数のインストールの構成

システム上の WebSphere MQ のインストールの見つけ方

プライマリー・インストールの選択

IBM WebSphere MQ (UNIX, Linux, and Windows) の複数インストールをサポートするシステムにおいて、プライマリー・インストールとは IBM WebSphere MQ システム全体が関係する場所が参照するインストールです。 プライマリー・インストールはオプションですが、便利です。

IBM WebSphere MQ Version 7.1 より前は、製品のインスタンスは一度に 1 つしかインストールできませんでした。 Windows システムでは、いくつかのグローバル環境変数がそのインストール済み環境を指すように設定されています。 UNIX and Linux システムでは、シンボリック・リンクが `/usr/lib`、 `/usr/bin`、および `/usr/include` に追加され、その単一インストール済み環境を指すようになりました。

Version 7.1 から、複数のバージョンの IBM WebSphere MQ を UNIX, Linux, and Windows にインストールできます。 これらのシステムのいずれかに、いつでも IBM WebSphere MQ の複数のインストール済み環境を持つことができます。また、オプションで、これらのインストール済み環境の 1 つをプライマリー・インストールとして構成することもできます。 単一のインストールを指す環境変数とシンボリック・リンクは、複数のバージョンが存在している場合にはあまり意味がありません。 ただし、関数によっては、それが機能するために、これらのシステム全体が関係する場所を必要とするものもあります。 例えば、 IBM WebSphere MQ、およびサード・パーティー製品を管理するためのカスタム・ユーザー・スクリプトです。これらの機能はプライマリー・インストールでしか動作しません。

UNIX and Linux システムでは、インストール済み環境をプライマリー・インストールとして設定すると、そのインストール済み環境の外部ライブラリーと制御コマンドへのシンボリック・リンクが `/usr/lib` および `/usr/bin` に追加されます。 プライマリー・インストールがない場合、シンボリック・リンクは作成されません。 プライマリー・インストールに対して作成されるシンボリック・リンクのリストについては、UNIX and Linux 上のプライマリー・インストールへの外部ライブラリーおよび制御コマンドのリンクを参照してください。

Windows システムでは、グローバル環境変数は、プライマリー・インストールがインストールされているディレクトリーを指します。 これらの環境変数は、 IBM WebSphere MQ ライブラリー、制御コマンド、およびヘッダー・ファイルを見つけるために使用されます。 さらに、 Windows システムでは、オペレーティング・システムの一部の機能には、単一プロセスにロードされるインターフェース・ライブラリーの中央登録が必要です。 複数のバージョンの IBM WebSphere MQ を使用すると、競合する IBM WebSphere MQ ライブラリーのセットが存在することになります。 これらの機能は、これら競合するライブラリー・セットを単一のプロセスにロードしようとします。 したがって、そのような機能を使用できるのはプライマリー・インストールの場合だけです。 プライマリー・インストールで使用するように制限されているいくつかの機能について詳しくは、 Windows を参照してください。

システム上に IBM WebSphere MQ Version 7.0.1 のインストール済み環境がある場合、このインストール済み環境が自動的にプライマリー・インストールになります。 Version 7.0.1 がインストールされている間は、プライマリー・インストールを変更できません。 システム上のすべてのインストール済み環境が Version 7.1 以降である場合は、プライマリー・インストールを行うかどうかを選択できます。 126 ページの表 11 のオプションを検討してください。

表 *11.* プライマリー・インストールのオプション.

次の表に、プライマリー・インストールに有効なインストール構成を示します。 単一の Version 7.1 以降では、1 次または非 1 次のいずれかにすることができます。 複数のインストール済み環境がある場合、1 つは Version 7.0.1 で、1 つ以上は Version 7.1 以降で、Version 7.0.1 が 1 次でなければならず、その他のインストール済み環境は非 1 次でなければなりません。 Version 7.1 以降の複数のインストール済み環境では、1 つのインストール済み環境をプライマリーにすることも、すべてのインストール済み環境を非プライマリーにすることもできます。

オプション	**有効なインストール構成**		**詳細情報**
	1 次	非プライマリー	
Version 7.1 以降の単一インストール済み環境。	Version 7.1、またはそれ以降。	なし	以前のリリースと同じように単一のインストールで作業し続ける場合は、そのインストールをプライマリー・インストールとして構成します。 このオプションについては、プライマリー・インストールとして構成されている IBM WebSphere MQ Version 7.1 以降の単一インストールを参照してください。
	なし	Version 7.1、またはそれ以降。	単一のインストールで作業をし続けるものの、シンボリック・リンクまたはグローバル環境変数を作成する必要がない場合は、そのインストールを非プライマリーとして構成します。 このオプションが意味するところについては、非プライマリーとして構成されている IBM WebSphere MQ Version 7.1 以降の単一インストールを参照してください。
複数のインストール: Version 7.0.1 および Version 7.1、またはそれ以降。	Version 7.0.1	Version 7.1、またはそれ以降。	IBM WebSphere MQ の複数のインストール済み環境をバージョン 7.0.1 で使用する場合は、バージョン 7.0.1 のインストール済み環境が自動的にプライマリー・インストールになります。 IBM WebSphere MQ バージョン 7.0.1 がインストールされている間は、どのインストール済み環境がプライマリー・インストールであるかを変更することはできません。 このオプションとその意味については、IBM WebSphere MQ の複数インストール (1 つは Version 7.0.1) を参照してください。
複数のインストール: Version 7.1、またはそれ以降。	Version 7.1、またはそれ以降。	Version 7.1、またはそれ以降。	バージョン 7.1 以上の WebSphere MQ のインストールが複数必要な場合は、それらのインストールの 1 つをプライマリーにするかどうかを選択できます。 このオプションについては、IBM WebSphere MQ Version 7.1 以降の複数インストールを参照してください。
	なし	Version 7.1、またはそれ以降。	

関連概念

プライマリー・インストールとして構成されている WebSphere MQ バージョン 7.1 以降の単一インストール

非プライマリーとして構成されている WebSphere MQ バージョン 7.1 以降の単一インストール

WebSphere MQ バージョン 7.1 以降の複数インストール

1 つのインストールはバージョン 7.0.1 とする WebSphere MQ の複数のインストール

関連タスク

プライマリー・インストールの変更

インストール場所の選択

インストールの計画

インストール名の選択

プライマリー・インストールとして構成されている IBM WebSphere MQ Version 7.1 以降の単一インストール

IBM WebSphere MQ のインストールをプライマリー・インストールとしてマーク付けすると、シンボリック・リンクまたはグローバル環境変数がシステムに追加され、必要最小限のシステム・セットアップで、アプリケーションが使用する IBM WebSphere MQ コマンドとライブラリーが自動的に使用可能になります。

IBM WebSphere MQ をどこにインストールするかを決定してください。

可能な場合には、システム検索パスを使用して IBM WebSphere MQ の制御コマンドまたは IBM WebSphere MQ ライブラリーを見つけられるよう、アプリケーションとスクリプトを構成します。 アプリケーションとスクリプトをこのように構成すると、IBM WebSphere MQ の次回のリリースにマイグレーションする、あるいは 2 番目のインストールをインストールするなど、将来のタスクに取り組む際に最大の柔軟性が得られます。 アプリケーションを接続するためのオプションについて詳しくは、複数のインストール環境でのアプリケーションの接続を参照してください。

Windows では、最初のインストールが自動的にプライマリー・インストールとして構成されます。 UNIX and Linux プラットフォームでは、システムへの最初のインストールをプライマリー・インストールにするには、手動でそのインストールを構成する必要があります。 **setmqinst** コマンドを使用してプライマリー・インストールを設定します。 詳しくは、プライマリー・インストールのアンインストール、アップグレード、および保守を参照してください。

関連タスク

プライマリー・インストールの変更
インストール場所の選択
インストールの計画
インストール名の選択

非プライマリーとして構成されている IBM WebSphere MQ Version 7.1 以降の単一インストール

IBM WebSphere MQ Version 7.1 以降を非プライマリーとしてインストールする場合、IBM WebSphere MQ ライブラリーをロードするために、アプリケーションのライブラリー・パスを構成する必要があります。 Windows では、一部の製品機能は、IBM WebSphere MQ がプライマリーとして構成されている場合にのみ使用可能です。

UNIX および Linux システム

非プライマリーのインストールを UNIX and Linux で実行する影響は、以下のとおりです。

- 組み込みライブラリー・パスを使用して IBM WebSphere MQ ライブラリーを見つけるアプリケーション (例えば、 RPATH) は、以下の条件に該当する場合、それらのライブラリーを見つけることができません。
  - IBM WebSphere MQ が RPATH で指定されているディレクトリーとは異なるディレクトリーにインストールされている。
  - `/usr` にシンボリック・リンクはありません。
- アプリケーションが外部ライブラリー・パス (LD_LIBRARY_PATH など) を使用してライブラリーを見つける場合は、*MQ_INSTALLATION_PATH*`/lib` ディレクトリーまたは *MQ_INSTALLATION_PATH*`/lib64` ディレクトリーを含むように外部ライブラリー・パスを構成する必要があります。 **setmqenv** および **crtmqenv** コマンドは、外部ライブラリーのパスを含む、現在のシェル内の多くの環境変数を構成することができます。
- ほとんどの IBM WebSphere MQ プロセスは、setuid/setgid として実行します。 その結果、それらのプロセスは、ユーザー出口をロードするときに、外部ライブラリーのパスを無視します。 IBM WebSphere MQ のライブラリーを参照しているユーザー出口がそれらのライブラリーを見つけることできるのは、それらの内部に組み込まれているライブラリー・パスで見つかる場合だけです。 これらは、`/usr` にシンボリック・リンクがある場合に解決されます。 IBM WebSphere MQ Version 7.1 以降で実行するよう意図されているユーザー出口については、IBM WebSphere MQ ライブラリーをまったく参照しないように作成する

ことが可能になりました。 代わりにユーザー出口は、IBM WebSphere MQ を介して IBM WebSphere MQ の関数を指す関数ポインターを渡します。 これによりユーザー出口は、その関数を使用することができます。 詳しくは、 出口とインストール可能サービスの作成とコンパイルを参照してください。

アプリケーションを接続するためのオプションについて詳しくは、複数のインストール環境でのアプリケーションの接続を参照してください。

UNIX and Linux プラットフォームの場合、システムへの最初のインストールが自動的にプライマリー・インストールとして構成されるわけではありません。 ただし、**dspmqver** コマンドの場所を特定するための単一のシンボリック・リンクが /usr/bin に組み込まれています。 シンボリック・リンクが不要である場合は、次のコマンドを使用してリンクを削除できます。

```
setmqinst -x -p MQ_INSTALLATION_PATH
```

Windows システム

非プライマリーのインストールを Windows で実行する影響は、以下のとおりです。

- 通常、アプリケーションはそれぞれのライブラリーを外部ライブラリーのパス (PATH) を使用して見つけます。 組み込みのライブラリー・パスあるいは明示的なライブラリーの場所という概念はありません。 インストールが非プライマリーの場合、グローバルな PATH 環境変数には IBM WebSphere MQ のインストール・ディレクトリーは含まれていません。 アプリケーションが IBM WebSphere MQ のライブラリーを見つけるためには、PATH 環境変数を更新して、IBM WebSphere MQ のインストール・ディレクトリーを参照するようにします。 **setmqenv** および **crtmqenv** コマンドは、外部ライブラリーのパスを含む、現在のシェル内の多くの環境変数を構成することができます。
- 一部の製品機能は、インストール済み環境がプライマリー・インストールとして構成されている場合にのみ使用できます。 Windows を参照してください。

Windows の場合、デフォルトでは最初のインストールが自動的にプライマリーとして構成されます。 プライマリー・インストールとして選択されているインストールは、手動でその選択を解除する必要があります。

関連タスク

プライマリー・インストールの変更
インストール場所の選択
インストールの計画
インストール名の選択

関連資料

setmqenv
crtmqenv

IBM WebSphere MQ Version 7.1 以降の複数インストール

IBM WebSphere MQ Version 7.1 以降のインストールの 1 つをプライマリー・インストールとして構成することを選択できます。 この選択は、アプリケーションがライブラリーをどのように見つけるかによって異なります。

IBM WebSphere MQ Version 7.1 に同梱されている IBM WebSphere MQ ライブラリー (mqm など) は、接続先のキュー・マネージャーが必要とするレベルのライブラリーを自動的に使用します。 つまり、アプリケーションがその IBM WebSphere MQ ライブラリーを IBM WebSphere MQ Version 7.1 のインストールから見つけた場合、そのアプリケーションはそのシステムの任意のキュー・マネージャーに接続できます。 ある IBM WebSphere MQ Version 7.1 インストールをプライマリーとして構成しておくと、アプリケーションが自分の IBM WebSphere MQ インターフェース・ライブラリーを見つけた場合、そのアプリケーションは任意のキュー・マネージャーに接続できることが保証されます。

複数インストール環境でのアプリケーションの接続について詳しくは、複数のインストール環境でのアプリケーションの接続を参照してください。

プライマリー・インストールをアンインストールした場合、プライマリー・インストールは自動的に変更されません。 別のインストールをプライマリー・インストールにする場合は、**setmqinst** コマンドを使用

して、手動でプライマリー・インストールを設定する必要があります。 詳しくは、プライマリー・インストールのアンインストール、アップグレード、および保守を参照してください。

関連概念
複数のインストール
関連タスク
プライマリー・インストールの変更
インストール場所の選択
インストールの計画
インストール名の選択

IBM WebSphere MQ の複数インストール (1 つは Version 7.0.1)

IBM WebSphere MQ Version 7.1 以降は、IBM WebSphere MQ Version 7.0.1 と共存できますが、いくつかの制限があります。

- UNIX and Linux システムでは、 Version 7.0.1 は固定のデフォルト・ロケーションにのみインストールできるため、 Version 7.1 以降をそのデフォルト・ロケーションにインストールすることはできません。
- IBM WebSphere MQ Version 7.0.1 は自動的にプライマリー・インストールとして構成されます。 UNIX and Linux システムの場合は、シンボリック・リンクが自動的に適切な IBM WebSphere MQ ディレクトリーに作成されます。 Windows では、製品が提供したものはすべてグローバルに登録されます。IBM WebSphere MQ Version 7.0.1 この方法でインストールする必要があります。 したがって、IBM WebSphere MQ Version 7.0.1 がインストールされている場合、IBM WebSphere MQ Version 7.1 以降のインストール済み環境をプライマリーにすることはできません。

IBM WebSphere MQ Version 7.1 以降のライブラリーは、IBM WebSphere MQ Version 7.0.1 以降で実行されている任意のキュー・マネージャーで操作可能です。 アプリケーションが Version 7.0.1 およびそれ以降のバージョンで実行されているキュー・マネージャーに接続する必要がある場合、以下の条件が満たされていれば正常に動作し続けることができます。

- アプリケーションが実行時に IBM WebSphere MQ Version 7.1 以降のライブラリーを見つけることができる。
- Version 7.0.1 で使用可能な機能のみを使用します。

複数インストール環境でのアプリケーションの接続について詳しくは、複数のインストール環境でのアプリケーションの接続を参照してください。

IBM WebSphere MQ Version 7.0.1 をアンインストールした場合、プライマリー・インストールは自動的に変更されません。 別のインストールをプライマリー・インストールにする場合は、**setmqinst** コマンドを使用して、手動でプライマリー・インストールを設定する必要があります。 詳しくは、プライマリー・インストールのアンインストール、アップグレード、および保守を参照してください。

関連概念
複数のインストール
関連タスク
インストール場所の選択
インストールの計画
インストール名の選択

ストレージおよびパフォーマンスの要件の計画

IBM WebSphere MQ システムに対して現実的で達成可能なストレージおよびパフォーマンスの目標を設定する必要があります。 以下のリンク先で、ご使用のプラットフォームでストレージおよびパフォーマンスに影響を与える要因についての情報を参照してください。

要件は、 IBM WebSphere MQ を使用しているシステム、および使用したいコンポーネントによって異なります。

サポートされるハードウェアおよびソフトウェア環境に関する最新情報については、 IBM WebSphere MQ のシステム要件 Web サイトを参照してください。

```
www.ibm.com/software/integration/wmq/requirements/
```

IBM WebSphere MQ は、キュー・マネージャー・データをファイル・システムに保管します。 IBM WebSphere MQ で使用するディレクトリー構造の計画と構成については、以下のリンクを参照してください。

- 131 ページの『ファイル・システム・サポートの計画』
- 132 ページの『ファイル共有システムの要件』
- 142 ページの『IBM WebSphere MQ ファイルの共有』
- 145 ページの『UNIX and Linux システムでのディレクトリー構造』
- 154 ページの『Windows システムでのディレクトリー構造』

UNIX and Linux でのシステム・リソース、共用メモリー、およびプロセス優先順位については、以下のリンクを使用してください。

- 158 ページの『IBM WebSphere MQ と UNIX System V IPC リソース』
- 158 ページの『AIX 上の共有メモリー』
- 158 ページの『WebSphere MQ および UNIX のプロセス優先順位』

関連概念

5 ページの『計画』
IBM WebSphere MQ 環境の計画時には、構成する IBM WebSphere MQ アーキテクチャー、リソース要件、およびロギングとバックアップ機能の必要性について考慮する必要があります。 このトピックにあるリンクを使用して、IBM WebSphere MQ が実行される環境を計画します。

14 ページの『IBM WebSphere MQ アーキテクチャーの設計』
Point-to-Point メッセージング・スタイルおよびパブリッシュ/サブスクライブ・メッセージング・スタイル用に IBM WebSphere MQ がサポートするさまざまなアーキテクチャーについて説明します。

UNIX および Linux でのハードウェア要件とソフトウェア要件
Windows でのハードウェア要件とソフトウェア要件

ディスク・スペースの要件

WebSphere MQ のストレージ要件は、インストールするコンポーネント、および必要なワークスペース量によって異なります。

ディスク・ストレージはインストールするオプション・コンポーネント用に必要となりますが、それにはオプション・コンポーネントが必要とする前提条件のコンポーネントも含まれます。 使用するキューの数、キューに入れるメッセージの数とサイズ、およびメッセージが持続メッセージかどうかによって、ストレージ要件の合計は異なります。 そのほかに、ディスクやテープなどのメディアにアーカイブとして保存するための容量も必要であり、所有アプリケーション・プログラムのためのスペースももちろん必要です。

以下の表に、さまざまなプラットフォームにいろいろな製品を組み合わせてインストールしたときに必要となる、おおよそのディスク・スペースを示します。 (値は 5 MB 単位になるように切り上げています。1 MB は 1,048,576 バイトです。)

プラットフォーム	クライアント・インストール [1]	サーバー・インストール [2]	WebSphere MQ MFT インストール [3]	フルインストール [4]
AIX	145 MB	190 MB	705 MB	915 MB
HP-UX	225 MB	310 MB	1075 MB	1340 MB
IBM i	215 MB	450 MB	80 MB	655 MB
Linux for System x (32 ビット)	85 MB	N/A	N/A	120 MB

プラットフォーム	クライアント・インストール [1]	サーバー・インストール [2]	WebSphere MQ MFT インストール [3]	フルインストール [4]
Linux for System x (64 ビット)	125 MB	170 MB	575 MB	935 MB
Linux on POWER Systems - Big Endian	130 MB	170 MB	565 MB	715 MB
Solaris x86-64、AMD64、EM64T、および互換プロセッサー	105 MB [5]	150 MB [5]	695 MB	860 MB
Solaris SPARC	105 MB [5]	150 MB [5]	680 MB	820 MB
Windows (32 ビット・インストール) [6]	390 MB	N/A	N/A	475 MB
Windows (64 ビット・インストール) [6]	445 MB	555 MB	710 MB	1005 MB

使用上の注意

1. クライアント・インストールには、以下のコンポーネントが含まれます。
   - のランタイム
   - クライアント
2. サーバー・インストールには、以下のコンポーネントが含まれます。
   - のランタイム
   - サーバー
3. IBM WebSphere MQ Managed File Transfer のインストールには、以下のコンポーネントが含まれます。
   - IBM WebSphere MQ Managed File Transfer Service、Logger、Agent、Tools、および Base コンポーネント
   - のランタイム
   - サーバー
   - Java
   - JRE
4. フルインストールには、選択可能なすべてのコンポーネントが含まれます。
5. Solaris Solaris プラットフォームでは、この組み合わせのコンポーネントをインストールするには、サイレント・インストールを実行する必要があります。
6. Windows ここにリストされているすべてのコンポーネントが、Windows システムにインストール可能なフィーチャーであるわけではありません。 それらの機能が別のフィーチャーに組み込まれている場合があります。 Windows システムの WebSphere MQ 機能を参照してください。

関連タスク

インストール内容の選択

ファイル・システム・サポートの計画

キュー・マネージャー・データはファイル・システムに格納されます。 キュー・マネージャーはファイル・システム・ロックを使用して、複数インスタンスキュー・マネージャーの複数インスタンスが同時にアクティブにならないようにします。

ファイル共有システム

ファイル共有システムでは、複数のシステムが同じ物理ストレージ・デバイスに同時にアクセスできるようにします。 複数のシステムが、ロックと並行性制御を適用する方法をとらずに、同じ物理ストレージ・デバイスに直接アクセスした場合に破損が起こることがあります。 オペレーティング・システムには、ローカル・プロセスのロックと並行性制御が用意されたローカル・ファイル・システムがあります。 ネットワーク・ファイル・システムには、分散システム用にロックと並行性制御が用意されています。

これまでのネットワーク・ファイル・システムはそれほど高速で稼働していませんでした。つまり、メッセージのログ記録の要件を満たす十分なロックと並行性制御を提供していませんでした。 現在のネットワーク・ファイル・システムでは、良好なパフォーマンスを提供し、 *RFC 3530* のネットワーク・ファイル・システム *(NFS)* バージョン *4* プロトコルなどの 信頼性の高いネットワーク・ファイル・システム・プロトコルの実装を可能にして、 確実にメッセージのログ記録の要件を満たすことができます。

ファイル共有システムおよび WebSphere MQ

複数インスタンスのキュー・マネージャーのキュー・マネージャー・データは、共有ネットワーク・ファイル・システムに保管されます。 Microsoft Windows および UNIX and Linux システムでは、キュー・マネージャーのデータ・ファイルおよびログ・ファイルは、共有ネットワーク・ファイル・システムに置く必要があります。

リリース v7.0.1 より前の WebSphere MQ は、ファイル共有システムとしてアクセスされるネットワーク・ストレージに保管されたキュー・マネージャー・データをサポートしていません。 キュー・マネージャー・データが共有ネットワーク・ストレージにある場合、そのキュー・マネージャー・データが、同時に実行しているキュー・マネージャーの別のインスタンスによってアクセスされていないことを確認する必要があります。

v7.0.1 以降の WebSphere MQ はロックを使用して、同じ複数インスタンス・キュー・マネージャーの複数インスタンスが同時にアクティブにならないようにします。 またこの同じロックは、2 つの異なるキュー・マネージャーが、同じセットのキュー・マネージャー・データ・ファイルを不注意に使用できないようにします。 ロックを使用できるキュー・マネージャーのインスタンスは一度に 1 つだけです。 そうすることで、WebSphere MQ は、ファイル共有システムとしてアクセスされるネットワーク・ストレージに保管されたキュー・マネージャー・データをサポートします。

ネットワーク・ファイル・システムのロック・プロトコルがすべて堅固であるわけではなく、またファイル・システムはデータ保全性というよりもパフォーマンスを優先して構成される場合があるため、**amqmfsck** コマンドを実行して、ネットワーク・ファイル・システムがキュー・マネージャー・データおよびログへのアクセスを正しく制御するかどうかをテストする必要があります。 このコマンドは、UNIX および IBM i システムにのみ適用されます。 Microsoft Windows では、サポートされるネットワーク・ファイル・システムは 1 つしかないため、**amqmfsck** コマンドは必要ありません。

関連タスク

134 ページの『共有ファイル・システムの動作の検証』

amqmfsck を実行して、UNIX システム上の共有ファイル・システムが、複数インスタンス・キュー・マネージャーのキュー・マネージャー・データを保管するための要件を満たすかどうか確認します。 IBM WebSphere MQ MQI client のサンプル・プログラム **amqsfhac** を **amqmfsck** と並行して実行し、障害時にキュー・マネージャーがメッセージの整合性を保持することを実証します。

ファイル共有システムの要件

ファイル共有システムは、IBM WebSphere MQ での作業の信頼性を高めるために、データ書き込み整合性、ファイルに対する排他的アクセスの保証、および障害時のロック解除を実現する必要があります。

ファイル共有システムが満たす必要がある要件

ファイル共有システムがメッセージを確実に記録するために満たす必要がある 3 つの基本要件があります。

1. データ書き込み整合性

データ書き込み整合性は、「フラッシュ時のディスクへの書き込み」と呼ばれることがあります。 キュー・マネージャーは、 物理デバイスに正常にコミットされているデータと同期できなければなりません。 トランザクション・システムでは、他の処理を続行する前に、いくつかの書き込みが安全にコミットされたことを確認する必要があります。

具体的には、 UNIX プラットフォーム上の IBM WebSphere MQ は、 *O_SYNC* オープン・オプションと `fsync()` システム・コールを使用して、リカバリー可能メディアへの書き込みを明示的に強制し、これらのオプションが正しく動作するかどうかに依存します。

重要 : Linux async オプションを使用して、ファイル・システムをマウントする必要があります。 このオプションを使用すると、引き続き同期書き込みのオプションがサポートされ、sync オプションよりもパフォーマンスが向上します。

しかし、ファイル・システムが Linux からエクスポートされた場合は、引き続き sync オプションを使用してファイル・システムをエクスポートしなければならないことに注意してください。

2. ファイルに対する排他的アクセスの保証

   複数のキュー・マネージャーを同期するために、 キュー・マネージャーにはファイルへの排他ロックを獲得する手段が必要です。

3. 障害時のロック解除

   キュー・マネージャーに障害が発生したり、ファイル・システムとの通信障害がある場合は、キュー・マネージャーがファイル・システムに再接続するのを待たずに、キュー・マネージャーによってロックされたファイルをアンロックして、他のプロセスで使用できるようにしなければなりません。

ファイル共有システムでは、IBM WebSphere MQ を確実に作動させるためにこれらの要件を満たす必要があります。 そうしないと、ファイル共有システムを複数インスタンス・キュー・マネージャー構成で使用するときに、キュー・マネージャー・データおよびログが破損します。

Microsoft Windows の複数インスタンスのキュー・マネージャーの場合、Microsoft Windows ネットワークによって使用される Common Internet File System (CIFS) プロトコルでネットワーク・ストレージにアクセスする必要があります。 Common Internet File System (CIFS) クライアントは、 Microsoft Windows 以外のプラットフォームでセマンティクスをロックするための IBM WebSphere MQ の要件を満たしていません。そのため、 Microsoft Windows 以外のプラットフォームで実行される複数インスタンス・キュー・マネージャーは、共有ファイル・システムとして Common Internet File System (CIFS) を使用してはなりません。

その他のサポートされているプラットフォームの複数インスタンスのキュー・マネージャーでは、Posix に準拠し、リース・ベースのロックをサポートしているネットワーク・ファイル・システム・プロトコルによってストレージにアクセスする必要があります。 最新のファイル・システム (ネットワーク・ファイル・システム (NFS) バージョン 4 など) は、専用ロックを使用して障害を検出し、障害の後にロックを解除します。 以前のファイル・システム (ネットワーク・ファイル・システム (NFS) バージョン 3 など) は、障害後にロックを解除するための信頼性のある仕組みを持たないため、複数インスタンス・キュー・マネージャーで使用してはなりません。

ファイル共有システムが要件を満たすかどうかの確認

使用する予定のファイル共有システムが、これらの要件を満たすかどうかを確認する必要があります。 また、ファイル・システムが信頼性を得られるように 正しく構成されているかどうかも確認する必要があります。 ファイル共有システムは、 信頼性を犠牲にしてパフォーマンスを向上させるような構成オプションを提供することがあります。

通常の環境では、IBM WebSphere MQ は属性キャッシュを使用して正しく作動するため、例えば NFS マウントで NOAC を設定するなどしてキャッシュを使用不可にする必要はありません。 複数のファイル・システム・クライアントがファイル・システム・サーバー上にある同じファイルへの書き込みアクセスを得ようと競合する場合、 属性キャッシュは問題を引き起こすことがあります。 これは、各クライアントが使用するキャッシュ内属性がサーバー上の属性と同じではない場合があるためです。 このようにアクセスされるファイルの例は、マルチインスタンス・キュー・マネージャー用のキュー・マネージャー・エラー・ログです。 キュー・マネージャー・エラー・ログはアクティブ・キュー・マネージャー・インスタンスとスタンバイ・キュー・マネージャー・インスタンスの 両方によって書き込まれることがあり、 キャッシュに

入っているファイル属性が存在すると、ファイルのロールオーバーが発生する前にエラー・ログが予想以上に大きくなることもあります。

ファイル・システムを確認するには、 134 ページの『共有ファイル・システムの動作の検証』タスクを実行します。 このタスクは、共有ファイル・システムが要件 2 と 3 を満たすかどうかを検査します。 要件 1 については、 ファイル共有システムの資料で、またはディスクへのデータのログ記録を検証することによって、確認する必要があります。

ディスク障害はディスクへの書き込み時にエラーを引き起こすことがあり、これは IBM WebSphere MQ では初期障害データ・キャプチャー機能エラーとして報告されます。 ファイル共有システムにディスク障害がないかどうかを検査するには、 オペレーティング・システム用のファイル・システム検査プログラムを実行します。 例えば、UNIX および Linux プラットフォームでは、ファイル・システム・チェッカーは fsck と呼ばれます。 Windows プラットフォームのファイル・システム検査プログラムは CHKDSK または SCANDISK と呼ばれます。

NFS サーバー・セキュリティー

注 : ネットワーク・ファイル・システム (NFS) サーバーにはキュー・マネージャー・データのみを書き込む必要があります。 NFS では、mount コマンドで次の 3 つのオプションを使用して、システムを保護します。

noexec
このオプションを使用すると、バイナリー・ファイルを NFS 上で実行できなくなります。 こうすることで、リモート・ユーザーがシステム上で望ましくないコードを実行できないようにします。

nosuid
このオプションを使用すると、セット・ユーザー ID ビットとセット・グループ ID ビットを使用できなくなります。 こうすることで、リモート・ユーザーが上位の特権を取得できないようにします。

nodev
このオプションを使用すると、文字およびブロック特殊装置が使用または定義できなくなります。 こうすることで、リモート・ユーザーが chroot ジェイルから出られないようにします。

共有ファイル・システムの動作の検証

amqmfsck を実行して、UNIX システム上の共有ファイル・システムが、複数インスタンス・キュー・マネージャーのキュー・マネージャー・データを保管するための要件を満たすかどうか確認します。 IBM WebSphere MQ MQI client のサンプル・プログラム **amqsfhac** を **amqmfsck** と並行して実行し、障害時にキュー・マネージャーがメッセージの整合性を保持することを実証します。

始める前に

ネットワーク・ストレージを備えた 1 台のサーバーと、それに接続しており、WebSphere MQ がインストールされているさらに 2 台のサーバーが必要です。 ファイル・システムを構成するには管理者 (root) 権限を持っていることが必要であり、**amqmfsck** を実行するには WebSphere MQ 管理者であることが必要です。

このタスクについて

132 ページの『ファイル共有システムの要件』では、複数インスタンス・キュー・マネージャーで共有ファイル・システムを使用するための、ファイル・システム要件について説明します。 IBM WebSphere MQ の技術情報、「Testing and support statement for WebSphere MQ multi-instance queue managers」に、IBM でテスト済みの共有ファイル・システムがリストされています。 この作業の手順は、リストに含まれていないファイル・システムでデータの整合性が保持されるかどうかを評価するために、ファイル・システムをテストする方法について説明しています。

複数インスタンス・キュー・マネージャーのフェイルオーバーは、ハードウェアまたはソフトウェアの障害によりトリガーできます。そうした障害には、キュー・マネージャーからデータやログ・ファイルに書き込みができなくなるネットワークの問題も含まれます。 ファイル・サーバーに障害を発生させることがテストの主眼となります。 しかし、ロックが正常に解放されることをテストするため、IBM WebSphere MQ サーバーにも障害を発生させる必要があります。 共有ファイル・システムの信頼性を確認するため、以下の障害、および環境に固有のその他の障害についてすべてテストしてください。

1. ディスクを同期させて、ファイル・サーバーのオペレーティング・システムをシャットダウンする。
2. ディスクを同期させないで、ファイル・サーバーのオペレーティング・システムを停止する。
3. 各サーバーのリセット・ボタンを押す。
4. 各サーバーのネットワーク・ケーブルを抜く。
5. 各サーバーの電源ケーブルを抜く。
6. 各サーバーの電源をオフにする。

キュー・マネージャーのデータおよびログを共有するために使用するディレクトリーを、ネットワーク・ストレージ上に作成します。 ディレクトリー所有者は WebSphere MQ 管理者であることが必要です。言い換えれば、UNIX の mqm グループのメンバーであることが必要です。 テストを実行するユーザーは、WebSphere MQ 管理者権限を持っている必要があります。

Linux での複数インスタンス・キュー・マネージャーの作成にあるファイル・システムのエクスポートおよびマウントの例を使用すると、ファイル・システムを構成するうえで役立ちます。 それぞれのファイル・システムでは異なる構成ステップが必要です。 ファイル・システムの資料を参照してください。

手順

各検査では、ファイル・システム検査プログラムの実行中に、前のリストの障害をすべて発生させてください。 **amqmfsck** と同時に **amqsfhac** を実行する場合は、このタスクと並行して、140 ページの『amqsfhac を実行してメッセージの整合性を検査する』のタスクを実行してください。

1. エクスポートされたディレクトリーを 2 つの IBM WebSphere MQ サーバーにマウントします。

   ファイル・システム・サーバー上に共有ディレクトリー shared と、複数インスタンス・キュー・マネージャーのデータを保存するためのサブディレクトリー qmdata を作成します。 Linux での複数インスタンス・キュー・マネージャーの共有ディレクトリーのセットアップ例については、 Linux での複数インスタンス・キュー・マネージャーの作成 の 例 を参照してください。
2. ファイル・システムの基本的な動作を検査します。

   1 台の IBM WebSphere MQ サーバー上で、パラメーターを付けずにファイル・システム検査プログラムを実行します。

   ```
   amqmfsck /shared/qmdata
   ```

   図 *40. IBM WebSphere MQ* サーバー *1* 上
3. 両方の IBM WebSphere MQ サーバーから同一のディレクトリーに同時に書き込む検査をします。

   両方の IBM WebSphere MQ サーバー上で、-c オプションを指定して、ファイル・システム検査プログラムを同時に実行します。

   ```
   amqmfsck -c /shared/qmdata
   ```

   図 *41. IBM WebSphere MQ* サーバー *1* 上

   ```
   amqmfsck -c /shared/qmdata
   ```

   図 *42. IBM WebSphere MQ* サーバー *2* 上
4. 両方の IBM WebSphere MQ サーバー上で、ロックの待機と解放を検査します。

   両方の IBM WebSphere MQ サーバー上で、-w オプションを指定して、ファイル・システム検査プログラムを同時に実行します。

```
amqmfsck -w /shared/qmdata
```

図 *43. IBM WebSphere MQ* サーバー *1* 上

```
amqmfsck -w /shared/qmdata
```

図 *44. IBM WebSphere MQ* サーバー *2* 上

5. データの整合性を検査します。

   a) テスト・ファイルをフォーマット設定します。

   検査対象のディレクトリー内に大きなファイルを作成します。 このファイルは、後続のフェーズを正常に完了できるよう、フォーマット設定されます。 フェイルオーバーをシミュレートするために第 2 のフェーズを中断するための時間が十分とれるよう、このファイルは十分な大きさであることが必要です。 デフォルト値の 262144 ページ (1 GB) を試してください。 低速のファイル・システムでは、フォーマット設定が約 60 秒以内で完了するよう、このデフォルト値がプログラムによって自動的に削減されます。

   ```
   amqmfsck -f /shared/qmdata
   ```

   サーバーは、次のメッセージで応答します。

   ```
   Formatting test file for data integrity test.
   Test file formatted with 262144 pages of data.
   ```

   図 *45. IBM WebSphere MQ* サーバー *1* 上

   b) 障害を発生させながら、ファイル・システム検査プログラムを使用してテスト・ファイルにデータを書き込みます。

   2 台のサーバー上で、検査プログラムを同時に実行します。 障害を発生させるサーバー上で検査プログラムを開始してから、障害を回避させるサーバー上で検査プログラムを開始します。 調査対象の障害を発生させます。

   最初の検査プログラムが、エラー・メッセージを表示して停止します。 2 番目の検査プログラムがテスト・ファイルに対するロックを取得し、最初の検査プログラムが書き込みを終了した箇所から、テスト・ファイルにデータを書き込みます。 2 番目の検査プログラムの完了を待ちます。

表 *12. 2* 台のサーバー上でデータの整合性検査を同時に実行する

IBM WebSphere MQ サーバー 1	**IBM WebSphere MQ サーバー 2**
`amqmfsck -a /shared/qmdata`	

表 *12. 2* 台のサーバー上でデータの整合性検査を同時に実行する *(*続き*)*

IBM WebSphere MQ サーバー 1	**IBM WebSphere MQ サーバー 2**
	`amqmfsck -a /shared/qmdata`

`Please start this program on a second machine`
`with the same parameters.`
`File lock acquired.`
`Start a second copy of this program with the same parameters on another server.`
 	
`Writing data into test file.`
 	
`To increase the effectiveness of the test,`
`interrupt the writing by ending the process,`
`temporarily breaking the network connection`
`to the networked storage,`
`rebooting the server of turning off the power.`
`To increase the effectiveness of the test,`
`interrupt the writing by ending the process,`
`temporarily breaking the network connection`
`to the networked storage,`
`rebooting the server of turning off the power.`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`

Turn the power off here	

	`File lock acquired.`
	`Reading test file`
	`Checking the integrity of the data read.`
	`Appending data into the test file`
	`after data already found.`
	`The test file is full of data.`
	`It is ready to be inspected for data integrity.`

検査のタイミングは、ファイル・システムの動作によって異なります。 例えば、電源異常の後でファイル・システムが、最初のプログラムによって取得されていたファイル・ロックを解放するまでには、通常 30 から 90 秒の時間が必要です。 時間が短すぎて最初のテスト・プログラムがファイルへの書き込みを終了する前に障害が起きなかった場合は、**amqmfsck** の -x オプションを使用して、テスト・ファイルを削除してください。 さらに大きなテスト・ファイルを使用して、テストを最初から実行してください。

c) テスト・ファイル内のデータの整合性を検査します。

```
amqmfsck -i /shared/qmdata
```

サーバーは、次のメッセージで応答します。

```
File lock acquired
Reading test file checking the integrity of the data read.
The data read was consistent.
The tests on the directory completed successfully.
```

図 *46. IBM WebSphere MQ* サーバー *2* 上

6. テスト・ファイルを削除します。

```
amqmfsck -x /shared/qmdata
Test files deleted.
```

図 *47. IBM WebSphere MQ* サーバー *2* 上

サーバーは、次のメッセージで応答します。

```
Test files deleted.
```

タスクの結果

テストが正常に完了すると、プログラムは終了コードとしてゼロを戻します。そうでない場合はゼロ以外を戻します。

例

3 つの例の最初のものでは、最小出力を生成するコマンドを示します。

1 台のサーバー上の基本ファイル・ロックのテストが正常終了

```
> amqmfsck /shared/qmdata
The tests on the directory completed successfully.
```

1 台のサーバー上の基本ファイル・ロックのテストが失敗

```
> amqmfsck /shared/qmdata
AMQ6245: Error Calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck' error '2'.
```

2 台のサーバー上のロックのテストが正常終了

表 *13. 2* 台のサーバー上のロックが正常終了

IBM WebSphere MQ サーバー 1	**IBM WebSphere MQ サーバー 2**

`> amqmfsck -w /shared/qmdata`
`Please start this program on a second`
`machine with the same parameters.`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -w /shared/qmdata` `Waiting for lock...`
`[ Return pressed ]` `Lock released.`	

	`Lock acquired.`
	`The tests on the directory completed`
	`successfully`

3 つの例の 2 番目のものでは、冗長モードを使用する同じコマンドを示します。

1 台のサーバー上の基本ファイル・ロックのテストが正常終了

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")'
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
```

```
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd1 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd1, F_SETLK, F_RDLCK)
System call: fd2 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd2, F_SETLK, F_RDLCK)
System call: close(fd2)
System call: write(fd1)
System call: close(fd1)
The tests on the directory completed successfully.
```

1 台のサーバー上の基本ファイル・ロックのテストが失敗

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_RDLCK)
System call: fdSameFile = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fdSameFile, F_SETLK, F_RDLCK)
System call: close(fdSameFile)
System call: write(fd)
AMQxxxx: Error calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck', errno 2
(Permission denied).
```

2 台のサーバー上のロックのテストが正常終了

表 *14. 2* 台のサーバー上のロックが正常終了 - 冗長モード

IBM WebSphere MQ サーバー 1	**IBM WebSphere MQ サーバー 2**

`> amqmfsck -wv /shared/qmdata`
`Calling 'stat("/shared/qmdata")'`
`Calling 'fd = open("/shared/qmdata/`
`amqmfsck.lkw",`
`O_EXCL | O_CREAT | O_RDWR, 0666)'`
`Calling 'fchmod(fd, 0666)'`
`Calling 'fstat(fd)'`
`Please start this program on a second`
`machine with the same parameters.`
`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -wv /shared/qmdata`
	`Calling 'stat("/shared/qmdata")'`
	`Calling 'fd = open("/shared/qmdata/`
	`amqmfsck.lkw",`
	`O_EXCL | O_CREAT | O_RDWR,0666)'`
	`Calling 'fd = open("/shared/qmdata/amqmfsck.lkw,`
	`O_RDWR, 0666)'`
	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)`
	`'Waiting for lock...`

`[ Return pressed ]`
`Calling 'close(fd)'`
`Lock released.`	

表 14. 2 台のサーバー上のロックが正常終了 - 冗長モード (続き)	
IBM WebSphere MQ サーバー 1	**IBM WebSphere MQ サーバー 2**

	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
	`Lock acquired.`
	`The tests on the directory completed successfully`

関連資料
amqmfsck (ファイル・システム検査)

amqsfhac を実行してメッセージの整合性を検査する

amqsfhac では、ネットワーク・ストレージを使用するキュー・マネージャーが、障害後もデータ保全性を維持していることを検査します。

始める前に

この検査には、4 台のサーバーが必要です。 2 台は複数インスタンス・キュー・マネージャー用で、1 台はファイル・システム用、もう 1 台は、IBM WebSphere MQ MQI client・アプリケーションとしての **amqsfhac** の実行用です。

手順のステップ 135 ページの『1』 に従って、複数インスタンス・キュー・マネージャー用にファイル・システムをセットアップします。

このタスクについて

手順

1. 手順のステップ 135 ページの『1』 で作成したファイル・システムを使用して、別のサーバー、QM1 上に複数インスタンス・キュー・マネージャーを作成します。

   複数インスタンス・キュー・マネージャーの作成を参照してください。
2. 両方のサーバー上でキュー・マネージャーを開始し、可用性を高くします。

   サーバー 1:

   ```
   strmqm -x QM1
   ```

   サーバー 2:

   ```
   strmqm -x QM1
   ```
3. **amqsfhac** を実行するためにクライアント接続をセットアップします。

   a)「 クライアント・インストールの検査 」の手順を使用して、クライアント接続をセットアップするか、 再接続可能クライアントのサンプル にあるサンプル・スクリプトを使用します。

   b) QM1 を実行している 2 台のサーバーに対応する 2 つの IP アドレスを使用するように、クライアント・チャネルを変更します。

   サンプル・スクリプトで、以下を変更します。

   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) +
   CONNAME('LOCALHOST(2345)') QMNAME(QM1) REPLACE
   ```

   終了:

   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) +
   CONNAME('server1(2345),server2(2345)') QMNAME(QM1) REPLACE
   ```

ここで、server1 および server2 は 2 台のサーバーのホスト名、2345 はチャネル・リスナーが listen しているポートです。 通常、このデフォルトは 1414 です。 デフォルトのリスナー構成では、1414 を使用できます。

4. テスト用として、QM1 に 2 つのローカル・キューを作成します。
   以下の MQSC スクリプトを実行します。

```
DEFINE QLOCAL(TARGETQ) REPLACE
DEFINE QLOCAL(SIDEQ) REPLACE
```

5. **amqsfhac** を使用して構成を検査します。

```
amqsfhac QM1 TARGETQ SIDEQ 2 2 2
```

6. ファイル・システムの整合性を検査するのと同時に、メッセージの整合性を検査します。

   手順のステップ 136 ページの『5』 の中で、**amqsfhac** を実行します。

```
amqsfhac QM1 TARGETQ SIDEQ 10 20 0
```

   アクティブなキュー・マネージャーのインスタンスを停止すると、**amqsfhac** は、もう 1 つのキュー・マネージャーのインスタンスがアクティブになったときにそのインスタンスに再接続します。 次の検査で障害をリバースできるようにするため、停止したキュー・マネージャーのインスタンスを再始動します。 フェイルオーバーが発生するのに十分な時間、検査プログラムが実行を続けるようにするため、ご使用の環境での試行に基づいて、反復の数を増やす必要があるかもしれません。

タスクの結果

ステップ 141 ページの『6』 での **amqsfhac** の実行例を、142 ページの図 48 に示します。 検査は成功しました。

検査で問題が検出されると、出力で障害が報告されます。 検査を実行したときに、MQRC_CALL_INTERRUPTED によって "Resolving to backed out" が報告される場合があります。 これが結果に影響することはありません。 これが報告されるかどうかは、ディスクへの書き込みがネットワーク・ファイル・ストレージによって、障害発生の前にコミットされたか、後にコミットされたかに応じて異なります。

```
Sample AMQSFHAC start
qmname = QM1
qname = TARGETQ
sidename = SIDEQ
transize = 10
iterations = 20
verbose = 0
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Resolving MQRC_CALL_INTERRUPTED
MQGET browse side tranid=14 pSideinfo->tranid=14
Resolving to committed
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Sample AMQSFHAC end
```

図 *48.* ***amqsfhac*** の実行が正常終了した場合の出力

関連資料

高可用性のサンプル・プログラム

IBM WebSphere MQ ファイルの共有

IBM WebSphere MQ ファイルの中には、アクティブ・キュー・マネージャーだけがアクセスするファイルもあれば、共有されるファイルもあります。

WebSphere MQ ファイルは、プログラム・ファイルとデータ・ファイルに分かれています。 プログラム・ファイルは通常、WebSphere MQ を実行している各サーバー上にローカルにインストールされます。 キュー・マネージャーは、デフォルト・データ・ディレクトリー内のデータ・ファイルおよびディレクトリーへのアクセスを共有します。 キュー・マネージャーは、143 ページの図 49 に示す `qmgrs` と `log` の各ディレクトリーに含まれる独自のキュー・マネージャー・ディレクトリー・ツリーに対しては排他的アクセスを要求します。

143 ページの図 49 は、 WebSphere MQ ディレクトリー構造の概要図です。 これは、キュー・マネージャー間で共有し、リモートにすることができるディレクトリーを示しています。 詳細はプラットフォームごとに異なります。 点線は構成可能なパスを表します。

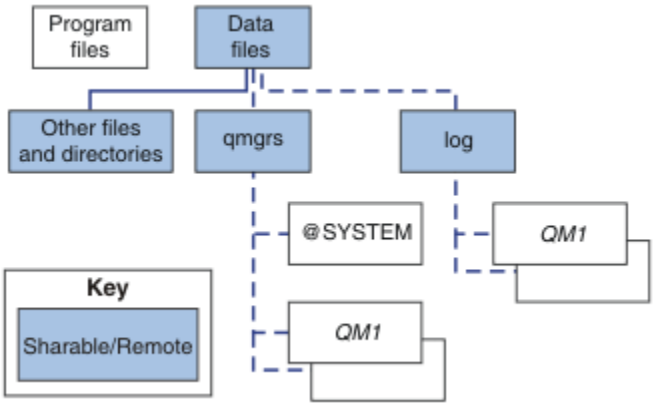


図 *49. WebSphere MQ* ディレクトリー構造の全体図

プログラム・ファイル

プログラム・ファイル・ディレクトリーは通常、デフォルトの位置に残され、ローカルであり、サーバー上のすべてのキュー・マネージャーによって共有されます。

データ・ファイル

データ・ファイル・ディレクトリーは通常、UNIX and Linux システムではデフォルトの位置 `/var/mqm` でローカルであり、Windows ではインストール済み環境上で構成可能です。 これはキュー・マネージャー間で共有されます。 デフォルトの位置をリモートにすることはできますが、WebSphere MQ の異なるインストール済み環境間では共有されません。 WebSphere MQ 構成内の `DefaultPrefix` 属性はこのパスを指します。

qmgrs

v7.0.1 から、キュー・マネージャー・データの場所を指定する代替方法が 2 つあります。

Prefix の使用

`Prefix` 属性は、`qmgrs` ディレクトリーの場所を指定します。 WebSphere MQ は、キュー・マネージャー名からキュー・マネージャー・ディレクトリー名を構成し、それを `qmgrs` ディレクトリーのサブディレクトリーとして作成します。

`Prefix` 属性は QueueManager スタンザに置かれ、`DefaultPrefix` 属性の値から継承されます。 デフォルトでは管理を簡素化するために、キュー・マネージャーは通常、同じ `qmgrs` ディレクトリーを共有します。

`QueueManager` スタンザは `mqs.ini` ファイル内にあります。

キュー・マネージャーの `qmgrs` ディレクトリーの場所を変更する場合、 その `Prefix` 属性の値を変更する必要があります。

UNIX and Linux プラットフォームの場合、143 ページの図 49 での `QM1` ディレクトリーの `Prefix` 属性は次のとおりです。

```
Prefix=/var/mqm
```

DataPath の使用

`DataPath` 属性は、キュー・マネージャー・データ・ディレクトリーの場所を指定します。

`DataPath` 属性は、キュー・マネージャー・データ・ディレクトリーの名前を含めて完全なパスを指定します。 `DataPath` 属性は、キュー・マネージャー・データ・ディレクトリーへの不完全なパスを指定する `Prefix` 属性とは異なります。

`DataPath` 属性が指定される場合、それは QueueManager スタンザに置かれます。 この属性が指定されている場合、`Prefix` 属性のどの値よりも優先されます。

`QueueManager` スタンザは `mqs.ini` ファイル内にあります。

キュー・マネージャーのキュー・マネージャー・データ・ディレクトリーの場所を変更する場合、DataPath 属性の値を変更する必要があります。

UNIX または Linux プラットフォームの場合、143 ページの図 49 での QM1 ディレクトリーの DataPath 属性は次のとおりです。

```
DataPath=/var/mqm/qmgrs/QM1
```

log

ログ・ディレクトリーは、キュー・マネージャー構成の Log スタンザ内のキュー・マネージャーごとに別個に指定されます。 キュー・マネージャー構成は、qm.ini にあります。

DataPath/*QmgrName*/@IPCC サブディレクトリー

DataPath/*QmgrName*/@IPCC サブディレクトリーは共有ディレクトリー・パスにあります。 これらは IPC ファイル・システム・オブジェクト用のディレクトリー・パスを構成するために使用されます。 複数のシステム間でキュー・マネージャーを共有する場合、キュー・マネージャーの名前空間を区別する必要があります。 V7.0.1 より前では、キュー・マネージャーは 1 つのシステムでのみ使用されていました。 IPC ファイル・システム・オブジェクトへのディレクトリー・パスを定義するにはサブディレクトリーの 1 つのセットで十分でした (144 ページの図 50 を参照)。

```
DataPath/QmgrName/@IPCC/esem
```

図 *50. V7.0.1* より前の *IPC* サブディレクトリーの例

V7.0.1 以降では、 IPC ファイル・システム・オブジェクトをシステムで区別する必要があります。 キュー・マネージャーが実行されるシステムごとに、1 つのサブディレクトリーが ディレクトリー・パスに追加されます (144 ページの図 51 を参照)。

```
DataPath/QmgrName/@IPCC/esem/myHostName/
```

図 *51. V7.0.1* 以降のリリースでの *IPC* サブディレクトリーの例

myHostName は、オペレーティング・システムから返されたホスト名 (最大で 最初の 20 文字まで) です。 システムによっては、切り捨て前のホスト名は最大 64 文字の長さの場合があります。 生成された *myHostName* の値は、以下の 2 つの理由で問題を引き起こすことがあります。

1. 最初の 20 文字が固有のものでない。
2. ホスト名が、同じホスト名をシステムに割り振るとは限らない DHCP アルゴリズムによって生成された。

これらの場合、環境変数 MQC_IPC_HOST を使用して *myHostName* を設定します (144 ページの図 52 を参照)。

```
export MQS_IPC_HOST=myHostName
```

図 *52.* 例*: MQC_IPC_HOST* の設定

その他のファイル およびディレクトリー

その他のファイルおよびディレクトリー (トレース・ファイルを含むディレクトリーや共通エラー・ログなど) は通常、ローカル・ファイル・システムで共有され、保持されます。

v7.0.1 より前までは、WebSphere MQ は、キュー・マネージャーによるキュー・マネージャー・データおよびログ・ファイルへの排他的アクセスを保証するために、外部管理に依存していました。 v7.0.1 からは、ファイル共有システムのサポートによって、WebSphere MQ は、ファイル・システム・ロックを使用してこれらのファイルへの排他的アクセスを管理します。 ファイル・システム・ロックでは、 特定のキュー・マネージャーのインスタンスのうち、一度に 1 つのみをアクティブにできます。

特定のキュー・マネージャーの最初のインスタンスを開始すると、 そのインスタンスは自身のキュー・マネージャー・ディレクトリーの所有権を獲得します。 2 番目のインスタンスを開始する場合、そのインスタンスは最初のインスタンスが停止している場合にのみ所有権を取得できます。 最初のキュー・マネージ

ャーがまだ実行中の場合、2 番目のインスタンスは開始に失敗し、キュー・マネージャーが他の場所で実行されていることが報告されます。 最初のキュー・マネージャーがすでに停止していれば、2 番目のキュー・マネージャーが キュー・マネージャーのファイルの所有権を引き継ぎ、実行中のキュー・マネージャーになります。

2 番目のキュー・マネージャーが最初のキュー・マネージャーから引き継ぐ手順は、自動化できます。 別のキュー・マネージャーが引き継ぐことを許可する `strmqm -x` オプションを指定して、最初のキュー・マネージャーを開始します。 その場合、2 番目のキュー・マネージャーは、 最初のキュー・マネージャーのファイルがアンロックされるまで待機してから、キュー・マネージャーのファイルの所有権を引き継いで始動を試みます。

UNIX and Linux システムでのディレクトリー構造

UNIX and Linux システム上の WebSphere MQ ディレクトリー構造は、管理を容易にし、パフォーマンスを向上させ、信頼性を向上させるために、異なるファイル・システムにマップすることができます。

WebSphere MQ の柔軟なディレクトリー構造を使用して、複数インスタンス・キュー・マネージャーの実行にファイル共有システムを活用します。

コマンド **crtmqm** *QM1* を使用して、145 ページの図 53 に示すディレクトリー構造を作成します。ここで、R は製品のリリースです。 これは、WebSphere MQ システムの v7.0.1 以降でキュー・マネージャーに作成される標準的なディレクトリー構造です。 一部のディレクトリー、ファイル、および .ini 属性設定は、分かりやすくするために省略しています。また、キュー・マネージャー名はマングリングによって変更される場合があります。 ファイル・システムの名前は、各種のシステムで異なります。

標準的インストールでは、作成するそれぞれのキュー・マネージャーは、 ローカル・ファイル・システム上の共通の `log` および `qmgrs` のディレクトリーを指します。 複数インスタンス構成では、 `log` および `qmgrs` のディレクトリーは、WebSphere MQ の他のインストール済み環境と共有されるネットワーク・ファイル・システム上にあります。

145 ページの図 53 は、WebSphere MQ v7.R on AIX (R は製品のリリース)。 代わりの複数インスタンス構成の例については、 149 ページの『UNIX and Linux システムでのディレクトリー構造の例』を参照してください。

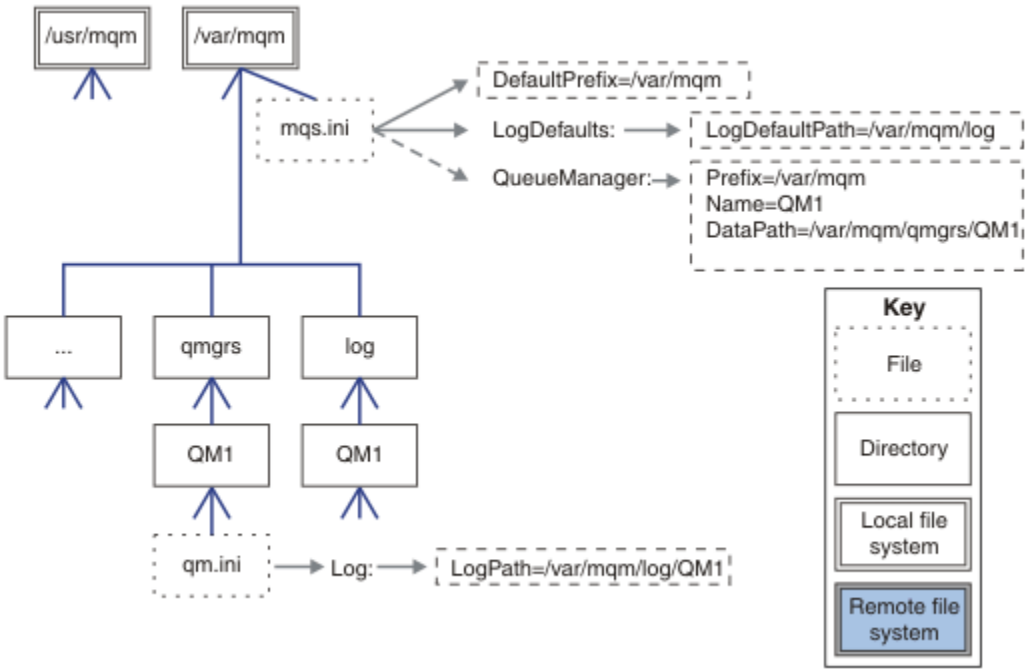


図 *53. UNIX and Linux* システムのデフォルトの *WebSphere MQ v7.R* ディレクトリー構造の例

製品はデフォルトで、AIX の場合は `/usr/mqm` に、その他のシステムの場合は `/opt/mqm` にインストールされます。 作業ディレクトリーは `/var/mqm` ディレクトリーにインストールされます。

注 : /var/mqm ファイル・システムを、IBM WebSphere MQ のインストール前に作成した場合は、mqm ユーザーがディレクトリーの全アクセス権 (例えばファイル・モード 755) を持っていることを確認してください。

log および qmgrs のディレクトリーは、 mqs.ini ファイルの LogDefaultPath および DefaultPrefix の属性の デフォルト値によって定義されているデフォルト位置に示されます。 キュー・マネージャーが作成されると、デフォルトでは、キュー・マネージャーのデータ・ディレクトリーが *DefaultPrefix*/qmgrs に作成され、ログ・ファイル・ディレクトリーが *LogDefaultPath* /log に作成されます。 LogDefaultPath および DefaultPrefix は、デフォルトでキュー・マネージャーおよびログ・ファイルが作成される位置にのみ影響します。 キュー・マネージャー・ディレクトリーの実際の場所は、 mqs.ini ファイルに保存され、ログ・ファイル・ディレクトリーの場所は qm.ini ファイルに保存されます。

キュー・マネージャーのログ・ファイル・ディレクトリーは、LogPath 属性の qm.ini ファイル内に定義されています。 **crtmqm** コマンドで -ld オプションを使用して、キュー・マネージャーの LogPath 属性を設定します (例: **crtmqm** -ld *LogPath* QM1)。 ld パラメーターを省略すると、代わりに LogDefaultPath の値が使用されます。

キュー・マネージャーのデータ・ディレクトリーは、 mqs.ini ファイル内の QueueManager スタンザの DataPath 属性に定義されています。 **crtmqm** コマンドで -md オプションを使用して、キュー・マネージャーの DataPath を設定します (例: **crtmqm** - md *DataPath* QM1)。 md パラメーターを省略すると、代わりに DefaultPrefix または Prefix 属性の値が使用されます。 Prefix は DefaultPrefix より優先されます。

通常、ログ・ディレクトリーとデータ・ディレクトリーの両方を 1 つのコマンドで指定して、QM1 を作成します。

```
crtmqm
  -md DataPath -ld
  LogPath QM1
```

キュー・マネージャーが停止しているときに qm.ini ファイルで DataPath および LogPath 属性を編集すれば、既存のキュー・マネージャーのキュー・マネージャー・ログ・ディレクトリーとデータ・ディレクトリーの位置を変更できます。

errors ディレクトリーへのパスは、/var/mqm 内の他のすべてのディレクトリーへのパスと同様に変更できません。 ただし、ディレクトリーを別のファイル・システムにマウントしたり、 別のディレクトリーにシンボリック・リンクしたりできます。

UNIX and Linux システムでのディレクトリーの内容

キュー・マネージャーと関連付けられたディレクトリーの内容。

製品ファイルの場所については、インストール場所の選択を参照してください。

その他のディレクトリー構成についての詳細は、131 ページの『ファイル・システム・サポートの計画』を参照してください。

147 ページの図 54 は、キュー・マネージャーがしばらく使用された後の WebSphere MQ の典型的なレイアウトです。 実際の構造は、キュー・マネージャーに対して行われた操作によって異なります。

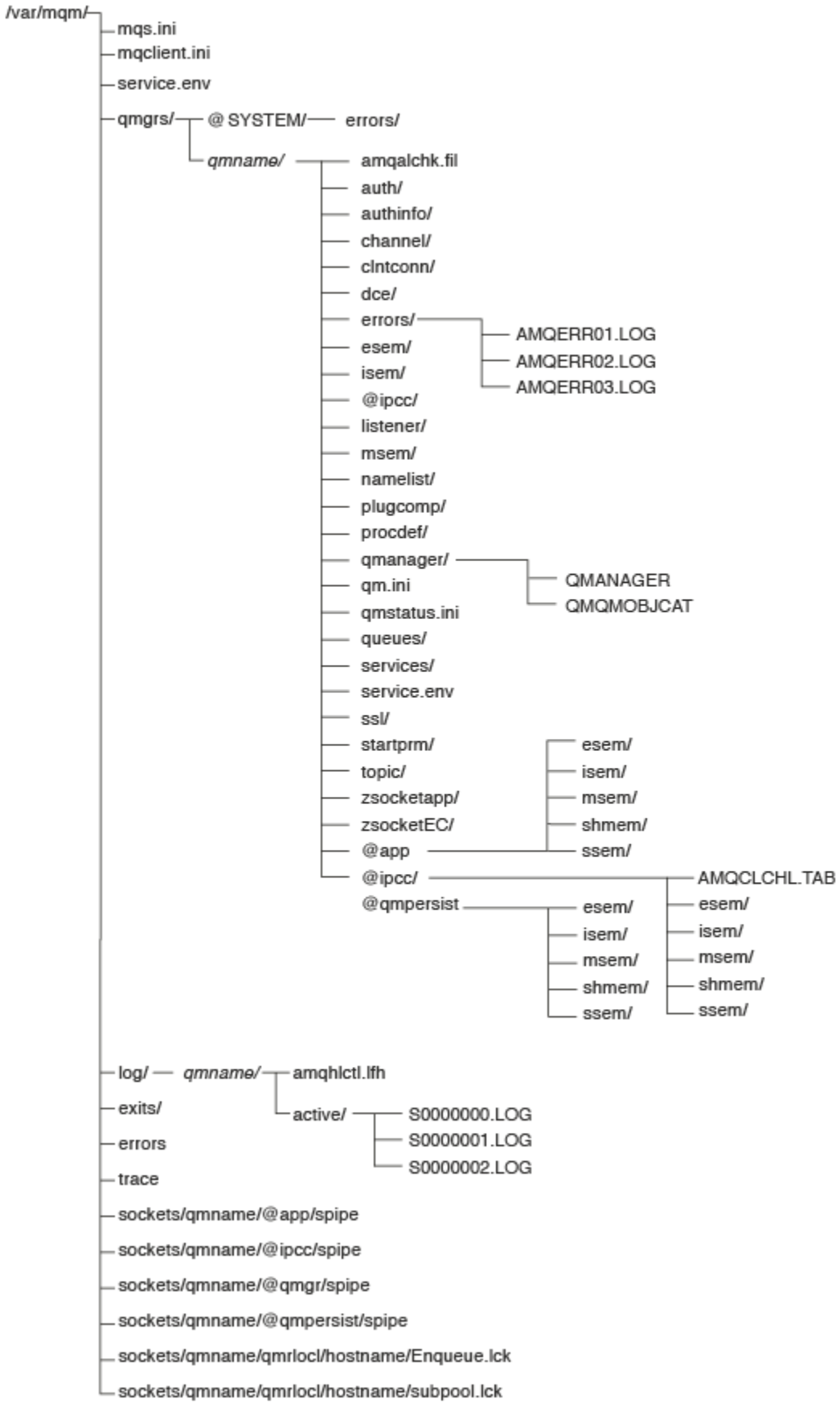


図 *54.* キュー・マネージャーが開始された後のデフォルト・ディレクトリー構造 *(UNIX* システム*)*

/var/mqm/

/var/mqm ディレクトリーには、個々のキュー・マネージャーではなく、WebSphere MQ インストール済み環境全体に適用される構成ファイルと出力ディレクトリーが含まれています。

表 15. *UNIX* システムでの */var/mqm* ディレクトリーの内容の説明	
mqs.ini	キュー・マネージャーの開始時に読み取られる、WebSphere MQ のインストール済み環境全体に関する構成ファイル。 ファイル・パスは、**AMQ_MQS_INI_LOCATION** 環境変数を使用して変更可能です。 これが、**strmqm** コマンドが実行されるシェルで設定され、エクスポートされていることを確認します。
mqclient.ini	WebSphere MQ MQI クライアント・プログラムが読み取るデフォルトのクライアント構成ファイル。 ファイル・パスは、**MQCLNTCF** 環境変数を使用して変更可能です。
service.env	マシンを有効範囲とした、サービス・プロセスに対する環境変数が含まれています。 ファイル・パスは固定です。
errors/	マシンを有効範囲としたエラー・ログ、および FFST ファイル。 ディレクトリー・パスは固定です。 FFST: IBM WebSphere MQ for UNIX and Linux システムも参照してください。
sockets/	各キュー・マネージャーに関する情報が含まれています。システムのみが使用します。
トレース/	トレース・ファイル。 ディレクトリー・パスは固定です。
出口/ **exits64/**	ユーザー・チャネル出口プログラムが含まれているデフォルトのディレクトリー。 場所は mqs.ini ファイルの ApiExit スタンザで変更可能です。

/var/mqm/qmgrs/*qmname*/

/var/mqm/qmgrs/*qmname*/ には、キュー・マネージャーのディレクトリーおよびファイルが含まれています。 このディレクトリーは、アクティブなキュー・マネージャー・インスタンスからの排他的アクセスによりロックされます。 ディレクトリー・パスは、mqs.ini ファイルで直接変更することも、**crtmqm** コマンドの **md** オプションを使用して変更することもできます。

表 16. *UNIX* システムでの */var/mqm/qmgrs/qmname* ディレクトリーの内容の説明	
qm.ini	キュー・マネージャーの開始時に読み取られるキュー・マネージャー構成ファイル。
errors/	キュー・マネージャーを有効範囲としたエラー・ログ。 *qmname* = @system には、不明なキュー・マネージャーまたは使用不可のキュー・マネージャーに対するチャネル関連のメッセージが含まれています。

表 16. *UNIX* システムでの */var/mqm/qmgrs/qmname* ディレクトリーの内容の説明 (続き)	
@ipcc/ **AMQCLCHL.TAB**	WebSphere MQ サーバーが作成し、WebSphere MQ MQI クライアント・プログラムが読み取るデフォルトのクライアント・チャネル制御テーブル。 ファイル・パスは、**MQCHLLIB** 環境変数および **MQCHLTAB** 環境変数を使用して変更可能です。
qmanager	キュー・マネージャー・オブジェクト・ファイル: QMANAGER キュー・マネージャー・オブジェクト・カタログ: QMQMOBJCAT
authinfo/	キュー・マネージャー内で定義された各オブジェクトは、これらのディレクトリーにあるファイルに関連付けられています。 ファイル名は、定義名にほぼ一致しています。WebSphere MQ のファイル名についての理解を参照してください。
チャネル/	
clntconn/	
リスナー/	
namelist/	
procdef/	
キュー/	
services/	
トピック/	
...	WebSphere MQ が使用するその他のディレクトリー (@ipcc など) は、WebSphere MQ のみが変更できます。

/var/mqm/log/*qmname*/

/var/mqm/log/*qmname*/ には、キュー・マネージャーのログ・ファイルが含まれています。 このディレクトリーは、アクティブなキュー・マネージャー・インスタンスからの排他的アクセスによりロックされます。 ディレクトリー・パスは、 qm.ini ファイルで変更することも、 **crtmqm** コマンドの **ld** オプションを使用して変更することもできます。

表 17. *UNIX* システムでの */var/mqm/log/qmname* ディレクトリーの内容の説明	
amqhlctl.lfh	ログ制御ファイル。
active/	このディレクトリーは、S0000000.LOG、S0000001.LOG、S0000002.LOG などの番号が付いたログ・ファイルを含んでいます。

UNIX and Linux システムでのディレクトリー構造の例

UNIX and Linux システムでの代替ファイル・システム構成の例。

さまざまな方法で WebSphere MQ ディレクトリー構造をカスタマイズすることで、多くの異なる目標を達成することができます。

- qmgrs および log ディレクトリーをリモート・ファイル共有システムに配置して、複数インスタンス・キュー・マネージャーを構成します。
- データ・ディレクトリーおよびログ・ディレクトリーに別個のファイル・システムを使用し、それぞれのディレクトリーを異なるディスクに割り振り、入出力競合を削減することでパフォーマンスを向上させます。
- パフォーマンスにより大きな影響を与えるディレクトリーに高速ストレージ・デバイスを使用します。 多くの場合、デバイスの取り付けがローカルか、リモートかではなく、 物理デバイスの待ち時間が、持続メッセージングのパフォーマンスにおけるさらに重要な要素となります。 以下に、 パフォーマンスが重要視されるディレクトリーをその重要度の高い順にリストします。

  1. log

2. qmgrs
3. その他のディレクトリー (/usr/mqm など)

- qmgrs ディレクトリーおよび log ディレクトリーを、例えば冗長ディスク・アレイなどの十分な回復力があるストレージに割り振られたファイル・システム上に作成します。
- 共通エラー・ログは、ネットワーク・ファイル・システム上ではなく、ローカルの var/mqm/errors に保管した方がよいでしょう。こうすることで、ネットワーク・ファイル・システムに関するエラーをログに記録できます。

150 ページの図 55 は、代替 WebSphere MQ ディレクトリー構造の派生元となるテンプレートです。 テンプレートでは、点線によって構成可能なパスを表しています。 この例の点線は、AMQ_MQS_INI_LOCATION 環境変数に保管されている構成情報、および mqs.ini ファイルと qm.ini ファイルに対応する実線に置き換えられます。

注 : パス情報は、mqs.ini または qm.ini のファイルに出現するとおりに示されます。 **crtmqm** コマンドにパス・パラメーターを提供する場合は、キュー・マネージャーのディレクトリー名を省略してください。 キュー・マネージャー名は、マングリング後に WebSphere MQ によってパスに追加されます。

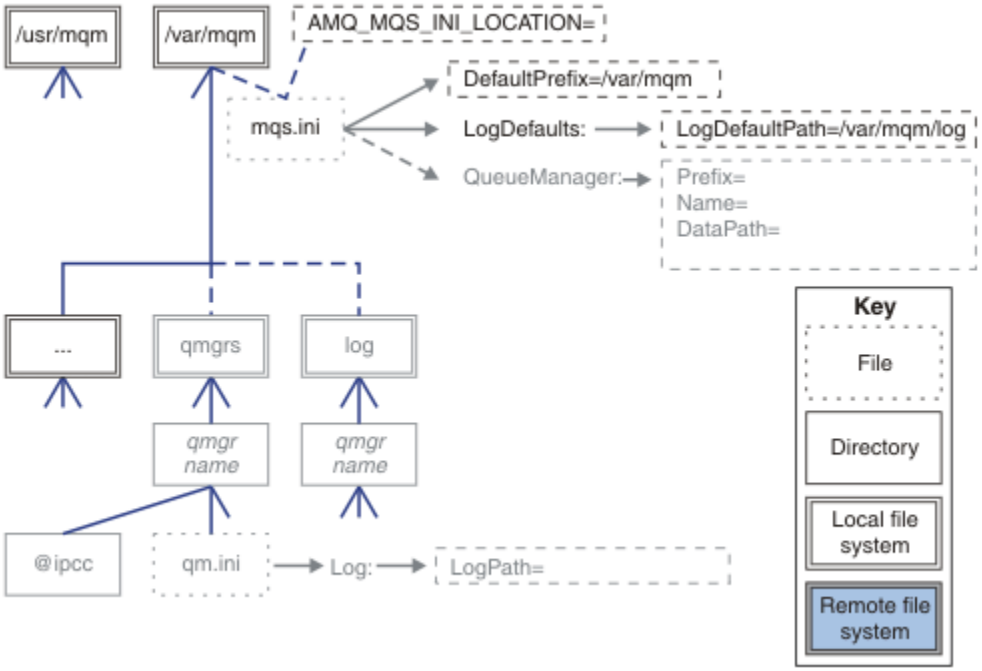


図 *55.* ディレクトリー構造パターン・テンプレート

構成されたディレクトリー構造の例を示します。 最初の例は、**crtmqm** *QM1* コマンドを発行して作成される、WebSphere MQ v7.0.1 の標準的なデフォルト・ディレクトリー構造を示しています。 2 番目の例は、WebSphere MQ の v7.0.1 より前のリリースを使用して作成されたキュー・マネージャーの標準的ディレクトリー構造を示しています。 ディレクトリー構造は変わりません。

バージョン 7.0.1 で新しく作成されたキュー・マネージャーには、v7 の以前のリリースで作成されたものとは異なる構成ファイルがあります。 v7.0.1 フィックスパックを除去して v7.0.0.2 に戻す必要がある場合は、構成ファイルを再作成する必要があります。 Prefix 属性を使用して新規キュー・マネージャーのデータ・ディレクトリーへのパスを定義するだけで済む場合もあれば、キュー・マネージャーのデータ・ディレクトリーおよびログ・ディレクトリーを別の場所に移動させることが必要になる場合もあります。 キュー・マネージャーを再構成する最も安全な方法は、キュー・マネージャーのデータ・ディレクトリーおよびログ・ディレクトリーを保存し、キュー・マネージャーを削除して再作成し、新しい場所にあるデータ・ディレクトリーおよびログ・ディレクトリーを、保存したもので置き換えることです。

v7.0.1 以降のリリースでの標準的なディレクトリー構造

151 ページの図 56 は、v7.0.1 でコマンド **crtmqm** *QM1* を発行して作成されたデフォルト・ディレクトリー構造です。

mqs.ini ファイルには、 DefaultPrefix の値を参照して作成された、QM1 キュー・マネージャー用のスタンザがあります。 qm.ini ファイルの Log スタンザには、mqs.ini の LogDefaultPath を参照することによって LogPath の値が設定されます。

DataPath および LogPath のデフォルト値を指定変更するには、 オプションの **crtmqm** パラメーターを使用します。

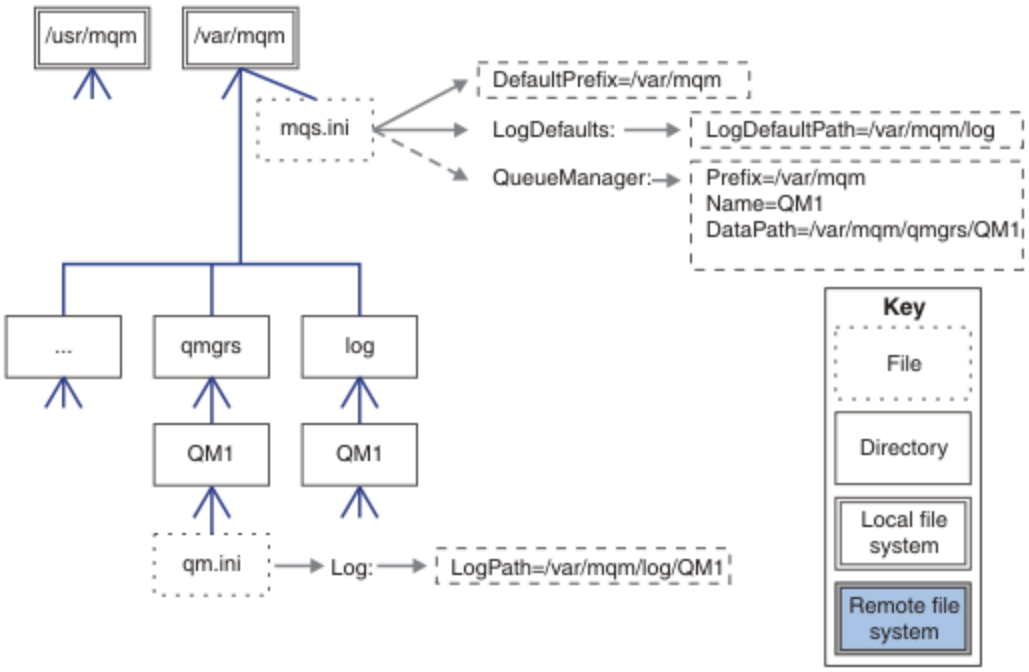


図 *56. UNIX and Linux* システムのデフォルトの *WebSphere MQ v7.R* ディレクトリー構造の例

v7.0.1 より前のリリースでの 標準的なディレクトリー構造

DataPath 属性は、WebSphere MQ v7.0.1 より前には存在しませんでした。この属性は mqs.ini ファイルにはありません。 qmgrs ディレクトリーの位置は、Prefix 属性を使用して構成されていました。 個々のディレクトリーの位置は、シンボリック・リンクを使用してさまざまなファイル・システムの位置を指すことで構成できました。

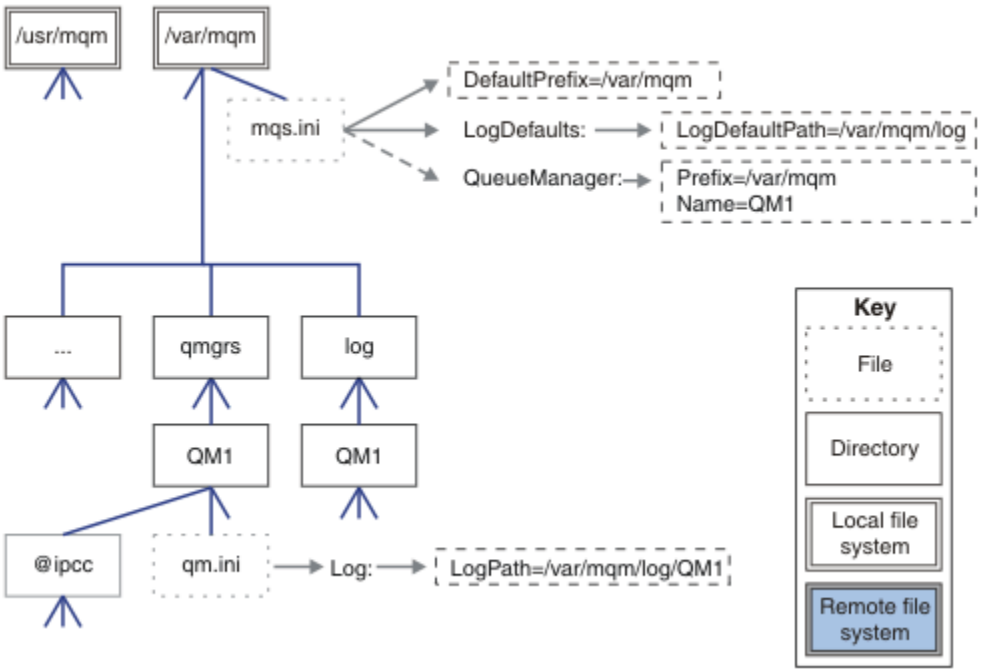


図 *57. v7.0.1* より前のリリースでの 標準的なディレクトリー構造

デフォルトの qmgrs および log ディレクトリーの共有 (リリース v7.0.1 以降)

153 ページの『すべて共有 (リリース v7.0.1 以降)』 の代わりに、 `qmgrs` ディレクトリーと `log` ディレクトリーを別々に共有することもできます (152 ページの図 58)。 この構成では、デフォルトの mqs.ini がローカル `/var/mqm` ファイル・システムに保管されるため、`AMQ_MQS_INI_LOCATION` を設定する必要はありません。 ファイルおよびディレクトリー (`mqclient.ini` や `mqserver.ini` など) も共有されません。

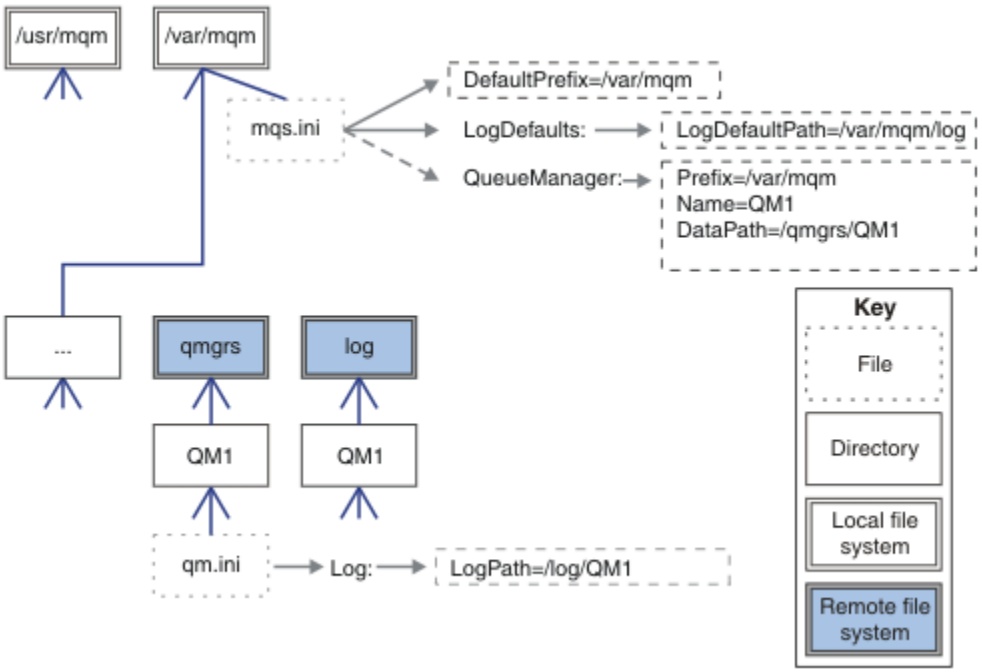


図 *58. qmgrs* および *log* ディレクトリーの共有

指定された qmgrs ディレクトリーと log ディレクトリーの共有 (リリース v7.0.1 以降)

153 ページの図 59 の構成では、log および qmgrs を、共通の /ha というリモート・ファイル共有システムに配置しています。 2 つの異なる方法で同じ物理構成を作成できます。

1. LogDefaultPath=/ha を設定してから、コマンド **crtmqm** -md */ha/qmgrs* QM1 を実行します。 結果は、153 ページの図 59 に示すとおりになります。
2. デフォルト・パスを変更せずに、コマンド **crtmqm** -ld */ha/log* - md */ha/qmgrs* QM1 を実行します。

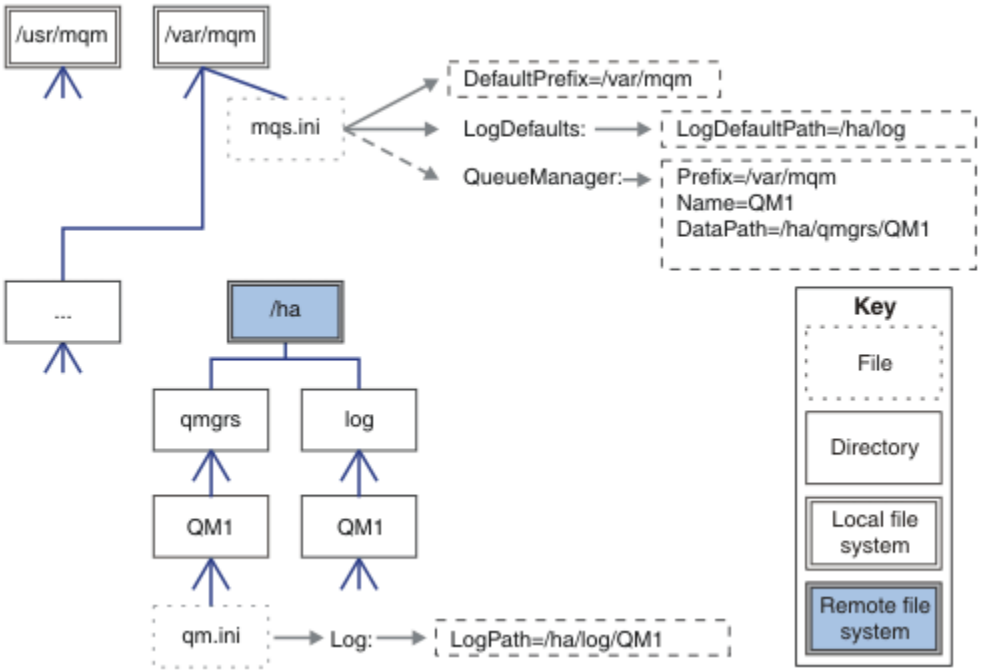


図 *59.* 名前付き *qmgrs* と *log* ディレクトリーの共有

すべて共有 (リリース v7.0.1 以降)

154 ページの図 60 は、 高速ネットワーク・ファイル・ストレージを備えたシステム用の簡単な構成です。

/var/mqm をリモート・ファイル共有システムとしてマウントします。 デフォルトでは、QM1 を始動すると、QM1 は /var/mqm を探索してそれをファイル共用システムから検出し、/var/mqm にある mqs.ini ファイルを読み取ります。 すべてのサーバー上のキュー・マネージャーに対して単一の /var/mqm/mqs.ini ファイルを使用する代わりに、各サーバー上の AMQ_MQS_INI_LOCATION 環境変数が異なる mqs.ini ファイルを指すように設定することができます。

注 : /var/mqm/errors/ 内の汎用エラー・ファイルの内容は、異なるサーバー上のキュー・マネージャー間で共有されます。

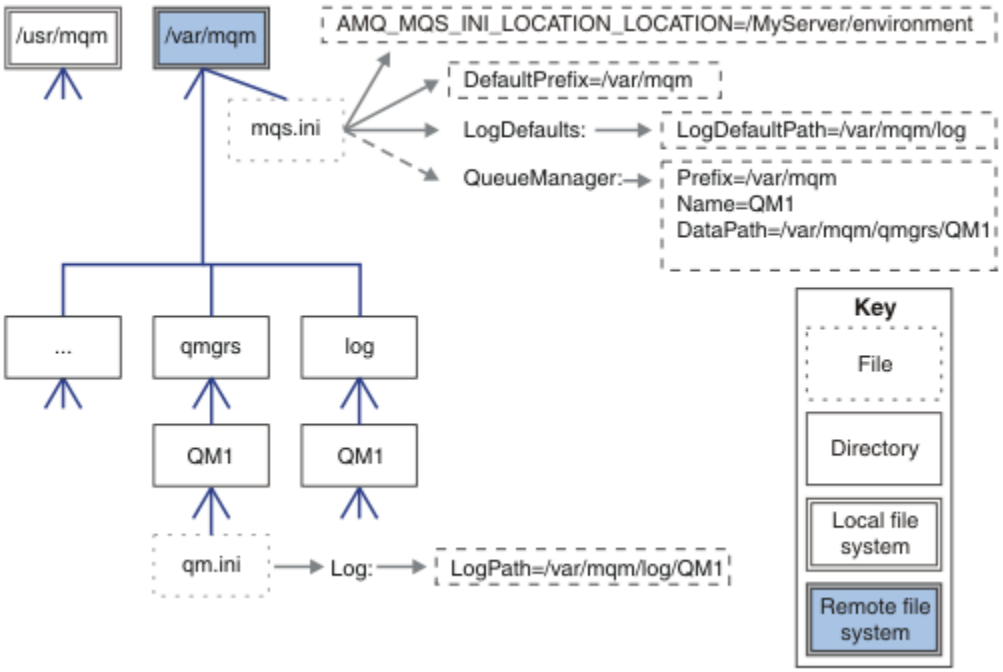


図 *60.* すべて共有

これは、複数インスタンス・キュー・マネージャーに使用することはできません。 複数インスタンス・キュー・マネージャーの各ホストは、ローカル・データ (セマフォーや共有メモリーなど) を継続的に把握しておくために、`/var/mqm` のローカル・コピーを個別に必要とするためです。 これらのエンティティーは、ホスト間で共有できません。

Windows システムでのディレクトリー構造

Windows 上のキュー・マネージャー構成情報およびディレクトリーを検出する方法を示します。

IBM WebSphere MQ for Windows のデフォルトのインストール・ディレクトリーは、以下のとおりです。

32 ビット
```
C:\Program Files\IBM\WebSphere MQ
```

64 ビット
```
C:\Program Files (x86)\IBM\WebSphere MQ
```

インストール情報は Windows レジストリーに格納されます。 IBM WebSphere MQ 情報が格納されるレジストリー・キーは以下のとおりです。

32 ビット
```
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\
```

64 ビット
```
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\IBM\WebSphere MQ\
```

各インストールに固有のサブキーがあります。

```
Installation\<InstallationName>\
```

IBM WebSphere MQ のデータ・ディレクトリーを指すパスは `WorkPath` というストリング値に格納され、ログのデフォルトのディレクトリーは `LogDefaultPath` に格納されます。 キュー・マネージャーのデータ・ディレクトリーは、 *WorkPath*`\qmgrs\`*Qmgrname* に作成されます。 キュー・マネージャーのログは *LogDefaultPath*`\`*QmgrName* に作成されます。 155 ページの図 61 を参照してください。

IBM WebSphere MQ のインストール時にキュー・マネージャー・データ・ディレクトリーおよびログ・ディレクトリーを定義する場合、*WorkPath* および *LogDefaultPath* はカスタマイズされたパス情報で更新されます。

WorkPath および *LogDefaultPath* は、キュー・マネージャーの作成にのみ使用されます。

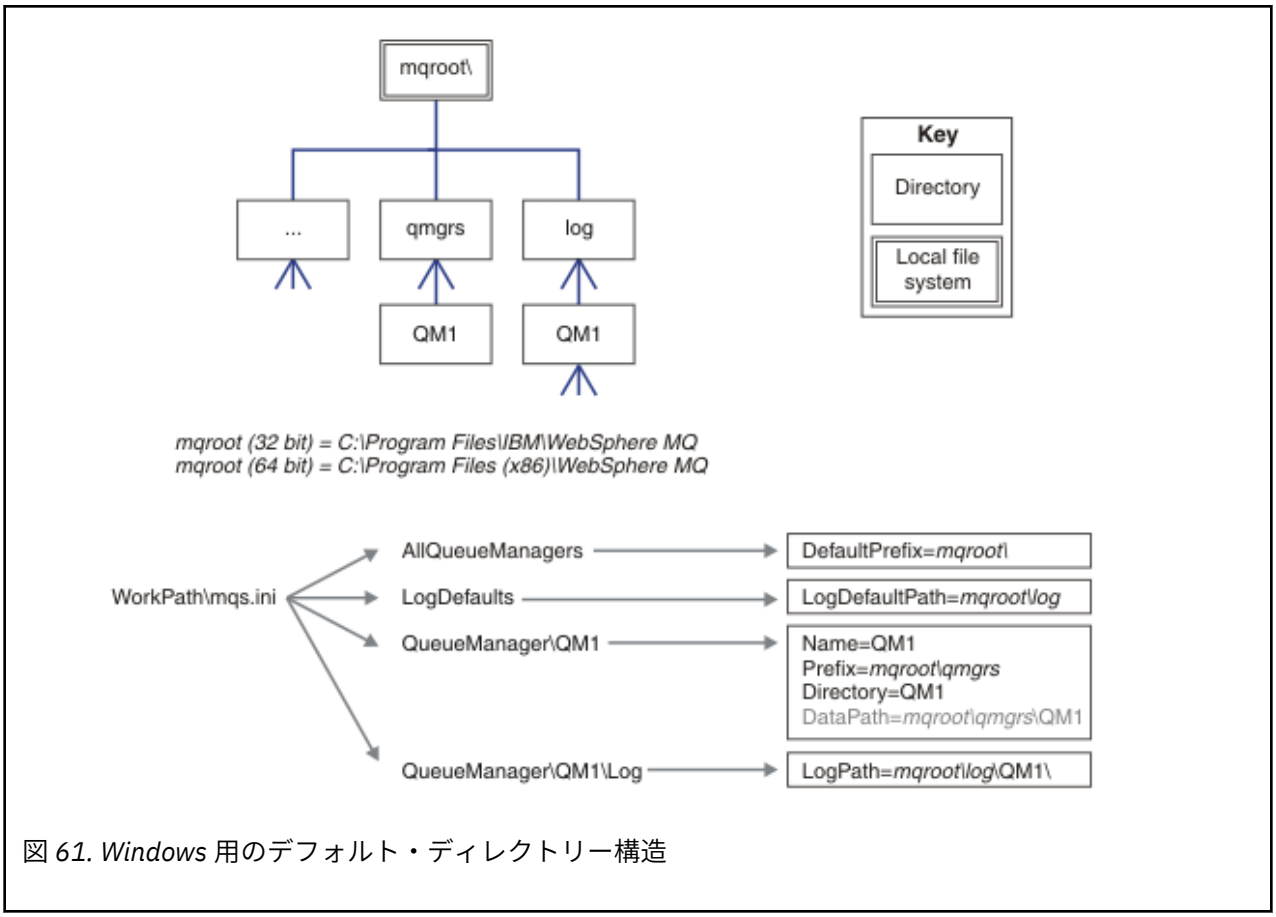


図 *61. Windows* 用のデフォルト・ディレクトリー構造

複数インスタンス・キュー・マネージャー

複数インスタンス・キュー・マネージャーを構成するには、ログ・ディレクトリーおよびデータ・ディレクトリーをネットワーク・ストレージ、できればキュー・マネージャーのインスタンスを実行しているサーバーとは別のサーバー上に置く必要があります。

crtmqm コマンドでは、2 つのパラメーター、**-md** および **-ld** が提供されており、キュー・マネージャー・データ・ディレクトリーおよびログ・ディレクトリーの場所を指定するのを容易にしています。 **-md** パラメーターを指定すると、以下の 4 つの効果があります。

1. `mqs.ini` スタンザ `QueueManager\`*`QmgrName`* には、キュー・マネージャーのデータ・ディレクトリーを指す *DataPath* という新しい変数が含まれています。 *Prefix* 変数とは異なり、パスにはキュー・マネージャー・ディレクトリーの名前が含まれます。
2. mqs.ini ファイルに格納されるキュー・マネージャー構成情報が、*Name*、*Prefix*、*Directory*、および *DataPath* に減らされます。

ディレクトリーの内容

WebSphere MQ ディレクトリーの場所および内容をリストします。

WebSphere MQ 構成には、主に 3 つのファイルおよびディレクトリーのセットがあります。

1. 実行可能ファイル、およびその他の読み取り専用ファイル (readme ファイルなど)、WebSphere MQ エクスプローラー・プラグインおよびヘルプ・ファイル、およびライセンス・ファイル (これらは保守が適用される場合のみ更新されます)。 これらのファイルについては、156 ページの表 18 で説明されています。
2. 特定のキュー・マネージャーに固有でない、潜在的に変更可能なファイルおよびディレクトリー。 これらのファイルおよびディレクトリーについては、156 ページの表 19 で説明されています。

3. サーバー上の各キュー・マネージャーに固有のファイルおよびディレクトリー。 これらのファイルおよびディレクトリーについては、156 ページの表 18 で説明されています。

リソース・ディレクトリーおよびファイル

リソース・ディレクトリーおよびファイルには、キュー・マネージャーを実行するためのすべての実行可能コードとリソースが含まれています。 インストール済み環境固有の IBM WebSphere MQ 構成レジストリー・キーにある変数 *FilePath* には、リソース・ディレクトリーへのパスが含まれます。

表 *18. FilePath* ディレクトリー内のディレクトリーおよびファイル

ファイル・パス	目次
FilePath\bin	コマンドおよび DLL
FilePath\bin64	コマンドおよび DLL (64 ビット)
FilePath\conv	データ変換テーブル
FilePath\doc	ウィザード・ヘルプ・ファイル
FilePath\MQExplorer	エクスプローラーおよびエクスプローラー・ヘルプ Eclipse プラグイン
FilePath\gskit8	グローバル・セキュリティー・キット
FilePath\java	Java リソース (JRE を含む)
FilePath\licenses	ライセンス情報
FilePath\Non_IBM_License	ライセンス情報
FilePath\properties	内部的に使用
FilePath\Tivoli	
FilePath\tools	開発リソースおよびサンプル
FilePath\Uninst	内部的に使用
FilePath\README.TXT	README ファイル

キュー・マネージャーに固有でないディレクトリー

ディレクトリーの中には、特定のキュー・マネージャーに固有でないファイル (トレース・ファイルやエラー・ログなど) が含まれているものがあります。 *DefaultPrefix* 変数には、そうしたディレクトリーへのパスが含まれています。 *DefaultPrefix* は、AllQueueManagers スタンザの一部です。

表 *19. DefaultPrefix* ディレクトリー内のディレクトリーおよびファイル

ファイル・パス	目次
DefaultPrefix\Config	内部的に使用
DefaultPrefix\conv	ccsid.tbl データ変換制御ファイル (データ変換で説明されています)
DefaultPrefix\errors	キュー・マネージャー以外のエラー・ログ、AMQERR*nn*.LOG
DefaultPrefix\exits	チャネル出口プログラム
DefaultPrefix\exits64	チャネル出口プログラム (64 ビット)
DefaultPrefix\ipc	使用されません
DefaultPrefix\Qmgrs	157 ページの表 20 で説明されています

表 *19. DefaultPrefix* ディレクトリー内のディレクトリーおよびファイル (続き)	
ファイル・パス	**目次**
DefaultPrefix\trace	トレース・ファイル
DefaultPrefix\amqmjpse.txt	内部的に使用

キュー・マネージャー・ディレクトリー

キュー・マネージャーを作成するときに、キュー・マネージャーに固有の新規のディレクトリーのセットが作成されます。

-md *filepath* パラメーターを指定してキュー・マネージャーを作成する場合、パスは mqs.ini ファイルのキュー・マネージャー・スタンザの *DataPath* 変数に格納されます。 **-md** *filepath* パラメーターを指定しないでキュー・マネージャーを作成する場合、キュー・マネージャー・ディレクトリーは *DefaultPrefix* に格納されているパスに作成され、そのパスが mqs.ini ファイルのキュー・マネージャー・スタンザの *Prefix* 変数にコピーされます。

表 *20. DataPath* および *Prefix/Qmgrs/QmgrName* ディレクトリー内のディレクトリーおよびファイル	
ファイル・パス	**目次**
DataPath\@ipcc	AMQCLCHL.TAB (クライアント接続テーブル) のデフォルトの場所。
DataPath\authinfo	内部的に使用
DataPath\channel	
DataPath\clntconn	
DataPath\errors	エラー・ログ、AMQERR*nn*.LOG
DataPath\listener	内部的に使用
DataPath\namelist	
DataPath\plugcomp	
DataPath\procdef	
DataPath\qmanager	
DataPath\queues	
DataPath\services	
DataPath\ssl	
DataPath\startprm	
DataPath\topic	
DataPath\active	
DataPath\active.dat	
DataPath\amqalchk.fil	
DataPath\master	
DataPath\master.dat	
DataPath\qm.ini	キュー・マネージャー構成
DataPath\qmstatus.ini	キュー・マネージャー状況

表 *20. DataPath* および *Prefix/Qmgrs/QmgrName* ディレクトリー内のディレクトリーおよびファイル *(続き)*	
ファイル・パス	**目次**
Prefix\Qmgrs*QmgrName*	内部的に使用
Prefix\Qmgrs\@SYSTEM	使用されません
Prefix\Qmgrs\@SYSTEM\errors	

IBM WebSphere MQ と UNIX System V IPC リソース

キュー・マネージャーはいくつかの IPC リソースを使用します。 **ipcs -a** を使用して、どのリソースが使用されているかを調べます。

この情報は、 UNIX and Linux システム上で稼働する IBM WebSphere MQ にのみ適用されます。

IBM WebSphere MQ は System V プロセス間通信 (IPC) リソース (セマフォー および共有メモリー・セグメント) を使用して、システム・コンポーネント間のデータを保管したり、渡したりします。 これらのリソースは、キュー・マネージャー・プロセスおよびキュー・マネージャーに接続するアプリケーションが使用します。 IBM WebSphere MQ MQI クライアントは、IBM WebSphere MQ トレース制御を除き、IPC リソースを使用しません。 UNIX コマンド **ipcs -a** を使用すると、マシンで現在使用されている IPC リソースの数とサイズの全情報を取得できます。

AIX 上の共有メモリー

AIX メモリー制限のため、特定のアプリケーション・タイプで接続できない場合、大抵は環境変数 EXTSHM=ON を設定することによって解決できます。

AIX 上のいくつかの 32 ビット・プロセスで、WebSphere MQ キュー・マネージャーへ接続する機能に影響を及ぼすオペレーティング・システムの制限が存在する場合があります。 WebSphere MQ への各標準接続では共有メモリーを使用しますが、他の UNIX and Linux プラットフォームとは異なり、AIX では、32 ビット・プロセスで接続できる共有メモリー・セットは 11 個だけです。

ほとんどの 32 ビット・プロセスではこの制限が生じることはありませんが、メモリー所要量の多いアプリケーションでは、理由コード 2102: MQRC_RESOURCE_PROBLEM で WebSphere MQ への接続が失敗する場合があります。 以下のアプリケーション・タイプで、このようなエラーが生じる場合があります。

- 32 ビット Java 仮想マシンで実行されるプログラム
- 大きいまたは非常に大きいメモリー・モデルを使用しているプログラム
- 多くのキュー・マネージャーまたはデータベースに接続しているプログラム
- それ自身の共有メモリー・セットに接続しているプログラム

AIX で提供されている、32 ビット・プロセス用の拡張共有メモリー・フィーチャーを使用することにより、より多くの共有メモリーを接続できます。 このフィーチャーを使用してアプリケーションを実行するには、キュー・マネージャーおよびプログラムを開始する前に、環境変数 EXTSHM=ON をエクスポートします。 ほとんどの場合、EXTSHM=ON フィーチャーを使用することによってこのエラーを防ぐことができますが、shmctl 関数の SHM_SIZE オプションを使用するプログラムとの互換性はありません。

WebSphere MQ MQI クライアント・アプリケーションおよびすべての 64 ビット・プロセスは、この制限の影響を受けません。 それらは EXTSHM が設定されているかどうかに関係なく、WebSphere MQ キュー・マネージャーに接続できます。

WebSphere MQ および UNIX のプロセス優先順位

プロセス優先順位の *nice* 値を設定する際の良い方法。

この情報は、 UNIX and Linux システム上で稼働する WebSphere MQ にのみ適用されます。

プロセスをバックグラウンドで実行する場合、呼び出し側シェルによって、そのプロセスの *nice* 値が高くなる場合があります (従って優先順位は下がります)。 これによって、全般的に WebSphere MQ のパフォー

マンスへの影響が見られる場合があります。 負荷の大きい状態で、優先順位が高く直ちに実行可能なスレッドが多数あり、いくつかのスレッドの優先順位が低い場合、オペレーティング・システムのスケジューリングの特性によって、優先順位の低いスレッドからプロセッサー時間が奪われる可能性があります。

`runmqlsr` など、キュー・マネージャーに関連付けられたプロセスで別個に開始されたものには、それらが関連付けられているキュー・マネージャーと同じ *nice* 値を持たせることをお勧めします。 シェルがバックグラウンド・プロセスに高い *nice* 値を割り当てることがないようにしてください。 例えば、ksh では、"`set +o bgnice`" 設定を使用して、バックグラウンド・プロセスの *nice* 値を ksh が上げないようにします。 実行中のプロセスの *nice* 値は、 "ps -efl" リストの *NI* 列を調べることによって確認できます。

また、WebSphere MQ アプリケーション・プロセスをキュー・マネージャーと同じ *nice* 値を使用して開始してください。 異なる *nice* 値で実行される場合、アプリケーション・スレッドがキュー・マネージャー・スレッドをブロックするかまたはその逆が生じ、パフォーマンスが 低下する可能性があります。

HP Integrity NonStop Server での IBM WebSphere MQ クライアント環境の計画

IBM WebSphere MQ 環境の計画時には、HP Integrity NonStop Server 環境と HP NonStop TMF を考慮する必要があります。 この情報を利用して、IBM WebSphere MQ Client for HP Integrity NonStop Server を実行する環境を計画します。

IBM WebSphere MQ Client for HP Integrity NonStop Server アーキテクチャーを計画する前に、IBM WebSphere MQ Client for HP Integrity NonStop Server の基本的な概念を習得しておいてください。 IBM WebSphere MQ Client for HP Integrity NonStop Server の技術概要のトピックを参照してください。

HP Integrity NonStop Server 環境の準備

インストール済み環境をすぐに検証するかしないかに応じて、インストールする前に環境を準備する必要があります。

インストールするには、以下の項目が必要です。

- 要件を満たすユーザー ID。 ユーザー ID の要件について詳しくは、HP Integrity NonStop Server でのユーザーとグループのセットアップを参照してください。
- インストール・ファイル用に指定できる OSS および Guardian ファイル・システムにおける検証済みの場所。
- 操作可能 OSS シェルと OSS ファイル・システム。 以下のタスクを実行することによって、ファイル・システムを検証できます。
  - OSS 環境 (シェル) にログオンします。 使用する OSS インストール・ルート・ディレクトリーへの書き込み権限が付与されていることを確認します。
  - MQM グループ内のユーザー ID を使用して TACL 環境にログオンします。 使用するボリュームが要件を満たしており、アクセス可能であること、またサブボリュームが存在していないことを確認します。

  別名 (存在する場合) または完全なプリンシパルを使用して、OSS と TACL にログインできます。

インストール済み環境が使用可能であることを続けてすぐに確認する場合、以下のオプション項目も必要になる場合があります。

- OSS 環境で操作可能でアクセス可能なローカル・ソケット・サブシステム。
- 操作可能な TCP/IP サブシステム。

TMF 調整済みグローバル作業単位を使用する場合、以下の項目が必要になります。

- 操作可能な TMF サブシステム。
- 操作可能な Pathway (TS/MP) サブシステム。

これらの重要なサブシステムの状況について疑問がある場合、システム管理者に相談してください。

IBM WebSphere MQ と HP NonStop TMF

IBM WebSphere MQ クライアント (HP Integrity NonStop Server 用) は、HP NonStop Transaction Management Facility (HP NonStop TMF) によって調整される作業単位に参加することができます。 HP NonStop TMF によるトランザクション調整は、キュー・マネージャーが IBM WebSphere MQ Version 7.1 以降である場合にのみサポートされます。

IBM WebSphere MQ に備わっている TMF/Gateway は、リモート・キュー・マネージャーと通信するために、TMF 調整からのトランザクションを eXtended Architecture (XA) トランザクション調整に変換します。 IBM WebSphere MQ に備わっている TMF/Gateway は、HP NonStop TMF で提供されるサービスを使って TMF とキュー・マネージャー・トランザクションの間のブリッジとなり、Pathway 環境で実行されるよう設計されています。

HP NonStop TMF ソフトウェアは、要件の大きい環境においてトランザクション保護とデータベース整合性を提供します。 HP NonStop TMF について詳しくは、「HP NonStop TMF Introduction」を参照してください。

IBM WebSphere MQ に付属の TMF/Gateway を構成する方法については、HP Integrity NonStop Server の構成を参照してください。

HP NonStop TMF の使用

HP NonStop Transaction Management Facility (TMF) は、HP Integrity NonStop Server 上のネイティブ・トランザクション・マネージャーであり、ファイル・システムおよびリレーショナル・データベース・マネージャー (SQL/MP および SQL/MX) に統合されています。

IBM WebSphere MQ クライアント (HP Integrity NonStop Server 用) は TMF を使用してグローバル作業単位を調整することができます。

グローバル作業単位を調整するために、TMF はトランザクション・マネージャーとして機能します。アプリケーションは TMF で提供される API を使ってグローバル作業単位を開始、コミット、およびバックアウトする必要があります。 アプリケーションは、BEGINTRANSACTION を呼び出してグローバル作業単位を開始した後、同期点制御内で MQPUT、MQPUT1、および MQGET 呼び出しを発行することにより、グローバル作業単位内の IBM WebSphere MQ リソースを更新します。 次に、アプリケーションは ENDTRANSACTION を呼び出してグローバル作業単位をコミットするか、ABORTTRANSACTION を呼び出してバックアウトすることができます。

TMF トランザクションを使用するアプリケーションは一度に 1 つのトランザクションだけをアクティブに処理できます。ただし RESUMETRANSACTION を使用すると、以前のアクティブ・トランザクションを完了または中止しなくても、アプリケーションは別のアクティブ・トランザクションに切り替えたり、TMF トランザクションとの関連付けをすべて解除したりすることができます。 すべての MQPUT、MQPUT1、または MQGET 呼び出しは、現行のアクティブ TMF トランザクションが存在する場合はその下で、存在しない場合はローカル作業単位の下で行われます。 したがって、正しい作業単位の中でこれらの呼び出しが確実に行われるよう、アプリケーションを注意深く考慮する必要があります。

グローバル作業単位内で、アプリケーションは IBM WebSphere MQ リソースの更新に加えて Enscribe ファイル、SQL/MP データベース、または SQL/MX データベースを更新することもできます。

グローバル作業単位の使用

グローバル作業単位は TMF トランザクションとして実施されます。 アプリケーションは BEGINTRANSACTION を呼び出してグローバル作業単位を開始し、ENDTRANSACTION を呼び出して作業単位をコミットするか、ABORTTRANSACTION を呼び出して作業単位をバックアウトします。 アプリケーションは、他の TMF API 呼び出しも使用できます。

アプリケーションは、別のアプリケーションから TMF トランザクションを継承できます。 例えば、あるアプリケーション (最初のアプリケーション) がトランザクション内で作業を実行した後、さらに処理を行うために 2 番目のアプリケーションに応答してトランザクションを渡すことができます。 このようにして、最初のアプリケーションと 2 番目のアプリケーションの両方が、IBM WebSphere MQ キューの更新およびファイルやデータベースの更新を伴う同じグローバル作業単位に参加できます。 アプリケーション間で TMF トランザクションを受け渡しできるということは、複数の IBM WebSphere MQ アプリケーションが同じグローバル作業単位内でメッセージング操作を実行できるということです。

アプリケーションは、複数のアクティブ TMF トランザクションを同時に管理および制御できます。 トランザクションはアプリケーション自体が開始することも、他のアプリケーションから継承することも、その両方を行うことも可能です。 つまり、アプリケーションは、複数のグローバル作業単位に同時に関与できます。

プロセス当たりの並行アクティブ TMF トランザクションの最大数は 1000 です。これは、アーキテクチャー上の制限です。 アプリケーションが複数の TMF トランザクションを管理している場合、現行トランザクションになるのはどの時点でも 1 つだけです。 あるいは、いずれのトランザクションも現行トランザクションにならないこともあります。 アプリケーションは、TMF API 呼び出し (RESUMETRANSACTION、ACTIVATERECEIVETRANSID、TMF_SET_TX_ID など) を使用して、現行トランザクションであるという状態をトランザクション間で移動させたり、どのトランザクションも現行トランザクションではないことを指定したりできます。 アプリケーションはこの制御レベルを使用して、メッセージング操作の実行がローカル作業単位内であるか、グローバル作業単位内であるか、または同期点制御の外側であるかを決定します。

- 現行 TMF トランザクションが存在しないときにアプリケーションが同期点制御内で MQPUT、MQPUT1、または MQGET を呼び出した場合、IBM WebSphere MQ はローカル作業単位内でその呼び出しを処理します。
- アプリケーションに現行 TMF トランザクションがある状態でアプリケーションが同期点制御内で MQPUT、MQPUT1、または MQGET を呼び出した場合、IBM WebSphere MQ は現行 TMF トランザクションによって実施されるグローバル作業単位内で呼び出しを処理します。
- アプリケーションが同期点制御の外側で MQPUT、MQPUT1、または MQGET を呼び出した場合、呼び出しの時点でアプリケーションに現行 TMF トランザクションがあるかどうかに関係なく、IBM WebSphere MQ は同期点制御の外側で呼び出しを処理します。

処理中にソフトウェア障害やハードウェア障害が発生しして、データ保全性を保持するためにトランザクションをバックアウトする必要があることを IBM WebSphere MQ またはオペレーティング・システムが決定しない限り、IBM WebSphere MQ が MQI 呼び出し中にアプリケーションの TMF トランザクションの状態を変更することはありません。 各 MQI 呼び出しは、アプリケーションに制御を戻す直前に、アプリケーションのトランザクション状態を復元します。

長期実行トランザクションの回避

TMF トランザクションのアクティブ状態が数十秒を超えるアプリケーションは設計しないようにしてください。 長期実行トランザクションは、TMF の循環監査証跡が満杯になる原因となります。 TMF はシステム全体にわたる重要なリソースであるため、TMF はアクティブ状態が長すぎるアプリケーション・トランザクションをバックアウトすることによって、自らを保護します。

キューからメッセージを取得することによってアプリケーション内の処理が進められ、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て作業単位内で処理するとします。 一般に、アプリケーションは待機オプションを指定して同期点制御内で MQGET を呼び出して、キューからメッセージを取得します。

一方、アプリケーションがグローバル作業単位を使用している場合は、長期実行トランザクションを回避するために、MQGET 呼び出しで指定する待機間隔を短くする必要があります。 これは、アプリケーションがメッセージを取得するために MQGET 呼び出しを複数回発行しなければならない場合があることを意味します。

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。

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
103-8510
東京 103-8510、日本

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.

© Copyright IBM Corp. 2007 年, 2024

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。これらの名前はすべて架空のものであり、名前や住所が類似する個人や企業が実在して 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 IBM 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 の米国およびその他の国における商標です。

この製品には、 Eclipse Project (http://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: