9.4

IBM MQ の管理

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、575 ページの『特記事項』に記載されている情報をお読みください。

本書は、 IBM® MQ バージョン 9 リリース 4、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

目次

IBM MQ の管理

IBM MQ キュー・マネージャーと関連リソースを管理する時には、そうしたリソースをアクティブ化したり管理したりするための一連のタスクから好みの方法を選択できます。

このタスクについて

IBM MQ オブジェクトに対しては、ローカル管理またはリモート管理を行うことができます。

ローカル管理

ローカル管理とは、ローカル・システムに定義したキュー・マネージャーで管理タスクを実行することです。 例えば、TCP/IP の端末エミュレーション・プログラム **telnet** を介して、他のシステムにアクセスし、そこで管理作業を行うことができます。 IBM MQ では、これをローカル管理と考えることができます。チャネルとは無関係であり、通信はオペレーティング・システムによって管理されるからです。

詳細については、127 ページの『ローカル IBM MQ オブジェクトの処理』を参照してください。

リモート管理

IBM MQ は、リモート管理を介して 1 つの地点からの管理をサポートします。 リモート管理を使用すると、別のシステムで処理されるコマンドを、ユーザーのローカル・システムから発行することができます。これは、IBM MQ Explorer にも適用されます。 例えば、リモート・コマンドを発行して、リモート・キュー・マネージャー上のキュー定義を変更することができます。 この場合、別のシステムにログオンする必要はありませんが、適切なチャネルを定義しておく必要があります。 ターゲット・システム上のキュー・マネージャーおよびコマンド・サーバーは、実行中である必要があります。

一部のコマンドは、このような方法では発行することができません。特に、キュー・マネージャーの作成や開始、およびコマンド・サーバーの開始などの際には、このような方法では発行できません。 このようなタスクを実行するためには、リモート・システムにログオンしてそこからコマンドを発行するか、あるいはユーザーの代わりにコマンドを発行するプロセスを作成する必要があります。 この制約事項は、IBM MQ Explorer にも適用されます。

詳細については、195 ページの『リモート IBM MQ オブジェクトの処理』を参照してください。

IBM MQ でキュー・マネージャーと関連リソースを作成して管理するには、さまざまな方法があります。例えば、コマンド行インターフェース、グラフィカル・ユーザー・インターフェース、管理 API などの方法です。

IBM MQ の管理で使用できるコマンド・セットは、オペレーティング・システムによって異なります。

- 8 ページの『IBM MQ の制御コマンド』
- 8 ページの『IBM MQ スクリプト (MQSC) コマンド』
- 8 ページの『プログラマブル・コマンド・フォーマット (PCF)』
- The administrative REST API
- IBM i 9 ページの『IBM i の制御言語 (CL)』

そのほかに、IBM MQ オブジェクトを作成して管理するためのオプションもあります。

- Windows Linux 9 ページの『IBM MQ Explorer』
- 9 ページの『IBM MQ Console』
- Windows 10 ページの『Microsoft Cluster Service (MSCS)』

z/OS IBM MQ for z/OS®での管理インターフェースおよびオプションについては、449 ページの『Administering IBM MQ for z/OS』を参照してください。

PCF コマンドを使用することにより、ローカル・キュー・マネージャーとリモート・キュー・マネージャーの両方に対する管理タスクとモニター・タスクの一部を自動化できます。 プラットフォームによっては、IBM MQ 管理インターフェース (MQAI) を使用して、これらのコマンドを簡略化することもできます。 管理

タスクを自動化するための詳細については、24 ページの『PCF コマンドによる IBM MQ 管理の自動化』を参照してください。

関連概念
IBM MQ の技術概要
関連タスク
計画
構成
関連資料
コマンド・セットの比較

IBM MQ キュー・マネージャーおよび関連リソースを管理する方法

IBM MQ 制御コマンド、 IBM MQ Script Commands (MQSC)、Programmable Command Formats (PCF)、administrative REST API、 IBM MQ Console、および IBM MQ Explorer を使用して、 IBM MQ キュー・マネージャーおよび関連リソースを管理できます。 IBM i の場合は、 IBM i 制御言語を使用することもできます。 Windows の場合は、 Microsoft Cluster Service (MSCS) を使用することもできます。

IBM MQ の制御コマンド

Multi

制御コマンドは多数の IBM MQ 管理タスクを実行する方法を提供します。 AIX®, Linux®, and Windows の場合、これらのコマンドはシステム・コマンド行で発行します。 IBM i の場合は、Qshell 内でこれらのコマンドを発行します。 10 ページの『制御コマンドによる IBM MQ for Multiplatforms の管理』を参照してください。

IBM MQ スクリプト (MQSC) コマンド

MQSC コマンドを使用すると、キュー・マネージャー自体、キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、および認証情報オブジェクトなどのキュー・マネージャー・オブジェクトを管理できます。

ALW AIX, Linux, and Windows では、 **runmqsc** コマンド・プロンプトを開き、そのプロンプトからローカルまたはリモートのキュー・マネージャーに MQSC コマンドを発行します。 これは対話式に行うことも、ASCII テキスト・ファイルから一連のコマンドを実行することもできます。 詳細については、17 ページの『runmqsc での MQSC コマンドの対話式実行』および 22 ページの『runmqsc の下でテキスト・ファイルから MQSC コマンドを実行する』を参照してください。

IBM i IBM i では、スクリプト・ファイル内にコマンドのリストを作成し、 **STRMQMMQSC** コマンドを使用してそのファイルを実行します。 詳しくは、389 ページの『IBM i での MQSC コマンドを使用した管理』を参照してください。

z/OS z/OS では、コマンドに応じていくつかのソースから MQSC コマンドを発行できます。 詳しくは、449 ページの『Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS』を参照してください。

プログラマブル・コマンド・フォーマット (PCF)

プログラマブル・コマンド・フォーマット (PCF) では、ネットワーク内のプログラムと PCF 対応のキュー・マネージャーとの間で交換できるコマンド・メッセージと応答メッセージが定義されています。 システム管理アプリケーション・プログラムで、IBM MQ オブジェクト (認証情報オブジェクト、チャネル、チャネル・リスナー、名前リスト、プロセス定義、キュー・マネージャー、キュー、サービス、ストレージ・クラス) を管理するために PCF コマンドを使用できます。 ネットワーク内の 1 つのポイントからアプリケーションを実行し、ローカル・キュー・マネージャーを使用して、キュー・マネージャー (ローカルまたはリモート) との間でコマンド情報と応答情報をやり取りすることもできます。

PCF の詳細については、25 ページの『IBM MQ プログラマブル・コマンド・フォーマットの概要』を参照してください。

PCF の定義や、コマンドと応答の構造については、『プログラマブル・コマンド・フォーマットのリファレンス』を参照してください。

administrative REST API

administrative REST API は、IBM MQ を管理するために使用できる RESTful インターフェースを提供します。 administrative REST API を使用するときは、IBM MQ オブジェクトを表す URL に対して HTTP メソッドを呼び出します。 例えば、以下の URL で HTTP メソッド GET を使用して、IBM MQ インストール済み環境に関する情報を要求することができます。

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

プログラミング言語の HTTP/REST 実装によって、 または cURL などのツールや REST クライアント・ブラウザー・アドオンを使用して、administrative REST API を使用することができます。

For more information, see The administrative REST API

IBM MQ Console

IBM MQ Console を使用して、Web ブラウザーから IBM MQ を管理できます。

詳細については、90 ページの『IBM MQ Console を使用した管理』を参照してください。

IBM MQ Explorer

Windows Linux

IBM MQ Explorer を使用して、以下の操作を実行できます。

- キュー・マネージャー、キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、クラスターなど、さまざまなリソースの定義と管理。
- ローカル・キュー・マネージャーとその関連プロセスの始動/停止
- 使用ワークステーション上または他のワークステーションからの、キュー・マネージャーとその関連オブジェクトの表示
- キュー・マネージャー、クラスター、およびチャネルの状況の確認
- キューの状況からの、特定のキューをオープンさせるアプリケーション、ユーザー、またはチャネルの確認

Windows および Linux for x86-64 システムでは、システム・メニューまたは `MQExplorer` 実行可能ファイルを使用して IBM MQ Explorer を開始できます。

Linux Linux では、IBM MQ Explorer を正常に開始するために、ファイルをホーム・ディレクトリーに書き込めることと、ホーム・ディレクトリーが存在していることが必要です。

詳細については、118 ページの『IBM MQ Explorer を使用した管理』を参照してください。

IBM MQ Explorer を使用して、 z/OS を含む他のプラットフォーム上のリモート・キュー・マネージャーを管理することができます。

IBM MQ 9.3.0 以降、IBM MQ Explorer は IBM MQ インストール・パッケージから削除されました。 これは別個のダウンロードとして引き続き入手可能であり、スタンドアロン IBM MQ Explorer ダウンロード (Fix Central から入手可能) からインストールできます。 詳しくは、 Linux および Windows でのスタンドアロン・アプリケーションとしての IBM MQ Explorer のインストールおよびアンインストールを参照してください。

IBM i の制御言語 (CL)

IBM i

これは、 IBM MQ for IBM i に対して管理コマンドを発行するための推奨される方法です。 これらのコマンドは、コマンド・ラインから実行できます。あるいは、制御言語プログラムを作成することも可能です。 これらのコマンドの機能は、PCF コマンドの機能とよく似ていますが、形式が違います。 CL コマンドはサーバー専用に設計されており、CL 応答は人間が理解できます。 一方、PCF コマンドはプラットフォームに関係なく、コマンドと応答の形式は、いずれもプログラムで使用されます。

IBM i 制御言語 (CL) の詳細については、 375 ページの『CL コマンドを使用した IBM MQ for IBM i の管理』 および IBM MQ for IBM i CL コマンドを参照してください。

Microsoft Cluster Service (MSCS)

Windows

Microsoft Cluster Service (MSCS) を使用すると、サーバーをクラスター に接続して、データおよびアプリケーションにさらに高い可用性を提供し、システムの管理を容易にすることができます。 MSCS は、サーバーまたはアプリケーション障害を自動的に検出し、リカバリーすることができます。

MSCS の文脈での「クラスター」を、IBM MQ クラスターと混同しないことが重要です。 違いは次のとおりです。

IBM MQ クラスター
これらは、1 つ以上のコンピューター上にある複数のキュー・マネージャーのグループで、自動相互接続を提供し、グループ間でロード・バランシングと冗長度が適切になるようにキューを共有できます。

MSCS クラスター
これらは、相互接続されたコンピューターのグループで、いずれかのコンピューターに障害が起きたときに、MSCS によってフェイルオーバー が実行され、障害が起きたコンピューターからアプリケーションの状態データがクラスター内の別のコンピューターへ転送され、そこで操作が再開されるように構成されています。

Microsoft クラスター・サービス (MSCS) のサポート は、 MSCS を使用するように IBM MQ for Windows システムを構成する方法に関する詳細情報を提供します。

関連タスク

12 ページの『MQSC コマンドを使用した IBM MQ の管理』
MQSC コマンドを使用すると、キュー・マネージャー自体、キュー、チャネル、キュー、プロセス定義、チャネル、クライアント接続チャネル、リスナー、サービス、名前リスト、クラスター、および認証情報オブジェクトなどのキュー・マネージャー・オブジェクトを管理できます。 MQSC コマンドは、すべてのプラットフォームで使用可能です。

関連資料

管理に関する参照情報

Multi 制御コマンドによる IBM MQ for Multiplatforms の管理

制御コマンドは多数の IBM MQ 管理タスクを実行する方法を提供します。 AIX、 Linux、および Windows の場合、これらのコマンドはシステム・コマンド行で発行します。 IBM i の場合は、Qshell 内でこれらのコマンドを発行します。

始める前に

キュー・マネージャーで作動する制御コマンドを使用する場合は、操作対象のキュー・マネージャーと関連付けられたインストール済み環境からコマンドを実行する必要があります。

CHCKLOCL(REQUIRED) で接続認証を使用するように構成されたキュー・マネージャーで作動する制御コマンドを使用するときに、接続の失敗が見られる場合、以下のいずれかを実行します。

- 制御コマンドで使用できる場合は、ユーザー ID とパスワードを指定します。
- 制御コマンドの MQSC に相当するものが存在する場合は、それらを使用します。
- 接続できない制御コマンドを実行する必要があるときに、-ns オプションを使用してキュー・マネージャーを開始します。

注 : 異なるプラットフォームでは、異なる順序で入力されたコマンド引数を受け入れることができます。 特に、 Linux で動作するコマンドは、他のプラットフォームでは動作しない可能性があることを意味します。 このため、構文図で指定されているように、常に引数を入力する必要があります。

制御コマンドの完全なリストについては、 IBM MQ 制御コマンド・リファレンスを参照してください。

手順

- Linux AIX

  AIX and Linux システムで制御コマンドを使用します。

  IBM MQ for AIX or Linux システムでは、 制御コマンドをシェル・ウィンドウに入力します。

  ほとんどの制御コマンドは、制御コマンドを発行するために、mqm グループのメンバーであるユーザー ID を使用する必要があります。 これについて詳しくは、AIX, Linux, and Windows 上の IBM MQ を管理する権限を参照してください。 さらに、環境固有の情報にも注意してください。 お客様の企業が使用する 1 つ以上のプラットフォームに対応します。

  UNIX and Linux 環境では、制御コマンドは、コマンド名、フラグ、引数を含め大/小文字が区別されます。 例えば、次のコマンドを入力するとします。

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

  - コマンド名は CRTMQM ではなく、crtmqm でなければなりません。
  - フラグは -U ではなく -u でなければなりません。
  - 送達不能キューの名前は SYSTEM.DEAD.LETTER.QUEUE になります。
  - 引数は、jupiter.queue.manager と指定され、JUPITER.queue.manager とは区別されます。

  コマンドは、例に倣って正確に入力してください。

- Windows

  Windows システムで制御コマンドを使用します。

  IBM MQ for Windows では、制御コマンドをコマンド・プロンプトに入力します。

  ほとんどの制御コマンドは、制御コマンドを発行するために、mqm グループのメンバーであるユーザー ID を使用する必要があります。 これについて詳しくは、AIX, Linux, and Windows 上の IBM MQ を管理する権限を参照してください。 さらに、環境固有の情報にも注意してください。 お客様の企業が使用する 1 つ以上のプラットフォームに対応します。

  制御コマンドとそれらのフラグには大/小文字の区別がありませんが、それらのコマンドに対する引数 (キュー名やキュー・マネージャー名など) には大/小文字の区別があります。

  例えば、次のコマンドを入力するとします。

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

  - 入力するコマンド名は、大文字、小文字、あるいは大文字と小文字が混在していても構いません。 例えば、crtmqm、CRTMQM、または CRTmqm のいずれでも構いません。
  - 入力するフラグは、-u、-U、/u、または /U のいずれでも構いません。
  - SYSTEM.DEAD.LETTER.QUEUE と jupiter.queue.manager は表示どおりに入力する必要があります。

- IBM i

  IBM i システムで制御コマンドを使用します。

  IBM MQ for IBM i では、Qshell 環境から制御コマンドを実行します。 Qshell を使用するには、 IBM i コマンド行に STRQSH と入力します。 F3 を押すと、いつでも終了してコマンド行に戻ることができます。

IBM i では、いくつかの制御コマンドはサポートされていません。 例えば、IBM i システム上に IBM MQ のコピーを複数持つことはできないため、複数インストール・コマンドはサポートされません。 IBM i でサポートされないコマンドには、IBM MQ 制御コマンド・リファレンスの ALW のフラグが付きます。

関連資料
IBM MQ 制御コマンド・リファレンス

MQSC コマンドを使用した IBM MQ の管理

MQSC コマンドを使用すると、キュー・マネージャー自体、キュー、チャネル、キュー、プロセス定義、チャネル、クライアント接続チャネル、リスナー、サービス、名前リスト、クラスター、および認証情報オブジェクトなどのキュー・マネージャー・オブジェクトを管理できます。 MQSC コマンドは、すべてのプラットフォームで使用可能です。

このタスクについて

使用可能な MQSC コマンドについて詳しくは、 MQSC コマンド・リファレンスを参照してください。

MQSC コマンドを発行する方法は、ご使用のプラットフォームによって異なります。

- ALW AIX, Linux, and Windows では、 **runmqsc** コマンド・プロンプトから MQSC コマンドをキュー・マネージャーに発行します。 このコマンド・プロンプトは、いくつかの方法で使用できます。
  - キーボードから MQSC コマンドを対話式に発行する。 17 ページの『runmqsc での MQSC コマンドの対話式実行』 を参照してください。
  - ASCII テキスト・ファイルから MQSC コマンドを発行する。 22 ページの『runmqsc の下でテキスト・ファイルから MQSC コマンドを実行する』 を参照してください。
  - リモート・キュー・マネージャーで MQSC コマンドを発行する。 201 ページの『リモート・キュー・マネージャーに対する MQSC コマンドの発行』 を参照してください。
- IBM i IBM i では、スクリプト・ファイル内にコマンドのリストを作成し、 **STRMQMMQSC** コマンドを使用してそのファイルを実行します。 詳しくは、389 ページの『IBM i での MQSC コマンドを使用した管理』 を参照してください。
- z/OS z/OS では、コマンドに応じていくつかのソースから MQSC コマンドを発行できます。 詳しくは、449 ページの『Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS』 を参照してください。

手順

- 12 ページの『MQSC コマンド構文』
- 14 ページの『MQSC: 特殊文字および総称値』
- 17 ページの『runmqsc での MQSC コマンドの対話式実行』
- 22 ページの『runmqsc の下でテキスト・ファイルから MQSC コマンドを実行する』
- 23 ページの『開始時の MQSC スクリプトからの自動構成』

関連タスク
MQSC コマンドで起こった問題の解決
関連資料
runmqsc (MQSC コマンドの実行)

MQSC コマンド構文

MQSC コマンドを使用して、キュー・マネージャー・オブジェクトを管理できます。 MQSC コマンドは、すべてのプラットフォームで使用可能です。 コマンド構文の一部のエレメントは、プラットフォーム固有のものです。

パラメーターの順序

各コマンドは 1 次パラメーター (verb) で始めて、その後に 2 次パラメーター (noun) を続けます。 さらに、オブジェクトの名前または総称名があれば (ほとんどのコマンドで指定される)、その後に (括弧に入れて) 続けます。 その後に、パラメーターを任意の順序で指定できます。パラメーターが対応する値を取る場合は、関連するパラメーターの直後にその値を指定する必要があります。

注 : z/OS z/OS の場合は、必ずしも 2 次パラメーターを 2 番目に指定する必要はありません。

ブランクとコンマ

キーワード、括弧、および値は任意の数のブランクおよびコンマで区切ることができます。 構文図に示されているコンマは、どれも 1 つ以上のブランクに置き換えることができます。 z/OS の場合を除き、(1 次パラメーターの後にある) 各パラメーターでは、直前のパラメーターとの間に少なくとも 1 つのブランクが必要です。

コマンドの先頭または終わり、およびパラメーター、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。 例えば、次のコマンドは有効です。

```
ALTER QLOCAL ('Account' )      TRIGDPTH ( 1)
```

引用符の対の中に置かれたブランクには意味があります。

ブランクが許可されている場所に余分なコンマがあってもよく、それらはブランクと同じように扱われます (ただし、引用符で囲まれたストリング内にある場合を除きます)。

反復パラメーター

パラメーターの繰り返しは許可されていません。 REPLACE NOREPLACE のように、それ自身の "NO" バージョンによるパラメーターの繰り返しも許可されていません。

ストリングと単一引用符

ブランク、小文字、または特殊文字を含むストリングは、以下のいずれかが当てはまらない限り、単一引用符で囲む必要があります。

- 特殊文字は以下の 1 つ以上の文字である。
  - ピリオド (.)
  - スラッシュ (/)
  - 下線 (_)
  - パーセント記号 (%)
- z/OS IBM MQ for z/OS の操作および制御パネルからコマンドが発行される。
- ストリングは末尾がアスタリスクの総称値である。 (IBM i では、これらは単一引用符で囲む必要があります)
- ストリングは単一アスタリスクである。例えば TRACE(*) (IBM i では、これらは単一引用符で囲む必要があります)
- ストリングはコロンを含んだ範囲指定である。例えば CLASS(01:03)

ストリング自体に単一引用符が含まれている場合、その単一引用符は 2 つの単一引用符で表されます。

Multi マルチプラットフォームでは、文字を含まないストリング (つまり、間にスペースを入れない 2 つの単一引用符) は、単一引用符で囲まれたブランク・スペースとして解釈されます。つまり、(' ') と同じように解釈されます。 この例外は、使用されている属性が以下の属性のいずれかである場合、スペースのない 2 つの単一引用符がゼロ長ストリングとして解釈される場合です。

- TOPICSTR
- SUB

- USERDATA
- SELECTOR

z/OS z/OS では、単一引用符で囲まれたブランク・スペースを使用したい場合は、それを (' ')として入力する必要があります。 文字を含まないストリング ('') は、入力 ()と同じです。

MQCHARV タイプに基づくストリング属性 (SELECTOR や SUBUSERDATA など) の末尾ブランクは、有意と見なされます。つまり、'abc ' と 'abc' は同一ではありません。

空の括弧

左括弧の後に右括弧が続き、間に有意情報がなければ、特に明記されている場合を除いて、その左括弧は無効です。 例えば、次のストリングは無効です。

```
NAME ( )
```

小文字と大文字

キーワードに大/小文字の区別はありません。AltER、alter、および ALTER はすべて許容されます。

引用符で囲まれていない文字はすべて大文字に変換されます。

同義語

一部のパラメーターには同義語が定義されています。 例えば、DEF は常に DEFINE の同義語であるので、DEF QLOCAL は有効です。 しかし、同義語は単にストリングを最も短くしたものというわけではありません。DEFI は DEFINE の有効な同義語ではありません。

注 : DELETE パラメーターの同義語はありません。 これは、DEFINE の同義語である DEF を使用したために、誤ってオブジェクトを削除することのないようにするためです。

特殊文字

MQSC コマンドは、特定の意味を表すために特定の特殊文字を使用します。 これらの特殊文字とその使用法については、14 ページの『MQSC: 特殊文字および総称値』を参照してください。

関連タスク

MQSC コマンドで起こった問題の解決

関連資料

runmqsc (MQSC コマンドの実行)

MQSC: 特殊文字および総称値

一部の文字 (円記号 (¥) や二重引用符 ('') など) MQSC コマンドで使用する場合、文字には特殊な意味があります。 パラメーターで使用できる一部の特殊文字は、総称値を持つことができますが、正しく指定する必要があります。

円記号 (¥) と二重引用符 ('') の前 文字に \を使用します。つまり、テキストに \ または " を使用する場合は、 \\ または \" を入力します。

パラメーターに総称値を指定できる場合、常に終わりにはアスタリスクが入力されます (例: ABC*)。 総称値は「で始まるすべての値」を意味するので、ABC* は「ABC で始まるすべての値」を意味します。 値の中で引用符を必要とする文字を使用する場合は、'abc*' のように、引用符の内側にアスタリスクを入れる必要があります。 アスタリスクは、値の最後または値の唯一の文字にする必要があります。

疑問符 (?) およびコロン (:) は総称値には指定できません。

フィールド内で (例えば、記述の一部で) これらの特殊文字を使用するときは、ストリングの全体を単一引用符で囲む必要があります。

表 1. 特別な意味を持つ文字の説明	
文字	**説明**
	ブランクは区切り記号として使われます。 複数のブランクは、アポストロフィ (')で囲まれたストリングを除き、単一のブランクに相当します。 これらのストリング属性内の末尾ブランクは、MQCHARV タイプに基づくものとして、重大として扱われます。
,	コンマは区切り記号として使われます。 コンマは、複数個でも 1 個と同等です。ただし、アポストロフィ (') で囲まれたストリングの内部にある場合を除きます。
'	アポストロフィは、ストリングの始まりまたは終わりを表します。 IBM MQ では、引用符で囲まれた文字はすべて入力したままになります。 ストリングの長さを計算する場合、そのストリングを囲んでいるアポストロフィは長さに含まれません。
''	ストリングの内部に単一引用符があると、IBM MQ はストリングの長さを計算するときにそれを 1 個の文字として扱い、ストリングの終わりとは見なしません。
=	z/OS z/OS での等号 は、コンマまたはブランクで終了するパラメーター値の始まりを表します。
(	左括弧は、パラメーター値または値リストの始まりを表します。
)	右括弧は、パラメーター値または値リストの終わりを表します。
:	コロンは範囲を表し、両端を含むことを意味します。 例えば、(1:5) は (1,2,3,4,5) を意味します。 この表記は、**TRACE** コマンドでのみ用います。
*	アスタリスクは全部を意味します。 例えば、**DISPLAY TRACE (*)** は、すべてのトレースの表示を意味し、**DISPLAY QUEUE (PAY*)** は、名前の最初の部分が PAY であるすべてのキューの表示を意味します。

MQSC 入力ファイルの構文

長いコマンドがある場合、または特定の一連のコマンドを繰り返し使用する場合は、入力ファイルを使用して MQSC コマンドを発行できます。 入力ファイルの内容は、このトピックで説明する構文に従う必要があります。

概要

MQSC コマンドは、標準入力装置(`stdin` とも呼ばれる) を介して入力されます。 通常、これはキーボードですが、入力ファイルからの入力を指定することができます。

この入力ファイルは、以下のプラットフォーム固有のツールのいずれかで使用できます。

- ALW AIX, Linux, and Windows では **runmqsc** コマンドを使用します。 22 ページの『runmqsc の下でテキスト・ファイルから MQSC コマンドを実行する』 を参照してください。
- IBM i IBM i では **STRMQM** コマンドを使用します。 389 ページの『IBM i での MQSC コマンドを使用した管理』 を参照してください。
- z/OS z/OS で、CSQINP1、CSQINP2、および CSQINPX 初期化データ・セットまたは CSQUTIL バッチ・ユーティリティーを使用する場合 449 ページの『Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS』 を参照してください。

構文

MQSC 入力ファイルの構文:

- IBM MQ 環境間での移植性を考えて、MQSC コマンド・ファイルの行の長さは、最大 72 文字に制限します。
- 各コマンドは新しい行から開始しなければなりません。

- 行頭にアスタリスク (*) が付いた行は無視されます。 この方法はファイルにコメントを挿入するときに使用できます。
- ブランク行は無視されます。
- 正符号 (+) は、コマンドが次の行の最初の非ブランク文字から継続されることを示します。 + を使用してコマンドを続ける場合は、次のパラメーターの前に少なくとも 1 つのブランクを忘れずに残してください (z/OS ではその必要はありません)。 コメントまたはブランク行は、コマンドを単一のストリングに再アセンブルするときにすべて廃棄されます。
- 負符号 (-)。これは、コマンドが次の行の行頭から続くこと示します。 コメントまたはブランク行は、コマンドを単一のストリングに再アセンブルするときにすべて廃棄されます。
- Escape PCF (プログラム式コマンド形式) コマンド内に組み込む MQSC コマンドを、正符号や負符号で続けることはできません。 コマンド全体を 1 つの Escape コマンド内に含める必要があります。 PCF コマンドの詳細は、25 ページの『IBM MQ プログラマブル・コマンド・フォーマットの概要』を参照してください。
- マルチプラットフォーム の場合、および z/OS では、CSQUTIL バッチ・ユーティリティー・プログラムから実行するコマンドの場合、直前の行末に正符号 (+) を入力していても、セミコロン文字 (;) を使用してコマンドを終了することができます。
- 行をキーボードの制御文字で終了することはできません (例えば、タブ記号など)。
- テキスト・ファイルから `stdin` をリダイレクトすることによってクライアント・モードで **runmqsc** コマンドを実行し、資格情報を指定するために **-u** フラグを指定した場合、 **runmqsc** コマンドはパスワードの入力を求めるプロンプトを出さず、代わりに `stdin` からパスワードを読み取ります。 `stdin` を介して提供されるデータの最初の行がパスワードであることを確認する必要があります。 これを行うには、「echo」や「cat」などのコマンド・ライン・ツールを使用し、パスワードに続けて MQSC スクリプトを **runmqsc** コマンド `stdin` に渡します。
- Windows Windows で、ポンド記号 (£) や論理否定 (¬) などの特定の特殊文字がコマンド・スクリプト内で使用されている場合 (オブジェクト記述の一部としてなど)、**DISPLAY QLOCAL** などのコマンドからの出力では、それらの文字は別の文字で表示されます。

12 ページの『MQSC コマンド構文』も参照してください。

例

以下の例は、コマンド **DEFINE QLOCAL** を示す MQSC コマンド・ファイルからの抜粋です。

```
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE +
DESCR(' ') +
PUT(ENABLED) +
DEFPRTY(0) +
DEFPSIST(NO) +
GET(ENABLED) +
MAXDEPTH(5000) +
MAXMSGL(1024) +
DEFSOPT(SHARED) +
NOHARDENBO +
USAGE(NORMAL) +
NOTRIGGER;
```

図 *1. MQSC* コマンド・ファイルからの抽出

runmqsc コマンドが完了すると、レポートが返されます。 次の例は、レポートからの抜粋です。

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

図 *2. MQSC* コマンド・レポート・ファイルからの抽出

MQSC コマンド・ファイルの例を使用して、テキスト・ファイルを作成することもできます。

amqscos0.tst
サンプル・プログラムが使用するオブジェクトの定義

amqscic0.tst
CICS® トランザクション用のキューの定義

Linux AIX AIX and Linux では、これらのファイルはディレクトリー *MQ_INSTALLATION_PATH*/samp にあります。 *MQ_INSTALLATION_PATH* は、IBM 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。

Windows Windows では、これらのファイルはディレクトリー *MQ_INSTALLATION_PATH*\tools\mqsc\samples にあります。 *MQ_INSTALLATION_PATH* は、IBM 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。

ALW runmqsc での MQSC コマンドの対話式実行

AIX, Linux, and Windows では、 **runmqsc** コマンド・プロンプトを使用して、MQSC コマンドをキュー・マネージャーに対話式に発行できます。 対話式実行は、特にクイック・テストに適しています。

始める前に

runmqsc コマンドは、 作業対象のキュー・マネージャーに関連付けられているインストール環境から 使用する必要があります。 dspmq -o installation コマンドを使用して、どのインストールがキュー・マネージャーと関連しているかを調べることができます。

MQPROMPT 環境変数を使用して任意のプロンプトを設定することにより、MQSC 環境にいることを簡単に確認したり、現在の環境の詳細を確認したりすることができます。 詳しくは、20 ページの『MQSC コマンド・プロンプトの設定』を参照してください。

Linux AIX AIX and Linux プラットフォームで MQSC コマンドを対話式に実行する場合、 **runmqsc** コマンド・プロンプトでは、追加のコマンド・ライン・エディター機能もサポートされます。 21 ページの『runmqsc のコマンド再呼び出しと完了、および Emacs コマンド・キーの使用可能化』を参照してください。

このタスクについて

runmqsc コマンドは、MQSC コマンドを発行できるコマンド・プロンプトを開くために使用します。 これらのコマンドとその構文については、 MQSC コマンド・リファレンスで説明されています。

このタスクの説明に従って **runmqsc** コマンド・プロンプトを開始する場合は、コマンドに設定されているフラグに応じて、以下の 3 つのモードのいずれかで実行するようにプロンプトを設定します。

- 検証モード。このモードでは、MQSC コマンドはローカル・キュー・マネージャー上で検証されますが、実行されません。
- 直接モード。このモードでは、MQSC コマンドはローカル・キュー・マネージャー上で実行されます。
- 間接モード。このモードでは、MQSC コマンドはリモート・キュー・マネージャー上で実行されます。

以下の手順では、プロンプトを直接モードで実行するように設定します。 その他のオプションについては、メイン・ステップの後の例で説明します。

手順

1. コマンド・ウィンドウまたはシェルを開き、以下のコマンドを入力します。

   ```
   runmqsc QMgrName
   ```

   ここで、*QMgrName* は、MQSC コマンドを処理するキュー・マネージャーの名前を指定します。 デフォルトのキュー・マネージャーで MQSC コマンドを処理するには、*QMgrName* をブランクにしておきます。
2. 必要に応じて任意の MQSC コマンドを入力します。 例えば、ORANGE.LOCAL.QUEUE というローカル・キューを作成するには、次のコマンドを入力します。

   ```
   DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
   ```

   パラメーターが多すぎて 1 行に収まらないコマンドの場合、コマンドが次の行に続くことを示すためには連結文字を使用する必要があります。

   - 負符号 (-) は、コマンドが次の行の先頭に続くことを示します。
   - 正符号 (+) は、コマンドが次の行の最初の非ブランク文字から継続されることを示します。

   コマンド入力は、継続文字ではない非ブランク行の最後の文字で終了します。 セミコロン (;) を入力して、明示的にコマンド入力を終了することもできます。
3. 次のコマンドを入力して、MQSC コマンドでの作業を終了します。

   ```
   end
   ```

   あるいは、 **exit** コマンド、 **quit** コマンド、またはご使用のオペレーティング・システムの EOF 文字を使用することもできます。

タスクの結果

MQSC コマンドを発行すると、そのアクションを確認するオペレーター・メッセージ、または操作エラーがあったことを示すオペレーター・メッセージがキュー・マネージャーから戻されます。 例えば、次のメッセージは、キューが作成されたことを確認しています。

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

以下のメッセージは、構文エラーがあったことを示すものです。

```
AMQ8405: Syntax error detected at or near end of command segment below:-
AMQ8426: Valid MQSC commands are:

ALTER
CLEAR
DEFINE
DELETE
DISPLAY
END
PING
```

```
REFRESH
RESET
RESOLVE
RESUME
START
STOP
SUSPEND
4 : end
```

これらのメッセージは、デフォルトで表示される標準出力装置に送信されます。 コマンドを正しく入力しなかった場合は、コマンドの参照情報で正しい構文を調べてください。 MQSC コマンド・リファレンスを参照してください。

例

以下は、前のステップで使用された `runmqsc` *QMgrName* コマンドのバリアントです。 これらのバリアントは、 **runmqsc** コマンド・プロンプトのさまざまな構成を作成します。

- 以下のコマンドは、コマンド・フィルターを使用して、単一の MQSC コマンドを MQSC インタープリターに渡します。

  On Windows:

  ```
  echo display chstatus(*) | runmqsc QMname
  ```

  On Linux:

  ```
  echo "display chstatus(*)" | runmqsc QMname
  ```

- 以下のコマンドはキュー・マネージャー名を指定しないため、MQSC コマンドはデフォルトのキュー・マネージャーで処理されます。

  ```
  runmqsc
  ```

- このコマンドは、コマンドの実行依頼に QMLOCAL を使用して、QMREMOTE キュー・マネージャーにコマンドを実行依頼します。

  ```
  runmqsc -w 30 -m QMLOCAL QMREMOTE
  ```

- このコマンドは、コマンドを実行せずに、ローカル・キュー・マネージャー上でコマンド構文が正しいことを検証します。 検査されるコマンドは、入力ファイル `myprog.in` から読み取られることに注意してください。

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

  入出力ファイルの処理について詳しくは、22 ページの『runmqsc の下でテキスト・ファイルから MQSC コマンドを実行する』を参照してください。

次のタスク

runmqsc コマンド構文、オプション・パラメーター、および戻りコードについて詳しくは、runmqsc (MQSC コマンドの実行)を参照してください。

関連タスク

22 ページの『runmqsc の下でテキスト・ファイルから MQSC コマンドを実行する』
長いコマンドがある場合、または特定のコマンド・シーケンスを繰り返し使用する場合は、テキスト・ファイルを使用して MQSC コマンドを発行できます。 テキスト・ファイルから `stdin` をリダイレクトできます。 出力をファイルにリダイレクトすることもできます。

関連資料

MQSC コマンド・リファレンス

ALW MQSC コマンド・プロンプトの設定

AIX, Linux, and Windows では、 **MQPROMPT** 環境変数を使用して、 **runmqsc** コマンドの実行時に表示されるプロンプトを設定します。 これにより、MQSC 環境にいることを確認したり、現在の環境の詳細を確認したりすることが容易になります。

このタスクについて

runmqsc コマンドの実行時に表示されるプロンプトを設定できます。 プロンプトは、 **runmqsc** コマンドが対話式に実行されるときと、入力がファイルまたは標準入力装置 (stdin) から **runmqsc** にリダイレクトされるときの両方に挿入されます。

コマンド・プロンプトにプレーン・テキストを含めることができます。また、 IBM MQ サービス・オブジェクト定義と同じ方法で +VARNAME+ 表記を使用して環境変数を挿入することもできます。 詳しくは、186 ページの『サービス定義での置き換え可能挿入の使用』を参照してください。

IBM MQ では、他にもいくつかの置き換え可能な挿入が提供されています。これらを以下の表で説明します。

置き換え可能な挿入	説明
MQ_HOST_NAME	システムのホスト名
MQ_FILE_SEP	プラットフォーム固有のファイル分離文字があります。 • Linux AIX AIX and Linux システムの場合、 **MQ_FILE_SEP** は /です。 • Windows Windows システムの場合、**MQ_FILE_SEP** の場所は \ です。
MQ_PATH_SEP	プラットフォーム固有のパス分離文字があります。 • Linux AIX AIX and Linux システムの場合、 **MQ_PATH_SEP** は :です。 • Windows Windows システムの場合、**MQ_PATH_SEP** の場所は ; です。
MQ_DATE_TIME	固定の YYYY-MM-DD hh:mm:ss.SSS 形式のローカル・システム日時。以下に例を示します。 `2020-12-25 17:41:37.408`

注:

- MQ 置き換え可能挿入値は、 **runmqsc** コマンドが関連付けられている IBM MQ インストール済み環境およびホスト・システムに関連します。
- **MQPROMPT** は、挿入が展開されるときに最大 256 文字に制限されます。 この値を **MQPROMPT** で拡張すると、 **MQPROMPT** ストリング全体が拡張されずに切り捨てられます。

例

以下の例では、プロンプトを MQSC に設定します。

- Linux AIX

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

- Windows

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

AIX 以下の例では、 AIX システムで **MQPROMPT** 変数を設定します。 プロンプトは、ユーザー名 (関連付けられたシステム環境変数から取得)、キュー・マネージャー名、および IBM MQ ホスト名 (IBM MQ 置き換え可能挿入から取得) を表示するように設定されます。

```
sh> export MQPROMPT="+USER+ @ +QMNAME+ @ +MQ_HOST_NAME+> "
sh> runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2024.
Starting MQSC for queue manager MY.QMGR.

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

```
C:\ > set "MQPROMPT=+USERNAME+ @ +QMNAME+ @ +MQ_HOST_NAME+> "
C:\ > runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2024.
Starting MQSC for queue manager MY.QMGR.

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

以下の例では、 MQ 置き換え可能挿入から取られたタイム・スタンプを上記の **MQPROMPT** の例に追加します。

```
sh> export MQPROMPT="+MQ_DATE_TIME+ +USER+ @ +QMNAME+ @ +MQ_HOST_NAME+> "
sh> runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2024.
Starting MQSC for queue manager MY.QMGR.

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

```
C:\ > set "MQPROMPT=+MQ_DATE_TIME+ +USERNAME+ @ +QMNAME+ @ +MQ_HOST_NAME+> "
C:\ > runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2024.
Starting MQSC for queue manager MY.QMGR.

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

Linux AIX runmqsc のコマンド再呼び出しと完了、および Emacs コマンド・キーの使用可能化

AIX and Linux で **runmqsc** コマンド・プロンプトを使用して、コマンド再呼び出し、コマンド完了、および Emacs コマンド・キーを使用可能にします。

このタスクについて

AIX and Linux システムでは、 **runmqsc** コマンド・プロンプトから、以下の追加のコマンド・ライン・エディター機能を使用可能にすることができます。

- 上下矢印キーを使用した、以前に入力されたコマンドの再呼び出し
- タブ・キーおよびスペース・キーを使用した、コマンドの次のキーワードの自動完了
- Emacs コマンド・キー、または類似のコマンド・キー機能

これらの機能を使用するには、curses ライブラリーをインストールする必要があります。 curses ライブラリーがシステムにインストールされていない場合、 **runmqsc** プロンプトにはコマンド・ライン・エディター機能がなく、 **runmqsc** コマンド・プロンプトの開始時にメッセージが表示されます。 インストールする curses ライブラリーの名前は、UNIX プラットフォームによって異なります。

- AIX AIX では、`curses` をインストールします。
- Linux Linux では、`ncurses` をインストールします。

手順

- ncurses または curses をインストールします。

  注: 以下の例では、 Linux の手順を使用しています。

  以下のコマンドを実行して、既存の ncurses パッケージを見つけます。

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

  必要な ncurses パッケージは以下のとおりです。

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

  以下のコマンドを実行して、上記のテキストにリストされている必要な ncurses パッケージをすべてインストールできます。

```
yum install ncurses*
```

- Emacs キー・バインディングをカスタマイズします。

  コマンドにバインドされるキーをカスタマイズできます。 例えば、デフォルトの Emacs キー・バインディングの代わりに vi バインディングにキーをバインドできます。

  このキーは、ホーム・ディレクトリーに保管されている .editrc ファイルを編集することによってカスタマイズされます。 詳しくは、 FreeBSD man ページの editrc を参照してください。

- コマンド再呼び出し、コマンド完了、および Emacs コマンド・キーを使用不可にします。

  これを行うには、環境変数 **MQ_OVERRIDE_LIBEDIT_LOAD** を TRUE に設定します。

  この環境変数は、 **runmqsc** コマンド・プロンプトに以下の通知メッセージが表示される場合の回避策として使用できます。

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

ALW runmqsc の下でテキスト・ファイルから MQSC コマンドを実行する

長いコマンドがある場合、または特定のコマンド・シーケンスを繰り返し使用する場合は、テキスト・ファイルを使用して MQSC コマンドを発行できます。 テキスト・ファイルから stdin をリダイレクトできます。 出力をファイルにリダイレクトすることもできます。

始める前に

このタスクは、実行する MQSC コマンドを含むテキスト・ファイルが作成済みであることを前提としています。 これらのファイルの詳細な構文と例については、15 ページの『MQSC 入力ファイルの構文』を参照してください。

MQPROMPT 環境変数を使用して、MQSC コマンド・プロンプトを任意のプロンプトに設定できます。 詳しくは、20 ページの『MQSC コマンド・プロンプトの設定』を参照してください。

このタスクについて

runmqsc コマンドの入力は標準入力装置 (stdin とも呼ばれる) から取り込まれます。 通常、これはキーボードですが、入力がシリアル・ポートまたはファイルからのものであることを指定できます。

runmqsc コマンドの出力は標準出力装置 (stdout とも呼ばれる) に出力されます。 通常、これは表示装置ですが、シリアル・ポートやファイルなどにリダイレクトすることができます。

手順

1. ローカル・キュー・マネージャーで、コマンドを実行せずに、ファイル内のコマンド構文が正しいことを確認してください。

   以下のいずれかのオプションとともに、 **runmqsc** コマンドで **-v** フラグを使用します。

   - 入力テキスト・ファイル名を識別するには、 **-f** オプションを使用します。 以下に例を示します。

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

   コマンドの検証時にリモート・キュー・マネージャーを指定することはできません。 つまり、 **-w** フラグは指定できません。

   戻されるレポートは、17 ページの図 2 に示されているものと同様です。
2. コマンド構文が正しい場合は、**-v** フラグを削除してから、**runmqsc** コマンドを再実行してください。

   リモート・キュー・マネージャーを指定できるようになったことに注意してください。

   - 例えば、以下のコマンドを実行します。

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

   16 ページの図 1 は、 `myprog.in` などのコマンド・ファイルからの抽出を示しています。 17 ページの図 2 は、 `results.out` などのレポート・ファイルからの出力の対応する抽出を示しています。

次のタスク

runmqsc コマンド構文、オプション・パラメーター、および戻りコードについて詳しくは、runmqsc (MQSC コマンドの実行)を参照してください。

関連タスク

20 ページの『MQSC コマンド・プロンプトの設定』
AIX, Linux, and Windows では、 **MQPROMPT** 環境変数を使用して、 **runmqsc** コマンドの実行時に表示されるプロンプトを設定します。 これにより、MQSC 環境にいることを確認したり、現在の環境の詳細を確認したりすることが容易になります。

17 ページの『runmqsc での MQSC コマンドの対話式実行』
AIX, Linux, and Windows では、 **runmqsc** コマンド・プロンプトを使用して、MQSC コマンドをキュー・マネージャーに対話式に発行できます。 対話式実行は、特にクイック・テストに適しています。

関連資料

MQSC コマンド・リファレンス

Multi 開始時の MQSC スクリプトからの自動構成

キュー・マネージャーの開始ごとに、MQSC スクリプトまたは MQSC スクリプトのセットの内容を自動的に適用するようにキュー・マネージャーを構成することができます。

この機能を使用することにより、変更可能で、次のキュー・マネージャーの再始動時に自動的に再生可能な構成を保持できます。 例えば、マウントされたドライブ上に 1 つ以上のスクリプトがある場合、開始時にすべてのキュー・マネージャーに対して最新バージョンが適用される集中構成を使用できます。

これが役立つ具体的なシナリオは、クラスター内のすべてのキュー・マネージャーで、すべてが適用される単一の構成セットを使用することにより、均一クラスターに同じ定義が含まれるようにすることです。 この例については、均一クラスターの新規作成を参照してください。

始める前に

以下を使用できます。

1. 単一スクリプト。MQSC コマンドを使用してテキスト・ファイルを作成します。
2. MQSC スクリプトのセット:

- 構成が存在するディレクトリーを指定する。
- そのディレクトリーで、それぞれが拡張子 .mqsc を持つファイルを作成します。例えば、queues.mqsc のようにします。

このスクリプトはキュー・マネージャーの開始ごとに再適用されるため、それらのコマンドは再生可能であることが重要です。 例えば、**DEFINE** コマンドには **REPLACE** ストリングが含まれている必要があります。そうでない場合、オブジェクトが既に存在するため、コマンドは 2 番目のキュー・マネージャーの開始時に失敗として表示されます。

MQSC スクリプトでは、接頭部 * が付いた行はコメントとして扱われます。

MQSC スクリプトの自動構成の有効化

重要 : MQTT タイプのチャネルに対してコマンドを発行してはなりません。これらのコマンドは、始動時の自動構成ではサポートされないためです。

新しいキュー・マネージャーを構成するには、 **crtmqm** コマンドに対して **-ic** フラグを使用し、特定のファイルまたはディレクトリーを指定します。 指定された値は、qm.ini ファイルの AutoConfig スタンザの下に、属性 **MQSCConfig** として保管されます。

有効なファイルまたはディレクトリーを指す AutoConfig スタンザ属性の **MQSCConfig** を追加することにより、自動 MQSC 構成を有効化するように既存のキュー・マネージャーを構成できます。 以下に例を示します。

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

自動構成の仕組み

キュー・マネージャーの始動時に、 **AutoConfig** スタンザ属性 **MQSCConfig** によって識別される構成は、有効な構文であることを確認するために **runmqsc** 妥当性検査を通過し、キュー・マネージャーのデータ・ツリー内の autocfg ディレクトリーに単一ファイル cached.mqsc として保管されます。

ディレクトリー内の複数のファイルが処理されるとき、それらはアルファベット順に処理され、MQSC の終了または終了コマンドが含まれている場合は、そのファイルの残りの内容がスキップされます。

キュー・マネージャーの最初の開始時に、ファイル、ディレクトリー、または無効な MQSC 構文を持つファイルを読み取ることができない場合、そのキュー・マネージャーは開始しません。このとき、コンソールとキュー・マネージャーのエラー・ログの両方に該当するエラー・メッセージが表示されます。

後続の再始動で、指示されたファイルまたはディレクトリーが読み取り不能であるか、無効な MQSC 構文を含んでいる場合、以前にキャッシュされたファイルが使用されます。このことは、キュー・マネージャーのエラー・ログに書き込まれるメッセージによって強調表示されます。

cached.mqsc の内容がキュー・マネージャーに適用される時点で、すべての MQSC コマンドが適用されると、キュー・マネージャーはアプリケーションから接続できるようになります。 適用される構成の **runmqsc** ログは、キュー・マネージャーのエラー・ディレクトリーに AMQ*qmgr*.mqsc という名前のファイルとして保管されます。

さらに、正常に完了しなかった MQSC コマンドはキュー・マネージャーのエラー・ログに記録され、コマンドが失敗した理由が示されます。

PCF コマンドによる IBM MQ 管理の自動化

タスクのモニターおよび一部の管理を自動化することがインストールに役立つと判断する場合があります。 プログラム式コマンド形式 (PCF) コマンドを使用して、ローカル・キュー・マネージャーおよびリモート・キュー・マネージャー 両方の管理タスクを自動化することができます。 このセクションでは、IBM MQ オブジェクトの管理経験のあることが前提となります。

PCF コマンド

IBM MQ プログラマブル・コマンド・フォーマット (PCF) コマンドは、管理タスクを管理プログラムに組み込むために使用できます。 このようにすると、プログラムから、キュー・マネージャー・オブジェクト (キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、および認証情報オブジェクト) を操作できるだけでなく、キュー・マネージャー自体も操作できます。

PCF コマンドは、MQSC コマンドが対象とする範囲と同じ機能範囲をカバーします。 単一のノードから、ネットワーク内の任意のキュー・マネージャーへ PCF コマンドを発行するプログラムを作成することができます。 これにより、管理タスクを中央集中方式にすると同時に自動化することができます。

各 PCF コマンドは、IBM MQ メッセージのアプリケーション・データ部分に組み込まれたデータ構造です。各コマンドは、他のメッセージの場合と同様に、MQI 機能 MQPUT を使用してターゲット・キュー・マネージャーに送られます。 メッセージを受信するキュー・マネージャーでコマンド・サーバーが稼働していれば、そのコマンド・サーバーは、そのコマンドをコマンド・メッセージと解釈して実行します。 応答を入手するには、アプリケーションが MQGET 呼び出しを発行します。応答データが別のデータ構造に戻されます。 次に、アプリケーションはその応答を処理し、その応答に応じてアクションを実行します。

注 : MQSC コマンドとは異なり、PCF コマンドおよびそれらの応答は、読み取り可能なテキスト形式ではありません。

次に、PCF コマンド・メッセージを作成するために必要のある事項をいくつか簡単に示します。

メッセージ記述子
標準 IBM MQ メッセージ記述子です。これを使用して以下を行います。

- メッセージ・タイプ (*MsqType*) には、MQMT_REQUEST を指定します。
- メッセージ形式 (*Format*) には、MQFMT_ADMIN を指定します。

アプリケーション・データ
これには、PCF ヘッダーを含む PCF メッセージが入ります。このメッセージの中で、以下のように指定します。

- PCF メッセージ・タイプ (*Type*) には MQCFT_COMMAND を指定します。
- コマンド ID には、*Change Queue* (MQCMD_CHANGE_Q) などのコマンドを指定します。

PCF データ構造とその実装方法の詳細説明については、25 ページの『IBM MQ プログラマブル・コマンド・フォーマットの概要』を参照してください。

PCF オブジェクトの属性

PCF でのオブジェクト属性は、MQSC コマンドのように 8 文字までに制限されていません。 このガイドでは、それらの属性はイタリックで示されます。 例えば、PCF では、RQMNAME に相当するものは *RemoteQMgrName* です。

エスケープ PCF

エスケープ PCF は、メッセージ・テキスト内に MQSC コマンドを含んでいる PCF コマンドです。 PCF を使用して、リモート・キュー・マネージャーにコマンドを送信することができます。 エスケープ PCF の詳細については、Escape を参照してください。

IBM MQ プログラマブル・コマンド・フォーマットの概要

プログラマブル・コマンド・フォーマット (PCF) では、ネットワーク内のプログラムと PCF 対応のキュー・マネージャーとの間で交換できるコマンド・メッセージと応答メッセージが定義されています。 PCF を使用すると、 キュー・マネージャーの管理やその他のネットワーク管理が単純化されます。 これによって、特に規模および複雑さが増していくネットワークの場合に、 分散ネットワークの管理が難しくなるという問題を解決できます。

プログラマブル・コマンド・フォーマットは、すべての IBM MQ プラットフォームでサポートされます。

PCF コマンドで解決できる問題

分散ネットワークの管理は、 複雑化する可能性があります。 ネットワークの規模および複雑さが増していくにつれ、 その管理に関する問題も増え続けます。

メッセージおよびキューイングに固有の 管理の例には、以下のものがあります。

- リソース管理。

  例えば、キューの作成や削除など。
- パフォーマンス・モニター。

  例えば、キューの最大長や メッセージ転送速度など。
- 制御。

  例えば、キューの最大長、最大メッセージ長、キューの有効化と無効化といった、 キューのパラメーターの調整など。
- メッセージ・ルーティング。

  ネットワークを経由する代替経路の定義。

IBM MQ PCF コマンドを使用して、キュー・マネージャーの管理やその他のネットワーク管理を単純化することができます。 PCF コマンドを使用すると、単一アプリケーションによって、ネットワーク内の単一キュー・マネージャーから ネットワーク管理を実行することができます。

PCF について

PCF とは、 プログラムとネットワーク内のキュー・マネージャー (PCF をサポートするもの) との間でやり取りできる コマンドおよび応答メッセージを定義するものです。 システム管理アプリケーション・プログラムで、IBM MQ オブジェクト (認証情報オブジェクト、チャネル、チャネル・リスナー、名前リスト、プロセス定義、キュー・マネージャー、キュー、サービス、ストレージ・クラス) を管理するために PCF コマンドを使用できます。 ネットワーク内の 1 つのポイントからアプリケーションを実行し、ローカル・キュー・マネージャーを使用して、キュー・マネージャー (ローカルまたはリモート) との間でコマンド情報と応答情報をやり取りすることもできます。

すべてのキュー・マネージャーは標準キュー名の管理キューを持っており、 このキューに対して、アプリケーションから PCF コマンドを送信できます。 また、すべてのキュー・マネージャーは、この管理キュー内のコマンド・メッセージを処理するための コマンド・サーバーを持っています。 このため、 PCF コマンド・メッセージは、ネットワーク内の任意のキュー・マネージャーで処理可能で、 応答データを、指定された応答キューを介してアプリケーションに返すことができます。 PCF コマンドおよび応答メッセージは、標準の Message Queue Interface (MQI) を使用して送受信されます。

使用できる PCF コマンドのリストは、パラメーターを含め、プログラマブル・コマンド・フォーマットの定義に記載されています。

IBM MQ プログラマブル・コマンド・フォーマットの使用

IBM MQ リモート管理のためにシステム管理プログラムで PCF を使用できます。

このセクションは、以下の項目から成っています。

- 26 ページの『PCF コマンド・メッセージ』
- 29 ページの『IBM MQ の PCF 応答』
- z/OS 31 ページの『Extended responses』
- IBM MQ オブジェクトの命名規則
- 33 ページの『IBM MQ の PCF コマンドの権限検査』

PCF コマンド・メッセージ

PCF コマンド・メッセージは、PCF ヘッダー、そのヘッダーに指定されているパラメーター、 およびユーザー定義のメッセージ・データで構成されます。 このメッセージは、 Message Queue Interface 呼び出しを使用して発行されます。

各コマンドとそのパラメーターは、PCF ヘッダーとその後の多数のパラメーター構造を含む個別のコマンド・メッセージとして送信されます。PCF ヘッダーについて詳しくは、MQCFH - PCF ヘッダーを参照してください。パラメーター構造の例については、MQCFST - PCF ストリング・パラメーターを参照してください。 PCF ヘッダーには、 コマンドと、その同じメッセージ内に続いて入っているパラメーター構造体の数が 示されます。 各パラメーター構造体は、コマンドにパラメーターを 指定します。

これらのコマンドへの応答は、コマンド・サーバーによって生成され、 同様の構造を持っています。 PCF ヘッダーがあり、その後にいくつかの パラメーター構造体が続きます。 応答は、複数のメッセージで構成される場合もありますが、 コマンドは、必ず 1 つのメッセージだけで構成されます。

Multi マルチプラットフォームでは、PCF コマンドの送信先のキューは常に SYSTEM.ADMIN.COMMAND.QUEUE。

z/OS z/OS では、コマンドは SYSTEM.COMMAND.INPUT(ただし SYSTEM.ADMIN.COMMAND.QUEUE は、その別名にすることができます。 このキューを処理するコマンド・サーバーは、コマンド・メッセージのメッセージ記述子の *ReplyToQ* および *ReplyToQMgr* フィールドによって定義されたキューに応答を送信します。

PCF コマンド・メッセージの発行方法

PCF コマンドおよび応答メッセージのキューへの書き込みおよびキューからの取り出しには、 MQPUT、MQGET などの標準の Message Queue Interface (MQI) 呼び出しを使用します。

注 :

宛先キュー・マネージャーで PCF コマンドが処理されるように、 そのキュー・マネージャーでコマンド・サーバーが稼働していることを 確認してください。

提供されているヘッダー・ファイルのリストについては、 IBM MQ COPY ファイル、ヘッダー・ファイル、インクルード・ファイル、およびモジュール・ファイルを参照してください。

PCF コマンドのメッセージ記述子

IBM MQ のメッセージ記述子については、MQMD - メッセージ記述子で詳細に説明しています。

PCF コマンド・メッセージの メッセージ記述子には、以下のフィールドが含まれています。

レポート
必要に即した有効な値。

MsgType
このフィールドは、応答を必要とするメッセージであることを示す MQMT_REQUEST で なければなりません。

Expiry
必要に即した有効な値。

Feedback
MQFB_NONE に設定してください

Multi **Encoding**
IBM MQ for Multiplatforms システムに送信する場合は、このフィールドに、メッセージ・データに使用されるエンコード方式を設定します。 必要であれば、変換が実行されます。

Multi **CodedCharSetId**
IBM MQ for Multiplatforms システムに送信する場合は、このフィールドに、メッセージ・データに使用されるコード化文字セット ID を設定します。 必要であれば、変換が実行されます。

Format
MQFMT_ADMIN に設定してください

優先順位
必要に即した有効な値。

Persistence
必要に即した有効な値。

MsgId
送信側アプリケーションで任意の値を指定できます。また、MQMI_NONE を指定して、 固有のメッセージ ID を生成するようにキュー・マネージャーに要求することもできます。

CorrelId
送信側アプリケーションで任意の値を指定できます。また、相関 ID がないことを示す MQCI_NONE を指定することもできます。

ReplyToQ
応答を受け取るキューの名前。

ReplyToQMgr
応答先のキュー・マネージャーの名前 (または空白)。

メッセージ・コンテキスト・フィールド
これらのフィールドには、適宜、有効な値を設定できます。 一般的には、 書き込みメッセージ・オプション MQPMO_DEFAULT_CONTEXT を使用して、 このメッセージ・コンテキスト・フィールドにデフォルト値を設定します。

バージョン 2 の MQMD 構造を使用している場合は、 以下の追加フィールドを設定する必要があります。

GroupId
MQGI_NONE に設定してください

MsgSeqNumber
1 に設定してください

オフセット
0 に設定してください

MsgFlags
MQMF_NONE に設定してください

OriginalLength
MQOL_UNDEFINED に設定してください

ユーザー・データの送信

PCF 構造を使用して、 ユーザー定義のメッセージ・データを送信することもできます。 この場合、メッセージ記述子の *Format* フィールドを MQFMT_PCF に設定する必要があります。

指定したキューにおける *PCF* メッセージの送信および受信

指定したキューへの PCF メッセージの送信

指定したキューへメッセージを送信するために、mqPutBag 呼び出しは指定したバッグの内容を PCF メッセージに変換し、指定したキューへそのメッセージを送信します。 バッグの内容は呼び出し後も変わりません。

この呼び出しへの入力として、次のものを指定しなければなりません。

- MQI 接続ハンドル。
- メッセージを置くキューのオブジェクト・ハンドル。
- メッセージ記述子。 メッセージ記述子の詳細については、MQMD - メッセージ記述子を参照してください。
- MQPMO 構造体を使用した書き込みメッセージ・オプション。 MQPMO 構造体の詳細については、MQPMO - 書き込みメッセージ・オプションを参照してください。
- メッセージへ変換するバッグのハンドル。

注: バッグに管理メッセージが含まれており、mqAddInquiry 呼び出しを使用して値がバッグに挿入された場合、MQIASY_COMMAND データ項目の値は、MQAI によって認識される INQUIRE コマンドでなければなりません。

mqPutBag 呼び出しの詳細な説明については、mqPutBag を参照してください。

指定したキューからの PCF メッセージの受信

指定したキューからメッセージを受信するために、mqGetBag 呼び出しは指定したキューから PCF メッセージを取得し、そのメッセージ・データをデータ・バッグへ変換します。

この呼び出しへの入力として、次のものを指定しなければなりません。

- MQI 接続ハンドル。
- メッセージの読み取り元のキューのオブジェクト・ハンドル。
- メッセージ記述子。 MQMD 構造内の **Format** パラメーターは、MQFMT_ADMIN、MQFMT_EVENT、または MQFMT_PCF でなければなりません。

  注: 作業単位内でメッセージを受け取り (つまり、MQGMO_SYNCPOINT オプションを指定)、そのメッセージの形式がサポートされない場合、その作業単位はバックアウトされることがあります。 そして、そのメッセージはキューで復元され、mqGetBag 呼び出しではなく、MQGET 呼び出しを使用して取得できます。 メッセージ記述子の詳細については、 MQGMO - メッセージ取得オプションを参照してください。
- MQGMO 構造体を使用した読み取りメッセージ・オプション。 MQGMO 構造体の詳細については、MQMD - メッセージ記述子を参照してください。
- 変換されたメッセージを入れるバッグのハンドル。

mqGetBag 呼び出しの詳細な説明については、mqGetBag を参照してください。

IBM MQ の PCF 応答

各コマンドに応じて、コマンド・サーバーは 1 つ以上の応答メッセージを生成します。 応答メッセージの形式は、コマンド・メッセージの形式と似ています。

この PCF ヘッダーには、応答対象のコマンドと同じコマンド ID 値が入ります (詳細については、MQCFH - PCF ヘッダーを参照)。 要求された Report オプションに応じて、 メッセージ ID および相関 ID が設定されます。

コマンド・メッセージの PCF ヘッダー・タイプが MQCFT_COMMAND の場合は、 標準応答のみが生成されます。 このようなコマンドは、Multiplatforms でのみサポートされます。 古いアプリケーションでは、z/OS 上の PCF はサポートされません。 IBM MQ Windows エクスプローラーはそのようなアプリケーションの 1 つです (ただし、 IBM WebSphere® MQ 6.0 以降の IBM MQ エクスプローラーは、 z/OS 上で PCF をサポートしています)。

コマンド・メッセージの PCF ヘッダー・タイプが MQCFT_COMMAND_XR の場合は、 拡張または標準応答のいずれかが生成されます。 このようなコマンドは、 z/OS および 一部の Multiplatforms でサポートされています。 z/OS で発行されたコマンドは、拡張応答のみを生成します。

単一のコマンドに、オブジェクトの総称名が指定されていた場合は、 一致するオブジェクトごとに個別の応答が、それぞれのメッセージで返されます。 応答生成において、総称名が指定されている単一のコマンドは、 複数の個々のコマンドとして扱われます (制御フィールド MQCFC_LAST または MQCFC_NOT_LAST は例外です)。 これ以外の場合、1 つのコマンド・メッセージは 1 つの応答メッセージを 生成します。

特定の PCF 応答は、要求されていなくても 構造体を返すことができます。 このような構造体は、応答の定義 (プログラマブル・コマンド・フォーマットの定義) に、常に返される ものとして示されます。 そのような応答では、応答の中でオブジェクトに名前を付けて、 そのデータを適用するオブジェクトを示す必要があるためです。

応答のメッセージ記述子

応答メッセージの メッセージ記述子には、以下のフィールドが含まれています。

MsgType
このフィールドは MQMT_REPLY です。

MsgId
このフィールドはキュー・マネージャーによって生成されます。

CorrelId
このフィールドはコマンド・メッセージの Report オプションに応じて 生成されます。

Format
このフィールドは MQFMT_ADMIN です。

Encoding
MQENC_NATIVE に設定してください。

CodedCharSetId
MQCCSI_Q_MGR に設定してください。

Persistence
コマンド・メッセージのものと同じです。

優先順位
コマンド・メッセージのものと同じです。

応答は MQPMO_PASS_IDENTITY_CONTEXT で生成されます。

Multi 標準応答
ヘッダー・タイプが MQCFT_COMMAND のコマンド・メッセージには、 標準応答が生成されます。 このようなコマンドは、Multiplatforms でのみサポートされます。

標準応答には、以下の 3 つのタイプがあります。

- OK 応答
- エラー応答
- データ応答

OK 応答

この応答は、 *CompCode* フィールドが MQCC_OK または MQCC_WARNING の コマンド形式ヘッダーで始まるメッセージで構成されます。

MQCC_OK の場合、*Reason* は MQRC_NONE です。

MQCC_WARNING の場合、*Reason* は、 警告の性質を示します。 この場合、コマンド形式ヘッダーの後に、 この理由コードに則した警告パラメーター構造体が 1 つ以上続いている可能性があります。

どちらの場合でも、照会コマンドに関しては、以下のセクションで説明しているように、 追加のパラメーター構造体が後に続いている可能性があります。

エラー応答

コマンドにエラーが発生した場合、1 つ以上のエラー応答メッセージが送信されます (通常であれば 応答メッセージが 1 つしかないコマンドであっても、複数の応答メッセージが送信される場合があります)。 これらのエラー応答メッセージには、適宜、MQCFC_LAST または MQCFC_NOT_LAST が設定されます。

このようなメッセージは、すべて、 *CompCode* 値が MQCC_FAILED であり、 *Reason* フィールドにその特定のエラーについて示されている 応答フォーマット・ヘッダーで始まっています。 一般的に、メッセージごとに異なる エラーが示されます。 また、各メッセージのヘッダーの後には、ゼロ個または 1 個の (複数になることはありません) エラーのパラメーター構造体が続いています。 このパラメーター構造体が 1 つ存在している場合、それは、 *Parameter* フィールドに 以下のいずれかが入った MQCFIN 構造体です。

- MQIACF_PARAMETER_ID

  この構造体の *Value* フィールドは、 エラーになったパラメーターのパラメーター ID です (MQCA_Q_NAME など)。

- MQIACF_ERROR_ID

  この値は、*Reason* 値 (コマンド形式ヘッダー内のもの) が MQRC_UNEXPECTED_ERROR の場合に使用されます。 MQCFIN 構造体の *Value* フィールドは、コマンド・サーバーが受け取った予期しない理由コードです。

- MQIACF_SELECTOR

  この値が入っているのは、コマンドと一緒に送信されたリスト構造体 (MQCFIL) に、重複するセレクターまたは無効なセレクターが含まれていた場合です。 コマンド形式ヘッダーの *Reason* フィールドでこのエラーについて示し、 MQCFIN 構造体の *Value* フィールドには エラーになったコマンドの MQCFIL 構造体のパラメーター値が入ります。

- MQIACF_ERROR_OFFSET

  この値が入っているのは、 Ping Channel コマンドでデータ比較エラーが発生した場合です。 この構造体の *Value* フィールドは、 Ping Channel 比較エラーのオフセットです。

- MQIA_CODED_CHAR_SET_ID

  この値が入っているのは、着信 PCF コマンド・メッセージの メッセージ記述子のコード化文字セット ID が、宛先キュー・マネージャーのものと 一致しなかった場合です。 この構造体の *Value* フィールドは、キュー・マネージャーのコード化文字セット ID です。

最後の (または唯一の) エラー応答メッセージは、 応答の要約です。この *CompCode* フィールドは MQCC_FAILED で、*Reason* フィールドは MQRCCF_COMMAND_FAILED です。 このメッセージのヘッダーの後には、 パラメーター構造体はありません。

データ応答

この応答は、照会コマンドに対する OK 応答 (前述の説明を参照) で構成されます。 OK 応答の後には、プログラマブル・コマンド・フォーマットの定義で説明しているように、要求されたデータが入っている追加の構造体が続きます。

アプリケーションは、 これらの追加のパラメーター構造体が特定の順番で返されることに依存するものであってはなりません。

z/OS *Extended responses*

Commands issued on z/OS generate extended responses.

There are three types of extended response:

- Message response, with type MQCFT_XR_MSG
- Item response, with type MQCFT_XR_ITEM
- Summary response, with type MQCFT_XR_SUMMARY

Each command can generate one, or more, sets of responses. Each set of responses comprises one or more messages, numbered sequentially from 1 in the *MsgSeqNumber* field of the PCF header. The *Control* field of the last (or only) response in each set has the value MQCFC_LAST. For all other responses in the set, this value is MQCFC_NOT_LAST.

Any response can include one, or more, optional MQCFBS structures in which the *Parameter* field is set to MQBACF_RESPONSE_SET, the value being a response set identifier. Identifiers are unique and identify the set of responses which contain the response. For every set of responses, there is an MQCFBS structure that identifies it.

Extended responses have at least two parameter structures:

- An MQCFBS structure with the *Parameter* field set to MQBACF_RESPONSE_ID. The value in this field is the identifier of the set of responses to which the response belongs. The identifier in the first set is arbitrary. In subsequent sets, the identifier is one previously notified in an MQBACF_RESPONSE_SET structure.

- An MQCFST structure with the *Parameter* field set to MQCACF_RESPONSE_Q_MGR_NAME, the value being the name of the queue manager from which the set of responses come.

Many responses have additional parameter structures, and these structures are described in the following sections.

You cannot determine in advance how many responses there are in a set other than by getting responses until one with MQCFC_LAST is found. Neither can you determine in advance how many sets of responses there are as any set might include MQBACF_RESPONSE_SET structures to indicate that additional sets are generated.

Extended responses to Inquire commands

Inquire commands normally generate an item response (type MQCFT_XR_ITEM) for each item found that matches the specified search criteria. The item response has a *CompCode* field in the header with a value of MQCC_OK, and a *Reason* field with a value of MQRC_NONE. It also includes other parameter structures describing the item and its requested attributes, as described in Definitions of the Programmable Command Formats.

If an item is in error, the *CompCode* field in the header has a value of MQCC_FAILED and the *Reason* field identifies the particular error. Additional parameter structures are included to identify the item.

Certain Inquire commands might return general (not name-specific) message responses in addition to the item responses. These responses are informational, or error, responses of the type MQCFT_XR_MSG.

If the Inquire command succeeds, there might, optionally, be a summary response (type MQCFT_XR_SUMMARY), with a *CompCode* value of MQCC_OK, and a *Reason* field value of MQRC_NONE.

If the Inquire command fails, item responses might be returned, and there might optionally be a summary response (type MQCFT_XR_SUMMARY), with a *CompCode* value of MQCC_FAILED, and a *Reason* field value of MQRCCF_COMMAND_FAILED.

Extended responses to commands other than Inquire

Successful commands generate message responses in which the *CompCode* field in the header has a value of MQCC_OK, and the *Reason* field has a value of MQRC_NONE. There is always at least one message; it might be informational (MQCFT_XR_MSG) or a summary (MQCFT_XR_SUMMARY). There might optionally be additional informational (type MQCFT_XR_MSG) messages. Each informational message might include a number of additional parameter structures with information about the command; see the individual command descriptions for the structures that can occur.

Commands that fail generate error message responses (type MQCFT_XR_MSG), in which the *CompCode* field in the header has a value of MQCC_FAILED and the *Reason* field identifies the particular error. Each message might include a number of additional parameter structures with information about the error: see the individual error descriptions for the structures that can occur. Informational message responses might be generated. There might, optionally, be a summary response (MQCFT_XR_SUMMARY), with a *CompCode* value of MQCC_FAILED, and a *Reason* field value of MQRCCF_COMMAND_FAILED.

Extended responses to commands using CommandScope

If a command uses the **CommandScope** parameter, or causes a command using the **CommandScope** parameter to be generated, there is an initial response set from the queue manager where the command was received. Then a separate set, or sets, of responses is generated for each queue manager to which the command is directed (as if multiple individual commands were issued). Finally, there is a response set from the receiving queue manager which includes an overall summary response (type MQCFT_XR_SUMMARY). The MQCACF_RESPONSE_Q_MGR_NAME parameter structure identifies the queue manager that generates each set.

The initial response set has the following additional parameter structures:

- MQIACF_COMMAND_INFO (MQCFIN). Possible values in this structure are MQCMDI_CMDSCOPE_ACCEPTED or MQCMDI_CMDSCOPE_GENERATED.
- MQIACF_CMDSCOPE_Q_MGR_COUNT (MQCFIN). This structure indicates the number of queue managers to which the command is sent.

IBM MQ の PCF コマンドの権限検査

PCF コマンドの処理では、コマンド・メッセージのメッセージ記述子内の *UserIdentifier* を使用して、必要な IBM MQ オブジェクト権限の検査が行われます。 権限検査の実施方法は、このトピックで説明するように、プラットフォームごとに異なります。

検査は、コマンドが処理されるシステム上で実行されます。 したがって、ユーザー ID は宛先システム上に存在し、そのコマンドを処理するために必要な権限を保持していなければなりません。 メッセージを リモート・システムから受信する場合、宛先システム上に ID を存在させる 1 つの方法は、ローカルおよびリモート・システムの両方に 同じユーザー ID を用意することです。

注 : z/OS z/OS での権限検査については、タスク 1: z/OS システム・パラメーターを識別するを参照してください。

IBM MQ for IBM i

IBM i

PCF コマンドを処理するには、ユーザー ID に、ターゲット・システム上の IBM MQ オブジェクトに対する *dsp* 権限が必要です。

また、IBM MQ オブジェクト権限の検査は、34 ページの表 2 に示すように特定の PCF コマンドに対して実行されます。

ほとんどの場合、これらの検査は、ローカル・システムで発行される同等の IBM MQ CL コマンドによって実行される検査と同じです。 IBM MQ 権限から IBM i システム権限へのマッピング、および IBM MQ CL コマンドの権限要件について詳しくは、 IBM i でのセキュリティーのセットアップ を参照してください。 出口に関するセキュリティーについて詳しくは、セキュリティー出口を使用したリンク・レベル・セキュリティーを参照してください。

以下の コマンドを処理するには、ユーザー ID が グループ・プロファイル QMQMADM のメンバーでなければなりません。

- Ping Channel
- Change Channel
- Copy Channel
- Create Channel
- Delete Channel
- Reset Channel
- Resolve Channel
- Start Channel
- Stop Channel
- Start Channel Initiator
- Start Channel Listener

IBM MQ for UNIX, Linux, and Windows

ALW

PCF コマンドを処理するためには、ユーザー ID が、宛先システム上の キュー・マネージャー・オブジェクトに対する *dsp* 権限を保持していなければなりません。 また、IBM MQ オブジェクト権限の検査は、34 ページの表 2 に示すように特定の PCF コマンドに対して実行されます。

以下の コマンドを処理するには、ユーザー ID が グループ *mqm* に属していなければなりません。

注 : Windows **の場合のみ、**ユーザー ID はグループ *Administrators* またはグループ *mqm* に属することができます。

- Change Channel
- Copy Channel
- Create Channel
- Delete Channel
- Ping Channel
- Reset Channel
- Start Channel
- Stop Channel
- Start Channel Initiator
- Start Channel Listener
- Resolve Channel
- Reset Cluster
- Refresh Cluster
- Suspend Queue Manager
- Resume Queue Manager

IBM MQ のオブジェクト権限 (Multiplatforms)

Multi

表 *2.* オブジェクト権限

コマンド	IBM MQ オブジェクト権限	クラス権限 (オブジェクト・タイプに対するもの)
Change Authentication Information	dsp および chg	n/a
Change Channel	dsp および chg	n/a
Change Channel Listener	dsp および chg	n/a
Change Client Connection Channel	dsp および chg	n/a
Change Namelist	dsp および chg	n/a
Change Process	dsp および chg	n/a
Change Queue	dsp および chg	n/a
Change Queue Manager	chg 注 *3* および *5* を参照	n/a
Change Service	dsp および chg	n/a
Clear Queue	clr	n/a
Copy Authentication Information	dsp	crt
Copy Authentication Information (Replace) 注 *1* を参照	最小*:* dsp 最大*:* chg	crt
Copy Channel	dsp	crt
Copy Channel (Replace) 注 *1* を参照	最小*:* dsp 最大*:* chg	crt

表 2. オブジェクト権限 (続き)		
コマンド	**IBM MQ オブジェクト権限**	**クラス権限 (オブジェクト・タイプに対するもの)**
Copy Channel Listener	dsp	crt
Copy Channel Listener (Replace) 注 *1* を参照	最小*:* dsp 最大*:* chg	crt
Copy Client Connection Channel	dsp	crt
Copy Client Connection Channel (Replace) 注 *1* を参照	最小*:* dsp 最大*:* chg	crt
Copy Namelist	dsp	crt
Copy Namelist (Replace) 注 *1* を参照	最小*:* dsp 最大*:* dsp および chg	crt
Copy Process	dsp	crt
Copy Process (Replace) 注 *1* を参照	最小*:* dsp 最大*:* chg	crt
Copy Queue	dsp	crt
Copy Queue (Replace) 注 *1* を参照	最小*:* dsp 最大*:* dsp および chg	crt
Create Authentication Information	*(*システムのデフォルト認証情報*)* dsp	crt
Create Authentication Information (Replace) 注 *1* を参照	*(*システムのデフォルト認証情報*)* dsp 最大*:* chg	crt
Create Channel	*(*システム・デフォルト・チャネル*)* dsp	crt
Create Channel (Replace) 注 *1* を参照	*(*システム・デフォルト・チャネル*)* dsp 最大*:* chg	crt
Create Channel Listener	*(*システム・デフォルト・リスナー*)* dsp	crt
Create Channel Listener (Replace) 注 *1* を参照	*(*システム・デフォルト・リスナー*)* dsp 最大*:* chg	crt
Create Client Connection Channel	*(*システム・デフォルト・チャネル*)* dsp	crt
Create Client Connection Channel (Replace) 注 *1* を参照	*(*システム・デフォルト・チャネル*)* dsp 最大*:* chg	crt
Create Namelist	*(*システム・デフォルト名前リスト*)* dsp	crt
Create Namelist (Replace) 注 *1* を参照	*(*システム・デフォルト名前リスト*)* dsp 最大*:* dsp および chg	crt
Create Process	*(*システム・デフォルト・プロセス*)* dsp	crt
Create Process (Replace) 注 *1* を参照	*(*システム・デフォルト・プロセス*)* dsp 最大*:* chg	crt
Create Queue	*(*システム・デフォルト・キュー*)* dsp	crt
Create Queue (Replace) 注 *1* を参照	*(*システム・デフォルト・キュー*)* dsp 最大*:* dsp および chg	crt
Create Service	*(*システム・デフォルト・キュー*)* dsp	crt

表 2. オブジェクト権限 (続き)		
コマンド	**IBM MQ オブジェクト権限**	**クラス権限 (オブジェクト・タイプに対するもの)**
Create Service (Replace) 注 *1* を参照	*(*システム・デフォルト・キュー*)* dsp 最大*:* chg	crt
Delete Authentication Information	dsp および dlt	n/a
Delete Authority Record	*(*キュー・マネージャー・オブジェクト*)* chg 注 *4* を参照	注 *4* を参照
Delete Channel	dsp および dlt	n/a
Delete Channel Listener	dsp および dlt	n/a
Delete Client Connection Channel	dsp および dlt	n/a
Delete Namelist	dsp および dlt	n/a
Delete Process	dsp および dlt	n/a
Delete Queue	dsp および dlt	n/a
Delete Service	dsp および dlt	n/a
Inquire Authentication Information	dsp	n/a
Inquire Authority Records	注 *4* を参照	注 *4* を参照
Inquire Channel	dsp	n/a
Inquire Channel Listener	dsp	n/a
Inquire Channel Status (**ChannelType** MQCHT_CLSSDR の場合)	inq	n/a
Inquire Client Connection Channel	dsp	n/a
Inquire Namelist	dsp	n/a
Inquire Process	dsp	n/a
Inquire Queue	dsp	n/a
Inquire Queue Manager	注 *3* を参照	n/a
Inquire Queue Status	dsp	n/a
Inquire Service	dsp	n/a
Ping Channel	ctrl	n/a
Ping Queue Manager	注 *3* を参照	n/a
キュー・マネージャーのリフレッシュ	(キュー・マネージャー・オブジェクト) chg	n/a
Refresh Security (**SecurityType** MQSECTYPE_SSL の場合)	(キュー・マネージャー・オブジェクト) chg	n/a
Reset Channel	ctrlx	n/a

表 2. オブジェクト権限 (続き)		
コマンド	**IBM MQ オブジェクト権限**	**クラス権限 (オブジェクト・タイプに対するもの)**
Reset Queue Manager	(キュー・マネージャー・オブジェクト) chg	n/a
Reset Queue Statistics	dsp および chg	n/a
Resolve Channel	ctrlx	n/a
Set Authority Record	(キュー・マネージャー・オブジェクト) chg 注 *4* を参照	注 *4* を参照
Start Channel	ctrl	n/a
Stop Channel	ctrl	n/a
Stop Connection	(キュー・マネージャー・オブジェクト) chg	n/a
Start Listener	ctrl	n/a
Stop Listener	ctrl	n/a
Start Service	ctrl	n/a
Stop Service	ctrl	n/a
Escape	注 *2* を参照	注 *2* を参照

注 :

1. このコマンドは、置き換えられるオブジェクトが存在する場合に適用されます。 存在しない場合は、Create、または Copy (Replace なし) に対するものと同様の権限検査が行われます。
2. 必要な権限は、エスケープ・テキストで定義される MQSC コマンドによって決まり、上記のコマンドのいずれかに相当します。
3. PCF コマンドを処理するためには、ユーザー ID が、宛先システム上の キュー・マネージャー・オブジェクトに対する dsp 権限を保持していなければなりません。
4. この PCF コマンドは、 コマンド・サーバーが -a パラメーターを指定して開始されている場合を除いて許可されます。 デフォルトでは、 コマンド・サーバー (-a パラメーターが指定されていない場合) はキュー・マネージャーの開始時に開始されます。 詳細については、プログラマブル・コマンド・フォーマット・リファレンスを参照してください。
5. ユーザー ID にキュー・マネージャーの chg 権限を付与するということは、 すべてのグループおよびユーザーに関する権限レコードを設定する能力を与えるということです。 一般のユーザーまたはアプリケーションには、 この権限を付与しないでください。

IBM MQ には、チャネル・セキュリティー出口点もいくつか用意されています。このため、セキュリティー検査用の独自のユーザー出口プログラムを導入することができます。 詳細については、チャネルの表示を参照してください。

Multi MQAI を使用して PCF の使い方を単純化する

IBM MQ 管理インターフェース (MQAI) は、 IBM MQ に対するプログラミング・インターフェースであり、AIX、 IBM i、 Linux、 、および Windows で使用できます。 このインターフェースは、IBM MQ キュー・マネージャーでデータ・バッグを使用して管理タスクを実行し、プログラマブル・コマンド・フォーマット (PCF) を使用するよりも簡単にオブジェクトのプロパティー (またはパラメーター) を処理します。

MQAI は、データ・バッグ を使用することにより、キュー・マネージャー上の管理タスクを実行します。データ・バッグを使用すると、PCF を使用するよりも簡単な方法で、オブジェクトのプロパティー (またはパラメーター) を処理することができます。

MQAI を使用する利点は、次のとおりです。

PCF メッセージの使用を単純化する
MQAI は IBM MQ を管理するためのより簡単な手段です。 MQAI を使用する場合、独自の PCF メッセージを作成する必要がありません。 このため、複雑なデータ構造体に関連する問題を回避できます。

MQI 呼び出しを使用して書き込まれたプログラムにおいてパラメーターを渡すには、PCF メッセージにコマンドおよびストリングまたは整数データの詳細を入れる必要があります。 この構成を手動で作成するには、 すべての構造体について複数のステートメントをプログラムに追加し、メモリー・スペースを割り振る必要があります。 このタスクは、時間がかかる大変な作業です。

MQAI を使用して書き込まれたプログラムはパラメーターを適切なデータ・バッグに渡し、構造ごとに 1 つのステートメントのみが必要です。 MQAI データ・バッグを使用すると、配列を処理して、ストレージを割り振る必要がなく、PCF の詳細を入れる必要がある程度なくなります。

エラー条件をより簡単に処理する
PCF コマンドからの戻りコードを取得することは困難です。 MQAI を使用すると、プログラムによるエラー状態の処理が簡単になります。

アプリケーション間でデータを交換する
アプリケーション・データは PCF 形式で送信され、MQAI によりパックおよびアンパックされます。 メッセージ・データが整数および文字ストリングで構成されている場合、MQAI を使用して PCF データ対応の IBM MQ 標準装備データ変換を利用することができます。 これによりデータ変換出口を書き込む必要がありません。

データ・バッグを作成してデータを設定した後、mqExecute 呼び出しを使用して、管理コマンド・メッセージをキュー・マネージャーのコマンド・サーバーに送信することができます。 この呼び出しは、応答メッセージを待機します。 mqExecute 呼び出しは、コマンド・サーバーとの交換を処理し、応答を応答バッグ に返します。

MQAI の使用例

以下のサンプル・プログラムは、MQAI を使用してさまざまなタスクを実行する方法を示しています。

- amqsaicq.c: ローカル・キューを作成します。
- amqsaiem.c: 単純なイベント・モニターを使用して、画面上のイベントを表示します。
- amqsailq.c: すべてのローカル・キューとその現在の深さのリストを印刷します。
- amqsaicl.c: すべてのチャネルとそのタイプのリストを印刷します。

MQAI アプリケーションの作成

MQAI を使用してアプリケーションを作成するには、IBM MQ の場合と同じライブラリーにリンクします。 IBM MQ アプリケーションの作成方法について詳しくは、プロシージャー型アプリケーションの構築を参照してください。

MQAI を使用した IBM MQ 構成のヒント

MQAI は、コマンド・サーバー自体を直接処理するのではなく、PCF メッセージを使用して、管理コマンドをコマンド・サーバーに送信します。 MQAI を使用した IBM MQ 構成のヒントは、38 ページの『MQAI で IBM MQ を構成するためのヒント』で説明されています。

関連資料
IBM MQ 管理インターフェース・リファレンス

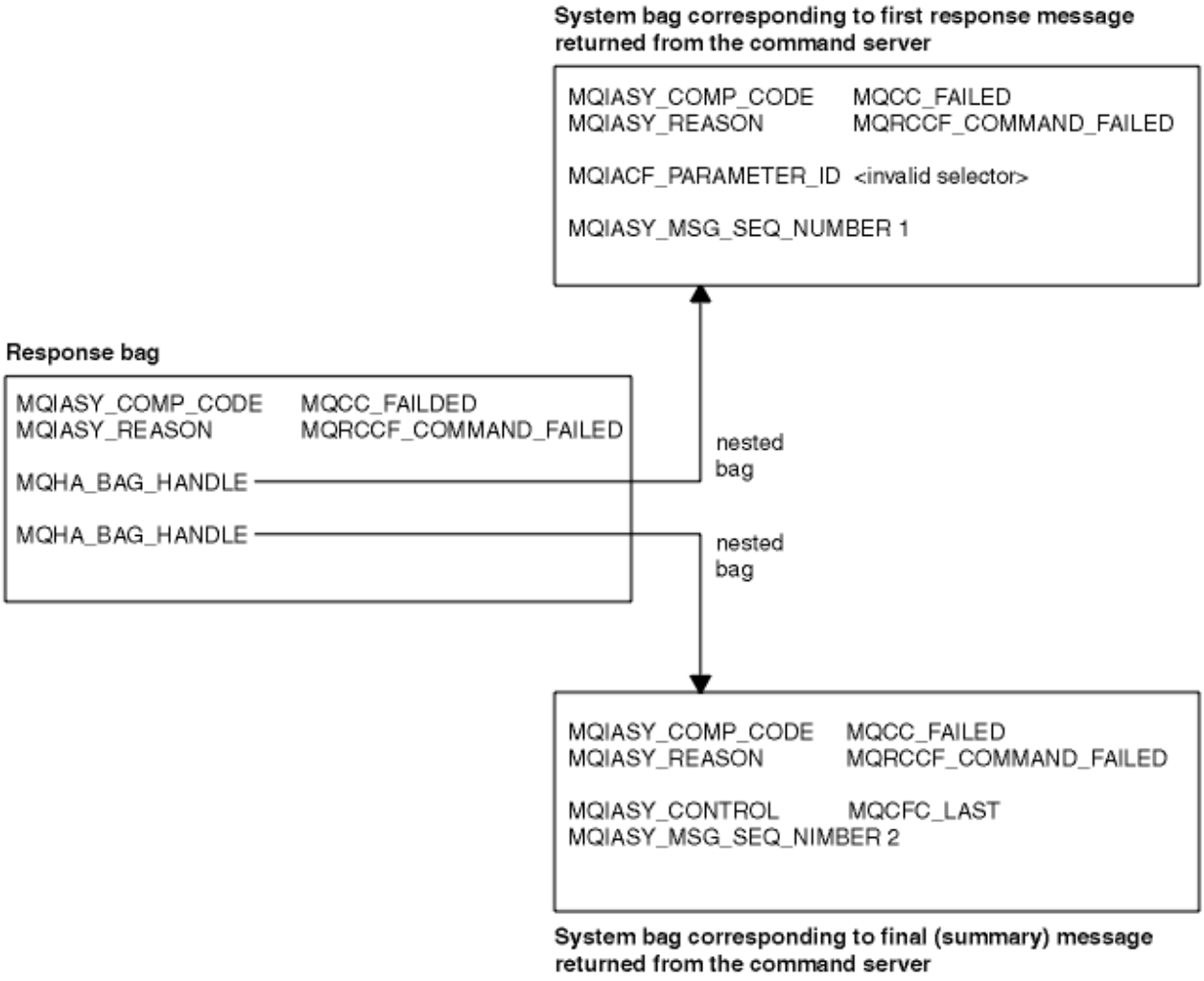
Multi MQAI で IBM MQ を構成するためのヒント

IBM MQ 管理インターフェース (MQAI) では、コマンド・サーバー自体を直接操作する代わりに、PCF メッセージを使用してコマンド・サーバーに管理コマンドを送信します。 ここでは、MQAI を使用して IBM MQ を構成するためのヒントをいくつか紹介します。

- IBM MQ では、文字ストリングが固定長になるようにブランクが埋め込まれます。 通常、C を使用する場合、ヌル終了ストリングを IBM MQ プログラミング・インターフェースへの入力パラメーターとして指定できます。
- ストリング属性の値をクリアするには、空ストリングではなく単一のブランク文字に設定します。
- 変更する属性を前もって検討し、その属性だけを照会します。
- キュー名やチャネル・タイプなど、特定の属性は変更できません。 変更可能な属性のみが変更の対象となるようにします。 特定の PCF 変更オブジェクトに関する必須パラメーターとオプション・パラメーターのリストを参照してください。 プログラマブル・コマンド・フォーマットの定義を参照してください。
- MQAI 呼び出しが失敗する場合、失敗の詳細の一部が応答バッグに返されます。 他の詳細は、セレクター MQHA_BAG_HANDLE がアクセスできるネストされたバッグにあります。 例えば、mqExecute 呼び出しが失敗し、MQRCCF_COMMAND_FAILED という理由コードが発行される場合、この情報は応答バッグに返されます。 この理由コードから、指定されたセレクターがコマンド・メッセージのタイプには無効であったことが考えられます。また、この情報の詳細は、バッグ・ハンドルによってアクセスできるネストされたバッグにあります。

  MQExecute の詳細については、72 ページの『mqExecute 呼び出しを使用した qm コマンド・サーバーへの管理コマンドの送信』を参照してください。

  次の図に、上記のシナリオを図示します。



Multi 拡張 MQAI トピック

索引付け、データ変換、およびメッセージ記述子の使用に関する情報

索引付け
索引は、バッグから既存のデータ項目を置換または削除する際に、挿入順序を保存するために使用されます。

データ変換
MQAI データ・バッグに含まれるストリングは、さまざまなコード化文字セットにすることができ、mqSetInteger 呼び出しを使用して変換できます。

メッセージ記述子の使用
MQAI は、データ・バッグの作成時に初期値に設定されるメッセージ記述子を生成します。

Multi *MQAI での索引付け*

索引は、既存のデータ項目をバッグから削除または置き換える時に使用されます。 索引付けには 3 つのタイプがあります。 これらにより、データ項目を容易に検索できるようになります。

バッグにあるデータ項目内の各セレクターおよび値には、次の 3 つの関連索引番号があります。

- 同一のセレクターの異なる複数の項目に関連する索引。
- 項目が属するセレクターのカテゴリー (ユーザーまたはシステム) に関連する索引。
- バッグにあるすべてのデータ項目 (ユーザーおよびシステム) に関連する索引。

これにより、40 ページの図 3 に示すように、ユーザー・セレクター、システム・セレクターまたはその両方によって索引付けすることができます。

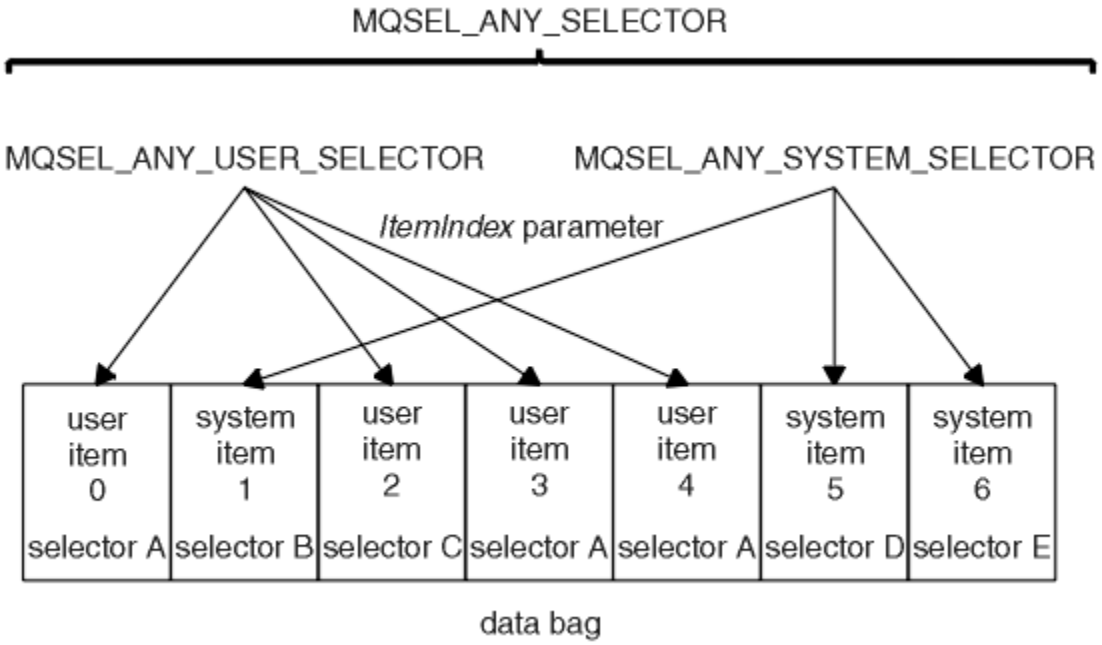


図 *3.* 索引付け

40 ページの図 3 では、ユーザー項目 3 (セレクター A) を次の対の索引で参照できます。

- セレクター A (ItemIndex 1)
- MQSEL_ANY_USER_SELECTOR (ItemIndex 2)
- MQSEL_ANY_SELECTOR (ItemIndex 3)

索引は、C 言語の配列のようにゼロ・ベースです。 つまり「n」個のオカレンスがある場合、索引の範囲は 0 から「n-1」までとなり、抜けている数字はありません。

索引は、既存のデータ項目をバッグから削除または置き換える時に使用されます。 この方法で索引を使用する場合、追加指示が保存されるので、他のデータ項目の索引に影響する場合があります。 この例については、69 ページの『バッグ内の情報の変更』 および 71 ページの『データ項目の削除』 を参照してください。

索引付けの 3 つのタイプにより、データ項目を簡単に検索できます。 例えば、バッグに特定のセレクター 1 つに対してインスタンスが 3 つある場合、mqCountItems 呼び出しによりセレクターのインスタンス数をカウントでき、mqInquire* 呼び出しはセレクターおよび索引の両方を指定してそれらの値だけを照会することができます。 これは、チャネル上の一部の出口ルーチンなど値のリストを指定できる属性に有効です。

Multi *MQAI でのデータ変換処理*

MQAI データ・バッグに含まれるストリングは、さまざまなコード化文字セットにすることができます。 これらのストリングは、mqSetInteger 呼び出しを使用して変換できます。

PCF メッセージと同様に、MQAI データ・バッグに含まれるストリングは、多様なコード化文字セットになります。 通常、PCF メッセージのストリングはすべて同じコード化文字セットです。つまり、キュー・マネージャーと同じセットになります。

データ・バッグの各ストリング項目には、ストリング自体と CCSID の 2 つの値が入ります。 バッグに追加されるストリングは、 mqAddString または mqSetString 呼び出しの **Buffer** パラメーターから取得されます。 CCSID は、MQIASY_CODED_CHAR_SET_ID のセレクターがあるシステム項目から取得されます。 これはバッグ *CCSID* と呼ばれ、mqSetInteger 呼び出しを使用して変更できます。

データ・バッグに入っているストリングの値を照会する場合、CCSID は呼び出しからの出力パラメーターになります。

41 ページの表 3 に、データ・バッグをメッセージに変換するとき、また逆にメッセージをデータ・バッグに変換するときに適用される規則を示します。

表 *3. CCSID* 処理

MQAI 呼び出し	**CCSID**	**呼び出しへの入力**	**呼び出しへの出力**
mqBagToBuffer	バッグ CCSID (1)	無視される	未変更
mqBagToBuffer	バッグのストリング CCSID	使用される	未変更
mqBagToBuffer	バッファーのストリング CCSID	適用外	バッグのストリング CCSID からコピーされる
mqBufferToBag	バッグ CCSID (1)	無視される	未変更
mqBufferToBag	バッファーのストリング CCSID	使用される	未変更
mqBufferToBagmqBufferToBag	バッグのストリング CCSID	適用外	バッファーのストリング CCSID からコピーされる
mqPutBag	MQMD CCSID	使用される	未変更 (2)
mqPutBag	バッグ CCSID (1)	無視される	未変更
mqPutBag	バッグのストリング CCSID	使用される	未変更
mqPutBag	送信されるメッセージのストリング CCSID	適用外	バッグのストリング CCSID からコピーされる
mqGetBag	MQMD CCSID	メッセージのデータ変換に使用される	返されるデータの CCSID に設定 (3)
mqGetBag	バッグ CCSID (1)	無視される	未変更
mqGetBag	メッセージのストリング CCSID	使用される	未変更
mqGetBag	バッグのストリング CCSID	適用外	メッセージのストリング CCSID からコピーされる

表 3. CCSID 処理 (続き)			
MQAI 呼び出し	**CCSID**	**呼び出しへの入力**	**呼び出しへの出力**
mqExecute	要求 - バッグ CCSID	要求メッセージの MQMD に使用される (4)	未変更
mqExecute	応答 - バッグ CCSID	応答メッセージのデータ変換に使用される (4)	返されるデータの CCSID に設定 (3)
mqExecute	要求バッグのストリング CCSID	要求メッセージに使用される	未変更
mqExecute	応答バッグのストリング CCSID	適用外	応答メッセージのストリング CCSID からコピーされる

注 :

1. バッグ CCSID は、セレクター MQIASY_CODED_CHAR_SET_ID があるシステム項目です。
2. MQCCSI_Q_MGR は、実際のキュー・マネージャー CCSID に変更されます。
3. データ変換が要求される場合、返されるデータの CCSID は出力値と同じです。 データ変換が要求されない場合、返されるデータの CCSID はメッセージ値と同じです。 データ変換が要求されていてもそのデータ変換が失敗した場合、メッセージは返されません。
4. CCSID が MQCCSI_DEFAULT の場合、キュー・マネージャーの CCSID が使用されます。

関連概念

203 ページの『コード化文字セット間のデータ変換』
IBM MQ で定義された形式 (組み込み形式とも呼ばれる) のメッセージ・データは、キュー・マネージャーによって 1 つのコード化文字セットからもう 1 つのコード化文字セットに変換することができます。ただし、2 つのコード化文字セットが、1 つの言語または類似する言語グループに関連付けられていることが必要です。

205 ページの『ccsid_part2.tbl ファイル』
ccsid_part2.tbl ファイルは、追加の CCSID 情報を提供するために使用されます。 ccsid_part2.tbl ファイルは、 IBM MQ 9.0 より前に使用されていた ccsid.tbl ファイルを置き換えます。

Multi *MQAI でのメッセージ記述子の使用*

MQAI が生成するメッセージ記述子は、データ・バッグの作成時に初期値に設定されます。

PCF コマンド・タイプはセレクター MQIASY_TYPE があるシステム項目から取得されます。 データ・バッグを作成する場合、この項目の初期値は作成するバッグのタイプに応じて設定されます。

表 4. PCF コマンド・タイプ	
バッグのタイプ	**MQIASY_TYPE 項目の初期値**
MQCBO_ADMIN_BAG	MQCFT_COMMAND
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_*	MQCFT_USER

MQAI がメッセージ記述子を生成する場合、**Format** パラメーターおよび **MsgType** パラメーターに使用される値は、42 ページの表 4 に示すように、セレクター MQIASY_TYPE があるシステム項目の値によって異なります。

表 5. MQMD の形式および MsgType パラメーター		
PCF コマンド・タイプ	**Format**	**MsgType**
MQCFT_COMMAND	MQFMT_ADMIN	MQMT_REQUEST

表 5. MQMD の形式および MsgType パラメーター (続き)		
PCF コマンド・タイプ	**Format**	**MsgType**
MQCFT_REPORT	MQFMT_ADMIN	MQMT_REPORT
MQCFT_RESPONSE	MQFMT_ADMIN	MQMT_REPLY
MQCFT_TRACE_ROUTE	MQFMT_ADMIN	MQMT_DATAGRAM
MQCFT_EVENT	MQFMT_EVENT	MQMT_DATAGRAM
MQCFT_*	MQFMT_PCF	MQMT_DATAGRAM

管理バッグまたはコマンド・バッグを作成する場合、メッセージ記述子の *Format* は MQFMT_ADMIN になり、*MsgType* は MQMT_REQUEST になることが、42 ページの表 5 からわかります。 これは、応答が返されると予測されるときにコマンド・サーバーに送信される PCF 要求メッセージに適しています。

メッセージ記述子の他のパラメーターは、43 ページの表 6 に示す値を取ります。

表 6. メッセージ記述子値	
パラメーター	**値**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	42 ページの表 5 を参照
Expiry	30 秒 (注 43 ページの『1』)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	バッグ CCSID により異なる (注 43 ページの『2』)
Format	42 ページの表 5 を参照
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	注 43 ページの『3』 を参照
ReplyToQMgr	ブランク

注:

1. **OptionsBag** パラメーターを使用すると、この値を mqExecute 呼び出しに指定変更することが できます。 詳しくは、mqExecute を参照してください。
2. 41 ページの『MQAI でのデータ変換処理』 を参照してください。
3. タイプ MQMT_REQUEST のメッセージのユーザー指定応答キューの名前または MQAI 生成の一時動的キューの名前。 あるいはブランク。

Multi ローカル・キューを作成するサンプル C プログラム (amqsaicq.c)

サンプル C プログラム `amqsaicq.c` は、 MQAI を使用してローカル・キューを作成します。

```
/********************************************************************************/
/*                                                                              */
/* Program name: AMQSAICQ.C                                                     */
/*                                                                              */
/* Description:  Sample C program to create a local queue using the             */
/*               IBM MQ Administration Interface (MQAI).                        */
/*                                                                              */
/* Statement:    Licensed Materials - Property of IBM                           */
/*                                                                              */
/*               84H2000, 5765-B73                                              */
/*               84H2001, 5639-B42                                              */
/*               84H2002, 5765-B74                                              */
/*               84H2003, 5765-B75                                              */
/*               84H2004, 5639-B43                                              */
/*                                                                              */
/*               (C) Copyright IBM Corp. 1999, 2024                             */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* Function:                                                                    */
/*    AMQSAICQ is a sample C program that creates a local queue and is an       */
/*    example of the use of the mqExecute call.                                 */
/*                                                                              */
/*     - The name of the queue to be created is a parameter to the program.     */
/*                                                                              */
/*     - A PCF command is built by placing items into an MQAI bag.              */
/*       These are:-                                                            */
/*            - The name of the queue                                           */
/*            - The type of queue required, which, in this case, is local.      */
/*                                                                              */
/*     - The mqExecute call is executed with the command MQCMD_CREATE_Q.        */
/*       The call generates the correct PCF structure.                          */
/*       The call receives the reply from the command server and formats into   */
/*       the response bag.                                                      */
/*                                                                              */
/*     - The completion code from the mqExecute call is checked and if there    */
/*       is a failure from the command server then the code returned by the     */
/*       command server is retrieved from the system bag that is                */
/*       embedded in the response bag to the mqExecute call.                    */
/*                                                                              */
/* Note: The command server must be running.                                    */
/*                                                                              */
/*                                                                              */

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

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

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

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

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

```
     exit(99);
   }

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

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


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

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

}
/*************************************************************************/
/*                                                                       */
/* Function:    CreateLocalQueue                                         */
/* Description: Create a local queue by sending a PCF command to the command  */
/*              server.                                                  */
/*                                                                       */
/*************************************************************************/
/*                                                                       */
/* Input Parameters:  Handle to the queue manager                        */
/*                    Name of the queue to be created                    */
/*                                                                       */
/* Output Parameters: None                                               */
/*                                                                       */
/* Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.     */
/*        The call generates the correct PCF structure.                  */
/*        The default options to the call are used so that the command is sent*/
/*        to the SYSTEM.ADMIN.COMMAND.QUEUE.                             */
/*        The reply from the command server is placed on a temporary dynamic  */
/*        queue.                                                         */
/*        The reply is read from the temporary queue and formatted into the   */
/*        response bag.                                                  */
/*                                                                       */
/*        The completion code from the mqExecute call is checked and if there */
/*        is a failure from the command server then the code returned by the  */
/*        command server is retrieved from the system bag that is        */
/*        embedded in the response bag to the mqExecute call.            */
/*                                                                       */
/*************************************************************************/
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG reason;                       /* reason code                   */
   MQLONG compCode;                     /* completion code               */
   MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute       */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute      */
   MQHBAG resultBag;                    /* result bag from mqExecute       */
   MQLONG mqExecuteCC;                  /* mqExecute completion code       */
   MQLONG mqExecuteRC;                  /* mqExecute reason code           */

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


   /*************************************************************************/
   /* Create a command Bag for the mqExecute call. Exit the function if the   */
   /* create fails.                                                         */
   /*************************************************************************/
```

```
mqCreateBag(MQCBO_ADMIN_BAG, &commandBag, &compCode, &reason);
CheckCallResult("Create the command bag", compCode, reason);
if (compCode !=MQCC_OK)
   return;

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

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

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

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

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


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

      /**************************************************************************/
      /* Get the completion code and reason code, returned by the command  */
      /* server, from the embedded error bag.                                   */
      /**************************************************************************/
      mqInquireInteger(resultBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                       &compCode, &reason);
      CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
```

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

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


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

}
```

Multi イベント・モニターを使用してイベントを表示するサンプル C プログラム (amqsaiem.c)

サンプル C プログラム amqsaiem.c は、 MQAI を使用する基本イベント・モニターを示しています。

```
*************************************************************************/
/*                                                                       */
/* Program name: AMQSAIEM.C                                              */
/*                                                                       */
/* Description:  Sample C program to demonstrate a basic event monitor   */
/*               using the IBM MQ Admin Interface (MQAI).                */
/* Licensed Materials - Property of IBM                                  */
/*                                                                       */
/* 63H9336                                                               */
/* (c) Copyright IBM Corp. 1999, 2024. All Rights Reserved.              */
/*                                                                       */
/* US Government Users Restricted Rights - Use, duplication or           */
/* disclosure restricted by GSA ADP Schedule Contract with               */
/* IBM Corp.                                                             */
/*************************************************************************/
/*                                                                       */
/* Function:                                                             */
/*    AMQSAIEM is a sample C program that demonstrates how to write a simple  */
/*    event monitor using the mqGetBag call and other MQAI calls.        */
/*                                                                       */
```

```
/*    The name of the event queue to be monitored is passed as a parameter    */
/*    to the program. This would usually be one of the system event queues:-  */
/*            SYSTEM.ADMIN.QMGR.EVENT        Queue Manager events             */
/*            SYSTEM.ADMIN.PERFM.EVENT       Performance events               */
/*            SYSTEM.ADMIN.CHANNEL.EVENT     Channel events                   */
/*            SYSTEM.ADMIN.LOGGER.EVENT      Logger events                    */
/*                                                                            */
/*    To monitor the queue manager event queue or the performance event queue,*/
/*    the attributes of the queue manager need to be changed to enable        */
/*    these events. For more information about this, see Part 1 of the        */
/*    Programmable System Management book. The queue manager attributes can   */
/*    be changed using either MQSC commands or the MQAI interface.            */
/*    Channel events are enabled by default.                                  */
/*                                                                            */
/* Program logic                                                              */
/*    Connect to the Queue Manager.                                           */
/*    Open the requested event queue with a wait interval of 30 seconds.      */
/*    Wait for a message, and when it arrives get the message from the queue  */
/*    and format it into an MQAI bag using the mqGetBag call.                 */
/*    There are many types of event messages and it is beyond the scope of    */
/*    this sample to program for all event messages. Instead the program      */
/*    prints out the contents of the formatted bag.                           */
/*    Loop around to wait for another message until either there is an error  */
/*    or the wait interval of 30 seconds is reached.                          */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* AMQSAIEM has 2 parameters - the name of the event queue to be monitored    */
/*                           - the queue manager name (optional)              */
/*                                                                            */
/******************************************************************************

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

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

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

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

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

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

```
   }

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

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

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

   return 0;

}

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

}

/*****************************************************************/
/*                                                               */
/* Function: GetQEvents                                          */
/*                                                               */
/*****************************************************************/
/*                                                               */
/* Input Parameters:  Handle to the queue manager                */
/*                    Name of the event queue to be monitored    */
/*                                                               */
/* Output Parameters: None                                       */
/*
/* Logic:   Open the event queue.                                */
/*          Get a message off the event queue and format the message into     */
/*          a bag.                                               */
/*          A real event monitor would need to be programmed to deal with     */
/*          each type of event that it receives from the queue. This is       */
/*          outside the scope of this sample, so instead, the contents of     */
/*          the bag are printed.                                 */
/*          The program waits for 30 seconds for an event message and then    */
/*          terminates if no more messages are available.        */
/*                                                               */
/*****************************************************************/
```

```
void GetQEvents(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG openReason;                       /* MQOPEN reason code             */
   MQLONG reason;                           /* reason code                    */
   MQLONG compCode;                         /* completion code                */
   MQHOBJ eventQueue;                       /* handle to event queue          */

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

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

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

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

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

     /******************************************************************************/
     /* Get the message from the event queue and convert it into the event    */
     /* bag.                                                                       */
     /******************************************************************************/
     mqGetBag(hConn, eventQueue, &md, &gmo, eventBag, &compCode, &reason);

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

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

     /******************************************************************************/
     /* Event message read - Print the contents of the event bag              */
     /******************************************************************************/
```

```
     else
     {
       if ( PrintBag(eventBag) )
           printf("\nError found while printing bag contents\n");

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

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

} /* end of GetQEvents */

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

int PrintBag(MQHBAG dataBag)
{
    int errors;

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

    return errors;
}

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

```
#define INDENT 4                        /* Number of spaces to indent     */
                                        /* embedded bag display           */


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

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

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

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

       /*********************************************************************/
       /* First inquire the type of the item for each item in the bag       */
       /*********************************************************************/
       mqInquireItemInfo(dataBag,             /* Bag handle                 */
                         MQSEL_ANY_SELECTOR,  /* Item can have any selector*/
                         i,                   /* Index position in the bag */
                         &selector,           /* Actual value of selector  */
                                              /* returned by call          */
                         &itemType,           /* Actual type of item       */
                                              /* returned by call          */
                         &compCode,           /* Completion code           */
                         &reason);            /* Reason Code               */

       if (compCode != MQCC_OK)
          errors++;

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

            if (compCode != MQCC_OK)
               errors++;
            else
               printf("%.*s  %-2d     %-4d     (%d)\n",
```

```
                  indent, blanks, i, selector, iValue);
        break

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

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


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

        /*****************************************************************/
        /* The call can return a warning if the string is too long for */
        /* the output buffer and has been truncated, so only check     */
        /* explicitly for call failure.                                  */
        /*****************************************************************/
        if (compCode == MQCC_FAILED)
            errors++;
        else
        {
           /*************************************************************/
           /* Remove trailing blanks from the string and terminate with*/
           /* a null. First check that the string should not have been */
           /* longer than the maximum buffer size allowed.              */
           /*************************************************************/
           if (stringLength > LENGTH)
              trimLength = LENGTH;
           else
              trimLength = stringLength;
           mqTrim(trimLength, stringVal, stringVal, &compCode, &reason);
           printf("%.*s  %-2d     %-4d     '%s' %d\n",
                  indent, blanks, i, selector, stringVal, ccsid);
        }
        break;

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

        /*****************************************************************/
        /* The call can return a warning if the string is too long for */
        /* the output buffer and has been truncated, so only check     */
        /* explicitly for call failure.                                  */
        /*****************************************************************/
```

```
                /*************************************************************/
                if (compCode == MQCC_FAILED)
                    errors++;
                else
                {
                   printf("%.*s  %-2d     %-4d     X'",
                          indent, blanks, i, selector);

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

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

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

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

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

Multi チャネル・オブジェクトを照会するサンプル C プログラム (amqsaicl.c)

サンプル C プログラム amqsaicl.c は、 MQAI を使用してチャネル・オブジェクトを照会します。

```
/*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAICL.C                                                  */
/*                                                                           */
/* Description:  Sample C program to inquire channel objects                 */
/*               using the IBM MQ Administration Interface (MQAI)            */
/*                                                                           */
/* <N_OCO_COPYRIGHT>                                                         */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 2008, 2024. All Rights Reserved.                  */
/*                                                                           */
/* US Government Users Restricted Rights - Use, duplication or               */
/* disclosure restricted by GSA ADP Schedule Contract with                   */
/* IBM Corp.                                                                 */
/* <NOC_COPYRIGHT>                                                           */
/*****************************************************************************/
/*                                                                           */
/* Function:                                                                 */
/*    AMQSAICL is a sample C program that demonstrates how to inquire        */
/*    attributes of the local queue manager using the MQAI interface. In     */
/*    particular, it inquires all channels and their types.                  */
/*                                                                           */
/*     - A PCF command is built from items placed into an MQAI administration */
```

```
/*       bag.                                                                 */
/*       These are:-                                                          */
/*            - The generic channel name "*"                                  */
/*            - The attributes to be inquired. In this sample we just want    */
/*              name and type attributes                                      */
/*                                                                            */
/*     - The mqExecute MQCMD_INQUIRE_CHANNEL call is executed.                */
/*       The call generates the correct PCF structure.                        */
/*       The default options to the call are used so that the command is sent */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                   */
/*       The reply from the command server is placed on a temporary dynamic   */
/*       queue.                                                               */
/*       The reply from the MQCMD_INQUIRE_CHANNEL is read from the            */
/*       temporary queue and formatted into the response bag.                 */
/*                                                                            */
/*     - The completion code from the mqExecute call is checked and if there  */
/*       is a failure from the command server, then the code returned by the  */
/*       command server is retrieved from the system bag that has been        */
/*       embedded in the response bag to the mqExecute call.                  */
/*                                                                            */
/* Note: The command server must be running.                                  */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* AMQSAICL has 2 parameter - the queue manager name (optional)               */
/*                          - output file (optional) default varies           */
/******************************************************************************/

/******************************************************************************/
/* Includes                                                                   */
/******************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#if (MQAT_DEFAULT == MQAT_OS400)
#include <recio.h>
#endif

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

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

/******************************************************************************/
/* DataTypes                                                                  */
/******************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
typedef _RFILE OUTFILEHDL;
#else
typedef FILE OUTFILEHDL;
#endif

/******************************************************************************/
/* Constants                                                                  */
/******************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "*SDR     ",    /* MQCHT_SENDER    */
  "*SVR     ",    /* MQCHT_SERVER    */
  "*RCVR    ",    /* MQCHT_RECEIVER  */
  "*RQSTR   ",    /* MQCHT_REQUESTER */
  "*ALL     ",    /* MQCHT_ALL       */
  "*CLTCN   ",    /* MQCHT_CLNTCONN  */
  "*SVRCONN ",    /* MQCHT_SVRCONN   */
  "*CLUSRCVR",    /* MQCHT_CLUSRCVR  */
  "*CLUSSDR "     /* MQCHT_CLUSSDR   */
};
#else
const struct
{
  char name[9];
} ChlTypeMap[9] =
```

```
{
  "sdr      ",    /* MQCHT_SENDER    */
  "svr      ",    /* MQCHT_SERVER    */
  "rcvr     ",    /* MQCHT_RECEIVER  */
  "rqstr    ",    /* MQCHT_REQUESTER */
  "all      ",    /* MQCHT_ALL       */
  "cltconn  ",    /* MQCHT_CLNTCONN  */
  "svrcn    ",    /* MQCHT_SVRCONN   */
  "clusrcvr ",    /* MQCHT_CLUSRCVR  */
  "clussdr  "     /* MQCHT_CLUSSDR   */
};
#endif

/********************************************************************************/
/* Macros                                                                       */
/********************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
  #define OUTFILE "QTEMP/AMQSAICL(AMQSAICL)"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = _Ropen((fname),"wr, rtncode=Y");
  #define CLOSEOUTFILE(hdl) \
    _Rclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    _Rwrite((hdl),(buf),(buflen));

#elif (MQAT_DEFAULT == MQAT_UNIX)
  #define OUTFILE "/tmp/amqsaicl.txt"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#else
  #define OUTFILE "amqsaicl.txt"
  #define OPENOUTFILE(fname) \
    fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#endif

#define ChlType2String(t) ChlTypeMap[(t)-1].name

/********************************************************************************/
/* Function: main                                                               */
/********************************************************************************/
int main(int argc, char *argv[])
{
   /*****************************************************************************/
   /* MQAI variables                                                            */
   /*****************************************************************************/
   MQHCONN hConn;                          /* handle to MQ connection         */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name             */
   MQLONG reason;                          /* reason code                     */
   MQLONG connReason;                      /* MQCONN reason code              */
   MQLONG compCode;                        /* completion code                 */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute         */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute       */
   MQHBAG cAttrsBag;                       /* bag containing chl attributes   */
   MQHBAG errorBag;                        /* bag containing cmd server error */
   MQLONG mqExecuteCC;                     /* mqExecute completion code       */
   MQLONG mqExecuteRC;                     /* mqExecute reason code           */
   MQLONG chlNameLength;                   /* Actual length of chl name       */
   MQLONG chlType;                         /* Channel type                    */
   MQLONG i;                               /* loop counter                    */
   MQLONG numberOfBags;                    /* number of bags in response bag  */
   MQCHAR chlName[MQ_OBJECT_NAME_LENGTH+1];/* name of chl extracted from bag  */
   MQCHAR OutputBuffer[100];               /* output data buffer              */
   OUTFILEHDL *outfp = NULL;               /* output file handle              */

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

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

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

/*****************************************************************************/
/* Open the output file                                                      */
/*****************************************************************************/
if (argc &gt; 2)
{
  OPENOUTFILE(outfp, argv[2]);
}
else
{
  OPENOUTFILE(outfp, OUTFILE);
}

if(outfp == NULL)
{
  printf("Could not open output file.\n");
  goto MOD_EXIT;
}
/*****************************************************************************/
/* Create an admin bag for the mqExecute call                                */
/*****************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag;, &compCode;, &reason;);
CheckCallResult("Create admin bag", compCode, reason);

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

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

/*****************************************************************************/
/* Put the channel type into the admin bag                                   */
/*****************************************************************************/
mqAddInteger(adminBag, MQIACH_CHANNEL_TYPE, MQCHT_ALL, &compCode;, &reason;);
CheckCallResult("Add channel type", compCode, reason);

/*****************************************************************************/
/* Add an inquiry for various attributes                                     */
/*****************************************************************************/
mqAddInquiry(adminBag, MQIACH_CHANNEL_TYPE, &compCode;, &reason;);
CheckCallResult("Add inquiry", compCode, reason);

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

/*****************************************************************************/
/* Check the command server is started. If not exit.                         */
/*****************************************************************************/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName="">\n");
   goto MOD_EXIT;
}
```

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

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

      /*****************************************************************/
      /* Get the channel name out of the channel attributes bag              */
      /*****************************************************************/
      mqInquireString(cAttrsBag, MQCACH_CHANNEL_NAME, 0, MQ_OBJECT_NAME_LENGTH,
                      chlName, &chlNameLength, NULL, &compCode, &reason);
      CheckCallResult("Get channel name", compCode, reason);

      /*****************************************************************/
      /* Get the channel type out of the channel attributes bag              */
      /*****************************************************************/

   mqInquireInteger(cAttrsBag, MQIACH_CHANNEL_TYPE, MQIND_NONE, &chlType,
                    &compCode, &reason);
      CheckCallResult("Get type", compCode, reason);

      /*****************************************************************/
      /* Use mqTrim to prepare the channel name for printing.                */
      /* Print the result.                                               */
      /*****************************************************************/
      mqTrim(MQ_CHANNEL_NAME_LENGTH, chlName, chlName, &compCode, &reason);
      sprintf(OutputBuffer, "%-20s%-9s", chlName, ChlType2String(chlType));
      WRITEOUTFILE(outfp,OutputBuffer,29)
    }
  }

  else                                            /* Failed mqExecute     */
  {
    printf("Call to get channel attributes failed: Cc = %ld : Rc = %ld\n",
                compCode, reason);
    /*****************************************************************/
    /* If the command fails get the system bag handle out of the mqexecute   */
    /* response bag.This bag contains the reason from the command server     */
    /* why the command failed.                                         */
    /*****************************************************************/
    if (reason == MQRCCF_COMMAND_FAILED)
    {
      mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag,
                   &compCode, &reason);
      CheckCallResult("Get the result bag handle", compCode, reason);

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

MOD_EXIT:
```

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

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

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

   /******************************************************************************/
   /* Close the output file if open                                              */
   /******************************************************************************/
   if(outfp != NULL)
      CLOSEOUTFILE(outfp);

   return 0;
}


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

Multi キューを照会して情報を印刷するサンプル C プログラム (amqsailq.c)

サンプル C プログラム amqsailq.c は、 MQAI を使用してローカル・キューの現在の深さを照会します。

```
/******************************************************************************/
/*                                                                            */
/* Program name: AMQSAILQ.C                                                   */
/*                                                                            */
/* Description:  Sample C program to inquire the current depth of the local   */
/*               queues using the IBM MQ Administration Interface (MQAI)      */
/*                                                                            */
/* Statement:    Licensed Materials - Property of IBM                         */
/*                                                                            */
/*               84H2000, 5765-B73                                            */
/*               84H2001, 5639-B42                                            */
/*               84H2002, 5765-B74                                            */
/*               84H2003, 5765-B75                                            */
/*               84H2004, 5639-B43                                            */
```

```
/*                                                                          */
/*              (C) Copyright IBM Corp. 1999, 2024                          */
/*                                                                          */
/****************************************************************************/
/*                                                                          */
/* Function:                                                                */
/*    AMQSAILQ is a sample C program that demonstrates how to inquire       */
/*    attributes of the local queue manager using the MQAI interface. In    */
/*    particular, it inquires the current depths of all the local queues.   */
/*                                                                          */
/*     - A PCF command is built by placing items into an MQAI administration*/
/*       bag.                                                               */
/*       These are:-                                                        */
/*            - The generic queue name "*"                                  */
/*            - The type of queue required. In this sample we want to       */
/*              inquire local queues.                                       */
/*            - The attribute to be inquired. In this sample we want the    */
/*              current depths.                                             */
/*                                                                          */
/*     - The mqExecute call is executed with the command MQCMD_INQUIRE_Q.   */
/*       The call generates the correct PCF structure.                      */
/*       The default options to the call are used so that the command is sent*/
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                 */
/*       The reply from the command server is placed on a temporary dynamic */
/*       queue.                                                             */
/*       The reply from the MQCMD_INQUIRE_Q command is read from the        */
/*       temporary queue and formatted into the response bag.               */
/*                                                                          */
/*     - The completion code from the mqExecute call is checked and if there*/
/*       is a failure from the command server, then the code returned by    */
/*       command server is retrieved from the system bag that has been      */
/*       embedded in the response bag to the mqExecute call.                */
/*                                                                          */
/*     - If the call is successful, the depth of each local queue is placed */
/*       in system bags embedded in the response bag of the mqExecute call. */
/*       The name and depth of each queue is obtained from each of the bags */
/*       and the result displayed on the screen.                            */
/*                                                                          */
/* Note: The command server must be running.                                */
/*                                                                          */
/****************************************************************************/
/*                                                                          */
/* AMQSAILQ has 1 parameter - the queue manager name (optional)             */
/*                                                                          */
/****************************************************************************/

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

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

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

/****************************************************************************/
/* Function: main                                                           */
/****************************************************************************/
int main(int argc, char *argv[])
{
   /*************************************************************************/
   /* MQAI variables                                                        */
   /*************************************************************************/
   MQHCONN hConn;                       /* handle to IBM MQ connection     */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name           */
   MQLONG reason;                       /* reason code                     */
   MQLONG connReason;                   /* MQCONN reason code              */
   MQLONG compCode;                     /* completion code                 */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute      */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute    */
   MQHBAG qAttrsBag;                    /* bag containing q attributes     */
   MQHBAG errorBag;                     /* bag containing cmd server error */
   MQLONG mqExecuteCC;                  /* mqExecute completion code       */
   MQLONG mqExecuteRC;                  /* mqExecute reason code           */
```

```
MQLONG qNameLength;                    /* Actual length of q name         */
MQLONG qDepth;                         /* depth of queue                  */
MQLONG i;                              /* loop counter                    */
MQLONG numberOfBags;                   /* number of bags in response bag  */
MQCHAR qName[MQ_Q_NAME_LENGTH+1];      /* name of queue extracted from bag*/


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

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

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

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

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

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

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

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


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

```
}

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

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

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

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

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

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

  /*****************************************************************/
  /* If the command fails get the system bag handle out of the mqExecute   */
  /* response bag. This bag contains the reason from the command server    */
  /* why the command failed.                                         */
  /*****************************************************************/
  if (reason == MQRCCF_COMMAND_FAILED)
  {
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag, &compCode,
                 &reason);
    CheckCallResult("Get the result bag handle", compCode, reason);

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

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

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

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

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

Multi データ・バッグと MQAI

データ・バッグは、IBM MQ 管理インターフェース (MQAI) でオブジェクトのプロパティーやパラメーターを処理する手段です。

データ・バッグ

- データ・バッグには、ゼロ個以上のデータ項目 が入っています。 これらのデータ項目がバッグに入ると、データ項目はバッグ内で配列されます。 これを追加配列 と呼びます。 各データ項目には、データ項目およびデータ項目の値 を識別するセレクター が含まれます。データ項目の値としては、整数、64 ビット整数、整数フィルター、ストリング、ストリング・フィルター、バイト・ストリング、バイト・ストリング・フィルター、または別のバッグ・ハンドルが可能です。 データ項目については、66 ページの『MQAI で使用できるデータ項目のタイプ』で詳しく説明されています。

  セレクターには、ユーザー・セレクター とシステム・セレクター の 2 種類があります。 これらについては、MQAI セレクターに説明されています。 セレクターは通常固有ですが、同じセレクターに複数の値を指定することが可能です。 複数の値を指定する場合、索引 により必要となるセレクターの特定オカレンスを識別します。 索引については、40 ページの『MQAI での索引付け』を参照してください。

  これらの概念の階層を図 1 に示します。

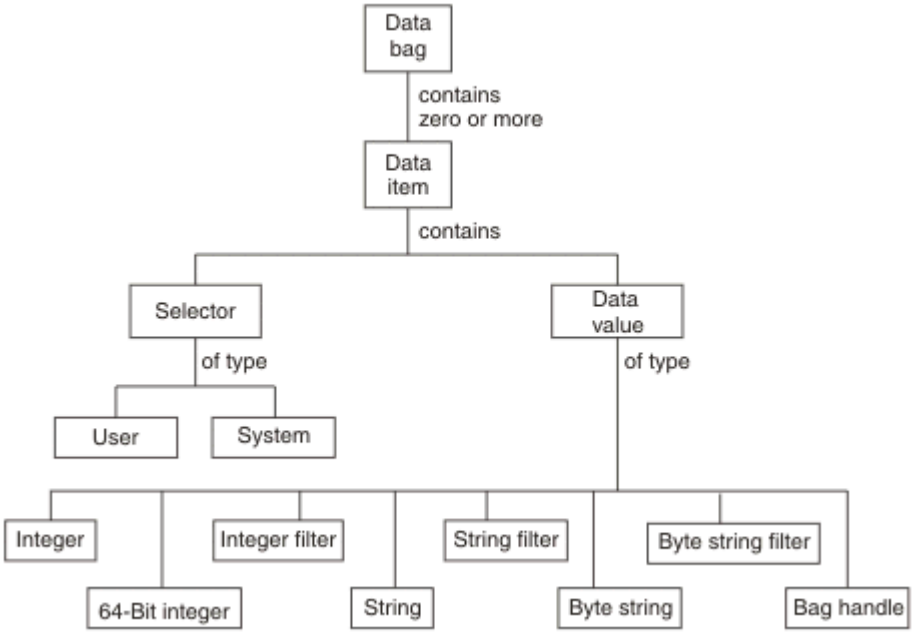


図 *4. MQAI* 概念の階層

この階層は、前の段落で説明されています。

データ・バッグのタイプ

実行したいタスクに応じて作成するデータ・バッグのタイプを選択することができます。

ユーザー・バッグ (user bag)
ユーザー・データに使用される簡単なバッグです。

管理バッグ (administration bag)
管理メッセージをコマンド・サーバーに送信することで IBM MQ オブジェクトを管理するときに使用されるデータに対して作成されるバッグです。 65 ページの『データ・バッグの作成および削除』で説明するように、管理バッグは自動的に特定のオプションを暗黙設定します。

コマンド・バッグ (command bag)
これも、IBM MQ オブジェクト管理用コマンドに対して作成されるバッグです。 しかし管理バッグと異なり、コマンド・バッグは、特定のオプションを使用できますが、そのオプションを自動的に暗黙設定しません。 オプションの詳細については、65 ページの『データ・バッグの作成および削除』を参照してください。

グループ・バッグ
グループ化されたデータ項目のセットを保持するために使用されるバッグです。 グループ・バッグは、IBM MQ オブジェクトの管理には使用できません。

また、**システム・バッグ**は、応答メッセージがコマンド・サーバーから返され、ユーザーの出力バッグに入れられたときに MQAI によって作成されます。 システム・バッグをユーザーが変更することはできません。

データ・バッグの使用: このトピックには、データ・バッグのさまざまな使用法がリストされています。

データ・バッグの使用

次のリストには、データ・バッグのさまざまな使用法が示されています。

- データ・バッグを作成および削除する: 65 ページの『データ・バッグの作成および削除』
- データ・バッグを使用してアプリケーション間でデータを送信する: 65 ページの『MQAI を使用したデータ・バッグの書き込みと受信』

- データ・バッグにデータ項目を追加する: 67 ページの『MQAI を使用してバッグにデータ項目を追加する方法』
- データ・バッグ内に照会コマンドを追加する: 67 ページの『バッグへの照会コマンドの追加』
- データ・バッグ内を照会する: 68 ページの『データ・バッグ内の照会』
- データ・バッグ内のデータ項目数をカウントする: 71 ページの『データ項目のカウント』
- データ・バッグ内の情報を変更する: 69 ページの『バッグ内の情報の変更』
- データ・バッグを初期化する: 70 ページの『mqClearBag 呼び出しによるバッグのクリア』
- データ・バッグを切り捨てる: 70 ページの『mqTruncateBag 呼び出しによるバッグの切り捨て』
- バッグとバッファーを変換する: 70 ページの『バッグおよびバッファーの変換』

Multi データ・バッグの作成および削除

データ・バッグの作成

MQAI を使用するには、mqCreateBag 呼び出しを使用して、まずデータ・バッグを作成します。 この呼び出しへの入力として、バッグの作成を制御するためのオプションを 1 つ以上指定します。

MQCreateBag 呼び出しの **Options** パラメーターでは、ユーザー・バッグ、コマンド・バッグ、グループ・バッグ、または管理バックのどれを作成するかを選択することができます。

ユーザー・バッグ、コマンド・バッグ、またはグループ・バッグを作成するには、以下を行うためのオプションをさらに 1 つ以上選択できます。

- バッグ内で同じセレクターが 2 つ以上隣接している場合、リスト形式を使用する。
- パラメーターが正しい順序になるように、データ項目が PCF メッセージに追加されたとおりにデータ項目を再配列する。 データ項目の詳細については、66 ページの『MQAI で使用できるデータ項目のタイプ』を参照してください。
- バッグに追加する項目に関して、ユーザー・セレクターの値を検査する。

管理バッグでは、これらのオプションは自動的に暗黙指定されます。

データ・バッグは、ハンドルによって識別されます。 バッグ・ハンドルは mqCreateBag から戻され、そのデータ・バッグを使用する他のすべての呼び出しで指定される必要があります。

mqCreateBag 呼び出しの詳細な説明については、mqCreateBag を参照してください。

データ・バッグの削除

ユーザーが作成したデータ・バッグは、mqDeleteBag 呼び出しを使用して削除もしなければなりません。 例えば、バッグがユーザー・コード内で作成されている場合、削除もユーザー・コード内で行う必要があります。

システム・バッグの作成および削除は、MQAI によって自動的に行われます。 この作業の詳細については、72 ページの『mqExecute 呼び出しを使用した qm コマンド・サーバーへの管理コマンドの送信』を参照してください。 ユーザー・コードでは、システム・バッグを削除できません。

mqDeleteBag 呼び出しの詳細な説明については、mqDeleteBag を参照してください。

Multi *MQAI* を使用したデータ・バッグの書き込みと受信

mqPutBag 呼び出しおよび mqGetBag 呼び出しを使用してデータ・バッグの書き込みおよび取得を行うことにより、アプリケーション間でデータを送信することもできます。 その結果、IBM MQ 管理インターフェース (MQAI) で、アプリケーションではなくバッファーを処理できるようになります。

mqPutBag 呼び出しは指定したバッグの内容を PCF メッセージに変換し、そのメッセージを指定したキューに送信します。mqGetBag 呼び出しは指定したキューからメッセージを削除し、そのメッセージを再びデータ・バッグに変換します。 したがって、mqPutBag 呼び出しは、mqBagToBuffer 呼び出しの後に

MQPUT を実行する場合と同等であり、mqGetBag 呼び出しは MQGET 呼び出しの後に mqBufferToBag を実行する場合と同等です。

特定のキューでの PCF メッセージの送受信について詳しくは、28 ページの『指定したキューにおける PCF メッセージの送信および受信』を参照してください。

注 : mqGetBag 呼び出しを使用する場合は、メッセージ内の PCF 詳細が正しくなければなりません。正しくない場合、適切なエラー結果および PCF メッセージが返されません。

Multi *MQAI* で使用できるデータ項目のタイプ

IBM MQ 管理インターフェースでは、データ項目を使用して、データ・バッグの作成時にデータを設定します。 これらのデータ項目には、ユーザー項目とシステム項目があります。

これらのユーザー項目には、管理対象となっているオブジェクトの属性などのユーザー・データが含まれます。 システム項目は、生成されるメッセージをさらに制御するために使用する必要があります (例えば、メッセージ・ヘッダーの生成)。 システム項目について詳しくは、66 ページの『システム項目と MQAI』を参照してください。

データ項目のタイプ

データ・バッグを作成した場合は、そこに整数項目または文字ストリング項目を取り込むことができます。3 つの項目タイプすべてについて照会を行えます。

データ項目は、整数項目または文字ストリング項目のいずれかになります。 以下に、MQAI 内で使用可能なデータ項目のタイプを示します。

- 整数
- 64 ビット整数
- 整数フィルター
- 文字ストリング
- ストリング・フィルター
- バイト・ストリング
- バイト・ストリング・フィルター
- バッグ・ ハンドル

データ項目の使用

以下に、データ項目を使用する方法を示します。

- 71 ページの『データ項目のカウント』.
- 71 ページの『データ項目の削除』.
- 67 ページの『MQAI を使用してバッグにデータ項目を追加する方法』.
- 68 ページの『データ項目のフィルター処理および照会』.

Multi システム項目と *MQAI*

IBM MQ 管理管理 (MQAI) では、以下の目的のためにシステム項目を使用できます。

- PCF ヘッダーの生成。 システム項目により、PCF コマンド ID、制御オプション、メッセージ順序番号、およびコマンド・タイプを制御できます。
- データ変換。 システム項目により、バッグにある文字ストリング項目の文字セット ID を処理します。

すべてのデータ項目と同様に、システム項目はセレクターおよび値で構成されます。 セレクターおよびその用途については、MQAI セレクターを参照してください。

システム項目は固有です。 1 つ以上のシステム項目をシステム・セレクターで識別できます。 各システム・セレクターのオカレンスは 1 回だけです。

ほとんどのシステム項目を変更できますが (69 ページの『バッグ内の情報の変更』を参照)、バッグ作成オプションはユーザーが変更することはできません。 システム項目を削除することはできません。 (71 ページの『データ項目の削除』を参照してください。)

Multi *MQAI* を使用してバッグにデータ項目を追加する方法

IBM MQ 管理インターフェース (MQAI) でデータ・バッグを作成する時に、データ項目を使用してデータを設定できます。 これらのデータ項目には、ユーザー項目とシステム項目があります。

データ項目の詳細については、66 ページの『MQAI で使用できるデータ項目のタイプ』を参照してください。

MQAI では、整数項目、64 ビット整数項目、整数フィルター項目、文字ストリング項目、ストリング・フィルター、バイト・ストリング項目、およびバイト・ストリング・フィルター項目をバッグに追加できます。これを 67 ページの図 5 に示します。 項目はセレクターによって識別されます。 大抵の場合、1 つのセレクターは 1 つの項目のみを識別しますが、そうでない場合もあります。 指定したセレクターのあるデータ項目が既にバッグに入っている場合、そのセレクターの追加インスタンスがバッグの末尾に追加されます。

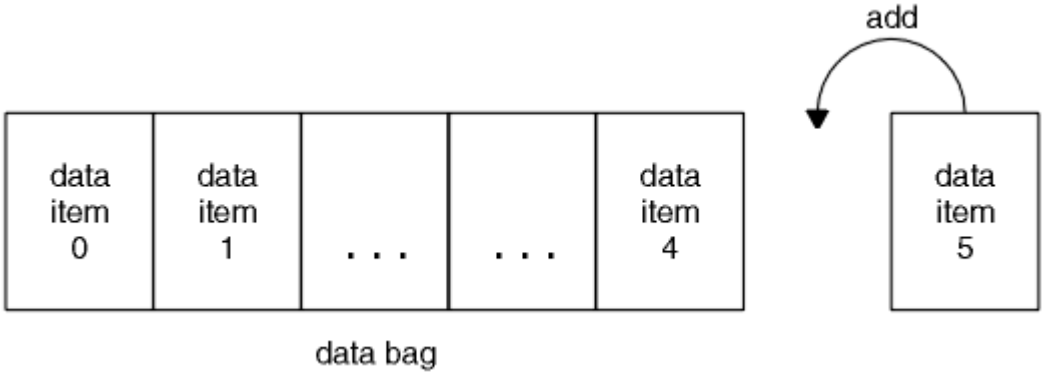


図 5. データ項目の追加

以下のように、mqAdd* 呼び出しを使用してバッグにデータ項目を追加します。

- 整数項目を追加するには、mqAddInteger で説明されているように mqAddInteger 呼び出しを使用します。
- 64 ビット整数項目を追加するには、mqAddInteger64 で説明されているように mqAddInteger64 呼び出しを使用します。
- 整数フィルター項目を追加するには、mqAddIntegerFilter で説明されているように mqAddIntegerFilter 呼び出しを使用します。
- 文字ストリング項目を追加するには、mqAddString で説明されているように mqAddString 呼び出しを使用します。
- ストリング・フィルター項目を追加するには、mqAddStringFilter で説明されているように mqAddStringFilter 呼び出しを使用します。
- バイト・ストリング項目を追加するには、mqAddByteString で説明されているように mqAddByteString 呼び出しを使用します。
- バイト・ストリング・フィルター項目を追加するには、mqAddByteStringFilter で説明されているように mqAddByteStringFilter 呼び出しを使用します。

バッグへのデータ項目の追加についてさらに詳しくは 66 ページの『システム項目と MQAI』を参照してください。

Multi バッグへの照会コマンドの追加

mqAddInquiry 呼び出しは、照会コマンドをバッグに追加するために使用されます。 呼び出しは、特に管理を目的としたものであるため、管理バッグでのみ使用できます。 これにより、IBM MQ から照会する属性のセレクターを指定することができます。

mqAddInquiry 呼び出しの詳細な説明については、mqAddInquiry を参照してください。

Multi データ項目のフィルター処理および照会

MQAI を使用して IBM MQ オブジェクトの属性の問い合わせを行う場合、以下の 2 つの方法で、プログラムに返されるデータを制御できます。

- mqAddInteger および mqAddString 呼び出しを使用して、返されるデータを **フィルター処理** することができます。 この方法では、以下のように、*Selector* と *ItemValue* のペアを指定します。

```
mqAddInteger(inputbag, MQIA_Q_TYPE, MQQT_LOCAL)
```

  この例では、キュー・タイプ (*Selector*) がローカル (*ItemValue*) でなければならないということを指定しており、この指定は、問い合わせているオブジェクト (この場合はキュー) の属性と一致していなければなりません。

  フィルター処理できる他の属性は、25 ページの『IBM MQ プログラマブル・コマンド・フォーマットの概要』で示されている PCF Inquire* コマンドに対応しています。 例えば、チャネルの属性に関する問い合わせを行うには、製品資料で Inquire Channel コマンドについて参照してください。 Inquire Channel コマンドの「必須パラメーター」および「オプション・パラメーター」で、フィルター操作に使用できるセレクターを指定します。

- mqAddInquiry 呼び出しを使用して、オブジェクトの特定の属性を **照会** することができます。 これでは、対象とするセレクターを指定します。 セレクターを指定しないと、オブジェクトのすべての属性が返されます。

以下は、キューの属性のフィルター処理および照会の例です。

```
/* Request information about all queues */
mqAddString(adminbag, MQCA_Q_NAME, "*")

/* Filter attributes so that local queues only are returned */
mqAddInteger(adminbag, MQIA_Q_TYPE, MQQT_LOCAL)

/* Query the names and current depths of the local queues */
mqAddInquiry(adminbag, MQCA_Q_NAME)
mqAddInquiry(adminbag, MQIA_CURRENT_Q_DEPTH)

/* Send inquiry to the command server and wait for reply */
mqExecute(MQCMD_INQUIRE_Q, ...)
```

Multi データ・バッグ内の照会

以下を照会できます。

- mqInquireInteger 呼び出しを使用して整数項目の値を照会します。 mqInquireInteger を参照してください。
- mqInquireInteger64 呼び出しを使用して 64 ビット整数項目の値を照会します。 mqInquireInteger64 を参照してください。
- mqInquireIntegerFilter 呼び出しを使用して整数フィルター項目の値を照会します。 mqInquireIntegerFilter を参照してください。
- mqInquireString 呼び出しを使用して文字ストリング項目の値を照会します。 mqInquireString を参照してください。
- mqInquireStringFilter 呼び出しを使用してストリング・フィルター項目の値を照会します。 mqInquireStringFilter を参照してください。
- mqInquireByteString 呼び出しを使用してバイト・ストリング項目の値を照会します。 mqInquireByteString を参照してください。
- mqInquireByteStringFilter 呼び出しを使用してバイト・ストリング・フィルター項目の値を照会します。 mqInquireByteStringFilter を参照してください。
- mqInquireBag 呼び出しを使用してバッグ・ハンドルの値を照会します。 mqInquireBag を参照してください。

mqInquireItemInfo 呼び出しを使用して、特定項目のタイプ (整数、64 ビット整数、整数フィルター、文字ストリング、ストリング・フィルター、バイト・ストリング、バイト・ストリング・フィルター、またはバッグ・ハンドル) を照会することもできます。 mqInquireItemInfo を参照してください。

Multi バッグ内の情報の変更

MQAI では、mqSet* 呼び出しを使用してバッグ内の情報を変更できます。 以下のことが可能です。

1. バッグ内のデータ項目を変更します。 変更される項目のオカレンスを識別することで、パラメーターの個々のインスタンスを置換できます (69 ページの図 6 を参照)。

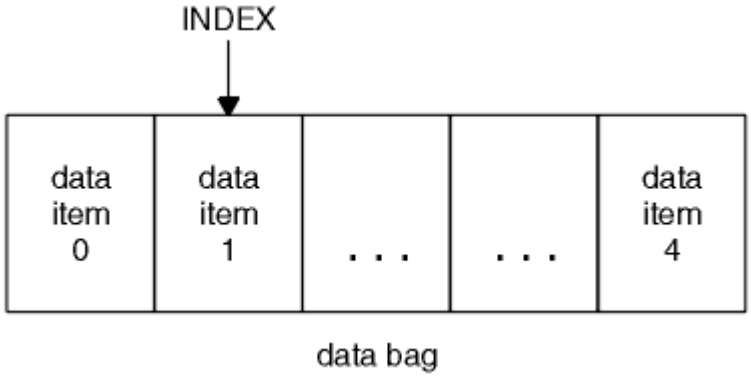


図 6. 単一データ項目の変更

2. 指定したセレクターの既存のオカレンスをすべて削除し、新規オカレンスをバッグの末尾に追加します。 (69 ページの図 7 を参照してください。) 特殊な索引値を使用すると、パラメーターの **すべての** インスタンスを置換できます。

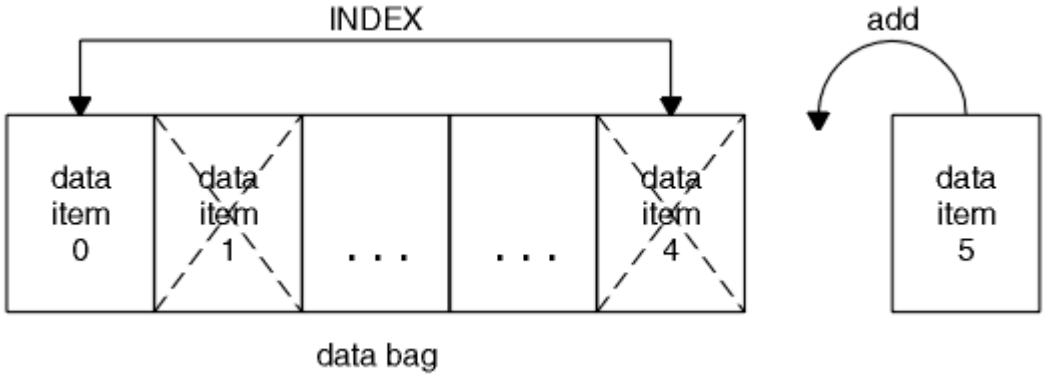


図 7. すべてのデータ項目の変更

注 : 索引ではバッグ内での挿入順序が保持されますが、他のデータ項目の索引に影響を与える可能性があります。

mqSetInteger 呼び出しを使用すると、バッグ内の整数項目を変更できます。 mqSetInteger64 呼び出しを使用すると、64 ビット整数項目を変更できます。 mqSetIntegerFilter 呼び出しを使用すると、整数フィルター項目を変更できます。 mqSetString 呼び出しを使用すると、文字ストリング項目を変更できます。 mqSetStringFilter 呼び出しを使用すると、ストリング・フィルター項目を変更できます。 mqSetByteString 呼び出しを使用すると、バイト・ストリング項目を変更できます。 mqSetByteStringFilter 呼び出しを使用すると、バイト・ストリング・フィルター項目を変更できます。 これらの呼び出しを使用し、指定したセレクターの既存のオカレンスをすべて削除し、新規をバッグの最後に追加することができます。 データ項目はユーザー項目またはシステム項目のいずれかです。

これらの呼び出しの詳細説明については、以下を参照してください。

- mqSetInteger
- mqSetInteger64
- mqSetIntegerFilter
- mqSetString

- mqSetStringFilter
- mqSetByteString
- mqSetByteStringFilter

Multi *mqClearBag* 呼び出しによるバッグのクリア

mqClearBag 呼び出しは、ユーザー・バッグからすべてのユーザー項目を削除し、システム項目を初期値にリセットします。 バッグ内に入っているシステム・バッグも削除されます。

mqClearBag 呼び出しの詳細な説明については、mqClearBag を参照してください。

Multi *mqTruncateBag* 呼び出しによるバッグの切り捨て

mqTruncateBag 呼び出しは、バッグの最後から、つまり最新の追加項目から順に項目を削除して、ユーザー・バッグのユーザー項目数を減らします。 例えば、同じヘッダー情報を使用して複数のメッセージを生成する時に、この呼び出しを使用できます。

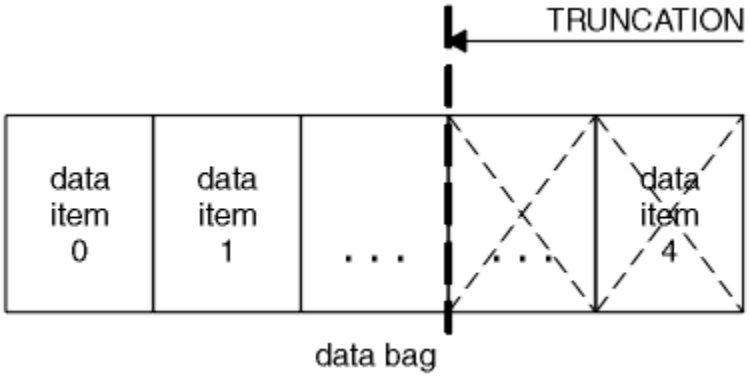


図 *8.* バッグの切り捨て

mqTruncateBag 呼び出しの詳細な説明については、mqTruncateBag を参照してください。

Multi バッグおよびバッファーの変換

アプリケーション間でデータを送信する場合、まずメッセージ・データがバッグに入れられます。 次に、バッグにあるデータが mqBagToBuffer 呼び出しにより PCF メッセージに変換されます。 PCF メッセージが MQPUT 呼び出しにより必須キューに送信されます。 これについては、図 70 ページの図 9 に示してあります。 mqBagToBuffer 呼び出しの詳細な説明については、mqBagToBuffer を参照してください。

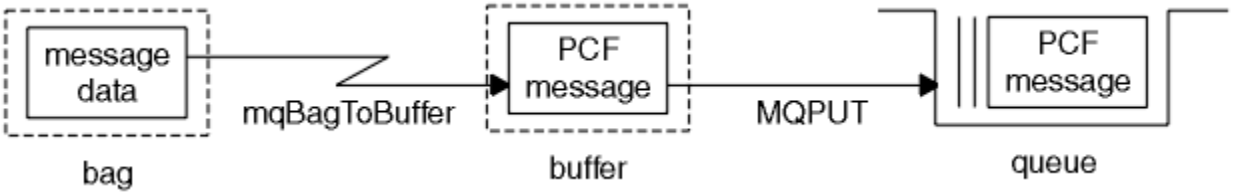


図 *9.* バッグの *PCF* メッセージへの変換

データを受信する場合は、MQGET 呼び出しによりメッセージがバッファーに受信されます。 バッファーに有効な PCF メッセージが含まれる場合は、バッファーにあるデータが mqBufferToBag 呼び出しによりバッグに変換されます。 これについては、図 71 ページの図 10 に示してあります。 mqBufferToBag 呼び出しの詳細な説明については、mqBufferToBag を参照してください。

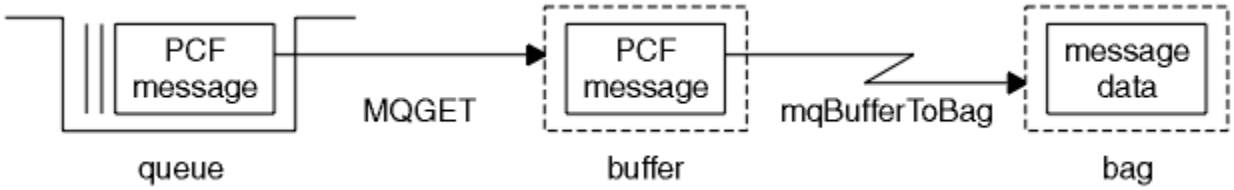


図 *10. PCF* メッセージのバッグ形式への変換

Multi データ項目のカウント

mqCountItems 呼び出しは、データ・バッグに保管されているユーザー項目またはシステム項目、あるいはその両方の数をカウントし、その数を返します。 例えば、 `mqCountItems( Bag, 7, ...)`は、セレクターが 7 のバッグ内の項目数を返します。 これは、個々のセレクター別、ユーザー・セレクター別、システム・セレクター別、またはすべてのセレクター別に項目をカウントできます。

注 : この呼び出しは、バッグにある固有のセレクター数ではなく、データ項目数をカウントします。 1 つのセレクターは複数回出現する可能性があるため、バッグ内の固有のセレクターのほうがデータ項目より少ない場合があります。

mqCountItems 呼び出しの詳細な説明については、mqCountItems を参照してください。

Multi データ項目の削除

いくつかの方法でバッグから項目を削除することができます。 以下のことが可能です。

- バッグから 1 つ以上のユーザー項目を削除する。 詳しくは、71 ページの『mqDeleteItem 呼び出しによるデータ項目のバッグからの削除』を参照してください。
- バッグからすべてのユーザー項目を削除する。つまり、バッグをクリアします。 詳細については、70 ページの『mqClearBag 呼び出しによるバッグのクリア』を参照してください。
- バッグの末尾からユーザー項目を削除する。つまり、バッグを切り捨てます。 詳しくは、70 ページの『mqTruncateBag 呼び出しによるバッグの切り捨て』を参照してください。

Multi *mqDeleteItem* 呼び出しによるデータ項目のバッグからの削除

mqDeleteItem 呼び出しは、1 つ以上のユーザー項目をバッグから削除します。 索引は、次のいずれかを削除するために使用されます。

1. 指定されたセレクターの単一オカレンス。 (71 ページの図 11 を参照してください。)

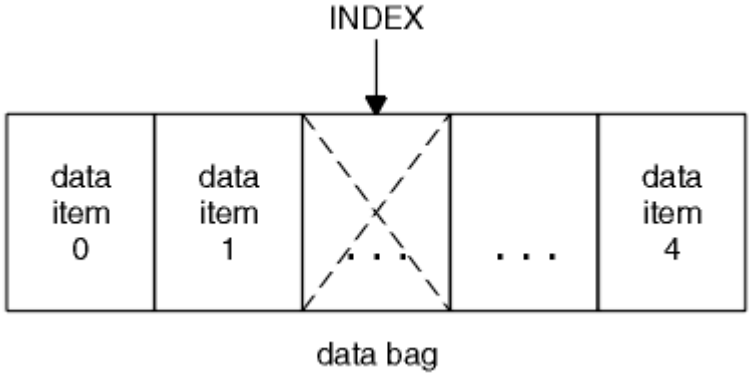


図 *11.* 単一データ項目の削除

または

2. 指定されたセレクターのすべてのオカレンス。 (72 ページの図 12 を参照してください。)

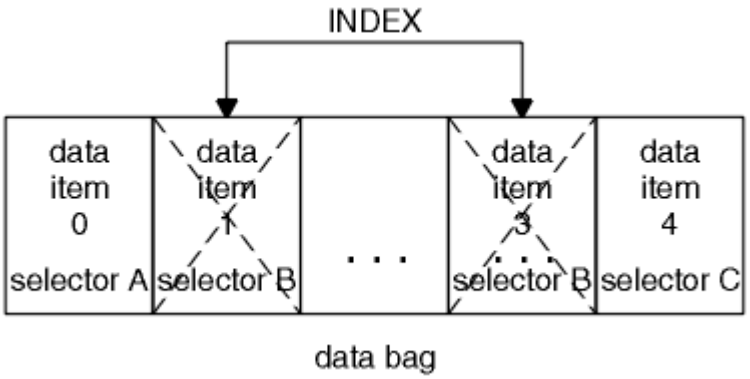


図 12. すべてのデータ項目の削除

注 : 索引ではバッグ内での挿入順序が保持されますが、他のデータ項目の索引に影響を与える可能性があります。 例えば、索引は項目の削除により残されたギャップを埋めるために再編成されるので、mqDeleteItem 呼び出しでは、削除した項目の直後にあるデータ項目の索引値は保持されません。

mqDeleteItem 呼び出しの詳細な説明については、mqDeleteItem を参照してください。

Multi mqExecute 呼び出しを使用した qm コマンド・サーバーへの管理コマンドの送信

データ・バッグを作成して内容を設定したら、mqExecute 呼び出しを使用して、キュー・マネージャーのコマンド・サーバーへ管理コマンド・メッセージを送信することができます。 これにより、コマンド・サーバーとのやり取りが処理され、応答がバッグに返されます。

データ・バッグを作成し内容を設定した後、管理コマンド・メッセージをキュー・マネージャーのコマンド・サーバーに送信することができます。 これを行う最も簡単な方法は、mqExecute 呼び出しを使用することです。 mqExecute 呼び出しは非持続メッセージとして管理コマンド・メッセージを送信し、応答を待機します。 応答は応答バッグで返されます。 これには、いくつかの IBM MQ オブジェクトや、例えば一連の PCF エラー応答メッセージなどに関連した属性の情報が入れられます。 そのため、応答バッグに戻りコードのみが含まれる場合もあれば、ネストされたバッグ が含まれる場合もあります。

応答メッセージは、システムによって作成されたシステム・バッグに入れられます。 例えば、オブジェクトの名前に関する問い合わせの場合、それらのオブジェクト名を保持するためのシステム・バッグが作成され、そのバッグがユーザー・バッグに挿入されます。 そして、これらのバッグへのハンドルが応答バッグに挿入され、ネストされたバッグにはセレクター MQHA_BAG_HANDLE によってアクセスできるようになります。 システム・バッグは、削除されなければ、応答バッグが削除されるまでストレージに置かれたままになります。

73 ページの図 13 にネスト の概念を示します。

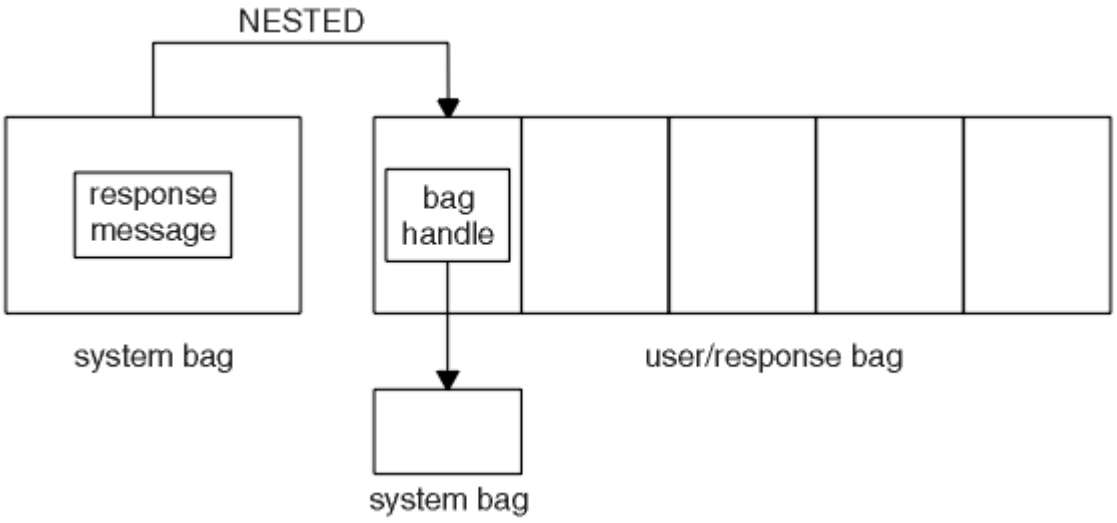


図 *13.* ネスト

mqExecute 呼び出しへの入力として、以下を指定する必要があります。

- MQI 接続ハンドル。
- 実行されるコマンド。 これは、MQCMD_* 値のいずれかになります。

  注 : この値が MQAI によって認識されない場合でも、この値は受け入れられます。 ただし、mqAddInquiry 呼び出しが使用されて値がバッグに挿入された場合、このパラメーターは、MQAI によって認識される INQUIRE コマンドでなければなりません。 つまり、パラメーターは MQCMD_INQUIRE_* の形式でなければなりません。
- オプションとして、呼び出しの処理を制御するオプションを含むバッグのハンドル。 ここでは、各応答メッセージを MQAI が待機する最大時間 (ミリ秒) を指定することもできます。
- 発行する管理コマンドの詳細を含む管理バッグのハンドル。
- 応答メッセージを受け取る応答バッグのハンドル。

以下のハンドルは、オプションです。

- 管理コマンドが置かれるキューのオブジェクト・ハンドル。

  オブジェクト・ハンドルが指定されない場合、管理コマンドは、現在接続されているキュー・マネージャーに属する SYSTEM.ADMIN.COMMAND.QUEUE に置かれます。 これはデフォルトです。
- 応答メッセージが置かれるキューのオブジェクト・ハンドル。

  MQAI によって自動的に作成される動的キューに応答メッセージを置くように選択することができます。 作成されたキューは呼び出しの間だけ存在し、mqExecute 呼び出しからの終了時に MQAI によって削除されます。

mqExecute 呼び出しの使用例については、コード例を参照してください。

REST API を使用した管理

administrative REST API を使用して、キュー・マネージャーやキューなどの IBM MQ オブジェクト、Managed File Transfer エージェントおよび転送を管理できます。 JSON 形式の情報が administrative REST API との間で送受信されます。 これらの RESTful API は、よく使用される DevOps や自動化ツールに IBM MQ 管理を組み込むのに役立ちます。

始める前に

注 : V 9.4.0 administrative REST API は、スタンドアロンの IBM MQ Web Server インストール済み環境では使用できません。 administrative REST API を実行する IBM MQ コンポーネントのインストール・オプションについて詳しくは、 IBM MQ Console および REST API を参照してください。

使用可能な REST リソースに関する参照情報は、administrative REST API リファレンスを参照してください。

手順

- 74 ページの『administrative REST API の使用開始』
- 77 ページの『administrative REST API の使用』
- 79 ページの『REST API によるリモート管理』
- 83 ページの『REST API タイム・スタンプ』
- 83 ページの『REST API エラー処理』
- 86 ページの『REST API ディスカバリー』
- 87 ページの『REST API 各国語サポート』

administrative REST API の使用開始

administrative REST API の使用をすぐに開始して、cURL を使用するいくつかの要求例を試すと、キューを作成、更新、表示、および削除することができます。

始める前に

administrative REST API を使い始めるにあたって、このタスクに含まれる例には以下の要件があります。

- この例では、cURL を使用して REST 要求を行い、システム上のキュー・マネージャーに関する情報を表示し、キューを作成、更新、表示、および削除します。 したがって、このタスクを実行するためには、ご使用のシステムに cURL がインストールされている必要があります。
- このタスクを実行するには、**dspmqweb** コマンドを使用するための特定の特権を持っているユーザーである必要があります。
  - z/OS z/OS の場合、**dspmqweb** コマンドを実行する権限と、`mqwebuser.xml` ファイルに対する書き込みアクセス権限を持っている必要があります。
  - Multi 他のすべてのオペレーティング・システムでは、特権ユーザーでなければなりません。

IBM i IBM i では、コマンドを QSHELL で実行する必要があります。

手順

1. administrative REST API、 administrative REST API for MFT、 messaging REST API、または IBM MQ Console で使用するために mqweb サーバーが構成されていることを確認します。
   基本レジストリーを使用した mqweb サーバーの構成について詳しくは、 mqweb サーバーの基本構成を参照してください。
2. z/OS
   z/OS では、 **dspmqweb** コマンドを使用できるように WLP_USER_DIR 環境変数を設定します。 次のコマンドを入力して、mqweb サーバー構成を指すように変数を設定します。

   ```
   export WLP_USER_DIR=WLP_user_directory
   ```

   ここで、 *WLP_user_directory* は、 `crtmqweb` に渡されるディレクトリーの名前です。 例 :

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

   詳しくは、mqweb サーバーの作成を参照してください。
3. 決定する REST API 次のコマンドを入力して URL にアクセスします。

   ```
   dspmqweb status
   ```

以下の手順の例では、REST API URL はデフォルトの URL です https://localhost:9443/ibmmq/rest/v1/。 デフォルト以外の URL を使用している場合は、以下の手順の URL を置き換えてください。

4. mqadmin ユーザーの基本認証を使用して、qmgr リソースで GET 要求を試行します。

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

5. mqsc リソースを使用して、キューを作成、表示、変更、および削除します。

この例では、キュー・マネージャー QM1 を使用します。 同じ名前のキュー・マネージャーを作成するか、ご使用のシステム上の既存のキュー・マネージャーに置き換えてください。

a) mqsc リソースで POST 要求を行って、ローカル・キューを作成します。

要求の本体で、新しいキューの名前を Q1 に設定します。 基本認証が使用されます。また、cURL REST 要求に、任意の値を含んだ ibm-mq-rest-csrf-token HTTP ヘッダーが設定されます。 この追加ヘッダーは、POST、PATCH、および DELETE 要求に必要です。

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

b) mqsc リソースに対して POST 要求を実行して、手順 75 ページの『5.a』 で作成したローカル・キューを表示します。

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

c) mqsc リソースに対して POST 要求を実行して、キューの説明を更新します。

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

d) mqsc リソースに対して POST 要求を実行して、新しいキューの説明を表示します。 応答に説明フィールドが含まれるように、要求本文に **responseParameters** 属性を指定します。

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

e) mqsc リソースに対して POST 要求を実行して、キューを削除します。

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

f) mqsc リソースで POST 要求を行って、キューが削除されたことを検証します。

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

次のタスク

- この例では、基本認証を使用して要求を保護します。 代わりに、トークン・ベース認証またはクライアント・ベース認証を使用することもできます。 詳しくは、REST API でのクライアント証明書認証の使用、IBM MQ Console、および REST API でのトークン・ベースの認証の使用を参照してください。

- administrative REST API の使用法と照会パラメーターで URL を構成する方法について詳しくは、77 ページの『administrative REST API の使用』を参照してください。
- 使用可能な administrative REST API リソースと使用可能なすべてのオプション照会パラメーターのための参照情報については、administrative REST API のリファレンスを参照してください。
- administrative REST API を使用してリモート・システム上の IBM MQ オブジェクトを管理する方法については、79 ページの『REST API によるリモート管理』を参照してください。
- MFT で administrative REST API を使用する方法については、76 ページの『REST API for MFT の概要』を参照してください。
- IBM MQ メッセージング用の RESTful インターフェースの messaging REST API をディスカバーします。 REST API を使用したメッセージング
- ブラウザー・ベースの GUI である IBM MQ Console については、90 ページの『IBM MQ Console を使用した管理』を参照してください。

REST API for MFT の概要

administrative REST API for Managed File Transfer の使用をすぐに開始していくつかの要求例を試すと、MFT エージェント状況を表示したり、転送のリストを表示したりできます。

始める前に

- この例では、cURL を使用して REST 要求を送信することで、転送のリストを表示し、MFT エージェント状況を表示します。 したがって、このタスクを実行するためには、ご使用のシステムに cURL がインストールされている必要があります。
- このタスクを実行するには、**dspmqweb** コマンドを使用するための特定の特権を持っているユーザーである必要があります。
  - z/OS z/OS の場合、**dspmqweb** コマンドを実行する権限と、`mqwebuser.xml` ファイルに対する書き込みアクセス権限を持っている必要があります。
  - Multi 他のすべてのオペレーティング・システムでは、特権ユーザーでなければなりません。

手順

1. mqweb サーバーが administrative REST API for MFT 用に構成されていることを確認します。
   - administrative REST API、administrative REST API for MFT、messaging REST API、または IBM MQ Console で使用するために mqweb サーバーが構成されていることを確認します。基本レジストリーを使用した mqweb サーバーの構成について詳しくは、 mqweb サーバーの基本構成を参照してください。
   - mqweb サーバーが構成されている場合、mqweb サーバーの基本構成のステップ 8 が完了して、administrative REST API for MFT が有効になっていることを確認します。
2. z/OS
   z/OS では、 **dspmqweb** コマンドを使用できるように WLP_USER_DIR 環境変数を設定します。 次のコマンドを入力して、mqweb サーバー構成を指すように変数を設定します。

   ```
   export WLP_USER_DIR=WLP_user_directory
   ```

   ここで、 *WLP_user_directory* は、 `crtmqweb` に渡されるディレクトリーの名前です。 例 :

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

   詳しくは、mqweb サーバーの作成を参照してください。
3. 決定する REST API 次のコマンドを入力して URL にアクセスします。

   ```
   dspmqweb status
   ```

以下の手順の例では、REST API URL はデフォルトの URL です `https://localhost:9443/ibmmq/rest/v1/`。 デフォルト以外の URL を使用している場合は、以下の手順の URL を置き換えてください。

4. 名前、タイプ、状態など、すべてのエージェントに関する基本的な詳細を返す GET 要求を `agent` リソースに対して行います。

```
curl -k https://localhost:9443/ibmmq/rest/v2/admin/mft/agent/ -X GET -u mftadmin:mftadmin
```

5. **fteCreateTransfer** コマンドを使用して、表示する転送をいくつか作成します。

   mqweb サーバーは、転送に関する情報をキャッシュに入れておき、要求されたときにその情報を返します。 このキャッシュは、mqweb サーバーが再始動するとリセットされます。 サーバーが再始動されたかどうかは、`console.log` ファイルと `messages.log` ファイルを表示するか、z/OS では開始タスクからの出力を参照することで確認できます。

6. mqweb サーバーが開始してから行われた最大 4 つの転送の詳細を返す GET 要求を `transfer` リソースに対して行います。

```
curl -k https://localhost:9443/ibmmq/rest/v2/admin/mft/transfer?limit=4 -X GET -u
mftadmin:mftadmin
```

次のタスク

- この例では、基本認証を使用して要求を保護します。 代わりに、トークン・ベース認証またはクライアント・ベース認証を使用することもできます。 詳しくは、REST API でのトークン・ベースの認証の使用および REST API と IBM MQ Console でのクライアント証明書認証の使用を参照してください。
- administrative REST API の使用法と照会パラメーターで URL を構成する方法について詳しくは、77 ページの『administrative REST API の使用』を参照してください。
- 使用可能な administrative REST API for MFT リソースと使用可能なすべてのオプション照会パラメーターのための参照情報については、administrative REST API のリファレンスを参照してください。
- IBM MQ メッセージング用の RESTful インターフェースの messaging REST API をディスカバーします。 REST API を使用したメッセージング
- ブラウザー・ベースの GUI である IBM MQ Console については、90 ページの『IBM MQ Console を使用した管理』を参照してください。

administrative REST API の使用

administrative REST API を使用するときは、キュー・マネージャーやキューなどのさまざまな IBM MQ オブジェクトを表す URL に対して HTTP メソッドを呼び出します。 HTTP メソッド (POST など) は、URL で表されるオブジェクトに対して実行する操作のタイプを表します。 その操作に関する詳細は、HTTP メソッドのペイロードの一部として JSON 形式で指定したり、照会パラメーター内にエンコードしたりします。 操作の実行結果に関する情報は、一般には HTTP 応答の本体として返されます。

始める前に

administrative REST API を使用する前に、以下の点を考慮してください。

- administrative REST API を使用するには、mqweb サーバーで認証を行う必要があります。 HTTP 基本認証、クライアント証明書認証、またはトークン・ベースの認証を使用して、認証を行うことができます。 これらの認証方式の使用方法について詳しくは、IBM MQ Console および REST API セキュリティーを参照してください。
- REST API は大文字小文字を区別します。 例えば、キュー・マネージャーの名前が `qmgr1` である場合に以下の URL に対して HTTP GET を実行しても、情報は表示されません。

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

- IBM MQ オブジェクト名で使用できる文字の一部は、URL 内に直接エンコードすることができません。 そのような文字を正しくエンコードするためには、適切な URL エンコード方式を使用する必要があります。

  - スラッシュ / は、%2F としてエンコードする必要があります。
  - % 記号は、%25 としてエンコードする必要があります。
- 一部のブラウザーの動作を考慮し、オブジェクトの名前をピリオドやスラッシュの文字のみにすることは避けてください。

このタスクについて

REST API を使用してオブジェクトに対して操作を実行する場合は、まず、そのオブジェクトを表す URL を構成する必要があります。 どの URL も、要求を送信するホスト名とポートを示す接頭部で始まります。 URL の残りの部分は、特定のオブジェクト、またはオブジェクトのセットを示します。これらはリソースと呼ばれます。

リソースに対して実行する操作によって、URL に照会パラメーターが必要かどうかが規定されます。 また、使用する HTTP メソッドや、追加情報を JSON 形式で URL に送信したり URL から戻したりするかどうかも決まります。 追加情報を HTTP 要求に含める場合もあれば、HTTP 応答の一部として追加情報が返される場合もあります。

URL を構成し、必要に応じて、HTTP 要求に含めて送信する JSON ペイロードを作成したら、IBM MQ に HTTP 要求を送信できます。 選択したプログラミング言語に組み込んだ HTTP 実装を使用して、要求を送信できます。 cURL などのコマンド・ライン・ツール、Web ブラウザーや Web ブラウザー・アドオンを使用して、要求を送信することもできます。

重要 : 少なくとも、手順 78 ページの『1.a』 と 78 ページの『1.b』 を実行する必要があります。

手順

1. URL を構成します。
   a) 以下のコマンドを入力して、接頭部 URL を特定します。

   ```
   dspmqweb status
   ```

   使用する URL には、 `/ibmmq/rest/` 句が含まれます。

   b) URL パスにリソースを追加します。

   使用可能な IBM MQ リソースは次のとおりです。

   - /admin/installation
   - /admin/qmgr
   - /admin/キュー
   - /admin/subscription (/admin/サブスクリプション)
   - /管理/チャネル
   - /action/qmgr/{qmgrName}/mqsc

   使用可能な Managed File Transfer リソースは次のとおりです。

   - /admin/エージェント
   - /admin/transfer
   - /admin/モニター

   例えば、キュー・マネージャーと対話するには、`/qmgr` を接頭部 URL に追加して、以下の URL を作成します。

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

   c) オプション: オプションのパス・セグメントを URL に追加します。

   各オブジェクト・タイプの参照情報では、オプションの各セグメントを中括弧 { } で囲むことで URL 内に指定できます。

例えば、キュー・マネージャー名 QM1 を URL に追加して、以下の URL を作成します。

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

d) オプション: オプションの照会パラメーターを URL に追加します。

疑問符 (?) を追加します。 変数名、等号 =、および URL に対する値または値のリスト。

例えば、キュー・マネージャー QM1 のすべての属性を要求するには、以下の URL を作成します。

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

e) オプションの照会パラメーターをさらに URL に追加します。

アンパーサンド & を URL に追加してから、 ステップ d をもう一度実行してください。

2. URL に対して適切な HTTP メソッドを起動します。 オプションの JSON ペイロードを指定し、認証のために適切なセキュリティー資格認定を提供します。 以下に例を示します。
   - 選択したプログラミング言語の HTTP/REST 実装を使用します。
   - REST クライアント・ブラウザー・アドオンや cURL などのツールを使用します。

REST API によるリモート管理

REST API を使用して、リモート・キュー・マネージャーと、それらのキュー・マネージャーに関連付けられている IBM MQ オブジェクトを管理できます。 このリモート管理には、同じシステム上にあるが、mqweb サーバーと同じ IBM MQ インストール済み環境にはないキュー・マネージャーが含まれます。 これにより、REST API を使用して、mqweb サーバーが稼働している 1 つのインストール済み環境のみで IBM MQ ネットワーク全体を管理できます。 リモート・キュー・マネージャーを管理するには、mqweb サーバーと同じインストール済み環境内の少なくとも 1 つのキュー・マネージャーがゲートウェイ・キュー・マネージャーとして機能するように administrative REST API ゲートウェイを構成する必要があります。 これで、REST API リソース URL でリモート・キュー・マネージャーを指定して、指定した管理操作を実行できるようになります。

始める前に

リモート管理は、administrative REST API ゲートウェイを無効にすることによって防止できます。 詳細については、administrative REST API ゲートウェイの構成を参照してください。

administrative REST API ゲートウェイを使用するには、以下の条件を満たす必要があります。

- mqweb サーバーを構成し、開始する必要があります。 mqweb サーバーの構成および開始の詳細については、74 ページの『administrative REST API の使用開始』を参照してください。
- ゲートウェイ・キュー・マネージャーとして構成するキュー・マネージャーは、mqweb サーバーと同じインストール済み環境にある必要があります。
- 管理するリモート・キュー・マネージャーは、IBM MQ 8.0 以降である必要があります。
- 要求で指定する属性が要求の送信先システムで有効であることを確認する必要があります。 例えば、ゲートウェイ・キュー・マネージャーが Windows 上にあり、リモート・キュー・マネージャーが z/OS 上にある場合、queue リソースで HTTP GET 要求に対して `dataCollection.statistics` 属性が返されるように要求することはできません。
- 要求で指定する属性が要求の送信先 IBM MQ のレベルで有効であることを確認する必要があります。 例えば、リモート・キュー・マネージャーが IBM MQ 8.0 を実行している場合、queue リソースで HTTP GET 要求に対して `extended.enableMediaImageOperations` 属性が返されるように要求することはできません。
- 以下のサポートされる REST リソースの 1 つを使用する必要があります。
  - `/queue`
  - `/subscription`
  - `/channel`

– /mqsc
– /qmgr

  リモート・キュー・マネージャーを照会すると、/qmgr リソースは、属性のサブセット (name、status.started、status.channelInitiatorState、status.ldapConnectionState、status.connectionCount、および status.publishSubscribeState) のみを返します。

このタスクについて

administrative REST API ゲートウェイを使用してリモート・キュー・マネージャーを管理するには、キュー・マネージャーをリモート管理用に準備する必要があります。 つまり、ゲートウェイ・キュー・マネージャーとリモート・キュー・マネージャーの間に伝送キュー、リスナー、および送信側チャネルと受信側チャネルを構成する必要があります。 これにより、リソース URL でキュー・マネージャーを指定することによって、リモート・キュー・マネージャーに REST 要求を送信できるようになります。 ゲートウェイ・キュー・マネージャーを指定するには、**setmqweb** コマンドを使用して mqRestGatewayQmgr 属性をゲートウェイ・キュー・マネージャーの名前に設定するか、要求と共に送信されるヘッダーでゲートウェイ・キュー・マネージャーの名前を送信します。 要求は、ゲートウェイ・キュー・マネージャーを経由してリモート・キュー・マネージャーに送信されます。 ゲートウェイ・キュー・マネージャーとして使用されたキュー・マネージャーを示すヘッダーと共に応答が返されます。

手順

1. ゲートウェイ・キュー・マネージャーと管理するリモート・キュー・マネージャー間の通信を構成します。 これらの構成ステップは、runmqsc および PCF によるリモート管理を構成するために必要なステップと同じです。

   これらのステップの詳細については、196 ページの『リモート管理のためのキュー・マネージャーの構成』を参照してください。
2. リモート・キュー・マネージャーのセキュリティーを構成します。

   a) リモート・キュー・マネージャーが実行されるシステム上に、関連するユーザー ID が存在することを確認します。 リモート・システムに存在しなければならないユーザー ID は、REST API ユーザーの役割によって異なります。

   - REST API ユーザーが MQWebAdmin または MQWebAdminRO グループに含まれている場合は、mqweb サーバーを開始したユーザー ID がリモート・システム上に存在している必要があります。 IBM MQ Appliance では、mqweb サーバーを開始するユーザーは mqsystem です。
   - REST API ユーザーが MQWebUser グループに含まれている場合は、その REST API ユーザー ID がリモート・システム上に存在している必要があります。

   b) リモート・キュー・マネージャー上の適切な REST API リソースにアクセスするために必要なレベルの権限が、関連するユーザー ID に付与されていることを確認します。

   - メッセージを SYSTEM.ADMIN.COMMAND.QUEUE に書き込む権限。
   - メッセージを SYSTEM.REST.REPLY.QUEUE に書き込む権限。
   - リモート管理用に定義された伝送キューにアクセスする権限。
   - キュー・マネージャー属性を表示する権限。
   - REST 要求を実行する権限。 詳細については、REST API リソースの参照トピックのセキュリティー要件に関するセクションを参照してください。
3. ゲートウェイとして使用するローカル・キュー・マネージャーを構成します。 デフォルトのゲートウェイ・キュー・マネージャーを構成するか、HTTP ヘッダーでゲートウェイ・キュー・マネージャーを指定するか、または両方のアプローチの組み合わせを使用することができます。

   - デフォルトのゲートウェイ・キュー・マネージャーを構成するには、**setmqweb** コマンドを使用します。

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

     ここで、*qmgrName* は、ゲートウェイ・キュー・マネージャーの名前です。

このゲートウェイ・キュー・マネージャーは、以下の記述の両方が当てはまる場合に使用されます。

- REST 要求の ibm-mq-rest-gateway-qmgr ヘッダーでキュー・マネージャーが指定されていない。
- REST API リソース URL で指定されたキュー・マネージャーがローカル・キュー・マネージャーではない。

- すべての REST 要求のゲートウェイ・キュー・マネージャーを構成するには、HTTP ヘッダー ibm-mq-rest-gateway-qmgr をゲートウェイ・キュー・マネージャーの名前に設定します。

4. 管理するリモート・キュー・マネージャーの名前をリソース URL に含めます。
   例えば、キューのリストをリモート・キュー・マネージャー remoteQM から取得するには、以下の URL を使用します。

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

タスクの結果

ibm-mq-rest-gateway-qmgr ヘッダーが REST 応答と共に返されます。 このヘッダーは、ゲートウェイ・キュー・マネージャーとして使用されたキュー・マネージャーを指定しています。

administrative REST API を使用してリモート・キュー・マネージャーを管理する操作がうまくいかない場合は、以下のようにしてください。

- リモート・キュー・マネージャーが実行中であることを確認します。
- リモート・システムでコマンド・サーバーが実行中であることを確認します。
- チャネル切断間隔が終了していないことを確認します。 例えば、チャネルが開始してしばらくしてからシャットダウンした場合です。 これは、チャネルを手動操作で開始した場合に特に重要です。

例

以下の例では、2 台のマシンに 3 つの IBM MQ インストール済み環境があります。 Machine 1 には、Installation 1 と Installation 2 があります。 Machine 2 には、Installation 3 があります。 mqweb サーバーは Installation 1 用に構成されています。 各インストール済み環境には 1 つのキュー・マネージャーがあり、これらのキュー・マネージャーはリモート管理用に構成されています。 つまり、以下のリスナー、チャネル、およびキューが構成され、開始されています。

- Machine 1 上の Installation 1 のキュー・マネージャー QM1:
  - 送信側チャネル QM1.to.QM2
  - 受信側チャネル QM2.to.QM1
  - 送信側チャネル QM1.to.QM3
  - 受信側チャネル QM3.to.QM1
  - 伝送キュー QM2
  - 伝送キュー QM3
  - ポート 1414 に構成されたリスナー
- Machine 1 上の Installation 2 のキュー・マネージャー QM2:
  - 送信側チャネル QM2.to.QM1
  - 受信側チャネル QM1.to.QM2
  - 伝送キュー QM1
  - ポート 1415 に構成されたリスナー
- Machine 2 上の Installation 3 のキュー・マネージャー QM3:
  - 送信側チャネル QM3.to.QM1
  - 受信側チャネル QM1.to.QM3

- 伝送キュー QM1
- デフォルトのリスナー

キュー Qon2 が QM2 に定義され、キュー Qon3 が QM3 に定義されています。

ユーザー mquser が両方のマシン上に定義され、REST API で MQWebAdmin 役割が付与され、各キュー・マネージャー上の適切なキューにアクセスする権限が付与されています。

setmqweb コマンドを使用して、キュー・マネージャー QM1 をデフォルトのゲートウェイ・キュー・マネージャーとして構成しています。

次の図は、この構成を示したものです。

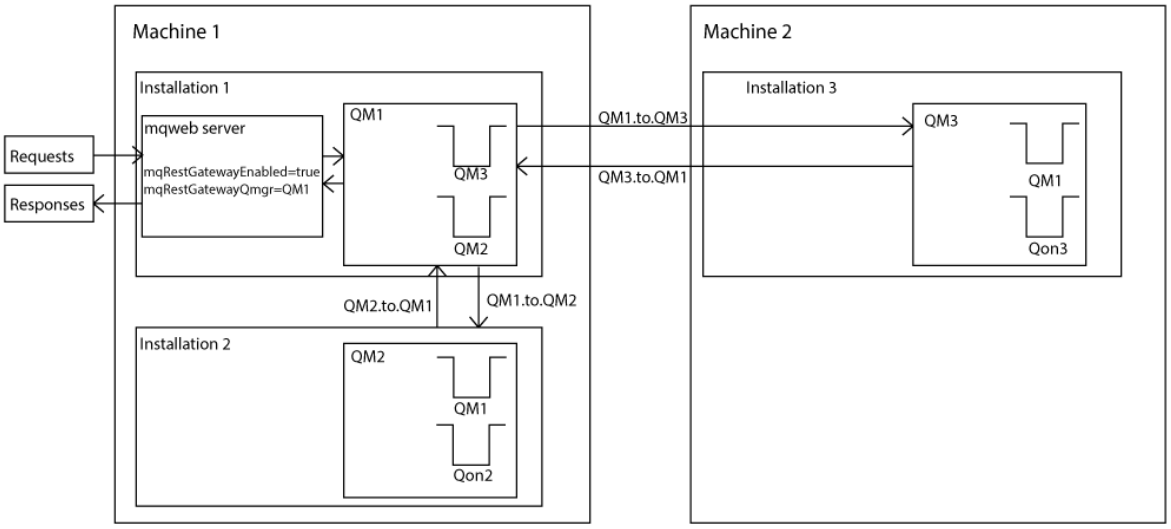


図 *14. REST API* を使用したリモート管理のための構成例の図。

以下の REST 要求が mqweb サーバーに送信されます。

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

以下の応答を受け取ります。

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

以下の REST 要求が mqweb サーバーに送信されます。

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

以下の応答を受け取ります。

```
{
   "queue" :
   [{
        "general": {
            "isTransmissionQueue": true,
            "description": "Transmission queue for remote admin."
        },
        "name": "QM1",
        "type": "local"
    },
   {
        "general": {
            "isTransmissionQueue": false,
            "description": "A queue on QM3."
        },
        "name" : "Qon3",
        "type" : "local"
    }]
}
```

REST API タイム・スタンプ

日時情報が administrative REST API によって戻されるときは、協定世界時 (UTC) で、設定された形式により戻されます。

日時は以下のタイム・スタンプ形式で戻されます。

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

例えば、2012-04-23T18:25:43.000Z となります。 Z は協定世界時 (UTC) でのタイム・ゾーンを示します。

このタイム・スタンプの精度は保証されていません。 例えば、リソース URL で指定されたキュー・マネージャーと同じタイム・ゾーンで mqweb サーバーが始動していない場合、タイム・スタンプは正確でない可能性があります。 また、夏時間調整が必要な場合には、タイム・スタンプは正確でない可能性があります。

REST API エラー処理

REST API は、該当する HTTP 応答コード (例えば「404 (Not Found)」) と JSON 応答を返してエラーを報告します。 200 から 299 の範囲にない HTTP 応答コードは、エラーと見なされます。

エラー応答形式

応答は、UTF-8 エンコードの JSON 形式です。 これには、次のようにネストされた JSON オブジェクトが含まれています。

- JSON オブジェクトの内側に、error という単一の JSON 配列が含まれている。
- その配列の各エレメントは、エラーに関する情報を表す JSON オブジェクトである。 各 JSON オブジェクトには、以下のプロパティーが含まれています。

  タイプ
  ストリング。

  エラーのタイプ。

  messageId
  ストリング。

  MQWBnnnnX 形式のメッセージの固有 ID。 この ID には以下のエレメントがあります。

  MQWB
  メッセージの発生元が IBM MQ Rest API であることを示す接頭部。

  nnnn
  メッセージを識別する固有の番号。

  X
  メッセージの重大度を示す 1 つの文字。

- I: メッセージが単に通知の場合。
- W: メッセージが問題の警告の場合。
- E: メッセージがエラーが発生したことを示している場合。
- S: メッセージが重大エラーが発生したことを示している場合。

メッセージ
ストリング。

エラーの記述。

説明
ストリング。

エラーの説明。

action
ストリング。

エラーを解決するために実行できる手順の説明。

qmgrName
z/OS このフィールドは、キュー・マネージャーがキュー共有グループのメンバーである z/OS でのみ使用可能です。 オプションの **commandScope** 照会パラメーター、または **queueSharingGroupDisposition** 属性を指定しておく必要があります。

ストリング。

エラーが発生したキュー・マネージャーの名前。

このフィールドは、messaging REST API には適用されません。

completionCode
このフィールドは、**type** が pcf、java、または rest の場合にのみ使用可能です。

数値。

障害に関連付けられる MQ 完了コード。

reasonCode
このフィールドは、**type** が pcf、java、または rest の場合にのみ使用可能です。

数値。

障害に関連付けられる MQ 理由コード。

例外
このフィールドは、**type** が java の場合にのみ使用可能です。

Array.

チェーン Java または JMS 例外の配列。 例外配列の各エレメントには、**stackTrace** ストリング配列が含まれています。

stackTrace ストリング配列には、各例外の詳細が複数の行に分割されて含まれています。

キュー共有グループでのエラー
z/OS

キュー共用グループでは、オプションの照会パラメーターの **commandScope** を特定のコマンドに指定できます。 このパラメーターは、コマンドが、キュー共有グループ内の他のキュー・マネージャーに波及することを可能にします。 これらのコマンドのいずれも独自に失敗することがあるので、キュー共有グループで一部のコマンドは成功し、一部のコマンドは失敗する結果になることがあります。

コマンドが部分的に失敗すると、HTTP エラー・コード 500 が返されます。 失敗が発生したキュー・マネージャーごとに、その失敗に関する情報が、error JSON 配列の要素として返されます。 コマンドを正常に実行したキュー・マネージャーごとに、キュー・マネージャーの名前が、success JSON 配列の要素として返されます。

例

- 以下の例は、存在しないキュー・マネージャーに関する情報を取得しようとしたときのエラー応答を示しています。

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

- z/OS 以下の例は、一部のキュー・マネージャーで存在しないキュー共有グループ内のキューを削除しようとしたときのエラー応答を示しています。

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

MFT 要求でのエラー

MFT REST API サービスが無効である場合に MFT REST API を呼び出すと、以下の例外が発生します。

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

MFT REST API サービスが使用可能になっていて、調整キュー・マネージャーが `mqwebuser.xml` ファイルに設定されていない場合は、以下の例外を受け取ります。

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

```
  "reasonCode": 0,
  "type": "rest"
}]}
```

REST API ディスカバリー

REST API に関する資料は、IBM Documentation に Swagger 形式で提供されています。 Swagger は、REST API を文書化するために一般的に使用されている方法です。 REST API の Swagger 資料は、mqweb サーバーで API ディスカバリー機能 (apiDiscovery) を有効にすることで表示できます。

始める前に

Stabilized

重要 : apiDiscovery フィーチャーは安定化されました。 この機能は引き続き使用できます。 現時点では、IBM MQ は mpOpenAPI フィーチャーの使用をサポートしていません。

API ディスカバリーを使用して Swagger 資料を表示するには、mqweb サーバーのセキュリティーを有効にする必要があります。 セキュリティーを有効にするために必要な手順について詳しくは、IBM MQ Console および REST API セキュリティーを参照してください。

手順

1. 以下のいずれかのディレクトリーにある mqwebuser.xml ファイルを見つけます。
   - ALW *MQ_DATA_PATH*/web/installations/*installationName*/servers/mqweb
   - z/OS *WLP_user_directory*/servers/mqweb

     ここで、*WLP_user_directory* は、mqweb サーバー定義を作成するために **crtmqweb** スクリプトを実行したときに指定したディレクトリーです。
2. 適切な XML を mqwebuser.xml ファイルに追加します。
   - <featureManager> タグが mqwebuser.xml ファイルにある場合は、以下の XML を <featureManager> タグ内に追加します。

     <feature>apiDiscovery-1.0</feature>
   - <featureManager> タグが mqwebuser.xml ファイルにない場合は、以下の XML を <server> タグ内に追加します。

     ```
     <featureManager>
          <feature>apiDiscovery-1.0</feature>
     </featureManager>
     ```
3. 次のいずれかの方法で、Swagger 資料を表示します。
   - ブラウザーに以下の URL を入力して、REST API を参照して試すことができる Web ページを表示します。

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

     各要求を認証することに加えて、POST、PATCH、または DELETE 要求ごとに ibm-mq-rest-csrf-token ヘッダーを含める必要があります。 このヘッダーの内容は、ブランクも含め、任意のストリングにすることができます。

     この要求ヘッダーを使用して、要求の認証に使用する資格情報が、資格情報の所有者によって使用されていることを確認できます。 つまり、トークンはクロスサイト・リクエスト・フォージェリー攻撃を防ぐために使用されます。
   - 以下の URL に対して HTTP GET を発行して、REST API 全体について説明している単一の Swagger 2 資料を取得します。

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

この資料は、使用可能な API をプログラムでナビゲートしたいアプリケーションで使用できます。

host
REST API を使用できるホスト名または IP アドレスを指定します。

デフォルト値は `localhost` です。

port
administrative REST API に使用されている HTTPS ポート番号を指定します。

デフォルト値は 9443 です。

ホスト名またはポート番号がデフォルトから変更されている場合は、REST API の URL から正しい値を判別できます。 URL を表示するには、 **dspmqweb status** コマンドを使用します。

関連情報
dspmqweb 状況 (mqweb サーバー状況の表示)

REST API 各国語サポート

REST API では、HTTP 要求の一部として各国語を指定できます (ただし、いくらかの制限があります)。

背景

HTTP ヘッダーにより、要求では特定の動作を指定でき、応答では追加情報を提供できます。

HTTP ヘッダーには、情報を特定の言語で返すように要求する機能が組み込まれています。 REST API は、可能であればそのヘッダーを尊重します。

各国語の指定

ACCEPT-LANGUAGE HTTP ヘッダーに 1 つ以上の言語タグを指定することができます。 オプションで各タグにランクを関連付けて、優先順位順に並んだリストを指定することも可能です。 このページにその原理が解説されています。

REST API はこのヘッダーを尊重し、ACCEPT-LANGUAGE ヘッダーから言語を選択し、その言語でメッセージを返します。 REST API でサポートできる言語が ACCEPT-LANGUAGE ヘッダーに指定されていない場合、メッセージはデフォルトの言語で返されます。 このデフォルトの言語は、REST API Web サーバーのデフォルト・ロケールに対応しています。

87 ページの『どのようなデータが翻訳されるか』のセクションで、どのようなデータが翻訳されるかについて説明します。

応答での該当言語の指定

REST API からの応答では、CONTENT-LANGUAGE HTTP ヘッダーにより、返されるメッセージの言語が示されます。

どのようなデータが翻訳されるか

エラー・メッセージと通知メッセージが翻訳されます。その他のテキストは翻訳されません。

- キュー・マネージャーから返されるデータは翻訳されません。例えば、REST API を介して MQSC コマンドを実行した場合は、キュー・マネージャーのロケールでキュー・マネージャーの応答が返されます。
- REST API に関する生成された (Swagger) 資料 (`apiDiscovery` 機能で公開される資料) は英語です。

サポートされる言語

英語に加えて、REST API のエラー・メッセージと通知メッセージは以下の言語に翻訳されます。

中国語 (簡体字)
言語タグ `zh_CN`

中国語 (繁体字)
言語タグ zh_TW

チェコ語
言語タグ cs

フランス語
言語タグ fr

ハンガリー語
言語タグ hu

イタリア語
言語タグ it

日本語
言語タグ ja

韓国語
言語タグ ko

ポーランド語
言語タグ pl

ポルトガル語 (ブラジル)
言語タグ pt_BR

ロシア語
言語タグ ru

スペイン語
言語タグ es

例

以下の例では、Web サーバーのデフォルト・ロケールは英語です。

サポートされる言語を 1 つ指定した場合
要求ヘッダーで ACCEPT-LANGUAGE を fr に設定します。 この設定により、翻訳可能テキストの優先言語がフランス語であることを指定します。

応答ヘッダーで CONTENT-LANGUAGE が fr に設定されます。 この設定は、応答のエラー・メッセージと通知メッセージがフランス語であることを示します。

複数の言語を含むリストを指定した場合
要求ヘッダーで ACCEPT-LANGUAGE を am, fr に設定します。 この設定により、翻訳可能テキストに対して受け入れ可能な言語がアムハラ語とフランス語であり、優先言語がアムハラ語であることを指定します。

応答ヘッダーで CONTENT-LANGUAGE が fr に設定されます。 この設定は、応答のエラー・メッセージと通知メッセージがフランス語であることを示します。REST API はアムハラ語をサポートしないためです。

サポートされない言語を 1 つ指定した場合
要求ヘッダーで ACCEPT-LANGUAGE を am に設定します。 この設定により、翻訳可能テキストの優先言語がアムハラ語であることを指定します。

応答ヘッダーで CONTENT-LANGUAGE が en に設定されます。 この設定は、応答のエラー・メッセージと通知メッセージが英語であることを示します。REST API はアムハラ語をサポートしないためです。

REST API のバージョン

REST API のバージョン番号は、REST 要求のベース URL の一部となります。 例えば、https://localhost:9443/ibmmq/rest/**v2**/admin/installation などです。 バージョン番号は、将来のリリースで REST API の変更が生じてもクライアントに影響が及ぶことを防ぐために使用されます。

IBM MQ 9.2.0 では、バージョン 2 の REST API が導入されました。 このバージョンの増加は、administrative REST API、 messaging REST API、および MFT REST API に適用されます。 このバージョンの引き上げにより、REST API で使用するリソース URL が変更されました。 バージョン 2 のリソース URL の URL 接頭部は、以下の URL です。

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

Stabilized REST API に導入される変更によっては、既存の REST API 機能が変更され、REST API を使用するクライアントを更新しなければならない可能性があります。 そのような変更のためにクライアントを更新せざるを得なくなるのを防ぐため、REST API のバージョン番号が大きくなり、既存の機能は前の番号で固定化されます。 既存の機能が変更される可能性のある新機能は、新しいバージョン番号で REST API に追加されます。 したがって、クライアントは、更新せずに前のバージョンの REST API を使い続けることができます。

クライアントの更新が必要になる可能性のある REST API の変更としては、以下の変更があります。

- REST API に送信される、またはそこから返される JSON 内の既存の属性のサポートの削除。
- URL、HTTP 動詞、またはヘッダーの削除。 例えば、URL またはヘッダーが名前変更される場合、あるいは別の動詞が使用される場合などです。
- 既存の URL に送信されるデータへの新しい必須 JSON 属性の追加。
- 既存の URL に送信されるデータへの新しい必須 HTTP ヘッダーの追加。
- 既存の URL への新しい必須照会パラメーターの追加。

このタイプの変更が、 Long Term Support (LTS) リリースに存在していた REST API 機能に導入されると、これらの変更の最初の REST API のバージョン番号が増加します。 REST API を使用するクライアントの変更が必要になる可能性がある、 Continuous Delivery (CD) リリース内で行われる後続の変更では、新しいバージョン番号が使用されます。

このバージョン番号は、以降の各 CD リリースが出される間、次の LTS リリースまで同じままです。 したがって、バージョン番号は LTS リリースが次のリリースになった場合に、最大でも 1 回大きくなるだけです。

Stabilized バージョン番号が大きくなると、既存の REST API 機能は前のバージョン番号で固定化されます。 つまり、 LTS リリースで使用可能であった既存の REST API 機能は、古いバージョン番号で引き続き使用できますが、そのバージョンに対してこれ以上の変更は行われません。 REST API に追加される新しい機能は、新しいバージョンの REST API に追加されます。 ただし、バージョンの増加より前に CD リリースで REST API に対して行われた追加は、古いバージョンの REST API に含まれることが保証されません。

Deprecated 既存のクライアントは、変更の必要なしに前のバージョン番号で REST API を使い続けることができます。 以前のバージョンの REST API は非推奨になって最終的には除去される可能性があります。

変更の中には、REST API を使用するクライアントの変更を必要としないものもあります。 このような変更では、バージョン番号は大きくなりません。 したがって、これらのタイプの変更が導入されたときに REST API を使用するクライアントを更新しなくても良いようにしておいてください。 REST API に対するこれらの変更としては、以下の変更が考えられます。

- REST API から返される既存データへの新しい JSON 属性の追加。
- 新しい URL の追加。
- 既存の URL への新しい HTTP 動詞の追加。
- 既存の URL への新しい状況コードの追加。
- 既存の URL に送信されるデータへの新しいオプション JSON 属性の追加。
- 既存の URL の新しい照会パラメーターの追加。
- 既存の URL に送信されるデータへの新しいヘッダーの追加。
- REST API からの新しいヘッダーの戻り。

新しい Continuous Delivery REST API 機能の変更

CD リリースで追加された新しい REST API 機能の場合、この新しい機能に加えられた変更のうち、 REST API クライアントに対する変更が必要になる可能性があるものは、バージョン番号を大きくしません。 つまり、新機能は、次の LTS リリースの前に、バージョン番号が大きくならずに変更される可能性があります。 機能が LTS リリースに組み込まれる場合、REST API クライアントの変更を必要とする可能性のあるその後の変更で、バージョン番号が大きくなります。

例

1. LTS リリース X では、REST API はバージョン 1 です。
2. CD リリース X.0.1 で、新しい URL のサポートが追加されます。 この変更は、REST API を使用するクライアントの変更を必要としません。 したがって、REST API はバージョン 1 のままです。
3. CD X.0.2 で、新しい URL のサポートが追加されます。 この変更は、REST API を使用するクライアントの変更を必要としません。 したがって、REST API はバージョン 1 のままです。
4. LTS リリース Y では、REST API はバージョン 1 です。
5. CD リリース Y.0.1 で、既存の URL が名前変更されます。 この変更は、REST API を使用するクライアントの変更を必要とする可能性があります。 そのため、新しいバージョンの REST API がバージョン 2 として作成されます。 名前変更された URL は、REST API のバージョン 2 に、既存のすべての機能とともに組み込まれます。 REST API に追加されたすべての新機能が、バージョン 2 に追加されます。 バージョン 1 は、 LTS リリース Y のレベルで安定化されたままです。
6. CD リリース Y.0.2 で、別の既存の URL が名前変更されます。 バージョンがすでに CD リリース Y で増加されているため、REST API はバージョン 2 のままです。 バージョン 1 は、 LTS リリース Y のレベルで安定化されたままです。
7. LTS リリース Z では、 REST API はバージョン 2 のままです。 バージョン 1 は、 LTS リリース Y のレベルで安定化されたままです。

IBM MQ Console を使用した管理

IBM MQ Console を使用して、基本的な管理タスクを実行できます。

注 : IBM MQ Console を使用する際は、どのキュー・マネージャーでもコマンド・サーバーを無効にしないでください。 コマンド・サーバーがキュー・マネージャーに対して使用不可になっている場合は、以下のようにします。

- IBM MQ Console が応答しなくなり、コマンドの処理に長い遅延が生じる
- キュー・マネージャーに対して発行されたすべてのコマンドがタイムアウトになります。

関連タスク

IBM MQ Console のトレース

IBM MQ Console の使用開始

mqweb サーバーを構成し、 IBM MQ Console の URI を決定し、コンソールに接続し、コンソールにログインします。

始める前に

このタスクを実行するには、**dspmqweb** コマンドを使用するための特定の特権を持っているユーザーである必要があります。

- z/OS z/OS の場合、**dspmqweb** コマンドを実行する権限と、`mqwebuser.xml` ファイルに対する書き込みアクセス権限を持っている必要があります。
- Multi 他のすべてのオペレーティング・システムでは、特権ユーザーでなければなりません。

IBM i IBM i では、コマンドを QSHELL で実行する必要があります。

このタスクについて

以下の制約事項に注意してください。

- z/OS
  - z/OS では、キュー・マネージャーを作成、削除、開始、停止することはできません。
  - z/OS では、チャネル・イニシエーターを開始および停止することができず、チャネル・イニシエーターの状況は表示されません。
  - リスナーを表示または管理することができません。
  - チャネルの開始、ping、解決、およびリセットの各コマンドは、CHLDISP(DEFAULT) でのみ発行できます。
  - QSGDISP(GROUP) で定義したオブジェクトを表示および管理することができません。
  - キュー・マネージャー・セキュリティーを管理することができません。
  - システム・リソース使用量をモニターすることができません。
- Multi
  - IBM MQ Console を使用して AMQP チャネルを操作することはできません。
  - IBM MQ Console を使用して MQTT チャネルを操作することはできません。

手順

1. IBM MQ Console で使用するように mqweb サーバーがまだ構成されていない場合は、mqweb サーバーを構成してください。

   基本レジストリーを使用した mqweb サーバーの構成について詳しくは、 mqweb サーバーの基本構成を参照してください。
2. z/OS

   z/OS では、 **dspmqweb** コマンドを使用できるように WLP_USER_DIR 環境変数を設定します。 次のコマンドを入力して、mqweb サーバー構成を指すように変数を設定します。

   ```
   export WLP_USER_DIR=WLP_user_directory
   ```

   ここで、 *WLP_user_directory* は、 crtmqweb に渡されるディレクトリーの名前です。 例 :

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

   詳しくは、mqweb サーバーの作成を参照してください。
3. 以下のコマンドを入力して、IBM MQ Console の URI を確認します。

   ```
   dspmqweb status
   ```

   このコマンドを実行すると、以下のような出力が生成されます。

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

   IBM MQ Console の URI の末尾は、console/ という接尾部になります。
4. ブラウザーで前のステップの URL を入力して、 IBM MQ Console に接続します。

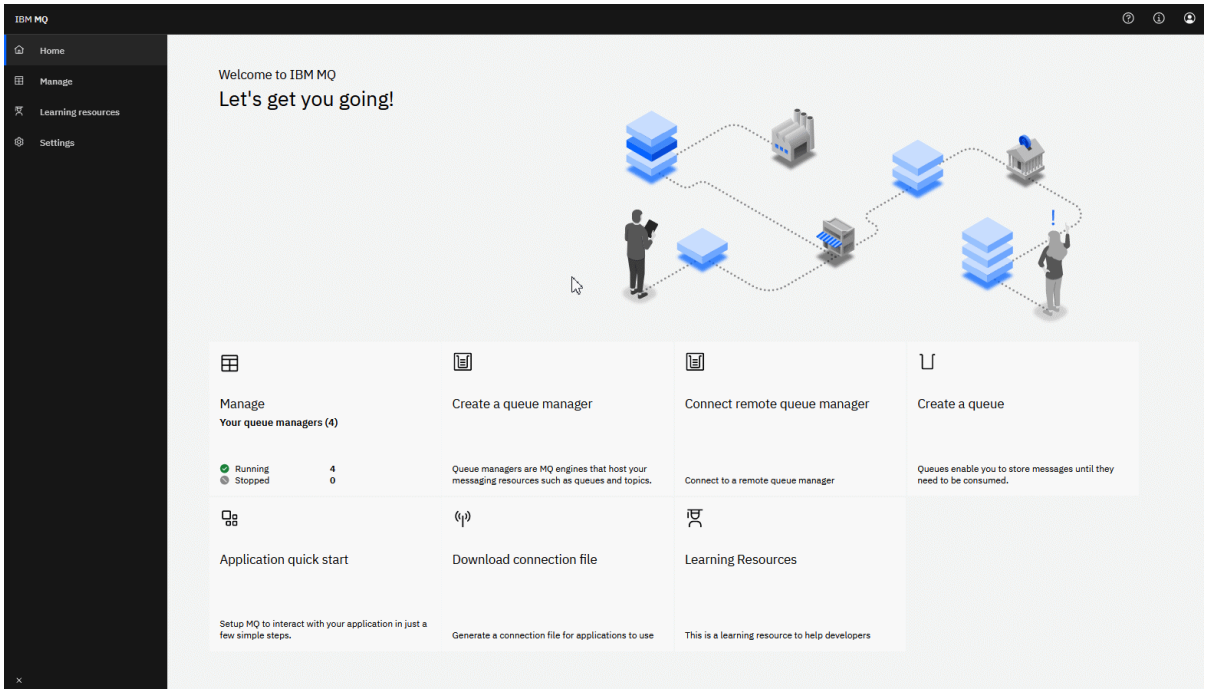
   mqweb サーバーで提供されるデフォルトの証明書はトラステッド証明書ではないため、ブラウザーによってセキュリティー例外が生成される可能性があります。 IBM MQ Console に進むことを選択してください。
5. IBM MQ Console にログインします。 ユーザー名 mqadmin、およびパスワード mqadmin を使用します。

次のタスク

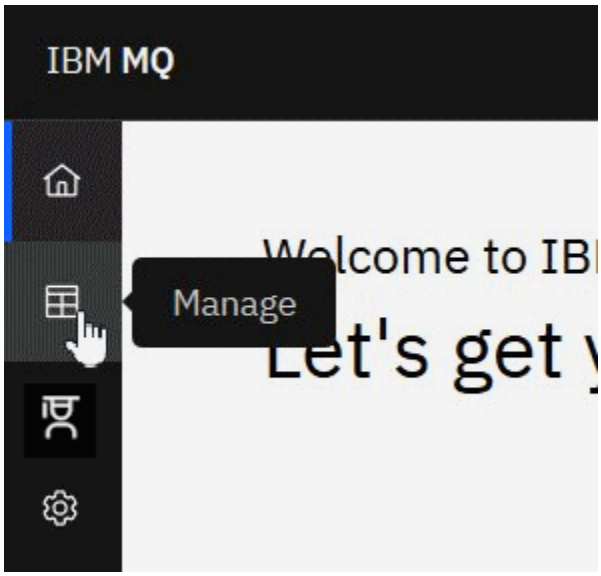
デフォルトでは、IBM MQ Console はトークン・ベースの認証を使用してユーザーを認証します。 クライアント証明書認証を使用することもできます。 詳しくは、REST API および IBM MQ Console でのクライアント証明書認証の使用 を参照してください。

V 9.4.0 IBM MQ Console のクイック・ツアー

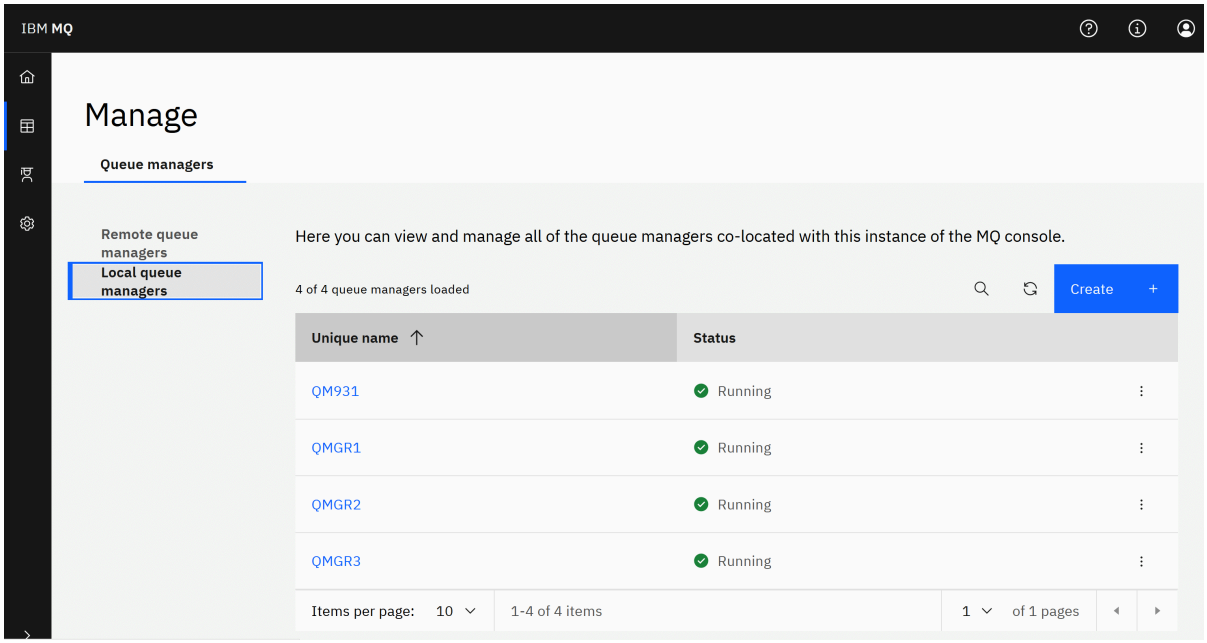
IBM MQ Console に初めてログインすると、ランディング・ページが表示されます。 ここから、既存のキュー・マネージャーを管理するか、キュー・マネージャーまたはキューを作成するか、いくつかの教育トピックにナビゲートするか、IBM Documentation で IBM MQ 製品情報を開くかを選択できます。 また、アプリケーションのクイック・スタートを起動することもできます。これにより、新規または既存のキュー・マネージャーとアプリケーションの間のメッセージングを素早く簡単にセットアップするプロセスがガイドされます。



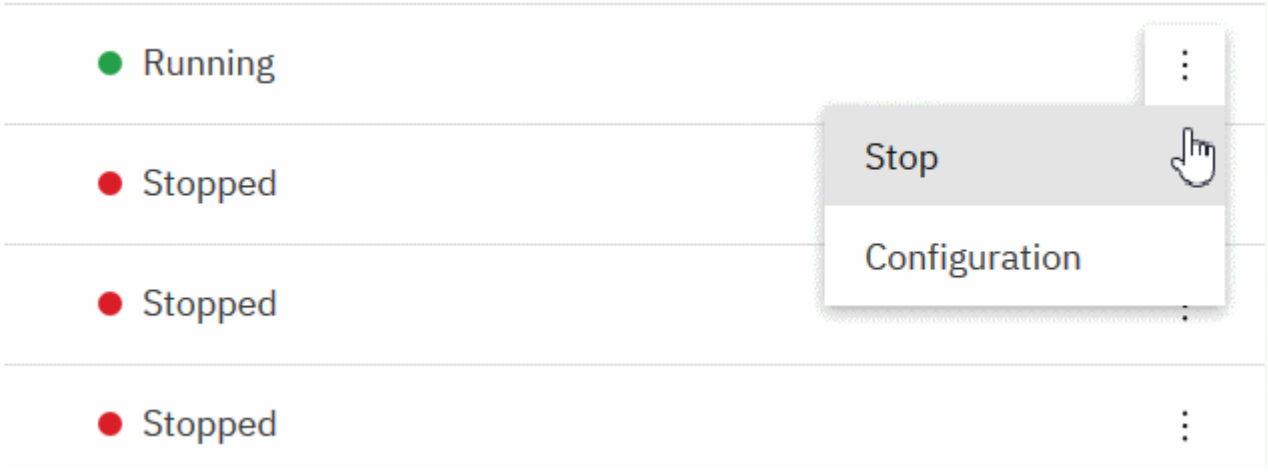
別の方法として、管理アイコンをクリックして、すぐに IBM MQ オブジェクトの管理を始めることもできます。



管理ビューに最初に表示されるのは、キュー・マネージャーとその現在の状態です。 また、新しいキュー・マネージャーを作成したり、リモート・キュー・マネージャーに接続したりすることもできます。

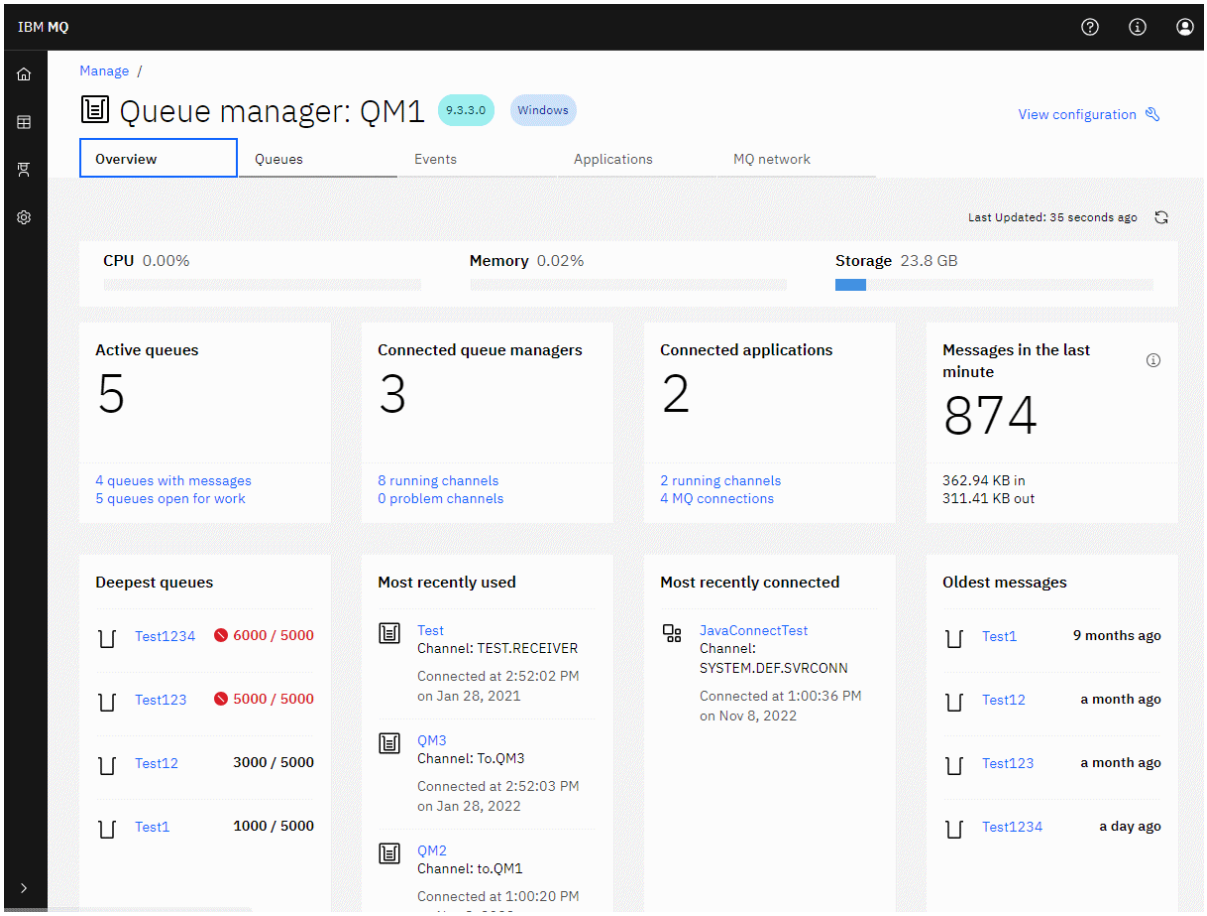


キュー・マネージャーごとにメニューがあり、実行中のキュー・マネージャーの停止または構成、停止したキュー・マネージャーの開始または削除を行えます。



キュー・マネージャーの権限レコード、認証情報オブジェクト、チャネル認証レコードは、キュー・マネージャーの「**構成**」ページの「**セキュリティー**」タブにあります。ここで新規のものを作成して追加できます。

実行中のキュー・マネージャーの名前をクリックして、そのダッシュボードを開きます。



キュー・マネージャーのダッシュボードからは、次のアクションを実行できます。

V 9.4.0 「**概要**」 タブで、以下の情報を表示します。

CPU
キュー・マネージャーによる CPU 使用率の推定値 (%)。 (z/OS には適用されません。)

メモリー
キュー・マネージャーによるメモリー使用量の推定パーセンテージ。 (z/OS または Windows には適用されません。)

記憶域
キュー・マネージャーが存在するディスクのフリー・スペースの推定パーセンテージ。 (z/OS には適用されません。)

アクティブなキュー
メッセージがあるキュー、または入力または出力用にオープンされているキューの数。

接続されたキュー・マネージャー
アクティブなチャネルから派生した、現在接続されているキュー・マネージャーの数。

接続されているアプリケーション
現在接続されているアプリケーションの数。

直近 1 分間のメッセージ数
10 秒ごとのメッセージ・スループットを示す PUT/GET システム・トピックの要約を表示します。 (z/OS には適用されません。)

サブスクリプション
サブスクリプションの数を表示します。 z/OS およびシステム・トピックのモニターが禁止されている他のプラットフォームでのみ表示されます (setmqweb プロパティーを参照)。

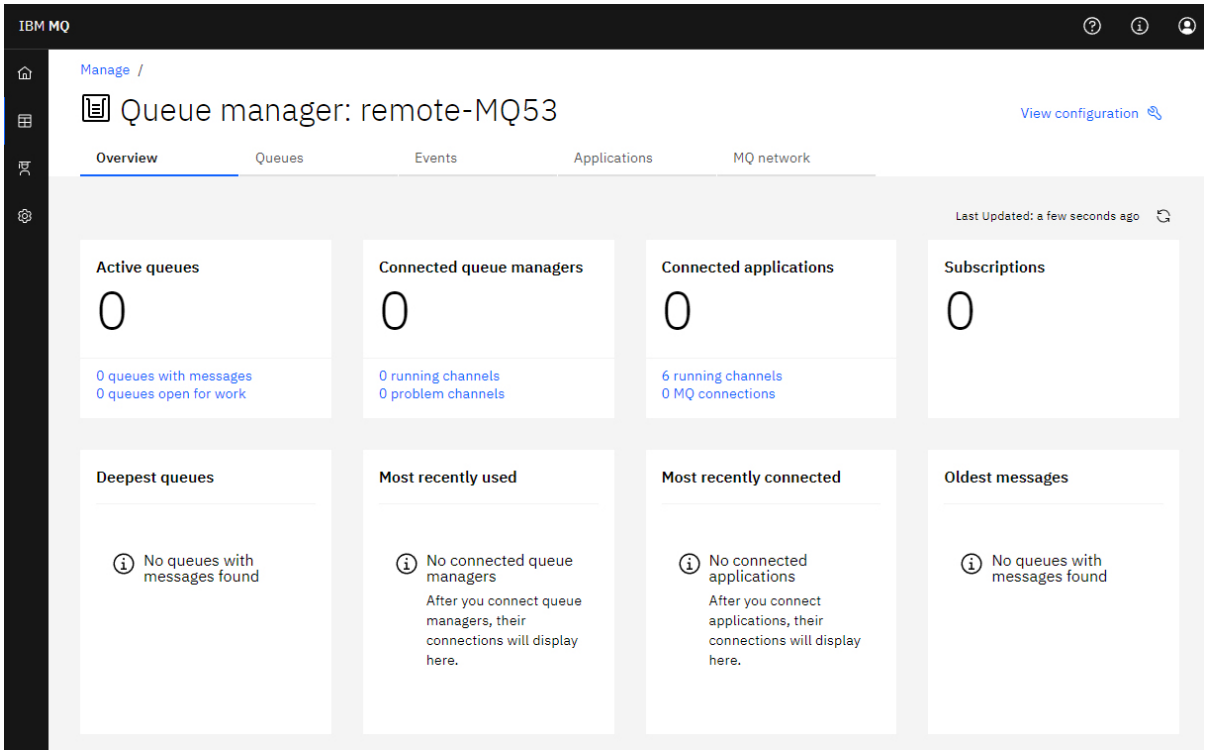
最大深度キュー数
キューを項目数順にリストします。 現在のキュー項目数と最大キュー項目数を示します。

最近使用したもの
現在接続されているキュー・マネージャーを、最終メッセージ日付順にリストします。

最近接続されたもの
アクティブなサーバー接続チャネルから派生した現在接続されているアプリケーションを、チャネル開始日時の順にリストします。

最も古いメッセージ
最も古いメッセージ日時の順にキューをリストします。

「**概要**」 タブに表示される情報は、システム・トピックのモニターから得られます (システム・トピックで公開されているメトリックを参照してください)。 z/OS は、システム・トピックのモニターをサポートしていません。他のプラットフォームでは、コンソール表示のためのモニターを無効にすることができます (setmqweb プロパティーを参照)。 このような場合、 「**概要**」 タブにはさらに限定された情報が表示され、その外観は以下の例のようになります。



「**キュー**」タブ:

- 新規キューを作成する
- 既存のメッセージを表示して新しいメッセージを作成し、キューを構成するには、キュー名をクリックします。

「**イベント**」 タブで、以下のようにします。

トピック

- 新規トピックを作成する
- 既存トピックを構成する
- トピック名をクリックして、一致するサブスクリプションを表示する

サブスクリプション

- 新規の管理対象サブスクリプションまたは新規の非管理対象サブスクリプションを作成する
- 既存のサブスクリプションを構成する

V 9.4.0 「**アプリケーション**」 タブで、以下のようにします。

概要
以下の統計の概要を示すタイルが含まれます。

接続されているアプリケーション
接続されているアプリケーションの数を表示します。 以下のタブへのリンクを提供します。

- **アプリケーション・インスタンス**
- **接続**

チャネル・インスタンスの実行
SVRCONN チャネル・インスタンスの数と、そのリンクから、「**アプリケーション・チャネル**」 タブで定義または停止されたチャネル・インスタンスの数を表示します。

接続
接続数のカウントを表示します。 「**接続**」 タブには、以下の情報へのリンクがあります。

- ローカル接続 (チャネル名のない接続)
- リモート接続 (チャネル名を持つ接続)

最も一般的なアプリケーション
使用している接続数の順に、頻度の高いアプリケーションのリストを表示します。

最も一般的なチャネル
頻度の高いチャネルのリストを、アクティブなインスタンスの数の順に表示します。

最も古いトランザクション
アプリケーション名ごとに最も古いトランザクションのリストを表示します。 これらのトランザクションには、オープン作業単位との接続があり、UOW の開始日時順に配列されます。

リモート接続バージョン
接続されている共通 IBM MQ バージョン (つまり、指定された REMOTE_VERSION を持つチャネル・インスタンス) のリストを表示します。

アプリケーション・チャネル・セキュリティー
共通接続チャネル・セキュリティー・プロトコル (つまり、指定された SECURITY_PROTOCOL を持つチャネル・インスタンス) のリストを表示します。

チャネル転送速度
メッセージおよびバイトの転送速度順に並べられた共通チャネルのリストを表示します。 チャネル開始日時を使用して期間を計算し、MSGS および MQIACH_BYTES_SENT/MQIACH_BYTES_RCVD を使用して率を計算します。

アプリケーション
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に関する情報を表示します。

チャネル
アプリケーションに接続されたチャネル上のアクティビティーを表示します。

アプリケーション・チャネル

- チャネルを開始、停止、Ping、および構成する ⋮
- 新規チャネルを作成する
- チャネルのリセット

アプリ・チャネル・インスタンス

- アプリケーション・チャネル・インスタンスの状況の表示
- チャネル上の未確定メッセージの解決 ⋮

V 9.4.0 「**MQ ネットワーク**」 タブで、以下のようにします。

概要
以下の統計の概要を示すタイルが含まれます。

キュー・マネージャー・チャネル・インスタンスの実行
非 SVRCONN チャネル・インスタンスの数を表示します。「**接続されたキュー・マネージャー**」タブに、以下のタイプのチャネル・インスタンスへのリンクを表示します。

- 定義済みチャネル
- 停止中のチャネル

接続されたキュー・マネージャー
MQCA_REMOTE_Q_MGR_NAME によって接続されたキュー・マネージャーの数を表示します。 また、MQCMD_INQUIRE_CLUSTER_Q_MGR によって戻されるキュー・マネージャーの数も提供します。

クラスター・メンバーシップ
キュー・マネージャー・クラスターが 1 つしかない場合は、クラスターの名前と、キュー・マネージャーがフル・リポジトリーか部分リポジトリーかを表示します。 クラスター内で表示されるキュー・マネージャーの数を表示します。 複数のクラスターがある場合は、クラスターの数に加えて、各クラスター内のフル・リポジトリー・キュー・マネージャーと部分リポジトリー・キュー・マネージャーの数が表示されます。

失敗したキュー・マネージャー・チャネル
再試行状態 (非停止/実行中) のチャネルのリストを表示します。 再試行状態にある場合、残りの再試行回数を計算します。 このリストには、以下の状況タイプのチャネルが含まれています。

- MQCHS_PAUSED
- MQCHS_RETRYING

最長のメッセージ遅延
XMIT 時間標識 (長期間) を持つチャネルのリストを表示します。

不在伝送キュー
キュー項目数がゼロ以外で、関連するハンドルがない伝送キューのリストを表示します。

リモート接続のバージョン
接続されている共通 IBM MQ バージョン (つまり、指定された REMOTE_VERSION を持つチャネル・インスタンス) のリストを表示します。

キュー・マネージャーのチャネル・セキュリティー
共通接続チャネル・セキュリティー・プロトコル (つまり、指定された SECURITY_PROTOCOL を持つチャネル・インスタンス) のリストを表示します。

クラスターの正常性

クラスターの正常性に関連するいくつかの独立した統計を表示します。 状況には以下が含まれます。

- クラスター・オブジェクト (キュー、トピック、キュー・マネージャー) の数。
- 中断状態のキュー・マネージャーの数 (MQIACF_SUSPEND を YES に設定)。
- SYSTEM.CLUSTER.COMMAND.QUEUE キュー。
- SYSTEM.TEMP。

これらの値がすべてゼロの場合、このタイルは表示されず、代わりに「**リスナー**」タイルが表示されます。

リスナー
リスナーのリスト、およびリスナーが実行状態であるかどうかを表示します。「**クラスターの正常性**」タイルが表示されていない場合にのみ表示されます。

接続されたキュー・マネージャー
このキュー・マネージャーに現在接続されているキュー・マネージャーの詳細を表示します。

キュー・マネージャー・チャネル

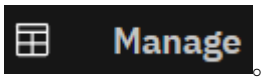
- チャネルを開始、停止、Ping、および構成する
- 新規チャネルを作成する

- チャネルのリセット

キュー・マネージャー・チャネル・インスタンス

- キュー・マネージャー・チャネル・インスタンスの状況の表示
- チャネル上の未確定メッセージの解決 ⋮

V 9.4.0 IBM MQ Console: ローカル・キュー・マネージャーの操作

ローカル・キュー・マネージャーの作成、構成、および制御は、「管理 (Manage)」ビューの最上位から行います 

。

このタスクについて

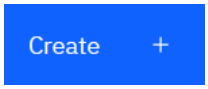
Multi 「管理」ビューには、以下に追加されたローカル・キュー・マネージャーがリストされます。 IBM MQ Console が実行されている IBM MQ インストール済み環境。 同じシステム上にあるが、異なる IBM MQ インストール済み環境に関連付けられているキュー・マネージャーは、リストされません。

z/OS z/OS の「管理」ビューには、 IBM MQ Console と同じバージョンのキュー・マネージャーがリストされ、 IBM MQ Console が実行されているシステム上で定義されます。 IBM MQ Console とは異なるバージョンのキュー・マネージャーはリストされません。

操作する個々のキュー・マネージャーをリストから選択できます。

注 : IBM MQ Console は、アクティブな場合 (つまり、1 次役割を持っている場合)、ローカル RDQM キュー・マネージャーに接続できますが、RDQM 固有の機能は提供しません。

手順

- 新規ローカル・キュー・マネージャーを作成するには、次のようにします。
  a) キュー・マネージャー・リスト・ビューで「作成」ボタン 

をクリックします。
  b) 新しいキュー・マネージャーの名前を入力します。 名前の長さは 48 文字までです。 有効な文字は、アルファベットと数字、":"、"/"、"_"、"%" です。
  c) オプション: キュー・マネージャーが listen する使用可能な TCP/IP ポートを入力します。 ポート番号は 65535 以下でなければなりません。
  d)「**作成**」 をクリックします。 新しいキュー・マネージャーが作成され、開始します。
- ローカル・キュー・マネージャーを開始するには、次のようにします。
  a) 開始するキュー・マネージャーをリスト内で見つけます。
  b) メニュー ⋮ から「**開始**」を選択します。
- ローカル・キュー・マネージャーを停止するには、次のようにします。
  a) ローカル・キュー・マネージャー・ウィジェットのリストから、停止するキュー・マネージャーを選択します。
  b) メニュー ⋮ から「**停止**」を選択します。
- ローカル・キュー・マネージャーを削除するには、次のようにします。
  a) キュー・マネージャーが稼働している場合はそれを停止します。
  b) メニュー ⋮ から 「**構成の表示**」 を選択し、 「**キュー・マネージャーの削除**」を選択します。

c) 確認ウィンドウにキュー・マネージャーの名前を入力して、キュー・マネージャーを削除することを確認します。 キュー・マネージャーおよび関連するすべてのオブジェクトが削除されます。

- ローカル・キュー・マネージャーのプロパティーを表示および編集するには、次のようにします。
  a) キュー・マネージャーが実行中であることを確認し、キュー・マネージャー・リストの中でそのキュー・マネージャーを見つけます。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。
  c)「**プロパティー**」タブが選択されていることを確認します。 プロパティーを表示し、必要に応じて編集します。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。 プロパティーに関する情報については、 キュー・マネージャー属性でプロパティー情報を表示することができます。
- ローカル・キュー・マネージャーのセキュリティー設定を操作するには、次のようにします。
  a) キュー・マネージャーが実行中であることを確認し、キュー・マネージャー・リストの中でそれを選択します。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。
  c)「**セキュリティー**」タブが選択されていることを確認します。
  d) 認証オブジェクト、許可レコード、またはチャネル認証オブジェクトを操作できます。 詳細については、次のトピックを参照してください。
  - 99 ページの『IBM MQ Console: 認証情報オブジェクトの処理』
  - 101 ページの『IBM MQ Console: キュー・マネージャー権限レコードの処理』
  - 102 ページの『IBM MQ Console: チャネル認証レコードの処理』

V 9.4.0 *IBM MQ Console:* 認証情報オブジェクトの処理

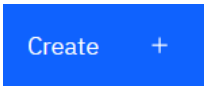
コンソールを使用して、キュー・マネージャーの認証情報オブジェクトを追加および削除することができます。 プロパティーを表示および設定したり、オブジェクトの権限レコードを管理したりすることもできます。

このタスクについて

認証情報ビューに、特定のキュー・マネージャーのための認証情報がリストされます。 操作する個々の認証情報をリストから選択できます。

キュー・マネージャー認証情報は、IBM MQ による Transport Layer Security (TLS) のサポートの一部を成すものです。 これらのオブジェクトには、OCSP、または LDAP サーバーの証明書失効リスト (CRL) のいずれかを使用して証明書失効検査を実行するために必要な定義に加えて、ユーザー ID 検査とパスワード検査を使用可能にするために必要な定義が入っています。

手順

- キュー・マネージャーの認証情報を表示するには、次のようにします。
  a) キュー・マネージャーが実行中であることを確認し、キュー・マネージャー・リストの中でそれを選択します。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。
  c)「**セキュリティー**」タブが選択されていることを確認します。
  d) ナビゲーション・パネルから「**認証情報**」を選択します。
- 認証情報オブジェクトを追加するには、次のようにします。
  a) 認証情報リスト・ビューで「作成」ボタン 

をクリックします。

b) 認証情報オブジェクトの名前を指定します。 有効な文字は、 アルファベットと数字、":"、"/"、"_"、"%" です。

c) 認証情報オブジェクトのタイプを指定します。

d) オブジェクト・タイプに適切な追加情報を指定します。

- **CRL LDAP** では、「**LDAP サーバー名**」を指定します。 この名前は、LDAP サーバーが稼働しているホストのホスト名、IPv4 のドット 10 進数アドレス、または IPv6 の 16 進数表記です。オプションでポート番号を付けます。 オプションで、LDAP サーバーにアクセスするユーザーのユーザー名とパスワードを指定することができます。
- **OCSP** では、「**OCSP 応答側 URL**」を指定します。 この URL は、証明書失効の検査に使用するレスポンダーの URL です。 この値は、OCSP 応答側のホスト名とポート番号を含む HTTP URL でなければなりません。 OCSP 応答側が HTTP のデフォルトであるポート 80 を使用する場合には、ポート番号を省略できます。 HTTP URL は RFC 1738 で定義されています。
- **IDPW OS** には追加要件はありませんが、任意で、この認証タイプにさらにオプションを指定することができます。
- **IDPW LDAP** では、「**LDAP サーバー名**」と「**短いユーザー**」の名前を指定します。 LDAP サーバー名は、LDAP サーバーが稼働しているホストのホスト名、IPv4 のドット 10 進数アドレス、または IPv6 の 16 進数表記です。オプションでポート番号を付けます。 短いユーザー名は、接続のショート・ネームとして使用する LDAP ユーザー・レコード内のフィールドです。 任意で、この認証タイプにさらにオプションを指定することができます。

e) **追加** をクリックします。

- 認証情報オブジェクトを削除するには、次のようにします。

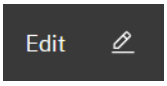
a) 削除する認証情報オブジェクトのスパナのアイコン  をリストから選択します。

b) オブジェクト・プロパティー・ビューで「**認証情報オブジェクトの削除**」をクリックします。

c)「**削除**」をクリックして、認証情報オブジェクトの削除を確定します。 オブジェクトが削除されます。

- 認証情報オブジェクトのプロパティーを表示および編集するには、次のようにします。

a) 表示する認証情報オブジェクトのスパナのアイコン  をリストから選択します。

b) 表示されたプロパティーを編集するには、「編集」ボタン 

クリック

c) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。

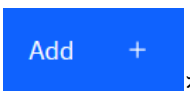
d)「**保存**」をクリックして、変更を保存します。

- 認証情報オブジェクトの権限レコードを表示および編集するには、次のようにします。

a) 権限レコードの表示対象となる認証情報オブジェクトのスパナのアイコン  をリストから選択します。

b)「**セキュリティー**」タブをクリックします。

c) 既存の権限レコードを編集または削除するには、メニュー から「**編集**」または「**削除**」を選択します。

d) 新規権限レコードを追加するには **追加** ボタン 

ボタンをクリックし、新規権限レコードの詳細を指定して、「**クリエイト作成**」をクリックします。

V 9.4.0 *IBM MQ Console:* キュー・マネージャー権限レコードの処理

ユーザーおよびグループに対して権限レコードを指定すると、そのユーザーおよびグループがキュー・マネージャーに対して持つアクセス権限を制御できます。

このタスクについて

権限レコードを使用すると、メッセージング・ユーザーまたはメッセージング・ユーザー・グループが特定のキュー・マネージャーに対して持つアクセス権限を微調整できます。 権限レコードには 2 つのタイプがあります。一般権限を制御する **キュー・マネージャー・アクセス** レコードと、キュー・マネージャーのオブジェクトを作成できるユーザーおよびグループを制御する **作成許可** レコードです。

手順

- キュー・マネージャーの権限レコードを表示するには、次のようにします。
  a) キュー・マネージャーが実行中であることを確認し、キュー・マネージャー・リストの中でそれを選択します。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。
  c)「**セキュリティー**」タブが選択されていることを確認します。
  d) ナビゲーション・パネルから「**権限レコード**」を選択します。 ビューの 2 つのペインに権限レコードが表示されます。これらのペインで、一般権限レコードと作成権限レコードを操作できます。
- 一般権限レコードを追加するには、次のようにします。
  a)「**キュー・マネージャー・アクセス**」 リスト・ビューの追加ボタン 

をクリックします。
  b) ユーザーとグループのどちらの権限レコードを追加するかを選択します。
  c) 権限レコードを追加するユーザーまたはグループの名前を指定します (権限レコードはこれをその名前とします)。
  d) 付与する権限を選択します。
  e)「**作成**」 をクリックします。
- 作成権限レコードを追加するには、次のようにします。
  a)「**作成許可**」 リスト・ビューの「追加」ボタン 

をクリックします。
  b) ユーザーとグループのどちらの権限レコードを追加するかを選択します。
  c) 権限レコードを追加するユーザーまたはグループの名前を指定します (権限レコードはこれをその名前とします)。
  d) 作成権限を付与するオブジェクトのタイプを選択します。
  e)「**作成**」 をクリックします。
- 権限レコードを削除するには、次のようにします。
  a) 削除する権限レコードを選択し、 「**削除**」を選択します。
  b)「**削除**」をクリックして、認証情報オブジェクトの削除を確定します。 オブジェクトが削除されます。
- 権限レコードのプロパティーを表示および編集するには、次のようにします。
  a) 表示する権限レコードをクリックします。
  b) 必要に応じて設定を変更し、「**保存**」をクリックして変更を保存します。

V 9.4.0 *IBM MQ Console:* チャネル認証レコードの処理

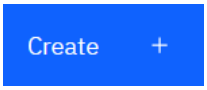
IBM MQ Console を使用して、キュー・マネージャー上のチャネル認証レコードを追加および削除することができます。 チャネル認証レコードのプロパティーを表示および設定することもできます。

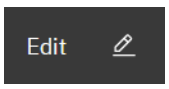
このタスクについて

チャネル認証レコードを使用すれば、接続システムに与えるアクセス権限をチャネル・レベルでさらに細かく制御できるようになります。

セキュリティーを強化するために、ブロック・チャネル認証レコードを使用して、チャネルへのアクセスをブロックできます。 また、アドレス・マップ・チャネル認証レコードを使用して、指定したユーザーにアクセスを許可することもできます。 チャネル認証レコードについて詳しくは、チャネル認証レコードを参照してください。

手順

- キュー・マネージャーのチャネル認証情報を表示するには、以下のようにします。
  a) キュー・マネージャーが実行中であることを確認し、キュー・マネージャー・リストの中でそれを選択します。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。
  c)「**セキュリティー**」タブが選択されていることを確認します。
  d) ナビゲーション・パネルから「**チャネル認証**」を選択します。
- チャネル認証レコードを追加するには、次のようにします。
  a) チャネル認証情報リスト・ビューで「作成」ボタン 

をクリックします。
  b) 使用する規則タイプを選択します。 「**許可**」、「**ブロック**」、「**警告**」のいずれかを選択します。
  c) チャネル認証規則を構成する対象となる ID のタイプを選択します。 選択した規則タイプに応じて、使用可能な ID タイプが異なります。
  d) 指定する ID に必要な情報を入力します。 デフォルトでは、値を入力できるように表示されるのは、最小限の推奨プロパティーです。 「**カスタム作成**」を選択すると、使用可能なすべてのプロパティーを表示できます。
  e)「**作成**」 をクリックして、チャネル認証レコードを作成します。
  チャネル認証レコードに使用可能な設定について詳しくは、チャネル認証レコードと SET CHLAUTH を参照してください。
- チャネル認証レコードを削除するには、次のようにします。
  a) 削除するチャネル認証レコードの横にあるスパナのアイコン  をクリックします。
  b)「チャネル認証の編集」ビューで「**チャネル認証オブジェクトの削除**」をクリックします。
  c)「**削除**」をクリックして、チャネル認証レコードの削除を確定します。 チャネル認証レコードが削除されます。
- チャネル認証レコードのプロパティーを表示および編集するには、次のようにします。
  a) 編集または表示するチャネル認証レコードの横にあるスパナのアイコン  をクリックします。 プロパティーが表示されます。
  b)「編集」ボタンをクリック 

  c) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。

d)「**保存**」をクリックして、変更を保存します。

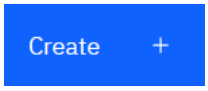
V 9.4.0 Multi *IBM MQ Console:* リスナーの操作

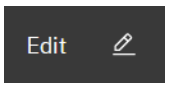
IBM MQ Console を使用して、リスナーの追加と削除、リスナーの開始と停止、リスナーのプロパティーの表示と設定、リスナーの権限レコードの管理を行うことができます。

このタスクについて

リスナー・ビューに、特定のキュー・マネージャーに存在するリスナーが表示されます。 操作するリスナーを個々に選択できます。

手順

- キュー・マネージャーのリスナーを表示するには、以下のようにします。
  a) キュー・マネージャーが実行中であることを確認し、キュー・マネージャー・リストの中でそれを選択します。
  b) メニュー ⋮ から「**構成の表示**」 を選択します。
  c)「**リスナー**」 タブを選択します。
- リスナーを作成するには、次のようにします。
  a)「作成」ボタンをクリック 

  b) 作成するリスナーについて、必要な情報を入力します。
  c)「**作成**」 をクリックします。 新しいリスナーが作成されます。
- リスナーを開始するには、次のようにします。
  a) 開始するリスナーをリスト内で見つけます。
  b) メニュー ⋮ から「**開始**」を選択します。
- リスナーを停止するには、次のようにします。
  a) 開始するリスナーをリスト内で見つけます。
  b) メニュー ⋮ から「**停止**」を選択します。
- リスナーのプロパティーを表示および編集するには、次のようにします。
  a) リスト内でリスナーを見つけます。
  b) メニュー ⋮ から「**構成の表示**」 を選択します。
  c)「**プロパティー**」タブが選択されていることを確認します。 プロパティーを編集するには、「編集」ボタンをクリック 

。
  d) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。 プロパティーについて詳しくは、MQ エクスプローラーの資料のリスナー・プロパティーを参照してください。
  e)「**保存**」をクリックして、変更を保存します。
- リスナーの権限レコードを表示および編集するには、次のようにします。
  a) リスト内でリスナーを見つけます。
  b) メニュー ⋮ から「**構成の表示**」 を選択します。

c)「**セキュリティ**」タブをクリックします。

d) キュー・マネージャーの権限レコードについての説明に従って、権限レコードを操作します。 101 ページの『IBM MQ Console: キュー・マネージャー権限レコードの処理』 を参照してください。

- リスナーを削除するには、次のようにします。

  a) リスト内でリスナーを見つけます。

  b) メニュー ⋮ から 「**構成の表示**」 を選択します。

  c)「**リスナーの削除**」をクリックします。

V 9.4.0 IBM MQ Console: リモート・キュー・マネージャーの追加

IBM MQ Console を使用して、リモート・システム上で実行されているキュー・マネージャーを管理することができます。

始める前に

- リモート・システム上のキュー・マネージャーをリモート側で管理できるように準備する必要があります。 105 ページの『コマンド行を使用した IBM MQ Console へのリモート・キュー・マネージャーの追加』 のステップ 106 ページの『1』、 106 ページの『2』、 106 ページの『3』、および 107 ページの『4』 を参照してください。
- IBM MQ Console からのリモート接続も有効にする必要があります。 詳しくは、 リモート・キュー・マネージャー接続の動作の構成を参照してください。

このタスクについて

リモート接続の詳細を指定するには、JSON 形式のクライアント接続定義テーブル (CCDT) を使用します。 JSON CCDT は、テキスト・エディター (105 ページの『コマンド行を使用した IBM MQ Console へのリモート・キュー・マネージャーの追加』 のステップ 107 ページの『5』 を参照) を使用して作成することも、IBM MQ Console を使用して作成することもできます。

あるいは、リモート・キュー・マネージャーを追加する際に接続詳細を直接指定することにより、IBM MQ Console から CCDT を作成することもできます。

(リモート・キュー・マネージャーの準備と CCDT の作成に加え) 必要なタスクすべてについて、コマンド・ラインを使用することで IBM MQ Console にリモート・キュー・マネージャーを接続することもできます。 105 ページの『コマンド行を使用した IBM MQ Console へのリモート・キュー・マネージャーの追加』 を参照してください。

重要 : 以下のメッセージが表示されたとします。

```
MQWB2026E: The request to connect to the remote queue manager 'rqmgr-qmgr_name' failed
with the error message:
'JMSCC0051: The property 'JMS_IBM_MQMD_AccountingToken' should be set using type '[B',
not 'java.lang.Object'.'
```

Java オブジェクト・タイプ `byte[]` が予期されているときに、`java.lang.Object` をアカウンティング・トークンに渡そうとしています。

手順

- 既存の CCDT を指定してリモート・キュー・マネージャーを追加するには、以下のようにします。

  a) ホーム・ページで「**リモート・キュー・マネージャーへの接続**」をクリックします。

  b) リモート・キュー・マネージャーの名前を指定します。

  c) オプションで、キュー・マネージャーの固有の名前を指定します。 固有の名前を指定しない場合は、実際に使用される名前には接頭部「remote-」が追加されます。

  d)「**JSON CCDT を使用して接続 (Connect using a JSON CCDT)**」が選択されていることを確認します。

  e)「**参照**」をクリックし、使用する JSON CCDT が入っているファイルを選択します。

f)「**次へ**」をクリックしてユーザー・ページに移動します。オプションでリモート・キュー・マネージャーに接続するためのユーザー名とパスワードを指定します。 この情報を指定しない場合、認証情報はリモート接続構成ファイルから取得されます。

g)「**次へ**」をクリックして「証明書」ページに移動します。 CCDT が「transmissionSecurity」情報を指定している場合、この情報が使用されます。 オプションで証明書を Base64 エンコード公開鍵として貼り付けることができます。これがグローバル・トラストストアに追加されます。

証明書がトラストストアに追加される前に、証明書が一時的に *WLP_USER_DIR*/generated.crts/*uniqueName-qmgrName*.crt に保管されます。 接続が正常に追加されると、証明書はこの場所から削除されます。

h)「**次へ**」をクリックして要約ページを表示します。「**戻る**」ボタンを使用すると、前のページに戻って修正を行うことができます。 この情報で問題がなければ、「**接続**」をクリックしてリモート・キュー・マネージャーに接続します。

- 手動でリモート・キュー・マネージャーを追加して接続情報を指定するには、以下のようにします。

a) ホーム・ページで「**リモート・キュー・マネージャーへの接続**」をクリックします。

b) リモート・キュー・マネージャーの名前を指定します。

c) オプションで、キュー・マネージャーの固有の名前を指定します。 固有の名前を指定しない場合は、実際に使用される名前には接頭部「remote-」が追加されます。

d)「**手動入力**」を選択します。

e) 接続で使用するクライアント接続チャネルの名前を入力します。

f) リモート・キュー・マネージャーが実行されているホストの名前を指定します。 リモートの MQ インストール済み環境が検出された場合は、ホスト名が表示され、接続先となるリモート・キュー・マネージャーのホストを選択することができます。 一部のネットワーク構成では、リモートの MQ インスタンスを検出することができません。 この場合は、ホスト名とポートを手動で追加します。

g)「**次へ**」をクリックしてユーザー・ページに移動します。オプションでリモート・キュー・マネージャーに接続するためのユーザー名とパスワードを指定します。 この情報を指定しない場合、認証情報はリモート接続構成ファイルから取得されます。

h)「**次へ**」をクリックして「証明書」ページに移動します。 ドロップダウン・リストから SSL CipherSpec を選択することができます。 オプションで証明書を Base64 エンコード公開鍵として貼り付けることができます。これがグローバル・トラストストアに追加されます。

証明書がトラストストアに追加される前に、証明書が一時的に *WLP_USER_DIR*/generated.crts/*uniqueName-qmgrName*.crt に保管されます。 接続が正常に追加されると、証明書はこの場所から削除されます。

i)「**次へ**」をクリックして要約ページを表示します。「**戻る**」ボタンを使用すると、前のページに戻って修正を行うことができます。 この情報で問題がなければ、「**接続**」をクリックしてリモート・キュー・マネージャーに接続します。

指定した接続情報は、ご使用の Web ディレクトリーにある CCDT ファイルに書き込まれます。 パスは *WLP_USER_DIR*/generated.ccdt/ccdt-*uniqueName* です。

タスクの結果

リモート・キュー・マネージャーが、IBM MQ Console のリモート・キュー・マネージャー・リストに表示されます。 接続が正常に行われると、ローカル・キュー・マネージャーのオブジェクトを扱う場合と同じ方法でリモート・キュー・マネージャーのオブジェクトを管理することができます。

V 9.4.0 コマンド行を使用した *IBM MQ Console* へのリモート・キュー・マネージャーの追加

コマンド行で **setmqweb remote** コマンドを使用して、リモート・キュー・マネージャーを IBM MQ Console に追加できます。 リモート・キュー・マネージャーは、IBM MQ Console と同じシステム上の別のインストール済み環境で実行されているキュー・マネージャーか、別のシステムで実行されているキュー・マネージャーのいずれかにすることができます。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。

- ALW の上 AIX, Linux, and Windows、MQSC コマンドを **runmqsc** コマンド・プロンプト。 見る MQSC コマンドを対話的に実行する **runmqsc** そしてテキストファイルから MQSC コマンドを実行する **runmqsc**。 このタスクでは、 AIX, Linux, and Windows で実行している場合、 QM1:

```
runmqsc QM1
```

- IBM i の上 IBM i スクリプトファイル内にコマンドのリストを作成し、**STRMQMMQSC** 指示。 見る MQSC コマンドを使用した管理 IBM i。
- z/OS の上 z/OSMQSC コマンドは、コマンドに応じて、さまざまなソースから発行できます。 見る MQSC および PCF コマンドを発行できるソース IBM MQ for z/OS。

mqweb サーバーが IBM MQ Console へのリモート・キュー・マネージャー接続を許可するように構成されていることを確認します。 詳しくは、 リモート・キュー・マネージャー接続動作の構成を参照してください。

手順

1. リモート・キュー・マネージャーで、 **DEFINE CHANNEL** MQSC コマンドを使用して、キュー・マネージャーのリモート管理を許可するサーバー接続チャネルを作成します。
   例えば、キュー・マネージャー QM1 のサーバー接続チャネル QM1.SVRCONN を作成するには、次の MQSC コマンドを入力します。

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

   DEFINE CHANNEL および使用可能なオプションについて詳しくは、DEFINE CHANNEL を参照してください。
2. キュー・マネージャーと、そのキュー・マネージャーに関連付けられている MQ オブジェクトを管理する権限が適切なユーザーに付与されていることを確認してください。
   - ALW AIX, Linux, and Windows では、標準コマンド行で **setmqaut** 制御コマンドを使用します。
   - z/OS z/OS では、 RACF プロファイルを定義して、許可ユーザーにキュー・マネージャーへのアクセス権限を付与します。

   例えば、 AIX, Linux, and Windows で、ユーザー exampleUser にキュー・マネージャー QM1 へのアクセスを許可するには、次の制御コマンドを入力します。

   ```
   setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +setall +dsp
   ```

   この許可ユーザーは、以下のいずれかのユーザーである可能性があります。
   - このキュー・マネージャーをリモート側で管理するシステム上で IBM MQ Console を実行する mqweb サーバーを開始するユーザー ID と同じユーザー ID。
   - ステップ 107 ページの『7』 の **setmqweb remote** コマンドに組み込まれるユーザー ID およびパスワードと一致するユーザー ID。 **setmqweb remote** コマンドにユーザー ID とパスワードを含めることにより、 IBM MQ Console がキュー・マネージャーに接続するときに、このユーザー ID とパスワードが認証に使用されます。
   - チャネル・セキュリティー規則によって決定されるユーザー ID。 例えば、サーバー接続チャネルでチャネル認証規則を設定して、リモート管理に IBM MQ Console を使用する IP アドレスからの接続を許可し、これらのすべての接続を、キュー・マネージャーの使用を許可されている特定のユーザー ID にマップすることができます。 詳しくは、 チャネルの新規 CHLAUTH ルールの作成を参照してください。
3. ALW

リモート・キュー・マネージャー上でリスナーが実行されていない場合は、**DEFINE LISTENER** MQSC コマンドを使用して、着信ネットワーク接続を受け入れるリスナーを作成します。
例えば、リモート・キュー・マネージャー QM1 のリスナー REMOTE.LISTENER をポート 1414 に作成するには、次の MQSC コマンドを入力します。

```
runmqsc QM1
DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
end
```

4. **START LISTENER** MQSC コマンドを使用して、リスナーが実行されていることを確認します。

   ALW 例えば、 AIX, Linux, and Windows でキュー・マネージャー QM1 のリスナー REMOTE.LISTENER を開始するには、次の MQSC コマンドを入力します。

   ```
   runmqsc QM1
   START LISTENER(REMOTE.LISTENER)
   end
   ```

   z/OS 例えば、 z/OS でリスナーを開始するには、次の MQSC コマンドを入力します。

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

   z/OS でリスナーを開始するには、その前にチャネル・イニシエーターのアドレス・スペースを開始する必要があります。
5. リモート・キュー・マネージャーの接続情報を格納する JSON CCDT ファイルを作成します。
   - リモート側で接続するキュー・マネージャーと同じインストール済み環境に関連付けられている IBM MQ Console を使用して、CCDT ファイルを生成します。

     「**ホーム**」パネルで「**接続ファイルをダウンロードします**」タイルをクリックします。
   - 接続を定義する JSON 形式の CCDT ファイルを作成します。 JSON 形式の CCDT の作成について詳しくは、JSON 形式の CCDT の構成を参照してください。

     CCDT ファイルには、name、clientConnection、および type の情報を含める必要があります。オプションで、transmissionSecurity 情報などの追加情報を含めることができます。 すべての CCDT チャネル属性定義について詳しくは、CCDT チャネル属性定義の完全なリストを参照してください。

     以下の例は、リモート・キュー・マネージャー接続のための基本的な JSON CCDT ファイルを示しています。 これは、ステップ 106 ページの『1』で作成したサーバー接続チャネルの例と同じ名前にチャネルの名前を設定し、リスナーによって使用されるポートと同じ値への接続ポートを設定します。 接続ホストは、リモート・キュー・マネージャーの例 (QM1) が実行されているシステムのホスト名に設定されます。

     ```
     {
       "channel": [{
           "name": "QM1.SVRCONN",
           "clientConnection": {
             "connection": [{
                 "host": "example.com",
                 "port": 1414
               }],
             "queueManager": "QM1"
           },
           "type": "clientConnection"
         }]
     }
     ```
6. IBM MQ Console が実行されているシステムに JSON CCDT ファイルをコピーします。
7. IBM MQ Console を実行しているインストール済み環境から、 **setmqweb remote** コマンドを使用して、リモート・キュー・マネージャーの情報を IBM MQ Console 構成に追加します。

   少なくとも、リモート・キュー・マネージャーを IBM MQ Console に追加するには、キュー・マネージャー名、キュー・マネージャーの固有の名前 (同じキュー・マネージャー名を持つ可能性がある他のリモート・キュー・マネージャーを区別するため) 、およびキュー・マネージャーの CCDT URL を指定す

る必要があります。 固有の名前は IBM MQ Console 内の表示名であるため、これがリモート・キュー・マネージャーであることを明確に示す名前 ("remote-QM2"など) を指定します。 他にも追加で指定できるオプションがいくつかあります。例えばリモート・キュー・マネージャー接続で使用するユーザー名とパスワード、トラストストアと鍵ストアの詳細などです。 **setmqweb remote** コマンドで指定できるパラメーターの完全なリストについては、 setmqweb remote を参照してください。

例えば、サンプル CCDT ファイルを使用してサンプル・リモート・キュー・マネージャー QM1 を追加するには、以下のコマンドを入力します。

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

タスクの結果

リモート接続リストが次にリフレッシュされるときに、リモート・キュー・マネージャーが IBM MQ Console のリモート・キュー・マネージャー・リストに表示されます。 接続が正常に行われると、ローカル・キュー・マネージャーのオブジェクトを扱う場合と同じ方法でリモート・キュー・マネージャーのオブジェクトを管理することができます。

例

以下の例では、キュー・マネージャー QM1 のリモート・キュー・マネージャー接続をセットアップします。 IBM MQ Console には、ユーザー exampleUser に付与された権限に基づいてキュー・マネージャーを管理する権限があります。 このユーザーの資格情報は、**setmqweb remote** コマンドを使用してリモート・キュー・マネージャー接続情報を構成するときに、 IBM MQ Console に提供されます。

1. リモート・キュー・マネージャー QM1 が存在するシステムで、サーバー接続チャネルとリスナーが作成されます。 リスナーが開始され、ユーザー exampleUser がキュー・マネージャーを管理するための許可が付与されます。 例えば、 AIX, Linux, and Windows では、以下のコマンドを実行します。

   ```
   runmqsc QM1
   #Define the server connection channel that will accept connections from the Console
   DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
   # Define the listener to use for the connection from the Console
   DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
   # Start the listener
   START LISTENER(REMOTE.LISTENER)
   end

   #Set mq authorization for exampleUser to access the queue manager
   setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +setall +dsp
   ```

2. IBM MQ Console が実行されているシステムで、以下の接続情報を含む QM1_ccdt.json ファイルが作成されます。

   ```
   {
     "channel": [{
         "name": "QM1.SVRCONN",
         "clientConnection": {
           "connection": [{
               "host": "example.com",
               "port": 1414
             }],
           "queueManager": "QM1"
         },
         "type": "clientConnection"
       }]
   }
   ```

3. IBM MQ Console が実行されているシステムで、キュー・マネージャー QM1 のリモート・キュー・マネージャー接続情報が mqweb サーバーに追加されます。 exampleUser の資格情報は、接続情報に含まれています。

   ```
   setmqweb remote add -uniqueName "remote-QM1" -qmgrName "QM1" -ccdtURL
   "c:\myccdts\QM1_ccdt.json" -username "exampleUser" -password "password"
   ```

4. IBM MQ Console は、リモート・キュー・マネージャー QM1 を示しています。

V 9.4.0 IBM MQ Console: オブジェクトの処理

各 IBM MQ キュー・マネージャーには、さまざまな種類のオブジェクトが関連付けられます。

このタスクについて

コンソールを使用して、以下のタイプの IBM MQ オブジェクトを操作できます。

- キュー
- イベント・オブジェクト:
  - トピック
  - サブスクリプション
- アプリケーション・オブジェクト:
  - 接続
  - アプリケーション・チャネル
  - アプリ・チャネル・インスタンス
- MQ ネットワーク・オブジェクト:
  - 接続されたキュー・マネージャー
  - キュー・マネージャー・チャネル
  - キュー・マネージャー・チャネル・インスタンス

手順

IBM MQ オブジェクトを操作するには、次のようにします。

1. キュー・マネージャーのリスト・ビューで、操作するオブジェクトを所有しているキュー・マネージャーをクリックします。
2. 「キュー」、「イベント」、「アプリケーション」、または MQ ネットワーク・タブをクリックして、処理するオブジェクトのタイプを選択します。
3. オブジェクトの処理について詳しくは、以下のいずれかのトピックを参照してください。

V 9.4.0 *IBM MQ Console:* キューの処理

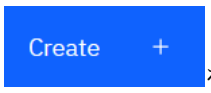
「**キュー**」タブで、特定のキュー・マネージャーに存在するキューを参照できます。 キューの追加と削除、キューでのメッセージの追加とクリア、メッセージの表示、キューのプロパティーの表示と設定、およびキューの権限レコードの管理を行えます。

このタスクについて

キューのビューに、特定のキュー・マネージャーに存在するキューがリストされます。 このキューのリストにアクセスするには、キュー・マネージャーをクリックし、「**キュー**」タブを選択します。 操作する個々のキューをリストから選択できます。

z/OS z/OS では、キューの権限レコードを表示および編集することはできません。

手順

- キューを追加するには、次のようにします。
  a)「**キュー ( )**」タブで、「作成」ボタン 

をクリックします。
  b) 作成するキューのタイプを選択します。
    - ローカル・キュー - このキューが属するキュー・マネージャーの内部に、メッセージを保管します。

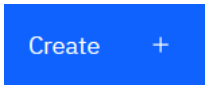
- 別名キュー - 同じキュー・マネージャーに属する別のキューを指すポインター。
- リモート・キュー - 別のキュー・マネージャーに属する別のキューを指すポインター。
- モデル・キュー - 動的キュー・マネージャーの作成時に使用されるキューのテンプレート。

c) 作成するキューのタイプについて、必要な情報を入力します。 デフォルトでは、値を入力できるように表示されるのは、最小限の推奨プロパティーです。 「**カスタム作成**」を選択すると、使用可能なすべてのプロパティーを表示できます。

d)「**作成**」 をクリックします。 新しいキューが作成されます。

- メッセージをキューに書き込むには、次のようにします。

a) キューのリスト・ビューで、メッセージを追加するキューをクリックします。 モデル・キューを選択することはできません。

b)「作成」ボタンをクリック 

c) キューに入れるメッセージを入力します。

d)「**作成**」 をクリックします。

- キューからメッセージを消去するには、次のようにします。

a) キュー・リストで、メッセージを消去するローカル・キューをクリックします。

b)「キューのクリア」アイコン  をクリックします。

c)「**キュー消去**」をクリックして、キューの消去を確定します。

- V 9.4.0

キューから個々のメッセージを削除するには、次のようにします。

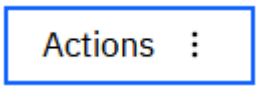
a) 削除したいメッセージを見つけます。

b) メッセージ  の横にある削除アイコンをクリックします。

c)「**削除**」をクリックして、メッセージをクリアすることを確認します。

- キュー上のメッセージをブラウズするには、キューのリスト・ビューでそのキューをクリックします。 そのキュー上のメッセージのリストが表示されます。
- キューを削除するには、次のようにします。

a) キュー・リストで、削除するローカル・キューをクリックします。

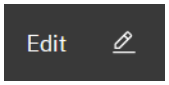
b)「アクション」ボタン 

をクリックし、 「**キューの削除**」を選択します。

c)「**削除**」をクリックして、キューの削除を確定します。 キューが削除されます。

- キューのプロパティーを表示および編集するには、次のようにします。

a) 編集するキューの横にあるメニュー ⋮ から 「**構成の表示**」 を選択します。

b)「編集」ボタンをクリック 

c) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。 プロパティーについては、 IBM MQ Explorer 資料の キュー・プロパティー を参照してください。

d)「**保存**」をクリックして、変更を保存します。

- キューの権限レコードを表示および編集するには、次のようにします。

a) 編集するキューの横にあるメニュー ⋮ から 「**構成の表示**」 を選択します。

b)「**セキュリティ**」タブをクリックします。

c) キュー・マネージャーの権限レコードについての説明に従って、権限レコードを操作します。 101 ページの『IBM MQ Console: キュー・マネージャー権限レコードの処理』 を参照してください。

- V 9.4.0
キューに関連付けられている IBM MQ オブジェクトを表示するには、以下のようにします。

a) 表示するキューの横にあるメニュー ⋮ から 「**関連オブジェクトの表示**」 を選択します。

b) 表示されるパネルにオブジェクトを表示します。 リンクをクリックすると、リストされている各オブジェクトに関する詳細が表示されます。

このパネルを使用して、どのアプリケーションがキューにメッセージを書き込んでいるかを表示したり、異なるキュー間の関係を確認したりすることができます。 これは、問題を特定して解決するのに役立ちます。

V 9.4.0 *IBM MQ Console: トピックの処理*

IBM MQ Console を使用して、トピックの追加と削除、トピックのプロパティーの表示と設定を行うことができます。

このタスクについて

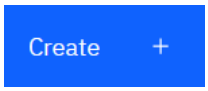
トピックのビューに、特定のキュー・マネージャーに存在するトピックがリストされます。 トピックには、キュー・マネージャーの 「**イベント**」 タブからアクセスします。 操作する個々のトピックをリストから選択できます。

z/OS z/OS では、トピックの権限レコードを表示および編集することはできません。

手順

- トピックを追加するには、次のようにします。

a) キュー・マネージャー・ビューで 「**イベント**」 タブを開き、 「**トピック**」をクリックします。

b)「作成」ボタンをクリック 

c) 作成するトピックについて、必要な情報を入力します。 デフォルトでは、値を入力できるように表示されるのは、最小限の推奨プロパティーです。 「**カスタム作成**」を選択すると、使用可能なすべてのプロパティーを表示できます。

d)「**作成**」 をクリックします。 新しいトピックが作成されます。

- トピックを削除するには、次のようにします。

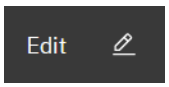
a) 削除するトピックの横にあるスパナのアイコン  をクリックします。

b)「キューの編集 (Edit queue)」ビューで、「**トピックの削除**」をクリックします。

c)「**削除**」をクリックして、トピックの削除を確定します。 トピックが削除されます。

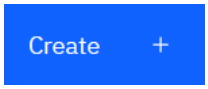
- トピックのプロパティーを表示および編集するには、次のようにします。

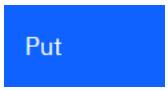
a) 編集するトピックの横にあるスパナのアイコン  をクリックします。

b)「編集」ボタンをクリック 

c) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。 プロパティーについて詳しくは、MQ エクスプローラーの資料のトピック・プロパティー を参照してください。

d)「**保存**」をクリックして、変更を保存します。

- トピックでメッセージをパブリッシュするには、一致するサブスクリプションが 1 つ以上存在する必要があります。
  a) トピックのリストで、パブリッシュするトピックをクリックします。
  b) 一致するサブスクリプション名をクリックします。
  c)「作成」ボタンをクリック 

  d) パブリッシュするメッセージを入力します。
  e)「 Put 」ボタンをクリックします 

メッセージが、一致するすべてのサブスクリプションに書き込まれます。
- トピックにサブスクライブするには、112 ページの『IBM MQ Console: サブスクリプションの処理』を参照してください。
- トピックの権限レコードを表示および編集するには、次のようにします。
  a) 権限レコードを編集するトピックの横にあるスパナのアイコン  をクリックします。
  b)「**セキュリティ**」タブをクリックします。
  c) キュー・マネージャーの権限レコードについての説明に従って、権限レコードを操作します (101 ページの『IBM MQ Console: キュー・マネージャー権限レコードの処理』を参照)。

V 9.4.0 *IBM MQ Console:* サブスクリプションの処理

IBM MQ Console を使用して、サブスクリプションの追加と削除、サブスクリプションのプロパティーの表示と設定を行うことができます。

このタスクについて

サブスクリプションのビューに、特定のキュー・マネージャーに存在するサブスクリプションがリストされます。 サブスクリプションには、キュー・マネージャーの 「**イベント**」 タブからアクセスします。 操作する個々のトピックをリストから選択できます。 操作する個々のサブスクリプションをリストから選択できます。

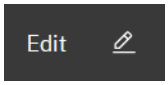
For more information about subscriptions, see サブスクライバーおよびサブスクリプション and サブの定義.

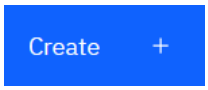
z/OS z/OS では、サブスクリプションの権限レコードを表示および編集することはできません。

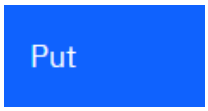
手順

- サブスクリプションを追加するには、次のようにします。
  a) キュー・マネージャー・ビューで 「**イベント**」 タブを開き、 「**サブスクリプション**」をクリックします。
  b) 管理対象サブスクリプションと非管理対象サブスクリプションのどちらを作成するかを選択します。
  c) 作成するサブスクリプションについて、必要な情報を入力します。 デフォルトでは、値を入力できるように表示されるのは、最小限の推奨プロパティーです。 「**カスタム作成**」を選択すると、使用可能なすべてのプロパティーを表示できます。
  d)「**作成**」 をクリックします。 新しいサブスクリプションが作成されます。
- サブスクリプションを削除するには、次のようにします。
  a) 削除するサブスクリプションの横にあるスパナのアイコン  をクリックします。
  b)「キューの編集 (Edit queue)」ビューで、「**サブスクリプションの削除**」をクリックします。

c)「**削除**」をクリックして、サブスクリプションの削除を確定します。 サブスクリプションは削除されます。

- サブスクリプションのプロパティーを表示および編集するには、次のようにします。
  a) 編集するサブスクリプションの横にあるスパナのアイコン  をクリックします。
  b)「編集」ボタンをクリック 

  c) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。
  d)「**保存**」をクリックして、変更を保存します。
- サブスクリプションでサブスクライブしているトピックでメッセージをパブリッシュするには、次のようにします。
  a) サブスクリプションのリストで、トピックにパブリッシュするサブスクリプションをクリックします。
  b)「作成」ボタンをクリック 

  c) パブリッシュするメッセージを入力します。
  d)「 Put 」ボタンをクリックします 

  パブリッシュしたトピックに一致するすべてのサブスクリプションにメッセージが書き込まれます。

V 9.4.0 *IBM MQ Console:* キュー・マネージャー・チャネルの操作

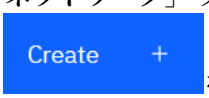
IBM MQ Console を使用して、キュー・マネージャーのチャネルを操作できます。キュー・マネージャーのチャネルの追加と削除、チャネルの開始と停止、チャネルのリセットと解決、チャネルの ping を行うことができます。 キュー・マネージャー・チャネルのプロパティーを表示および設定したり、チャネルの権限レコードを管理したりすることもできます。

このタスクについて

キュー・マネージャーのチャネルは、ネットワークを介してキュー・マネージャー間でメッセージを伝送するための論理的な通信リンクです。 キュー・マネージャーのチャネルのビューには、実行中のチャネルの数、再試行中のチャネルの数、停止したチャネルの数を示すクイック・ビューを表示するパネルが含まれています。

z/OS z/OS では、チャネルの権限レコードを表示および編集することはできません。

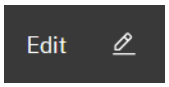
手順

- キュー・マネージャー・チャネルを追加するには、次のようにします。
  a) キュー・マネージャー・ビューで 「**MQ ネットワーク**」 タブを開き、 「**キュー・マネージャー・チャネル**」 をクリックし、「作成」ボタン 

  をクリックします。
  b) 作成するキュー・マネージャー・チャネルのタイプを選択して、次のボタン 

  をクリックします。
  c) 作成するチャネルについて、必要な情報を入力します。 デフォルトでは、値を入力できるように表示されるのは、最小限の推奨プロパティーです。 「**カスタム作成**」を選択すると、使用可能なすべてのプロパティーを表示できます。

d)「**作成**」 をクリックします。 **非アクティブ**状態の新しいチャネルが作成されます。

- キュー・マネージャー・チャネルを開始するには、次のようにします。
  a) 開始するチャネルをリスト内で見つけます。
  b) メニュー ⋮ から「**開始**」を選択します。
- キュー・マネージャー・チャネルを停止するには、次のようにします。
  a) 停止するチャネルをリスト内で見つけます。
  b) メニュー ⋮ から「**停止**」を選択します。
- キュー・マネージャー・チャネルのプロパティーを表示するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。
  c)「**プロパティー**」タブが選択されていることを確認します。 プロパティーを編集するには、「編集」ボタンをクリック 

。
  d) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。 プロパティーについて詳しくは、MQ エクスプローラーの資料のチャネル・プロパティーを参照してください。
  e)「**保存**」をクリックして、変更を保存します。
- キュー・マネージャー・チャネルをリセットするには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**拡張**」を選択します。
  c)「**リセット**」セクションで、メッセージ・シーケンス番号を指定します。
  送信する次のメッセージのシーケンス番号が送信側と受信側で異なるためにチャネルが開始されない場合は、チャネルをリセットする必要があります。 メッセージ・シーケンス番号で、その番号を指定します。
  d)「**チャネルのリセット**」をクリックします。
- 送信側チャネルまたはサーバー・チャネルを解決するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**拡張**」を選択します。
  c)「**解決**」セクションで、「**メッセージを伝送キューに復元する**」または「**メッセージを廃棄する**」をクリックして、メッセージの現行バッチをコミットするか、バックアウトするかを選択します。
- キュー・マネージャー・チャネルを ping するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**Ping**」を選択します。
- キュー・マネージャー・チャネルの権限レコードを表示および編集するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。
  c)「**セキュリティ**」タブをクリックします。

d) キュー・マネージャーの権限レコードについての説明に従って、権限レコードを操作します (101 ページの『IBM MQ Console: キュー・マネージャー権限レコードの処理』を参照)。

- キュー・マネージャー・チャネルを削除するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**構成**」を選択します。
  c)「**チャネルの削除**」をクリックします。

V 9.4.0 *IBM MQ Console: アプリケーション・チャネルの操作*

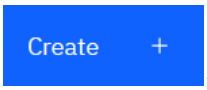
IBM MQ Console を使用して、アプリケーションのチャネルを操作できます。チャネルの追加と削除、チャネルの開始と停止、チャネルのリセットと解決、チャネルの ping を行うことができます。 アプリケーション・チャネルのプロパティーを表示および設定したり、チャネルの権限レコードを管理したりすることもできます。

このタスクについて

アプリケーションのチャネルは、アプリケーションがネットワークを介してキュー・マネージャーに接続するために使用する論理的な通信リンクです。 アプリケーションのチャネルのビューには、実行中のチャネルの数、再試行中のチャネルの数、停止したチャネルの数を示すクイック・ビューを表示するパネルが含まれています。

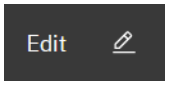
z/OS z/OS では、チャネルの権限レコードを表示および編集することはできません。

手順

- アプリケーション・チャネルを追加するには、次のようにします。
  a) キュー・マネージャー・ビューで 「**アプリケーション**」 タブを開き、 「**アプリケーション・チャネル**」 をクリックし、「作成」ボタン 

をクリックします。
  b) 次のボタン 

をクリックします。
  c) 作成するチャネルについて、必要な情報を入力します。 デフォルトでは、値を入力できるように表示されるのは、最小限の推奨プロパティーです。 「**カスタム作成**」を選択すると、使用可能なすべてのプロパティーを表示できます。
  d)「**作成**」 をクリックします。 **非アクティブ**状態の新しいチャネルが作成されます。
- アプリケーション・チャネルを開始するには、以下のようにします。
  a) 開始するチャネルをリスト内で見つけます。
  b) メニュー ⋮ から「**開始**」を選択します。
- アプリケーション・チャネルを停止するには、次のようにします。
  a) 停止するチャネルをリスト内で見つけます。
  b) メニュー ⋮ から「**停止**」を選択します。
- アプリケーション・チャネルのプロパティーを表示するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から 「**構成の表示**」 を選択します。

c)「**プロパティー**」タブが選択されていることを確認します。 プロパティーを編集するには、「編集」ボタンをクリック 

。

d) 必要に応じてプロパティーを編集してください。 プロパティー・テキスト・ボックスが無効になっている場合、プロパティーは読み取り専用であるか、コマンド行からしか編集できません。 プロパティーについて詳しくは、MQ エクスプローラーの資料のチャネル・プロパティー を参照してください。

e)「**保存**」をクリックして、変更を保存します。

- アプリケーション・チャネルをリセットするには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**拡張**」を選択します。
  c)「**リセット**」セクションで、メッセージ・シーケンス番号を指定します。
  送信する次のメッセージのシーケンス番号が送信側と受信側で異なるためにチャネルが開始されない場合は、チャネルをリセットする必要があります。 メッセージ・シーケンス番号で、その番号を指定します。
  d)「**チャネルのリセット**」をクリックします。
- 送信側チャネルまたはサーバー・チャネルを解決するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**拡張**」を選択します。
  c)「**解決**」セクションで、「**メッセージを伝送キューに復元する**」または「**メッセージを廃棄する**」をクリックして、メッセージの現行バッチをコミットするか、バックアウトするかを選択します。
- チャネルを ping するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**Ping**」を選択します。
- アプリケーション・チャネルの権限レコードを表示および編集するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**構成**」を選択します。
  c)「**セキュリティ**」タブをクリックします。
  d) キュー・マネージャーの権限レコードについての説明に従って、権限レコードを操作します (101 ページの『IBM MQ Console: キュー・マネージャー権限レコードの処理』 を参照)。
- アプリケーション・チャネルを削除するには、次のようにします。
  a) リスト内でチャネルを見つけます。
  b) メニュー ⋮ から「**構成**」を選択します。
  c)「**チャネルの削除**」をクリックします。

V 9.4.0 *IBM MQ Console: アプリケーションの操作*

IBM MQ Console を使用して、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に関する情報を表示できます。

このタスクについて

アプリケーションは、サーバー接続チャネルを使用してネットワーク経由でキュー・マネージャーに接続されます。 アプリケーション・ビューには、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の数を示すクイック・ビューを表示するパネルが含まれています。

手順

- アプリケーション情報を表示するには:
  a) キュー・マネージャー・ビューから、 「**アプリケーション**」 タブを開きます。
  b)「**接続されたアプリケーション**」 をクリックして、アプリケーション・ビューを開きます。
  c) アプリケーションのインスタンスが複数ある場合は、下矢印をクリックして各インスタンスの詳細を表示します。
  d) ビュー内のオブジェクトをクリックすると、詳細が表示されます。

V 9.4.0 z/OS ***IBM MQ Console: Working with storage classes***

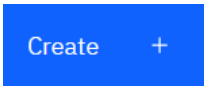
You can use the IBM MQ Console to add, view, delete and update storage classes on z/OS queue managers.

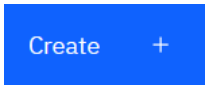
About this task

The storage classes view lists the storage classes that exist for a specific queue manager. You access **Storage classes** from the sidebar on the queue manager **Queues** tab.

See Storage classes for IBM MQ for z/OS and DEFINE STGLASS for more information about storage classes.

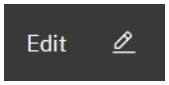
Procedure

- To add a storage class:
  a) From the queue manager view, open the **Queues** tab, and click **Storage classes**.
  b) On the **Storage classes** screen, click the **Create** 

button.
  c) Provide the required information for the storage class you are creating.
  By default, the minimum recommended properties you need to provide values for are displayed. You can view all of the available properties by selecting **Custom create**.
  d) Click the **Create** 

button.
  The new storage class is created.
- To delete a storage class:
  a) Click the spanner button  next to the storage class that you want to delete.
  b) In the Edit storage class view, click **Delete storage class**.
  c) Confirm that you want to delete the queue by clicking **Delete**. The storage class is deleted.
- To view and edit the properties of a storage class:

a) Click the spanner button next to the storage class that you want to edit.

b) Click the Edit button 

c) Edit the properties as required. If the property text box is disabled, the property is read-only, or can be set only at the time of creation.

d) Click **Save** to save your changes.

V 9.4.0 V 9.4.0 z/OS ***IBM MQ Console: Working with page sets and buffer pools***

You can use the IBM MQ Console to view page sets and buffer pools on z/OS queue managers.

About this task

The page sets and buffer pools views list the page sets and buffer pools that exist for a specific queue manager. You access the **Page sets** and **Buffer pools** views from the sidebar of the queue manager **Queues** tab

See Page sets for IBM MQ for z/OS for more information about page sets, and Buffers and buffer pools for IBM MQ for z/OS for more information about buffer pools.

Procedure

- To view the properties of a page set

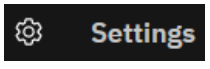
  Click the spanner button next to the page set that you want to view.
- To view the properties of a buffer pool

  Click the spanner button next to the buffer pool that you want to view.

IBM MQ Console の設定

IBM MQ Console の一般設定をいくつか指定できます。

設定アイコン 

をクリックして、 IBM MQ Console 設定ビューに切り替えます。

この設定を使用して以下の機能を制御できます。

- キュー・マネージャーの自動リフレッシュ (10 秒間隔)。 この機能をオン/オフにすることができます。
- システム・オブジェクトを表示するかどうか。 これをすべてのオブジェクト・タイプに対して指定することも、個々にオブジェクト・タイプを選択することもできます。
- トレース情報を収集するかどうか。

Windows Linux IBM MQ Explorer を使用した管理

IBM MQ Explorer を使用すると、Windows、または Linux x86-64 のみが稼働するコンピューターから、ネットワークをローカル側またはリモート側で管理することができます。

IBM MQ for Windows および IBM MQ for Linux x86-64 には、管理タスクを実行するため、制御コマンドや MQSC コマンドを使用する代わりに IBM MQ Explorer という管理インターフェースが用意されています。 コマンド・セットの比較には、IBM MQ Explorer を使用して実行できる操作内容が示されています。

IBM MQ Explorer を使用すると、Windows または Linux x86-64 を実行しているコンピューターから、関心のあるキュー・マネージャーおよびクラスターの IBM MQ Explorer を指定して、ネットワークのローカル管理またはリモート管理を実行できます。 サポートされるプラットフォーム (z/OS を含む)で稼働中のキュー・マネージャーにリモートで接続することができるので、コンソールから、メッセージング・バックボーン全体を表示、探索、および変更することができます。

IBM MQ Explorer がリモート IBM MQ キュー・マネージャーを管理できるように構成するには、121 ページの『IBM MQ Explorer の前提ソフトウェアと定義』を参照してください。

これにより、Windows または Linux x86-64 システム・ドメイン内でローカルまたはリモートで、IBM MQ の作業環境のセットアップおよび微調整に関連する通常のタスクを実行できます。

Linux では、複数の Eclipse がインストールされている場合、IBM MQ Explorer の開始に失敗することがあります。 その場合、もう一方の Eclipse のインストールで使用するのとは異なるユーザー ID を使用して、IBM MQ Explorer を開始します。

Linux では、IBM MQ Explorer を正常に開始するために、ファイルをホーム・ディレクトリーに書き込めることと、ホーム・ディレクトリーが存在していることが必要です。

IBM MQ Explorer は、 Fix Central から入手可能なスタンドアロン IBM MQ Explorer ダウンロードからインストールできます。 詳しくは、 Linux および Windows でのスタンドアロン・アプリケーションとしての IBM MQ Explorer のインストールおよびアンインストールを参照してください。

Windows Linux IBM MQ Explorer で実行できる処理

IBM MQ Explorer で一連のコンテンツ・ビューとプロパティー・ダイアログを使用して、管理タスクを実行できます。 1 つ以上の Eclipse プラグインを作成して、IBM MQ Explorer を拡張することもできます。

IBM MQ Explorer のタスク

IBM MQ Explorer を使用して、以下のタスクを実行できます。

- キュー・マネージャーの作成と削除 (ローカル・マシン上のキュー・マネージャーのみ)。
- キュー・マネージャーの起動と停止 (ローカル・マシン上のキュー・マネージャーのみ)。
- キュー、チャネルなどの IBM MQ オブジェクトの定義、定義の表示、変更。
- キュー内のメッセージの表示。
- チャネルの開始と停止。
- チャネル、リスナー、キュー、およびサービス・オブジェクトの状況情報の表示。
- クラスターを構成するキュー・マネージャーの表示
- 特定のキューがオープンしているアプリケーション、ユーザー、またはチャネルの検査を参照してください。
- 「新しいクラスターの作成」ウィザードによる、新しいキュー・マネージャー・クラスターの作成。
- 「クラスターへのキュー・マネージャーの追加」ウィザードによる、クラスターへのキュー・マネージャーの追加。
- Transport Layer Security (TLS) チャネル・セキュリティーで使用される、認証情報オブジェクトの管理。
- チャネル・イニシエーター、トリガー・モニター、およびリスナーの作成と削除。
- コマンド・サーバー、チャネル・イニシエーター、トリガー・モニター、およびリスナーの始動/停止。
- キュー・マネージャーの始動時に特定のサービスを自動で開始するための設定。
- キュー・マネージャーのプロパティーの変更。
- ローカルのデフォルト・キュー・マネージャーの変更。
- IBM MQ オブジェクトから JMS オブジェクトを作成する、および JMS オブジェクトからの IBM MQ オブジェクト。

- 現在サポートされる任意のタイプ用の JMS 接続ファクトリーの作成。
- リスナー用の TCP ポート番号やチャネル・イニシエーター・キュー名など、任意のサービスのパラメーターの変更。
- サービス・トレースの始動/停止。

コンテンツ・ビューとプロパティー・ダイアログ

管理タスクは、一連のコンテンツ・ビューとプロパティー・ダイアログを使用して実行します。

コンテンツ・ビュー
コンテンツ・ビューは、次の情報を表示するパネルです。

- 属性、および IBM MQ 自体に関連する管理オプション。
- 属性、および 1 つ以上の関連オブジェクトに関連する管理オプション。
- 属性、およびクラスターの管理オプション

プロパティー・ダイアログ
プロパティー・ダイアログは、一連のフィールドに 1 つのオブジェクトに関連した属性を表示したパネルで、属性の一部は編集できます。

IBM MQ Explorer をナビゲートするには、ナビゲーター・ビューを使用します。 ナビゲーターにより、必要なコンテンツ・ビューを選択できます。

IBM MQ Explorer の拡張

IBM MQ Explorer では、Eclipse フレームワークや Eclipse がサポートする他のプラグイン・アプリケーションに整合したスタイルで情報が表示されます。

IBM MQ Explorer を拡張すると、システム管理者は、IBM MQ Explorer をカスタマイズして IBM MQ を管理する方法を改善できます。

詳しくは、MQ エクスプローラーの拡張を参照してください。

Windows Linux IBM MQ Explorer を使用するかどうかの決定

ご使用のシステムで IBM MQ Explorer を使用するかどうかを決める際には、このトピックにリストされている情報について考慮してください。

以下の点に留意する必要があります。

オブジェクト名
IBM MQ Explorer を使用してキュー・マネージャーおよびその他のオブジェクトに小文字の名前を使用する場合には、MQSC コマンドを使用してオブジェクトを処理するときに、オブジェクト名を単一引用符で囲む必要があります。単一引用符で囲まない場合、IBM MQ はオブジェクト名を認識しません。

大型キュー・マネージャー
IBM MQ Explorer は、小型のキュー・マネージャーで最大限の効果を発揮します。 1 つのキュー・マネージャーの中に大量のオブジェクトが格納されている場合、表示に必要な情報を IBM MQ Explorer が抽出するのに時間がかかる場合があります。

クラスター
IBM MQ では、何百あるいは何千のキュー・マネージャーのクラスターを構成することが可能です。 IBM MQ Explorer は、ツリー構造を使用してクラスター内のキュー・マネージャーを表現します。 クラスターの物理的なサイズは、IBM MQ Explorer の動作速度にあまり影響しません。そのクラスター内のキュー・マネージャーを選択するまで、IBM MQ Explorer がそれらに接続することはないからです。

IBM MQ Explorer の設定

この項では、IBM MQ Explorer をセットアップするのに必要なステップについて概説します。

- 121 ページの『IBM MQ Explorer の前提ソフトウェアと定義』
- 121 ページの『IBM MQ Explorer のセキュリティー』

- 125 ページの『IBM MQ Explorer でのキュー・マネージャーとクラスターの表示と非表示』
- 126 ページの『クラスター・メンバーシップと IBM MQ Explorer』
- 126 ページの『IBM MQ Explorer のデータ変換』

IBM MQ Explorer の前提ソフトウェアと定義

IBM MQ Explorer を使用する前に、以下の要件を満たしている必要があります。

IBM MQ Explorer によるリモート・キュー・マネージャーへの接続に使用できるのは、TCP/IP 通信プロトコルのみです。

次の項目について確認します。

1. コマンド・サーバーが、すべてのリモート管理キュー・マネージャーで稼働している。
2. 適切な TCP/IP リスナー・オブジェクトがすべてのリモート・キュー・マネージャーで実行されている。このオブジェクトは、IBM MQ リスナーでかまいません。または、AIX and Linux システムでは inetd デーモンでもかまいません。
3. デフォルト名が SYSTEM.ADMIN.SVRCONN のサーバー接続チャネルが、すべてのリモート・キュー・マネージャーに存在する。

   このチャネルは、次の MQSC コマンドを使用して作成できます。

   ```
   DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
   ```

   このコマンドにより、基本的なチャネル定義が作成されます。 セキュリティーの設定などで、より複雑な定義を作成する場合は、追加のパラメーターが必要です。 詳しくは、DEFINE CHANNEL を参照してください。
4. システム・キュー SYSTEM.MQEXPLORER.REPLY.MODEL が存在している。

IBM MQ Explorer のセキュリティー

特定のオブジェクトへのユーザー・アクセスを制御する必要のある環境で IBM MQ を使用する場合、IBM MQ Explorer の使用におけるセキュリティー問題について検討する必要があります。

IBM MQ Explorer の使用許可

任意のユーザーが IBM MQ Explorer を使用できますが、キュー・マネージャーに接続、アクセス、およびキュー・マネージャーを管理するには一定の権限が必要です。

IBM MQ Explorer を使用してローカル管理用タスクを実行するには、ユーザーが管理用タスクを実行するのに必要な権限を持っている必要があります。 ユーザーが mqm グループのメンバーである場合は、すべてのローカル管理用タスクを実行する権限を持っています。

IBM MQ Explorer を使用してリモート・キュー・マネージャーに接続し、リモート管理用タスクを実行する場合、IBM MQ Explorer を実行するユーザーは次の権限を持っている必要があります。

- ターゲット・キュー・マネージャー・オブジェクトでの CONNECT 権限
- ターゲット・キュー・マネージャー・オブジェクトでの INQUIRE 権限
- ターゲット・キュー・マネージャー・オブジェクトでの DISPLAY 権限
- キュー SYSTEM.MQEXPLORER.REPLY.MODEL に対する INQUIRE 権限
- キュー SYSTEM.MQEXPLORER.REPLY.MODEL に対する DISPLAY 権限
- キュー SYSTEM.MQEXPLORER.REPLY.MODEL に対する INPUT (取得) 権限
- キュー SYSTEM.MQEXPLORER.REPLY.MODEL に対する OUTPUT (書き込み) 権限
- キュー SYSTEM.ADMIN.COMMAND.QUEUE に対する OUTPUT (書き込み) 権限
- キュー SYSTEM.ADMIN.COMMAND.QUEUE に対する INQUIRE 権限
- 選択したアクションを実行する権限

注 : INPUT 権限は、キューからユーザーへの入力 (取得操作) に関係します。 OUTPUT 権限は、ユーザーからキューへの出力 (書き込み操作) に関係します。

IBM MQ for z/OS 上のリモート・キュー・マネージャーに接続し、IBM MQ Explorer を使用してリモート管理用タスクを実行するには、以下を指定する必要があります。

- システム・キューの RACF® プロファイル、 SYSTEM.MQEXPLORER.REPLY.MODEL
- AMQ.MQEXPLORER.* キューの RACF プロファイル

さらに、IBM MQ Explorer を実行するユーザーには次の権限が必要です。

- RACF システム・キューに対する UPDATE 権限、 SYSTEM.MQEXPLORER.REPLY.MODEL
- AMQ.MQEXPLORER.* キューに対する RACF UPDATE 権限
- ターゲット・キュー・マネージャー・オブジェクトでの CONNECT 権限
- 選択したアクションを実行する権限
- MQCMDS クラスのすべての hlq.DISPLAY.object プロファイルに対する READ 権限

IBM MQ オブジェクトに権限を付与する方法については、 AIX, Linux, and Windows システムでの IBM MQ オブジェクトへのアクセス権限の付与を参照してください。

ユーザーが、実行する許可が与えられていないオペレーションを実行しようとすると、ターゲット・キュー・マネージャーが許可障害プロシージャーを呼び出し、そのオペレーションは失敗します。

IBM MQ Explorer のデフォルト・フィルターは、すべての IBM MQ オブジェクトを表示することになっています。 ユーザーが DISPLAY 権限を持っていない IBM MQ オブジェクトがあると、許可障害が生成されます。 権限イベントが記録される場合は、表示の対象を、ユーザーの DISPLAY 権限の対象であるオブジェクト範囲のみに制限してください。

IBM MQ Explorer からリモート・キュー・マネージャーに接続するためのセキュリティー

IBM MQ Explorer と各リモート・キュー・マネージャー間のチャネルを保護する必要があります。

IBM MQ Explorer は、MQI クライアント・アプリケーションとして、リモート・キュー・マネージャーに接続します。 したがって、リモートの各キュー・マネージャーには、サーバー接続チャネル定義と適切な TCP/IP リスナーがなければなりません。 サーバー接続チャネルを保護していない場合、悪意のあるアプリケーションが同じサーバー接続チャネルに接続し、無制限の権限によりキュー・マネージャー・オブジェクトにアクセスしてしまう可能性があります。 サーバー接続チャネルを保護するためには、チャネルの MCAUSER 属性に非ブランクの値を指定するか、チャネル認証レコードを使用するか、またはセキュリティー出口を使用します。

MCAUSER 属性のデフォルト値は、ローカルのユーザー ID です。 サーバー接続チャネルの MCAUSER 属性にブランク以外のユーザー名を指定した場合、このチャネルを使用してキュー・マネージャーに接続するプログラムは、すべて、指定されたユーザー ID で実行され、同じレベルの権限を持つことになります。 チャネル認証レコードを使用している場合には、これは起こりません。

IBM MQ Explorer でのセキュリティー出口の使用

IBM MQ Explorer を使用して、デフォルトのセキュリティー出口とキュー・マネージャー固有のセキュリティー出口を指定できます。

IBM MQ Explorer から、新しいすべてのクライアント接続で使用されるデフォルトのセキュリティー出口を定義できます。 このデフォルト出口は、接続を作成する際に指定変更できます。 また、セキュリティー出口は単一のキュー・マネージャーに対しても一連のキュー・マネージャーに対しても定義可能であり、これは接続を作成する際に有効になります。 出口は、IBM MQ Explorer を使用して指定します。 詳細については、IBM MQ Explorer のヘルプを参照してください。

IBM MQ Explorer による *TLS* 対応の *MQI* チャネルを使用したリモート・キュー・マネージャーへの接続

IBM MQ Explorer は、MQI チャネルを使用してリモート・キュー・マネージャーに接続します。 TLS セキュリティーを使用して MQI チャネルを保護する場合は、クライアント・チャネル定義テーブルを使用して MQI チャネルを確立する必要があります。

クライアント・チャネル定義テーブルを使用して MQI チャネルを確立する方法については、「IBM MQ MQI clients」を参照してください。

クライアント・チャネル定義テーブルを使用してチャネルを確立すると、 123 ページの『リモート・キュー・マネージャーをホストするシステム上でのタスク』 および 123 ページの『IBM MQ Explorer をホストするシステム上でのタスク』で説明されているように、 IBM MQ Explorer を使用して、TLS 対応の MQI チャネルを使用してリモート・キュー・マネージャーに接続できます。

リモート・キュー・マネージャーをホストするシステム上でのタスク

リモート・キュー・マネージャーをホストするシステムでは、次のタスクを実行します。

1. チャネルのサーバー接続とクライアント接続のペアを定義し、両方のチャネルのサーバー接続の *SSLCIPH* 属性に適切な値を指定します。 *SSLCIPH* 属性について詳しくは、 「TLS を使用したチャネルの保護」を参照してください。
2. キュー・マネージャーの `@ipcc` ディレクトリーにあるチャネル定義テーブル `AMQCLCHL.TAB` を、IBM MQ Explorer をホストしているシステムに送信します。
3. 指定されたポートで TCP/IP リスナーを開始します。
4. CA と個人の TLS 証明書を両方とも、キュー・マネージャーの SSL ディレクトリーに配置します。
   - Linux AIX AIX and Linux システムの場合 `/var/mqm/qmgrs/+QMNAME+/SSL`
   - Windows Windows システムの場合 `C:&#xa5;Program Files&#xa5;IBM&#xa5;MQ\qmgrs\+QMNAME+\SSL`

     ここで、 `+QMNAME+` は、キュー・マネージャーの名前を表すトークンです。
5. `key.kdb` という名前の CMS タイプの鍵データベース・ファイルを作成します。 鍵データベースの作成に使用する **runmqakm** コマンドで `-stash` パラメーターを指定して、鍵データベースのパスワードをファイルに隠しておきます。
6. CA 証明書を直前のステップで作成したキー・データベースに追加します。
7. キュー・マネージャーの個人証明書をキー・データベースにインポートします。

Windows システムでの TLS の処理方法の詳細については、AIX, Linux, and Windows での TLS の取り扱いを参照してください。

IBM MQ Explorer をホストするシステム上でのタスク

IBM MQ Explorer をホストするシステムでは、以下のタスクを実行します。

1. `key.jks` という名前のタイプ JKS の鍵データベース・ファイルを作成します。 このキー・データベース・ファイル のパスワードを設定します。

   IBM MQ Explorer が TLS セキュリティー用に使用する鍵ストアは、Java 鍵ストア (JKS) ファイルでなければなりません。
2. CA 証明書を直前のステップで作成したキー・データベースに追加します。
3. キュー・マネージャーの個人証明書をキー・データベースにインポートします。
4. Windows および Linux システムでは、システム・メニュー、 `MQExplorer` 実行可能ファイル、または **strmqcfg** コマンドを使用して IBM MQ Explorer を開始します。
5. IBM MQ Explorer・ツールバーから「**ウィンドウ**」**->**「**設定**」をクリックしてから、**IBM MQ エクスプローラー**を展開し、「**SSL Client 証明書ストア**」をクリックします。 123 ページの『IBM MQ Explorer をホストするシステム上でのタスク』のステップ 1 で作成された JKS ファイルの名前およびパスワードをトラステッド証明書ストアおよび個人証明書ストアの両方に入力してから、「**OK**」をクリックします。
6. 「**設定**」ウィンドウを閉じ、「**キュー・マネージャー**」を右クリックします。 「**キュー・マネージャーの表示/非表示**」をクリックしてから、「**キュー・マネージャーの表示/非表示**」画面で 「**追加**」をクリックします。

7. キュー・マネージャーの名前を入力し、「**直接接続する**」オプションを選択します。 「次へ」をクリックします。
8. 「**クライアント・チャネル定義テーブルを使用 (CCDT)**」を選択し、リモート・キュー・マネージャーをホストしているシステム上の 123 ページの『リモート・キュー・マネージャーをホストするシステム上でのタスク』のステップ 2 でリモート・キュー・マネージャーから転送したチャネル・テーブル・ファイルの位置を指定します。
9. 「**完了 (Finish)**」 をクリックします。 IBM MQ Explorer からリモート・キュー・マネージャーにアクセスできるようになりました。

IBM MQ Explorer で別のキュー・マネージャーを経由して接続する方法

IBM MQ Explorer を使うと、IBM MQ Explorer が既に接続している中間キュー・マネージャーを経由して、キュー・マネージャーに接続することができます。

この場合は、IBM MQ Explorer が PCF コマンド・メッセージを中間キュー・マネージャーに書き込み、次の点を指定します。

- ターゲット・キュー・マネージャーの名前としての、オブジェクト記述子 (MQOD) の *ObjectQMgrName* パラメーター。 キュー名の解決について詳しくは、ネーム・レゾリューションを参照してください。
- ローカル・ユーザー ID としての、メッセージ記述子 (MQMD) の *UserIdentifier* パラメーター。

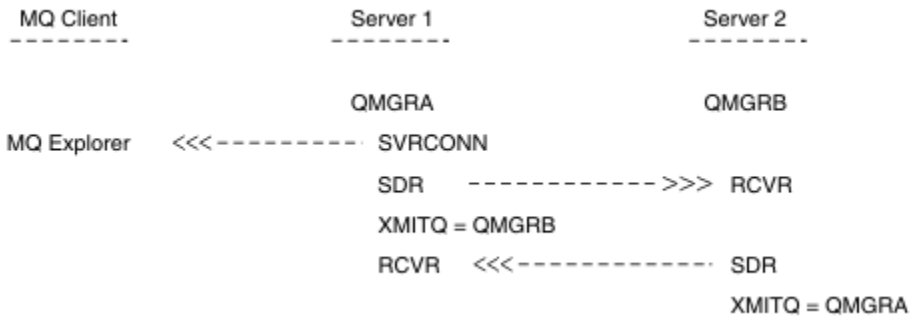
この接続が、中間キュー・マネージャーを経由してターゲット・キュー・マネージャーに接続するために使用される場合は、再びユーザー ID が、メッセージ記述子 (MQMD) の *UserIdentifier* パラメーターとして送られます。 ターゲット・キュー・マネージャーの MCA リスナーがこのメッセージを受け取るには、MCAUSER 属性を設定するか、書き込み権限を持つユーザー ID がすでに存在している必要があります。

ターゲット・キュー・マネージャーのコマンド・サーバーは、メッセージ記述子 (MQMD) の *UserIdentifier* パラメーターにあるユーザー ID を指定して、メッセージを伝送キューに書き込みます。 この書き込みが正常に行われるには、そのユーザー ID が書き込み権限付きで、ターゲット・キュー・マネージャーにすでに存在している必要があります。

以下の例は、中間キュー・マネージャーを経由して、キュー・マネージャーから IBM MQ Explorer に接続する方法を示しています。

キュー・マネージャーへのリモート管理接続を確立します。 以下の点を検証します。

- サーバー上のキュー・マネージャーは、アクティブであって、しかもサーバー接続チャネル (SVRCONN) が定義されている。
- リスナーがアクティブである。
- コマンド・サーバーがアクティブである。
- SYSTEM.MQ EXPLORER.REPLY.MODEL キューが作成済みであり、しかも十分な権限を持っている。
- キュー・マネージャーのリスナー、コマンド・サーバー、および送信側のチャネルが開始済みである。



この例のそれぞれの指定の意味は次のとおりです。

- IBM MQ Explorer は、クライアント接続を使用してキュー・マネージャー QMGRA (サーバー 1 上で稼働) に接続されています。

- Server2 上のキュー・マネージャー QMGRB は、中間キュー・マネージャー (QMGRA) を介して IBM MQ Explorer に接続できるようになりました。
- QMGRB を IBM MQ Explorer に接続するときは、中間キュー・マネージャーとして QMGRA を選択します。

この状態では、 IBM MQ Explorer から QMGRB への直接接続はありません。 QMGRB への接続は QMGRA を介して行われます。

サーバー 2 上のキュー・マネージャー QMGRB は、送信側/受信側チャネルを使ってサーバー 1 上の QMGRA に接続されます。 QMGRA と QMGRB の間のチャネルは、リモート管理可能なようにセットアップされる必要があります。196 ページの『リモート管理のためのキュー・マネージャーの構成』を参照してください。

IBM MQ Explorer でのキュー・マネージャーとクラスターの表示と非表示

IBM MQ Explorer では、一度に複数のキュー・マネージャーを表示できます。 「キュー・マネージャーの表示/非表示」パネル (キュー・マネージャー・ツリー・ノードのメニューから選択可能) により、別の (リモートの) マシンに関する情報を表示するかどうかを選択できます。 ローカル・キュー・マネージャーは自動的に検出されます。

リモート・キュー・マネージャーを表示するには、次の手順に従います。

1. 「**キュー・マネージャー**」ツリー・ノードを右クリックし、「**キュー・マネージャーの表示/非表示**」を選択します。
2. **追加** をクリックします。 「Show/Hide Queue Managers (キュー・マネージャーの表示/非表示)」パネルが表示されます。
3. リモート・キュー・マネージャーの名前、およびホスト名または IP アドレスを該当するフィールドに入力します。

   ホスト名または IP アドレスは、クライアントのデフォルトのサーバー接続チャネルである SYSTEM.ADMIN.SVRCONN か、またはユーザー定義のサーバー接続チャネルを使用して、リモート・キュー・マネージャーへのクライアント接続を確立するときに使用されます。
4. 「**完了 (Finish)**」 をクリックします。

「キュー・マネージャーの表示/非表示」パネルには、すべての可視キュー・マネージャーのリストも表示されます。 このパネルを使用して、ナビゲーション・ビューからキュー・マネージャーを隠すこともできます。

IBM MQ Explorer にクラスターのメンバーであるキュー・マネージャーが表示されると、クラスターが検出され、自動的に表示されます。

このパネルからリモート・キュー・マネージャーのリストをエクスポートするには、次の手順に従います。

1. 「Show/Hide Queue Managers (キュー・マネージャーの表示/非表示)」パネルを閉じます。
2. IBM MQ Explorer のナビゲーション・ペインで最も高い **IBM MQ** ツリー・ノードを右クリックし、「**エクスポート**」 **IBM MQ Explorer** 「**設定**」 を選択します。
3. 「**IBM MQ Explorer**」 **>** 「**IBM MQ Explorer 設定**」をクリックします。
4. 「**接続情報**」 **>** 「**リモート・キュー・マネージャー**」を選択します。
5. エクスポートされた設定を保管するファイルを選択します。
6. 最後に、「**完了**」をクリックして、リモート・キュー・マネージャーの接続情報を指定したファイルにエクスポートします。

リモート・キュー・マネージャーのリストをインポートするには、次の手順に従います。

1. IBM MQ Explorer のナビゲーション・ペインで最上位の **IBM MQ** ツリー・ノードを右クリックし、「**インポート**」 **IBM MQ Explorer** 「**設定**」 を選択します。
2. 「**IBM MQ Explorer**」 **>** 「**IBM MQ Explorer 設定**」をクリックします。
3. 「**Browse (参照)**」をクリックして、リモート・キュー・マネージャーの接続情報を含むファイルのパスに ナビゲートします。
4. 「**Open (オープン)**」をクリックする。 ファイルにリモート・キュー・マネージャーのリストが含まれる場合、「**接続情報**」 **>** 「**リモート・キュー・マネージャー**」ボックスが選択されます。

5. 最後に、「**完了**」をクリックして、リモート・キュー・マネージャーの接続情報を IBM MQ Explorer にインポートします。

クラスター・メンバーシップと IBM MQ Explorer

IBM MQ Explorer は、クラスターのメンバーであるキュー・マネージャーについての情報を必要とします。

キュー・マネージャーがクラスターのメンバーである場合は、クラスター・ツリー・ノードにデータが自動的に取り込まれます。

IBM MQ Explorer の稼働中にキュー・マネージャーがクラスターのメンバーになった場合、クラスターに関する最新の管理データに合わせて IBM MQ Explorer を保守して、このエクスプローラーがクラスターと効率よく通信し、要求された時には正しいクラスター情報を表示できるようにする必要があります。 そのため、以下の情報を IBM MQ Explorer に提供する必要があります。

- リポジトリー・キュー・マネージャー。
- リポジトリー・キュー・マネージャーがリモートのキュー・マネージャーにある場合は、その接続名。

この情報によって、IBM MQ Explorer では以下のことが可能になります。

- リポジトリー・キュー・マネージャーを使用して、クラスター内のキュー・マネージャーのリストを取得する。
- クラスター・メンバーであるキュー・マネージャーを管理する (ただし、サポートされるプラットフォームおよびコマンド・レベルであること)。

以下の場合は、管理できません。

- 選択されたリポジトリーが利用不可能になった。 IBM MQ Explorer は、代わりのリポジトリーに自動的に切り替わりません。
- 選択されたリポジトリーが TCP/IP ではアクセスできない。
- 選択されたリポジトリーのあるキュー・マネージャーが稼働しているプラットフォームとコマンド・レベルが、IBM MQ Explorer によってサポートされていない。

ローカル、リモート、どちらのキュー・マネージャーでも、クラスター・メンバーとして管理できます。 ただし、リモート・キュー・マネージャーの場合は、接続に TCP/IP を使用する必要があります。 クラスター・メンバーがローカル・キュー・マネージャーの場合、IBM MQ Explorer はクライアント接続によってではなく、直接接続します。

IBM MQ Explorer のデータ変換

IBM MQ Explorer は、CCSID 1208 (UTF-8) で動作します。 これにより IBM MQ Explorer では、リモート・キュー・マネージャーからのデータを正しく表示できます。 キュー・マネージャーに直接接続するか、または中間キュー・マネージャーを使用して接続するかどうかに関係なく、IBM MQ Explorer では、送られてくるメッセージすべてを CCSID 1208 (UTF-8) に変換する必要があります。

IBM MQ Explorer とキュー・マネージャーとの間の接続を確立しようとしたとき、そのキュー・マネージャーの CCSID を IBM MQ Explorer が認識できない場合、エラー・メッセージが発行されます。

サポートされている変換は、コード・ページ変換で説明されています。

Windows IBM MQ Taskbar アプリケーションの使用 (Windows のみ)

IBM MQ Taskbar アプリケーションは、サーバーの Windows システム・トレイにアイコンを表示します。 このアイコンから IBM MQ の現在の状況が分かるとともに、いくつかの単純なアクションを実行できるメニューにアクセスできます。

Windows の場合、IBM MQ アイコンは、サーバー上のシステム・トレイの中にあり、状況を示す色分けされたシンボルにオーバーレイされます。色の意味は次のとおりです。

緑色
正常に作動。現在、アラートは何もありません。

青色
不確定。IBM MQ は現在、始動またはシャットダウン中です。

黄色
アラート。1 つ以上のサービスに障害が発生しています (発生しました)。

メニューを表示するには、IBM MQ アイコンを右クリックします。 メニューから、以下のアクションを実行できます。

- 「**開く**」 をクリックして、 IBM MQ アラート・モニターを開きます。
- 「**終了**」 をクリックして、 IBM MQ Taskbar アプリケーションを終了します。
- **IBM MQ Explorer** をクリックして、 IBM MQ Explorer を開始します。
- IBM MQ を停止するには、 「**停止**」 **IBM MQ** をクリックします。
- IBM MQ アラート・モニターに関する情報を表示するには、 「**バージョン情報**」 **IBM MQ** をクリックします。

Windows IBM MQ アラート・モニター・アプリケーション (Windows のみ)

IBM MQ アラート・モニターは、ローカル・マシン上の IBM MQ に発生した問題を検出して記録するためのエラー検出ツールです。

アラート・モニターでは、IBM MQ サーバーのローカル・インストール・マシンの現在の状況についての情報が表示されます。 また、Windows Advanced Configuration and Power Interface (ACPI) がモニターされ、ACPI 設定が確実に実行されるようにします。

IBM MQ アラート・モニターでは、以下のことが可能です。

- IBM MQ Explorer に直接アクセスする。
- 未解決のすべてのアラートに関する情報を表示する。
- ローカル・マシン上の IBM MQ サービスをシャットダウンする。
- ネットワークを介して、構成可能なユーザー・アカウントや Windows ワークステーション/サーバーにアラート・メッセージを転送する。

ローカル IBM MQ オブジェクトの処理

Message Queue Interface (MQI) を使用するアプリケーション・プログラムをサポートするための、ローカル IBM MQ オブジェクトを管理できます。

このタスクについて

ここでは、ローカル管理とは、IBM MQ オブジェクトを作成、表示、変更、コピー、および削除することを意味しています。

このセクションで説明するアプローチに加えて、IBM MQ Explorer を使用して、ローカル IBM MQ オブジェクトを管理することもできます。 詳しくは、118 ページの『IBM MQ Explorer を使用した管理』を参照してください。

手順

- 以下のトピックにある情報を使用すると、ローカル IBM MQ オブジェクトを管理できます。
  - MQI を使用するアプリケーション・プログラム
  - 12 ページの『MQSC コマンドを使用した IBM MQ の管理』
  - 136 ページの『キュー・マネージャーの属性の表示および変更』
  - 139 ページの『ローカル・キューの処理』
  - 151 ページの『別名キューの処理』
  - 153 ページの『モデル・キューの処理』

- 182 ページの『サービスの取り扱い』
- 190 ページの『トリガー操作のためのオブジェクトの管理』

キュー・マネージャーの処理

制御コマンドを使用して、キュー・マネージャーを開始および停止できます。 MQSC コマンドを使用して、キュー・マネージャー属性を表示または変更できます。

関連タスク

Multiplatforms でのキュー・マネージャーの作成

Multi キュー・マネージャーの開始

キュー・マネージャーを作成する際は、それを開始して、コマンドまたは MQI 呼び出しを処理できるようにしなければなりません。

このタスクについて

strmqm コマンドを使用してキュー・マネージャーを開始できます。 **strmqm** コマンドとそのオプションの詳細については、strmqm を参照してください。

Windows Linux 代わりに、Windows および Linux (x86 および x86-64 プラットフォーム) システムでは、IBM MQ Explorer を使用してキュー・マネージャーを開始することができます。

Windows Windows では、IBM MQ Explorer を使用してシステムが始動したときに、自動的にキュー・マネージャーを始動することができます。 詳しくは、118 ページの『IBM MQ Explorer を使用した管理』を参照してください。

手順

- **strmqm** コマンドを使用してキュー・マネージャーを開始するには、コマンドに続いて、開始するキュー・マネージャーの名前を入力します。

  例えば、QMB という名前のキュー・マネージャーを開始するには、次のコマンドを入力します。

  ```
  strmqm QMB
  ```

  注 : strmqm コマンドは、作業対象のキュー・マネージャーに関連付けられたインストール済み環境から使用する必要があります。 `dspmq -o installation` コマンドを使用して、どのインストール済み環境にキュー・マネージャーが関連付けられているかを調べることができます。

  `strmqm` コマンドは、キュー・マネージャーが開始して、接続要求を受け入れる用意ができるまで、制御を戻しません。

- Windows Linux

  IBM MQ Explorer を使用してキュー・マネージャーを開始するには、以下のステップを完了します。

  a) IBM MQ Explorer を開きます。

  b) ナビゲーター・ビューでキュー・マネージャーを選択します。

  c)「**開始**」をクリックします。

タスクの結果

キュー・マネージャーが開始します。

キュー・マネージャーの開始に数秒より長い時間がかかると、開始の進行状況の詳細を示す情報メッセージが IBM MQ から断続的に発行されます。

Multi キュー・マネージャーの停止

endmqm コマンドを使用して、キュー・マネージャーを停止できます。 このコマンドでは、制御 (静止) シャットダウン、即時シャットダウン、プリエンプティブ・シャットダウン、 および待機シャットダウンという 4 つの方法でキュー・マネージャーを停止できます。 または、Windows および Linux では、IBM MQ Explorer を使用してキュー・マネージャーを停止できます。

このタスクについて

endmqm コマンドで単一インスタンス・キュー・マネージャーを停止する方法は 4 つあります。

制御 (静止) 状態でのシャットダウン
デフォルトでは、**endmqm** コマンドが指定されたキュー・マネージャーの静的シャットダウンを実行します。 静止状態でのシャットダウンは、接続されたアプリケーションすべてが切断されるまで待機するため、完了するまで時間がかかる場合があります。

即時シャットダウン
即時シャットダウンの場合、現在の MQI 呼び出しを完了することはできますが、新しい呼び出しは失敗します。 このタイプのシャットダウンは、アプリケーションがキュー・マネージャーに接続中でも実行されます。

プリエンプティブ・シャットダウン
キュー・マネージャーは即時に停止します。 このタイプのシャットダウンは、例外的な状況でのみ使用します。例えば、キュー・マネージャーが通常の **endmqm** コマンドで停止しない場合などです。

待機シャットダウン
このタイプのシャットダウンは、キュー・マネージャーが停止した後でのみ制御 がユーザーに戻るということを除けば、制御されたシャットダウンと同じです。

endmqm コマンドは、 単一インスタンスのキュー・マネージャーを停止する場合と同じ方法で、 複数インスタンス・キュー・マネージャーのすべてのインスタンスを停止します。 **endmqm** は、 アクティブ・インスタンス、または複数インスタンス・キュー・マネージャーの 1 つのスタンバイ・インスタンスのいずれかで発行できます。 ただし、キュー・マネージャーを終了するには、アクティブ・インスタンスで **endmqm** を実行する必要があります。

必須ではないキュー・マネージャーの保守タスクを中断するかどうかにかかわらず、指定した秒数の目標時間内にキュー・マネージャーを終了するオプションがあります。 131 ページの『ターゲット時間内でのキュー・マネージャーの終了』を参照してください。

重要:

- 持続メッセージは、使用されるシャットダウンのタイプ (IBM MQ プロセスの手動終了を含む) に関係なく持続しますが、非持続メッセージは、どのタイプのシャットダウンでも存続することは保証できません。

  キュー・プロパティー NPMCLASS (HIGH) を指定すると、非永続メッセージが最良の方法で保存されます。 **endmqm -t**、 **endmqm -tp**、 **endmqm -p**、または手動で終了する IBM MQ プロセスを使用すると、 IBM MQ シャットダウンまたは再始動のサイクルで NPMCLASS (HIGH) メッセージが、 **endmqm -w** または **endmqm -i** と比較して存続する可能性が低くなります。

- **-p** オプションと **-tp** オプションを使用する場合は特に、突然のシャットダウン方式を使用した結果として、キュー・マネージャーの終了と再始動の両方にかかる時間が長くなる可能性があります。

  最終手段として IBM MQ プロセスを終了することでキュー・マネージャーを終了した場合は、 キュー・マネージャーの再始動時にキュー・マネージャーの状態の調整が多く必要になる可能性があります。

endmqm コマンドとそのオプションの詳細については、endmqm を参照してください。

ヒント: キュー・マネージャーのシャットダウンにおける問題は、アプリケーションによって頻繁に引き起こされます。 例えば、次のような場合です。

- アプリケーションが MQI 戻りコードを正しく検査しない場合

- アプリケーションが静止の通知を要求しない場合
- アプリケーションが (MQDISC 呼び出しを出して)、キュー・マネージャーからの切断を行わずに終了する場合

キュー・マネージャーの停止中に問題が発生した場合は、Ctrl-C を使用して **endmqm** コマンドを中断してください。 その後、別の **endmqm** コマンドを発行できますが、この場合は、必要なタイプのシャットダウンを指定するパラメーターを付加します。

Windows Linux **endmqm** コマンドを使用する代わりに、on Windows andLinux , を使用して IBM MQ Explorer キュー・マネージャを停止し、制御されたシャットダウンまたは即時シャットダウンを実行することができます。

手順

- **endmqm** コマンドを使用してキュー・マネージャーを停止するには、コマンドに続いて必要に応じてパラメーターを入力し、停止するキュー・マネージャーの名前を入力します。

  注 : **endmqm** コマンドは、作業対象のキュー・マネージャーに関連付けられたインストール済み環境から使用する必要があります。 どのインストール済み環境がキュー・マネージャーと関連付けられているかを調べるには、**dspmq** コマンドを使用します。

  ```
  dspmq -o installation
  ```

  - 制御 (静止) 状態でのシャットダウンを実行するには、以下の例に示すように **endmqm** コマンドを入力します。これにより、QMB というキュー・マネージャーが停止します。

    ```
    endmqm QMB
    ```

    または、以下の例に示すように、**-c** パラメーターを指定して **endmqm** コマンドを入力することは、endmqm QMB コマンドと同等です。

    ```
    endmqm -c QMB
    ```

    どちらの場合も、制御は即時にユーザーに戻り、キュー・マネージャーが停止した時点は通知されません。 すべてのアプリケーションが停止してキュー・マネージャーが終了するまでコマンドを待機してから制御をユーザーに戻す場合は、以下の例に示すように、代わりに **-w** パラメーターを使用します。

    ```
    endmqm -w QMB
    ```

  - 即時シャットダウンを実行するには、以下の例に示すように、**-i** パラメーターを指定して **endmqm** コマンドを入力します。

    ```
    endmqm -i QMB
    ```

  - プリエンプティブ・シャットダウンを実行するには、以下の例に示すように、**-p** パラメーターを指定して **endmqm** コマンドを入力します。

    ```
    endmqm -p QMB
    ```

    重要 : プリエンプティブ・シャットダウンは、接続されているアプリケーションに予測不能な結果を及ぼす可能性があります。 このオプションは、通常の **endmqm** コマンドを使用したキュー・マネージャーを停止する他の試みがすべてした場合を除いて使用しないでください。

    ALW プリエンプティブ・シャットダウンが機能しない場合は、代わりに 132 ページの『手動によるキュー・マネージャーの停止』を試してください。

- 自動クライアント再接続を要求するには、**endmqm** コマンドを入力して -r パラメーターを指定します。 このパラメーターには、クライアントがキュー・マネージャー・グループ内の他のキュー・マネージャーへの接続を再確立する効果があります。

  注 : デフォルトの **endmqm** コマンドを使用してキュー・マネージャーを終了しても、クライアントの自動再接続はトリガーされません。
- 複数インスタンス・キュー・マネージャーのアクティブ・インスタンスをシャットダウンした後にスタンバイ・インスタンスに移行するには、複数インスタンス・キュー・マネージャーのアクティブ・インスタンスで **endmqm** コマンドを入力して -s パラメーターを指定します。
- 複数インスタンス・キュー・マネージャーのスタンバイ・インスタンスを終了して、アクティブ・インスタンスの実行を続けるには、複数インスタンス・キュー・マネージャーのスタンバイ・インスタンスで **endmqm** コマンドを入力して -x パラメーターを指定します。
- Windows Linux

  Windows および Linux で、 IBM MQ Explorer を使用してキュー・マネージャーを停止するには、以下のステップを実行します。

  a) IBM MQ Explorer を開きます。

  b) ナビゲーター・ビューからキュー・マネージャーを選択します。

  c)「**停止**」をクリックします。
  「**キュー・マネージャーの終了**」パネルが表示されます。

  d)「**制御**」または「**即時**」を選択します。

  e)「**OK**」をクリックします。
  キュー・マネージャーが停止します。

関連タスク

AIX での複数インスタンスのキュー・マネージャーへの保守レベル・アップデートの適用
Linux での複数インスタンスのキュー・マネージャーへの保守レベル・アップデートの適用
Windows での複数インスタンスのキュー・マネージャーへの保守レベル・アップデートの適用

関連資料

endmqm (キュー・マネージャーの終了)

ターゲット時間内でのキュー・マネージャーの終了

キュー・マネージャーの保守作業を中断してもしなくても、指定した秒数の目標時間内にキュー・マネージャーを終了することができます。

endmqm コマンドを使用する場合、ターゲット時刻を指定する方法は 2 つあります。 **-t** オプションを使用すると、すべてのキュー・マネージャー保守タスクを完了できます。これにより、キュー・マネージャーの終了フェーズが長くなる可能性があります。 **-tp** オプションは、指定された目標時間に準拠するために必要な場合、不要なキュー・マネージャー保守タスクを中断します。

重要でない保守作業には、キュー・ファイルの圧縮、および NPMCLASS (HIGH) メッセージの永続化が含まれます。 このページの残りの部分では、「ハウスキーピング」という語が使用されます。

アプリケーションの使用パターンによっては、キュー・ファイルの圧縮に長時間かかる可能性があるため、1 次目標がキュー・マネージャーの迅速な終了である場合は、 **-tp** オプションを使用してください。

ターゲット時間を指定するときに、シャットダウン・タイプとして **-w**、**-i**、または **-p** を指定して、開始するシャットダウン・タイプを示します。

注 : immediate のシャットダウンは、実行中のアプリケーションが静止するという点で controlled のシャットダウンとは異なり、依然として正常に行われます。 immediate シャットダウンでもハウスキーピングは実行されます。 時間制限シャットダウンは、これらのアクションが目標時間の達成を妨げている場合に、それらのアクションを終了します。

キュー・マネージャーは、ターゲット時間を達成するために、必要に応じてシャットダウン・タイプを引き上げます。 以下に例を示します。

- **-t** ターゲット 10 秒を **-w** で開始すると、静止に 7 秒、 ハウスキーピングを含むキュー・マネージャーの即時シャットダウンに 2 秒かかった後、それ以上はハウスキーピングを行わずに即時シャットダウンされる可能性があります。

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

- **-tp** ターゲットを 10 秒にすると、静止に 7 秒、ハウスキーピングを含むキュー・マネージャーの即時シャットダウンに 2 秒、ハウスキーピングを含まない即時シャットダウンに 1 秒かかった後、IBM MQ プロセスの終了が開始される可能性があります。

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

- **-i** で **-tp** ターゲットを 2 秒にすると、ハウスキーピングを含むキュー・マネージャーの即時シャットダウンに 1 秒、ハウスキーピングを含まない即時シャットダウンに 1 行かかった後、IBM MQ プロセスの終了が開始される可能性があります。

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

- **-w** の 1 秒のターゲットは、 wait で 0.1 秒にすることができます。例えば、接続されたアプリケーションに IBM MQ 戻りコードを送信するのに十分な長さ、ハウスキーピングを含むキュー・マネージャーの即時シャットダウン (0.9 秒)、ハウスキーピングを行わない即時シャットダウン (immediate shutdown) などです。その後、 IBM MQ プロセスの終了を開始します。

関連資料
endmqm (キュー・マネージャーの終了)

ALW 手動によるキュー・マネージャーの停止

通常の方法でキュー・マネージャーの停止および削除を行えなかった場合には、キュー・マネージャーを手動で停止することができます。

このタスクについて

キュー・マネージャーを停止する標準的な方法は、**endmqm** コマンドを使用する方法です (129 ページの『キュー・マネージャーの停止』を参照)。 通常の方法でキュー・マネージャーを停止できない場合は、手動でキュー・マネージャーを停止することができます。 これを行う方法は、使用しているプラットフォームによって異なります。

手順

- Windows
  Windows でキュー・マネージャーを停止するには、132 ページの『Windows でのキュー・マネージャーの手動停止』を参照してください。
- Linux AIX
  AIX または Linux のキュー・マネージャーを停止するには、134 ページの『AIX and Linux でのキュー・マネージャーの手動停止』を参照してください。

関連タスク
マルチプラットフォームでのキュー・マネージャーの作成と管理
関連資料
endmqm

Windows *Windows* でのキュー・マネージャーの手動停止

Windows で **endmqm** コマンドを使用してキュー・マネージャーを停止できない場合は、実行中のプロセスを終了し、IBM MQ サービスを停止することによって、キュー・マネージャーを手動で停止することができます。

このタスクについて

ヒント : Windows タスク・マネージャーおよび **tasklist** コマンドでは、タスクに関する限定的な情報だけが表示されます。 For more information to help to determine which processes relate to a particular queue manager, consider using a tool such as プロセス・エクスプローラー (`procexp.exe`), which is available for download from the Microsoft website at http://www.microsoft.com.

Windows でキュー・マネージャーを停止するには、以下の手順を実行します。

手順

1. Windows タスク・マネージャーを使用して、実行中のプロセスの名前 (ID) をリストします。
2. Windows タスク・マネージャーまたは **taskkill** コマンドを使用して、次の順序でプロセスを停止します (プロセスが実行中の場合)。

表 7. 実行している場合に停止する *Windows* プロセス

プロセス名	**説明**
AMQZMUC0	重要なプロセス・マネージャー
AMQZXMA0	実行コントローラー
AMQZFUMA	OAM プロセス
AMQZLAA0	LQM エージェント
AMQZLSA0	LQM エージェント
AMQZMUF0	ユーティリティー・マネージャー
AMQZMGR0	プロセス・コントローラー
AMQZMUR0	再開始可能なプロセス・マネージャー
AMQFQPUB	パブリッシュ・サブスクライブ・プロセス
AMQFCXBA	ブローカー・ワーカー・プロセス
AMQRMPPA	プロセス・プール・プロセス
AMQCRSTA	非スレッド化応答側ジョブ・プロセス
AMQCRS6B	LU62 受信側チャネルおよびクライアント接続
AMQRRMFA	リポジトリー・プロセス (クラスターの)
AMQPCSEA	コマンド・サーバー
RUNMQTRM	サーバーのトリガー・モニターを呼び出します。
RUNMQDLQ	送達不能キュー・ハンドラーを呼び出します。
RUNMQCHI	チャネル・イニシエーター・プロセス
RUNMQLSR	チャネル・リスナー・プロセス
AMQXSSVN	共有メモリー・サーバー

3. Windows の「コントロールパネル」で、 **管理ツール** > 「**サービス**」 から IBM MQ サービスを停止します。
4. すべての方法を試行しても、キュー・マネージャーが停止しなかった場合は、システムをリブートします。

Linux AIX *AIX and Linux* でのキュー・マネージャーの手動停止

AIX または Linux で **endmqm** コマンドを使用してキュー・マネージャーを停止できない場合は、実行中のプロセスを終了し、IBM MQ サービスを停止することによって、キュー・マネージャーを手動で停止することができます。

このタスクについて

AIX または Linux 上のキュー・マネージャーを停止するには、以下のステップを実行します。

手動でキュー・マネージャーを停止した場合、FFST が行われることがあり、FDC ファイルが `/var/mqm/errors` に格納されます。 これはキュー・マネージャーに障害が発生したことを意味するわけではないので、注意してください。

この手動による方法で停止した後でも、キュー・マネージャーを通常どおり再始動できるはずです。

手順

1. **ps** コマンドを使用して、まだ実行中のキュー・マネージャー・プログラムのプロセス ID (PID) を見つけます。
   例えば、キュー・マネージャーの名前が QMNAME である場合、次のコマンドを使用します。

   ```
   ps -ef | grep QMNAME
   ```

2. **ps** コマンドを使用してディスカバーされた PID を指定し、**kill** コマンドを使用して、まだ実行中のキュー・マネージャー・プロセスを終了します。
   プロセスを終了するには、 **kill -KILL <pid>** または同等の **kill -9 <pid>** コマンドを使用します。

   毎回そのコマンドを発行して、強制終了したい PID を 1 つずつ処理する必要があります。

   重要 : 9(SIGKILL) 以外のシグナルを使用すると、プロセスはおそらく停止せず、予測不能な結果になります。

   次の順序でプロセスを終了します。

表 *8.* 実行している場合に停止する *AIX and Linux* プロセス

プロセス名	**説明**
amqzmuc0	重要なプロセス・マネージャー
amqzxma0	実行コントローラー
amqzfuma	OAM プロセス
amqzlaa0	LQM エージェント
amqzlsa0	LQM エージェント
amqzmuf0	ユーティリティー・マネージャー
amqzmur0	再開始可能なプロセス・マネージャー
amqzmgr0	プロセス・コントローラー
amqfqpub	パブリッシュ・サブスクライブ・プロセス
amqfcxba	ブローカー・ワーカー・プロセス
amqrmppa	プロセス・プール・プロセス
amqcrsta	非スレッド化応答側ジョブ・プロセス
amqcrs6b	LU62 受信側チャネルおよびクライアント接続

表 8. 実行している場合に停止する *AIX and Linux* プロセス (続き)	
プロセス名	**説明**
amqrrmfa	リポジトリー・プロセス (クラスターの)
amqpcsea	コマンド・サーバー
runmqtrm	サーバーのトリガー・モニターを呼び出します。
runmqdlq	送達不能キュー・ハンドラーを呼び出します。
runmqchi	チャネル・イニシエーター・プロセス
runmqlsr	チャネル・リスナー・プロセス

関連タスク

129 ページの『キュー・マネージャーの停止』

endmqm コマンドを使用して、キュー・マネージャーを停止できます。 このコマンドでは、制御 (静止) シャットダウン、即時シャットダウン、プリエンプティブ・シャットダウン、 および待機シャットダウンという 4 つの方法でキュー・マネージャーを停止できます。 または、Windows および Linux では、IBM MQ Explorer を使用してキュー・マネージャーを停止できます。

Multi キュー・マネージャーの再始動

strmqm コマンドを使用してキュー・マネージャーを再始動できます。Windows および Linux x86-64 システムでは、IBM MQ Explorer からキュー・マネージャーを再始動することもできます。

このタスクについて

strmqm コマンドを使用してキュー・マネージャーを再始動できます。 **strmqm** コマンドとそのオプションの詳細については、strmqm を参照してください。

Windows Linux Windows および Linux x86-64 システムでは、キュー・マネージャーを開始するときと同じ方法で IBM MQ Explorer を使用して、キュー・マネージャーを再始動することができます。

手順

- **strmqm** コマンドを使用してキュー・マネージャーを再始動するには、コマンドに続いて、再始動するキュー・マネージャーの名前を入力します。

  例えば、`strmqm saturn.queue.manager` という名前のキュー・マネージャーを開始するには、次のコマンドを入力します。

  ```
  strmqm saturn.queue.manager
  ```

- Windows Linux

  IBM MQ Explorer を使用してキュー・マネージャーを開始するには、以下のステップを完了します。

  a) IBM MQ Explorer を開きます。

  b) ナビゲーター・ビューでキュー・マネージャーを選択します。

  c)「**開始**」をクリックします。

タスクの結果

キュー・マネージャーが再始動します。

キュー・マネージャーの再始動に数秒より長い時間がかかると、開始の進行状況の詳細を示す情報メッセージが IBM MQ から断続的に発行されます。

キュー・マネージャーの属性の表示および変更

DISPLAY QMGR MQSC コマンドは、キュー・マネージャーのキュー・マネージャー・パラメーターを表示するために使用します。 **ALTER QMGR** MQSC コマンドを使用して、ローカル・キュー・マネージャーのキュー・マネージャー・パラメーターを変更します。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

手順

- **runmqsc** で指定したキュー・マネージャーの属性を表示するには、**DISPLAY QMGR** MQSC コマンドを使用します。

```
DISPLAY QMGR
```

以下の例は、このコマンドの一般的な出力を示しています。

```
DISPLAY QMGR
     1 : DISPLAY QMGR
AMQ8408I: Display Queue Manager details.
   QMNAME(QM1)                             ACCTCONO(DISABLED)
   ACCTINT(1800)                           ACCTMQI(OFF)
   ACCTQ(OFF)                              ACTIVREC(MSG)
   ACTVCONO(DISABLED)                      ACTVTRC(OFF)
   ADVCAP(DISABLED)                        ALTDATE(2022-05-05)
   ALTTIME(14.24.34)                       AMQPCAP(NO)
   AUTHOREV(DISABLED)                      CCSID(437)
   CERTLABL(ibmwebspheremqqm1)             CERTVPOL(ANY)
   CHAD(DISABLED)                          CHADEV(DISABLED)
   CHADEXIT( )                             CHLEV(DISABLED)
   CHLAUTH(ENABLED)                        CLWLDATA( )
   CLWLEXIT( )                             CLWLLEN(100)
   CLWLMRUC(999999999)                     CLWLUSEQ(LOCAL)
   CMDEV(DISABLED)                         CMDLEVEL(930)
   COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)    CONFIGEV(DISABLED)
   CONNAUTH(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
   CRDATE(2020-12-22)                      CRTIME(15.42.49)
   CUSTOM( )                               DEADQ( )
   DEFCLXQ(SCTQ)                           DEFXMITQ( )
   DESCR( )                                DISTL(YES)
   IMGINTVL(60)                            IMGLOGLN(OFF)
   IMGRCOVO(YES)                           IMGRCOVQ(YES)
   IMGSCHED(MANUAL)                        INHIBTEV(DISABLED)
   INITKEY( )                              IPADDRV(IPV4)
   LOCALEV(DISABLED)                       LOGGEREV(DISABLED)
   MARKINT(5000)                           MAXHANDS(256)
   MAXMSGL(4194304)                        MAXPROPL(NOLIMIT)
   MAXPRTY(9)                              MAXUMSGS(10000)
   MONACLS(QMGR)                           MONCHL(OFF)
   MONQ(OFF)                               PARENT( )
   PERFMEV(DISABLED)                       PLATFORM(WINDOWS10)
   PSMODE(ENABLED)                         PSCLUS(ENABLED)
   PSNPMSG(DISCARD)                        PSNPRES(NORMAL)
   PSRTYCNT(5)                             PSSYNCPT(IFPER)
   QMID(QM1_2020-12-22_15.42.49)           REMOTEEV(DISABLED)
   REPOS( )                                REPOSNL( )
   REVDNS(ENABLED)                         ROUTEREC(MSG)
   SCHINIT(QMGR)                           SCMDSERV(QMGR)
   SPLCAP(DISABLED)                        SSLCRLNL( )
   SSLCRYP( )                              SSLEV(DISABLED)
   SSLFIPS(NO)                             KEYRPWD( )
   SSLKEYR(C:\ProgramData\IBM\MQ\qmgrs\QM1\ssl\key)
   SSLRKEYC(32767)                         STATACLS(QMGR)
   STATCHL(OFF)                            STATINT(1800)
   STATMQI(OFF)                            STATQ(OFF)
   STRSTPEV(ENABLED)                       SUITEB(NONE)
   SYNCPT                                  TREELIFE(1800)
   TRIGINT(999999999)                      VERSION(09030000)
   XRCAP(NO)
```

注: SYNCPT は読み取り専用キュー・マネージャー属性です。

ALL パラメーターは、**DISPLAY QMGR** コマンドでのデフォルトです。 このパラメーターによって、すべてのキュー・マネージャー属性が表示されます。 特に、出力では、デフォルト・キュー・マネージャー名、送達不能キューの名前、およびコマンド・キューの名前が示されます。

これらのキューが作成されていることを、次のコマンドを入力して確認することができます。

```
DISPLAY QUEUE (SYSTEM.*)
```

これにより、語幹 SYSTEM.* に一致したキューのリストが表示されます。 括弧は必ず付けてください。

- **runmqsc** コマンドに指定されたキュー・マネージャーの属性を変更するには、変更する属性および値を指定した MQSC コマンド **ALTER QMGR** を使用します。

  例えば、jupiter.queue.manager の属性を変更するには、次のコマンドを使用します。

```
runmqsc jupiter.queue.manager
ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

  ALTER QMGR コマンドにより、使用されている送達不能キューが変更され、禁止イベントが使用可能になります。

  ALTER QMGR コマンドでパラメーターが指定されない場合、それらのパラメーターの既存の値が変更されずに残ります。

関連タスク

Multiplatforms でのキュー・マネージャーの作成

関連資料

キュー・マネージャーの属性

runmqsc (MQSC コマンドの実行)

DISPLAY QMGR

ALTER QMGR

Multi キュー・マネージャーの削除

dltmqm 制御コマンドを使用して、キュー・マネージャーを削除できます。 あるいは、 Windows システムおよび Linux システムでは、 IBM MQ Explorer を使用してキュー・マネージャーを削除することができます。

始める前に

重要:

- キュー・マネージャーを削除すると、それに関連したキューやそのメッセージなどすべてのリソースの他、すべてのオブジェクト定義も削除されるため、十分な注意が必要です。 **dltmqm** 制御コマンドを使用する場合、考えを変えるプロンプトは表示されません。Enter キーを押すと、関連するすべてのリソースが失われます。
- Windows Windows では、キュー・マネージャーを削除すると、そのキュー・マネージャーは自動始動リスト (128 ページの『キュー・マネージャーの開始』を参照) からも除去されます。 コマンドが完了すると、IBM MQ queue manager ending メッセージが表示されます。キュー・マネージャーが削除されたことは通知されません。
- クラスター・キュー・マネージャーを削除しても、クラスターからはキュー・マネージャーは除去されません。 詳しくは、 dltmqm で使用上の注意を参照してください。

このタスクについて

dltmqm 制御コマンドを使用して、キュー・マネージャーを削除できます。 **dltmqm** コマンドとそのオプションの詳細については、dltmqm を参照してください。 信頼できる管理者のみにこのコマンドの使用権限を与えるようにします。 (セキュリティーについては、AIX, Linux, and Windows でのセキュリティーのセットアップを参照してください。)

Windows Linux あるいは、Windows および Linux (x86 および x86-64 プラットフォーム) システムでは、IBM MQ Explorer を使用してキュー・マネージャーを削除することもできます。

手順

- **dltmqm** コマンドを使用してキュー・マネージャーを削除するには、以下のステップを完了します。
  a) キュー・マネージャーを停止させます。
  b) 以下のコマンドを発行します。

  ```
  dltmqm QMB
  ```

  注 : dltmqm コマンドは、作業対象のキュー・マネージャーに関連付けられたインストール済み環境から使用する必要があります。 `dspmq -o installation` コマンドを使用して、どのインストール済み環境にキュー・マネージャーが関連付けられているかを調べることができます。
- Windows Linux
  IBM MQ Explorer を使用してキュー・マネージャーを削除するには、以下のステップを完了します。
  a) IBM MQ Explorer を開きます。
  b) ナビゲーター・ビューでキュー・マネージャーを選択します。
  c) キュー・マネージャーが停止していない場合は、停止させます。
  キュー・マネージャーを停止するには、それを右クリックしてから、「**停止**」をクリックします。
  d) キュー・マネージャーを削除してください。
  キュー・マネージャーを削除するには、それを右クリックしてから、「**削除**」をクリックします。

タスクの結果

キュー・マネージャーが削除されます。

MQI チャネルの停止中

サーバー接続のチャネルに STOP CHANNEL コマンドを発行すると、クライアント接続のチャネルを停止するために使用する方式を選択できます。 つまり、MQGET Wait 呼び出しを発行するクライアント・チャネルを制御でき、チャネルを停止する方法とタイミングを決定できます。

STOP CHANNEL コマンドは以下の 3 つのモードで発行できます。これらは、チャネルを停止する方法を示しています。

静止
現在のメッセージが処理されてからチャネルを停止します。

会話の共有が有効になっていると、IBM MQ MQI client はその停止要求を適切なタイミングで認識するようになります。このタイミングは、ネットワークの速度に依存します。 その後の IBM MQ への呼び出し発行の結果により、クライアント・アプリケーションはその停止要求を認識します。

強制
即時にチャネルを停止します。

終了
即時にチャネルを停止します。 チャネルがプロセスとして実行されている場合は、チャネルのプロセスが終了します。スレッドとして実行されている場合は、スレッドが終了します。

これは段階的なプロセスです。 終了モードが使用される場合、まず静止モード、次に強制モードでサーバー接続チャネルの停止が試行され、さらに必要であれば、終了モードが使用されます。 終了のさ

まざまな段階で、クライアントがさまざまな戻りコードを受け取る場合があります。 プロセスまたはスレッドが終了されると、クライアントは通信エラーを受け取ります。

アプリケーションに戻される戻りコードは、発行される MQI 呼び出し、および発行される STOP CHANNEL コマンドによって異なります。 クライアントは、MQRC_CONNECTION_QUIESCING または MQRC_CONNECTION_BROKEN のどちらかの戻りコードを受け取ります。
MQRC_CONNECTION_QUIESCING を検出したクライアントは、現在のトランザクションを完了させ、終了させようと試みます。 これは MQRC_CONNECTION_BROKEN ではできません。 素早くトランザクションを完了、終了できなかったクライアントは、数秒後に CONNECTION_BROKEN を受信します。
MODE(FORCE) または MODE(TERMINATE) を使った STOP CHANNEL コマンドは、MODE(QUIESCE) を使った場合よりも、CONNECTION_BROKEN になる可能性が高くなります。

関連概念
チャネル

ローカル・キューの処理

この項では、ローカル・キュー、モデル・キュー、および別名キューを管理するために使用できる MQSC コマンドの例をいくつか示します。

これらのコマンドの詳細については、MQSC コマンドを参照してください。

関連資料
キューの命名上の制約
その他のオブジェクトの命名上の制約

DEFINE QLOCAL を使用してローカル・キューを定義する

アプリケーションにとって、ローカル・キュー・マネージャーとは、アプリケーションが接続されているキュー・マネージャーです。 ローカル・キュー・マネージャーによって管理されるキューは、そのキュー・マネージャーに対してローカルであるといいます。 ローカル・キューを作成するには、MQSC コマンド **DEFINE QLOCAL** を使用します。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

ローカル・キューを作成するには、MQSC コマンド **DEFINE QLOCAL** を使用します。 デフォルト・ローカル・キューの定義内に定義されているデフォルトを使用するか、またはデフォルト・ローカル・キューの定義からのキュー特性を修正することもできます。

注 : デフォルト・ローカル・キューは、SYSTEM.DEFAULT.LOCAL.QUEUE という名前で、システムのインストール時に作成されます。

手順

- ローカル・キューを作成するには、以下の例に示すように、**DEFINE QLOCAL** コマンドを入力します。
  この例では、**DEFINE QLOCAL** コマンドは、以下の特性を持つ ORANGE.LOCAL.QUEUE という名前のキューを定義します。
  - 読み取りが可能、書き込みが可能、優先順位に従って操作が行われる。
  - 通常 キュー。つまり、開始キューや伝送キューではなく、トリガー・メッセージを生成しない。
  - キューの最大サイズは、5000 個のメッセージで、最大メッセージ長は、4194304 バイトである。

```
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) +
       DESCR('Queue for messages from other systems') +
       PUT(ENABLED) +
```

```
       GET(ENABLED) +
       NOTRIGGER +
       MSGDLVSQ(PRIORITY) +
       MAXDEPTH(5000) +
       MAXMSGL(4194304) +
       USAGE(NORMAL)
```

注:

1. 例の中で示す属性値は、説明のための値を除いてすべてデフォルト値です。 これらの例は、具体例を示すことを目的としています。 デフォルト値が自分の希望するものであるか、デフォルト値が変更されていないことが確実ならば、これらは省略することができます。 140 ページの『DISPLAY QUEUE を使用してデフォルト・オブジェクトの属性を表示する』も参照してください。
2. **USAGE(NORMAL)** は、このキューが伝送キューではないことを示します。
3. 名前が ORANGE.LOCAL.QUEUE である同じキュー・マネージャーにすでにローカル・キューがあると、このコマンドは失敗します。 既存のキューの定義を上書きする場合には、**REPLACE** 属性を使用してください。また、142 ページの『ALTER QLOCAL または DEFINE QLOCAL を使用してローカル・キュー属性を変更する』も参照してください。

関連資料

DEFINE QLOCAL

DISPLAY QUEUE を使用してデフォルト・オブジェクトの属性を表示する

DISPLAY QUEUE MQSC コマンドを使用すると、 IBM MQ オブジェクトが定義されたときにデフォルト・オブジェクトから取得された属性を表示できます。

始める前に

注: このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

IBM MQ オブジェクトを定義する際に、指定していない属性はデフォルト・オブジェクトから得られます。例えば、ローカル・キューを定義すると、このキューは、定義の中で省略された属性を、SYSTEM.DEFAULT.LOCAL.QUEUE と呼ばれるデフォルト・ローカル・キューから継承します。 **DISPLAY QUEUE** コマンドを使用すると、それらの属性が正確にわかります。

手順

- ローカル・キューのデフォルト・オブジェクトの属性を表示するには、次のコマンドを使用します。

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

DISPLAY コマンドの構文は、対応する **DEFINE** コマンドの構文とは異なっています。 **DISPLAY** コマンドでは、単にキュー・マネージャーを指定しますが、**DEFINE** コマンドでは、キューのタイプ (つまり、QLOCAL、QALIAS、QMODEL、または QREMOTE) を指定する必要があります。

属性を個別に指定すると、属性を選択的に表示できます。 以下に例を示します。

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
MAXDEPTH +
MAXMSGL +
CURDEPTH;
```

このコマンドにより、次のような 3 つの指定の属性が表示されます。

```
AMQ8409: Display Queue details.
QUEUE(ORANGE.LOCAL.QUEUE)        TYPE(QLOCAL)
```

```
CURDEPTH(0)                MAXDEPTH(5000)
MAXMSGL(4194304)
```

CURDEPTH は、現行キューのサイズ、つまりキュー上のメッセージ数です。 これは、表示すると便利な属性です。キュー項目数を監視することによって、キューが満杯にならないようにすることができます。

関連資料
DISPLAY QUEUE
DEFINE キュー

DEFINE QLOCAL を使用してローカル・キュー定義をコピーする

DEFINE QLOCAL MQSC コマンドで **LIKE** 属性を使用して、キュー定義をコピーできます。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

LIKE 属性を指定して **DEFINE** コマンドを使用すると、システムのデフォルト・ローカル・キューの属性ではなく、指定したキューと同じ属性を持つキューを作成できます。 この同じ形式の **DEFINE** コマンドを使用して、キュー定義をコピーし、元のキューの属性を変更したものを 1 つ以上代わりに使用することもできます。

注 :

1. **DEFINE** コマンドの **LIKE** 属性を使用した場合、キューの属性のみをコピーします。 キュー上のメッセージはコピーしません。
2. **LIKE** を指定せずにローカル・キューを定義すると、次のようになります。

```
DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE)
```

手順

- システムのデフォルト・ローカル・キューの属性ではなく、指定したキューと同じ属性を持つキューを作成するには、次の例に示すように、**DEFINE** コマンドを入力します。

  キューの作成時に入力されたのとまったく同じにコピーされるようにキューの名前を入力します。 名前に小文字が含まれている場合には、単一引用符で名前を囲みます。

  この例では、システムのデフォルト・ローカル・キューの属性ではなく、キュー ORANGE.LOCAL.QUEUE と同じ属性を持つキューが作成されます。

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
LIKE (ORANGE.LOCAL.QUEUE)
```

- キュー定義をコピーしつつ、元の属性の 1 つ以上を変更するには、次の例に示すように、**DEFINE** コマンドを入力します。

  このコマンドは、キュー ORANGE.LOCAL.QUEUE の属性をキュー THIRD.QUEUE にコピーしますが、新しいキューの最大メッセージ長は 4194304 バイトではなく 1024 バイトにするように指定しています。

```
DEFINE QLOCAL (THIRD.QUEUE) +
LIKE (ORANGE.LOCAL.QUEUE) +
MAXMSGL(1024);
```

関連資料
DEFINE キュー

ALTER QLOCAL または DEFINE QLOCAL を使用してローカル・キュー属性を変更する

REPLACE 属性を指定して **ALTER QLOCAL** または **DEFINE QLOCAL** MQSC コマンドを使用することにより、2 つの方法でキュー属性を変更できます。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

ALTER コマンドまたは **DEFINE** コマンドの **REPLACE** 属性を使用して、既存の定義を、指定した新しい定義に置き換えることができます。 **ALTER** を使用する場合と **DEFINE** を使用する場合の違いは、**ALTER** で **REPLACE** を使用すると、未指定のパラメーターは変更されませんが、**DEFINE** で **REPLACE** を使用すると、すべてのパラメーターが設定される点です。

手順

- キュー属性を変更するには、次の例に示すように **ALTER** コマンドまたは **DEFINE** コマンドを使用します。

  これらの例では、キュー ORANGE.LOCAL.QUEUE の最大メッセージ長が 10,000 バイトに減らされています。

  - **ALTER** コマンドを使用

    ```
    ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
    ```

    このコマンドにより、1 つの属性、つまり最大メッセージ長の属性は変更されますが、他の属性はすべて変更されません。
  - **REPLACE** オプションを指定した **DEFINE** コマンドを使用する場合は、例えば以下のようにします。

    ```
    DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
    ```

    このコマンドにより、最大メッセージ長だけでなく、他のすべての属性も変更されます。他のすべての属性にはデフォルト値が与えられます。 そのため、例えば、それまで書き込み禁止にされていたキューは、書き込み可能に変更されます。キュー SYSTEM.DEFAULT.LOCAL.QUEUE で指定されているとおり、書き込み可能がデフォルトであるからです。

    既存のキューの最大メッセージ長を短くしても、既存のメッセージは影響を受けません。 ただし、新しいメッセージはこの新しい基準に適合する必要があります。

関連資料
ALTER キュー
ALTER QLOCAL
DEFINE キュー
DEFINE QLOCAL

CLEAR QLOCAL を使用してローカル・キューをクリアする

CLEAR QLOCAL MQSC コマンドを使用して、ローカル・キューを消去できます。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

次の場合には、キューの内容をクリアすることができません。

- 同期点でコミットされていないメッセージで、そのキューに書き込まれているものがある場合
- アプリケーションがそのキューを現在オープンしている場合

このタスクについて

CLEAR QLOCAL コマンドを使用してローカル・キューをクリアする場合は、そのキューの名前がローカル・キュー・マネージャーに定義されている必要があります。

注 : いったん上記のコマンドを発行すると、そのコマンドを取り消すためのプロンプトは表示されません。 Enter キーを押すとき、メッセージが失われます。

手順

ローカル・キューからメッセージをクリアするには、次の例に示すように **CLEAR QLOCAL** を使用します。

この例では、MAGENTA.QUEUE という名前のローカル・キューからすべてのメッセージが削除されます。

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

関連資料
CLEAR QLOCAL

DELETE QLOCAL を使用してローカル・キューを削除する

DELETE QLOCAL MQSC コマンドを使用して、ローカル・キューを削除できます。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

コミットされていないメッセージを含むキューは削除できません。

コミットされたメッセージが 1 つ以上あり、コミットされていないメッセージがまったくないキューは、**PURGE** オプションを指定した場合に限り、削除できます。 その場合は、指定したキューにコミットされたメッセージがあっても削除が実行されるので、それらのメッセージもパージされます。

PURGE の代わりに **NOPURGE** を指定すると、コミットされたメッセージがキューに含まれている場合、そのキューが削除されることはありません。

手順

- ローカル・キューを削除するには、次の例に示すように **DELETE QLOCAL** コマンドを使用します。
  この例では、コミットされたメッセージがキューにない場合に、キュー PINK.QUEUE が削除されます。

  ```
  DELETE QLOCAL (PINK.QUEUE) NOPURGE
  ```

  この例では、コミットされたメッセージがキューにあっても、キュー PINK.QUEUE が削除されます。

  ```
  DELETE QLOCAL (PINK.QUEUE) PURGE
  ```

関連資料
DELETE QLOCAL

サンプル・プログラムを使用してキューをブラウズする

キュー上のメッセージの内容を調べる必要がある場合、IBM MQ では、この目的のためのサンプル・キュー・ブラウザーが用意されています。

このタスクについて

このブラウザーは、ソース形式と実行可能ファイル形式の両方のものが以下の場所に用意されています。*MQ_INSTALLATION_PATH* は、IBM MQ がインストールされているディレクトリーの上位ディレクトリーを表しています。

Windows Windows では、サンプル・キュー・ブラウザーのファイル名およびパスは以下のとおりです。

Source
MQ_INSTALLATION_PATH\tools\c\samples\

実行可能モジュール
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

Linux AIX AIX and Linux では、ファイル名とパスは以下のとおりです。

Source
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

実行可能モジュール
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

手順

- サンプル・プログラムを実行するには、次の例に示すようにコマンドを入力します。
  このサンプル・プログラムには 2 つの入力パラメーターが必要です。メッセージをブラウズするキューの名前と、そのキューを所有するキュー・マネージャーです。 以下に例を示します。

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

タスクの結果

このコマンドの一般的な結果を以下の例に示します。

```
AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 850
  Format : 'MQEVENT '
  Priority : 0  Persistence : 0
  MsgId : X'414D512073617475726E2E71756575650005D30033563DB8'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'saturn.queue.manager                          '
  ** Identity Context
  UserIdentifier : '            '
  AccountingToken :
```

```
   X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '                                        '
  ** Origin Context
  PutApplType    : '7'
  PutApplName    : 'saturn.queue.manager        '
  PutDate  : '19970417'    PutTime  : '15115208'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '104'

****  Message   ****

length - 104 bytes

00000000: 0700 0000 2400 0000 0100 0000 2C00 0000 '....→.......,...'
00000010: 0100 0000 0100 0000 0100 0000 AE08 0000 '................'
00000020: 0100 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030: 0000 0000 3000 0000 7361 7475 726E 2E71 '....0...saturn.q'
00000040: 7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager    '
00000050: 2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060: 2020 2020 2020 2020                     '                '

No more messages
MQCLOSE
MQDISC
```

関連資料

ブラウザー・サンプル・プログラム

大規模キューの使用可能化

IBM MQ は、2 TB より大きいキューをサポートします。

Windows Windows システムでは、特に機能拡張する必要なしに大きいファイルのサポートが有効です。

Linux AIX AIX and Linux システムでは、数ギガバイトまたは数テラバイトのキュー・ファイルを作成する前に、ラージ・ファイル・サポートを明示的に使用可能にする必要があります。 サポート可能にする方法については、オペレーティング・システム資料を参照してください。

一部のユーティリティー (tar など) は、数ギガバイトまたは数テラバイトのファイルを処理できません。 大きいファイルをサポート可能にする前に、オペレーティング・システムの資料で使用するユーティリティーの制限事項について調べてください。

キューに必要なストレージ量の計画については、MQ Performance documents でプラットフォーム別のパフォーマンス・レポートを参照してください。

ローカル・キューおよびモデル・キューの新しい属性を使用して、キュー・ファイルのサイズを制御できます。 詳しくは、145 ページの『IBM MQ キュー・ファイルの変更』 を参照してください。

Multi IBM MQ キュー・ファイルの変更

ローカル・キューおよびモデル・キューの属性を使用して、キュー・ファイルのサイズを制御できます。 2 つのキュー状況属性を使用して、キュー・ファイルの現在のサイズと、現在拡張できる最大サイズ (そのファイルで現在使用中のブロック・サイズに基づくサイズ) を表示できます。

キュー・ファイルの変更に使用する属性

ローカル・キューおよびモデル・キューの属性は以下のとおりです。

MAXFSIZE

キューで使用するキュー・ファイルの最大サイズをメガバイト単位で示します。

この属性の値は、MQSC コマンド IBM MQ Explorer または administrative REST API を使用して設定または表示できます。 この属性の値を IBM MQ Console に表示することもできます。

詳しくは、 MAXFSIZE および 146 ページの『IBM MQ キュー・ファイルのサイズの変更』 を参照してください。

この属性に相当する PCF は、 **MQIA_MAX_Q_FILE_SIZE** です。 Change Queue、Copy Queue、および Create Queue を参照してください。

キュー状況には、以下の 2 つの属性があります。

CURFSIZE
キュー・ファイルの現在のサイズを、最も近いメガバイトに丸めてメガバイト単位で示します。

この属性の値は、MQSC コマンド IBM MQ Explorer または administrative REST API を使用して設定または表示できます。

詳しくは、CURFSIZE を参照してください。

この属性に相当する PCF は、 **MQIA_CUR_Q_FILE_SIZE** です。 Inquire Queue および Inquire Queue (応答)を参照してください。

CURMAXFS
キューで現在使用中のブロック・サイズに基づいて、キュー・ファイルが拡張できる現在の最大サイズを最も近いメガバイトに丸めて示します。

この属性の値は、MQSC コマンド IBM MQ Explorer または administrative REST API を使用して設定または表示できます。

詳しくは、CURMAXFS を参照してください。

この属性に相当する PCF は、 **MQIA_CUR_MAX_FILE_SIZE** です。 Inquire Queue および Inquire Queue (応答)を参照してください。

ブロック・サイズと細分度

キュー・ファイルは、ブロックと呼ばれるセグメントに分割されます。 キュー・ファイルの最大サイズを増やすには、キューのブロック・サイズまたは細分度をキュー・マネージャーで変更する必要があります。

新しく定義されたキューが大きな **MAXFSIZE** 値で作成される場合、キューは適切なブロック・サイズで作成されます。 ただし、 **ALTER QLOCAL** MQSC コマンドなどを使用して既存のキューの **MAXFSIZE** 値を増やした場合は、キュー・マネージャーがキューを再構成するために、キューを空にすることが必要になる場合があります。

詳しくは、 148 ページの『IBM MQ キュー・ファイルで保管できるデータ量の計算』 を参照してください。

重要 : ファイル・システムとオペレーティング・システムによっては、ファイル・システム全体のサイズ、または個々のファイルのサイズに制限がある場合があります。 お客様の企業で使用しているシステムの制限を確認する必要があります。

関連資料
ALTER QUEUES
DISPLAY QUEUE
DISPLAY QSTATUS

Multi *IBM MQ* キュー・ファイルのサイズの変更

キュー・ファイルの最大サイズを増減できます。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。

- ALW の上 AIX, Linux, and Windows、MQSC コマンドを **runmqsc** コマンド・プロンプト。 見る MQSC コマンドを対話的に実行する **runmqsc** そしてテキストファイルから MQSC コマンドを実行する **runmqsc**。
- IBM i の上 IBM i スクリプトファイル内にコマンドのリストを作成し、**STRMQMMQSC** 指示。 見る MQSC コマンドを使用した管理 IBM i。

キュー・ファイルの新しいサイズを設定する前に、 DISPLAY QLOCAL MQSC コマンドを使用して、変更するキュー・ファイルのサイズを確認してください。 例えば、以下のコマンドを発行します。

```
DISPLAY QLOCAL(SYSTEM.DEFAULT.LOCAL.QUEUE) MAXFSIZE
```

以下の出力が返されます。

```
AMQ8409I: Display queue details
  QUEUE(SYSTEM.DEFAULT.LOCAL.QUEUE)        TYPE(QLOCAL)
  MAXFSIZE(DEFAULT)
```

これは、キュー・ファイルの最大サイズがデフォルト値の 2,088,960 MB であることを示しています。

このタスクについて

次の手順は、以下を行う方法を示しています。

- キュー・ファイルを拡張できる最大サイズを減らします。
- キュー・ファイルを拡張できる最大サイズを増やします。

重要 : アプリケーションの作成方法やパフォーマンスへの影響の可能性について考慮することなく、キュー・ファイルのサイズを増やさないように注意してください。 非常に大きなキュー・ファイル内のメッセージにランダムに行うアクセスは、非常に遅くなる可能性があります。

キュー・ファイルの最大サイズをデフォルトより大きくすることを検討している場合は、相関 ID や IBM MQ classes for JMS JM 3.0 または IBM MQ classes for Jakarta Messaging セレクター・ストリングなどのメッセージ・セレクターを使用することに注意する必要があります。 キュー・ファイルが大きくなるほど、先入れ先出し法によるキューへのアクセスが適しています。

個々のキュー・ファイルのデータ量を大量にするのは、循環ロギングを構成したキュー・マネージャー、または個々のキューのメディア・イメージ処理を有効にしていないキュー・マネージャーに限ってください。

キュー・マネージャーの操作に影響を与える可能性があるので、SYSTEM キューのサイズは制限しないでください。

手順

1. 最大キュー・ファイル・サイズを減らします。
   a) 次の MQSC コマンドを発行して、サイズが 500 ギガバイトの SMALLQUEUE というローカル・ファイルを作成します。

   ```
   DEFINE QLOCAL(SMALLQUEUE) MAXFSIZE(512000)
        2 : DEFINE QLOCAL(SMALLQUEUE) MAXFSIZE(512000)
   AMQ8006I: IBM MQ queue created
   ```

   すると、メッセージ AMQ8006I を受け取ります。

   注 : ファイルに既に含まれているデータ量よりも少ない値をキューに構成した場合は、新しいメッセージをキューに書き込むことができません。

   十分なスペースがないキュー・ファイルにアプリケーションがメッセージを書き込もうとすると、アプリケーションは戻りコード MQRC_Q_SPACE_NOT_AVAILABLE を受け取ります。 キューで十分な数のメッセージの破壊読み出しを実行すれば、アプリケーションがキューに新しいメッセージを書き込めるようになります。
2. 最大キュー・ファイル・サイズを増やします。
   a) 以下の MQSC コマンドを発行して、サイズが 5 テラバイトの LARGEQUEUE というローカル・ファイルを作成します。

   ```
   DEFINE QLOCAL(LARGEQUEUE) MAXFSIZE(5242880)
   ```

```
     3 : DEFINE QLOCAL(LARGEQUEUE) MAXFSIZE(5242880)
AMQ8006I: IBM MQ queue created
```

Multi *IBM MQ* キュー・ファイルで保管できるデータ量の計算

キューに保管できるデータ量は、キューが分割された個々のブロックのサイズによって制限されます。 MQSC コマンドを使用して、ブロック・サイズと細分度を確認し、キュー・ファイルのサイズを確認します。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。

- ALW の上 AIX, Linux, and Windows、MQSC コマンドを **runmqsc** コマンド・プロンプト。 見る MQSC コマンドを対話的に実行する **runmqsc** そしてテキストファイルから MQSC コマンドを実行する **runmqsc**。
- IBM i の上 IBM i スクリプトファイル内にコマンドのリストを作成し、**STRMQMMQSC** 指示。 見る MQSC コマンドを使用した管理 IBM i。

手順

- ブロック・サイズと細分度を確認します。

  デフォルトのブロック・サイズは 512 バイトです。 2 テラバイトを超えるキュー・ファイルをサポートするには、キュー・マネージャーでブロック・サイズを増やす必要があります。

  ブロック・サイズは、キューに MAXFSIZE を構成したときに自動的に計算されます。ただし、既にメッセージが含まれているキューには、変更したブロック・サイズを適用できません。 キューが空になると、キュー・マネージャーは、構成された **MAXFSIZE** をサポートするように自動的にブロック・サイズを変更します。

  DISPLAY QSTATUS コマンドに CURMAXFS という新しい属性が追加されました。これにより、新しいブロック・サイズを使用するようにキューが変更されたことを確認できます。

  以下の例では、 **CURMAXFS** 値 4177920 により、キュー・ファイルのサイズが現在約 4 テラバイトに拡大できることが確認されます。 キューに構成されている **MAXFSIZE** の値が **CURMAXFS** の値より大きい場合、キュー・マネージャーは、キュー・ファイルのブロック・サイズを再構成する前に、キューが空になるのを待機しています。

  ```
  DISPLAY QSTATUS(LARGEQUEUE)  CURMAXFS
       2 : DISPLAY QSTATUS(LARGEQUEUE)  CURMAXFS
  AMQ8450I: Display queue status details
    QUEUE(LARGEQUEUE)                       TYPE(QUEUE)
    CURMAXFS(4177920)                       CURDEPTH(100000)
  ```

- キュー・ファイルのサイズを確認します。

  DISPLAY QSTATUS コマンドの CURFSIZE 属性を使用すると、ディスク上のキュー・ファイルの現行サイズをメガバイト単位で表示できます。 これは、ファイル・システムに直接アクセスできない IBM MQ Appliance などのプラットフォームで役立ちます。

  ```
  DISPLAY QSTATUS(SMALLQUEUE)  CURFSIZE
       1 : DISPLAY QSTATUS(SMALLQUEUE)  CURFSIZE
  AMQ8450I: Display queue status details
    QUEUE(SMALLQUEUE)                       TYPE(QUEUE)
    CURDEPTH(4024)                          CURFSIZE(10)
  ```

  注 : キューからメッセージが除去されても、 **CURFSIZE** 属性はすぐには減少しません。

  通常、キュー・ファイル内のスペースは、キューをオープンしているアプリケーションがなく、キューに未確定メッセージが保管されていない場合にのみ解放されます。 キュー・マネージャーによってロー

ドされるキュー・ファイルの必要な切り捨てまたは圧縮は、 チェックポイント、キュー・マネージャーのシャットダウン、またはキューのメディア・イメージの記録中に行われます。

関連資料
ALTER QUEUES
DISPLAY QSTATUS

リモート・キューの処理

リモート・キューは、リモート・キュー・マネージャー上のキューを参照するローカル・キュー・マネージャー上の定義です。 ローカル位置からリモート・キューを定義する必要はありませんが、定義すると、アプリケーションは、リモート・キューが置かれているキュー・マネージャーの ID で修飾された名前を指定する代わりに、ローカルで定義された名前でリモート・キューを参照することができます。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

アプリケーションは、ローカル・キュー・マネージャーに接続し、その後 MQOPEN 呼び出しを出します。 オープン呼び出しで指定されるキュー名は、ローカル・キュー・マネージャー上の リモート・キュー定義のキュー名です。 リモート・キュー定義は、ターゲット・キューの名前、ターゲット・キュー・マネージャーの名前、および任意で伝送キューの名前を提供します。 リモート・キューにメッセージを書き込むためには、アプリケーションは、MQPUT 呼び出しから戻されたハンドルを指定して、MQPUT 呼び出しを出します。 キュー・マネージャーは、リモート・キュー名およびリモート・キュー・マネージャー名を、メッセージの先頭につく伝送ヘッダーで使用します。 この情報は、ネットワーク内の正しい宛先にメッセージを転送するために使用されます。

管理者は、リモート・キュー定義を変更することにより、メッセージの宛先を制御できます。

手順

- リモート・キュー・マネージャーが所有するキューにメッセージを書き込みます。

  アプリケーションは、キュー・マネージャー (`saturn.queue.manager` など) に接続します。 ターゲット・キューは、別のキュー・マネージャーが所有しています。

  MQOPEN 呼び出しで、アプリケーションは次のフィールドを指定します。

フィールド値	説明
ObjectName CYAN.REMOTE.QUEUE	リモート・キュー・オブジェクトのローカル名を指定します。 これはターゲット・キューおよびターゲット・キュー・マネージャーを定義します。
ObjectType (Queue)	このオブジェクトをキューと識別します。
ObjectQmgrName ブランクまたは `saturn.queue.manager`	このフィールドの指定はオプションです。 ブランクの場合、ローカル・キュー・マネージャーの名前と見なされます。 (これは、リモート・キュー定義が存在するキュー・マネージャーです。)

  この後、アプリケーションはこのキューにメッセージを書き込むために、MQPUT 呼び出しを出します。

ローカル・キュー・マネージャーでは、次の MQSC コマンドを使用してリモート・キューのローカル定義を作成することができます。

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
DESCR ('Queue for auto insurance requests from the branches') +
RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
RQMNAME (jupiter.queue.manager) +
XMITQ (INQUOTE.XMIT.QUEUE)
```

ここで、

QREMOTE (CYAN.REMOTE.QUEUE)
リモート・キュー・オブジェクトのローカル名を指定します。 これは、このキュー・マネージャーに接続されたアプリケーションが、リモート・キュー・マネージャー jupiter.queue.manager 上のキュー AUTOMOBILE.INSURANCE.QUOTE.QUEUE をオープンするために、MQOPEN 呼び出しに指定する必要のある名前です。

DESCR ('Queue for auto insurance requests from the branches')
キューの用途を説明する追加テキストを提供します。

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
リモート・キュー・マネージャーのターゲット・キューの名前を指定します。 これは、アプリケーションが送信する、キュー名 CYAN.REMOTE.QUEUE を指定したメッセージの実際のターゲット・キューです。 キュー AUTOMOBILE.INSURANCE.QUOTE.QUEUE を、ローカル・キューとして リモート・キュー・マネージャーに定義する必要があります。

RQMNAME (jupiter.queue.manager)
ターゲット・キュー AUTOMOBILE.INSURANCE.QUOTE.QUEUE を所有するリモート・キュー・マネージャーの名前を指定します。

XMITQ (INQUOTE.XMIT.QUEUE)
伝送キューの名前を指定します。 この指定はオプションです。伝送キューの名前を指定しないと、リモート・キュー・マネージャーと同じ名前のキューが使用されます。

いずれの場合でも、伝送キューであることを指定する **Usage** 属性 (MQSC コマンドに USAGE(XMITQ) を指定) を持つローカル・キューとして、該当する伝送キューを定義する必要があります。

- リモート・キューにメッセージを書き込む (代替方法)。

  リモート・キューのローカル定義を使用する方法以外にも、リモート・キューにメッセージを書き込む方法があります。 アプリケーションは、リモート・キュー・マネージャー名を含んでいる完全なキュー名を、MQOPEN 呼び出しの一部として指定することができます。 この場合、リモート・キューのローカル定義は不要です。 ただし、この場合、アプリケーションがリモート・キュー・マネージャーの名前を認識しているか、あるいは実行時にリモート・キュー・マネージャーの名前にアクセスできなければなりません。

- リモート・キューを使用する他のコマンドを使用します。

  MQSC コマンドを使用すると、リモート・キュー・オブジェクトの属性を表示または変更したり、リモート・キュー・オブジェクトを削除したりできます。 以下に例を示します。

  – リモート・キューの属性を表示する。

    ```
    DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
    ```

  – 書き込みを有効にするためにリモート・キューを変更する。 これは、ターゲット・キューには影響を与えません。このリモート・キューを指定するアプリケーションのみに影響を与えます。

    ```
    ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
    ```

  – このリモート・キューを削除する。 これは、ターゲット・キューに影響を与えません。そのローカル定義にのみ影響を与えます。

```
DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

注: リモート・キューを削除する場合、削除するのはリモート・キューのローカル表示のみです。 リモート・キューやリモート・キュー上のメッセージは削除されません。

リモート・キュー定義は別名として使用できます。

キューを別のキュー・マネージャーに置くだけでなく、リモート・キューのローカル定義をキュー・マネージャー別名および応答先キュー別名に使用することもできます。 いずれのタイプの別名も、リモート・キューのローカル定義を使用して解決されます。 その宛先に到着するようにメッセージの適切なチャネルをセットアップしなければなりません。

キュー・マネージャー別名

別名とは、ターゲット・キュー・マネージャーの名前 (メッセージ内に指定されている) をメッセージ経路上のキュー・マネージャーによって変更するためのプロセスです。 キュー・マネージャー別名は重要です。キュー・マネージャーのネットワーク内でメッセージの宛先を制御するのに、この別名を使用できるためです。

別名を実行するには、制御点でキュー・マネージャーのリモート・キュー定義を変更します。 送信アプリケーションは、指定されたキュー・マネージャー名が別名であることを認識しません。

キュー・マネージャーの別名について詳しくは、 別名とは? を参照してください。

応答先キュー別名

オプションとして、アプリケーションは、要求メッセージ をキューに入れる際に、応答先キューの名前を指定することができます。

メッセージを処理するアプリケーションは、その応答先キューの名前を取り出すときに、必要に応じて応答メッセージ の送り先を確認します。

応答先キュー別名とは、応答先キューの名前 (要求メッセージ内で指定されている) をメッセージ経路上のキュー・マネージャーによって変更するためのプロセスです。 送信アプリケーションは、指定された応答先キュー名が別名であることを認識しません。

応答先キュー別名を使用すると、応答先キューの名前を変更でき、オプションでそのキュー・マネージャーを変更することもできます。 これによって、応答メッセージに使用される経路を制御することができます。

要求メッセージ、応答メッセージ、および応答先キューについて詳しくは、メッセージのタイプおよび応答先キューおよびキュー・マネージャーを参照してください。

応答先キュー別名について詳しくは、応答先キューの別名およびクラスターを参照してください。

別名キューの処理

別名キューを定義して、他のキューまたはトピックを間接的に参照できます。

始める前に

注: このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

重要: 配布リストでは、トピック・オブジェクトを指す別名キューの使用はサポートされていません。 別名キューが配布リスト内のトピック・オブジェクトを指し示す場合、IBM MQ は MQRC_ALIAS_BASE_Q_TYPE_ERROR を戻します。

このタスクについて

別名キューが参照するキューは、以下のいずれかが可能です。

- ローカル・キュー (139 ページの『DEFINE QLOCAL を使用してローカル・キューを定義する』を参照)。
- リモート・キューのローカル定義 (149 ページの『リモート・キューの処理』を参照)。
- トピック。

別名キューは、実際のキューではなく、実際の (または宛先) キューに解決される定義です。 別名キュー定義は、ターゲット・キューを指定します。 アプリケーションが別名キューに MQOPEN 呼び出しを行うとき、キュー・マネージャーは、別名をターゲット・キュー名に解決します。

別名キューは、ローカルで定義された別の別名キューに解決できません。 ただし、別名キューは、ローカル・キュー・マネージャーがメンバーであるクラスター内の他の場所に定義された別名キューには解決できます。 詳しくは、ネーム・レゾリューション を参照してください。

別名キューは、以下の場合に役立ちます。

- ターゲット・キューへの異なるレベルのアクセス権限を異なるアプリケーションに与えている場合。
- 異なるアプリケーションが異なる方法で同じキューを処理することを許可している場合。 (異なるデフォルトの優先順位または異なるデフォルトの持続性の値を割り当てる場合など。)
- これは、保守、移行、およびワークロード・バランシングを単純化する場合。 (アプリケーションを変更しないで、別名を使用し続けたままターゲット・キュー名を変更する場合など。)

例えば、MY.ALIAS.QUEUE という名前のキューにメッセージを入れるようなアプリケーションが開発されたとします。 このアプリケーションは、MQOPEN 要求を出すときにこのキューの名前を指定し、メッセージをこのキューに書き込む場合には、間接的にこのキューの名前を指定します。 アプリケーションは、キューが別名キューであることを認識しません。 この別名を使用した各 MQI 呼び出しについて、キュー・マネージャーは実際のキュー名に解決します。このキュー名はローカル・キューか、このキュー・マネージャーに定義されたリモート・キューのいずれかです。

TARGET 属性の値を変更することにより、MQI 呼び出しを別のキュー (おそらく別のキュー・マネージャー上) にリダイレクトすることができます。 これは、保守、移行、および負荷平衡に役立ちます。

手順

- 別名キューを定義します。

  次の MQSC コマンドは、別名キューを作成します。

  ```
  DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
  ```

  このコマンドは、MQI 呼び出し (MY.ALIAS.QUEUE を指定している) をキュー YELLOW.QUEUE にリダイレクトします。 このコマンドは、ターゲット・キューを作成しないので、キュー YELLOW.QUEUE が実行時に存在しなければ、MQI 呼び出しは失敗します。

  別名定義を変更すると、MQI 呼び出しを別のキューにリダイレクトできます。 以下に例を示します。

  ```
  ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
  ```

  このコマンドは、MQI 呼び出しを別のキュー MAGENTA.QUEUE にリダイレクトします。

  別名キューを使用すると、単一のキュー (ターゲット・キュー) が、異なるアプリケーションについては異なる属性を持っているよう見えるようにすることもできます。 これは、アプリケーションごとに 1 つの別名、つまり合計 2 つの別名を定義すると行えます。 2 つのアプリケーションがあるとします。

  - アプリケーション ALPHA は、メッセージを YELLOW.QUEUE に書き込むことができますが、そこからメッセージを読み取ることはできません。
  - アプリケーション BETA は、YELLOW.QUEUE からメッセージを読み取ることはできますが、そこにメッセージを書き込むことはできません。

  以下の MQSC コマンドは、アプリケーション ALPHA の書き込み可能および書き込み不可の別名を定義します。

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
TARGET (YELLOW.QUEUE) +
PUT (ENABLED) +
GET (DISABLED)
```

以下のコマンドは、アプリケーション BETA の PUT を使用不可にし、GET を使用可能にする別名を定義します。

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
TARGET (YELLOW.QUEUE) +
PUT (DISABLED) +
GET (ENABLED)
```

ALPHA は、MQI 呼び出しの中でキュー名 ALPHAS.ALIAS.QUEUE を使用しますが、BETA は、キュー名 BETAS.ALIAS.QUEUE を使用します。 これらはいずれも同じキューをアクセスしますが、その方法は異なっています。

キュー別名を定義する際には、ローカル・キューの場合と同様にして、LIKE 属性および REPLACE 属性を使用することができます。

- 別名キューを使用する他のコマンドを使用します。

  該当する MQSC コマンドを使用すると、キュー別名の属性の表示、変更、あるいはキュー別名オブジェクトの削除ができます。 以下に例を示します。

  DISPLAY QALIAS コマンドを使用して、別名キューの属性を表示します。

  ```
  DISPLAY QALIAS (ALPHAS.ALIAS.QUEUE)
  ```

  ALTER QALIAS コマンドを使用して、基本キュー名を変更します。別名は解決されます。キューがオープンしている場合も、`force` オプションを使用すると、強制的に変更が実施されます。

  ```
  ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGET(ORANGE.LOCAL.QUEUE) FORCE
  ```

  DELETE QALIAS コマンドを使用して、このキューの別名を削除します。

  ```
  DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
  ```

  アプリケーションがそのキューを現在オープンしている場合、別名キューを削除することはできません。

関連概念
配布リスト
関連資料
ALTER QALIAS
DEFINE QALIAS
DELETE QALIAS

モデル・キューの処理

モデル・キューは、アプリケーションがキューを必要とするときにそのキューを作成するための便利な方法を提供します。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

キュー・マネージャーは、モデル・キューとして定義されているキュー名を指定した MQI 呼び出しをアプリケーションから受け取ると、動的キュー を作成します。 新しい動的キューの名前は、そのキューの作成時にキュー・マネージャーによって生成されます。 モデル・キュー とは、動的キューの属性を指定しているテンプレートのことで、動的キューはこのモデル・キューから作成されます。

手順

- モデル・キューを定義します。

  DEFINE QMODEL MQSC コマンドを使用して、ローカル・キューを定義するのと同じ方法で、一連の属性を持つモデル・キューを定義します。 作成される動的キューが一時キューとなるか永続キューとなるかをモデル・キューには指定できるが、ローカル・キューにはできないこと以外は、モデル・キューとローカル・キューは同じ一連の属性を持っています。 (永続キューはキュー・マネージャーが再始動しても維持されますが、一時キューは維持されません。) 以下に例を示します。

  ```
  DEFINE QMODEL (GREEN.MODEL.QUEUE) +
  DESCR('Queue for messages from application X') +
  PUT (DISABLED) +
  GET (ENABLED) +
  NOTRIGGER +
  MSGDLVSQ (FIFO) +
  MAXDEPTH (1000) +
  MAXMSGL (2000) +
  USAGE (NORMAL) +
  DEFTYPE (PERMDYN)
  ```

  このコマンドにより、モデル・キュー定義が作成されます。 **DEFTYPE** 属性により、このテンプレートから作成される実際のキューは、永続動的キューになります。 指定されていない属性は、SYSYTEM.DEFAULT.MODEL.QUEUE デフォルト・キューから自動的にコピーされます。

  モデル・キューを定義する際には、ローカル・キューの場合と同様にして、**LIKE** 属性および **REPLACE** 属性を使用することができます。

- モデル・キューで他のコマンドを使用します。

  該当する MQSC コマンドを使用すると、モデル・キューの属性を表示または変更したり、モデル・キュー・オブジェクトを削除したりできます。 以下に例を示します。

  DISPLAY QUEUE コマンドを使用して、モデル・キューの属性を表示します。

  ```
  DISPLAY QUEUE (GREEN.MODEL.QUEUE)
  ```

  ALTER QMODEL コマンドを使用して、このモデルから作成された動的キューに書き込みができるようにモデルを変更します。

  ```
  ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
  ```

  DELETE QMODEL コマンドを使用して、このモデル・キューを削除します。

  ```
  DELETE QMODEL (RED.MODEL.QUEUE)
  ```

関連資料

ALTER QMODEL
DEFINE QMODEL
DELETE QMODEL
DISPLAY QUEUE

送達不能キューの取り扱い

正しい宛先に送達できないメッセージを後で取り出すために保管することができるよう、各キュー・マネージャーには通常、送達不能キューとして使用するローカル・キューがあります。 送達不能キューについてキュー・マネージャーに通知し、送達不能キューで見つかったメッセージの処理方法を指定するようにします。 送達不能キューを使用すると、メッセージが送達される順序に影響するので、これらを使用しなくてもかまいません。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

SYSTEM.DEAD.LETTER.QUEUE という名前のサンプル送達不能キューがプロダクトで使用可能です。 このキューは、キュー・マネージャーを作成すると、自動的に作成されます。 必要ならば、この定義を修正し、名前変更することができます。

送達不能キューには、以下に示すものを除いて、特別な要件はありません。

- ローカル・キューでなければならない。
- その MAXMSGL (最大メッセージ長) 属性は、キュー・マネージャーが取り扱う最大メッセージ**および**送達不能ヘッダー (MQDLH) のサイズをキューに収容できるようにしておく必要があります。

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

手順

- 送達不能キューについてキュー・マネージャーに通知する。

  これを行うには、 **crtmqm** コマンドで送達不能キュー名を指定するか (例えば、`crtmqm -u DEAD.LETTER.QUEUE`)、 **ALTER QMGR** コマンドで **DEADQ** 属性を使用して後で指定します。 送達不能キューを使用するためには、その前にそれを定義しておくことも必要です。
- 送達不能キューで検出されたメッセージの処理方法を指定します。

  USEDLQ チャネル属性を設定して、メッセージが配信できない場合に送達不能キューを使用するかどうかを決定します。 キュー・マネージャーのいくつかの機能が送達不能キューを使用する一方で、他の機能がそれを使用しないように、この属性を構成できます。 さまざまな MQSC コマンドでの USEDLQ チャネル属性の使用について詳しくは、DEFINE CHANNEL、DISPLAY CHANNEL、ALTER CHANNEL、および DISPLAY CLUSQMGR を参照してください。

  IBM MQ 送達不能キュー・ハンドラーを使用して、送達不能キューで検出されたメッセージを処理または除去する方法を指定します。 155 ページの『IBM MQ 送達不能キューのメッセージの処理』を参照してください。

関連概念

送達不能キュー

関連タスク

未配布メッセージのトラブルシューティング

関連資料

ALTER QMGR

crtmqm (キュー・マネージャーの作成)

IBM MQ 送達不能キューのメッセージの処理

送達不能キュー (DLQ) 上のメッセージを処理するには、 IBM MQ に用意されているデフォルトの DLQ ハンドラーを使用します。 このハンドラーは、DLQ のメッセージと、定義した規則テーブル内の項目を突き合わせます。

このタスクについて

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、常にこのヘッダーがあります。メッセージを DLQ に書き込むアプリケーションはこのヘッダー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ります。

すべての IBM 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。 IBM MQ には、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・ルーチンが用意されています。このルーチンは、 **runmqdlq** MQSC 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。

関連タスク

未配布メッセージのトラブルシューティング

関連資料

送達不能キュー

z/OS ALW 送達不能キュー・ハンドラーの呼び出し

runmqdlq 制御コマンドを使用して、送達不能キュー (DLQ) ハンドラーを呼び出します。 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。

始める前に

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。

このタスクについて

以下の例は、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、その入力を stdin から受け取り ます。 ルール・テーブルから stdin をリダイレクトすることにより、ルール・テーブルを **runmqdlq** に関連付けます。

runmqdlq コマンドの詳細については、runmqdlq を参照してください。

手順

- DLQ およびキュー・マネージャーは、 **runmqdlq** コマンドのパラメーターとして指定できます。

  例えば、コマンド・プロンプトから次のようにします。

  ```
  runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER <qrule.rul
  ```

- 規則テーブルで DLQ とキュー・マネージャーに名前を付けることができます。

以下に例を示します。

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

関連概念

送達不能キュー

関連タスク

未配布メッセージのトラブルシューティング

z/OS ALW サンプル *DLQ* ハンドラー ***amqsdlq***

IBM MQ には、 **runmqdlq** コマンドを使用して呼び出される送達不能キュー・ハンドラーに加えて、サンプル DLQ ハンドラー **amqsdlq** のソースとして、 **runmqdlq** に用意されている機能に類似した機能が用意されています。

amqsdlq をカスタマイズして、要件を満たす DLQ ハンドラーを提供することができます。 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。 (デフォルトの DLQ ハンドラーとサンプルの **amqsdlq** は両方とも、送達不能ヘッダー MQDLH で始まる DLQ 上のメッセージのみを処理します。 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、IBM 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。

IBM MQ for Windows では、amqsdlq のソースは、次のディレクトリー内にあります。

```
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
```

IBM MQ for UNIX および Linux システムでは、 **amqsdlq** のソースは以下のディレクトリーで提供されます。

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

amqsdlqc という名前のサンプル・プログラムのビルド・バージョンが含まれています。 これを使用して、リモート・キュー・マネージャーにクライアント・モードで接続することができます。 **amqsdlqc** を利用するには、環境変数 *MQSERVER*、**MQCHLLIB**、**MQCHLTAB** のうちの 1 つを設定して、キュー・マネージャーに接続する方法を指定する必要があります。 以下に例を示します。

```
export MQSERVER="SYSTEM.DEF.SVRCONN/TCP/myappliance.co.uk(1414)"
```

z/OS ALW *DLQ* ハンドラーの規則テーブル

送達不能キュー・ハンドラー規則テーブルでは、DLQ に入ってくるメッセージを DLQ ハンドラーで処理する方法を定義します。

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。

関連概念
送達不能キュー
関連タスク
未配布メッセージのトラブルシューティング


DLQ 制御データ
送達不能キュー・ハンドラー規則テーブルの制御データ項目にキーワードを組み込むことができます。

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*| '' (デフォルト)
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* 名前| '' (デフォルト)
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 (デフォルト))
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 (デフォルト) |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)、アクティビティーのレベルが低い 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 IBM 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。

DLQ 規則 *(*パターンとアクション*)*

パターン・マッチング・キーワード (送達不能キュー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|* (デフォルト))
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 (*PutApplName*|* (デフォルト))
DLQ 内にあるメッセージのメッセージ記述子 MQMD の *PutApplName* フィールドで指定した、 MQPUT または MQPUT1 呼び出しの実行元アプリケーションの名前。

APPLTYPE (*PutAppl* タイプ|* (デフォルト))
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*|* (デフォルト))
メッセージの送り先のメッセージ・キューの名前。

DESTQM (*QueueManagerName*|* (デフォルト))
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|* (デフォルト))
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|* (デフォルト))
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*|* (デフォルト))
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|* (デフォルト))
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*|* (デフォルト))
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*|* (デフォルト))
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 (*QueueManagerName*|* (デフォルト))
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*|* (デフォルト))
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。

RETRY
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 (キュー名|&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) を指定するルールが、ブランクの 応答キュー フィールドを持つメッセージと一致すると、エラー・メッセージが出されないようにするには、メッセージ・パターンに REPLYQ (?*) を指定します。

FWDQM (*QueueManager* 名| & DESTQM | & REPLYQM | '' (デフォルト)
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 (デフォルト) |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 (デフォルト) | 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 (デフォルト))
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。

DLQ 規則テーブルの規則

送達不能キュー・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。
- 行の最後の非ブランク文字として正符号 (+) を使用した場合、その行の規則が次の行の最初の非ブランク文字に続くことを表します。 行の最後の非ブランク文字として負符号 (-) を使用した場合、その行の規則が次の行の先頭に続くことを表します。 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、数値パラメーターの一部としてアスタリスク (*) が含まれているため、MSGTYPE('2*') は無効です。

- 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。

z/OS ALW *DLQ* ルール・テーブルの処理方法

送達不能キュー・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。

関連概念

送達不能キュー

関連タスク

未配布メッセージのトラブルシューティング

z/OS ALW *DLQ* ハンドラー規則テーブルの例

runmqdlq コマンドの送達不能キュ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が入っています。

```
*********************************************************************
*    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)
```

関連概念

送達不能キュー

関連タスク

未配布メッセージのトラブルシューティング

関連資料

runmqdlq (送達不能キュー・ハンドラーの実行)

IBM i *IBM i での送達不能キュー・ハンドラーの呼び出し*

IBM MQ for IBM i では、 **STRMQMDLQ** コマンドを設定して DLQ ハンドラーを呼び出します。

始める前に

DLQ ハンドラーを実行するためには、DLQ 自体、および DLQ のメッセージが転送されるメッセージ・キューにアクセスする許可を持っている必要があります。 DLQ がメッセージ・コンテキストのユーザー ID の権限を使用してキューにメッセージを書き込むようにするためには、他のユーザーの ID を使用する許可も必要です。

注 : DLQ へのメッセージ書き込みを避けることが望ましい場合がよくあります。 DLQ の使用および回避について詳しくは、155 ページの『送達不能キューの取り扱い』を参照してください。

このタスクについて

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、およびアプリケーションは DLQ にメッセージを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・エージェントによって DLQ に書き込まれたメッセージには、必ず MQDLH が付いています。 DLQ にメッセージを書き込むアプリケーションの場合は、MQDLH を指定する必要があります。

MQDLH 構造体の *Reason* フィールドには、メッセージが DLQ 上にある理由を識別する理由コードが入ります。

すべての IBM MQ 環境には、DLQ 上のメッセージを処理するために定期的に実行されるルーチンが必要です。 IBM MQ には、 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・ルーチンが用意されています。DLQ ハンドラーは、 **STRMQMDLQ** コマンドを使用して呼び出します。 ユーザー作成の規則表 から、DLQ 内のメッセージを処理するための命令を DLQ ハンドラーに与えるようにします。 つまり、DLQ ハンドラーは、DLQ 上のメッセージと、規則テーブルの項目を突き合わせます。 DLQ メッセージが規則テーブルの項目に一致すると、DLQ ハンドラーが、その項目に関連付けられているアクションを実行します。

手順

- DLQ ハンドラーの呼び出し

  STRMQMDLQ コマンドを使用して、DLQ ハンドラーを呼び出します。 処理したい DLQ と、使用したいキュー・マネージャーを、次の 2 つの方法で指定できます。

– コマンド・プロンプトから **STRMQMDLQ** へのパラメーターとして。 以下に例を示します。

```
STRMQMDLQ UDLMSGQ(ABC1.DEAD.LETTER.QUEUE) SRCMBR(QRULE) SRCFILE(library/QTXTSRC)
MQMNAME(MY.QUEUE.MANAGER)
```

– 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)
```

注 : 規則テーブルは、ソース物理ファイル内のメンバーで、任意の名前を付けられます。

これらの例は、デフォルトのキュー・マネージャーが所有する 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。

関連概念

送達不能キュー

関連タスク

未配布メッセージのトラブルシューティング

IBM i *IBM i での DLQ ハンドラーの規則テーブル*

送達不能キュー・ハンドラー規則テーブルでは、IBM i DLQ に入ってくるメッセージを DLQ ハンドラーで処理する方法を定義します。

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。 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。
- 指定できる値は、縦線 (|) で区分されています。 いずれか 1 つを指定できます。
- キーワードはすべてオプションです。

INPUTQ (*QueueName*| '' (デフォルト)
処理対象の DLQ の名前です。

1. **STRMQMDLQ** コマンドのパラメーターとして UDLMSGQ 値 (または *DFT) を指定すると、規則テーブルの INPUTQ 値がすべて指定変更されます。
2. **STRMQMDLQ** コマンドのパラメーターとしてブランクの UDLMSGQ 値を指定すると、規則テーブルの INPUTQ 値が使用されます。
3. **STRMQMDLQ** コマンドのパラメーターとしてブランクの UDLMSGQ 値、および規則テーブルにブランクの INPUTQ 値を指定すると、システムのデフォルト送達不能キューが使用されます。

INPUTQM (*QueueManager* 名前| '' (デフォルト)
INPUTQ キーワードに指定された DLQ を所有するキュー・マネージャーの名前です。

キュー・マネージャーを指定していない場合、または規則テーブルに INPUTQM(' ') を指定した場合は、インストール済み環境のデフォルト・キュー・マネージャーがシステムで使用されます。

RETRYINT (間隔| 60 (デフォルト))
DLQ ハンドラーが、最初の試行で処理されなかった DLQ のメッセージの再処理を試みる秒単位の間隔であり、その間、試行が繰り返し要求されます。 デフォルトでは、再試行間隔は 60 秒です。

WAIT (YES (デフォルト) |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 ハンドラーはいつまでも待機します。

NO
DLQ ハンドラーは、DLQ が空になったか、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
DLQ ハンドラーは、キューが空になったか、または処理できるメッセージがなくなったことを検出した後、新しいメッセージの着信を *nnn* 秒だけ待ってから終了します。

使用頻度の高い DLQ については WAIT (YES)、アクティビティーのレベルが低い DLQ については WAIT (NO) または WAIT (*nnn*) を指定します。 DLQ ハンドラーの終了が可能な場合は、トリガー操作を使用して再起動します。

規則テーブルに制御データを組み込む代わりに、**STRMQMDLQ** コマンドへの入力パラメーターとして DLQ の名前を指定できます。 規則テーブル、および **STRMQMDLQ** コマンドへの入力の両方に値が指定されている場合、**STRMQMDLQ** コマンドに指定された値が優先されます。

注 : 規則表に制御データ項目を含める場合は、必ず表の先頭に入れてください。

IBM i *DLQ 規則 (パターンとアクション) (IBM i)*
IBM i の各送達不能キュー規則のパターンとアクションの説明。

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 が使用禁止であったために DLQ に書き込まれた持続メッセージを、宛先キューに送達することを 3 回試行するように DLQ ハンドラーに指示します。

このセクションでは、規則に組み込むことができるキーワードについて説明します。 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。
- 指定できる値は、縦線 (|) で区分されています。 いずれか 1 つを指定できます。
- ACTION を除いて、どのキーワードも指定は任意です。

このセクションではまず、DLQ 上のメッセージと突き合わせるパターン照合キーワードについて説明します。 アクションのキーワード (一致したメッセージを DLQ ハンドラーによって処理する方法を示したキーワード) を取り上げます。

IBM i *DLQ パターン・マッチング・キーワード (IBM i)*
例を挙げて、パターン・マッチング・キーワードについて説明します。 これらのキーワードを使用して、IBM i の送達不能キューに入るメッセージと突き合わせる値を指定します。 パターン・マッチング・キーワードはすべてオプションです。

APPLIDAT (*ApplIdentity* データ|* (デフォルト))
メッセージ記述子 MQMD に指定された、DLQ 内のメッセージの *ApplIdentityData* 値。

APPLNAME (*PutApplName*|* (デフォルト))
DLQ 内のメッセージのメッセージ記述子 MQMD にある「*PutApplName*」フィールドに 指定された、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*|* (デフォルト))
DLQ 内のメッセージのメッセージ記述子 MQMD に指定された *PutApplType* 値。

DESTQ (*QueueName*|* (デフォルト))
メッセージの送り先のメッセージ・キューの名前。

DESTQM (*QueueManagerName*|* (デフォルト))
メッセージの送り先のメッセージ・キューのキュー・マネージャーの名前。

FEEDBACK (フィードバック|* (デフォルト))
MsgType 値が MQMT_REPORT であるとき、*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宛先キューへの着信の確認を必要とするものを識別することができます。

FORMAT (フォーマット|* (デフォルト))
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*|* (デフォルト))
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を必要とするものを識別することができます。

PERSIST (永続性|* (デフォルト))
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保存するものを指定することができます。

REASON (*ReasonCode*|* (デフォルト))
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*|* (デフォルト))
DLQ 内のメッセージのメッセージ記述子 MQMD に指定された応答先キューの名前。

REPLYQM (*QueueManagerName*|* (デフォルト))
REPLYQ キーワードに指定された応答先キューのキュー・マネージャー名。

USERID (*UserIdentifier*|* (デフォルト))
メッセージ記述子 MQMD に指定した DLQ 上のメッセージを発信したユーザーのユーザー ID。

IBM i *DLQ アクション・キーワード (IBM i)*
これらの送達不能キュー・アクション・キーワードを使用して、IBM i の送達不能キューにある一致メッセージの処理方法を指定します。

ACTION (DISCARD|IGNORE|RETRY|FWD)
この規則に定義されたパターンと一致した DLQ 内のメッセージについて行われるアクション。

DISCARD
メッセージは DLQ から削除されます。

IGNORE
メッセージが DLQ に保持されます。

RETRY
DLQ ハンドラーは、再度メッセージを宛先キューに書き込もうとします。

FWD
DLQ ハンドラーは、FWDQ キーワードに指定されたキューにメッセージを転送します。

ACTION キーワードは必ず指定する必要があります。 アクションを実行するための試行の回数は、RETRY キーワードで制御されます。 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。

FWDQ (キュー名|&DESTQ|&REPLYQ)
ACTION キーワードでメッセージの転送を指定した場合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。

メッセージ・パターン内に REPLYQ (?*) を指定して、FWDQ (&REPLYQ) を指定する規則がブランクの 応答キュー フィールドを持つメッセージと一致すると、エラー・メッセージを避けることができます。

FWDQM (*QueueManager* 名| & DESTQM | & REPLYQM | '' (デフォルト)
メッセージが転送されるキューのキュー・マネージャー。

QueueManagerName
ACTION (FWD) キーワードを指定した場合のメッセージの転送先となるキューのキュー・マネージャー名。

&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 (デフォルト) |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 (デフォルト) | CTX)
DLQ ハンドラーがメッセージを書き込む際の権限。

DEF
DLQ ハンドラー自体の権限でメッセージを書き込みます。

CTX
メッセージ・コンテキストのユーザー ID の権限でメッセージを書き込みます。 PUTAUT (CTX) の指定には、このユーザーの ID を使用する許可が必要です。

RETRY (*RetryCount*|1 (デフォルト))
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。

注 : 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 ハンドラーを再始動すると、規則を適用するために行われる試行回数は、ゼロにリセットされます。

IBM i *DLQ 規則テーブルの規則 (IBM i)*
IBM i の送達不能キュー規則テーブルは、構文や構造や内容に関する規則に準拠していなければなりません。

- 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、どの規則にも 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。
- 規則の先頭または終わり、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。
- 移植性を確保するため、行の有効長は 72 文字以下にしてください。
- 行の最後の非ブランク文字として正符号 (+) を使用した場合、その行の規則が次の行の最初の非ブランク文字に続くことを表します。 行の最後の非ブランク文字として負符号 (-) を使用した場合、その行の規則が次の行の先頭に続くことを表します。 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')
  ```

  これは 'ABCD' となります。

  ```
  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。 後続ブランクを除いて、任意の単一文字の代わりに疑問符 (?) を使用できます。 ゼロ以上の連続した文字の代わりにアスタリスク (*) を使用できます。 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。

- キーワード、ACTION、HEADER、RETRY、FWDQ、FWDQM、および PUTAUT のパラメーターにワイルドカード文字を含めることはできません。
- 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。 しかし、引用符で囲んだストリングの中の先行ブランクと組み込みブランクは、ワイルドカード突き合わせでも有効です。
- 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。 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、アスタリスクを数値パラメーターの一部として使用す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、数値パラメーターの一部としてアスタリスク (*) が含まれているため、`MSGTYPE('2*')` は無効です。
- 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ついてすべて有効です。

`'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

- ブランクまたは小文字を含むストリング、あるいは、ピリオド (.)、スラッシュ (/)、下線 (_)、およびパーセント記号 (%) を除く特殊文字を含むストリングは、一重引用符で囲む必要があります。 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。 ストリングに引用符が含まれる場合、一重引用符を 2 つ使用して、引用符の始めと終わりを示す必要があります。 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。

IBM i *IBM i での DLQ 規則テーブルの処理方法*

送達不能キュー・ハンドラーは、IBM i の送達不能キューに入っているメッセージに合致するパターンが組み込まれている規則を規則テーブル内で検索します。

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。 一致するパターンを持つ規則が検出されると、規則テーブルにより、その規則が指示するアクションが 試行されます。 DLQ ハンドラーは、規則を試行するたびにその規則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、試行回数が RETRY キーワードに指定された数に一致するまで、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・キーワードは、規則がアクションのみで構成できるようにデフォルト解釈されます。 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. 規則テーブルは、DLQ ハンドラーが開始したとき検証され、そのときエラーにフラグが付けられます。(DLQ ハンドラーが発行するエラー・メッセージについては、メッセージおよび理由コードで説明されています)。 いつでも規則テーブルを変更できますが、DLQ ハンドラーが再始動されて初めてその変更が有効になります。
4. DLQ ハンドラーは、メッセージ、MQDLH、メッセージ記述子の内容を変更しません。 DLQ ハンドラーは、メッセージ・オプション **MQPMO_PASS_ALL_CONTEXT** を使用して、常にメッセージを他のキューに書き込みます。
5. 規則テーブルの検証の目的が反復エラーの生成の防止であるため、規則テーブルの連続している構文エラーは認識されないことがあります。
6. DLQ ハンドラーは、 **MQOO_INPUT_AS_Q_DEF** オプションを指定して 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。

IBM i *IBM i での DLQ ハンドラー規則テーブルの例*

IBM i の送達不能キュー・ハンドラー規則テーブルのサンプル・コード。 このサンプル規則テーブルには、1 つの制御データ項目といくつかの規則が含まれています。

```
*************************************************************************
*    An example rules table for the STRMQMDLQ command        *
*************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* STRMQMDLQ, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to STRMQMDLQ,
* use the DLQ defined for the local queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.

* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation is always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never does things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
```

```
* where the problem is investigated.

REPLYQM(CCCC.*) +
ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it must be able
* to cope with the message being lost, so we can afford to
* discard the message.

PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)

* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We do not have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

すべての *DLQ* メッセージを確実に処理する

送達不能キュー・ハンドラーは、表示されているが、除去されなかった DLQ のすべてのメッセージのレコードを保持しています。 DLQ に含まれているメッセージの数ができるだけ少ないことを確認してください。

このタスクについて

フィルターとして DLQ ハンドラーを使用して、DLQ からメッセージの小サブセットを抽出する場合、DLQ ハンドラーは、処理しなかった DLQ 内のメッセージのレコードを引き続き保持します。 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。

これらの理由により、DLQ に含まれるメッセージの数をできるだけ少なくする必要があります。 理由は何であれ、廃棄されない、または他のキューに転送されないメッセージがキューに累積されるのを許容すると、DLQ ハンドラーのワークロードが増えて、DLQ 自体が満杯になる危険があります。

DLQ ハンドラーが DLQ を空にできるようにするには、以下の処置を行います。

手順

- 無視するメッセージについては、メッセージを別のキューに移動するアクションを使用してください。

  DLQ にメッセージを残す **ACTION (IGNORE)** コマンドを使用しないようにしてください。また、テーブル内の他の規則によって明示的にアドレス指定されていないメッセージについては、 **ACTION (IGNORE)** が想定されることに注意してください。 代わりに、メッセージを別のキューに移動するアクションを使用してください。 以下に例を示します。

  ```
  ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
  ```

- テーブル内の最終規則を "catch-all" にして、テーブル内の以前の規則によってアドレス指定されていないメッセージを処理します。

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)
```

これにより、表の最終ルールに該当するメッセージがキュー REALLY.DEAD.QUEUE に転送され、そこで手動で処理できます。 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。

管理トピックの操作

MQSC コマンドを使用して、管理トピックを管理します。

これらのコマンドの詳細については、MQSC コマンドを参照してください。

関連概念
管理トピック・オブジェクト
関連タスク
174 ページの『管理トピックの定義』
MQSC コマンド **DEFINE TOPIC** を使用して、管理トピックを作成します。 管理トピックを定義する場合は、オプションで各トピック属性を設定することができます。

175 ページの『管理トピック・オブジェクトの属性の表示』
MQSC コマンド **DISPLAY TOPIC** を使用して、管理トピック・オブジェクトを表示します。

176 ページの『管理トピックの属性の変更』
トピック属性は 2 つの方法で変更できます。1 つは **ALTER TOPIC** コマンドを使用する方法で、もう 1 つは **REPLACE** 属性を指定した **DEFINE TOPIC** コマンドを使用する方法です。

176 ページの『管理トピック定義のコピー』
DEFINE コマンドに LIKE 属性を指定すると、トピック定義をコピーできます。

177 ページの『管理トピック定義の削除』
MQSC コマンド **DELETE TOPIC** を使用すると、管理トピックを削除できます。

管理トピックの定義

MQSC コマンド **DEFINE TOPIC** を使用して、管理トピックを作成します。 管理トピックを定義する場合は、オプションで各トピック属性を設定することができます。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

明示的に設定されないトピックの属性はすべて、デフォルト管理トピック SYSTEM.DEFAULT.TOPIC から継承されます。このトピックは、システムのインストール済み環境がインストールされたときに作成されています。

例

例えば、以下の **DEFINE TOPIC** コマンドは、次の特性を持つ ORANGE.TOPIC と呼ばれるトピックを定義します。

- トピック・ストリング ORANGE に解決する。 トピック・ストリングを使用できる方法については、トピック・ストリングの結合を参照してください。

- ASPARENT に設定される属性はすべて、このトピックの親トピックによって定義される属性を使用する。 このアクションは、ルート・トピックである SYSTEM.BASE.TOPIC が見つかるまで、トピック・ツリーの上に向かって反復されます。 詳しくは、 トピック・ツリーを参照してください。

```
DEFINE TOPIC (ORANGE.TOPIC) +
TOPICSTR (ORANGE) +
DEFPRTY(ASPARENT) +
NPMSGDLV(ASPARENT)
```

注 :

- トピック・ストリングの値を除き、表示される属性値はすべてデフォルト値です。 ここに示されている値は説明のみを目的としたものです。 デフォルト値が自分の希望するものであるか、デフォルト値が変更されていないことが確実ならば、これらは省略することができます。 175 ページの『管理トピック・オブジェクトの属性の表示』も参照してください。
- 名前が ORANGE.TOPIC である管理トピックが、同じキュー・マネージャーに既にある場合、このコマンドは失敗します。 既存のトピックの定義を上書きする場合には、REPLACE 属性を使用してください。また、176 ページの『管理トピックの属性の変更』も参照してください。

関連資料

DEFINE TOPIC

管理トピック・オブジェクトの属性の表示

MQSC コマンド **DISPLAY TOPIC** を使用して、管理トピック・オブジェクトを表示します。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

このコマンドは、すべてのトピックを表示します。

```
DISPLAY TOPIC(ORANGE.TOPIC)
```

DISPLAY TOPIC コマンドを使用して属性を個別に指定することにより、属性を選択的に表示することができます。 以下に例を示します。

```
DISPLAY TOPIC(ORANGE.TOPIC) +
TOPICSTR +
DEFPRTY +
NPMSGDLV
```

このコマンドは、指定された 3 つの属性を表示します。

```
AMQ8633: Display topic details.
     TOPIC(ORANGE.TOPIC)                                     TYPE(LOCAL)
     TOPICSTR(ORANGE)                                        DEFPRTY(ASPARENT)
     NPMSGDLV(ASPARENT)
```

実行時に使用されるトピック ASPARENT 値を表示するには、 **DISPLAY TPSTATUS** コマンドを使用します。 例えば、次を使用します。

```
DISPLAY TPSTATUS(ORANGE) DEFPRTY NPMSGDLV
```

このコマンドは、以下のような詳細を表示します。

```
AMQ8754: Display topic status details.
TOPICSTR(ORANGE)            DEFPRTY(0)
NPMSGDLV(ALLAVAIL)
```

管理トピックを定義する場合、そのトピックは、明示的に指定されていない属性を、SYSTEM.DEFAULT.TOPIC と呼ばれるデフォルトの管理トピックから取得します。 これらのデフォルト属性を表示するには、次のコマンドを使用します。

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

関連資料
DISPLAY TOPIC
DISPLAY TPSTATUS

管理トピックの属性の変更

トピック属性は 2 つの方法で変更できます。1 つは **ALTER TOPIC** コマンドを使用する方法で、もう 1 つは **REPLACE** 属性を指定した **DEFINE TOPIC** コマンドを使用する方法です。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

例えば、 ORANGE.TOPIC を 5 にするには、以下のいずれかのコマンドを使用します。

- **ALTER** コマンドを使用

  ```
  ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
  ```

  このコマンドにより、1 つの属性、つまりこのトピックに送信されるメッセージのデフォルトの優先順位の属性は 5 に変更されます。その他のすべての属性は同じままです。
- **DEFINE** コマンドを使用

  ```
  DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
  ```

  このコマンドは、このトピックに送信されるメッセージのデフォルトの優先順位を変更します。 その他の属性にはすべてデフォルト値が与えられます。

このトピックに送信されるメッセージの優先順位を変更しても、既存のメッセージは影響を受けません。ただし、新しいメッセージは、公開アプリケーションによって提供されていない場合は、指定された優先順位を使用します。

関連資料
ALTER TOPIC
DISPLAY TOPIC

管理トピック定義のコピー

DEFINE コマンドに LIKE 属性を指定すると、トピック定義をコピーできます。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

以下のコマンドは、トピック MAGENTA.TOPIC(元のトピック ORANGE.TOPIC ではなく、TOPIC。 コピーされるトピックの名前は、そのトピックの作成時に入力されたのとまったく同じに入力します。 名前に小文字が含まれている場合には、単一引用符で名前を囲みます。

```
DEFINE TOPIC (MAGENTA.TOPIC) +
LIKE (ORANGE.TOPIC)
```

この形式の **DEFINE** コマンドを使用してトピック定義をコピーし、なおかつオリジナルの属性を変更することもできます。 以下に例を示します。

```
DEFINE TOPIC(BLUE.TOPIC) +
TOPICSTR(BLUE) +
LIKE(ORANGE.TOPIC)
```

トピック BLUE.TOPIC の属性をトピック GREEN.TOPIC にコピーし、パブリケーションをそれぞれの正しいサブスクライバー・キューに配信できない場合は、それらのパブリケーションをデッド・レター・キューに配置しないように指定することもできます。 以下に例を示します。

```
DEFINE TOPIC(GREEN.TOPIC) +
TOPICSTR(GREEN) +
LIKE(BLUE.TOPIC) +
USEDLQ(NO)
```

関連資料

DEFINE TOPIC

管理トピック定義の削除

MQSC コマンド **DELETE TOPIC** を使用すると、管理トピックを削除できます。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

```
DELETE TOPIC(ORANGE.TOPIC)
```

アプリケーションは、パブリケーションのトピックを開くことができなくなるか、またはオブジェクト名 ORANGE.TOPIC を使用して新しいサブスクリプションを作成することができなくなります。 トピック・オープンを持つアプリケーションを公開すると、解決されたトピック・ストリングのパブリッシュを続行できます。 このトピックに対して既に行われているサブスクリプションは、トピックが削除された後も引き続きパブリケーションを受信します。

このトピック・オブジェクトを参照してはいないものの、このトピック・オブジェクトが表していた解決済みのトピック・ストリング (この例では「ORANGE」) を使用しているアプリケーションは、作業を続行します。 この場合、それらのアプリケーションは、トピック・ツリー内のそれよりも上のトピック・オブジェクトからプロパティーを継承します。 詳しくは、 トピック・ツリーを参照してください。

関連資料

DELETE TOPIC

サブスクリプションの操作

MQSC コマンドを使用して、サブスクリプションを管理します。

このタスクについて

サブスクリプションは、以下の 3 つのタイプのいずれかで、タイプは **SUBTYPE** 属性で定義されています。

ADMIN
ユーザーによって管理定義されます。

PROXY
キュー・マネージャー間でパブリケーションを経路指定するための、内部で作成されるサブスクリプション。

API
例えば MQI MQSUB 呼び出しを使用するなどして、プログラマチックに作成されます。

これらのコマンドの詳細については、MQSC コマンドを参照してください。

管理サブスクリプションの定義

MQSC コマンド **DEFINE SUB** を使用して、管理サブスクリプションを作成します。 また、デフォルトのローカル・サブスクリプション定義内に定義されているデフォルトを使用することもできます。 あるいは、システムがインストールされたときに作成されたデフォルトのローカル・サブスクリプション、SYSTEM.DEFAULT.SUB の特性からサブスクリプション特性を修正することもできます。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

以下の **DEFINE SUB** コマンドは、以下の特性を持つ ORANGE という名前のサブスクリプションを定義します。

- 永続サブスクリプション。 つまり、このサブスクリプションは、キュー・マネージャーを再始動しても持続し、有効期限はありません。
- ORANGE トピック・ストリングに基づいて作成され、パブリッシュ・アプリケーションによってメッセージ優先順位が設定されているパブリケーションを受信します。
- このサブスクリプションに対して配信されたパブリケーションは、ローカル・キュー SUBQ に送信されます。 このキューは、そのサブスクリプションの定義よりも前に定義されていなければなりません。

```
DEFINE SUB (ORANGE) +
TOPICSTR (ORANGE) +
DESTCLAS (PROVIDED) +
DEST (SUBQ) +
EXPIRY (UNLIMITED) +
PUBPRTY (ASPUB)
```

注 :

- このサブスクリプションとトピック・ストリング名は一致している必要はありません。
- 宛先およびトピック・ストリングの値を除き、ここに示されているすべての属性値はデフォルト値です。 ここに示されている値は説明のみを目的としたものです。 デフォルト値が自分の希望するものであるか、デフォルト値が変更されていないことが確実ならば、これらは省略することができます。 179 ページの『サブスクリプションの属性の表示』も参照してください。
- 名前が ORANGE であるローカル・サブスクリプションが、同じキュー・マネージャーに既にある場合、このコマンドは失敗します。 キューの既存の定義を上書きする場合は **REPLACE** 属性を使用しますが、180 ページの『ローカル・サブスクリプションの属性の変更』も参照してください。
- キュー SUBQ が存在していない場合、このコマンドは失敗します。

関連資料
DEFINE SUB

サブスクリプションの属性の表示

DISPLAY SUB コマンドを使用すると、キュー・マネージャーが認識している任意のサブスクリプションの構成済み属性を表示できます。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

```
DISPLAY SUB(ORANGE)
```

属性を個別に指定すると、属性を選択的に表示できます。 以下に例を示します。

```
DISPLAY SUB(ORANGE) +
        SUBID +
        TOPICSTR +
        DURABLE
```

このコマンドにより、次のような 3 つの指定の属性が表示されます。

```
AMQ8096: IBM MQ subscription inquired.
     SUBID(414D5120414141202020202020202020EE921E4E20002A03)
     SUB(ORANGE)                                                    TOPICSTR(ORANGE)
     DURABLE(YES)
```

TOPICSTR は、このサブスクライバーが稼働している解決済みトピック・ストリングです。 サブスクリプションがトピック・オブジェクトを使用するように定義されている場合、そのオブジェクトからのトピック・ストリングは、サブスクリプションの作成時に指定されたトピック・ストリングへの接頭部として使用されます。 SUBID は、サブスクリプションが作成されるときにキュー・マネージャーによって割り当てられる固有 ID です。 これは、一部のサブスクリプション名が長すぎるか、またはそれが非実用的になる可能性がある別の文字セットに含まれている可能性があるため、表示する便利な属性です。

サブスクリプションを表示するための代替方法としては、以下のように SUBID を使用する方法があります。

```
DISPLAY SUB +
        SUBID(414D5120414141202020202020202020EE921E4E20002A03) +
        TOPICSTR +
        DURABLE
```

このコマンドの出力は、前のコマンドの出力と同じです。

```
AMQ8096: IBM MQ subscription inquired.
 SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                                  TOPICSTR(ORANGE)
   DURABLE(YES)
```

デフォルトでは、キュー・マネージャー上のプロキシー・サブスクリプションは表示されません。 それらを表示するには、PROXY の **SUBTYPE** または ALL を指定します。

DISPLAY SBSTATUS コマンドを使用すると、ランタイム属性を表示できます。 例えば、次のコマンドを使用します。

```
DISPLAY SBSTATUS(ORANGE) NUMMSGS
```

以下の出力が表示されます。

```
AMQ8099: IBM MQ subscription status inquired.
   SUB(ORANGE)
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   NUMMSGS(0)
```

管理サブスクリプションを定義する場合、そのサブスクリプションは、明示的に指定されていない属性を、SYSTEM.DEFAULT.SUB と呼ばれるデフォルトのサブスクリプションから取得します。 これらのデフォルト属性を表示するには、次のコマンドを使用します。

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

関連資料

DISPLAY SUB

ローカル・サブスクリプションの属性の変更

サブスクリプション属性は 2 つの方法で変更できます。1 つは **ALTER SUB** コマンドを使用する方法で、もう 1 つは **REPLACE** 属性を指定した **DEFINE SUB** コマンドを使用する方法です。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

ORANGE という名前のサブスクリプションに送信されるメッセージの優先順位を 5 に変更する場合は、以下のいずれかのコマンドを使用します。

- **ALTER** コマンドを使用

  ```
  ALTER SUB(ORANGE) PUBPRTY(5)
  ```

  このコマンドにより、1 つの属性、つまりこのサブスクリプションに送信されるメッセージの優先順位の属性は 5 に変更されます。 その他のすべての属性はそのままで、変更はありません。
- **DEFINE** コマンドを使用

  ```
  DEFINE SUB(ORANGE) PUBPRTY(5) REPLACE
  ```

  このコマンドは、このサブスクリプションに送信されるメッセージの優先順位だけでなく、それぞれデフォルト値が与えられた他のすべての属性も変更します。

このサブスクリプションに送信されるメッセージの優先順位を変更すると、既存のメッセージは影響を受けません。 ただし、新しいメッセージはすべて、指定された優先順位になります。

関連資料

ALTER SUB

DEFINE SUB

ローカル・サブスクリプション定義のコピー

DEFINE コマンドに **LIKE** 属性を指定すると、サブスクリプション定義をコピーできます。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

```
DEFINE SUB(BLUE) +
       LIKE(ORANGE)
```

サブ REAL の属性をサブ THIRD.SUB にコピーし、配信されたパブリケーションの **correlID** が、パブリッシャー **correlID** ではなく THIRD であることを指定することもできます。 以下に例を示します。

```
DEFINE SUB(THIRD.SUB) +
       LIKE(BLUE) +
       DESTCORL(ORANGE)
```

関連資料

DEFINE SUB

ローカル・サブスクリプションの削除

MQSC コマンド **DELETE SUB** を使用すると、ローカル・サブスクリプションを削除できます。

始める前に

注 : このタスクの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

例

```
DELETE SUB(ORANGE)
```

サブスクリプションは、SUBID を使用しても削除できます。

```
DELETE SUB SUBID(414D5120414141202020202020202020EE921E4E20002A03)
```

関連資料

DELETE SUB

サブスクリプションとの突き合わせによるメッセージの検査

サブスクリプションが定義されると、そのサブスクリプションはキューに関連付けられます。 このサブスクリプションに一致するパブリッシュ済みメッセージは、このキューに送られます。 MQSC コマンドを使用して、現在サブスクリプションのキューに入れられているメッセージを確認します。

始める前に

注 : このタスクのステップでは、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

このタスクについて

以下の MQSC コマンドは、メッセージを受信したサブスクリプションのみを表示することに注意してください。

現在、サブスクリプション用のキューに入れられているメッセージがないか検査するには、以下の手順を実行します。

手順

1. サブスクリプション・タイプ DISPLAY SBSTATUS(*sub_name*) NUMMSGS のキューに入れられたメッセージを確認するには、 179 ページの『サブスクリプションの属性の表示』 を参照してください。
2. **NUMMSGS** 値がゼロより大きい場合は、 DISPLAY SUB(*sub_name*)DEST と入力して、サブスクリプションに関連付けられているキューを識別します。
3. 返されるキューの名前を使用して、144 ページの『サンプル・プログラムを使用してキューをブラウズする』 で説明されている技法を使用すると、メッセージを表示できます。

関連資料
DISPLAY SBSTATUS

サービスの取り扱い

サービス・オブジェクトは、追加プロセスをキュー・マネージャーの一部として管理するための手段です。サービスを使用して、キュー・マネージャーの開始および停止時に始動および停止するプログラムを定義することができます。 IBM MQ サービスは必ず、キュー・マネージャーを開始したユーザーのユーザー ID の制御下で開始されます。

このタスクについて

サービス・オブジェクトには、以下のいずれかのタイプを指定できます。

サーバー
サーバーは、**SERVTYPE** パラメーターが SERVER に指定されているサービス・オブジェクトです。 サーバー・サービス・オブジェクトは、指定したキュー・マネージャーの開始時に実行されるプログラムの定義です。 サーバー・サービス・オブジェクトでは、通常、長期間稼働するプログラムを定義します。 例えば、サーバー・サービス・オブジェクトを使用して、**runmqtrm** などのトリガー・モニター・プロセスを実行することができます。

同時に実行できるサーバー・サービス・オブジェクトのインスタンスは、1 つだけです。 実行中のサーバー・サービス・オブジェクトの状況を、MQSC コマンド **DISPLAY SVSTATUS** を使用してモニターすることができます。

コマンド
コマンドは、**SERVTYPE** パラメーターが COMMAND に指定されているサービス・オブジェクトです。 コマンド・サービス・オブジェクトは、サーバー・サービス・オブジェクトとよく似ていますが、コマンド・サービス・オブジェクトは、同時に複数のインスタンスを実行できますが、それぞれの状況を MQSC コマンド **DISPLAY SVSTATUS** でモニターすることはできません。

MQSC コマンド **STOP SERVICE** を実行してプログラムを停止する場合は、MQSC コマンド **START SERVICE** で開始されたプログラムがまだアクティブであるかどうかの確認は行われません。

関連資料
DEFINE SERVICE
DISPLAY SVSTATUS
START SERVICE
STOP SERVICE

サービス・オブジェクトの定義

MQSC コマンド **DEFINE SERVICE** でサービス・オブジェクトを定義します。

始める前に

注 : このタスクを実行するには、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

手順

- MQSC コマンド **DEFINE SERVICE** を使用して、サービス・オブジェクトを定義します。

  定義しなければならない属性は以下のとおりです。

  SERVTYPE
  サービス・オブジェクトのタイプを定義します。 次の値を指定できます。

  SERVER
  サーバー・サービス・オブジェクト。

  同時に実行できるサーバー・サービス・オブジェクトのインスタンスは、1 つだけです。 サーバー・サービス・オブジェクトの状況は、MQSC コマンド **DISPLAY SVSTATUS** を使用してモニターできます。

  COMMAND
  コマンド・サービス・オブジェクト。

  コマンド・サービス・オブジェクトでは、複数のインスタンスを同時に実行することができます。 コマンド・サービス・オブジェクトの状況は、モニターできません。

  STARTCMD
  サービスを開始するために実行されるプログラム。 プログラムの完全修飾パスを指定する必要があります。

  STARTARG
  開始プログラムに渡される引数。

  STDERR
  サービス・プログラムの標準のエラー (stderr) のリダイレクト先のファイルのパスを指定します。

  STDOUT
  サービス・プログラムの標準出力 (stdout) のリダイレクト先のファイルのパスを指定します。

  STOPCMD
  サービスを停止するために実行されるプログラム。 プログラムの完全修飾パスを指定する必要があります。

  STOPARG
  停止プログラムに渡される引数。

  CONTROL
  サービスの開始方法と停止方法を指定します。

  MANUAL
  サービスを自動的に開始または停止しません。 **START SERVICE** コマンドおよび **STOP SERVICE** コマンドの使用によって、サービスの開始と停止を制御します。 これがデフォルト値です。

  QMGR
  定義するサービスは、キュー・マネージャーの開始および停止に合わせて開始および停止されます。

  STARTONLY
  サービスはキュー・マネージャーの開始に合わせて開始されますが、キュー・マネージャーが停止してもサービスに対しては停止を要求しません。

関連タスク

184 ページの『サービスの管理』
サービス・オブジェクトのインスタンスは、キュー・マネージャーによって自動的に開始および停止することも、MQSC コマンド **START SERVICE** および **STOP SERVICE** を使用して開始および停止することもできます。

関連資料

DEFINE SERVICE
DISPLAY SVSTATUS

START SERVICE
STOP SERVICE

サービスの管理

サービス・オブジェクトのインスタンスは、キュー・マネージャーによって自動的に開始および停止することも、MQSC コマンド **START SERVICE** および **STOP SERVICE** を使用して開始および停止することもできます。

始める前に

注 : このタスクを実行するには、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

手順

- サービス・オブジェクトのインスタンスを自動的に開始または停止するには、キュー・マネージャーで **CONTROL** パラメーターを設定します。手動でこれを行うには、MQSC コマンド **START SERVICE** および **STOP SERVICE** を使用します。

  サービス・オブジェクトのインスタンスを開始すると、キュー・マネージャーのエラー・ログに、サービス・オブジェクトの名前と開始されたプロセスのプロセス ID が入ったメッセージが書き込まれます。以下に、開始するサーバー・サービス・オブジェクトのログ・エントリーの例を示します。

```
02/15/2005 11:54:24 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5028: The Server 'S1' has started. ProcessId(13031).

EXPLANATION:
The Server process has started.
ACTION:
None.
```

  以下に、開始するコマンド・サービス・オブジェクトのログ・エントリーの例を示します。

```
02/15/2005 11:53:55 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5030: The Command 'C1' has started. ProcessId(13030).

EXPLANATION:
The Command has started.
ACTION:
None.
```

  インスタンス・サーバー・サービスを停止すると、キュー・マネージャーのエラー・ログに、サービスの名前と停止プロセスのプロセス ID が入ったメッセージが書き込まれます。 以下に、サーバー・サービス・オブジェクトの停止に関するログ項目の例を示します。

```
02/15/2005 11:54:54 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5029: The Server 'S1' has ended. ProcessId(13031).

EXPLANATION:
The Server process has ended.
ACTION:
None.
```

関連タスク

185 ページの『service.env ファイルでの追加の環境変数の定義』

サービスが開始されると、 サービス・プロセスが開始される環境がキュー・マネージャーの環境から継承されます。 service.env 環境指定変更ファイルの 1 つに定義したい変数を追加することによって、サービス・プロセスの環境に設定する追加の環境変数を定義することができます。

関連資料

Multiplatforms での STOP SERVICE (サービスの停止)

Multiplatforms での START SERVICE (サービスの開始)

service.env ファイルでの追加の環境変数の定義

サービスが開始されると、 サービス・プロセスが開始される環境がキュー・マネージャーの環境から継承されます。 service.env 環境指定変更ファイルの 1 つに定義したい変数を追加することによって、サービス・プロセスの環境に設定する追加の環境変数を定義することができます。

このタスクについて

環境変数を追加できるファイルには、以下の 2 つがあります。

- マシン・スコープの service.env ファイル
- キュー・マネージャーの有効範囲 service.env ファイル

使用可能な場合には両方のファイルが処理されますが、キュー・マネージャーの有効範囲のファイル内の定義は、マシンの有効範囲のファイル内の定義よりも優先されます。

service.env ファイルには、任意の環境変数を指定できます。 例えば、 IBM MQ サービスが多数のコマンドを実行する場合、 service.env ファイルに **PATH** ユーザー変数を設定すると便利な場合があります。

注 : 変数に設定した値を環境変数にすることはできません。例えば、 CLASSPATH= *%CLASSPATH%* は正しくありません。 同様に、 Linux PATH= *$PATH* :/opt/mqm/bin では、予期しない結果になります。

CLASSPATH は大文字でなければならず、クラスパス・ステートメントにはリテラルのみを含めることができます。 一部のサービス (Telemetry など) は、独自のクラスパスを設定します。 service.env で定義されている **CLASSPATH** が追加されます。

service.env ファイルで定義されている変数の形式は、名前と値の変数のペアのリストです。 変数ごとに 1 行に定義する必要があります。各変数が明示的に定義されるものと見なされ、空白文字を含みます。

手順

- 環境変数をマシン・スコープの service.env ファイルに追加します。
  このファイルは以下の場所にあります。
  - Linux AIX AIX and Linux システム上の。/var/mqm
  - Windows Windows システムでのインストール時に選択したデータ・ディレクトリー。
- 環境変数をキュー・マネージャーの有効範囲の service.env ファイルに追加します。
  このファイルは、キュー・マネージャーのデータ・ディレクトリーにあります。 例えば、 QMNAME という名前のキュー・マネージャーの環境指定変更ファイルの場所は、次のようになります。
  - Linux AIX AIX and Linux システムの場合、 /var/mqm/qmgrs/QMNAME/service.env
  - Windows Windows システムの場合、 C:\ProgramData\IBM\MQ\qmgrs\QMNAME\service.env

service.env ファイルの例

```
#********************************************************************#
#*                                                                  *#
```

```
#* <N_OCO_COPYRIGHT>                                                   *#
#* Licensed Materials - Property of IBM                               *#
#*                                                                    *#
#* 63H9336                                                            *#
#* (C) Copyright IBM Corporation 2005, 2024.                          *#
#*                                                                    *#
#* <NOC_COPYRIGHT>                                                    *#
#*                                                                    *#
#*********************************************************************#
#*********************************************************************#
#* Module Name: service.env                                           *#
#* Type       : IBM MQ service environment file                       *#
#* Function   : Define additional environment variables to be set     *#
#*              for SERVICE programs.                                 *#
#* Usage      : <VARIABLE>=<VALUE>                                    *#
#*                                                                    *#
#*********************************************************************#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE
```

関連タスク

186 ページの『サービス定義での置き換え可能挿入の使用』
サービス・オブジェクトの定義内のトークンを置き換えることができます。 置換されるトークンは、サービス・プログラムの実行時に、展開されたテキストに自動的に置き換えられます。

関連資料

環境変数の説明

サービス定義での置き換え可能挿入の使用

サービス・オブジェクトの定義内のトークンを置き換えることができます。 置換されるトークンは、サービス・プログラムの実行時に、展開されたテキストに自動的に置き換えられます。

このタスクについて

置換トークンは、以下の共通トークンのリストから、またはファイル `service.env` に定義されている任意の変数から取得できます。

手順

- 置き換え可能な挿入を使用するには、**STARTCMD**、**STARTARG**、**STOPCMD**、 **STOPARG**、**STDOUT** または **STDERR** のストリングのいずれかに + 文字で囲んだトークンを挿入します。

  この例については、187 ページの『サーバー・サービス・オブジェクトの使用』および 189 ページの『コマンド・サービス・オブジェクトの使用』を参照してください。

  サービス・オブジェクトの定義でトークンを置換する目的で使用できる共通トークンを以下に示します。

  MQ_INSTALL_PATH
  IBM MQ がインストールされている位置。

  MQ_DATA_PATH
  IBM MQ データ・ディレクトリーの位置。

  - Linux AIX AIX and Linux システムでは、IBM MQ データ・ディレクトリーのインストール先は `/var/mqm/` です。
  - Windows Windows システムでは、IBM MQ データ・ディレクトリーの位置は IBM MQ のインストール時に選択されたデータ・ディレクトリーです。

  QMNAME
  現在のキュー・マネージャー名。

  MQ_SERVICE_NAME
  サービスの名前。

MQ_SERVER_PID
このトークンは、**STOPARG** 引数および **STOPCMD** 引数でのみ使用できます。

サーバー・サービス・オブジェクトの場合、このトークンは **STARTCMD** 引数および **STARTARG** 引数によって開始されたプロセスのプロセス ID に置き換えられます。 それ以外の場合、このトークンは 0 に置き換えられます。

MQ_Q_MGR_DATA_PATH
キュー・マネージャーのデータ・ディレクトリーの位置。

MQ_Q_MGR_DATA_NAME
キュー・マネージャーの変換された名前。 名前変換の詳細については、 IBM MQ ファイル名の理解 を参照してください。

サーバー・サービス・オブジェクトの使用

これらの例は、トリガー・モニターまたはその他のプログラムを開始するためにサーバー・サービス・オブジェクトを定義、使用、および変更する方法を示しています。

始める前に

注 : これらの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

これらの例は、特に明記されていない限り、 UNIX スタイルのパス区切り文字を使用して作成されています。

手順

1. **DEFINE SERVICE MQSC コマンドを使用して、サーバー・サービス・オブジェクトを定義します。**

```
DEFINE SERVICE(S1) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+bin/runmqtrm') +
STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
STOPCMD('+MQ_INSTALL_PATH+bin/amqsstop') +
STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

説明

+MQ_INSTALL_PATH+ は、インストール・ディレクトリーを表すトークンです。
+QMNAME+ は、キュー・マネージャーの名前を表すトークンです。
ACCOUNTS.INITIATION.QUEUE は、始動キューです。
amqsstop は、IBM MQ の付属サンプル・プログラムです。このサンプル・プログラムは、キュー・マネージャーに、プロセス ID に対するすべての接続を切断するよう要求します。 amqsstop は PCF コマンドを生成するため、コマンド・サーバーが稼働している必要があります。
+MQ_SERVER_PID+ は、停止プログラムに渡されるプロセス ID を表すトークンです。

共通トークンのリストについては、186 ページの『サービス定義での置き換え可能挿入の使用』を参照してください。

2. **サーバー・サービス・オブジェクトのインスタンスは、キュー・マネージャーが次に開始されたときに実行されます。 ただし、 START SERVICE MQSC コマンドを使用して、サーバー・サービス・オブジェクトのインスタンスを即時に開始することはできます。**

```
START SERVICE(S1)
```

3. **DISPLAY SVSTATUS MQSC コマンドを使用して、サーバー・サービス・プロセスの状況を表示します。**

```
DISPLAY SVSTATUS(S1)
```

4. **ALTER SERVICE MQSC コマンドを使用して、サーバー・サービス・プロセスを手動で再始動することにより、サーバー・サービス・オブジェクトを変更し、更新を反映させます。**

   サーバー・サービス・オブジェクトを変更して、始動キューを JUPITER.INITIATION.QUEUE と指定します。

   ```
   ALTER SERVICE(S1) +
   STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
   ```

   注 : 実行中のサービスは、再始動されるまで、そのサービス定義に対する更新を取得しません。
5. **STOP SERVICE および START SERVICE MQSC コマンドを使用して、サーバー・サービス・プロセスを再始動し、変更が反映されるようにします。**

   ```
   STOP SERVICE(S1)
   ```

   続いて、

   ```
   START SERVICE(S1)
   ```

   サーバー・サービス・プロセスが再開され、188 ページの『4』 で行われた変更を取得します。

   注 : MQSC コマンド **STOP SERVICE** は、サービス定義に **STOPCMD** 引数を指定した場合にのみ使用できます。

引数の受け渡しのその他の例

- **キュー・マネージャーの開始時に runserv というプログラムを開始するためのサーバー・サービス・オブジェクトを定義します。**

  これは、 **DEFINE SERVICE** MQSC コマンドを使用して行います。

  この例は、Windows スタイルのパス区切り文字を使用して記述されます。

  開始プログラムに渡される引数の 1 つは、スペースを含むストリングです。 この引数は、単一のストリングとして渡す必要があります。 これを行うには、コマンド・サービス・オブジェクトを定義するために、以下のコマンドに示すように二重引用符を使用します。

  ```
  DEFINE SERVICE(S1) SERVTYPE(SERVER) CONTROL(QMGR) +
  STARTCMD('C:\Program Files\Tools\runserv.exe') +
  STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
  STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')


  DEFINE SERVICE(S4) +
  CONTROL(QMGR) +
  SERVTYPE(SERVER) +
  STARTCMD('C:\Program Files\Tools\runserv.exe') +
  STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
  STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')
  ```

  説明

  +QMNAME+ は、キュー・マネージャーの名前を表すトークンです。

  "C:\Program Files\Tools\" はスペースを含むストリングで、単一のストリングとして渡されます。
- **キュー・マネージャーの開始時にトリガー・モニターを自動的に開始するために使用できるサーバー・サービス・オブジェクトを定義します。**

  これは、 **DEFINE SERVICE** MQSC コマンドを使用して行います。

  ```
  DEFINE SERVICE(TRIG_MON_START) +
  CONTROL(QMGR) +
  ```

```
SERVTYPE(SERVER) +
STARTCMD('runmqtrm') +
STARTARG('-m +QMNAME+ -q +IQNAME+')
```

説明

+QMNAME+ は、キュー・マネージャーの名前を表すトークンです。

+IQNAME+ は、ユーザーが service.env ファイルの内の 1 つに定義する環境変数で、開始キューの名前を表します。

関連資料

ALTER SERVICE

DEFINE SERVICE

DISPLAY SVSTATUS

START SERVICE

STOP SERVICE

コマンド・サービス・オブジェクトの使用

以下の例は、キュー・マネージャーの開始時または停止時にオペレーティング・システムのシステム・ログに項目を書き込むプログラムを開始するためのコマンド・サービス・オブジェクトを定義する方法を示しています。

始める前に

注 : これらの例では、**DEFINE SERVICE** MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

これらの例は、 UNIX スタイルのパス分離文字を使用して作成されています。

このタスクについて

以下に例を示します。

logger は、オペレーティング・システムのシステム・ログに項目を書き込むことができる IBM MQ に付属のサンプル・プログラムです。

+QMNAME+ は、キュー・マネージャーの名前を表すトークンです。

手順

- キュー・マネージャーの開始時または停止時にオペレーティング・システムのシステム・ログに項目を書き込むプログラムを開始するコマンド・サービス・オブジェクトを定義します。

```
DEFINE SERVICE(S2) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STARTCMD('/usr/bin/logger') +
STARTARG('Queue manager +QMNAME+ starting') +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

- キュー・マネージャーの停止時にのみオペレーティング・システムのシステム・ログに項目を書き込むプログラムを開始するには、コマンド・サービス・オブジェクトを定義します。

```
DEFINE SERVICE(S3) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

関連資料

DEFINE SERVICE

トリガー操作のためのオブジェクトの管理

これらの例は、キュー上の特定の条件が満たされたときにアプリケーションを自動的に開始する方法を示しています。 その条件の一例として、キュー上のメッセージ数が指定の数に達した場合があります。 この機能は、トリガー操作 と呼ばれています。 トリガー操作をサポートしているオブジェクトを定義する必要があります。

始める前に

注 : これらの例では、MQSC コマンドを実行する必要があります。 これを行う方法はプラットフォームによって異なります。 見る管理 IBM MQMQSC コマンドの使用。

これらの例は、 UNIX スタイルのパス分離文字を使用して作成されています。

このタスクについて

トリガー操作の詳細については、 トリガーによる IBM MQ アプリケーションの開始を参照してください。

手順

- トリガー操作のためのアプリケーション・キューを定義します。

  アプリケーション・キューとは、アプリケーションが MQI を介してメッセージ用に使用するローカル・キューのことです。 トリガー操作では、いくつかのキュー属性をアプリケーション・キューに定義する必要があります。

  トリガー操作自体は、**Trigger** 属性 (MQSC コマンドの中の TRIGGER) によって使用可能になります。 以下に示す例では、トリガー・イベントは、MOTOR.INSURANCE.QUEUE というローカル・キューに優先順位 5 以上のメッセージが 100 個入れられたときに生成されます。

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
MAXMSGL (2000) +
DEFPSIST (YES) +
INITQ (MOTOR.INS.INIT.QUEUE) +
TRIGGER +
TRIGTYPE (DEPTH) +
TRIGDPTH (100)+
TRIGMPRI (5)
```

  ここで、

  QLOCAL (MOTOR.INSURANCE.QUEUE)
  定義するアプリケーション・キューの名前です。

  PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
  トリガー・モニター・プログラムで開始されるアプリケーションを定義するプロセス定義の名前です。

  MAXMSGL (2000)
  キューに入れるメッセージの最大長です。

  DEFPSIST (YES)
  デフォルトでメッセージをこのキュー上で存続させるように指定します。

  INITQ (MOTOR.INS.INIT.QUEUE)
  キュー・マネージャーがトリガー・メッセージを入れる開始キューの名前です。

  TRIGGER
  トリガー属性値です。

  TRIGTYPE (DEPTH)
  要求した優先順位 (TRIGMPRI) を持つメッセージの数が TRIGDPTH で指定した数に達したときにトリガー・イベントを生成するように指定します。

  TRIGDPTH (100)
  トリガー・イベントを生成するのに必要なメッセージ数です。

TRIGMPRI (5)
トリガー・イベントを生成するかどうかを決める際に、キュー・マネージャーが考慮に入れるメッセージの優先順位です。 優先度 5 以上のメッセージだけが考慮に入れられます。

- 開始キューの定義

トリガー・イベントが発生すると、キュー・マネージャーは、アプリケーション・キュー定義に指定された開始キューにトリガー・メッセージを入れます。 開始キューには特別の設定値はありませんが、参考として以下に示す MOTOR.INS.INIT.QUEUE というローカル・キューの定義を使用することができます。

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
GET (ENABLED) +
NOSHARE +
NOTRIGGER +
MAXMSGL (2000) +
MAXDEPTH (1000)
```

- プロセスの定義

プロセス定義を作成するには、DEFINE PROCESS コマンドを使用します。 プロセス定義は、アプリケーション・キューからメッセージを処理するために使用されるようにアプリケーションを定義します。 アプリケーション・キュー定義は、使用されるプロセスを命名することによって、そのメッセージを処理するために使用されるアプリケーションとアプリケーション・キューを関連付けます。 この関連付けは、アプリケーション・キュー MOTOR.INSURANCE.QUEUE の PROCESS 属性によって行われます。 次の MQSC コマンドは、この例で識別されている必須プロセス MOTOR.INSURANCE.QUOTE.PROCESS を定義します。

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
DESCR ('Insurance request message processing') +
APPLTYPE (UNIX) +
APPLICID ('/u/admin/test/IRMP01') +
USERDATA ('open, close, 235')
```

説明

MOTOR.INSURANCE.QUOTE.PROCESS
プロセス定義の名前です。

DESCR ('Insurance request message processing')
この定義を関連付けるアプリケーション・プログラムの記述です。 このテキストは、DISPLAY PROCESS コマンドを使用すると表示されます。 これは、プロセスが行う事柄を識別するのに役立ちます。 ストリングの中でスペースを使用する場合は、ストリングを単一引用符で囲む必要があります。

APPLTYPE (UNIX)
開始するアプリケーションのタイプです。

APPLICID ('/u/admin/test/IRMP01')
アプリケーションの実行可能プログラムの名前で、完全修飾ファイル名として指定されます。 Windows システムでは、標準的な APPLICID 値は `c:\appl\test\irmp01.exe` です。

USERDATA ('open, close, 235')
アプリケーションで使用できるユーザー定義のデータです。

- プロセス定義の属性の表示

定義の結果を調べるには、DISPLAY PROCESS コマンドを使用します。 以下に例を示します。

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)


24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
DESCR ('Insurance request message processing')
APPLICID ('/u/admin/test/IRMP01')
USERDATA (open, close, 235)
```

```
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
APPLTYPE (UNIX)
```

MQSC コマンド ALTER PROCESS を使用して既存のプロセス定義を変更したり、DELETE PROCESS を使用してプロセス定義を削除したりできます。

2 つのシステム間での dmpmqmsg ユーティリティーの使用

dmpmqmsg ユーティリティー (以前の *qload*) を使用すると、キューまたはそのメッセージの内容をファイルにコピーまたは移動することができます。

概要

dmpmqmsg で作成したファイルは、必要に応じて保存し、後でキューにメッセージを再ロードするために使用することができます。

重要 :

1. このファイルは、ユーティリティーが理解できる特定の形式になっています。 しかし、このファイルは人間も読める形式になっているため、再ロードする前にエディターで更新できます。 ファイルを編集しない場合は、その形式を変更してはなりません。
2. **dmpmqmsg** ユーティリティーは、 AIX, Linux, and Windows 用のランタイム・ファイルセットに付属しているため、 IBM MQ サーバーとクライアントの両方で使用できます。

考えられる用途は、次のとおりです。

- キューにあるメッセージをファイルに保存する。 これはアーカイブなどの目的で行われ、後で元のキューに再ロードされます。
- 以前にファイルに保存されたメッセージをキューに再ロードする。
- キューから古いメッセージを削除する。
- 保管場所からテスト・メッセージを「適用」する (必要に応じて、メッセージ間の正確な時間も保持する)。

重要 : SupportPac MO03 は、ローカルまたはクライアントのバインディングを指定する場合に **-l** パラメーターを使用していました。 **-l** は **-c** パラメーターに置き換えられました。

コード・ページ情報には、**-c** の代わりに **-P** が使用されるようになりました。

コマンドと使用可能なパラメーターについて詳しくは、dmpmqmsg を参照してください。

Linux で Windows マシンを使用して dmpmqmsg ユーティリティーを使用する例

Linux マシン上にキュー・マネージャーがあり、そのキュー (*Q1*) のメッセージを、同じキュー・マネージャー上の別のキュー (*Q2*) に移動します。 **dmpmqmsg** ユーティリティーを Windows マシンから開始するとします。

キュー (*Q1*) には、サンプル・アプリケーション **amqsput** (ローカル・キュー・マネージャー) または **amqsputc** (リモート・キュー・マネージャー) を使って追加した 4 つのメッセージがあります。

Linux マシンでは、次のように表示されます。

```
display ql(Q1) CURDEPTH
        2 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(4)
```

Linux のキュー・マネージャーを指すように MQSERVER 環境変数を設定します。 以下に例を示します。

```
set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/veracruz.x.com(1414)
```

veracruz は、マシンの名前です。

dmpmqmsg ユーティリティーを実行して、キュー *Q1* から読み取り、出力を `c:\temp\mqqload.txt` に保管します。

MQSERVER によって確立された Linux ホストおよびポートで実行されているキュー・マネージャー *QM_VER* に、リモート・クライアントとして接続します。 リモート・クライアントとしての接続は、属性 `-c` を使って行います。

```
dmpmqmsg -m QM_VER -i Q1 -f c:\temp\mqqload.txt -c
Read      - Files:    0   Messages:       4   Bytes:           22
Written - Files:    1   Messages:       4   Bytes:           22
```

出力ファイル `c:\temp\mqqload.txt` には、 **dmpmqmsg** ユーティリティーが認識するフォーマットを使用したテキストが含まれています。

Windows マシンで、 **dmpmqmsg** コマンド (`-i` オプションの代わりに `-o` オプションを使用) を発行して、Windows マシン上のファイルから Linux マシン上のキュー (*Q2*) をロードします。

```
dmpmqmsg -m QM_VER -o Q2 -f c:\temp\mqqload.txt -c
Read      - Files:    1   Messages:       4   Bytes:           22
Written - Files:    0   Messages:       4   Bytes:           22
```

Linux マシン上のキューに、ファイルからリストアされた 4 つのメッセージがあることに注目してください。

```
display ql(Q2) CURDEPTH
        6 : display ql(Q2) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q2)
TYPE(QLOCAL)
     CURDEPTH(4)
```

Linux マシンで、以下の操作を行います。

メッセージを元のキューから削除します。

```
clear qlocal(Q1)
        4 : clear qlocal(Q1)
AMQ8022: IBM MQ queue cleared.
```

元のキューにメッセージが存在していないことを確認します。

```
display ql(Q1) CURDEPTH
        5 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(0)
```

コマンドおよびそのパラメーターの説明については、dmpmqmsg を参照してください。

関連概念

193 ページの『dmpmqmsg ユーティリティーの使用例』

dmpmqmsg ユーティリティー (以前の **qload**) を使用するための簡単な方法。

dmpmqmsg ユーティリティーの使用例

dmpmqmsg ユーティリティー (以前の **qload**) を使用するための簡単な方法。

ファイルへのキューのアンロード

コマンド行で以下のオプションを使用して、キューにあるメッセージをファイルに保存します。

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile
```

このコマンドは、キューからメッセージのコピーを取り、指定のファイルに保存します。

一連のファイルへのキューのアンロード

ファイル名に insert 文字を使用して、一連のファイルにキューをアンロードできます。 このモードでは、各メッセージが新規ファイルに書き込まれます。

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile%n
```

このコマンドは、キューをファイル myfile1、myfile2、myfile3 などにアンロードします。

ファイルからのキューのロード

194 ページの『ファイルへのキューのアンロード』で保存したメッセージをキューに再ロードするには、コマンド行で以下のオプションを使用します。

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

このコマンドは、キューをファイル myfile1、myfile2、myfile3 などにアンロードします。

一連のファイルからのキューのロード

ファイル名に insert 文字を使用して、一連のファイルからキューにロードできます。 このモードでは、各メッセージが新規ファイルに書き込まれます。

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

このコマンドは、キューをファイル myfile1、myfile2、myfile3 などにロードします。

1 つのキューから別のキューへのメッセージのコピー

194 ページの『ファイルへのキューのアンロード』のファイル・パラメーターを別のキュー名に置き換え、次のオプションを使用します。

```
dmpmqmsg -m QM1 -i Q1 -o Q2
```

このコマンドにより、メッセージを 1 つのキューから別のキューにコピーすることができます。

1 つのキューから別のキューへの最初の 100 個のメッセージのコピー

前の例のコマンドを使用し、 -r#100 オプションを追加します。

```
dmpmqmsg -m QM1 -i Q1 -o Q2 -r#100
```

1 つのキューから別のキューへのメッセージの移動

194 ページの『ファイルからのキューのロード』のバリエーションです。 キューを参照するだけの **-i** (小文字) と、キューから破壊的に取得する **-I** (大文字) の間の違いに注目してください。

```
dmpmqmsg -m QM1 -I Q1 -o Q2
```

1 日を経過したメッセージを 1 つのキューから別のキューへ移動する操作

この例は、経過日数選択の使用について示しています。 経過日数範囲より古いか、新しいか、特定の範囲内のメッセージを選択できます。

```
dmpmqmsg -m QM1 -I Q1 -o Q2 -T1440
```

現在キューにあるメッセージの経過日数の表示

コマンド行で次のオプションを使用します。

```
dmpmqmsg -m QM1 -i Q1 -f stdout -dT
```

メッセージ・ファイルの処理

194 ページの『ファイルへのキューのアンロード』に従ってキューからメッセージをアンロードした後、そのファイルを編集することもできます。

キューからアンロードしたときに指定しなかった表示オプションの 1 つを使用するように、ファイルの形式を変更することもできます。

キューのアンロード後でも、**dmpmqmsg** ユーティリティーを使用して、ファイルを必要な形式に再処理できます。 コマンド行で次のオプションを使用します。

```
dmpmqmsg -f c:\oldfile -f c:\newfile -dA
```

コマンドおよびそのパラメーターの説明については、dmpmqmsg を参照してください。

リモート IBM MQ オブジェクトの処理

MQSC コマンド、PCF コマンド、または administrative REST API を使用して、リモート・キュー・マネージャー上の IBM MQ オブジェクトを管理できます。 これらの方法のいずれかを使用するためには、その前に、ローカル・キュー・マネージャーとリモート・キュー・マネージャーの間の伝送キューとチャネルを定義して、リモート・キュー・マネージャーへのコマンドの送信とローカル・キュー・マネージャーへの応答の受信を行えるようにしておく必要があります。 あるいは、キュー・マネージャー・クラスターを構成することもできます。この場合も、同じリモート管理方法を使用できます。

このタスクについて

リモート管理用のキュー・マネージャーを準備するには、ローカル・キュー・マネージャーで以下のオブジェクトを構成する必要があります。

- リスナー。
- リモート・キュー・マネージャーと同じ名前の伝送キュー。
- リモート・キュー・マネージャーの接続の詳細が定義された送信側チャネル。
- リモート・キュー・マネージャー上の送信側チャネルと同じ名前の受信側チャネル。

リモート・キュー・マネージャーでも以下のオブジェクトを構成する必要があります。

- リスナー。

- ローカル・キュー・マネージャーと同じ名前の伝送キュー。
- ローカル・キュー・マネージャーの接続の詳細が定義された送信側チャネル。
- ローカル・キュー・マネージャー上の送信側チャネルと同じ名前の受信側チャネル。

これらのオブジェクトの構成について詳しくは、196 ページの『リモート管理のためのキュー・マネージャーの構成』を参照してください。

あるいは、キュー・マネージャー・クラスターを構成することもできます。 クラスター は、単一のネットワークを介してキュー・マネージャー間の直接の通信が可能になるように設定されたキュー・マネージャーの集合体です。この場合、伝送キュー、チャネル、およびキューの煩雑な定義の必要はありません。 クラスターは、簡単にセットアップでき、通常は、一部論理的に関連付けられるキュー・マネージャーを含み、データまたはアプリケーションを共用する必要があります。 最小クラスターでも、システム管理のコストが削減されます。

クラスター内のキュー・マネージャーのネットワークを確立する場合、従来の分散キューイング環境を確立するのに比べて、定義が少なくてすみます。 作成する定義が少ないので、ネットワークを迅速かつ簡単にセットアップまたは変更することが可能で、定義にエラーが発生するリスクが軽減されます。

クラスターをセットアップするには、キュー・マネージャーにつき 1 つのクラスター送信側 (CLUSSDR) 定義と 1 つのクラスター受信側 (CLUSRCVR) 定義が必要です。 伝送キュー定義またはリモート・キュー定義は必要ありません。 リモート管理の基本は、クラスター内で使用されるときは同じですが、定義自体は大幅に単純化されます。

クラスターの構成について詳しくは、キュー・マネージャー・クラスターの構成を参照してください。

手順

- リモート IBM MQ オブジェクトを管理する方法については、以下のサブトピックを参照してください。
  - 196 ページの『リモート管理のためのキュー・マネージャーの構成』
  - 200 ページの『リモート管理でコマンド・サーバーを管理する』
  - 201 ページの『リモート・キュー・マネージャーに対する MQSC コマンドの発行』
  - 203 ページの『コード化文字セット間のデータ変換』

リモート管理のためのキュー・マネージャーの構成

administrative REST API、MQSC コマンド、または PCF コマンドを使用して、ローカル・キュー・マネージャーからリモート・キュー・マネージャーを管理できます。 リモート・キュー・マネージャーは、同じシステム上、異なるインストール済み環境内 (つまり、同じ環境を使用する異なるシステム上)、または異なる IBM MQ 環境に存在する可能性があります。 ローカル・キュー・マネージャーからキュー・マネージャーをリモート管理するためには、その前に、送信側チャネル、受信側チャネル、リスナー、伝送キューを、キュー・マネージャーごとに作成しておく必要があります。 これらのチャネルとキューを使用することで、コマンドをリモート・キュー・マネージャーに送信して、応答をローカル・キュー・マネージャーで受信することができるようになります。 これらのキューとチャネルの作成手順は、administrative REST API、MQSC コマンド、または PCF コマンドのどれを使用する場合も同じです。

始める前に

- 以下の手順では、サンプル・キュー・マネージャーとして `source.queue.manager` と `target.queue.manager` を使用します。 ご使用のシステムにこれらのキュー・マネージャーを作成してから開始して以下のステップに従うか、該当する各ステップで、現在使用しているキュー・マネージャー名に置き換えて操作する必要があります。
- 以下の手順では、トランスポート・タイプとして TCP/IP を使用します。 このタスクを完了するためには、両方のシステムの IP アドレスが分かっていなければなりません。
- 以下の手順では、ネットワーク・ポート 1818 を使用するリスナーをローカル・システムに作成し、ネットワーク・ポート 1819 を使用するリスナーをリモート・システムに作成します。 他のポートを使用することもできますが、該当する各ステップで、ご使用のポート値に置き換えて操作する必要があります。

- 手順の中のコマンドは、ローカルで実行するか、Telnet などのネットワーク機能を介して実行する必要があります。

このタスクについて

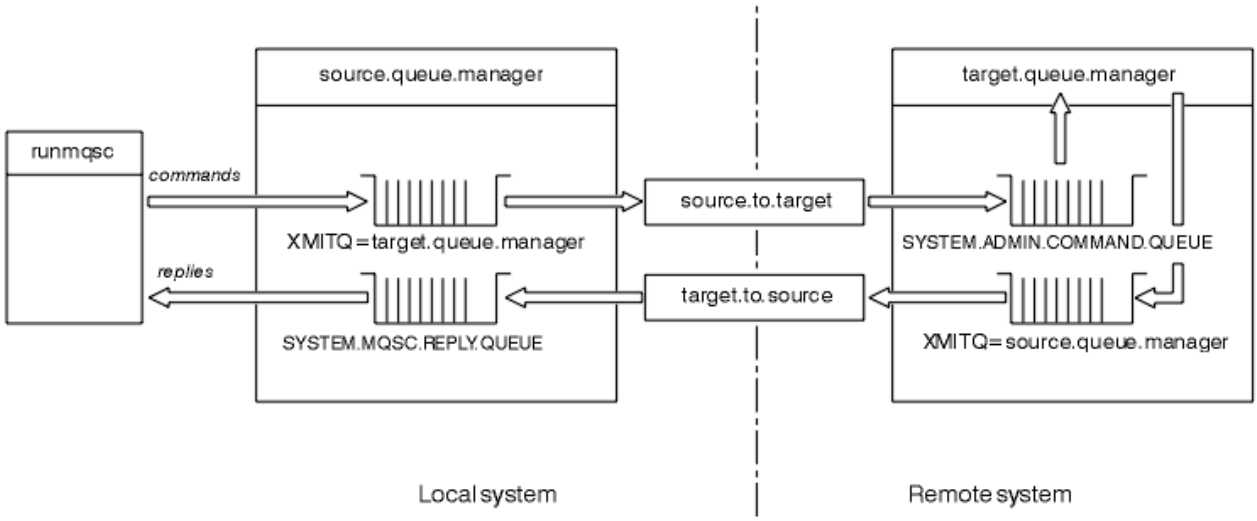


図 *15.* リモート管理のためのチャネルとキューのセットアップ

197 ページの図 15 は、リモート管理に必要なキュー・マネージャー、キュー、およびチャネルの構成を示しています。

- オブジェクト `source.queue.manager` はソース・キュー・マネージャーであり、administrative REST API、MQSC コマンド、または PCF コマンドをここから発行することや、これらのコマンドの結果をここに返すことができます。
- オブジェクト `target.queue.manager` は、コマンドを処理し、オペレーター・メッセージを生成するターゲット・キュー・マネージャーの名前です。
- コマンドは、リモート・キュー・マネージャーと同じ名前の伝送キューに書き込まれます。 この例では、`target.queue.manager` です。 伝送キューは、専用ローカル・キューであり、メッセージを MCA が受信し、リモート・キュー・マネージャーに送信する前に、一時的にメッセージを保管します。
- コマンドは、`source.to.target` チャネルによってリモート・キュー・マネージャーの `SYSTEM.ADMIN.COMMAND.QUEUE` に送信されます。 チャネルの各端は、個別に定義されます。 一方の終端は送信側であり、もう一方の終端は受信側です。 2 つの定義は同じ名前にして、共に単一のメッセージ・チャネルを構成する必要があります。
- コマンド出力は、コマンドの送信元ローカル・キュー・マネージャーと同じ名前のリモート伝送キューに書き込まれます。 この例では、`source.queue.manager` です。
- 出力は `target.to.source` チャネルによって適切な応答キューに送信され、ここから元のコマンドによって取得されて出力されます。

手順

1. リモート・システムのキュー・マネージャーで、コマンド・キュー `SYSTEM.ADMIN.COMMAND.QUEUE` が存在することを確認します。 このキューは、キュー・マネージャーが作成されるときにデフォルトとして作成されます。
2. リモート・システムで、キュー・マネージャー上でコマンド・サーバーが実行中であることを確認します。 コマンド・サーバーが実行中でない場合、リモート管理はできません。
   a) キュー・マネージャーに対して **runmqsc** を開始します。 例えば、キュー・マネージャーが `target.queue.manager` の場合は、次のコマンドを入力します。

   ```
   runmqsc target.queue.manager
   ```

b) 次のコマンドを入力して、コマンド・サーバーの状況を表示します。

```
DISPLAY QMSTATUS CMDSERV
```

c) 以下のコマンドを入力して、 **runmqsc** コマンド・プロンプトを終了します。

```
end
```

d) コマンド・サーバーが開始していない場合は開始します。 例えば、キュー・マネージャーが target.queue.manager の場合は、次のコマンドを入力します。

```
strmqcsv target.queue.manager
```

3. ローカル・キュー・マネージャーのチャネル、リスナー、および伝送キューを定義します。

a) キュー・マネージャーに対して **runmqsc** を開始します。 例えば、キュー・マネージャーが source.queue.manager の場合は、次のコマンドを入力します。

```
runmqsc source.queue.manager
```

b) 送信側チャネルを定義します。 この送信側チャネルの名前は、リモート・キュー・マネージャー上の受信側チャネルと同じでなければなりません。 例えば、次の MQSC コマンドを入力しますが、 **CONNAME** の値をリモート・キュー・マネージャーの IP アドレスとリスナーのポート番号に置き換えます。

```
DEFINE CHANNEL ('source.to.target') +
CHLTYPE(SDR) +
CONNAME (localhost:1819) +
XMITQ ('target.queue.manager') +
TRPTYPE(TCP)
```

c) 受信側チャネルを定義します。 この受信側チャネルの名前は、リモート・キュー・マネージャー上の送信側チャネルと同じでなければなりません。 例えば、以下のコマンドを入力します。

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(RCVR) +
TRPTYPE(TCP)
```

d) ローカル・キュー・マネージャー上のリスナーを定義します。 例えば、以下のコマンドを入力します。

```
DEFINE LISTENER ('source.queue.manager') +
TRPTYPE (TCP) +
PORT (1818)
```

e) ローカル・キュー・マネージャー上の伝送キューを定義します。 この伝送キューの名前は、リモート・キュー・マネージャーと同じでなければなりません。 例えば、以下のコマンドを入力します。

```
DEFINE QLOCAL ('target.queue.manager') +
USAGE (XMITQ)
```

f) リスナーを始動します。 例えば、以下のコマンドを入力します。

```
START LISTENER ('source.queue.manager')
```

g) 以下のコマンドを入力して、 **runmqsc** コマンド・プロンプトを終了します。

```
end
```

4. リモート・キュー・マネージャーのチャネル、リスナー、伝送キューを定義します。

a) キュー・マネージャーに対して **runmqsc** を開始します。 例えば、キュー・マネージャーが target.queue.manager の場合は、次のコマンドを入力します。

```
runmqsc target.queue.manager
```

b) 送信側チャネルを定義します。 この送信側チャネルの名前は、ローカル・キュー・マネージャー上の受信側チャネルと同じでなければなりません。 例えば、次の MQSC コマンドを入力しますが、**CONNAME** の値をローカル・キュー・マネージャーの IP アドレスとリスナーのポート番号に置き換えます。

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(SDR) +
CONNAME (localhost:1818) +
XMITQ ('source.queue.manager') +
TRPTYPE(TCP)
```

c) 受信側チャネルを定義します。 この受信側チャネルの名前は、ローカル・キュー・マネージャー上の送信側チャネルと同じでなければなりません。例えば、次のコマンドを入力します。

```
DEFINE CHANNEL ('source.to.target') +
CHLTYPE(RCVR) +
TRPTYPE(TCP)
```

d) リスナーを定義します。 例えば、以下のコマンドを入力します。

```
DEFINE LISTENER ('target.queue.manager') +
TRPTYPE (TCP) +
PORT (1819)
```

e) 伝送キューを定義します。 この伝送キューの名前は、ローカル・キュー・マネージャーと同じでなければなりません。 例えば、以下のコマンドを入力します。

```
DEFINE QLOCAL ('source.queue.manager') +
USAGE (XMITQ)
```

f) リスナーを始動します。 例えば、以下のコマンドを入力します。

```
START LISTENER ('target.queue.manager')
```

g) 次のコマンドを入力して、**runmqsc** を終了します。

```
end
```

5. ローカル・システム上の送信側チャネルを開始します。

a) キュー・マネージャーに対して **runmqsc** を開始します。 例えば、キュー・マネージャーが source.queue.manager の場合は、次のコマンドを入力します。

```
runmqsc source.queue.manager
```

b) 送信側チャネルを開始します。 例えば、以下のコマンドを入力します。

```
START CHANNEL ('source.to.target')
```

c) 次のコマンドを入力して、**runmqsc** を終了します。

```
end
```

6. リモート・システム上の送信側チャネルを開始します。

a) キュー・マネージャーに対して runmqsc を開始します。 例えば、キュー・マネージャーが target.queue.manager の場合は、次のコマンドを入力します。

```
runmqsc target.queue.manager
```

b) 送信側チャネルを開始します。 例えば、以下のコマンドを入力します。

```
START CHANNEL ('target.to.source')
```

c) 次のコマンドを入力して、**runmqsc** を終了します。

```
end
```

7. ローカル・システムからリモート・キュー・マネージャーに MQSC コマンドを送信して、構成が正常に完了したことをテストします。

a) ローカル・システムからリモート・キュー・マネージャーの **runmqsc** コマンド・プロンプトを開始します。 例えば、以下のコマンドを入力します。

```
runmqsc -w 30 -m source.queue.manager target.queue.manager
```

b) 次のコマンドを入力して、リモート・キュー・マネージャー上のキューを表示します。

```
DISPLAY QUEUE (*)
```

成功すると、リモート・キュー・マネージャー上のキューのリストが表示されます。

c) これらのステップがうまくいかない場合は、両方のシステムのチャネルが実行状態であることを確認します。 チャネルが実行中でなく開始しない場合は、チャネルと伝送キューが正しく構成されていることと、コマンド・サーバーが実行中であることを確認してください。 例えば、送信側チャネルに正しい CONNAME が指定されていることと、伝送キューの名前が正しいことを確認します。 また、キュー・マネージャーのログでセキュリティー例外を確認してください。問題解決に役立つ場合があります。

タスクの結果

ローカル・システムからリモート・キュー・マネージャーをリモート管理できるようにキュー・マネージャーが構成されました。

次のタスク

- MQSC コマンドを使用したリモート管理について詳しくは、201 ページの『リモート・キュー・マネージャーに対する MQSC コマンドの発行』を参照してください。
- PCF コマンドを使用した管理プログラムの作成について詳しくは、26 ページの『IBM MQ プログラマブル・コマンド・フォーマットの使用』を参照してください。
- リモート管理での administrative REST API の使用について詳しくは、79 ページの『REST API によるリモート管理』を参照してください。

リモート管理でコマンド・サーバーを管理する

各キュー・マネージャーには、コマンド・サーバーが関連付けられています。 コマンド・サーバーは、リモート・キュー・マネージャーからの着信コマンド、またはアプリケーションからの PCF コマンドを処理します。 コマンド・サーバーは処理のために、そのコマンドをキュー・マネージャーに渡し、完了コードやオペレーター・メッセージを戻します。 コマンド・サーバーの開始、停止、および状況の表示を行えます。 コマンド・サーバーは、PCF コマンド、MQAI に関するすべての管理およびリモート管理にも必須です。

始める前に

コマンド・サーバーは、キュー・マネージャー属性 **SCMDSERV** の値によって、キュー・マネージャーの開始時に自動的に始動される場合と手動で始動しなければならない場合があります。 コマンド・サーバーが自動的に始動する場合は、strmqcsv または endmqcsv コマンドを使用してコマンド・サーバーを始動した

り停止したりすることはできません。 MQSC コマンド **ALTER QMGR** を使用することによって、**SCMDSERV** 属性の値を変更できます。 デフォルトでは、コマンド・サーバーは自動的に始動されます。

キュー・マネージャーを停止すると、そのキュー・マネージャーと関連付けられているコマンド・サーバーも終了します。

手順

- コマンド・サーバーの状況を表示します。
  a) 以下のコマンドを入力して、該当するキュー・マネージャーの **runmqsc** コマンド・プロンプトを開始します。

  ```
  runmqsc target.queue.manager
  ```

  ここで、target.queue.manager は、表示されるコマンド・サーバーに関連したキュー・マネージャーです。
  b) 次の MQSC コマンドを入力して、コマンド・サーバーの状況を表示します。

  ```
  DISPLAY QMSTATUS CMDSERV
  ```

  c) 以下のコマンドを入力して、 **runmqsc** コマンド・プロンプトを終了します。

  ```
  end
  ```

- コマンド・サーバーが自動的に始動するように設定されていない場合は、次のコマンドを入力して、コマンド・サーバーを始動します。

  ```
  strmqcsv target.queue.manager
  ```

  ここで、target.queue.manager は、開始されるコマンド・サーバーに関連したキュー・マネージャーです。
- コマンド・サーバーが自動的に始動するように設定されていない場合は、次のコマンドを入力して、コマンド・サーバーを停止します。

  ```
  endmqcsv target.queue.manager
  ```

  ここで、target.queue.manager は、停止されるコマンド・サーバーに関連したキュー・マネージャーです。

  デフォルトでは、コマンド・サーバーは制御された方法で停止します。 コマンドに -i フラグを追加することによって、コマンド・サーバーを即時に停止できます。

リモート・キュー・マネージャーに対する MQSC コマンドの発行

リモート管理のためのキュー・マネージャーを構成した後、ローカル・システム上で特定の形式の **runmqsc** コマンドを使用することによって、リモート・キュー・マネージャーで MQSC コマンドを実行できます。 各コマンドは、リモート・キュー・マネージャーのコマンド・キュー SYSTEM.ADMIN.COMMAND.QUEUE に Escape PCF として送信されます。 応答は SYSTEM.MQSC.REPLY.QUEUE キューで受信されます。

始める前に

MQSC コマンドを使用してキュー・マネージャーをリモート管理するためには、その前に、196 ページの『リモート管理のためのキュー・マネージャーの構成』のステップを完了して、チャネル、伝送キュー、リスナー、およびコマンド・サーバーを構成しておく必要があります。

手順

1. リモート・キュー・マネージャーでコマンド・サーバーが実行中であることを確認します。

   キュー・マネージャー上でコマンド・サーバーを始動する方法については、200 ページの『リモート管理でコマンド・サーバーを管理する』を参照してください。
2. その後、ソース・キュー・マネージャーで、次の 2 つの方法のいずれかで MQSC コマンドを実行できます。
   - 対話式に実行する。以下のコマンドを使用して **runmqsc** を開始します。
     - リモート・キュー・マネージャーが z/OS 上にある場合は、次のコマンドを入力します。

       ```
       runmqsc -w 30 -x -m source.queue.manager target.queue.manager
       ```
     - リモート・キュー・マネージャーが Multiplatforms にある場合は、次のコマンドを入力します。

       ```
       runmqsc -w 30 -m source.queue.manager target.queue.manager
       ```
   - コマンド・ファイルから実行する。
     a. リモート・システムで実行する MQSC コマンドを 1 行 1 コマンドでテキスト・ファイルに記述します。
     b. **runmqsc** コマンドで -v フラグを使用して、ローカル・キュー・マネージャーで MQSC コマンドを検証します。 -v フラグを指定すると、コマンドが有効かどうかが検査されますが、コマンドは実行されません。 特定のコマンドがリモート・キュー・マネージャーには適用できてもローカル・キュー・マネージャーには適用できない場合は、それらのコマンドが失敗する可能性があることに注意してください。

        ```
        runmqsc -v source.queue.manager < myCmdFile.in > results.out
        ```

        myCmdFile.in にはチェックする MQSC コマンドが含まれており、 results.out ファイルには、コマンドの検証結果が含まれています。
     c. 以下のいずれかのコマンドを入力して、リモート・キュー・マネージャーでコマンド・ファイルを実行します。
        - リモート・キュー・マネージャーが z/OS 上にある場合は、次のコマンドを入力します。

          ```
          runmqsc -w 30 -x -m source.queue.manager target.queue.manager < myCmdFile.in >
          results.out
          ```
        - リモート・キュー・マネージャーが Multiplatforms にある場合は、次のコマンドを入力します。

          ```
          runmqsc -w 30 -m source.queue.manager target.queue.manager < myCmdFile.in >
          results.out
          ```

   使用されるパラメーターは、以下のパラメーターです。

   -w *seconds*
   MQSC コマンドを間接モードで実行することを指定します。この場合、コマンドはコマンド・サーバーの入力キューに置かれ、順番に実行されます。

   変数 *seconds* は、リモート・キュー・マネージャーからの応答を待機する時間の長さを秒単位で指定します。 この時間が経過した後に受信した応答は破棄されますが、MQSC コマンドはリモート・キュー・マネージャーでそのまま実行されます。 コマンドがタイムアウトになると、ローカル・キュー・マネージャーで次のメッセージが生成されます。

   ```
   AMQ8416: MQSC timed out waiting for a response from the command server.
   ```

MQSC コマンドの発行を停止すると、ローカル・キュー・マネージャーは、到着したタイムアウト応答を表示し、それ以降の応答を破棄します。

-x
リモート・キュー・マネージャーが z/OS 上のキュー・マネージャーであることを指定します。

-m *localQMgrName*
リモート・キュー・マネージャーにコマンドを実行依頼するために使用するローカル・キュー・マネージャーの名前を指定します。

次のタスク

リモート側で MQSC コマンドを実行するのが困難な場合は、以下のようにしてください。

- リモート・キュー・マネージャーが実行中であることを確認します。
- リモート・システムでコマンド・サーバーが実行中であることを確認します。
- チャネル切断間隔が終了していないことを確認します。 例えば、チャネルが開始してしばらくしてからシャットダウンした場合です。 これは、チャネルを手動操作で開始した場合に特に重要です。
- ローカル・キュー・マネージャーから送信される要求がターゲット・キュー・マネージャーにとって意味をなしていることを確認してください。 例えば、リモート・キュー・マネージャーではサポートされないパラメーターが要求に含まれている場合があります。
- MQSC コマンドに関する問題の解決も参照してください。

コード化文字セット間のデータ変換

IBM MQ で定義された形式 (組み込み形式とも呼ばれる) のメッセージ・データは、キュー・マネージャーによって 1 つのコード化文字セットからもう 1 つのコード化文字セットに変換することができます。ただし、2 つのコード化文字セットが、1 つの言語または類似する言語グループに関連付けられていることが必要です。

例えば、ID (CCSID) がそれぞれに 850 と 500 であるコード化文字セット間の変換は、両方とも西欧の言語に該当するため、サポートされます。

ASCII への EBCDIC 改行 (NL) 文字変換については、 mqs.ini ファイルのすべてのキュー・マネージャー・スタンザ および **AMQ_CONVEBCDICNEWLINE** 環境変数を参照してください。

サポートされている変換は、データ変換処理に定義されています。

CCSID 37 と 500 の間の変換は、 IBM MQ Appliance、 Windows、 Linux、および macOS でサポートされています。

キュー・マネージャーがメッセージを組み込み形式に変換できない場合

CCSID が別の各国語グループを表している場合には、キュー・マネージャーはメッセージを組み込み形式に 自動的に変換することはできません。 例えば、CCSID 850 と CCSID 1025 (キリル文字スクリプトを使用する言語用の EBCDIC コード化文字セット) 間の変換はサポートされていません。これは、一方のコード化文字セットの文字の多くが、もう一方のコード化文字セットで表現できないためです。 さまざまな各国語で稼働しているキュー・マネージャーのネットワークがあり、一部のコード化文字セット間でのデータ変換がサポートされていない場合に、デフォルト変換を使用することができます。

`ccsid_part2.tbl` が適用されるプラットフォームの場合、詳しくは、 `ccsid_part2.tbl` を使用した 207 ページの『デフォルトのデータ変換の指定』 を参照してください。 `ccsid_part2.tbl` ファイルが適用されるプラットフォーム以外のプラットフォームでのデフォルトのデータ変換については、 204 ページの『デフォルトのデータ変換』 で説明しています。

拡張 Unicode データ変換サポート

本製品は、データ変換において Unicode 8.0 標準で定義されているすべての Unicode 文字をサポートします。 これには、サロゲート・ペア (U+FFFF より上の Unicode コード・ポイントを表す `X'D800'` から `X'DFFF'` の範囲内の 2 バイト UTF-16 文字のペア) を含む UTF-16 の完全サポートが含まれます。

1 つの CCSID 内の事前作成された文字が他の CCSID 内の結合文字シーケンスにマップされた場合、文字シーケンスの結合もサポートされます。

Unicode と CCSID 1388、1390、1399、4933、5488、16884 との間のデータ変換が一部のプラットフォームで拡張され、現在これらの CCSID 用に定義されているすべてコード・ポイントがサポートされるようになりました (Unicode 補助面のコード・ポイントにマップされるものも含む)。

CCSID 1390、1399、および 16884 の場合は、これに JIS X 0213 (JIS2004) 標準で定義された文字も含まれます。

Unicode と 6 つの新規 CCSID (1374 から 1379) との間の変換のサポートも追加されました。

ccsid_part2.tbl ファイル

追加ファイル `ccsid_part2.tbl` が提供されています。

`ccsid_part2.tbl` ファイルは `ccsid.tbl` ファイルより優先され、次の意味を持ちます:

- CCSID 項目の追加や変更が可能になります
- デフォルトのデータ変換を指定します
- さまざまなコマンド・レベルのデータを指定します

`ccsid_part2.tbl` は、以下のプラットフォームにのみ適用されます。

- Linux Linux - すべてのバージョン
- Windows Windows

Windows IBM MQ for Windows では、 `ccsid_part2.tbl` はデフォルトでディレクトリー *MQDataRoot*`\conv\table` にあります。 また、IBM MQ for Windows では、サポートされるすべてのコード・セットがこれに記録されています。

Linux IBM MQ for Linux では、 `ccsid_part2.tbl` はディレクトリー *MQDataRoot*`/conv/table` にあり、サポートされているコード・セットは、IBM MQ に用意されている変換テーブルに入っています。

`ccsid_part2.tbl` ファイルは、追加の CCSID 情報を提供するために以前のバージョンの IBM MQ で使用されていた既存の `ccsid.tbl` ファイルを置き換えますが、`ccsid.tbl` ファイルは引き続き IBM MQ によって解析されるため、削除してはなりません。

詳細については、205 ページの『ccsid_part2.tbl ファイル』を参照してください。

ccsid.tbl file

`ccsid_part2.tbl` が適用されないプラットフォームでは、ファイル `ccsid.tbl` は以下の目的で使用されます。

- AIX AIX では、サポートされるコード・セットはオペレーティング・システムによって内部的に保持されます。
- このファイルは、追加のコード・セットを指定します。 追加のコード・セットを指定するには、`ccsid.tbl` を編集する必要があります (これを行う方法については、ファイル内で説明します) 。
- このファイルは、すべてのデフォルトのデータ変換を指定します。

`ccsid.tbl` に記録された情報を更新することができます。例えば、ご使用のオペレーティング・システムの将来のリリースで追加のコード化文字セットがサポートされている場合に、これを行うことができます。

デフォルトのデータ変換

データ変換が通常はサポートされていない 2 つのマシン間でチャネルをセットアップする場合、チャネルが作動するようにデフォルトのデータ変換を使用可能にしなければなりません。

ccsid_part2.tbl が適用されないプラットフォームでデフォルトのデータ変換を有効にするには、デフォルトの EBCDIC CCSID とデフォルトの ASCII CCSID を指定するように ccsid.tbl ファイルを編集します。 この方法に関する指示は、このファイルに入っています。 チャネルを使用して接続されるすべてのマシン上でこれを実行しなければなりません。 変更内容を有効にするには、キュー・マネージャーを再始動します。

デフォルトのデータ変換プロセスは、次のようになります。

- ソース CCSID とターゲット CCSID 間の変換がサポートされていなくても、CCSID のソース環境およびターゲット環境の両方が EBCDIC または ASCII のいずれかである場合は、文字データは変換されずにターゲット・アプリケーションに渡されます。
- 一方の CCSID が ASCII コード化文字セットを表し、もう一方の CCSID が EBCDIC コード化文字セットを表す場合、IBM MQ は ccsid.tbl で定義されているデフォルトのデータ変換機構 CCSID を使用してデータを変換します。

注 : メッセージ用として指定されたコード化文字セットとデフォルトのコード化文字セット中で同じコード値を持つ文字に、変換対象文字を制限してください。 IBM MQ オブジェクト名に有効な文字セットのみを使用する (IBM MQ オブジェクトの命名 で定義されている) 場合 一般的には、この要件を満たすことになります。 日本で使用されている EBCDIC CCSID 290、930、1279、および 5026 では例外が発生します。この場合、小文字は他の EBCDIC CCSID で 使用されるものとは異なるコードを持ちます。

ユーザー定義形式でのメッセージの変換

ユーザー定義形式のメッセージを、キュー・マネージャーによって 1 つのコード化文字セットから別のコード化文字セットに変換することはできません。 ユーザー定義形式のデータが変換を必要とする場合は、形式ごとにデータ変換出口が必要となります。 ユーザー定義の形式で文字データを変換するためにデフォルトの CCSID を使用しないでください。 ユーザー定義形式のデータ変換およびデータ変換出口の作成について詳しくは、データ変換出口の作成を参照してください。

キュー・マネージャー CCSID の変更

ALTER QMGR コマンドの **CCSID** 属性を使用してキュー・マネージャーの CCSID を変更した場合は、キュー・マネージャーを停止して再始動し、コマンド・サーバーおよびチャネル・プログラムを含むすべての実行中のアプリケーションが停止して再始動されるようにします。

これは、キュー・マネージャー CCSID が変更されるときに実行しているアプリケーションが既存の CCSID を使用し続けるために必要です。

ccsid_part2.tbl ファイル

ccsid_part2.tbl ファイルは、追加の CCSID 情報を提供するために使用されます。 ccsid_part2.tbl ファイルは、 IBM MQ 9.0 より前に使用されていた ccsid.tbl ファイルを置き換えます。

注 : IBM MQ 9.0 が追加の CCSID 情報を提供する前に使用された ccsid.tbl ファイルは、引き続き IBM MQ によって構文解析されますが、削除すべきではありません。 ただし、ccsid_part2.tbl 内の項目は、ccsid.tbl 内の他の項目より優先されます。

ccsid.tbl ではなく ccsid_part2.tbl を使用するようにしてください。ccsid_part2.tbl には以下の特長があるからです。

- Unicode エンコード値のサポートが含まれている。 IBM MQ 9.0 以降、この製品はデータ変換に関して Unicode 8.0 標準に定義されているすべての Unicode 文字をサポートしています (UTF-16 のフルサポートを含む)。 詳しくは、203 ページの『コード化文字セット間のデータ変換』を参照してください。
- CCSID 項目のバージョンを指定することで、選択したコマンド・レベルのみにそれらの項目を適用できる。

ccsid_part2.tbl ファイルを使用することによって、次のことを実行できます。

- CCSID 項目を追加または変更します
- デフォルトのデータ変換を指定します

- さまざまなコマンド・レベルのデータを指定します

ccsid_part2.tbl ファイルは、以下のプラットフォームにのみ適用されます。

- Linux - すべてのバージョン
- Windows

ccsid_part2.tbl ファイルの場所は、ご使用のプラットフォームによって異なります。

- Linux のすべてのバージョンの *MQDataRoot*/conv/table ディレクトリー。
- Windows 上の *MQDataRoot*\conv\table ディレクトリー

CCSID 項目の追加または変更

ccsid_part2.tbl ファイル内の項目は、次のような形式になっています:

```
<CCSID number> <Base CCSID> <DBCS CodePage> <SBCS CodePage>
<Type> <Encoding> <ACRI> <Name>
```

CCSID 1200 (UTF-16) の項目の例として、次のようなものがあります:

```
  1200 1200 1200 1200 3 8 0 UTF-16
```

注 : ACRI の値について詳しくは、ccsid_part2.tbl ファイル内のコメントを参照してください。

ccsid_part2.tbl の形式は次のとおりです:

タイプは次のとおりです:
1=SBCS
2=DBCS
3=MBCS

エンコードは次のとおりです:
1=EBCDIC
2 = ASCII
3 = ISO
4 = UCS-2
5 = UTF-8
6 = Euc
7 = GB18030
8 = UTF-16
9 = UTF-32

ファイルの編集時は、以下の点に留意してください:

- 行頭で # 記号を使用するとコメントが指定できます。 こうすることで、その行を IBM MQ が構文解析しなくなります。
- 行内にコメントは指定できません。
- ブランク行を作成してはなりません。
- ファイルの最後に新規項目を追加してはなりません。

  新規 CCSID 項目は ACRI テーブル情報の前に追加する必要があります。

デフォルトのデータ変換の指定

デフォルトの変換 CCSID を定義できます。2 つの CCSID 間でサポートされている変換がない場合に、これを使用して ASCII またはそれに類似したものと EBCDIC CCSID との間の変換を行います。

この機能を有効にすると、デフォルトの変換は送信やメッセージ・ヘッダーに使用され、ユーザー・データの変換にも使用できます。

以下のような 2 行を作成すると、デフォルトの変換が有効になります。

```
default       0      500     1     1    0
default       0      850     1     2    0
```

1 行目では、EBCDIC CCSID のデフォルトを 500 に設定しています。2 行目では ASCII および類似の CCSID のデフォルトを 850 に設定しています。

別のコマンド・レベルのデータの指定

IBM MQ の別のコマンド・レベルの CCSID 項目を指定するには、次のセクションを適用できる IBM MQ のコマンド・レベル (複数可) の前にコロン記号を使用します。

数値は、キュー・マネージャーまたはクライアントを実行すべき最小コマンド・レベルを表しています。例えば、現行のキュー・マネージャーがコマンド・レベル 900 で実行されていて、800 または 900 コマンド・レベル・フラグを検出した場合は、CCSID が読み取られます。

ただし、レベル 800 のキュー・マネージャーは 900 セクション内の CCSID は無視します。

指定されたコマンド・レベルは、コマンド・レベル・フラグの後から新しいコマンド・レベル・フラグが検出されるまで、検出されたすべての CCSID 項目に適用できます。

コマンド・レベルをすべてのコマンド・レベルに設定する必要がある場合は、数値 0 を指定します。

最初に `ccsid_part2.tbl` を構文解析する際、IBM MQ は、検出したすべての CCSID を IBM MQ のすべてのコマンド・レベルに有効として扱います。

バージョン管理は、IBM MQ が最初のコマンド・レベル・フラグを検出したときのみ使用が開始されます。

次のコード・スニペットは、バージョン管理の使用例です。

```
# Comment Block
# End of Comment Block
# Because no command level flag is specified and we're at the start of the file
# the following CCSIDs will be read on all versions
    819   819       0      819     1    3     0    ISO8859-1
    923   923       0      923     1    3     0    ISO8859-15
   1051  1051       0     1051     1    3     0    IBM-1051
# The colon :900 below shows that the CCSIDs after will only be for MQ cmd level 900 and above
:900
   8629   437       0      437     1    2     0    IBM-437
  12725   437       0      437     1    2     0    IBM-437
  16821   437       0      437     1    2     0    IBM-437
  20917   437       0      437     1    2     0    IBM-437
# The colon :0 below shows that the CCSIDs after will be for all version of MQ
:0
   4946   850       0      850     1    2     0    IBM-850
  33618   850       0      850     1    2     0    IBM-850
  61697   850       0      850     1    2     0    IBM-850
  61698   850       0      850     1    2     0    IBM-850
```

Managed File Transfer の管理

Managed File Transfer を管理するには、Managed File Transfer コマンドを使用します。 また、 IBM MQ Explorer を使用して管理用タスクの一部を行うこともできます。

エージェント・コマンド・キューにメッセージを入れて転送を開始する

ファイル転送メッセージをソース・エージェントのコマンド・キューに書き込むことにより、ファイル転送を開始することもできます。 例示コマンド・キュー名は SYSTEM.FTE.COMMAND.AGENT01 です。 正しいソース・エージェントのコマンド・キューにメッセージが届くようにする必要があります。XML のソース情報と一致しないエージェントによってメッセージが受け取られる場合、メッセージは拒否されます。

転送要求 XML は、 FileTransfer.xsd スキーマに準拠している必要があり、ルート・エレメントとして <request> エレメントを使用する必要があります。 転送要求メッセージの構造と内容に関する情報については、『ファイル転送要求メッセージ・フォーマット』を参照してください。 エージェントのコマンド・キューにどのように転送要求メッセージを 書き込むかは、タスクにより異なります。 例えば、IBM MQ Java API を使用して、メッセージをキューにプログラマチックに書き込むことができます。

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

276 ページの『メッセージからファイルへのデータ転送』
Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

287 ページの『プロトコル・ブリッジ』
プロトコル・ブリッジを使用すれば、Managed File Transfer (MFT) ネットワークから、MFT ネットワークの外部 (ローカル・ドメインとリモート・ロケーションの両方) にあるファイル・サーバーに格納されているファイルにアクセスできます。 このファイル・サーバーでは、FTP、FTPS、または SFTP ネットワーク・プロトコルを使用できます。 それぞれのファイル・サーバーで少なくとも 1 つの専用エージェントが必要です。 この専用エージェントは、 プロトコル・ブリッジ・エージェントとして知られています。 ブリッジ・エージェントは、複数のファイル・サーバーと相互作用できます。

327 ページの『IBM Integration Bus からの MFT の操作』
FTEOutput ノードと FTEInput ノードを使用して、IBM Integration Bus から Managed File Transfer を操作できます。

327 ページの『MFT のリカバリーと再始動』
エージェントまたはキュー・マネージャーが何らかの理由 (例えば、電源やネットワークの障害など) で使用できない場合、Managed File Transfer は、以下のシナリオで示すようにリカバリーを行います。

関連タスク

209 ページの『MFT エージェントの開始』
Managed File Transfer エージェントを使用してファイル転送を行うには、まずエージェントを開始する必要があります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

223 ページの『進行中のファイル転送のモニター』
IBM MQ Explorer の「**ファイル転送管理-現在の転送進行状況**」タブを使用して、進行中のファイル転送をモニターできます。 このファイル転送は、IBM MQ Explorer またはコマンド行のいずれかから開始できます。 このタブには、スケジュール済み転送が開始した時点でのスケジュール済み転送の進行も表示されます。

225 ページの『「転送ログ」のファイル転送の状況の表示』
IBM MQ Explorer の 「**転送ログ**」 を使用して、ファイル転送の詳細を表示できます。 対象にできるのは、コマンド行または IBM MQ Explorer のいずれかから開始された転送です。 また、「**転送ログ**」に表示される内容をカスタマイズすることもできます。

227 ページの『MFT リソースのモニター』
キューやディレクトリーなどの Managed File Transfer リソースをモニターできます。 そのリソースで条件が満たされると、リソース・モニターがファイル転送などのタスクを開始します。 IBM MQ Explorer 用

Managed File Transfer プラグインの **fteCreateMonitor** コマンドまたは 「**モニター**」 ビューを使用して、リソース・モニターを作成できます。

258 ページの『ファイル転送テンプレートの処理』
ファイル転送テンプレートを使用すると、繰り返しの転送または複雑な転送を行うための共通のファイル転送設定を保管できます。 転送テンプレートは **fteCreateTemplate** コマンドを使用してコマンド行から作成します。また、IBM MQ Explorer で、「**ファイル転送管理のテンプレート新規作成**」ウィザードを使用して転送テンプレートを作成することも、ファイル転送の作成時に「**転送設定をテンプレートとして保存する**」チェック・ボックスを選択してテンプレートを保存することもできます。 「**転送テンプレート**」ウィンドウには、Managed File Transfer ネットワーク内に作成した転送テンプレートがすべて表示されます。

214 ページの『MFT エージェントのリスト』
特定のキュー・マネージャーに登録された Managed File Transfer エージェントは、コマンド行または IBM MQ Explorer を使用してリストできます。

215 ページの『MFT エージェントの停止』
Managed File Transfer エージェントはコマンド行から停止できます。 エージェントを停止するときには、停止する前にエージェントを静止させて、エージェントが現行のファイル転送を完了するようにします。さらに、コマンド行で **-i** パラメーターを指定して、エージェントをただちに停止することもできます。 エージェントが停止してしまうと、再始動するまでそのエージェントを使用してファイルを転送することはできません。

MFT ロガーの構成

関連資料

ファイルの転送に関するガイドライン

MFT エージェントの開始

Managed File Transfer エージェントを使用してファイル転送を行うには、まずエージェントを開始する必要があります。

このタスクについて

Managed File Transfer Agent は、コマンド行から開始できます。 この場合、エージェント・プロセスはユーザーがシステムからログオフすると停止します。

ALW AIX, Linux, and Windows では、ユーザーがシステムからログオフしてもエージェントが実行を続行し、ファイル転送を受信し続けるようにエージェントを構成することができます。

z/OS z/OS では、対話式セッションがなくても、JCL から開始したタスクとしてエージェントを開始するようにエージェントを構成できます。

エージェントを実行中にリカバリー不能エラーが発生した場合、初期障害データ・キャプチャー (FDC) が生成され、エージェントは停止することにご注意ください。

手順

- コマンド行からエージェントを開始するには、**fteStartAgent** コマンドを使用します。
  細については、 **fteStartAgent** を参照してください。
- ALW
  システムからログオフしてもエージェントが実行を続行するように構成するには、次のようにします。
  - Windows Windows では、エージェントを Windows サービスとして実行するように構成します。詳しくは、210 ページの『Windows サービスとしての MFT エージェントの開始』を参照してください。
  - Linux AIX AIX and Linux では、リブート時にスクリプト・ファイルを使用してエージェントが自動的に開始するように構成します。 詳しくは、212 ページの『AIX and Linux システム始動時の MFT エージェントの開始』を参照してください。

- z/OS
  z/OS では、対話式セッションがなくても、JCL から開始したタスクとしてエージェントを開始するようにエージェントを構成します。
  詳細については、214 ページの『Starting an MFT agent on z/OS』を参照してください。

関連タスク

214 ページの『MFT エージェントのリスト』
特定のキュー・マネージャーに登録された Managed File Transfer エージェントは、コマンド行または IBM MQ Explorer を使用してリストできます。

215 ページの『MFT エージェントの停止』
Managed File Transfer エージェントはコマンド行から停止できます。 エージェントを停止するときには、停止する前にエージェントを静止させて、エージェントが現行のファイル転送を完了するようにします。 さらに、コマンド行で **-i** パラメーターを指定して、エージェントをただちに停止することもできます。 エージェントが停止してしまうと、再始動するまでそのエージェントを使用してファイルを転送することはできません。

関連資料

MFT エージェントの状況値
fteStartAgent

Windows Windows サービスとしての MFT エージェントの開始

エージェントを Windows サービスとして開始することにより、Windows からログオフしても、引き続きエージェントを実行し、ファイル転送を受け取ることができます。

このタスクについて

Windows 上のコマンド行からエージェントを開始すると、エージェント・プロセスは、Windows にログオンするために使用したユーザー名を使用して実行されます。 システムからログオフすると、エージェント・プロセスは停止します。 エージェントが停止しないようにするには、Windows サービスとして実行されるようにエージェントを構成することができます。 Windows サービスとして実行させることにより、エージェントを、Windows 環境の始動または再始動時に自動的に開始するように構成することもできます。

以下の手順に従って、Windows サービスとして実行するエージェントを開始します。 エージェントを Windows サービスとして実行するには、サポートされているいずれかの Windows バージョンで Managed File Transfer を実行する必要があります。 サポートされる環境のリストについては、「System Requirements for IBM MQ」を参照してください。

実際のステップは、既にエージェントを作成しているか、あるいはエージェントを作成中であるかによって異なります。 どちらのオプションも以下のステップで説明されています。

手順

1. Managed File Transfer エージェントを作成する場合は、 **fteCreateAgent**、 **fteCreateCDAgent**、または **fteCreateBridgeAgent** コマンドを使用します。 エージェントを Windows サービスとして実行するには、 **-s** パラメーターを指定します。 以下の例では、エージェント・キュー・マネージャー QMGR1 を含むエージェント AGENT1 が作成されます。 Windows サービスは、関連パスワード `ftepassword` を含む、ユーザー名 `fteuser` を使用して実行されます。

   ```
   fteCreateAgent -agentName AGENT1 -agentQMgr QMGR1 -s -su fteuser -sp ftepassword
   ```

   オプションで、**-s** パラメーターの後にサービスの名前を指定することができます。 名前を指定しなかった場合、サービスの名前は `mqmftAgent`*AGENTQMGR* となります。ここで、*AGENT* はユーザーが指定したエージェント名であり、*QMGR* はエージェント・キュー・マネージャー名です。 この例では、サービスのデフォルト名は `mqmftAgentAGENT1QMGR1` です。

注 : -su パラメーターを使用して指定する Windows ユーザー・アカウントには、 **Log on as a service** 権限が必要です。 これを構成する方法については、Windows サービスとして実行されている MFT エージェントまたはロガーのトラブルシューティングを参照してください。

詳しくは、 fteCreateAgent、 fteCreateCDAgent: Connect:Direct® ブリッジ・エージェントの作成、または fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)を参照してください。

2. 前のステップに従ってエージェントを作成した場合は、**fteCreateAgent**、**fteCreateCDAgent**、または **fteCreateBridgeAgent** コマンドによって生成された MQSC コマンドを実行します。 これらのコマンドは、エージェントが必要とする IBM MQ キューを作成します。

   例えば、エージェントの名前が *AGENT1*、エージェント・キュー・マネージャーの名前が *QMGR1*、および調整キュー・マネージャーの名前が *COORDQMGR1* の場合、以下のコマンドを実行します。

   ```
   runmqsc QMGR1 MQ_DATA_PATH\mqft\config\COORDQMGR1\agents\AGENT1\AGENT1_create.mqsc
   ```

3. 前のステップでエージェントを作成しておらず、代わりに既存のエージェントを Windows サービスとして実行するように構成する場合、エージェントが実行中であれば最初にエージェントを停止してから、その構成を変更します。

   a) 以下の例では、AGENT1 という名前のエージェントを使用します。 以下のコマンドを実行します。

   ```
   fteStopAgent AGENT1
   ```

   b) **fteModifyAgent** コマンドを使用して、Windows サービスとして実行されるようにエージェントを構成します。

   ```
   fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword
   ```

   詳しくは、fteModify エージェント: Windows サービスとしての MFT エージェントの実行を参照してください。

4. **fteStartAgent** コマンドを使用してエージェントを開始します。 代わりに、Windows デスクトップのスタート・メニューから選択した「コントロール パネル」の「管理ツール」から選択可能な Windows の「サービス」ツールを使用して、サービスを開始することもできます。

   ```
   fteStartAgent AGENT1
   ```

   Windows からログオフしても、サービスは引き続き実行されます。 Windows がシャットダウン後に再始動したときにもサービスが再始動するようにするために、 Windows サービス・ツールの 「**スタートアップの種類**」 フィールドはデフォルトで「**自動**」 に設定されています。 Windows の再始動時にサービスを再始動しない場合は、これを 「**手動**」 に変更します。

5. オプション: エージェントを停止するには、fteStopAgent コマンドを使用するか、あるいは Windows の「サービス」ツールを使用します。 例えば、コマンド行から、以下のコマンドを実行します。

   ```
   fteStopAgent AGENT1
   ```

   - **fteStopAgent** コマンドをサービスとして実行すると、このコマンドは **-i** パラメーターが指定されているかどうかに関わりなく、常にこのパラメーターを使用して実行されます。 **-i** パラメーターは、進行中の転送を完了せずにエージェントを即時停止します。 これは、Windows サービスの制限によるものです。

次のタスク

Windows サービスの開始に問題がある場合は、 Windows サービスとして実行されている MFT エージェントまたはロガーのトラブルシューティングを参照してください。 このトピックでは、Windows サービス・ログ・ファイルの場所についても説明します。

関連タスク

214 ページの『MFT エージェントのリスト』
特定のキュー・マネージャーに登録された Managed File Transfer エージェントは、コマンド行または IBM MQ Explorer を使用してリストできます。

215 ページの『MFT エージェントの停止』
Managed File Transfer エージェントはコマンド行から停止できます。 エージェントを停止するときには、停止する前にエージェントを静止させて、エージェントが現行のファイル転送を完了するようにします。 さらに、コマンド行で **-i** パラメーターを指定して、エージェントをただちに停止することもできます。 エージェントが停止してしまうと、再始動するまでそのエージェントを使用してファイルを転送することはできません。

関連資料

fteCreateAgent (MFT エージェントの作成)
fteCreateCDAgent (Connect:Direct ブリッジ・エージェントの作成)
fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)
fteModify エージェント (Windows サービスとしての MFT エージェントの実行)

関連情報

MFT `agent.properties` ファイル

Linux AIX AIX and Linux システム始動時の MFT エージェントの開始

Managed File Transfer Agent は、AIX and Linux のシステム始動時に開始するように構成できます。 ログオフしても、エージェントは引き続き実行され、ファイル転送を受け取ることができます。

fteCreateAgent、**fteCreateCDAgent**、または **fteCreateBridgeAgent** のいずれかの Managed File Transfer コマンドを使用してエージェントを作成および構成した場合、以下のコマンドを単に実行するスクリプト・ファイルを使用して、 AIX and Linux マシンでのリブート中に自動的に開始するようにエージェントを構成できます。

```
su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
```

mq_install_root は、必要な Managed File Transfer インストール済み環境のルート・ディレクトリーです。デフォルトは次のとおりです。 `/opt/mqm` および エージェント名 は、開始される Managed File Transfer Agent の名前です。 このスクリプト・ファイルの使用法は、具体的なオペレーティング・システムに応じて異なります。 例えば、Linux では追加のオプションを使用できます。

Linux

Linux

Linux システムの場合、システム・ブート・プロセス中にアプリケーションを開始する方法は複数あります。 一般に、以下の手順の実行を検討してください。

1. `/etc/rc.mqmft` という名前のファイルをコンテンツで作成します

   ```
   #!/bin/sh
   su -l mqmft_user"-c mq_install_root/bin/fteStartAgent agent_name"
   ```

   ここで、*mqmft_user* はエージェント・プロセスを実行するユーザー ID です。 このユーザー ID は mqm グループのメンバーである必要があります。
2. ファイル実行可能モジュールを作成します。例えば、次のようにします。

   ```
   chmod 755 /etc/rc.mqmft
   ```

3. 次に、以下の行を /etc/inittab に追加します。

```
mqmft:5:boot:/etc/rc.mqmft
```

Linux でのブート時にエージェントを開始するその他の方法では、 /etc/rc.d/rc.local ファイルにスクリプト行を追加するか、 Linux SuSe にスクリプト行を追加して、スクリプト行を /etc/init.d/boot.local ファイルに追加します。 ご使用の環境に最も適した方法を選択してください。 サポートされている特定の Linux ディストリビューションで始動時にエージェントを開始するその他の方法について、以下にさらに説明します。

SLES 10 および 11

SUSE Linux Enterprise Server (SLES) 10 および 11 システムの場合は、以下の手順を実行します。

1. システム・ルート・ユーザー ID として、独自の /etc/init.d/rc.rclocal ファイルを作成します。
2. 以下の行を rc.rclocal ファイルに追加します。

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: rc.rclocal
# Required-Start: $network $syslog
# Required-Stop: $network $syslog
# Default-Stop: 0 1 2 6
# Description: MQMFT agent startup
### END INIT INFO
su -l mqmft_user"-c mq_install_root/bin/fteStartAgent agent_name"
```

3. 以下のコマンドを実行します。

```
chmod 755 rc.rclocal

chkconfig --add rc.rclocal
```

systemd を使用した Linux での Managed File Transfer エージェントの開始

Linux

以下の手順を実行します。

1. /etc/systemd/ システム・フォルダー内にファイルを作成し、それに <agentname>.service のような名前を付けます。 以下の内容を追加します。ここで <agentname> は MFT_AGT_LNX_0 です。

```
# vi /etc/systemd/system/MFT_AGT_LNX_0.service
[Unit]
Description=IBM MQ MFT MFT_AGT_LNX_0
[Service]
ExecStart=/opt/mqm/bin/fteStartAgent MFT_AGT_LNX_0
ExecStop=/opt/mqm/bin/fteStopAgent MFT_AGT_LNX_0
Type=forking
User=mqm
Group=mqm
KillMode=none
```

2. サービスを有効にするには、以下のコマンドを実行します。

```
# systemctl enable MFT_AGT_LNX_0
# systemctl daemon-reload
```

3. エージェントを開始してその状況を確認するには、以下のコマンドを実行します。

```
# systemctl start MFT_AGT_LNX_0
# systemctl status MFT_AGT_LNX_0
```

関連タスク

215 ページの『MFT エージェントの停止』

Managed File Transfer エージェントはコマンド行から停止できます。 エージェントを停止するときには、停止する前にエージェントを静止させて、エージェントが現行のファイル転送を完了するようにします。 さらに、コマンド行で **-i** パラメーターを指定して、エージェントをただちに停止することもできます。 エージェントが停止してしまうと、再始動するまでそのエージェントを使用してファイルを転送することはできません。

関連資料

fteCreateAgent

fteCreateCDAgent: Connect:Direct ブリッジ・エージェントの作成

fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)

z/OS Starting an MFT agent on z/OS

On z/OS, in addition to running the **fteStartAgent** command from a z/OS UNIX System Services session, you can start an agent as a started task from JCL without the need for an interactive session.

A started task is used because it runs under a specific user ID and is not affected by users logging off.

Note: Started tasks are typically run under an administrative user that might not have log-on privileges and so it is not possible to log on to the z/OS system as the user that the agent is running under. The **fteStartAgent**, **fteStopAgent**, **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, cannot be issued for that agent.

You can use the agent property **adminGroup** with Managed File Transfer agents on z/OS. You can define a security manager group, for example MFTADMIN and then add the started task userid and administrator TSO ids to this group. Edit the agent properties file and set the **adminGroup** property to be the name of this security manager group.

```
adminGroup=MFTADMIN
```

Members of this group can then issue the **fteStartAgent**, **fteStopAgent**, and **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, for the agent that is running as a started task.

For more information, see the **adminGroup** property in The MFT `agent.properties` file.

As a Java application, an agent is a z/OS UNIX System Services application that you can run from JCL by using the BFGAGSTP member, from a generated Managed File Transfer command PDSE library data set for an agent. For more information about how to create an MFT command PDSE library data set, and customize it for the required agent, see Creating an MFT Agent or Logger command data set.

Related concepts

Enabling MFT agents to connect to remote z/OS queue managers

Related reference

“Stopping an MFT agent on z/OS” on page 215

If you are running a Managed File Transfer Agent on z/OS as a started task from JCL, the agent accepts the z/OS operator commands **MODIFY** and **STOP**, in addition to the **fteStopAgent** command.

The MFT `agent.properties` file

MFT エージェントのリスト

特定のキュー・マネージャーに登録された Managed File Transfer エージェントは、コマンド行または IBM MQ Explorer を使用してリストできます。

このタスクについて

コマンド行を使用してエージェントをリストするには、fteListAgents コマンドを参照してください。

IBM MQ Explorer を使用してエージェントをリストするには、「ナビゲーター」ビューで、調整キュー・マネージャー名の下の「**エージェント**」をクリックします。

エージェントが **fteListAgents** コマンドによってリストされないか、または IBM MQ Explorer で表示されない場合、MFT エージェントが **fteListAgents** コマンドによってリストされない場合に行う事柄のトピックの診断フローチャートを使用して問題を見つけ、修正してください。

関連資料

fteListAgents: 調整キュー・マネージャーの MFT エージェントのリスト

MFT エージェントの状況値

fteShowAgentDetails

MFT エージェントの停止

Managed File Transfer エージェントはコマンド行から停止できます。 エージェントを停止するときには、停止する前にエージェントを静止させて、エージェントが現行のファイル転送を完了するようにします。 さらに、コマンド行で **-i** パラメーターを指定して、エージェントをただちに停止することもできます。 エージェントが停止してしまうと、再始動するまでそのエージェントを使用してファイルを転送することはできません。

始める前に

キュー・マネージャーと関連付けられたエージェントの名前を確認する場合は、IBM MQ Explorer またはコマンド行を使用してエージェントをリストできます。これについては、fteListAgents コマンドを参照してください。

このタスクについて

コマンド行からエージェントを停止する場合は、fteStopAgent を参照してください。

fteStopAgent を使用してエージェントを制御された方法で停止した場合、エージェントは新しい管理対象転送要求を受け入れず、進行中の転送が完了するのを待ってから、実際にシャットダウンします。 IBM MQ 9.3.0 以降では、エージェントがまだ過渡状態にあり、そのためにまだシャットダウンされておらず、まだ再始動できないことを示すために、進行中の転送が完了するまでエージェントは STOPPING 状態になります。 この状況は、**fteListAgents** および **fteShowAgentDetails** コマンドの出力、MFT REST API 照会、および IBM MQ Explorer の MFT プラグインの 「**エージェント**」 ビューに表示されます。

Windows エージェントを Windows サービスとして実行するように構成した場合、**fteStopAgent** コマンドを実行すると、Windows サービスが停止します。 または、Windows の「サービス」ツールを使用してサービスを停止することによって、エージェントを停止できます。 詳しくは、210 ページの『Windows サービスとしての MFT エージェントの開始』のトピックを参照してください。

関連タスク

209 ページの『MFT エージェントの開始』
Managed File Transfer エージェントを使用してファイル転送を行うには、まずエージェントを開始する必要があります。

関連資料

MFT エージェントの状況値

fteStopAgent

215 ページの『Stopping an MFT agent on z/OS』
If you are running a Managed File Transfer Agent on z/OS as a started task from JCL, the agent accepts the z/OS operator commands **MODIFY** and **STOP**, in addition to the **fteStopAgent** command.

z/OS Stopping an MFT agent on z/OS

If you are running a Managed File Transfer Agent on z/OS as a started task from JCL, the agent accepts the z/OS operator commands **MODIFY** and **STOP**, in addition to the **fteStopAgent** command.

A started task is used because it runs under a specific user ID and is not affected by users logging off.

Note: Started tasks are typically run under an administrative user that might not have log-on privileges and so it is not possible to log on to the z/OS system as the user that the agent is running under. The **fteStartAgent**, **fteStopAgent**, **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, cannot be issued for that agent.

You can use the agent property **adminGroup** with Managed File Transfer agents on z/OS. You can define a security manager group, for example MFTADMIN and then add the started task userid and administrator TSO ids to this group. Edit the agent properties file and set the **adminGroup** property to be the name of this security manager group.

```
adminGroup=MFTADMIN
```

Members of this group can then issue the **fteStartAgent**, **fteStopAgent**, and **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, for the agent that is running as a started task.

For more information, see the **adminGroup** property in The MFT `agent.properties` file .

Controlled agent shutdown by using the z/OS MODIFY command (F)

The **MODIFY** command allows you to stop an agent in a controlled way as an alternative to the **fteStopAgent** command. The agent completes any transfers currently in progress but the agent does not start any new transfers.

For example:

```
F job_name,APPL=STOP
```

where *job_name* is the job that the agent process is running under.

Immediate agent shutdown by using the z/OS STOP command (P)

The **STOP** command is equivalent to an immediate stop by using the **fteStopAgent** command with the **-i** parameter. The agent is stopped immediately even if the agent is currently transferring a file.

For example:

```
P job_name
```

where *job_name* is the job that the agent process is running under.

Related concepts

Enabling MFT agents to connect to remote z/OS queue managers

Related reference

“Starting an MFT agent on z/OS” on page 214

On z/OS, in addition to running the **fteStartAgent** command from a z/OS UNIX System Services session, you can start an agent as a started task from JCL without the need for an interactive session.

The MFT `agent.properties` file

新規ファイル転送の開始

新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

このタスクについて

新規ファイル転送をコマンド行から開始するには、fteCreateTransfer コマンドを参照してください。

IBM MQ Explorer の 「**ファイル転送管理の新規作成**」 ウィザードを使用して新規ファイル転送を開始するには、以下のステップを実行します。

手順

1. 「ナビゲーター」ビューで、「**ファイル転送管理**」をクリックします。 「**ファイル転送管理 - メイン**」が「コンテンツ」ビューに表示されます。
2. すべての調整キュー・マネージャーが「ナビゲーター」ビューに表示されます。 転送に使用するエージェントの登録対象となる調整キュー・マネージャーの名前を展開します。 転送に使用するつもりのもの以外の調整キュー・マネージャーに現在接続している場合は、「ナビゲーター」ビューでその調整キュー・マネージャーの名前を右クリックして、「**切断**」をクリックします。 使用する調整キュー・マネージャーの名前を右クリックして、「**接続**」をクリックします。
3. 以下の方式のいずれかを使用して、「**ファイル転送管理の新規作成**」ウィザードを開始します。
   a) 「ナビゲーター」ビューで、関連した調整キュー・マネージャー、「**転送テンプレート**」、「**転送ログ**」、または「**保留中の転送**」のいずれかのノードの名前を右クリックします。 その後「**新規の転送**」をクリックしてウィザードを起動します。
   b) 「**ファイル**」 > 「**新規**」 > 「**その他**」 > 「**ファイル転送管理ウィザード**」 > 「**新規の転送ウィザード**」をクリックします。
4. ウィザード・パネルの指示に従います。 各パネルには、コンテキスト・ヘルプも提供されています。 Windows 上でコンテキスト・ヘルプにアクセスするには、F1 キーを押します。 Linux 上では、Ctrl+F1 キーまたは Shift+F1 キーを押します。

関連概念

217 ページの『転送定義ファイルの使用』
ファイル転送を作成するために使用できる転送定義ファイルを指定できます。 転送定義ファイルは、転送を作成するために必要な情報の一部またはすべてを定義した XML ファイルです。

関連タスク

220 ページの『スケジュール済みファイル転送の作成』
IBM MQ Explorer またはコマンド行のいずれかを使用して新規ファイル転送をスケジュールに入れられます。 スケジュール済みの転送には、単一のファイルまたは 1 つのグループの複数のファイルを含めることができます。 スケジュール済みファイル転送は、1 回実行することも複数回転送を繰り返すこともできます。

222 ページの『ファイル転送のトリガー』
転送を実行するために満たす必要がある特定のトリガー条件を、ファイル転送に対して設定できます。 トリガー条件が満たされない場合にはファイル転送は実行されず、転送が行われなかったことを記録するためのログ・メッセージがオプションで送信されます。 その後ファイル転送要求は廃棄されます。 例えば、ソース・エージェントがあるシステム上の指定ファイルが設定サイズを超えた場合のみ、 またはソース・エージェントがあるシステム上に特定の指定ファイルが存在する場合のみ ファイル転送が実行されるようにセットアップできます。 トリガー・ファイル転送は、IBM MQ Explorer かコマンド行のいずれかを使用してセットアップできます。

328 ページの『停止した転送のリカバリーに対するタイムアウトの設定』
停止したファイル転送について、ソース・エージェントのすべての転送に適用される転送リカバリー・タイムアウトを設定できます。 また、転送ごとに転送リカバリー・タイムアウトを設定することもできます。 停止したファイル転送のリカバリーをソース・エージェントで試行し続ける特定の期間を秒単位で設定した場合、転送が成功しないままエージェントがタイムアウトに達すると、転送は失敗します。

関連資料

fteCreateTransfer: 新規ファイル転送の開始

ファイル転送要求メッセージ・フォーマット

ファイルの転送に関するガイドライン

転送定義ファイルの使用

ファイル転送を作成するために使用できる転送定義ファイルを指定できます。 転送定義ファイルは、転送を作成するために必要な情報の一部またはすべてを定義した XML ファイルです。

転送定義ファイルは、複数のソース・ファイルと宛先ファイルを 1 つの転送操作に 指定する際に便利です。 転送定義ファイルを使用して、複雑なファイル転送を実行依頼できます。 転送定義ファイルを再利用したり共有したりすることも可能です。

転送定義ファイルには 2 つの形式を使用できますが、これらの形式はどちらも少し異なりますが、どちらも FileTransfer.xsd スキーマに準拠しています。 このスキーマは、Managed File Transfer インストール済み環境の samples\schema ディレクトリーで見つけることができます。

以下の 2 つのフォーマットの転送定義ファイルがサポートされています。

- 転送のソース・ファイルと宛先ファイルの定義。 この定義では、ルートとして **transferSpecifications** エレメントを使用します。
- 転送全体の定義。ソース・ファイルと宛先ファイル、ソース・エージェントと宛先エージェントを含みます。 この定義では、ルートとして **request** エレメントを使用します。
  - このフォーマットのファイルは、**fteCreateTransfer** コマンドの **-gt** パラメーターを使用して生成できます。

転送のソース・ファイルと宛先ファイルだけを指定する転送定義ファイル・フォーマットの例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<transferSpecifications xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <item checksumMethod="MD5" mode="text">
    <source recursive="false" disposition="leave">
      <file>textTransferTest.txt</file>
    </source>
    <destination type="directory" exist="overwrite">
      <file>c:\targetfiles</file>
    </destination>
  </item>
</transferSpecifications>
```

このフォーマットの転送定義ファイルを実行依頼する場合は、コマンド行でソース・エージェントと宛先エージェントを指定する必要があります。

```
fteCreateTransfer -sa AGENT1 -sm agent1qm -da AGENT2 -dm agent2qm -td
  c:\definitions\example1.xml
```

転送で必要なすべての情報を指定する転送定義ファイル・フォーマットの例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="3.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>fteuser</userID>
    </originator>
    <sourceAgent agent="AGENT1" QMgr="agent1qm"/>
    <destinationAgent agent="AGENT2" QMgr="agent2qm"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\*.jpg</file>
        </source>
        <destination type="directory" exist="error">
          <file>/targetfiles/images</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

fteCreateTransfer コマンドの **-gt** パラメーターを使用して、このフォーマットのファイルを生成できます。 このフォーマットの転送定義ファイルを実行依頼する場合は、コマンド行で他の情報を指定する必要はありません。

```
fteCreateTransfer -td c:\definitions\example2.xml
```

コマンド行でソース・エージェントと宛先エージェントの情報をオーバーライドすることもできます。その場合は、転送定義ファイルに加えて通常のパラメーターを渡します。 以下に例を示します。

```
fteCreateTransfer -da AGENT9 -dm agent9qm -td c:\definitions\example2.xml
```

この例では、コマンド行オプションを使用して、転送定義ファイル内で **AGENT9** として定義されている宛先エージェントと、転送定義ファイルで **agent9qm** として定義されている宛先キュー・マネージャーをオーバーライドします。

上記で説明したどちらのフォーマットでも、1 つ以上の <item> エレメントを使用できます。 <item> エレメントの詳細については、 ファイル転送要求のメッセージ形式を参照してください。 これらの転送項目には、 転送の動作を制御する追加属性を持つソース・ファイルと宛先ファイルのペアが それぞれ定義されます。 例えば、以下の動作を指定します。

- 転送はチェックサムを使用するかどうか
- 転送はテキストかバイナリーか
- 転送が完了した後にソース・ファイルを削除するかどうか
- ファイルが存在する場合、宛先ファイルを上書きするかどうか

転送定義ファイルを使用することの 1 つの利点は、コマンド行からは使用できない追加のオプションを指定できることです。 例えば、メッセージからファイルへの転送を行っているときに、転送定義ファイルを使用して groupId 属性を指定することができます。 この属性は、キューから読み取られるメッセージの IBM MQ グループ ID を指定します。 転送定義ファイルの別の利点は、ファイル・ペアごとに異なるオプションを指定できることです。 例えば、チェックサムを使用するかどうか、またファイルをテキスト・モードで転送するかバイナリー・モードで転送するかを、個々のファイルごとに指定することができます。 コマンド行を使用する場合には、転送に含まれる各ファイルに同じオプションが適用されます。

以下に例を示します。

```
<item checksumMethod="none" mode="binary">
  <source disposition="leave">
    <file>c:\sourcefiles\source1.doc</file>
  </source>
  <destination type="file" exist="error">
    <file>c:\destinationfiles\destination1.doc</file>
  </destination>
</item>

<item checksumMethod="MD5" mode="text">
  <source disposition="delete">
    <file>c:\sourcefiles\source2.txt</file>
  </source>
  <destination type="file" exist="overwrite">
    <file encoding="UTF8" EOL="CRLF">c:\destinationfiles\destination2.txt</file>
  </destination>
</item>

<item checksumMethod="none" mode="text">
  <source recursive="false" disposition="leave">
    <file>c:\originfiles\source3.txt</file>
  </source>
  <destination type="file" exist="overwrite">
    <file>c:\targetfiles\destination3.txt</file>
  </destination>
</item>
```

z/OS 分散システムから z/OS システムにファイルを転送するための項目も使用できます。

z/OS
```
<item checksumMethod="none" mode="text">
  <source recursive="false" disposition="leave">
    <file>textTransferTest.txt</file>
  </source>
  <destination type="dataset" exist="overwrite">
    <file encoding="IBM-1047">//TEXT.TRANS.TEST</file>
  </destination>
</item>
```

z/OS この例では、ソース・エージェントからのファイル `textTransferTest.txt` を、テキスト・モードの宛先エージェント上のデータ・セット `//TEXT.TRANS.TEST` に転送します。 この転送によって、ソース・データが、ソース・エージェントのデフォルトのエンコード (ソース・エンコード属性の指定なし) から、コード・ページ IBM-1047 に 変換されます。

スケジュール済みファイル転送の作成

IBM MQ Explorer またはコマンド行のいずれかを使用して新規ファイル転送をスケジュールに入れられます。 スケジュール済みの転送には、単一のファイルまたは 1 つのグループの複数のファイルを含めることができます。 スケジュール済みファイル転送は、1 回実行することも複数回転送を繰り返すこともできます。

このタスクについて

ファイル転送スケジュールは、1 回行うようにセットアップすることもできますし、次の間隔で行うようにセットアップすることもできます。

- 毎分
- 毎時
- 毎日
- 毎週
- 毎月
- 毎年

その後、次の時点でスケジュールの発生を停止するように指定できます。

- 定義された日時
- 定義された発生回数の後

また、期限なくスケジュールの発生が継続するように指定することもできます。

スケジュールされた転送が毎日同じ時刻に実行される場合は、エージェント・プロパティー・ファイルの **adjustScheduleTimeForDaylightSaving** 属性を使用して、クロックが変更されたときにスケジュールが実行される時刻を調整します。 詳しくは、MFTagent.properties ファイルを参照してください。

コマンド・ラインを使用してスケジュール済みファイル転送を新規作成するには、 fteCreateTransfer コマンドのスケジューリング・パラメーター (**-tb**、 **-ss**、 **-oi**、 **-of**、 **-oc**、および **-es**) を使用します。

IBM MQ Explorer の 「**ファイル転送管理の新規作成**」 ウィザードを使用してスケジュール済みファイル転送を新規作成するには、以下のステップを実行します。

手順

1. 「ナビゲーター」ビューで、「**ファイル転送管理**」をクリックします。 「**ファイル転送管理 - メイン**」が「コンテンツ」ビューに表示されます。
2. すべての調整キュー・マネージャーが「ナビゲーター」ビューに表示されます。 転送に使用するエージェントの登録対象となる調整キュー・マネージャーの名前を展開します。 転送に使用するつもりのもの以外の調整キュー・マネージャーに現在接続している場合は、「ナビゲーター」ビューでその調整キュ

ー・マネージャーの名前を右クリックして、「**切断**」をクリックします。 使用する調整キュー・マネージャーの名前を右クリックして、「**接続**」をクリックします。

3. 以下の方式のいずれかを使用して、「**ファイル転送管理の新規作成**」ウィザードを開始します。
   a)「ナビゲーター」ビューで、関連した調整キュー・マネージャー、「**転送テンプレート**」、「**転送ログ**」、または「**保留中の転送**」のいずれかのノードの名前を右クリックします。 その後「**新規の転送**」をクリックしてウィザードを起動します。
   b)「**ファイル**」 > 「**新規**」 > 「**その他**」 > 「**ファイル転送管理ウィザード**」 > 「**新規の転送ウィザード**」をクリックします。
4. ウィザード・パネルの指示に従います。 「**スケジュール転送を有効にする**」チェック・ボックスを選択してあることを確認し、「**スケジュール**」タブにスケジュールの詳細を入力します。 スケジュール済みファイル転送は、転送に影響する可能性がある問題がなければ、スケジュール開始時刻から 1 分以内に開始します。 例えば、スケジュール済み転送の開始を妨げるネットワークまたはエージェントの問題があるかもしれません。 各パネルにはコンテキスト・ヘルプがあります。 Windows 上でコンテキスト・ヘルプにアクセスするには、F1 キーを押します。 Linux 上では、Ctrl+F1 キーまたは Shift+F1 キーを押します。

タスクの結果

スケジュール済みファイル転送に含まれるメッセージの詳細については、 『スケジュール済みファイル転送ログ・メッセージ・フォーマット』を参照してください。

保留中のファイル転送の処理

IBM MQ Explorer から、保留中のスケジュール済みファイル転送を表示することができます。 「**保留中の転送**」ウィンドウには、現在接続している調整キュー・マネージャーに登録されている保留中の転送がすべて表示されます。

このタスクについて

まだ開始されていないスケジュール済みファイル転送の状況を表示するには、以下のステップを実行します。

手順

1.「ナビゲーター」ビューで「**ファイル転送管理**」を展開します。 「**ファイル転送管理 - メイン**」が「コンテンツ」ビューに表示されます。
2. すべての調整キュー・マネージャーが「ナビゲーター」ビューに表示されます。 スケジュール済みの転送に使用した調整キュー・マネージャーの名前を展開します。 接続先の調整キュー・マネージャーを変更する場合は、「ナビゲーター」ビューで使用する調整キュー・マネージャーの名前を右クリックして、「**接続**」をクリックします。
3.「**保留中の転送**」をクリックします。 「**保留中の転送**」ウィンドウが「コンテンツ」ビューに表示されます。
4.「**保留中の転送**」ウィンドウに、スケジュール済みのファイル転送に関する以下の詳細が表示されます。
   a)「**名前**」。スケジュール済みファイル転送の番号。 この番号は自動的に割り当てられます。
   b)「**ソース**」。ソース・エージェントの名前。
   c)「**ソース・ファイル**」。ホスト・システムにおける、転送するファイルの名前。
   d)「**宛先**」。宛先エージェントの名前。
   e)「**宛先ファイル**」。宛先システムに転送された後のファイルの名前。
   f)「**スケジュール済みの開始時刻 (選択したタイム・ゾーン)**」。ファイル転送を開始するようスケジュールされた、管理者が選択したタイム・ゾーンでの時刻と日付。 表示されるタイム・ゾーンを変更するには、 **ウィンドウ > 設定 > IBM MQ Explorer > ファイル転送管理** をクリックし、 **タイム・ゾーン:** リストから別のタイム・ゾーンを選択します。 「**OK**」をクリックします。
   g)「**繰り返し周期**」。スケジュール済み転送を繰り返す選択をした場合、転送を繰り返す指定間隔。数値で表示されます。

h)「**繰り返しのタイプ**」。スケジュール済み転送を繰り返す選択をした場合、ファイル転送のために指定した繰り返しの間隔のタイプ。 タイプは、次のいずれかの値になります。分、時間、日、週、月、または年。

i)「**繰り返し期限**」。スケジュール済み転送を繰り返す選択をした場合、ファイル転送の繰り返しを停止する時間の詳細。 例えば、指定した日時、指定した発生回数の後など。

タスクの結果

「**保留中の転送**」ウィンドウに表示されている内容を最新表示するには、「コンテンツ」ビューのツールバーにある「リフレッシュ」ボタン をクリックします。

保留中のファイル転送を取り消すには、特定の転送を右クリックし、「**キャンセル**」をクリックします。 転送を取り消すと、ファイル転送要求が完全に廃棄されます。

ファイル転送のトリガー

転送を実行するために満たす必要がある特定のトリガー条件を、ファイル転送に対して設定できます。 トリガー条件が満たされない場合にはファイル転送は実行されず、転送が行われなかったことを記録するためのログ・メッセージがオプションで送信されます。 その後ファイル転送要求は廃棄されます。 例えば、ソース・エージェントがあるシステム上の指定ファイルが設定サイズを超えた場合のみ、 またはソース・エージェントがあるシステム上に特定の指定ファイルが存在する場合のみ ファイル転送が実行されるようにセットアップできます。 トリガー・ファイル転送は、IBM MQ Explorer かコマンド行のいずれかを使用してセットアップできます。

このタスクについて

リソースを継続的にモニターして、トリガー条件が満たされるかどうかを判断することができます。 リソース・モニターの詳細については、 227 ページの『MFT リソースのモニター』を参照してください。

設定できるトリガー条件は 3 種類あります。 条件は以下のとおりです。

- ソース・エージェントと同じシステムに特定のファイルが存在する場合
- ソース・エージェントと同じシステムに特定のファイルが存在しない場合
- ソース・エージェントがあるシステム上の特定のファイルが特定のサイズを 超えている (サイズはバイト、KB、MB、または GB で指定できます) 場合。 これらの単位では、2^{10} 規則を使用します。例えば 1 KB は 1024 バイトを示し、1 MB は 1024 KB を示します。

上記のリストにあるトリガー・タイプは、次の 2 つの方法で 結合できます。

- 単一の条件では、ソース・エージェントがあるシステム上の複数のファイルを指定できます。 この場合、指定したいずれかのファイルが条件を満たした場合に (ブール演算子 OR) 転送がトリガーされます。
- 複数の条件を指定できます。 この場合、条件すべてが満たされた場合のみ (ブール演算子 AND) 転送はトリガーされます。

トリガー転送をスケジュール済み転送と結合させることもできます。 詳しくは、スケジュール済みファイル転送の作成を参照してください。 この場合、トリガー条件はスケジュールが開始する時点で評価されます。繰り返しスケジュールの場合には、スケジュールが開始する時点ごとに評価されます。

トリガー転送は、プロトコル・ブリッジ・エージェントではサポートされません。

コマンドラインを使用してトリガーしたファイル転送を作成するには、 fteCreateTransfer コマンドで **-tr** パラメーターを使用します。

IBM MQ Explorer の 「**ファイル転送管理の新規作成**」 ウィザードを使用して、スケジュールされたファイル転送を作成するには、以下のステップを実行します。

手順

1.「ナビゲーター」ビューで、「**ファイル転送管理**」をクリックします。 「**ファイル転送管理 - メイン**」が「コンテンツ」ビューに表示されます。

2. すべての調整キュー・マネージャーが「ナビゲーター」ビューに表示されます。 スケジュール済みの転送に使用した調整キュー・マネージャーの名前を展開します。 接続先の調整キュー・マネージャーを変更する場合は、「ナビゲーター」ビューで使用する調整キュー・マネージャーの名前を右クリックして、「**接続**」をクリックします。
3. 以下の方式のいずれかを使用して、「**ファイル転送管理の新規作成**」ウィザードを開始します。
   a)「ナビゲーター」ビューで、関連した調整キュー・マネージャー、「**転送テンプレート**」、「**転送ログ**」、または「**保留中の転送**」のいずれかのノードの名前を右クリックします。 その後「**新規の転送**」をクリックしてウィザードを開きます。
   b)「**ファイル**」 > 「**新規**」 > 「**その他**」 > 「**ファイル転送管理ウィザード**」 > 「**新規の転送ウィザード**」をクリックします。
4. ウィザード・パネルの指示に従います。 「**トリガー**」タブの「**トリガー転送を有効にする**」チェック・ボックスが選択されていることを確認し、そのタブにあるフィールドすべてに入力してトリガーをセットアップします。 各パネルにはコンテキスト・ヘルプがあります。 Windows 上でコンテキスト・ヘルプにアクセスするには、F1 キーを押します。 Linux 上では、**Ctrl+F1** キーまたは Shift+F1 キーを押します。

進行中のファイル転送のモニター

IBM MQ Explorer の 「**ファイル転送管理-現在の転送進行状況**」 タブを使用して、進行中のファイル転送をモニターできます。 このファイル転送は、IBM MQ Explorer またはコマンド行のいずれかから開始できます。 このタブには、スケジュール済み転送が開始した時点でのスケジュール済み転送の進行も表示されます。

このタスクについて

IBM MQ Explorer を使用してリモート・システムの調整キュー・マネージャーに関連する転送をモニターする場合は、224 ページの『リモート調整キュー・マネージャーをモニターするための IBM MQ Explorer の構成』 のトピックにある手順に従ってください。

直前のファイル転送情報は、IBM MQ Explorer を停止して再始動した後は保持されません。 再始動すると、過去の転送に関する情報は「**現在の転送進行状況**」タブから消去されます。 IBM MQ Explorer が開いているときはいつでも、 「**完了した転送を削除**」 を使用して、完了した転送を消去できます。

手順

IBM MQ Explorer またはコマンド行を使用して新規のファイル転送を開始した後、「**現在の転送進行状況**」タブで転送の進行をモニターできます。 進行中の各転送について、以下の情報が表示されます。

a) **ソース**. ソース・システムからファイルを転送するために使用するエージェントの名前。

b) **宛先**。 宛先システムでファイルを受け取るために使用するエージェントの名前。

c)「**現在のファイル**」。 現在転送中のファイルの名前。 既に転送されている個々のファイルの部分は、 B、KiB、 MiB に表示されます。 GiB または TiB とともに、ファイルの合計サイズが括弧内にあります。 表示される単位はファイルのサイズによって異なります。

B は 1 秒あたりのバイト数を示します。 KiB/s は 1 秒あたりのキビバイト数を示します (1 キビバイトは 1024 バイト)。 MiB/s は 1 秒あたりのメビバイト数を示します (1 メビバイトは 1 048 576 バイト)。 GiB/s は 1 秒あたりのギビバイト数を示します (1 ギビバイトは 1 073 741 824 バイト)。 TiB/s は 1 秒あたりのテビバイト数を示します (1 テビバイトは 1 099 511 627 776 バイト)。

d)「**ファイル数**」。 複数のファイルを転送している場合、この数は、ファイルのグループ全体を通して現在の転送がどの程度進んだかを示します。

e)「**進行状況**」。 進行状況表示バーには、現在のファイル転送の完了率 (パーセント) が示されます。

f)「**転送速度**」。 ファイルが転送される速度。KiB/s 単位 (1 秒あたりのキビバイト数。1 キビバイトは 1024 バイト)。

g)「**開始 (選択したタイム・ゾーン)**」。 ファイル転送が開始された時刻。管理者が選択したタイム・ゾーンで表示されます。 表示されるタイム・ゾーンを変更するには、**ウィンドウ > 設定 > IBM MQ Explorer >**

ファイル転送管理 をクリックし、**タイム・ゾーン:** リストから別のタイム・ゾーンを選択します。「**OK**」をクリックします。

ファイルの転送中に転送がリカバリー状態に入ると、開始された時刻は更新され、ファイル転送が再開された時刻を反映します。

タスクの結果

このタブの情報は定期的に自動的に最新表示されますが、「**現在の転送進行状況**」タブに表示されている内容を強制的に最新表示するには、「コンテンツ」ビューのツールバーにある「**リフレッシュ**」 をクリックします。

「**現在の転送進行状況**」タブからファイル転送を削除するには、「コンテンツ」ビューのツールバーにある「**完了した転送を削除**」 をクリックします。 このボタンをクリックしても、ファイル転送の詳細がタブから削除されるだけです。現行のまたはスケジュール済みの転送は停止またはキャンセルされません。

「**現在の転送進行状況**」タブを閉じた後にそこに戻る場合は、「**ウィンドウ**」 > 「**ビューの表示**」 > 「**その他**」 > 「**その他**」 > 「**ファイル転送管理 - 現在の転送進行状況**」をクリックすることで、タブを表示できます。 「**OK**」をクリックします。

次のタスク

さらに、カスタム・ファイル転送モニター用のアプリケーションを開発することも可能です。 そのためには、対象の Managed File Transfer 管理トピックのサブスクリプションをプログラマチックに作成するか、管理方式で作成します。そうすれば、モニター・アプリケーションでそのトピックの Managed File Transfer ファイル転送アクティビティー・パブリケーションを受け取れるようになります。 サブスクリプション・トピックとパブリケーション・メッセージのフォーマットの詳細については、ファイル転送進行メッセージの例を参照してください。

関連タスク

224 ページの『リモート調整キュー・マネージャーをモニターするための IBM MQ Explorer の構成』
リモート・システムで実行中の調整キュー・マネージャーに関連するファイル転送をモニターするには、IBM MQ Explorer を使用します。 IBM MQ Explorer を実行することができるシステムが必要です。 リモート調整キュー・マネージャーに接続できるように IBM MQ Explorer コンポーネントをインストールする必要があります。

225 ページの『「転送ログ」のファイル転送の状況の表示』
IBM MQ Explorer の 「**転送ログ**」 を使用して、ファイル転送の詳細を表示できます。 対象にできるのは、コマンド行または IBM MQ Explorer のいずれかから開始された転送です。 また、「**転送ログ**」に表示される内容をカスタマイズすることもできます。

リモート調整キュー・マネージャーをモニターするための IBM MQ Explorer の構成

リモート・システムで実行中の調整キュー・マネージャーに関連するファイル転送をモニターするには、IBM MQ Explorer を使用します。 IBM MQ Explorer を実行することができるシステムが必要です。 リモート調整キュー・マネージャーに接続できるように IBM MQ Explorer コンポーネントをインストールする必要があります。

このタスクについて

前提: リモート調整キュー・マネージャーに接続する権限があること。それには、リモート接続を許可するようにキュー・マネージャーを構成します。

この構成方法について詳しくは、クライアント・モードでチャネル認証を使用してキュー・マネージャーに接続する操作、および MFT 固有リソースの権限の管理を参照してください。

Windows または Linux を実行していないシステム上のエージェント間でキュー・マネージャーとファイル転送をモニターするには、以下のステップを使用して、 IBM MQ Explorer をリモート・システムに接続するように構成します。

手順

1. IBM MQ Explorer を開始します。
2. IBM MQ Explorer がロードされたら、「**ファイル転送管理**」フォルダーを右クリックして「**新規構成**」を選択します。
3. ウィザードに従って調整およびコマンド・キュー・マネージャーを選択し、次に構成の名前を定義します。
4. 「**完了**」をクリックして定義を完了します。
5. 定義を完了したら、その定義を右クリックして「**接続**」を選択します。

タスクの結果

これで、IBM MQ Explorer を開始して、調整キュー・マネージャーに関連する Managed File Transfer ネットワークの転送アクティビティーをモニターするために使用できます。

関連タスク

223 ページの『進行中のファイル転送のモニター』
IBM MQ Explorer の 「**ファイル転送管理-現在の転送進行状況**」 タブを使用して、進行中のファイル転送をモニターできます。 このファイル転送は、IBM MQ Explorer またはコマンド行のいずれかから開始できます。 このタブには、スケジュール済み転送が開始した時点でのスケジュール済み転送の進行も表示されます。

225 ページの『「転送ログ」のファイル転送の状況の表示』
IBM MQ Explorer の 「**転送ログ**」 を使用して、ファイル転送の詳細を表示できます。 対象にできるのは、コマンド行または IBM MQ Explorer のいずれかから開始された転送です。 また、「**転送ログ**」に表示される内容をカスタマイズすることもできます。

「転送ログ」のファイル転送の状況の表示

IBM MQ Explorer の 「**転送ログ**」 を使用して、ファイル転送の詳細を表示できます。 対象にできるのは、コマンド行または IBM MQ Explorer のいずれかから開始された転送です。 また、「**転送ログ**」に表示される内容をカスタマイズすることもできます。

手順

1. 「ナビゲーター」ビューで「**ファイル転送管理**」を展開して、転送ログを表示する調整キュー・マネージャーの名前を展開します。
2. 「ナビゲーター」ビューで「**転送ログ**」をクリックします。 「**転送ログ**」が「コンテンツ」ビューに表示されます。
3. 「**転送ログ**」ウィンドウに、ファイル転送に関する以下の詳細が表示されます。
   a) 「**ソース**」。ソース・ファイルが格納されているシステム上のエージェントの名前。
   b) 「**宛先**」。ファイルの転送先となるシステム上のエージェントの名前。
   c) 「**完了状態**」。ファイル転送の状況。 状態は、「開始」、「進行中」、「成功」、「一部成功」、「取り消し済み」、または「失敗」のいずれかの値です。
   d) 「**所有者**」。転送要求を実行依頼したホストでのユーザー ID。
   e) 「**開始 (選択したタイム・ゾーン)**」。Managed File Transfer エージェントによってファイル転送要求が受け入れられた時刻と日付。管理者が選択したタイム・ゾーンで表示されます。 表示されるタイム・ゾーンを変更するには、**ウィンドウ > 設定 > IBM MQ Explorer > Managed File Transfer** をクリックし、 **タイム・ゾーン:** リストから別のタイム・ゾーンを選択します。 「**OK**」をクリックします。
   f) 「**状態の記録日時 (選択したタイム・ゾーン)**」 (この列はデフォルトでは表示されません。 「**転送ログの列の構成**」  ウィンドウを使用して、この列を表示するように選択できます)。 完了状態が記録された、管理者が選択したタイム・ゾーンでの 時刻と日付。
   g) **ジョブ名** ユーザーが **fteCreateTransfer** の **-jn** パラメーターを使用して、または Ant スクリプトで指定した ID。

h)「**転送 ID**」。ファイル転送のための固有 ID。

i)「**Connect: Direct**」。「**プロセス番号**」、「**プロセス名**」、「**1 次ノード**」、「**2 次ノード**」、「**ソース・タイプ**」、および「**宛先タイプ**」の詳細がリストされます。

タスクの結果

注 : 転送ログの内部形式は、APAR IC99545 対応の IBM MQ 8.0.0 Fix Pack 1 で変更されています。 そのため、IBM MQ Explorer を V8.0.0.1 以降にアップグレードした後に V8.0.0.0 に復元した場合は、IBM MQ Explorer が V8.0.0.1 であったときに行われた転送に関する監査 XML は表示されません。 これらの転送の「**プロパティー**」ウィンドウの XML パネルには、空のテキスト・ボックスが表示されます。

完了した転送に関する詳細を表示するには、正符号 (+)をクリックすることにより、関心のある転送を展開します。 その後、その転送に含まれているすべてのソース・ファイル名と宛先ファイル名を表示できます。ただし、多数のファイルから成る転送が現在進行中の場合には、 これまでに既に転送されたファイルのみを表示できます。

「**転送ログ**」に表示されている内容を最新表示するには、「コンテンツ」ビューのツールバーにある「**リフレッシュ**」ボタン をクリックします。 「転送ログ」内のファイル転送情報は、IBM MQ Explorer の停止と再始動を行うまでログの中に残ります。 完了したファイル転送をすべてログから削除する場合は、「コンテンツ」ビューのツールバーにある「**完了した転送を削除**」 をクリックします。

完了した個別のファイル転送をログから削除するには、転送を右クリックし、「**削除**」をクリックします。転送を削除しても、進行中またはスケジュール済みの転送は停止または取り消されることはありません。保管された履歴データのみが削除されます。

転送の固有 ID をクリップボードにコピーするには、 その転送項目を右クリックしてから「**ID のコピー** 」をクリックします。

転送のメタデータおよび完全な監査 XML は、「**プロパティー**」アクションの下のポップアップ・メニューから入手できます。

関連タスク

223 ページの『進行中のファイル転送のモニター』
IBM MQ Explorer の 「**ファイル転送管理-現在の転送進行状況**」 タブを使用して、進行中のファイル転送をモニターできます。 このファイル転送は、IBM MQ Explorer またはコマンド行のいずれかから開始できます。 このタブには、スケジュール済み転送が開始した時点でのスケジュール済み転送の進行も表示されます。

226 ページの『転送ログの構成』
IBM MQ Explorer の 「**転送ログ**」 に表示される情報とその表示方法を構成できます。

328 ページの『停止した転送のリカバリーに対するタイムアウトの設定』
停止したファイル転送について、ソース・エージェントのすべての転送に適用される転送リカバリー・タイムアウトを設定できます。 また、転送ごとに転送リカバリー・タイムアウトを設定することもできます。停止したファイル転送のリカバリーをソース・エージェントで試行し続ける特定の期間を秒単位で設定した場合、転送が成功しないままエージェントがタイムアウトに達すると、転送は失敗します。

転送ログの構成

IBM MQ Explorer の 「**転送ログ**」 に表示される情報とその表示方法を構成できます。

このタスクについて

「**転送ログ**」の列の順序を再配置するには、移動する列のタイトルをクリックし、その列を新しい位置にドラッグします。 列の新しい順序は、次に IBM MQ Explorer を停止して再始動するまでしか保持されません。

「**転送ログ**」の項目をフィルター操作するには、「**表示するログ項目のフィルタリング**」フィールドにストリングを入力します。 すべての項目をログに復元するには、フィールドに入力したストリングを削除します。 このフィールドでは、有効な任意の Java 正規表現を使用できます。 詳しくは、MFT が使用する正規表現を参照してください。

転送ログに表示される列をカスタマイズするには、「**転送ログの列の構成**」  を使用します。 以下のステップを使用して「**転送ログの列の構成**」ウィンドウを開始して使用します。

手順

1. 「コンテンツ」ビューで「**転送ログ**」が開いていることを確認します。 「コンテンツ」ビュー・ツールバーで「**転送ログの列の構成**」  をクリックします。 「**転送ログの列の構成**」ウィンドウが開きます。
2. 「**転送ログ**」の表示をカスタマイズするには、表示または非表示にする列の各チェック・ボックスを選択またはクリアします。 「**すべて選択**」をクリックして「**OK**」をクリックすると、すべてのチェック・ボックスを選択できます。「**すべて選択解除**」をクリックして「**OK**」をクリックすると、すべてのチェック・ボックスをクリアできます。

関連タスク

223 ページの『進行中のファイル転送のモニター』
IBM MQ Explorer の 「**ファイル転送管理-現在の転送進行状況**」 タブを使用して、進行中のファイル転送をモニターできます。 このファイル転送は、IBM MQ Explorer またはコマンド行のいずれかから開始できます。 このタブには、スケジュール済み転送が開始した時点でのスケジュール済み転送の進行も表示されます。

225 ページの『「転送ログ」のファイル転送の状況の表示』
IBM MQ Explorer の 「**転送ログ**」 を使用して、ファイル転送の詳細を表示できます。 対象にできるのは、コマンド行または IBM MQ Explorer のいずれかから開始された転送です。 また、「**転送ログ**」に表示される内容をカスタマイズすることもできます。

MFT リソースのモニター

キューやディレクトリーなどの Managed File Transfer リソースをモニターできます。 そのリソースで条件が満たされると、リソース・モニターがファイル転送などのタスクを開始します。 IBM MQ Explorer 用 Managed File Transfer プラグインの **fteCreateMonitor** コマンドまたは 「**モニター**」 ビューを使用して、リソース・モニターを作成できます。

このタスクについて

Managed File Transfer リソース・モニターは、以下の用語を使用します。

リソース・モニター
リソース・モニターとは、事前定義された一定間隔でリソース (ディレクトリーやキューなど) をポーリングし、リソースのコンテンツが変更されたかどうかを確認するプロセスです。 変更されている場合、コンテンツはそのモニターの条件セットと比較されます。 条件が一致する場合、このモニター用のタスクが開始されます。

リソース
リソース・モニターがポーリング間隔で検査してトリガー条件と比較するシステム・リソース。 キュー、ディレクトリーあるいはネストされたディレクトリー構造をモニター対象のリソースにすることができます。

条件とトリガー条件
条件とは、評価される式です (通常、モニター対象リソースのコンテンツと比較して評価されます)。 式の評価の結果が真であると、その条件はトリガーの全体条件に与えられます。

トリガー条件とは、すべての条件が満たされたときに満たされる総合的な条件です。 トリガー条件が満たされると、タスクは処理可能になります。

タスク
タスクとは、トリガー条件または条件のセットが満たされたときに開始される操作です。 サポートされるタスクは、ファイル転送とコマンド呼び出しです。

トリガー・ファイル
トリガー・ファイルとは、タスク (通常は転送) が開始できることを示す、モニター対象ディレクトリーに置かれるファイルです。 例えば、処理されるすべてのファイルが、既知の場所に到着し、転送が可能であること、あるいは別の場合には処理が可能であることを示します。 トリガー・ファイルの名前に基づいて、変数置換によって転送対象のファイルを指定することも可能です。 詳細については、239

ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』を参照してください。

トリガー・ファイルは、ready ファイルまたは go ファイルとも呼ばれます。 ただし、この資料では通常、トリガー・ファイルと呼んでいます。

リソース・モニターは、プロトコル・ブリッジ・エージェントまたは Connect:Direct ブリッジ・エージェントではサポートされません。

関連概念

エージェントの過負荷を回避するように MFT リソース・モニターを構成するためのガイダンス

関連資料

fteCreateMonitor: MFT リソース・モニターの作成

fteListMonitors: MFT リソース・モニターのリスト

fteDeleteMonitor: MFT リソース・モニターの削除

MFT モニター要求メッセージ・フォーマット

MFT のリソース・モニターの概念

Managed File Transfer のリソース・モニター機能の主要概念の概要。

リソース・モニター

リソース・モニターを作成するには、 **fteCreateMonitor** コマンドを使用します。このコマンドは、コマンド行から新規リソース・モニターを作成して開始します。 リソース・モニターは、Managed File Transfer エージェントに関連付けられて、エージェントが実行されるときにのみアクティブになります。 モニター中のエージェントが停止すると、リソース・モニターも停止します。 リソース・モニターが作成されたときにエージェントが既に実行している場合、リソース・モニターは即時開始されます。 モニター・エージェントは、リソース・モニターにより開始されるタスクのソース・エージェントでもある必要があります。

リソース・モニター名は、そのエージェント内で固有である必要があります。 リソース・モニター名は、1 文字以上の長さでなければならず、アスタリスク (*)、パーセント (%)、疑問符 (?) の文字は使用できません。 リソース・モニター名の大/小文字の指定は無視され、すべて大文字に変換されます。 既に存在する名前のリソース・モニターを作成しようとすると、その要求は無視されて、リソース・モニター・ログのトピックにその試みが記録されます。

注 : スケジュールされた転送を含むタスク定義を使用してリソース・モニターを作成することはできません。

IBM MQ 9.3.0 より前では、リソース・モニターを停止する唯一の方法は、モニター操作を実行しているエージェントを停止することです。 リソース・モニターを再始動するには、一緒にエージェントを再始動する必要があります。 IBM MQ 9.3.0 以降、エージェントを停止または再始動することなく、リソース・モニターを開始および停止することができます。 詳しくは、231 ページの『リソース・モニターの開始および停止』を参照してください。

エージェントに作成できるリソース・モニターの数に制限はなく、すべてのモニターは同じ優先度で実行されます。 モニター対象リソースのオーバーラップ、トリガー条件の矛盾、およびリソースをポーリングする頻度の影響を考慮してください。

リソース・モニターのオーバーラップがあると、以下の状態が発生する場合があります。

- ソースとなるロケーション/項目で競合が発生する可能性がある。
- 同じソース項目に対して重複した転送要求が発生する可能性がある。
- ソース項目の競合が原因で、転送において予期しないエラーや障害が発生する。

同じロケーションを複数のモニターがスキャンし、同じ項目を対象としてトリガーする可能性がある場合には、2 つの異なるモニターがその同じ項目に対する管理対象転送要求を実行依頼するという問題が発生する可能性があります。

リソース・モニターは、各ポーリング間隔の時間が過ぎると、リソースのコンテンツを調べます。 リソースのコンテンツは、トリガー条件と比較されて、もし条件が満たされるとそのリソース・モニターに関連付けられているタスクが呼び出されます。

タスクは、非同期に開始されます。 条件の一致があり、タスクが開始された場合、リソース・モニターはリソース・コンテンツに対してさらに変更がないかポーリングします。 例えば、`reports.go` という名前のファイルがモニター対象ディレクトリーに到着したために一致が発生した場合、そのタスクは 1 回開始されます。 たとえそのファイルがまだ存在していても、次のポーリング間隔でタスクが再度開始されることはありません。 しかし、もしファイルが削除されてディレクトリーに再び置かれるか、あるいは、そのファイルが更新される (最終変更日時属性が変更されるなど) と次のトリガー条件の検査により、再びタスクが呼び出されることになります。

IBM MQ 9.1.5 より前は、リソース・モニターがポーリング間隔よりも長い時間を要するポーリングを実行した場合、次のポーリングは現在のポーリングが終了するとすぐに、間隔を空けずに、開始されていました。そのため、リソース・モニターがエージェントにどれだけ速く処理を実行依頼できるかに影響を与える可能性がありました。 これにより、最初のポーリングで検出されたアイテムが 2 回目のポーリングでもまた検出された場合に、パフォーマンスの問題が発生する可能性がありました。

リソース・モニターは、 ScheduledExecutor サービスを使用し、前回のポーリングが完了し、構成されたポーリング間隔が経過した後でのみ、次のポーリングを開始します。 これは、ポーリング時間がポーリング間隔よりも長かった場合に、前のポーリングの後にすぐに別のポーリングを開始するのではなく、ポーリングとポーリングの間に必ず間隔を空けることを意味します。

ファイルの転送が失敗した場合は、リソース・モニター・ヒストリーをクリアすることができます。これにより、ファイルを削除してディレクトリーに再び配置したり、ファイルを更新して最終変更日時の属性を変更したりすることなく、別の転送要求を実行依頼することができます。 ファイルを転送したくてもファイルの変更が不可能な場合などに、ヒストリーをクリアできるのは便利です。 詳細については、257 ページの『リソース・モニターのヒストリーのクリア』を参照してください。

リソース

Managed File Transfer のリソース・モニターは、次の 2 つのタイプのリソースのコンテンツをポーリングできます。

ディレクトリーまたはネストされたディレクトリー構造

ディレクトリーをモニターして、トリガー・ファイルが存在するかどうかを確認する、というのが 1 つの一般的なシナリオです。 外部アプリケーションは、複数のファイルを処理して既知のソース・ディレクトリーに配置する場合があります。 アプリケーションが処理を完了すると、トリガー・ファイルをモニター対象の場所に配置することによって、ファイルを転送する準備ができているか、ファイルを転送する準備ができていることを示します。 トリガー・ファイルは、Managed File Transfer リソース・モニターによって検出され、ソース・ディレクトリーから別の Managed File Transfer Agent へのそれらのファイルの転送が開始されます。

デフォルトで、指定されたディレクトリーがモニターされています。 サブディレクトリーも検査するには、**fteCreateTransfer** コマンドの再帰レベルを設定します。

ディレクトリーをモニターする 2 つの例を以下に示します。

- トリガー・ファイル (例えば、 `trigger.file`) をモニターしてから、ワイルドカード (例えば、 `*.zip`) を転送します。
- `*.zip` のモニターを実行してから、`${FilePath}` を転送します (例えば、転送をトリガーしたファイルなど)。 変数置換について詳しくは、239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』を参照してください。

注: `*.zip` をモニターするモニターを作成しないで、 `*.zip` を転送しないでください。 モニターは、システム上のすべての .zip ファイルについて、 `*.zip` の転送を開始しようとします。 つまり、モニターは `*.zip` に対して * 数の転送を生成します。

ディレクトリーをモニターするためのリソース・モニターを作成する例については、236 ページの『ディレクトリーのモニターおよび変数置換の使用』を参照してください。

IBM MQ キュー
キューをモニターする例としては、外部アプリケーションがメッセージを生成し、既知のキューにメッセージを同じグループ ID で書き込む場合などがあります。 アプリケーションがキューにメッセージを入れ終わると、そのグループは完了したことが示されます。 メッセージの完了グループは Managed File Transfer リソース・モニターによって検出され、 ソース・キューからファイルへのメッセージのグループの転送が開始されます。 キューをモニターするためのリソース・モニターを作成する例については、238 ページの『例: MFT リソースの構成』を参照してください。

注 : 指定できるのは、1 つのキューにつき 1 つのモニターだけです。 ある IBM MQ キューをポーリングするために複数のモニターを指定した場合は、予測不能な動作が発生します。

データ・セットのモニターはサポートされていません。

条件とトリガー条件

リソースに他のストリングまたはパターンと一致する値が含まれている場合に条件が満たされます。 条件は、以下のいずれでも構いません。

- ファイル名 (パターン) が一致する。
- ファイル名 (パターン) の一致がない。
- ファイル・サイズ
- ポーリングを繰り返してもファイル・サイズが変わらない場合に一致する。

ファイル名の一致は、次のように表すことができます。

- ストリングの完全一致
- 簡単なワイルドカード・マッチング (MFT でのワイルドカード文字の使用を参照)
- 正規表現の一致

また、ファイル名は、一致することのないファイル名を識別するワイルドカードまたは Java 正規表現を使用して、ファイル名の一致から除外することもできます。

一致したファイルが検出されると、最終変更日時のタイム・スタンプが保存されます。 その後のポーリングでファイルが変更されたことが検出されると、トリガー条件が再度満たされてタスクが開始されます。 条件でファイルが存在しないときを検出するようになっている場合、モニター対象ディレクトリーにそのファイル名パターンと一致するファイルがないと、タスクが開始されます。 次にファイル名パターンと一致するファイルがそのディレクトリーに追加されると、そのファイルが削除された場合にのみタスクが開始されます。

タスク

Managed File Transfer では、リソース・モニターによる以下の 2 タイプのタスク開始の構成がサポートされています。

ファイル転送タスク
ファイル転送タスクは、他のファイル転送と同じように定義されます。 モニターが必要とするタスク XML を生成する便利な方法は、**-gt** パラメーターを指定して fteCreateTransfer コマンドを実行することです。 このコマンドは、転送仕様を含むタスク定義を XML 文書として生成します。 次に、FteCreateMonitor コマンドの **-mt** パラメーターの値として、タスク XML 文書の名前を渡します。 **fteCreateMonitor** は実行時にタスク XML 文書を読み取ります。 **fteCreateMonitor** の実行後にタスク XML ファイルに加えられた変更は、モニターで使用されません。

ファイル転送タスクを使用している場合、1 つのタスクに一括してまとめるトリガー条件の数を選択できます。 デフォルトでは、1 つのトリガー条件が 1 つのタスクを開始します。 **-bs** オプションを指定して FteCreateMonitor コマンドを実行すると、1 つのタスクにまとめてバッチ処理されるトリガー条件の数を選択できます。

コマンド・タスク
コマンド・タスクは、Ant スクリプトを実行するか、実行可能プログラムを呼び出すか、または、JCL ジョブを実行することができます。 詳しくは、232 ページの『コマンドおよびスクリプトを開始する MFT モニター・タスクの構成』を参照してください。

トリガー・ファイル

リソース・モニター内のトリガー・ファイルの内容を使用して、単一の転送要求で転送するファイルのセットを定義することができます。 一致するトリガー・ファイルが検出されるたびに、ソース・ファイル・パスに関して (オプションで宛先ファイル・パスに関しても) その内容が解析されます。 次いで、それらのファイル・パスを使用して、ユーザーが指定するタスク転送 XML ファイル内のファイル項目が定義され、それが単一の転送要求としてエージェントに実行依頼されます。 リソース・モニターの定義により、トリガー内容が使用可能かどうかが決まります。

各トリガー・ファイルの形式は、テキストの各行につき、転送する単一のファイル・パスとします。 各行のデフォルト形式は、単一のソース・ファイル・パス、またはコンマで区切ったソースと宛先のファイル・パスです。

詳細および例については、248 ページの『トリガー・ファイルの使用』を参照してください。

リソース・モニターの開始および停止

IBM MQ 9.3.0 より前では、リソース・モニターを停止する唯一の方法は、モニター操作を実行しているエージェントを停止することです。 リソース・モニターを再始動するには、一緒にエージェントを再始動する必要があります。 詳しくは、209 ページの『MFT エージェントの開始』 および 215 ページの『MFT エージェントの停止』 を参照してください。

IBM MQ 9.3.0 以降は、エージェントを停止または再始動する必要はなく、**fteStartMonitor** コマンドおよび **fteStopMonitor** コマンドを使用してリソース・モニターを開始および停止できます。 これは、例えば以下の状態において便利です。

- エージェントに複数のリソース・モニターがあり、一部のリソース・モニターのみにエラーが発生しているが残りのリソース・モニターは依然として正常に動作しており、障害が発生したリソース・モニターだけを再始動したい場合。
- 何らかのメンテナンス作業を行うためにリソース・モニターを停止したい場合。また、しばらく必要のないリソース・モニターを不必要に稼働させて、貴重なシステム・リソースを消費したくない場合。

詳しくは、253 ページの『MFT リソース・モニターの開始』および 254 ページの『MFT リソース・モニターの停止』を参照してください。

表 9. 実行するコマンドごとの異なるリソース・モニターの振る舞い

コマンド	リソース・モニターの振る舞い
fteStartMonitor	エージェントが実行中の場合、リソース・モニターが現在停止していれば、開始されます。
fteStopMonitor	エージェントが実行中の場合、リソース・モニターが現在開始済みであれば、停止します。
fteStartAgent	リソース・モニターは、エージェントの始動の一部として開始されます。(以前の **fteStopMonitor** の呼び出しとは無関係)
fteStopAgent	実行中のすべてのリソース・モニターが停止されます。

リソース・モニターのバックアップとリストア

以前に定義したリソース・モニターをバックアップしておけば、後からその定義を再利用することができます。 使用できる各種オプションを以下にまとめます。

- **fteCreateMonitor** コマンドで **-ox** パラメーターを使用すれば、1 つのリソース・モニター構成を XML ファイルにエクスポートできます。**-ix** パラメーターを使用すれば、XML ファイルからリソース・モニター構成をインポートしてリソース・モニターをリストアできます。
- **-ox** を指定した **fteListMonitors** コマンドを使用して、単一リソース・モニターの定義を XML ファイルにエクスポートします。

- 指定されたディレクトリーに複数のリソース・モニター定義をエクスポートするには、 **-od** を指定して **fteListMonitors** コマンドを使用します。 各リソース・モニター定義が別々の XML ファイルに保存されます。 **-od** オプションを使用して、単一のリソース・モニター定義を指定のディレクトリーにエクスポートすることもできます。

詳細については、255 ページの『MFT リソース・モニターのバックアップとリストア』を参照してください。

リソース・モニターのロギング

Managed File Transfer は、リソース・モニター・ロギングを含みます。 詳しくは、250 ページの『MFT リソース・モニターのロギング』を参照してください。

関連概念

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』
アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。 毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

関連タスク

232 ページの『コマンドおよびスクリプトを開始する MFT モニター・タスクの構成』
リソース・モニターの関連タスクは、ファイル転送の実行に限定されません。 また、実行可能プログラム、Ant スクリプト、または JCL ジョブなどのモニター・エージェントから他のコマンドを呼び出すようにモニターを構成することもできます。 コマンドを呼び出すには、モニター・タスク定義 XML を編集して、引数およびプロパティーなど、対応するコマンド呼び出しパラメーターを指定した 1 つ以上のコマンド・エレメントを含めます。

238 ページの『例: MFT リソースの構成』
fteCreateMonitor コマンドで **-mq** パラメーターを使用することにより、リソース・モニターによってモニターされるリソースとして IBM MQ キューを指定できます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

関連資料

fteCreateMonitor: MFT リソース・モニターの作成
fteListMonitors: MFT リソース・モニターのリスト
fteDeleteMonitor: MFT リソース・モニターの削除

コマンドおよびスクリプトを開始する MFT モニター・タスクの構成

リソース・モニターの関連タスクは、ファイル転送の実行に限定されません。 また、実行可能プログラム、Ant スクリプト、または JCL ジョブなどのモニター・エージェントから他のコマンドを呼び出すようにモニターを構成することもできます。 コマンドを呼び出すには、モニター・タスク定義 XML を編集して、引数およびプロパティーなど、対応するコマンド呼び出しパラメーターを指定した 1 つ以上のコマンド・エレメントを含めます。

このタスクについて

モニター・エージェントから呼び出せるようにする実行可能プログラム、Ant スクリプト、または JCL ジョブへのファイル・パスを、モニター・エージェントの `commandPath` に含める必要があります。 コマンド・パスのプロパティーについては、commandPath MFT プロパティーを参照してください。

以下のいずれかの方法で、タスク定義 XML 文書を作成できます。

- `FileTransfer.xsd` スキーマに従って、タスク定義 XML 文書を手動で作成します。

- 生成された XML 文書をタスク定義の基礎として使用する。

転送タスクまたはコマンド・タスクのどちらを使用するかにかかわらず、タスク定義は <request>ルート・エレメントで開始する必要があります。 <request> の子エレメントは、<managedTransfer> または <managedCall> のいずれかでなければなりません。 実行するコマンドまたはスクリプトが 1 つの場合は、通常 <managedCall> を選択し、ファイル転送とオプションで最大 4 つのコマンド呼び出しをタスクに含める場合は <managedTransfer> を選択します。

手順

- FileTransfer.xsd スキーマに従って手動でタスク定義 XML 文書を作成するには、233 ページの『スキーマに従って手動でタスク定義 XML を作成する』を参照してください。
- 生成された文書を変更することによってタスク定義を作成する場合は、**fteCreateTransfer -gt** パラメーターによって生成された XML 文書を編集します。 詳しくは、235 ページの『生成済み文書の変更によるタスク定義文書の作成』を参照してください。

スキーマに従って手動でタスク定義 *XML* を作成する

スキーマ FileTransfer.xsd に従って、タスク定義 XML ファイルを手動で作成することができます。

このタスクについて

スキーマ FileTransfer.xsd は、 *MQ_INSTALLATION_PATH*/mqft/samples/schema 内にあります。 このスキーマについて詳しくは、ファイル転送要求メッセージ・フォーマットを参照してください。

例

以下の例は、 <managedCall> エレメントを使用して RunCleanup.xml という Ant スクリプトを呼び出す、 cleanuptask.xml, として保存されたタスク定義 XML 文書の例を示しています。 RunCleanup.xml Ant スクリプトは、モニター・エージェントの コマンド・パス 上に配置する必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedCall>
     <originator>
       <hostName>hostName</hostName>
       <userID>userID</userID>
       <mqmdUserID>mqmdUserID</mqmdUserID>
     </originator>
     <agent QMgr="QM1" agent="AGENT1"/>
     <reply QMGR="QM1">reply</reply>
     <transferSet priority="1">
       <metaDataSet>
         <metaData key="name1">value1</metaData>
       </metaDataSet>
       <call>
           <command name="RunCleanup.xml" type="antscript" retryCount="2"
          retryWait="30" successRC="0">
           <target>check_exists</target>
           <target>copy_to_archive</target>
           <target>rename_temps</target>
             <target>delete_files</target>
           <property name="trigger.filename" value="${FileName}"/>
           <property name="trigger.path" value="${FilePath}"/>
           </command>
       </call>
     </transferSet>
     <job>
       <name>JOBCLEAN1</name>
     </job>
   </managedCall>
</request>
```

<agent> エレメントは、 commandPath 上の指定された Ant スクリプトで構成される Managed File Transfer Agent を指定します。

<call><command>... 構造は、実行する実行可能ファイルまたはスクリプトを定義します。 このコマンドは、オプションの type 属性を取得し、この属性には以下のいずれかの値を指定できます。

antscript
Ant スクリプトを別個の JVM で実行します。

executable
実行可能プログラムを呼び出します。

jcl
JCL ジョブを呼び出します。

type 属性を省略すると、デフォルト値の 実行可能 が使用されます。

name 属性は、実行する Ant スクリプト、実行可能プログラム、または JCL ジョブの名前を、パス情報なしで指定します。 エージェントは、エージェントの agent.properties ファイル内の コマンド・パス プロパティーによって指定されたロケーションで、スクリプトまたはプログラムを検索します。

retrycount 属性は、プログラムが成功の戻りコードを返さない場合に、プログラムの呼び出しを再試行する回数を指定します。 この属性に指定する値は、負の値を指定することはできません。 retrycount 属性を指定しない場合は、デフォルト値のゼロが使用されます。

retrywait 属性は、プログラム呼び出しを再試行するまでの待機時間を秒単位で指定します。 この属性に指定する値は、負の値を指定することはできません。 retrywait 属性を指定しない場合は、デフォルト値のゼロが使用されます。

successrc 属性は、プログラム呼び出しがいつ正常に実行されるかを決定するために使用される式です。 コマンドの処理戻りコードは、この式を使用して評価されます。 値は、ブール値の OR を表す垂直バー文字 (|)、またはブール値の AND を表すアンパーサンド (&) 文字で結合された 1 つ以上の式で構成することができます。 各式は、以下のいずれかのタイプの式とすることができます。

- 処理戻りコードとの等価テストを示す数値。
- 処理戻りコードとの大なりテストを示す、接頭部に「大なり」文字 (>) が付いた数値。
- 処理戻りコードとの小なりテストを示す、接頭部に「小なり」文字 (<) が付いた数値。
- 処理戻りコードとの不等テストを示す、接頭部に感嘆符文字 (!) が付いた数値。 例えば、>2&<7&!5|0|14 は、戻りコード 0、3、4、6、14 を正常と解釈します。 これ以外の戻りコードは、すべて失敗と解釈されます。

successrc 属性を指定しない場合は、デフォルト値のゼロが使用されます。 これは、ゼロの戻りコードを戻した場合にのみ、コマンドは正常に実行されたと判断されるという意味です。

Ant スクリプトの場合、通常は <target> エレメントと <property> エレメントを指定します。 <target> エレメントの値は、 Ant スクリプト内のターゲット名と一致している必要があります。

実行可能プログラムの場合は、 <argument> エレメントを指定できます。 ネストされた argument エレメントを使用すると、プログラム呼び出しの一部として呼び出されるプログラムに渡される引数が指定されます。 このプログラム実引数は、argument エレメントの出現する順序で argument エレメントにより指定された値から構成されます。 ゼロ個以上の argument エレメントをプログラム呼び出しのネストされたエレメントとして指定できます。

管理者は、 <managedCall> エレメントを含むタスク定義 XML 文書を使用して、通常どおりにモニターを定義および開始します。 以下に例を示します。

```
fteCreateMonitor -ma AGENT1 -mm QM1 -md /monitored -mn MONITOR01 -mt
   /tasks/cleanuptask.xml -pi 30 -pu seconds -tr match,*.go
```

転送定義 XML 文書へのパスは、 **fteCreateMonitor** コマンドを実行するローカル・ファイル・システムに ある必要があります (この例では、/tasks/cleanuptask.xml)。 cleanuptask.xml 文書は、リソース・モニターのみを作成するために使用されます。 cleanuptask.xml 文書が参照するタスク (Ant スクリプトまたは JCL ジョブ) は、モニター・エージェントのコマンド・パス内になければなりません。 モニター・トリガー条件が満たされると、タスク定義 XML 内のすべての変数はモニターからの実際の値で置換されます。 したがって、例えば ${FilePath} は、 /monitored/cleanup.go を使用してエージェントに送信

される要求メッセージ内で置き換えられます。 要求メッセージは、エージェントのコマンド・キューに置かれます。 コマンド・プロセッサーは、要求がプログラム呼び出し用であることを検出し、指定されたプログラムを開始します。 タイプ antscript のコマンドが呼び出されると、新規 JVM が開始され、 Ant タスクが新規 JVM で実行されます。 変数置換の使用法について詳しくは、『変数置換を使用したタスクのカスタマイズ』を参照してください。

関連概念

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』
アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。 毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

関連資料

ファイル転送要求メッセージ・フォーマット
commandPath MFT プロパティー

生成済み文書の変更によるタスク定義文書の作成

fteCreateTransfer の **-gt** オプションによって生成された XML 文書を変更することによって、モニター・タスク定義文書を作成できます。

このタスクについて

生成された文書には、<request> の後に <managedTransfer> エレメントが付きます。 このタスク定義を有効な <managedCall> 構造に変換するには、以下のステップを実行します。

手順

1. <managedTransfer> 開始タグと終了タグを <managedCall> タグに置き換えます。
2. <schedule> エレメントおよび子ノードをすべて削除します。
3. <sourceAgent> の開始タグと終了タグを <agent> に置き換えて、モニター・エージェントの構成の詳細を一致させます。
4. <destinationAgent> エレメントと <trigger> エレメントを削除します。
5. <item> エレメントを削除します。
6. preSourceCall、postSourceCall、preDestinationCall、または postDestinationCall エレメントをすべて削除します。
7. 新しい <call>...</call> 構造を <transferSet> エレメント内に挿入します。 この構造には、以下の例に示すコマンド定義が含まれます。

```
<call>
          <command name="RunCleanup.xml" type="antscript" retryCount="2"
         retryWait="30" successRC="0">
          <target>check_exists</target>
          <target>copy_to_archive</target>
          <target>rename_temps</target>
            <target>delete_files</target>
          <property name="trigger.filename" value="${FileName}"/>
          <property name="trigger.path" value="${FilePath}"/>
          </command>
      </call>
```

例

また、すべてのファイル転送の詳細を含む <managedTransfer> エレメントを保持し、最大 4 つのコマンド呼び出しを挿入することもできます。 この場合、以下の呼び出しエレメントの選択を <metaDataSet> エレメントと <item> エレメントの間に挿入します。

preSourceCall
ソース・エージェント上のプログラムを呼び出してから転送を開始します。

postSourceCall
転送を完了した後にソース・エージェント上のプログラムを呼び出します。

preDestinationCall
宛先エージェント上のプログラムを呼び出してから転送を開始します。

postDestinationCall
転送を完了した後に宛先エージェント上のプログラムを呼び出します。

これらの各エレメントは、前の例で説明したように、<command> エレメント構造になります。 FileTransfer.xsd スキーマは、さまざまな呼び出しエレメントで使用されるタイプを定義します。

次の例では、タスク定義文書内の preSourceCall、postSourceCall、preDestinationCall、および postDestinationCall を示します。

```
⋮
    <transferSet priority="1">
       <metaDataSet>
         <metaData key="key1">value1</metaData>
       </metaDataSet>
       <preSourceCall>
         <command name="send.exe" retryCount="0" retryWait="0" successRC="0"
           type="executable">
           <argument>report1.pdf</argument>
           <argument>true</argument>
         </command>
       </preSourceCall>
       <postSourceCall>
         <command name="//DO_IT.JCL" retryCount="0" retryWait="0" successRC="0"
           type="jcl">
           <argument>argument</argument>
         </command>
       </postSourceCall>
       <preDestinationCall>
         <command name="ant_script.xml" retryCount="0" retryWait="0" successRC="0"
           type="antscript">
           <target>step1</target>
           <property name="name" value="value"/>
         </command>
       </preDestinationCall>
       <postDestinationCall>
         <command name="runit.cmd" retryCount="0" retryWait="0" successRC="0" />
       </postDestinationCall>
       <item checksumMethod="none" mode="binary">
⋮
```

異なるタイプのコマンドを転送に混入できます。 引数、ターゲット、およびプロパティーの各要素はオプションです。

ディレクトリーのモニターおよび変数置換の使用

fteCreateMonitor コマンドを使用して、ディレクトリーをモニターすることができます。 置換変数の値をタスク XML 定義に置換して、転送動作の定義に使用できます。

このタスクについて

この例では、ソース・エージェントの名前は AGENT_HOP です。 AGENT_HOP モニターがモニターするディレクトリーは、/test/monitored と呼ばれます。 エージェントは、ディレクトリーを 5 分おきにポーリングします。

.zip ファイルがディレクトリーに書き込まれると、そのディレクトリーにファイルを書き込むアプリケーションは、同じディレクトリーにトリガー・ファイルを書き込みます。 トリガー・ファイルの名前は、.zip ファイルの名前と同じになりますが、ファイル拡張子は異なります。 例えば、ファイル file1.zip がディレクトリーに書き込まれた後で、ファイル file1.go がディレクトリーに書き込まれます。 リソース・モニターは、パターン *.go と一致するファイルのディレクトリーをモニターし、その後、変数置換を使用して、関連する .zip ファイルの転送を要求します。

手順

1. モニター起動時にモニターが実行するタスクを定義するタスク XML を作成します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>blue.example.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP" />
     <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <file>/test/monitored/${fileName{token=1}{separator=.}}.zip</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/out/${fileName{token=1}{separator=.}}.zip</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

   トリガー・ファイルに関連する値に置き換えられる変数を、**太字**で強調表示しています。 このタスク XML は、ファイル /home/USER1/task.xml に保存されます。

2. ディレクトリー /test/monitored をモニターするためのリソース・モニターを作成します。

   以下のコマンドを実行依頼します。

```
fteCreateMonitor -ma AGENT_HOP -mm QM_HOP -md /test/monitored
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr match,*.go -pi 5 -pu minutes
```

3. ユーザーまたはプログラムは、ファイル jump.zip をディレクトリー /test/monitored に書き込んでから、ファイル jump.go をディレクトリーに書き込みます。
4. このモニターは、ファイル jump.go が存在することによってトリガーされます。 エージェントは、トリガー・ファイルに関する情報を、タスク XML に置換します。

   この結果、タスク XML は以下のように変換されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>blue.example.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP" />
     <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <file>/test/monitored/jump.zip</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/out/jump.zip</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

タスクの結果

タスク XML によって定義された転送が実行されます。 jump.zip ファイルは、 AGENT_HOP によって /test/monitored ディレクトリーから読み取られ、 AGENT_SKIP が実行されているシステム上にある /out/jump.zip というファイルに転送されます。

関連概念

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』

アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。 毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

関連タスク

232 ページの『コマンドおよびスクリプトを開始する MFT モニター・タスクの構成』

リソース・モニターの関連タスクは、ファイル転送の実行に限定されません。 また、実行可能プログラム、Ant スクリプト、または JCL ジョブなどのモニター・エージェントから他のコマンドを呼び出すようにモニターを構成することもできます。 コマンドを呼び出すには、モニター・タスク定義 XML を編集して、引数およびプロパティーなど、対応するコマンド呼び出しパラメーターを指定した 1 つ以上のコマンド・エレメントを含めます。

関連資料

fteCreateMonitor: MFT リソース・モニターの作成

例: MFT リソースの構成

fteCreateMonitor コマンドで **-mq** パラメーターを使用することにより、リソース・モニターによってモニターされるリソースとして IBM MQ キューを指定できます。

このタスクについて

この例では、モニターされるリソースは *MONITORED_QUEUE* というキューです。 このキューは、モニター・エージェントのキュー・マネージャー *QM_NEPTUNE* に存在していなければなりません。 キューがモニター対象になる条件は、メッセージの完全グループが存在することです。 条件が満たされた場合に実行されるタスクは、ファイル task.xml に定義されます。

注 : 個々のキューをモニターするために、複数のリソース・モニターを作成しないでください。 作成した場合、予測不能な動作が発生します。

手順

次のコマンドを入力します。

```
fteCreateMonitor -ma AGENT_NEPTUNE -mn myMonitor -mm QM_NEPTUNE -mq MONITORED_QUEUE
  -mt task.xml -tr completeGroups -pi 5 -pu minutes
```

条件 completeGroups が真であるかどうかを調べるため、モニターは 5 分ごとにキューをチェックします。 キューに 1 つ以上の完全なグループがある場合、モニターは、 task.xml ファイルに定義されているタスクを、完全なグループごとに 1 回実行します。

関連概念

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』

アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。 毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

関連タスク

232 ページの『コマンドおよびスクリプトを開始する MFT モニター・タスクの構成』

リソース・モニターの関連タスクは、ファイル転送の実行に限定されません。 また、実行可能プログラム、Ant スクリプト、または JCL ジョブなどのモニター・エージェントから他のコマンドを呼び出すようにモニターを構成することもできます。 コマンドを呼び出すには、モニター・タスク定義 XML を編集して、引数およびプロパティーなど、対応するコマンド呼び出しパラメーターを指定した 1 つ以上のコマンド・エレメントを含めます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

関連資料

fteCreateMonitor: MFT リソース・モニターの作成

変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ

アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。 毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

重要 : 変数名は大/小文字を区別しません。

置換に使用される変数は、正のトリガー条件でのみ使用可能です。 `match` および `fileSize` トリガー条件によってのみ、変数は置換されます。 不一致 条件が使用され、タスク定義に置換変数名がある場合、タスクは呼び出されず、モニターは 110 とエラー・メッセージ BFGDM0060E の戻りコードを出します。

モニターされるリソースがキューの場合

モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値を、タスク XML 定義に置換できます。

ユーザー定義メッセージ・プロパティーには、接頭部 `usr.` が付きますが、変数名にはこの接頭部を含めません。 変数名は、中括弧 {} で囲んで、その前にドル記号 ($) 文字を付加する必要があります。

例えば、`${destFileName}` は、ソース・キューから読み取られる最初のメッセージの `usr.destFileName` メッセージ・プロパティーの値に置き換えられます。 詳しくは、MFT がソース・キューのメッセージから読み取る MQ メッセージ・プロパティーおよび 244 ページの『キューのモニターおよび変数置換の使用』を参照してください。

変数がメッセージ・プロパティーとして定義されていない場合、モニターは BFGDM0060E エラーを報告し、戻りコード 110 を戻します (モニター・タスク変数置換が失敗しました)。 これに加えて、エージェントは以下のエラー・メッセージをイベント・ログ (`outputN.log`) に書き込みます。

```
BFGDM0113W: Trigger failure for <monitor name> for reason BFGDM0060E: A monitor task could
not complete as a variable substitution <variable name> was not present.
```

モニターに対して中程度のリソース・モニター・ロギングまたは詳細リソース・モニター・ロギングが使用可能になっている場合、モニターは以下のメッセージをエージェントのリソース・モニター・イベント・ログ (`resmoneventN.log`) に書き込みます。

```
BFGDM0060E: A monitor task could not complete as a variable substitution
<variable name> was not present.
```

リソース・モニターのロギングについて詳しくは、250 ページの『MFT リソース・モニターのロギング』を参照してください

デフォルトで用意されている置換変数を以下の表にまとめます。 例えば、`${AGENTNAME}` は、リソース・モニター・エージェントの 名前に置換されます。

表 *10.* デフォルトで用意されている置換変数	
変数	**説明**
AGENTNAME	リソース・モニター・エージェントの名前。
QUEUENAME	モニターされるキューの名前。
ENCODING	キューにある最初のメッセージ、またはグループにある最初のメッセージの文字エンコード。
MESSAGEID	キューにある最初のメッセージ、またはグループにある最初のメッセージの IBM MQ メッセージ ID。
GROUPID	グループの IBM MQ グループ ID、または単一のメッセージのみ検出された場合はメッセージ ID。 この変数は完全なグループをモニターしている場合のみ、設定されます。
CurrentTimeStamp	モニターがトリガーした時の現地時間に基づいたタイム・スタンプ。 タイム・スタンプ値はエージェントに固有です。
CurrentTimeStampUTC	モニターがトリガーした時の UTC タイム・ゾーンでのタイム・スタンプ。 タイム・スタンプ値はエージェントに固有です。

モニターされるリソースがディレクトリーの場合

次の表に、タスク XML 定義で置換できる一連の変数名を示します。

表 *11.* 置換可能な変数	
変数	**説明**
FilePath	トリガー・ファイルの完全なパス名。
FileName	トリガーのファイル名の部分。
LastModifiedTime	トリガー・ファイルの最終変更時刻。 エージェントを実行しているタイム・ゾーンの現地時間が ISO 8601 の時間形式で表示されます。
LastModifiedDate	トリガー・ファイルの最終変更日。 エージェントを実行しているタイム・ゾーンの現地日付が ISO 8601 の日付形式で表示されます。
LastModifiedTimeUTC	トリガー・ファイルの最終変更時刻。 この時刻は、UTC タイム・ゾーンに変換された現地時間として表され、ISO 8601 時間として書式設定されます。
LastModifiedDateUTC	トリガー・ファイルの最終変更日。 この日付は、UTC タイム・ゾーンに変換された現地日付として表され、 ISO 8601 日付として書式設定されます。
AgentName	リソース・モニター・エージェントの名前。
CurrentTimeStamp	モニターがトリガーした時の現地時間に基づいたタイム・スタンプ。 タイム・スタンプ値はエージェントに固有です。
CurrentTimeStampUTC	モニターがトリガーした時の UTC タイム・ゾーンでの時刻に基づいたタイム・スタンプ。 タイム・スタンプ値はエージェントに固有です。

モニター対象リソースがトリガー・ファイルの場合

次の表に、リソース・モニターがトリガー・ファイルの内容を使用して転送する必要のあるファイルを判別するときに置き換えることができる変数名のセットを示します。

表 12. トリガー・ファイルを使用する際に置換できる変数	
変数	**説明**
contentSource	ソース・ファイルの完全パス名。
contentDestination	宛先ファイルの完全パス名。

変数名の前には、ドル記号 ($) 文字が必要です。また、中括弧で囲まれていなければなりません。例えば、${FilePath} は、一致するトリガー・ファイルの完全修飾ファイル・パスに置き換えられます。

変数名に適用してさらに細分化できる特殊キーワードが 2 つあります。 次のとおりです。

トークン
置換するトークン索引 (左側は 1 から始まり、右側は -1 から始まる)

分離文字
変数値をトークン化する単一文字。 デフォルトは、AIX and Linux プラットフォームではスラッシュ文字 (/)、 Windows プラットフォームでは円記号文字 (¥) ですが、区切り文字には、変数値に使用できる任意の有効な文字を使用できます。

変数名で separator キーワードが指定された場合には、変数値は、その分離文字でトークンに分割されます。

token キーワードに割り当てた値は、変数名を置き換えるために使用するトークンを選択する索引として使用されます。 トークン索引は、変数内の最初の文字に対して相対的なもので、 1 から始まります。 token キーワードが指定されていない場合は、変数全体が挿入されます。

メッセージ XML のエージェント名に 置換される値はすべて、大/小文字を区別せずに扱われます。 Managed File Transfer Agent 名はすべて大文字です。 Paris という値がメッセージ XML のエージェント属性に置換された場合には、 この値はエージェント PARIS への参照として解釈されます。

関連概念
241 ページの『例: リソース・モニター定義の変数置換』
XML と IBM MQ Explorer を使用したリソース・モニター定義の変数置換の例。

関連タスク
変数置換によって複数のファイルが 1 つのファイル名に送られる場合の対応策

例: リソース・モニター定義の変数置換

XML と IBM MQ Explorer を使用したリソース・モニター定義の変数置換の例。

変数置換の仕組みを示す例

一致するトリガー・ファイルへのファイル・パスが、 Windows プラットフォームでは c:\MONITOR\REPORTS\Paris\Report2009.doc 、 AIX and Linux プラットフォームでは /MONITOR/REPORTS/Paris/Report2009.doc であると仮定すると、変数は以下の表に示すように置換されます。

表 13. 変数の置換方法	
変数の指定	**変数置換後**
${FilePath}	Windows :c:\MONITOR\REPORTS\Paris\Report2009.doc AIX and Linux : /MONITOR/REPORTS/Paris/Report2009.doc

表 13. 変数の置換方法 (続き)	
変数の指定	**変数置換後**
${FilePath{token=1}{separator=.}}	Windows :c:\MONITOR\REPORTS\Paris\Report2009 AIX and Linux : /MONITOR/REPORTS/Paris/Report2009
${FilePath{token=2}{separator=.}}	Windows: 資料 AIX and Linux : 資料
${FilePath{token=3}}	Windows : レポート AIX and Linux : パリ

負のトークン索引を指定して、変数の最後の文字を基準にした相対指定でトークンを選択することもできます (以下の表を参照)。 表の例では、同じ変数値 c:\MONITOR\REPORTS\Paris\Report2009.doc (Windows の場合) と /MONITOR/REPORTS/Paris/Report2009.doc (AIX and Linux の場合) を使用しています。

表 14. 負のトークン索引の使用例	
変数の指定	**変数置換後**
${FilePath}	Windows :c:\MONITOR\REPORTS\Paris\Report2009.doc AIX and Linux : /MONITOR/REPORTS/Paris/Report2009.doc
${FilePath{token=-2}{separator=.}}	Windows :c:\MONITOR\REPORTS\Paris\Report2009 AIX and Linux : /MONITOR/REPORTS/Paris/Report2009
${FilePath{token=-2}{separator=\}}	Windows : パリ AIX and Linux : パリ
${FilePath{token=-4}}	Windows : モニター AIX and Linux : モニター

置換に使用される変数は、以下の肯定的なトリガー条件および noSizeChange オプションに対してのみ使用可能です。これは、肯定トリガー条件規則の例外です。

- 一致
- fileSize
- noSizeChange

不一致 条件が使用され、タスク定義に置換変数名がある場合、タスクは呼び出されず、モニターは 110 とエラー・メッセージ BFGDM0060E の戻りコードを出します。

XML の使用例

以下のタスク定義 XML の例では、転送のソース・エージェントとしてモニター・エージェント名 (Paris) を使用し、転送の宛先エージェント名としてファイル・パスの penultimate ディレクトリー名

(Report2009) を使用し、転送後のファイル名を、トリガー・ファイルのルートに拡張子 .rpt を付けた名前に変更しています。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="${AgentName}" QMgr="QM1" />
     <destinationAgent agent="${FilePath{token=-2}}" QMgr="QMD" />
     <transferSet>
       <item mode="binary" checksumMethod="MD5">
         <source recursive="false" disposition="leave">
           <file>c:/incoming/reports/summary/report.doc</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/${FileName{token=1}{separator=.}}.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

この結果、タスク XML は以下のように変換されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT1" QMgr="QM1" />
     <destinationAgent agent="Paris" QMgr="QMD" />
     <transferSet>
       <item mode="binary" checksumMethod="MD5">
         <source recursive="false" disposition="leave">
           <file>c:/incoming/reports/summary/report.doc</file>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/Report2009.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

<destinationAgent> エレメントの agent 属性にある変数 ${FilePath{token=-2}} は、値「Paris」で置き換えられます。 この値は大/小文字を区別せずに扱われ、エージェント PARIS への参照として解釈されます。

IBM MQ Explorer の使用例

IBM MQ Explorer でリソース・モニターを作成する時に、モニター・プロパティーとトリガー条件を指定すると、転送項目をモニターに追加するためのオプションが表示されます。 次の例は、"**転送項目の追加パネル**"で${FilePath}と${FileName}変数を使用して、リソースモニタの一致から生じる転送をカスタマイズする方法を示しています。

例 1

トリガー条件に適合した時にソース・ファイルを別の場所に転送するために、${FilePath} 変数を使用します。

- ソース・**ファイル名**を ${FilePath} に設定します。
- 宛先の「**タイプ**」のドロップダウン・メニューから、「**ディレクトリー**」を選択します。

- 宛先の **ファイル名** を、ソース・ファイルの転送先のロケーションに設定します。例えば、C:\MFT\out\にすることができます。

例 2

ソース・ファイルを別の場所に転送し、ファイル拡張子を変更するために、${FilePath} 変数と一緒に ${FileName} 変数を使用します。

以下の例では、ソース・ファイルのファイル・パスが C:\MONITOR\REPORTS\Paris\Report2009.doc に等しいと想定されています。

- ソース・**ファイル名**を ${FilePath} に設定します。
- 宛先**ファイル名**をソース・ファイルの転送先の場所に設定し、その後に ${FileName{token=1}{separator=.}}、さらにその後にファイルの新しい拡張子を指定します。 例えば、ソースファイル名 C:\MFT\out\Report2009.rpt と等しい C:\MFT\out\${FileName{token=1}{separator=.}}.rpt , とすることができます。

例 3

ソース・ファイルのファイル・パスの一部を使用して転送の宛先を指定するために、トークンや区切り文字と一緒に ${FilePath} 変数を使用します。

以下の例では、ソース・ファイルのファイル・パスが C:\MONITOR\REPORTS\Paris\Report2009.doc に等しいことを前提としています。

ソース・ファイル・パスの一部を使用してファイルの宛先を指定できます。C:\MONITOR\REPORTS\Paris\Report2009.doc のファイル・パスの例を使用して、ファイルがソース・ファイルの場所 (この例では Paris) に応じてフォルダーに転送される場合は、以下のことを行うことができます。

- ソース・**ファイル名**を ${FilePath} に設定します。
- 宛先**ファイル名**をそれぞれの場所に対応するフォルダーが置かれる宛先に設定し、ファイル・パスの宛先の部分とファイル名を追加します。 例えば、ソースファイル名 C:\MFT\out\Paris\Report2009.doc と等しい C:\MFT\out\${FilePath{token=-2}{separator=\}}\${FileName} , とすることができます。

関連概念

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』

アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

関連タスク

変数置換によって複数のファイルが 1 つのファイル名に送られる場合の対応策

キューのモニターおよび変数置換の使用

fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

このタスクについて

この例では、ソース・エージェントは AGENT_VENUS という名前であり、QM_VENUS に接続します。AGENT_VENUS がモニターするキューは START_QUEUE という名前であり、QM_VENUS にあります。 エージェントは、キューを 30 分おきにポーリングします。

メッセージの完全に揃ったグループがキューに書き込まれると、モニター・タスクは、いくつかの宛先エージェントの 1 つのファイルにメッセージのグループを送信します。この宛先エージェントは、すべてキ

ュー・マネージャー QM_MARS に接続しています。 メッセージのグループが転送されるファイルの名前は、グループの最初のメッセージの IBM MQ メッセージ・プロパティー usr.fileName で定義します。 メッセージのグループが送信されるエージェントの名前は、グループの最初のメッセージの IBM MQ メッセージ・プロパティー usr.toAgent で定義します。 usr.toAgent ヘッダーが未設定の場合は、宛先エージェント用に使用されるデフォルト値は、AGENT_MAGENTA です。

useGroups="true" を指定する場合、groupId="${GROUPID}" を指定しないと、転送ではキュー内の最初のメッセージのみが取得されます。 そのため、例えば変数置換を使用して fileName を生成した場合、a.txt の内容が正しくなくなる可能性があります。 これは、fileName はモニターによって生成されますが、転送では、実際には fileName というファイルを生成するメッセージではなく、別のメッセージを取得するためです。

手順

1. モニター起動時にモニターが実行するタスクを定義するタスク XML を作成します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
     <destinationAgent agent="${toAgent}" QMgr="QM_MARS" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/${fileName}.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

   IBM MQ メッセージ・ヘッダーの値で置き換えられる変数は、**太字**で強調表示しています。 このタスク XML は、ファイル /home/USER1/task.xml に保存されます。

2. キュー START_QUEUE をモニターするリソース・モニターを作成します。

   以下のコマンドを実行依頼します。

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. ユーザーまたはプログラムは、メッセージのグループをキュー START_QUEUE に書き込みます。

   このグループの最初のメッセージは、次の IBM MQ メッセージ・プロパティーを設定しています。

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. 完全に揃ったグループが書き込まれると、モニターが起動されます。 エージェントは、IBM MQ メッセージ・プロパティーをタスク XML に置換します。

   この結果、タスク XML は以下のように変換されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
```

```
        <hostName>reportserver.com</hostName>
        <userID>USER1</userID>
      </originator>
      <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
      <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS" />
      <transferSet>
        <item mode="binary" checksumMethod="none">
          <source>
            <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
          </source>
          <destination type="file" exist="overwrite">
            <file>/reports/larmer.rpt</file>
          </destination>
        </item>
      </transferSet>
    </managedTransfer>
</request>
```

タスクの結果

タスク XML によって定義された転送が実行されます。 AGENT_VENUS によって START_QUEUE から読み取られるメッセージの完全なグループは、 AGENT_VIOLET が実行されているシステム上の `/reports/larmer.rpt` というファイルに書き込まれます。

次のタスク

各メッセージの別個のファイルへの転送

キューをモニターして、すべてのメッセージが別個のファイルに転送されるようにする場合には、このトピックで前述した方法と同様の方法を使用することができます。

1. **fteCreateMonitor** コマンドに **-tr completeGroups** パラメーターを指定して、前述のようにモニターを作成します。
2. タスク XML で、次のように指定します。

```
<queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
```

ただし、メッセージをソース・キューに入れる場合は、それらのメッセージを IBM MQ グループには入れないでください。 IBM MQ メッセージ・プロパティーを各メッセージに追加します。 例えば、メッセージごとに固有のファイル名の値を持つ `usr.filename` プロパティーを指定します。 こうすることで効果的に、Managed File Transfer Agent がソース・キュー内の各メッセージを異なるグループとして扱います。

関連概念

276 ページの『メッセージからファイルへのデータ転送』
Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』
アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。 毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

キュー・リソース・モニターが開始した転送によって作成された宛先ファイルに 間違ったデータが含まれる場合の対処法

関連タスク

232 ページの『コマンドおよびスクリプトを開始する MFT モニター・タスクの構成』
リソース・モニターの関連タスクは、ファイル転送の実行に限定されません。 また、実行可能プログラム、Ant スクリプト、または JCL ジョブなどのモニター・エージェントから他のコマンドを呼び出すようにモニターを構成することもできます。 コマンドを呼び出すには、モニター・タスク定義 XML を編集して、引数

およびプロパティーなど、対応するコマンド呼び出しパラメーターを指定した 1 つ以上のコマンド・エレメントを含めます。

238 ページの『例: MFT リソースの構成』
fteCreateMonitor コマンドで **-mq** パラメーターを使用することにより、リソース・モニターによってモニターされるリソースとして IBM MQ キューを指定できます。

関連資料
fteCreateMonitor: MFT リソース・モニターの作成
MFT がソース・キューのメッセージから読み取る MQ メッセージ・プロパティー

メッセージからファイルへの転送の再試行動作のモニターの構成

リソース・モニターにより起動されたメッセージからファイルへの転送が失敗し、モニターを起動したメッセージ・グループがキューに残っている場合、その転送は後続のポーリング間隔で再発信されます。 転送が再発信される回数は、モニター・エージェントの **monitorGroupRetryLimit** プロパティーにより制限されます。

このタスクについて

メッセージからファイルへの転送が新たに起動されるたびに、転送タスクに対して新しい転送 ID が生成されます。

エージェントを再始動すると、転送がトリガーされた回数が agent.properties ファイル内の **monitorGroupRetryLimit** の値を超えた場合でも、モニターは転送を再度トリガーします。 **monitorGroupRetryLimit** プロパティーの値は、メッセージ・グループがまだキューに存在している場合、モニターがメッセージからファイルへの転送を再度起動する最大回数です。 このプロパティーのデフォルト値は 10 です。 このプロパティーの値は、任意の正整数値または -1 に設定できます。 このプロパティーに値 -1 が指定された場合、モニターは、起動条件が満たされなくなるまで何度でも転送を再度起動します。

転送の試行により、起動された転送回数が **monitorGroupRetryLimit** の値を超えてしまった場合、エージェントはイベント・ログにエラーを書き込みます。

1 つのメッセージはあたかも 1 つのグループであったように処理され、メッセージがキューに残っており、転送が起動された回数が **monitorGroupRetryLimit** の値未満である間は、ポーリング間隔ごとに転送が起動されます。

モニター・エージェントで **monitorGroupRetryLimit** プロパティーを設定するには、次の手順を実行します。

手順

1. **fteStopAgent** コマンドを使用してモニター・エージェントを停止します。
2. モニター・エージェントの agent.properties ファイルを編集して、以下の行を組み込みます。

   ```
   monitorGroupRetryLimit=number_of_retries
   ```

   agent.properties ファイルは、ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*monitoring_agent_name* に置かれています。
3. **fteStartAgent** コマンドを使用してモニター・エージェントを開始します。

関連タスク
238 ページの『例: MFT リソースの構成』
fteCreateMonitor コマンドで **-mq** パラメーターを使用することにより、リソース・モニターによってモニターされるリソースとして IBM MQ キューを指定できます。

トリガー・ファイルの使用

リソース・モニター内のトリガー・ファイルの内容を使用して、単一の転送要求で転送するファイルのセットを定義することができます。 一致するトリガー・ファイルが検出されるたびに、ソース・ファイル・パスに関して (オプションで宛先ファイル・パスに関しても) その内容が解析されます。 次いで、それらのファイル・パスを使用して、ユーザーが指定するタスク転送 XML ファイル内のファイル項目が定義され、それが単一の転送要求としてエージェントに実行依頼されます。 リソース・モニターの定義により、トリガー内容が使用可能かどうかが決まります。

-tc (トリガー内容) パラメーターを指定することによって、モニターを作成するときに、ファイル内容によるトリガー発行を使用可能にできます。 この **-tc** パラメーターは、ファイル・トリガー・オプションの match および noSizeChange にのみ適用されます。 モニターの作成について詳しくは、**fteCreateMonitor**: MFT リソース・モニターの作成を参照してください。

トリガー・コンテンツ・ファイルを使用する場合、各行のデフォルトの形式は次のいずれかです。

- 単一のソース・ファイル・パス、または
- コンマで区切られたソース・ファイル・パスと宛先ファイル・パス

ここで、空白文字はファイル・パスの一部として処理されます。 **fteCreateMonitor** コマンドで **-tcr** パラメーターと **-tcc** パラメーターを指定することにより、デフォルトの行形式を変更することができます。 詳しくは、249 ページの『拡張オプション』を参照してください。

トリガー・ファイルが解析された後、ファイル・パスのリストが生成され、ユーザーが指定した転送タスク XML に適用されます。 すべてのモニターの場合と同様に、転送タスク XML の形式は、単一の項目またはファイルが定義された **fteCreateTransfer** コマンドによって生成される、完全な転送タスク XML です。 単一項目では、ソースおよび宛先ファイルのパスで置き換えるものとして、置換変数 ${contentSource} と、オプションで ${contentDestination} を使用する必要があります。 モニターは転送タスク XML を拡張して、トリガー・ファイル内の行ごとにファイル項目 (ファイル・パス) が含まれるようにします。

-tc パラメーターはトリガー・ファイルごとに 1 つの転送要求を暗黙指定するので、**-bs** パラメーターと共にファイル内容によるトリガー発行を使用することはできません。

例

以下の例では、trig で終わるファイルに対してトリガーするモニターを定義し、そのファイル内のファイル・パスを読み取ります。

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -dd /file/destdir ${contentSource}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
```

fteCreateTransfer コマンドは、ソース・ファイル・パスが ${contentSource}の単一ファイルに対して task.xml というファイルを作成します。 以下に例を示します。

```
<item checksumMethod="MD5" mode="binary">
    <source disposition="leave" recursive="false">
          <file>${contentSource}</file>
    </source>
</item>
```

fteCreateMonitor コマンドは、/home/trigdir ディレクトリーにあって trig で終わるファイルをスキャンし、その内容を使用して、そのトリガー・ファイル内のすべてのパスに対し、task.xml に基づく単一転送要求を作成します。 トリガー・ファイルの必須の形式は、各行に 1 つのファイル・パス (ソースのみ) で、コンマ分離文字はありません。 以下に例を示します。

```
/home/file/first.txt
/home/file/second.txt
/home/different/third.txt
⋮
```

すべてのファイルは、ファイル・パスではなく、ファイル名で /file/destdir ディレクトリーに送信されます。つまり、 /home/file/first.txt が /file/destdir/first.txt に送達されます。

あるいは、**fteCreateTransfer** コマンドの **-dd /file/destdir** パラメーターを -df ${contentDestination} に変更し、トリガー・ファイルの内容の形式を *source file path*,*destination file path* とすれば、同じ宛先エージェントに対して異なる宛先パスを定義できます。 以下に例を示します。

```
/home/file/first.txt,/home/other/sixth.txt
```

これで、宛先ロケーションは /home/other/sixth.txt になります

置換変数はトークン化することができます。 例えば、 ${contentDestination{token=-1}}を使用して、指定されたパスからファイル名部分を分離することができます。 したがって、 **fteCreateTransfer** 宛先が -df /file/destdir/${contentDestination{token=-1}}として定義されている場合、 /home/file/first.txt の新しい宛先は /file/destdir/sixth.txt になります。

拡張オプション

-tcr *regex* パラメーターを使用することにより、トリガー・ファイルの内容のデフォルト行形式を変更することができます。 必要な行形式に一致する正規表現を提供し、1 つか 2 つのキャプチャー・グループを提供します。 最初のキャプチャー・グループはソースで、2 番目のオプションのキャプチャー・グループは宛先です。 以下に例を示します。

- ソースと宛先のパスをハイフンで分離します。

  ```
  ((?:[^-])+)-((?:[^-])+)
  ```

  この例では、分離文字は 3 つの場所で定義されており、ハイフン (-) の 3 つの出現個所のすべては任意の文字に変更することができます。 特殊文字は、必ずすべてエスケープしてください。
- ソースと宛先のパスを、コンマと後続のスペースで分離します。 番号記号 (#) で示されるコメントは無視されます。

  ```
  ((?:[^,])+),((?:[^,])+) *(?:#.*)+
  ```

  ファイル・パスに番号記号 (#) を含めることはできません。 通常、エントリーは /home/source/from.txt,/home/destination/to.txt *# some comment* のようになります。

-tcr パラメーターを使用する場合には、式でエラーを検出できるよう、また正しくトリガー・ファイルを解析できるよう、正規表現が適切に設計され、テストされていることを確認してください。

-tcc destSrc パラメーターを使用することにより、キャプチャーの順序を逆にすることができます。 このパラメーターを指定する場合、最初のキャプチャー・グループは宛先ファイル・パス、2 番目のグループはソース・ファイル・パスになります。

エラーの処理方法

空のトリガー・ファイル
トリガー・ファイルが空である場合、結果としてファイル転送が行われません。 つまり、モニターは転送要求を作成しますが、ファイル項目は何も指定されません。

エラーのあるトリガー・ファイル
トリガー・ファイル内の何らかのエントリーが、期待される形式に照らして解析に失敗した場合、転送要求は生成されません。 モニター・エラー・ログがパブリッシュされ、イベント・ログにもエラーが記録されます。 トリガー・ファイルは処理済みとしてマークを付けられ、そのファイルが更新されるまでは、モニターはファイルを再度処理しようとはしません。

一致しない転送タスク XML
転送タスク XML は、トリガー・ファイルと一致している必要があります。つまり、転送タスク XML に ${contentSource} と ${contentDestination}の両方が含まれている場合、そのモニターのすべてのトリガー・ファイルには、ソース・ファイル・パスと宛先ファイル・パス、および逆方向のファイル・パスが

最初のケースでは、トリガー・ファイルがソース・ファイル・パスのみを提供している場合に、モニターが ${contentDestination} の置換失敗を報告します。

例

以下の例は、トリガー・ファイルの内容がソース・ファイル・パスだけである、基本的な内容トリガーです。

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -dd /file/destdir ${contentSource}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
```

-tcr パラメーターは、スペース文字で分離された任意の文字のシーケンスのキャプチャー・グループを 2 つ定義します。 **-tcc destSrc** パラメーターおよびオプションは、キャプチャー・グループがまず宛先として、それからソースとして処理されることを示します。

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -df ${contentDestination} $
{contentSource}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
    -tcr "((?:[^ ])+) ((?:[^ ])+)" -tcc destSrc
```

関連概念

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』

アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

関連タスク

244 ページの『キューのモニターおよび変数置換の使用』

fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

関連資料

fteCreateMonitor: MFT リソース・モニターの作成

fteCreateTransfer: 新規ファイル転送の開始

MFT リソース・モニターのロギング

ロギングを使用して、リソース・モニターに関する診断情報を取得できます。

このタスクについて

fteSetAgentLoglevel コマンドまたは agent.properties ファイルを使用してリソース・モニター・ロギングを制御することにより、リソース・モニターのロギングを使用できます。

情報の取得に既存のトレース・ポイントが引き続き使用されることに注意してください。

リソース・モニター・ログは、 resmonevent*N*.log という名前のファイルに書き込まれます。ここで、*N* は数値を表します。例えば、 resmonevent0.log です。 イベント・ログ・ファイルは、モニターがリソース(例えば、ディレクトリーまたはキュー) をポーリングするときに実行されるいくつかのアクションを記録します。

重要: 1 つのエージェントのリソース・モニターはすべて同じログ・ファイルに書き込まれます。

resmonevent*N*.log ファイルの出力例については、「MFT ディレクトリー・リソース・モニターがファイルをトリガーしない場合の対処方法」を参照してください。

次の表に、リソース・モニターがログ・ファイルに書き込むイベントのタイプをリストします。 3 番目の列は、各イベントの取り込みに必要なログ・レベルを示しており、最低レベルが INFO で、最高レベルが VERBOSE です。

高いログ・レベルを設定すると、それより低いレベルのイベントも書き込まれることに注意してください。例えば、ログ・レベルを MODERATE に設定すると、INFO レベルのイベントも書き込まれますが、VERBOSE レベルのイベントは書き込まれません。

数値	イベント	ログ・レベル	説明
1	作成されたモニター	INFO	リソース・モニターが作成されました。
2	削除されたモニター	INFO	リソース・モニターが削除されました。
3	モニターが停止しました	INFO	リソース・モニターが停止されました。
4	モニター開始	INFO	リソース・モニターが開始されました。
5	ポーリングの開始	INFO	リソース・モニターが新しいポーリング周期を開始しました。
6	ポーリングの終了	INFO	リソース・モニターのポーリング周期が終了しました。
7	パターンの一致	VERBOSE	トリガー・モニター・ディレクトリー上のファイル、または指定されたパターンに一致するキュー内のメッセージが見つかりました。
8	パターンの不一致	VERBOSE	トリガー・モニター・ディレクトリー上の一致しないファイル、または指定されたパターンに一致しないキュー内のメッセージが見つかりました。
9	転送要求	INFO	リソース・モニターにより転送が開始されました。
10	ディレクトリーが深すぎる	VERBOSE	リソース・モニターによってモニターされるディレクトリーに、リソース・モニター構成で指定された数より多くのポーリング対象サブディレクトリーが含まれています。
11	ファイルのロック	MODERATE	リソース・モニターがモニターするトリガー・ファイルが、別のプロセスによってロックされています。
12	ファイル・サイズが小さい	MODERATE	トリガー・ファイルが、リソース・モニター構成で指定されているサイズより小さいサイズです。
13	ファイル・サイズが固定されていない	MODERATE	トリガー・ファイルが、リソース・モニター構成で予想されるより頻繁に変更されています。
14	ポーリングが多すぎる	MODERATE	リソース・モニターによるサイズが固定されていないトリガー・ファイルのポーリング回数が多すぎます。
15	一致する項目	INFO	リソース・モニターによってポーリングされるディレクトリーで見つかったトリガー・ファイルの総数。
16	転送項目	INFO	転送要求内の項目の総数。
17	FDC 生成	MODERATE	リソース・モニターが例外を生成しました。

数値	イベント	ログ・レベル	説明
18	転送要求	INFO	リソース・モニターにより転送要求が実行依頼されました。
19	モニター開始の失敗	MODERATE	リソース・モニターの開始に失敗しました。
20	履歴の消去	INFO	モニター履歴情報がクリアされました。
21	モニター履歴のクリアに失敗しました	INFO	モニター履歴情報をクリアしようとしましたが、失敗しました。
22	転送 ID	INFO	転送要求の ID がモニターによって実行依頼されました。
23	バッチ処理	INFO	一致した項目の転送要求の総数: *N*。ここで、*N* は数値です。
V 9.4.0 24	接続	VERBOSE	リソース・モニターがエージェント・キュー・マネージャーに接続しました。
V 9.4.0 25	切断	VERBOSE	リソース・モニターがエージェント・キュー・マネージャーから切断されました。
V 9.4.0 26	切断中のエラー	VERBOSE	エージェント・キュー・マネージャーから切断しているときに、リソース・モニターで問題が発生しました。

手順

- **fteSetAgentLoglevel** コマンドを使用してリソース・モニターのロギングのオン/オフを切り替えるには、fteSetAgentLogLevel で **logMonitor** パラメーターの説明と各種オプションの使用例を参照してください。
- agent.properties ファイルを使用してリソース・モニター・ロギングを制御するには、 MFT agent.properties ファイル を参照して、以下のロギング・アクティビティーを実行できる追加プロパティーの説明を参照してください。
  - ロギングをオンまたはオフにする
  - 各ログ・ファイルのサイズを制限する
  - リソース・モニターで生成可能なログの数を制限する

例

以下のサンプル・メッセージは、キュー・マネージャー MFTDEMO 上のエージェント HA2 の verbose レベルのロギングを設定します。

```
<?xml version="1.0"?>
<log:log version="6.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xmlns:log="https://www.ibm.com/log">
    <log:originator>
      <log:request>
        <log:hostName>192.168.7.1</log:hostName>
        <log:userID>johndoe</log:userID>
      </log:request>
    </log:originator>
    <log:endpoint agent="HA2" QMgr="MFTDEMO"/>
    <log:logMonitor>MON1="verbose"</log:logMonitor>
</log:log>
```

関連資料

fteSetAgentLogLevel コマンド
MFT agent.properties ファイル

MFT リソース・モニターの開始

IBM MQ 9.3.0 以降では、**fteStartMonitor** コマンドを使用することにより、エージェントを停止または再始動せずにリソース・モニターを開始できます。

始める前に

agent.properties ファイルで **authorityChecking** 属性を true に設定することによってユーザー権限管理が有効になっている場合、リソース・モニターを開始するには、 モニター 権限または モニター操作 権限のいずれかが必要です。 ユーザー権限管理の詳細については、MFT エージェント・アクションのユーザー権限の制限を参照してください。

このタスクについて

fteStartMonitor コマンドは、 Managed File Transfer コマンド・コンポーネントがインストールされている任意のシステムから実行できます。これは、どこからでもリソース・モニターを開始でき、リソース・モニターを所有するエージェントが実行されているシステムに制限されないことを意味します。 このコマンドの必須パラメーターおよびオプション・パラメーターの詳細については、fteStartMonitor (MFT リソース・モニターの開始) を参照してください。

手順

- エージェントの状態を **fteStartMonitor** コマンドの実行前または実行後に確認するには、以下の例に示すように、**fteListMonitors** コマンドに **-v** パラメーターを指定して使用します。

  ```
  fteListMonitors -ma monitoring_agent_name -v
  ```

- 同じマシンで実行されているエージェントのリソース・モニターを開始するには、次のように **fteStartMonitor** コマンドを入力します。

  ```
  fteStartMonitor -mn monitor_name -ma agent_name
  ```

- 別のマシンで実行されているエージェントのリソース・モニターを開始するには、次のように **fteStartMonitor** コマンドを入力します。

  ```
  fteStartMonitor -mn monitor_name -ma agent_name -mm AgentQueueManager
  ```

  コマンド・キュー・マネージャーがモニター・エージェントのエージェント・キュー・マネージャーでもある場合は、**-mm** パラメーターはオプションです。それ以外の場合は、**-mm** パラメーターを使用してエージェント・キュー・マネージャーを指定する必要があります。

タスクの結果

エージェントが実行中の場合、リソース・モニターが現在停止していれば、開始されます。 このコマンドは、以下のメッセージを出力し、エージェントの output0.log にイベントを記録します。

```
BFGCL0816I: エージェント 'エージェント名' のリソース・モニター 'モニター名' を開始する要求が発行されました。
BFGCL0251I: 要求が正常に完了しました。
```

リソース・モニターを開始できない場合にコマンドが出力するメッセージについては、fteStartMonitor (MFT リソース・モニターの開始) を参照してください。

関連概念

228 ページの『MFT のリソース・モニターの概念』
Managed File Transfer のリソース・モニター機能の主要概念の概要。

関連タスク

254 ページの『MFT リソース・モニターの停止』
IBM MQ 9.3.0 以降では、**fteStopMonitor** コマンドを使用することにより、エージェントを停止または再始動せずにリソース・モニターを停止できます。

関連資料
fteStartMonitor (MFT リソース・モニターの開始)

MFT リソース・モニターの停止

IBM MQ 9.3.0 以降では、**fteStopMonitor** コマンドを使用することにより、エージェントを停止または再始動せずにリソース・モニターを停止できます。

始める前に

agent.properties ファイルで **authorityChecking** 属性を true に設定することによってユーザー権限管理が有効になっている場合、リソース・モニターを停止するには、 モニター 権限または モニター操作 権限のいずれかが必要です。 ユーザー権限管理の詳細については、MFT エージェント・アクションのユーザー権限の制限を参照してください。

このタスクについて

fteStopMonitor コマンドは、 Managed File Transfer コマンド・コンポーネントがインストールされている任意のシステムから実行できます。これは、リソース・モニターをどこからでも停止でき、リソース・モニターを所有するエージェントが実行されているシステムに制限されないことを意味します。 このコマンドの必須パラメーターおよびオプション・パラメーターの詳細については、fteStopMonitor (MFT リソース・モニターの停止) を参照してください。

リソース・モニターが停止すると、エージェントのリソース・モニター・イベント・ログ resmoneventnumber.log にメッセージが書き込まれます。 **fteStopMonitor** コマンドを使用して、リソース・モニターが停止された場合、メッセージには停止要求を発行したユーザーの名前が含まれます。

ユーザー '*<mquser_id>*' によってリソース・モニターが停止しました

リソース・モニターが **fteStopMonitor** コマンドを使用して停止された場合であっても、リソース・モニターは、そのエージェントが再始動されると自動的に開始されます。

エージェントはモニター停止要求を並列処理ではなく、順次処理するため、エージェントがモニター M1 の停止要求を受け取り、その直後に続けてモニター M2 の停止要求を受け取った場合、M2 の停止を試行する前に、まず M1 を停止します。

手順

- エージェントの状態を **fteStopMonitor** コマンドの実行前または実行後に確認するには、以下の例に示すように、**fteListMonitors** コマンドに **-v** パラメーターを指定して使用します。

  ```
  fteListMonitors -ma monitoring_agent_name -v
  ```

- 同じマシンで実行されているエージェントのリソース・モニターを停止するには、次のように **fteStopMonitor** コマンドを入力します。

  ```
  fteStopMonitor -mn monitor_name -ma agent_name
  ```

- 別のマシンで実行されているエージェントのリソース・モニターを停止するには、次のように **fteStopMonitor** コマンドを入力します。

  ```
  fteStopMonitor -mn monitor_name -ma agent_name -mm AgentQueueManager
  ```

  コマンド・キュー・マネージャーがモニター・エージェントのエージェント・キュー・マネージャーでもある場合は、**-mm** パラメーターはオプションです。それ以外の場合は、**-mm** パラメーターを使用してエージェント・キュー・マネージャーを指定する必要があります。

タスクの結果

エージェントが実行中の場合、リソース・モニターが現在開始済みであれば、停止します。 このコマンドは、以下のメッセージを出力し、エージェントの output0.log にイベントを記録します。

```
BFGCL0813I: エージェント「SOURCE」のリソース・モニター「MNTR」の停止の要求が発行されました（A request to stop
resource monitor 'MNTR' of agent 'SOURCE' has been issued)。
BFGCL0251I: 要求が正常に完了しました。
```

リソース・モニターを停止できない場合にコマンドが出力するメッセージについては、fteStopMonitor (MFT リソース・モニターの停止) を参照してください。

関連概念

228 ページの『MFT のリソース・モニターの概念』
Managed File Transfer のリソース・モニター機能の主要概念の概要。

関連タスク

253 ページの『MFT リソース・モニターの開始』
IBM MQ 9.3.0 以降では、**fteStartMonitor** コマンドを使用することにより、エージェントを停止または再始動せずにリソース・モニターを開始できます。

関連資料

fteStopMonitor (MFT リソース・モニターの停止)

MFT リソース・モニターのバックアップとリストア

リソース・モニターをバックアップしておけば、後で使用することができます。そのためには、リソース・モニターの定義を XML ファイルにエクスポートします。エクスポートした定義をインポートすれば、バックアップから新しいリソース・モニターを作成できます。

このタスクについて

以前に定義したリソース・モニターをバックアップしておけば、後からその定義を再利用することができます。例えば、別のインフラストラクチャーでリソース・モニターを再作成する場合や、キュー・マネージャーの問題が原因でリソース・モニターの再作成が必要になる場合などが考えられます。

1 つのリソース・マネージャー定義をバックアップするには、**fteCreateMonitor** コマンドまたは **fteListMonitors** コマンドで **-ox** パラメーターを使用します。 どちらの場合も、リソース・マネージャー定義を XML ファイルにエクスポートすることでバックアップを作成します。 **fteCreateMonitor** コマンドの **-ix** パラメーターを使用してその定義を XML ファイルからインポートすれば、新しいリソース・マネージャーを作成できます。

-ox パラメーターを使用すると、一度にバックアップできるリソース・モニター定義は 1 つのみです。

-od パラメーターが **fteListMonitors** コマンドに追加されました。 このパラメーターを指定すれば、複数のリソース・モニター定義を指定のディレクトリーに一括してエクスポートすることで、バックアップを一度に作成できます。 各リソース・モニター定義は、 *agent name*.*monitor name*.xml という形式の名前を持つ個別の XML ファイルに保存されます。

バックアップするリソース・モニターが多数存在する場合は、**-od** パラメーターが非常に便利です。リソース定義ごとに **fteListMonitors -ox** コマンドを別々に実行したり、リソース・モニターごとに別々のスクリプトで **fteListMonitors -ox** コマンドを実行したりする代わりに、**fteListMonitors -od** コマンドを 1 回実行するだけです。

手順

- 1 つのリソース・モニター定義を XML ファイルにエクスポートしてバックアップを作成するには、以下のいずれかのコマンドを使用します。
  - **-ox** パラメーターを指定した **fteCreateMonitor** コマンド。
  - **-ox** パラメーターを指定した **fteListMonitors** コマンド。

  -ox パラメーターを使用する場合は、以下の例に示すように、 **-ma** パラメーターと **-mn** パラメーターも指定する必要があります。

  ```
  fteListMonitors -ma AGENT1 -mn MONITOR1 -ox filename1.xml
  ```

- 複数のリソース・モニター定義を、指定されたディレクトリー内の XML ファイルにエクスポートすることによってバックアップするには、以下の例に示すように、 **-od** パラメーターを指定して **fteListMonitors** コマンドを使用します。

```
fteListMonitors -od /usr/mft/resmonbackup
```

  複数のリソース・モニターを一括してバックアップする場合は、有効なターゲット・ディレクトリーを指定する必要があります。 ターゲット・パスを指定しないと、以下の例のようなエラー・メッセージが表示されます。

  BFGCL0762E: 出力ディレクトリーが指定されていません。 有効なパスを指定して、コマンドを再実行してください。

  -od パラメーターを **-ox** パラメーターと組み合わせて使用することはできません。そうしないと、以下のエラー・メッセージが表示されます。

  BFGCL0761E: '-od' パラメーターと '-ox' パラメーターを両方一緒に指定することは無効です。

  バックアップに含めるリソース・モニターのセットを定義できます。 例えば、 **-ma** パラメーターを使用してエージェントの名前を指定すると、以下の例に示すように、そのエージェントのすべてのリソース・モニターをバックアップできます。

```
fteListMonitors -ma AGENT1 -od /usr/mft/resmonbackup
```

  エージェント名とモニター名のいずれかまたは両方と突き合わせるために使用するパターンの定義時には、アスタリスク文字 (*) を組み込んだワイルドカード・マッチングも使用できます。 以下の例では、指定のパターンに合致する名前のエージェントにある、指定のパターンに合致する名前のすべてのリソース・モニターをバックアップします。

```
fteListMonitors -ma AGENT* -mn MON* -od /usr/mft/resmonbackup
```

  コマンドの実行中に、以下のような進行状況レポート・メッセージが表示されます。

  合計 *number* 個の一致するリソース・モニター定義が見つかりました。
  number 個中 *index* 個のリソース・モニター定義がファイル・システムに保存されました。

  詳細オプションを使用した場合でも合計実行数は表示されますが、以下の部分の代わりに、

  number 個中 *index* 個のリソース・モニター定義がファイル・システムに保存されました

  保存されるリソース・モニター定義の名前がこのコマンドで表示されます。例えば、以下のようになります。

  BFGCL0762I: エージェント 'XFERAGENT' のモニター 'FILEMON' の定義が FILEMON.XFERAGENT.XML をファイル・システムにコピーします。

- 特定のエージェントの 1 つのリソース・モニターを、指定されたディレクトリー内の XML ファイルにエクスポートすることによってバックアップするには、 **-od** パラメーターを指定して **fteListMonitors** コマンドを使用します。

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -od /usr/mft/resmonbackup
```

  -od パラメーターを使用して単一のリソース・モニターをバックアップする方法は、 **-ox** パラメーターを使用する方法と似ていますが、出力ファイル名の形式が *agent name*.*monitor name*.xml である点が異なります。

- バックアップからリソース・モニター定義をリストアする場合は、以下の例のように **-ix** パラメーターを付けて **fteCreateMonitor** コマンドを使用します。

```
fteCreateMonitor -ix file name
```

  -od パラメーターの使用方法のその他の例については、「fteListMonitors: MFT リソース・モニターのリスト」を参照してください。

関連資料

fteCreateMonitor: MFT リソース・モニターの作成
fteListMonitors: MFT リソース・モニターのリスト

リソース・モニターのヒストリーのクリア

リソース・モニターのヒストリーをクリアできることで、ファイル転送が失敗した場合に、そのファイルの転送要求を再び送信できます。 リソース・モニターのヒストリーをクリアするには、**fteClearMonitorHistory** コマンドか IBM MQ Explorer を使用します。

始める前に

agent.properties ファイルで **authorityChecking** 属性を true に設定することによってユーザー権限管理が有効になっている場合、モニター履歴をクリアするユーザーは、以下の表に示す適切な権限を持っている必要があります。

表 *15.* ***fteClearMonitorHistory*** コマンドの実行に必要なユーザー権限

モニター・ヒストリーをクリアするユーザー	**MFT のアクセス権限**	**必要な権限**
リソース・モニターを作成したユーザー。	モニター	SYSTEM.FTE.AUTHMON1.<モニター・エージェント名> 上でのブラウズ リソース・モニターの作成や削除に必要な権限と同じです。
リソース・モニターを作成したユーザー以外のユーザー。	モニター操作	SYSTEM.FTE.AUTHPS1.<エージェント名> に対する設定 リソース・モニターの削除に必要な権限と同じです。

ユーザー権限管理の詳細については、MFT エージェント・アクションのユーザー権限の制限を参照してください。

必要な権限を持たないユーザーがリソース・モニター・ヒストリーをクリアしようとすると、**fteClearMonitorHistory** コマンドはエラー・メッセージを出力し、その失敗をエージェントの output0.log ファイルに記録します。 詳細については、fteClearMonitorHistory: リソース・モニターのヒストリーのクリアを参照してください。

このタスクについて

ファイル転送が開始されたものの何らかの理由で転送できなかった場合は、そのファイルが次回のポーリングでリソース・モニターによって転送対象として再び選択されることはありません。前回のポーリングに含まれていて、その時以降変更されていないということが、モニター・ヒストリーに示されているためです (228 ページの『MFT のリソース・モニターの概念』を参照)。

IBM MQ 9.1.3 より前では、ファイル転送の失敗後にファイル転送を再び開始するには、ファイルを削除してディレクトリーに配置し直すか、ファイルを更新して最終変更日時属性を変更するか、リソース・モニター自体を再作成することが必要でした。

ただし、**fteClearMonitorHistory** コマンドまたは IBM MQ Explorer を使用して、リソース・モニターのヒストリーをクリアすることができます。 ヒストリーをクリアすれば、ファイルを削除してディレクトリーに配置し直したり、ファイルを更新して最終変更日時属性を変更したりしなくても、転送できなかったファイルの別の転送要求を送信できます。これは、ファイルを転送したくてもファイルの変更が不可能な場合などに便利です。 リソース・モニターのヒストリーをクリアできるので、転送できなかったファイルの別の転送要求を送信するためにリソース・モニターを再作成する必要もありません。

z/OS Managed File Transfer on z/OS に同梱されているサンプル SCSQFCMD メンバーには、モニターのヒストリーを消去する JCL スクリプトが含まれています。

手順

- **fteClearMonitorHistory** コマンドでリソース・モニターのヒストリーをクリアするには、以下の形式でコマンドを入力します。

  ```
  fteClearMonitorHistory -p <configuration> -ma <agent name> -mn <monitor name> -w 1000
  ```

  必須のパラメーターは **-ma** と **-mn** だけです。 他のパラメーターはすべてオプションです。 **fteClearMonitorHistory** コマンドの詳しい使い方と使用例については、fteClearMonitorHistory: リソース・モニターのヒストリーのクリアを参照してください。

  ヒストリーのクリアが成功すると、コマンドから以下のメッセージが出力されます。

  ```
  BFGCL0780I: エージェント 「agent name」のリソース・モニター「monitor name」のヒストリーをクリアする要求が送信されました。
  BFGCL0251I: 要求が正常に完了しました。
  ```

  成功をエージェントの output0.log ファイルに記録します。

  リソース・モニターのヒストリーをクリアしようとして失敗した場合、 **fteClearMonitorHistory** はエラー・メッセージを出力し、その失敗をエージェントの output0.log ファイルに記録します。

- IBM MQ Explorer の MFT プラグインでリソース・モニター・ビューを使用してリソース・モニターのヒストリーをクリアする場合は、リソース・モニターを右クリックして、ドロップダウン・メニューから「**ヒストリーのクリア**」を選択します。

  ヒストリーのクリアが成功すると、以下のメッセージが表示されます。

  ```
  BFGUI00171: Resource monitor history cleared successfully.
  ```

  ヒストリーをクリアしようとして失敗すると、エラー・メッセージが表示されます。 以下に例を示します。

  ```
  BFGUI0016E Failed to clear history of specified resource monitor - reason 2059
  ```

ファイル転送テンプレートの処理

ファイル転送テンプレートを使用すると、繰り返しの転送または複雑な転送を行うための共通のファイル転送設定を保管できます。 転送テンプレートは **fteCreateTemplate** コマンドを使用してコマンド行から作成します。また、IBM MQ Explorer で、「**ファイル転送管理のテンプレート新規作成**」ウィザードを使用して転送テンプレートを作成することも、ファイル転送の作成時に「**転送設定をテンプレートとして保存する**」チェック・ボックスを選択してテンプレートを保存することもできます。 「**転送テンプレート**」ウィンドウには、Managed File Transfer ネットワーク内に作成した転送テンプレートがすべて表示されます。

このタスクについて

コマンド行から転送テンプレートを作成するには、 fteCreateTemplate コマンドを使用します。 次に、コマンド行で作成した転送テンプレートを送信するには、 IBM MQ Explorer で 「**送信**」 をクリックします。

IBM MQ Explorer で転送テンプレートを表示するには、以下のステップを実行します。

手順

1. 「ナビゲーター」ビューで「**ファイル転送管理**」を展開します。 「**ファイル転送管理 - メイン**」が「コンテンツ」ビューに表示されます。
2. すべての調整キュー・マネージャーが「ナビゲーター」ビューにリストされます。 スケジュール済みの転送に使用した調整キュー・マネージャーの名前を展開します。 接続先の調整キュー・マネージャーを変更する場合は、「ナビゲーター」ビューで使用する調整キュー・マネージャーの名前を右クリックして、「**接続**」をクリックします。
3. 「**転送テンプレート**」をクリックします。 「**転送テンプレート**」ウィンドウが「コンテンツ」ビューに表示されます。
4. 「**転送テンプレート**」ウィンドウには、ファイル転送に関する以下の詳細がリストされます。
   a) 「**名前**」。ファイル転送テンプレートの名前。

b)「**ソース**」。ソース・システムからファイルを転送するために使用するエージェントの名前。

c)「**ソース・ファイル**」。ホスト・システムにおける、転送するファイルの名前。
このフィールドを表示するには、転送テンプレート情報を展開する必要があります。

d)「**宛先**」。宛先システムでファイルを受け取るために使用するエージェントの名前。

e)「**宛先ファイル**」。宛先システムに転送された後のファイルの名前。
このフィールドを表示するには、転送テンプレート情報を展開する必要があります。

f)「**スケジュール済みの開始時刻 (選択したタイム・ゾーン)**」 ファイル転送を開始するようスケジュールされた、管理者が使用するタイム・ゾーンでの時刻と日付。 表示されるタイム・ゾーンを変更するには、 **ウィンドウ > 設定 > IBM MQ Explorer > Managed File Transfer** をクリックし、 **タイム・ゾーン:** リストから別のタイム・ゾーンを選択します。 「**OK**」をクリックします。

g)「**トリガー・イベント**」。ファイル転送を起動して開始させるイベントのタイプ。 タイプは次のいずれかの値になります。存在、存在しない、または超過。

タスクの結果

「**転送テンプレート**」ウィンドウに表示されている内容を最新表示するには、「コンテンツ」ビューのツールバーにある「リフレッシュ」ボタン をクリックします。

転送テンプレートを実行依頼し、テンプレートで定義されている転送を開始するには、テンプレート名を右クリックし、「**実行依頼**」をクリックします。

転送テンプレートを変更するには、テンプレート名を右クリックして「**編集**」をクリックします。 元のテンプレートに含まれているすべてのファイルが転送グループの一部として表示されます (それらのファイルが元のテンプレートでグループの一部として組み込まれていない場合でも、そのような動作になります)。テンプレートからファイルを削除する場合は、グループからそのファイル指定を選択し、「**選択した項目を削除**」をクリックする必要があります。 テンプレートに新しいファイル指定を追加する場合は、テンプレート・パネルにあるフィールドを使用して、「**グループに追加**」ボタンをクリックします。 編集を行うと、編集済みテンプレートに新しい名前を付けるように求められます。

転送テンプレートからファイル転送を作成するには、テンプレート名を右クリックして「**新規の転送として編集**」をクリックします。

転送テンプレートの複写コピーを作成するには、テンプレート名を右クリックして「**複製**」をクリックします。 重複転送テンプレートは、元のテンプレートと同じ名前で自動的に保存され、 "(コピー)"に追加されます。

転送テンプレートを削除するには、テンプレート名を右クリックして「**削除**」をクリックします。

関連タスク

259 ページの『IBM MQ Explorer を使用したファイル転送テンプレートの作成』
ファイル転送テンプレートを IBM MQ Explorer またはコマンド行から作成することができます。 その後そのテンプレートを使用してそのテンプレート詳細を使用する新規ファイル転送を作成したり、そのテンプレートを送信してファイル転送を開始したりできます。

関連資料

fteCreateTemplate: 新規ファイル転送テンプレートの作成

fteListTemplates

fteDeleteTemplates

IBM MQ Explorer を使用したファイル転送テンプレートの作成

ファイル転送テンプレートを IBM MQ Explorer またはコマンド行から作成することができます。 その後そのテンプレートを使用してそのテンプレート詳細を使用する新規ファイル転送を作成したり、そのテンプレートを送信してファイル転送を開始したりできます。

このタスクについて

ファイル転送テンプレートをコマンド行から作成するには、fteCreateTemplate コマンドを使用します。

IBM MQ Explorer の 「**ファイル転送管理用テンプレートの新規作成**」 ウィザードを使用してファイル転送テンプレートを作成するには、以下のステップを実行します。

手順

1. 「ナビゲーター」ビューで、「**ファイル転送管理**」をクリックします。 「**ファイル転送管理 - メイン**」が「コンテンツ」ビューに表示されます。
2. すべての調整キュー・マネージャーが「ナビゲーター」ビューに表示されます。 スケジュール済みの転送に使用した調整キュー・マネージャーの名前を展開します。 接続先の調整キュー・マネージャーを変更する場合は、「ナビゲーター」ビューで使用する調整キュー・マネージャーの名前を右クリックして、「**接続**」をクリックします。
3. 「**転送テンプレート**」を右クリックしてから「**テンプレートの新規作成**」をクリックして、「**ファイル転送管理のテンプレート新規作成**」ウィザードを開始します。
4. ウィザード・パネルの指示に従います。 各パネルにはコンテキスト・ヘルプがあります。 Windows 上でコンテキスト・ヘルプにアクセスするには、F1 キーを押します。 Linux 上では、Ctrl+F1 キーまたは Shift+F1 キーを押します。

   転送に関する必要な全詳細を含むテンプレートを作成してある場合、「**転送の要約**」ページで「**転送設定をテンプレートとして保存する**」チェック・ボックスがまだ選択されていない場合には、このチェック・ボックスを必ず選択します。 また、「名前」フィールドにテンプレートの名前を入力します。 転送に関する必要な全詳細がまだ含まれていないテンプレートを作成している場合、「**転送設定をテンプレートとして保存する**」チェック・ボックスに自動的にチェック・マークが付けられます。

関連タスク

258 ページの『ファイル転送テンプレートの処理』

ファイル転送テンプレートを使用すると、繰り返しの転送または複雑な転送を行うための共通のファイル転送設定を保管できます。 転送テンプレートは **fteCreateTemplate** コマンドを使用してコマンド行から作成します。また、IBM MQ Explorer で、「**ファイル転送管理のテンプレート新規作成**」ウィザードを使用して転送テンプレートを作成することも、ファイル転送の作成時に「**転送設定をテンプレートとして保存する**」チェック・ボックスを選択してテンプレートを保存することもできます。 「**転送テンプレート**」ウィンドウには、Managed File Transfer ネットワーク内に作成した転送テンプレートがすべて表示されます。

関連資料

fteCreateTemplate: 新規ファイル転送テンプレートの作成

fteListTemplates

fteDeleteTemplates

ファイル転送テンプレート定義のバックアップ

ファイル転送テンプレートには、転送のソース・ファイルおよび宛先ファイルの仕様を定義した XML 文書が含まれています。 この XML ファイルを **fteCreateTemplate** コマンドの入力として使用すると、ファイル転送テンプレートを再作成できます。

このタスクについて

転送テンプレートのソース・ファイルと宛先ファイルの仕様が含まれた XML 文書をバックアップするには、fteCreateTransfer コマンドまたは IBM MQ Explorer を使用します。 転送テンプレートの XML 形式のバックアップ・ファイルを作成するには、以下の手順を実行します。

手順

- 方法 1: fteCreateTransfer コマンドで **-gt** パラメーターを使用して、転送テンプレート XML メッセージを新規ファイルに生成します。
- 方法 2: IBM MQ Explorer を使用してテンプレートを作成します。
  「転送テンプレートの要約」ページまで進み、以下を実行します。

a)「要求メッセージ *XML* のプレビュー」をコピーします。

b) この転送テンプレートの XML メッセージを新しいファイルに保存します。

- 方法 3: IBM MQ Explorer を使用して、既存のテンプレートをバックアップします。

  a)「**ファイル転送管理**」 >「**キュー・マネージャー名**」 >「**転送テンプレート**」と移動します。

  b)「転送」パネルで、バックアップする必要があるテンプレートを強調表示し、右クリックしてポップアップ・メニューから「**編集**」を選択します。

  c)「転送テンプレートの要約」ページが表示されるまで、「**次へ**」をクリックします。

  d)「要求メッセージ *XML* のプレビュー」をコピーします。

  e) この転送テンプレートの XML メッセージを新しいファイルに保存します。

タスクの結果

上記のいずれかの方法で作成した転送テンプレートの XML メッセージのファイルを、fteCreateTemplate コマンドへの入力として使用できます。 このコマンドの使用法について詳しくは、**fteCreateTemplate** コマンドを参照してください。

関連資料

fteCreateTemplate コマンド

fteListtTmplates コマンド

ファイルからメッセージへのデータ転送

Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

メッセージからファイルへの転送に関しては、276 ページの『メッセージからファイルへのデータ転送』を参照してください。

ファイルからメッセージへの転送の宛先エージェントは、プロトコル・ブリッジ・エージェントまたは Connect:Direct ブリッジ・エージェントであることはできません。

ファイル・データを IBM MQ のメッセージ・データに転送できます。 IBM MQ のメッセージは、各種アプリケーションで読み取ったり使用したりできます。 ファイルからメッセージへの転送では、以下のタイプの転送がサポートされています。

- 1 つのファイルから 1 つのメッセージへ。 メッセージには、IBM MQ グループ ID が設定されていません。
- 1 つのファイルから複数のメッセージへ (ファイルを指定の長さのメッセージに分割します)。 すべてのメッセージには、同じ IBM MQ グループ ID が割り当てられます。
- 1 つのファイルから複数のメッセージへ (テキスト・ファイルを Java 正規表現の区切り文字で分割します)。 すべてのメッセージには、同じ IBM MQ グループ ID が割り当てられます。
- 1 つのファイルから複数のメッセージへ (バイナリー・ファイルを 16 進数の区切り文字で分割します)。 すべてのメッセージには、同じ IBM MQ グループ ID が割り当てられます。

  区切り文字として一連のバイトを使用してバイナリー・ファイルを分割するには、**fteCreateTransfer** コマンドで **-sqdb** パラメーターを指定します。 詳細については、『-sqdb パラメーター』を参照してください。

デフォルトでは、ファイルからメッセージへの転送で作成されるメッセージは、永続メッセージになります。 そのメッセージを非永続メッセージに設定したり、永続性の値を宛先キューで定義したりすることも可能です。

ファイルを複数のメッセージに分割するように指定すると、同じファイルから作成されるすべてのメッセージには、同じ IBM MQ グループ ID が割り当てられます。 ファイルを複数のメッセージに分割するように指定しない場合は、1 つのメッセージのみがファイルから作成され、このメッセージには、IBM MQ グループ ID が設定されません。

ファイルを大きいメッセージに転送する場合、または多数の小さいメッセージに転送する場合は、IBM MQ または Managed File Transfer の一部のプロパティーに変更が必要になる場合があります。 詳しくは、メッ

セージ・サイズに関連する MQ 属性および MFT プロパティーを設定する際のガイダンスを参照してください。

注 : 宛先キューがクラスター・キューであるか、クラスター・キューの別名である場合には、エージェント・プロパティー enableClusterQueueInputOutput が true に設定されていなければ、キューへのファイルの転送時にエラー・メッセージを受け取ります。 詳しくは、宛先キューがクラスター・キューであるか、クラスター・キューの別名である場合の対処法を参照してください。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

264 ページの『例: 1 つのファイルから 1 つのメッセージへの転送』
fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

265 ページの『例: 1 つのファイルを長さによって複数のメッセージに分割する操作』
fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

269 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作』
1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

267 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作』
特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』
fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』
ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料

275 ページの『ファイルからメッセージへの転送の失敗』
ファイルからメッセージへの転送で、エージェントがファイル・データを宛先キューに書き込み始めた後に障害が発生すると、エージェントは、メッセージをコンシュームするアプリケーションに障害の発生を通知するためのメッセージをキューに書き込みます。

MFT が宛先キューに書き込むメッセージで設定する MQ メッセージ・プロパティー
メッセージ・サイズに関連する MQ 属性および MFT プロパティーを設定する際のガイダンス

ファイルからメッセージへの転送を実行するためのエージェントの構成

エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

このタスクについて

enableQueueInputOutput プロパティーが true に設定されていない宛先エージェントに対して、ファイルからメッセージへの転送を実行しようとすると、その転送は失敗します。 調整キュー・マネージャーにパブリッシュされる転送ログ・メッセージには、以下のメッセージが組み込まれます。

```
BFGIO0197E: An attempt to write to a queue was rejected by the destination agent. The
agent must have enableQueueInputOutput=true set in the agent.properties file to
support transferring to a queue.
```

エージェントがキューへの書き込みと読み取りを行えるようにするには、以下のステップを実行します。

手順

1. **fteStopAgent** コマンドを使用して宛先エージェントを停止します。
2. agent.properties ファイルを編集して、enableQueueInputOutput=true という行を組み込みます。

   agent.properties ファイルは、ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*destination_agent_name* に置かれています。
3. オプション: agent.properties ファイルを編集して、enableClusterQueueInputOutput=true という行を組み込みます。 agent.properties ファイルは、ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*destination_agent_name* に置かれています。
4. **fteStartAgent** コマンドを使用して宛先エージェントを開始します。

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

264 ページの『例: 1 つのファイルから 1 つのメッセージへの転送』
fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

265 ページの『例: 1 つのファイルを長さによって複数のメッセージに分割する操作』
fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

269 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作』
1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

267 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作』

特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』
fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』
ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

関連資料
fteStopAgent
fteStartAgent
MFT agent.properties ファイル
275 ページの『ファイルからメッセージへの転送の失敗』
ファイルからメッセージへの転送で、エージェントがファイル・データを宛先キューに書き込み始めた後に障害が発生すると、エージェントは、メッセージをコンシュームするアプリケーションに障害の発生を通知するためのメッセージをキューに書き込みます。

例: 1 つのファイルから 1 つのメッセージへの転送

fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

このタスクについて

ソース・ファイルは /tmp/single_record.txt と呼ばれ、ソース・エージェント AGENT_NEPTUNE と同じシステム上にあります。 ソース・エージェント AGENT_NEPTUNE はキュー・マネージャー QM_NEPTUNE を使用します。 宛先エージェントは AGENT_VENUS で、このエージェントはキュー・マネージャー QM_VENUS に接続します。 宛先キュー RECEIVING_QUEUE は、キュー・マネージャー QM_MERCURY にあります。 QM_MERCURY は、キュー・マネージャー QM_VENUS と同じ IBM MQ ネットワークにあり、そのキュー・マネージャーからのアクセスが可能です。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -dm QM_VENUS
                  -dq RECEIVING_QUEUE@QM_MERCURY /tmp/single_record.txt
```

宛先エージェントが使用するキュー・マネージャーとは異なるキュー・マネージャーに宛先キューがある場合は、次の形式で **-dq** パラメーターの値を指定する必要があります。
queue_name@*queue_manager_name* この値に @*queue_manager_name* を指定しない場合、宛先エージェントは宛先キューが宛先エージェント・キュー・マネージャーにあると想定します。 例外は、enableClusterQueueInputOutput エージェント・プロパティーが true に設定されている場合です。 この場合、宛先エージェントは標準の IBM MQ 解決手順を使用して、キューが置かれている場所を判別します。

ソース・エージェントの AGENT_NEPTUNE は、ファイル /tmp/single_record.txt からデータを読み取り、このデータを宛先エージェント AGENT_VENUS に転送します。 宛先エージェント AGENT_VENUS は

そのデータを RECEIVING_QUEUE@QM_MERCURY キューにある永続メッセージに送信します。 メッセージには、IBM MQ グループ ID が設定されていません。

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

265 ページの『例: 1 つのファイルを長さによって複数のメッセージに分割する操作』
fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

269 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作』
1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

267 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作』
特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』
fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』
ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料

275 ページの『ファイルからメッセージへの転送の失敗』
ファイルからメッセージへの転送で、エージェントがファイル・データを宛先キューに書き込み始めた後に障害が発生すると、エージェントは、メッセージをコンシュームするアプリケーションに障害の発生を通知するためのメッセージをキューに書き込みます。

例: 1 つのファイルを長さによって複数のメッセージに分割する操作

fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

このタスクについて

ソース・ファイルは /tmp/source.file と呼ばれ、サイズは 36 KB です。 ソース・ファイルは、ソース・エージェント AGENT_NEPTUNE と同じシステムにあります。 ソース・エージェント AGENT_NEPTUNE はキュー・マネージャー QM_NEPTUNE に接続します。 宛先エージェントは AGENT_MERCURY で、このエージェントはキュー・マネージャー QM_MERCURY に接続します。 宛先キュー RECEIVING_QUEUE もキュー・マネージャー QM_MERCURY にあります。 この転送では、ソース・ファイルを 1 KB のサイズのいくつかのセクションに分割し、各セクションを RECEIVING_QUEUE のメッセージに書き込みます。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
                  -dq RECEIVING_QUEUE -qs 1K /tmp/source.file
```

ソース・エージェントの AGENT_NEPTUNE は、ファイル /tmp/source.file からデータを読み取り、このデータを宛先エージェント AGENT_MERCURY に転送します。 宛先エージェント AGENT_MERCURY はそのデータを RECEIVING_QUEUE@QM_MERCURY キューの 36 個の 1 KB 永続メッセージに書き込みます。 メッセージは、すべて同じ IBM MQ グループ ID を持ち、グループの最後のメッセージは IBM MQ LAST_MSG_IN_GROUP フラグ・セットを持ちます。

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

264 ページの『例: 1 つのファイルから 1 つのメッセージへの転送』
fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

269 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作』
1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

267 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作』
特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』
fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』

ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料
275 ページの『ファイルからメッセージへの転送の失敗』
ファイルからメッセージへの転送で、エージェントがファイル・データを宛先キューに書き込み始めた後に障害が発生すると、エージェントは、メッセージをコンシュームするアプリケーションに障害の発生を通知するためのメッセージをキューに書き込みます。

例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作

特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

このタスクについて

ファイルを可変長のセクションに分割し、各セクションをそれぞれのメッセージに書き込みます。 特定の正規表現にマッチングするテキストの地点でテキスト・ファイルを分割します。 ソース・ファイルは `/tmp/names.text` と呼ばれ、以下の内容が含まれています。

```
Jenny Jones,John Smith,Jane Brown
```

ファイルを分割する箇所を指定する正規表現は、コンマ文字 (`,`) です。

ソース・ファイルは、キュー・マネージャー QM_NEPTUNE に接続しているソース・エージェント *AGENT_NEPTUNE* と同じシステムにあります。 宛先キュー *RECEIVING_QUEUE* は、キュー・マネージャー *QM_MERCURY* にあります。 *QM_MERCURY* は、宛先エージェント *AGENT_MERCURY* が使用するキュー・マネージャーでもあります。 この転送では、ソース・ファイルをいくつかのセクションに分割し、各セクションを *RECEIVING_QUEUE* のメッセージに書き込みます。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
                  -dq RECEIVING_QUEUE -t text -dqdp postfix -dqdt "," /tmp/names.text
```

ソース・エージェントの AGENT_NEPTUNE は、ファイル `/tmp/names.text` からデータを読み取り、このデータを宛先エージェント AGENT_MERCURY に転送します。 宛先エージェント AGENT_MERCURY はデータをキュー *RECEIVING_QUEUE* の 3 つの永続メッセージに書き込みます。 メッセージは、すべて同じ IBM MQ グループ ID を持ち、グループの最後のメッセージは IBM MQ LAST_MSG_IN_GROUP フラグ・セットを持ちます。

メッセージ内のデータは次のとおりです。

- 1 番目のメッセージ:

  ```
  Jenny Jones
  ```

- 2 番目のメッセージ:

  ```
  John Smith
  ```

- 3 番目のメッセージ:

```
Jane Brown
```

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

264 ページの『例: 1 つのファイルから 1 つのメッセージへの転送』
fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

265 ページの『例: 1 つのファイルを長さによって複数のメッセージに分割する操作』
fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

269 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作』
1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』
fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』
ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料

275 ページの『ファイルからメッセージへの転送の失敗』
ファイルからメッセージへの転送で、エージェントがファイル・データを宛先キューに書き込み始めた後に障害が発生すると、エージェントは、メッセージをコンシュームするアプリケーションに障害の発生を通知するためのメッセージをキューに書き込みます。

MFT が使用する正規表現

例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作

1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

このタスクについて

1 つのテキスト・ファイルを 1 つのキューにある複数のメッセージに転送します。 ファイルを可変長のセクションに分割し、各セクションをそれぞれのメッセージに書き込みます。 特定の正規表現にマッチングするテキストの地点でテキスト・ファイルを分割します。 ソース・ファイルは /tmp/customers.text と呼ばれ、以下の内容が含まれています。

```
Customer name: John Smith
Customer contact details: john@example.net
Customer number: 314

Customer name: Jane Brown
Customer contact details: jane@example.com
Customer number: 42

Customer name: James Jones
Customer contact details: jjones@example.net
Customer number: 26
```

ファイルを分割する場所を指定する正規表現は、 Customer\snumber:\s\d+ で、"Customer number: " : の後に任意の桁数の数字が続くテキストにマッチします。 コマンド行で指定する正規表現は、コマンド・シェルによって評価されないようにするために、二重引用符で囲む必要があります。 正規表現は Java 正規表現として評価されます。 詳しくは、MFT が使用する正規表現を参照してください。

デフォルトでは、正規表現にマッチング可能な文字の数は、5 個に設定されています。 この例で使用する正規表現にマッチングするのは、5 文字より長いストリングです。 5 文字より長いマッチング項目を許可するには、エージェント・プロパティー・ファイルを編集して、**maxDelimiterMatchLength** プロパティーを組み込みます。

デフォルトでは、正規表現にマッチングするテキストは、メッセージに組み込まれません。 この例のように、正規表現にマッチングするテキストをメッセージに組み込むには、**-qi** パラメーターを使用します。 ソース・ファイルは、キュー・マネージャー QM_NEPTUNE に接続しているソース・エージェント AGENT_NEPTUNE と同じシステムにあります。 宛先キュー RECEIVING_QUEUE は、キュー・マネージャー QM_MERCURY にあります。 QM_MERCURY は、宛先エージェント AGENT_MERCURY が使用するキュー・マネージャーでもあります。 この転送では、ソース・ファイルをいくつかのセクションに分割し、各セクションを RECEIVING_QUEUE のメッセージに書き込みます。

手順

1. 次のコマンドを使用して、宛先エージェントを停止します。

   ```
   fteStopAgent AGENT_MERCURY
   ```

2. AGENT_MERCURY のエージェント・プロパティー・ファイルに以下の行を追加します。

   ```
   maxDelimiterMatchLength=25
   ```

   注 : maxDelimiterMatchLength の値を大きくすると、パフォーマンスが低下する可能性があります。
3. 次のコマンドを使用して、宛先エージェントを開始します。

   ```
   fteStartAgent AGENT_MERCURY
   ```

4. 次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
 -dq RECEIVING_QUEUE
     text -dqdt "Customer\snumber:\s\d+" -qi -dqdp postfix /tmp/customers.text
```

ソース・エージェントの AGENT_NEPTUNE は、ファイル /tmp/customers.text からデータを読み取り、このデータを宛先エージェント AGENT_MERCURY に転送します。 宛先エージェント AGENT_MERCURY はそのデータをキュー RECEIVING_QUEUE の 3 つの永続メッセージに書き込みます。 メッセージは、すべて同じ IBM MQ グループ ID を持ち、グループの最後のメッセージは IBM MQ LAST_MSG_IN_GROUP フラグ・セットを持ちます。

メッセージ内のデータは次のとおりです。

- 1 番目のメッセージ:

```
Customer name: John Smith
Customer contact details: john@example.net
Customer number: 314
```

- 2 番目のメッセージ:

```
Customer name: Jane Brown
Customer contact details: jane@example.com
Customer number: 42
```

- 3 番目のメッセージ:

```
Customer name: James Jones
Customer contact details: jjones@example.net
Customer number: 26
```

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

264 ページの『例: 1 つのファイルから 1 つのメッセージへの転送』
fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

265 ページの『例: 1 つのファイルを長さによって複数のメッセージに分割する操作』
fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

267 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作』
特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』
fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ

メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』
ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料

MFT `agent.properties` ファイル

MFT が使用する正規表現

例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定

fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

このタスクについて

fteCreateTransfer コマンドに `-qmp true` パラメーターを組み込みます。 この例では、コマンドを実行依頼するユーザーの MQMD ユーザー ID は `larmer` です。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM -qmp true
      -t text /tmp/source_file.txt
```

宛先エージェント AGENT_SATURN によってキュー・マネージャー MyQM のキュー MY_QUEUE に書き込まれる最初のメッセージの IBM MQ メッセージ・プロパティーは、以下の値に設定されます。

```
usr.WMQFTETransferId=414cbaedefa234889d999a8ed09782395ea213ebbc9377cd
usr.WMQFTETransferMode=text
usr.WMQFTESourceAgent=AGENT_JUPITER
usr.WMQFTEDestinationAgent=AGENT_SATURN
usr.WMQFTEFileName=source_file.txt
usr.WMQFTEFileSize=1024
usr.WMQFTEFileLastModified=1273740879040
usr.WMQFTEFileIndex=0
usr.WMQFTEMqmdUser=larmer
```

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput

を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

264 ページの『例: 1 つのファイルから 1 つのメッセージへの転送』
fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

265 ページの『例: 1 つのファイルを長さによって複数のメッセージに分割する操作』
fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

269 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作』
1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

267 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作』
特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』
ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料

275 ページの『ファイルからメッセージへの転送の失敗』
ファイルからメッセージへの転送で、エージェントがファイル・データを宛先キューに書き込み始めた後に障害が発生すると、エージェントは、メッセージをコンシュームするアプリケーションに障害の発生を通知するためのメッセージをキューに書き込みます。

MFT が宛先キューに書き込むメッセージで設定する MQ メッセージ・プロパティー

例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定

ユーザー定義のメタデータが、転送で宛先キューに書き込まれる最初のメッセージで、IBM MQ メッセージ・プロパティーとして設定されます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

このタスクについて

パラメーター `-qmp true` および `-md account=123456` を **fteCreateTransfer** コマンドに組み込んで、 RFH2 ヘッダーの `usr.account` プロパティーを 123456 に設定します。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM
                  -qmp true -md account=123456 /tmp/source_file.txt
```

IBM MQ メッセージ・プロパティーの標準セットに加えて、ユーザー定義のプロパティーが、最初のメッセージのメッセージ・ヘッダーに設定されます。その最初のメッセージは、宛先エージェント AGENT_SATURN により、キュー・マネージャー MyQM 上のキュー MY_QUEUE に書き込まれるものです。 ヘッダーは次の値に設定されます。

```
usr.account=123456
```

ユーザー定義のメタデータの名前の先頭には、接頭部 usr が追加されます。

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

264 ページの『例: 1 つのファイルから 1 つのメッセージへの転送』
fteCreateTransfer コマンドで **-dq** パラメーターを使用することにより、ファイル転送の宛先にするキューを指定できます。 ソース・ファイルは、宛先キューで設定されている最大メッセージ長より小さいサイズでなければなりません。 宛先キューは、宛先エージェントが接続するキュー・マネージャーと同じキュー・マネージャーにある必要はありませんが、これらの 2 つのキュー・マネージャー同士が通信できなければなりません。

265 ページの『例: 1 つのファイルを長さによって複数のメッセージに分割する操作』
fteCreateTransfer コマンドの **-qs** パラメーターを使用して、1 つのファイルを複数の IBM MQ メッセージに分割することができます。 ファイルを固定長の各セクションに分割し、各セクションをそれぞれのメッセージに書き込みます。

269 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを分割し、その区切り文字をメッセージに組み込む操作』
1 つのテキスト・ファイルを、所定の Java 正規表現にマッチングするそれぞれの箇所で分割し、その正規表現の一致を結果に含めて、複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターと **-qi** パラメーターを使用します。

267 ページの『例: 正規表現区切り文字を使用してテキスト・ファイルを複数のメッセージに分割する操作』
特定の Java 正規表現にマッチングするそれぞれの箇所で 1 つのテキスト・ファイルを分割して複数のメッセージに転送します。 そのために、**fteCreateTransfer** コマンドの **-dqdt** パラメーターを使用します。

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』
fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料

MFT が宛先キューに書き込むメッセージで設定する MQ メッセージ・プロパティー

例: ファイルからメッセージへの転送のためのユーザー定義メッセージ・プロパティーの追加

メッセージからファイルへの管理対象転送に Managed File Transfer を使用する場合には、結果のメッセージにユーザー定義のメッセージ・プロパティーを含めることができます。

このタスクについて

カスタム・メッセージ・プロパティーを定義するために、以下のいずれかの方式を使用することができます。

- 転送要求に **-md** パラメーターを指定します。 詳しくは、272 ページの『例: ファイルからメッセージへの転送に関するユーザー定義プロパティーの設定』を参照してください。
- Ant タスクを使用します。fte:filecopy または fte:filemove のいずれかを使用できます。 以下の例は fte:filecopy タスクです。

```
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">
<!-- Initialise the properties used in this script.-->

<target name="init" description="initialise task properties">
               <property name="src.file" value="/home/user/file1.bin"/>
               <property name="dst.queue" value="TEST.QUEUE@qm2"/>
               <fte:uuid property="job.name" length="8"
prefix="copyjob#"/>
</target>
<target name="step1" depends="init" description="transfer file">

<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
             src="agent1@qm1" dst="agent2@qm2"
            rcproperty="copy.result">

<fte:metadata>
<fte:entry name="fileName" value="${FileName}"/>
</fte:metadata>

<fte:filespec srcfilespec="${src.file}" dstqueue="${dst.queue}"
dstmsgprops="true"/>

</fte:filecopy>

</target>
</project>
```

- リソース・モニターと変数置換を使用します。 以下の例は、転送タスク XML を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor
xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="5.00"
xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinit ion ./Monitor.xsd">
 <name>METADATA</name>
 <pollInterval units="minutes">5</pollInterval>
 <batch maxSize="5"/>
 <agent>AGENT1</agent>
 <resources>
   <directory recursionLevel="0">e:\temp</directory>
 </resources>
 <triggerMatch>
   <conditions>
     <allOf>
       <condition>
         <fileMatch>
           <pattern>*.txt</pattern>
         </fileMatch>
       </condition>
     </allOf>
   </conditions>
 </triggerMatch>
 <tasks>
   <task>
     <name/>
```

```
     <transfer>
       <request version="5.00"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
         <managedTransfer>
           <originator>
             <hostName>mqjason.raleigh.ibm.com.</hostName>
             <userID>administrator</userID>
           </originator>
           <sourceAgent QMgr="AGENTQM" agent="AGENT1"/>
           <destinationAgent QMgr="AGENTQM" agent="AGENT2"/>
           <transferSet priority="0">
             <metaDataSet>
               <metaData key="FileName">${FileName}</metaData>
             </metaDataSet>
             <item checksumMethod="MD5" mode="text">
               <source disposition="delete" recursive="false">
                 <file>${FilePath}</file>
               </source>
               <destination type="queue">
                 <queue persistent="true"
setMqProps="true">TEST.QUEUE@AGENTQM</queue>
               </destination>
             </item>
           </transferSet>
           <job>
             <name>Metadata_example</name>
           </job>
         </managedTransfer>
       </request>
     </transfer>
   </task>
 </tasks>
 <originator>
   <hostName>mqjason.raleigh.ibm.com.</hostName>
   <userID>administrator</userID>
 </originator>
</monitor:monitor>
```

関連タスク

271 ページの『例: ファイルからメッセージへの転送に関する IBM MQ メッセージ・プロパティーの設定』

fteCreateTransfer コマンドの **-qmp** パラメーターを使用して、転送によって宛先キューに書き込まれる最初のメッセージに IBM MQ メッセージ・プロパティーを設定するかどうかを指定できます。 IBM MQ メッセージ・プロパティーを使用すれば、アプリケーションで IBM MQ メッセージ記述子 (MQMD) または MQRFH2 ヘッダーにアクセスしなくても、処理対象のメッセージを選択したり、メッセージに関する情報を取得したりすることが可能になります。

関連資料

fte:filecopy の Ant タスク

fte:filemove の Ant タスク

ファイルからメッセージへの転送の失敗

ファイルからメッセージへの転送で、エージェントがファイル・データを宛先キューに書き込み始めた後に障害が発生すると、エージェントは、メッセージをコンシュームするアプリケーションに障害の発生を通知するためのメッセージをキューに書き込みます。

障害が発生した場合、以下のようなメッセージが宛先キューに書き込まれます。

- 内容はブランクです
- エージェントが宛先キューに書き込んだ直前のメッセージと同じ IBM MQ グループ ID が付きます
- IBM MQ の LAST_MSG_IN_GROUP フラグが設定されます
- メッセージ・プロパティーが有効になっている場合は、追加の IBM MQ メッセージ・プロパティーが組み込まれています。 詳しくは、MFT が宛先キューに書き込むメッセージで設定する MQ メッセージ・プロパティーのトピックを参照してください。

例

以下のコマンドを実行して転送を要求します。

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq RECEIVING_QUEUE
                  -qmp true -qs 1K /tmp/source1.txt
```

ファイル source1.txt は 48 KB です。 この転送では、このファイルを 1 KB のメッセージに分割し、それらのメッセージを宛先キュー RECEIVING_QUEUE に書き込みます。

転送の進行中、エージェントが 16 個のメッセージを RECEIVING_QUEUE に書き込んだ後に、ソース・エージェントで障害が発生します。

エージェントは、ブランクのメッセージを RECEIVING_QUEUE に書き込みます。 ブランクのメッセージでは、メッセージ・プロパティーの標準セットに加えて、以下のメッセージ・プロパティーが設定されます。

```
usr.WMQFTEResultCode = 40
usr.WMQFTESupplement = BFGTR0036I: The transfer failed to complete successfully.
```

IBM MQ 9.3.0 以降、区切り文字サイズ検査エラーのためにファイルからの転送が失敗すると、空のメッセージが 1 つだけ送信されるようになりました。 さらに、転送の失敗が宛先エージェント上での区切り文字設定サイズ超過によるものだった場合は、ユーザー・プロパティーがこのメッセージに追加されます。

関連概念

261 ページの『ファイルからメッセージへのデータ転送』
Managed File Transfer のファイルからメッセージへの転送機能を使用すれば、1 つのファイルにあるデータを IBM MQ のキューにある 1 つまたは複数のメッセージに転送できます。

関連タスク

263 ページの『ファイルからメッセージへの転送を実行するためのエージェントの構成』
エージェントは、デフォルトで、ファイルからメッセージへの転送またはメッセージからファイルへの転送を実行できません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。 IBM MQ・クラスター・キューへの書き込みを有効にするには、エージェント・プロパティー enableClusterQueueInputOutput も true に設定する必要があります。

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

関連資料

MFT agent.properties ファイル

MFT が宛先キューに書き込むメッセージで設定する MQ メッセージ・プロパティー

メッセージからファイルへのデータ転送

Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

ファイルからメッセージへの転送に関しては、261 ページの『ファイルからメッセージへのデータ転送』を参照してください。

重要 : メッセージからファイルへの転送のソース・エージェントは、プロトコル・ブリッジ・エージェントまたは Connect:Direct ブリッジ・エージェントであることはできません。

IBM MQ のメッセージ・データをファイルに転送できます。 メッセージからファイルへの転送では、以下のタイプの転送がサポートされています。

- 1 つのメッセージから 1 つのファイルへ
- 複数のメッセージから 1 つのファイルへ
- IBM MQ グループ ID が同じ複数のメッセージから 1 つのファイルへ
- 複数のメッセージから 1 つのファイルへ (各メッセージのデータの間にあるテキスト区切り文字またはバイナリー区切り文字をファイルに書き込みます)

ファイルを大きいメッセージから転送する場合、または多数の小さいメッセージから転送する場合は、IBM MQ または Managed File Transfer の一部のプロパティーに変更が必要になる場合があります。 メッセージ・サイズに関連した MQ 属性および MFT プロパティーを設定するためのガイダンス.を見て新しい情報を手に入れよう。

メッセージからファイルへの転送では、ソース・エージェントは、以前のバージョンの IBM MQ での破壊的な GET とは異なり、ソース・キューからメッセージを参照します。 すべてのメッセージ (メッセージのグループ化が使用されている場合はグループ内のすべてのメッセージ) が参照され、データが宛先ファイルに書き込まれた後に、メッセージはソース・キューから削除されます。 これにより、転送が失敗したり、キャンセルされたりした場合にメッセージがソース・キューに残ることができます。 この変更のため、メッセージからファイルへの転送を実行するには、GET 権限と BROWSE 権限が必要になります。

Managed File Transfer は、転送 ID と、転送要求 XML ペイロード内の groupId 属性の値を比較します。 比較した 2 つの ID が等しい場合、ソース・エージェントはその ID を、メッセージからファイルへの転送のための入力キューに対して行われる 1 回目の MQGET の試行で、メッセージ ID のマッチ・オプション (グループ ID のマッチ・オプションと対照) として使用します。

関連タスク

238 ページの『例: MFT リソースの構成』

fteCreateMonitor コマンドで **-mq** パラメーターを使用することにより、リソース・モニターによってモニターされるリソースとして IBM MQ キューを指定できます。

関連資料

MFT がソース・キューのメッセージから読み取る MQ メッセージ・プロパティー

メッセージ・サイズに関連する MQ 属性および MFT プロパティーを設定する際のガイダンス

メッセージからファイルへの転送を実行するためのエージェントの構成

デフォルトでは、エージェントがメッセージからファイルへの転送またはファイルからメッセージへの転送を実行することはできません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。

このタスクについて

enableQueueInputOutput プロパティーが true に設定されていないソース・エージェントから、メッセージからファイルへの転送を実行しようとすると、その転送は失敗します。 調整キュー・マネージャーにパブリッシュされる転送ログ・メッセージには、以下のメッセージが組み込まれます。

```
BFGIO0197E: An attempt to read from a queue was rejected by the source agent.
The agent must have enableQueueInputOutput=true set in the agent.properties file
to support transferring from a queue.
```

エージェントがキューへの書き込みと読み取りを行えるようにするには、以下のステップを実行します。

手順

1. **fteStopAgent** コマンドを使用してソース・エージェントを停止します。
2. agent.properties ファイルを編集して、enableQueueInputOutput=true という行を組み込みます。

   agent.properties ファイルは、ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*source_agent_name* に置かれています。
3. **fteStartAgent** コマンドを使用してソース・エージェントを開始します。

関連概念

276 ページの『メッセージからファイルへのデータ転送』

Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケ

ーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

関連タスク

278 ページの『例: 1 つのキューから 1 つのファイルへの転送』
fteCreateTransfer コマンドで **-sq** パラメーターを使用することにより、ファイル転送のソースとして IBM MQ キューを指定できます。

279 ページの『例: キューにあるメッセージのグループを 1 つのファイルに転送する操作』
fteCreateTransfer コマンドで **-sq** パラメーターおよび **-sqgi** パラメーターを使用することにより、IBM MQ キュー上の単一の完全なグループをファイル転送のソースとして指定できます。

280 ページの『例: 各メッセージのデータの前にテキスト区切り文字を挿入する操作』
テキスト・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdt** および **-sqdp** パラメーターを使用して、個々のメッセージのデータの前にテキスト区切り文字を挿入する動作を指定できます。

282 ページの『例: 各メッセージのデータの後にバイナリー区切り文字を挿入する操作』
バイナリー・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdb**、および **-sqdp** パラメーターを使用して、個々のメッセージのデータの後にバイナリー区切り文字を挿入する動作を指定できます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

285 ページの『例: IBM MQ メッセージ・プロパティーを使用したメッセージからファイルへの転送の失敗』
`usr.UserReturnCode` IBM MQ メッセージ・プロパティーをゼロ以外の値に設定することによって、メッセージからファイルへの転送を失敗させることができます。 さらに、`usr.UserSupplement` IBM MQ メッセージ・プロパティーを設定することによって、失敗の理由に関する補足情報を指定することもできます。

関連資料

MFT `agent.properties` ファイル

例: 1 つのキューから 1 つのファイルへの転送

fteCreateTransfer コマンドで **-sq** パラメーターを使用することにより、ファイル転送のソースとして IBM MQ キューを指定できます。

このタスクについて

START_QUEUE キューにある 3 個のメッセージにソース・データが格納されています。 このキューは、ソース・エージェントのキュー・マネージャー *QM_NEPTUNE* に存在していなければなりません。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE
                  -da AGENT_VENUS -df /out/three_to_one.txt
                  -sq START_QUEUE
```

START_QUEUE キューにあるメッセージのデータが、`AGENT_VENUS` を実行しているシステムの `/out/three_to_one.txt` ファイルに書き込まれます。

関連概念

276 ページの『メッセージからファイルへのデータ転送』

Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

関連タスク

277 ページの『メッセージからファイルへの転送を実行するためのエージェントの構成』
デフォルトでは、エージェントがメッセージからファイルへの転送またはファイルからメッセージへの転送を実行することはできません。 この機能を有効にするには、エージェント・プロパティー `enableQueueInputOutput` を true に設定する必要があります。

279 ページの『例: キューにあるメッセージのグループを 1 つのファイルに転送する操作』
fteCreateTransfer コマンドで **-sq** パラメーターおよび **-sqgi** パラメーターを使用することにより、IBM MQ キュー上の単一の完全なグループをファイル転送のソースとして指定できます。

280 ページの『例: 各メッセージのデータの前にテキスト区切り文字を挿入する操作』
テキスト・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdt** および **-sqdp** パラメーターを使用して、個々のメッセージのデータの前にテキスト区切り文字を挿入する動作を指定できます。

282 ページの『例: 各メッセージのデータの後にバイナリー区切り文字を挿入する操作』
バイナリー・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdb**、および **-sqdp** パラメーターを使用して、個々のメッセージのデータの後にバイナリー区切り文字を挿入する動作を指定できます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

285 ページの『例: IBM MQ メッセージ・プロパティーを使用したメッセージからファイルへの転送の失敗』
`usr.UserReturnCode` IBM MQ メッセージ・プロパティーをゼロ以外の値に設定することによって、メッセージからファイルへの転送を失敗させることができます。 さらに、`usr.UserSupplement` IBM MQ メッセージ・プロパティーを設定することによって、失敗の理由に関する補足情報を指定することもできます。

関連資料

MFT がソース・キューのメッセージから読み取る MQ メッセージ・プロパティー
fteCreateTransfer: 新規ファイル転送の開始

例: キューにあるメッセージのグループを 1 つのファイルに転送する操作

fteCreateTransfer コマンドで **-sq** パラメーターおよび **-sqgi** パラメーターを使用することにより、IBM MQ キュー上の単一の完全なグループをファイル転送のソースとして指定できます。

このタスクについて

この例では、START_QUEUE キューに 10 個のメッセージがあるとします。 このキューは、ソース・エージェントのキュー・マネージャー *QM_NEPTUNE* に存在していなければなりません。 最初の 3 個のメッセージは、IBM MQ グループ ID `41424b3ef3a2202020202020202020202020202020201111` のグループに属しています。このグループは、完全に揃ったグループではありません。 次の 5 個のメッセージは、IBM MQ グループ ID `41424b3ef3a2202020202020202020202020202020202222` のグループに属しています。このグループは、完全に揃ったグループです。 残りの 2 個のメッセージは、IBM MQ グループ ID `41424b3ef3a2202020202020202020202020202020203333` のグループに属しています。このグループは、完全に揃っています。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS
                  -df /out/group.txt -sqgi -sq START_QUEUE
```

キュー *START_QUEUE* の最初の完全なグループ (IBM MQ グループ ID 41424b3ef3a2202020202020202020202020202020202222 のグループ) に属するメッセージ内のデータは、 AGENT_VENUS が実行されているシステム上のファイル /out/group.txt に書き込まれます。

関連概念

276 ページの『メッセージからファイルへのデータ転送』
Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

関連タスク

277 ページの『メッセージからファイルへの転送を実行するためのエージェントの構成』
デフォルトでは、エージェントがメッセージからファイルへの転送またはファイルからメッセージへの転送を実行することはできません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。

278 ページの『例: 1 つのキューから 1 つのファイルへの転送』
fteCreateTransfer コマンドで **-sq** パラメーターを使用することにより、ファイル転送のソースとして IBM MQ キューを指定できます。

280 ページの『例: 各メッセージのデータの前にテキスト区切り文字を挿入する操作』
テキスト・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdt** および **-sqdp** パラメーターを使用して、個々のメッセージのデータの前にテキスト区切り文字を挿入する動作を指定できます。

282 ページの『例: 各メッセージのデータの後にバイナリー区切り文字を挿入する操作』
バイナリー・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdb**、および **-sqdp** パラメーターを使用して、個々のメッセージのデータの後にバイナリー区切り文字を挿入する動作を指定できます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

285 ページの『例: IBM MQ メッセージ・プロパティーを使用したメッセージからファイルへの転送の失敗』
usr.UserReturnCode IBM MQ メッセージ・プロパティーをゼロ以外の値に設定することによって、メッセージからファイルへの転送を失敗させることができます。 さらに、usr.UserSupplement IBM MQ メッセージ・プロパティーを設定することによって、失敗の理由に関する補足情報を指定することもできます。

関連資料

fteCreateTransfer: 新規ファイル転送の開始

例: 各メッセージのデータの前にテキスト区切り文字を挿入する操作

テキスト・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdt** および **-sqdp** パラメーターを使用して、個々のメッセージのデータの前にテキスト区切り文字を挿入する動作を指定できます。

このタスクについて

この例では、START_QUEUE キューに 4 個のメッセージがあるとします。 このキューは、ソース・エージェントのキュー・マネージャー *QM_NEPTUNE* にあります。 各メッセージからのデータの前に挿入されるテキスト区切り文字は、 Java リテラル・ストリングとして表すことができます (例: `\n\u002D\u002D\u002D\n`)。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -df /out/output.txt
                  -t text -sqdt "\n\u002D\u002D\u002D\n" -sqdp prefix -sq START_QUEUE
```

ソース・エージェント AGENT_NEPTUNE によって、START_QUEUE にある 4 個のメッセージそれぞれのデータの先頭にテキスト区切り文字が追加されます。 このデータは、宛先ファイル `/out/output.txt` に書き込まれます。

関連概念

276 ページの『メッセージからファイルへのデータ転送』
Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

関連タスク

277 ページの『メッセージからファイルへの転送を実行するためのエージェントの構成』
デフォルトでは、エージェントがメッセージからファイルへの転送またはファイルからメッセージへの転送を実行することはできません。 この機能を有効にするには、エージェント・プロパティー `enableQueueInputOutput` を true に設定する必要があります。

278 ページの『例: 1 つのキューから 1 つのファイルへの転送』
fteCreateTransfer コマンドで **-sq** パラメーターを使用することにより、ファイル転送のソースとして IBM MQ キューを指定できます。

279 ページの『例: キューにあるメッセージのグループを 1 つのファイルに転送する操作』
fteCreateTransfer コマンドで **-sq** パラメーターおよび **-sqgi** パラメーターを使用することにより、IBM MQ キュー上の単一の完全なグループをファイル転送のソースとして指定できます。

282 ページの『例: 各メッセージのデータの後にバイナリー区切り文字を挿入する操作』
バイナリー・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdb**、および **-sqdp** パラメーターを使用して、個々のメッセージのデータの後にバイナリー区切り文字を挿入する動作を指定できます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

285 ページの『例: IBM MQ メッセージ・プロパティーを使用したメッセージからファイルへの転送の失敗』
`usr.UserReturnCode` IBM MQ メッセージ・プロパティーをゼロ以外の値に設定することによって、メッセージからファイルへの転送を失敗させることができます。 さらに、`usr.UserSupplement` IBM MQ メッセージ・プロパティーを設定することによって、失敗の理由に関する補足情報を指定することもできます。

関連資料

fteCreateTransfer: 新規ファイル転送の開始

例: 各メッセージのデータの後にバイナリー区切り文字を挿入する操作

バイナリー・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdb**、および **-sqdp** パラメーターを使用して、個々のメッセージのデータの後にバイナリー区切り文字を挿入する動作を指定できます。

このタスクについて

この例では、START_QUEUE キューに 3 個のメッセージがあるとします。 このキューは、ソース・エージェントのキュー・マネージャー *QM_NEPTUNE* にあります。 各メッセージのデータの後に挿入するバイナリー区切り文字は、16 進数バイトのコンマ区切りリストとして記述する必要があります (例: x34,xE7,xAE)。

手順

次のコマンドを入力します。

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -df /out/binary.file
                  -sqdp postfix -sqdb x34,xE7,xAE -sq START_QUEUE
```

ソース・エージェント AGENT_NEPTUNE によって、START_QUEUE にある 3 個のメッセージそれぞれのデータの後にバイナリー区切り文字が追加されます。 このデータは、宛先ファイル `/out/binary.file` に書き込まれます。

関連概念

276 ページの『メッセージからファイルへのデータ転送』
Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

関連タスク

277 ページの『メッセージからファイルへの転送を実行するためのエージェントの構成』
デフォルトでは、エージェントがメッセージからファイルへの転送またはファイルからメッセージへの転送を実行することはできません。 この機能を有効にするには、エージェント・プロパティー `enableQueueInputOutput` を true に設定する必要があります。

278 ページの『例: 1 つのキューから 1 つのファイルへの転送』
fteCreateTransfer コマンドで **-sq** パラメーターを使用することにより、ファイル転送のソースとして IBM MQ キューを指定できます。

279 ページの『例: キューにあるメッセージのグループを 1 つのファイルに転送する操作』
fteCreateTransfer コマンドで **-sq** パラメーターおよび **-sqgi** パラメーターを使用することにより、IBM MQ キュー上の単一の完全なグループをファイル転送のソースとして指定できます。

280 ページの『例: 各メッセージのデータの前にテキスト区切り文字を挿入する操作』
テキスト・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdt** および **-sqdp** パラメーターを使用して、個々のメッセージのデータの前にテキスト区切り文字を挿入する動作を指定できます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

285 ページの『例: IBM MQ メッセージ・プロパティーを使用したメッセージからファイルへの転送の失敗』
`usr.UserReturnCode` IBM MQ メッセージ・プロパティーをゼロ以外の値に設定することによって、メッセージからファイルへの転送を失敗させることができます。 さらに、`usr.UserSupplement` IBM MQ メ

ッセージ・プロパティーを設定することによって、失敗の理由に関する補足情報を指定することもできます。

関連資料
fteCreateTransfer: 新規ファイル転送の開始

キューのモニターおよび変数置換の使用

fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

このタスクについて

この例では、ソース・エージェントは AGENT_VENUS という名前であり、QM_VENUS に接続します。 AGENT_VENUS がモニターするキューは START_QUEUE という名前であり、QM_VENUS にあります。 エージェントは、キューを 30 分おきにポーリングします。

メッセージの完全に揃ったグループがキューに書き込まれると、モニター・タスクは、いくつかの宛先エージェントの 1 つのファイルにメッセージのグループを送信します。この宛先エージェントは、すべてキュー・マネージャー QM_MARS に接続しています。 メッセージのグループが転送されるファイルの名前は、グループの最初のメッセージの IBM MQ メッセージ・プロパティー `usr.fileName` で定義します。 メッセージのグループが送信されるエージェントの名前は、グループの最初のメッセージの IBM MQ メッセージ・プロパティー `usr.toAgent` で定義します。 `usr.toAgent` ヘッダーが未設定の場合は、宛先エージェント用に使用されるデフォルト値は、AGENT_MAGENTA です。

`useGroups="true"` を指定する場合、`groupId="${GROUPID}"` を指定しないと、転送ではキュー内の最初のメッセージのみが取得されます。 そのため、例えば変数置換を使用して fileName を生成した場合、`a.txt` の内容が正しくなくなる可能性があります。 これは、fileName はモニターによって生成されますが、転送では、実際には fileName というファイルを生成するメッセージではなく、別のメッセージを取得するためです。

手順

1. モニター起動時にモニターが実行するタスクを定義するタスク XML を作成します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
     <destinationAgent agent="${toAgent}" QMgr="QM_MARS" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/${fileName}.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

   IBM MQ メッセージ・ヘッダーの値で置き換えられる変数は、**太字**で強調表示しています。 このタスク XML は、ファイル `/home/USER1/task.xml` に保存されます。

2. キュー START_QUEUE をモニターするリソース・モニターを作成します。
   以下のコマンドを実行依頼します。

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. ユーザーまたはプログラムは、メッセージのグループをキュー START_QUEUE に書き込みます。
   このグループの最初のメッセージは、次の IBM MQ メッセージ・プロパティーを設定しています。

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. 完全に揃ったグループが書き込まれると、モニターが起動されます。 エージェントは、IBM MQ メッセージ・プロパティーをタスク XML に置換します。
   この結果、タスク XML は以下のように変換されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
   <managedTransfer>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
     </originator>
     <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
     <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS" />
     <transferSet>
       <item mode="binary" checksumMethod="none">
         <source>
           <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
         </source>
         <destination type="file" exist="overwrite">
           <file>/reports/larmer.rpt</file>
         </destination>
       </item>
     </transferSet>
   </managedTransfer>
</request>
```

タスクの結果

タスク XML によって定義された転送が実行されます。 AGENT_VENUS によって START_QUEUE から読み取られるメッセージの完全なグループは、 AGENT_VIOLET が実行されているシステム上の `/reports/larmer.rpt` というファイルに書き込まれます。

次のタスク

各メッセージの別個のファイルへの転送

キューをモニターして、すべてのメッセージが別個のファイルに転送されるようにする場合には、このトピックで前述した方法と同様の方法を使用することができます。

1. **fteCreateMonitor** コマンドに **-tr completeGroups** パラメーターを指定して、前述のようにモニターを作成します。
2. タスク XML で、次のように指定します。

```
<queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
```

ただし、メッセージをソース・キューに入れる場合は、それらのメッセージを IBM MQ グループには入れないでください。 IBM MQ メッセージ・プロパティーを各メッセージに追加します。 例えば、メッセージごとに固有のファイル名の値を持つ `usr.filename` プロパティーを指定します。 こうすることで効果的に、Managed File Transfer Agent がソース・キュー内の各メッセージを異なるグループとして扱います。

関連概念

276 ページの『メッセージからファイルへのデータ転送』

Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

239 ページの『変数置換を使用した MFT リソース・モニター・タスクのカスタマイズ』
アクティブなリソース・モニターのトリガー条件が満たされると、定義されたタスクが呼び出されます。 毎回同じ宛先エージェントまたは同じ宛先ファイル名を使用して転送またはコマンド・タスクを呼び出すことができますが、実行時にタスク定義を変更することもできます。 これは、タスク定義 XML に変数名を挿入することで行います。 トリガー条件が満たされているとモニターが判断し、タスク定義に変数名が含まれている場合は、変数名を変数値と置換してから、タスクを呼び出します。

キュー・リソース・モニターが開始した転送によって作成された宛先ファイルに 間違ったデータが含まれる場合の対処法

関連タスク

232 ページの『コマンドおよびスクリプトを開始する MFT モニター・タスクの構成』
リソース・モニターの関連タスクは、ファイル転送の実行に限定されません。 また、実行可能プログラム、Ant スクリプト、または JCL ジョブなどのモニター・エージェントから他のコマンドを呼び出すようにモニターを構成することもできます。 コマンドを呼び出すには、モニター・タスク定義 XML を編集して、引数およびプロパティーなど、対応するコマンド呼び出しパラメーターを指定した 1 つ以上のコマンド・エレメントを含めます。

238 ページの『例: MFT リソースの構成』
fteCreateMonitor コマンドで **-mq** パラメーターを使用することにより、リソース・モニターによってモニターされるリソースとして IBM MQ キューを指定できます。

関連資料

fteCreateMonitor: MFT リソース・モニターの作成

MFT がソース・キューのメッセージから読み取る MQ メッセージ・プロパティー

例: IBM MQ メッセージ・プロパティーを使用したメッセージからファイルへの転送の失敗

usr.UserReturnCode IBM MQ メッセージ・プロパティーをゼロ以外の値に設定することによって、メッセージからファイルへの転送を失敗させることができます。 さらに、usr.UserSupplement IBM MQ メッセージ・プロパティーを設定することによって、失敗の理由に関する補足情報を指定することもできます。

このタスクについて

この例では、キュー INPUT_QUEUE とファイル /home/user/output.file の間で転送が進行中です。

ユーザーはメッセージを作成し、これをキュー INPUT_QUEUE の上に配置しています。 ソース・エージェントはキュー INPUT_QUEUE からメッセージをコンシュームし、転送データを宛先エージェントに送信しています。 宛先エージェントがこのデータをファイル /home/user/output.file に書き込んでいます。

メッセージをキュー INPUT_QUEUE に書き込んでいるユーザーは、進行中の転送を停止し、既に宛先ファイルに書き込まれたデータをすべて削除しようとします。

手順

1. ユーザーは、次の IBM MQ メッセージ・プロパティーを設定したメッセージをキュー INPUT_QUEUE に書き込みます。

```
usr.UserReturnCode=1
usr.UserSupplement="Cancelling transfer - sent wrong data."
```

2. ソース・エージェントは、IBM MQ メッセージ・プロパティーを読み取り、キューからのメッセージの処理を停止します。 宛先エージェントは、宛先ディレクトリーに書き込まれたファイル・データをすべて削除します。
3. ソース・エージェントは、転送の失敗を報告する転送ログ・メッセージを調整キュー・マネージャーに送信します。

   このメッセージには、次の情報が含まれています。

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
               ID="414d5120514d312020202020202020207e970d4920008702" agentRole="sourceAgent"
             xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
  <action time="2008-11-02T21:28:09.593Z">progress</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
           version="6.1 build 7601 Service Pack 1" />
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
           version="6.1 build 7601 Service Pack 1" />
  </destinationAgent>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1   </mqmdUserID>
  </originator>
  <transferSet index="0" size="1"
               startTime="2008-11-02T21:28:09.281Z"
               total="1">
    <item mode="binary">
      <source>
        <queue>INPUT_QUEUE@QM1</queue>
      </source>
      <destination exist="error">
        <file>/home/user/output.file</file>
      </destination>
      <status resultCode="1">
        <supplement>Cancelling transfer - sent wrong data.</supplement>
      </status>
    </item>
  </transferSet>
</transaction>
```

関連概念

276 ページの『メッセージからファイルへのデータ転送』
Managed File Transfer のメッセージからファイルへの転送機能を使用すれば、IBM MQ の 1 つのキューにある 1 つ以上のメッセージのデータを、1 つのファイル、1 つのデータ・セット (z/OS の場合)、または 1 つのユーザー・ファイル・スペースに転送できます。 IBM MQ メッセージを作成または処理するアプリケーションがあれば、Managed File Transfer のメッセージからファイルへの転送機能を使用して、Managed File Transfer ネットワーク内の任意のシステムにあるファイルにメッセージを転送することができます。

関連タスク

277 ページの『メッセージからファイルへの転送を実行するためのエージェントの構成』
デフォルトでは、エージェントがメッセージからファイルへの転送またはファイルからメッセージへの転送を実行することはできません。 この機能を有効にするには、エージェント・プロパティー enableQueueInputOutput を true に設定する必要があります。

278 ページの『例: 1 つのキューから 1 つのファイルへの転送』
fteCreateTransfer コマンドで **-sq** パラメーターを使用することにより、ファイル転送のソースとして IBM MQ キューを指定できます。

279 ページの『例: キューにあるメッセージのグループを 1 つのファイルに転送する操作』
fteCreateTransfer コマンドで **-sq** パラメーターおよび **-sqgi** パラメーターを使用することにより、IBM MQ キュー上の単一の完全なグループをファイル転送のソースとして指定できます。

280 ページの『例: 各メッセージのデータの前にテキスト区切り文字を挿入する操作』

テキスト・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdt** および **-sqdp** パラメーターを使用して、個々のメッセージのデータの前にテキスト区切り文字を挿入する動作を指定できます。

282 ページの『例: 各メッセージのデータの後にバイナリー区切り文字を挿入する操作』
バイナリー・モードでソース・キューからファイルへの転送を実行する場合は、**fteCreateTransfer** コマンドの **-sq**、**-sqdb**、および **-sqdp** パラメーターを使用して、個々のメッセージのデータの後にバイナリー区切り文字を挿入する動作を指定できます。

244 ページの『キューのモニターおよび変数置換の使用』
fteCreateMonitor コマンドを使用して、キューをモニターし、モニターしたキューからファイルにメッセージを転送できます。 モニターされるキューから読み取られる最初のメッセージにある任意の IBM MQ メッセージ・プロパティーの値をタスク XML 定義に置換して、転送動作の定義に使用できます。

関連資料

MFT がソース・キューのメッセージから読み取る MQ メッセージ・プロパティー

プロトコル・ブリッジ

プロトコル・ブリッジを使用すれば、Managed File Transfer (MFT) ネットワークから、MFT ネットワークの外部 (ローカル・ドメインとリモート・ロケーションの両方) にあるファイル・サーバーに格納されているファイルにアクセスできます。 このファイル・サーバーでは、FTP、FTPS、または SFTP ネットワーク・プロトコルを使用できます。 それぞれのファイル・サーバーで少なくとも 1 つの専用エージェントが必要です。 この専用エージェントは、 プロトコル・ブリッジ・エージェントとして知られています。 ブリッジ・エージェントは、複数のファイル・サーバーと相互作用できます。

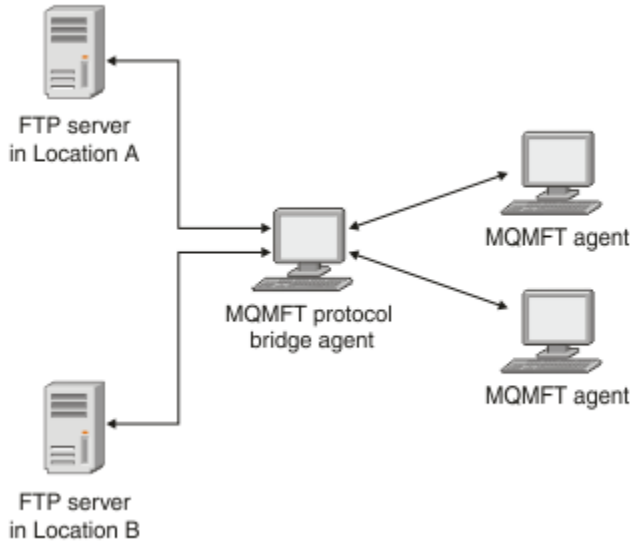
プロトコル・ブリッジは、Managed File Transfer のサービス・コンポーネントの一部として使用可能です。MFT を実行する単一のシステムに、さまざまなファイル・サーバーに接続する複数の専用エージェントを作成することができます。

プロトコル・ブリッジ・エージェントを使用して、複数のエンドポイントに同時にファイル転送を行うことができます。 MFT には `ProtocolBridgeProperties.xml` と呼ばれるファイルがあるので、このファイルを編集して、ファイルの転送先にするさまざまなプロトコル・ファイル・サーバーを定義できます。 **fteCreateBridgeAgent** コマンドは、`ProtocolBridgeProperties.xml` にデフォルトのプロトコル・ファイル・サーバーの詳細情報を追加します。 このファイルについての説明はプロトコル・ブリッジ・プロパティー・ファイルのフォーマットにあります。

プロトコル・ブリッジ・エージェントを使用して、以下のアクションを実行できます。

- MFT ネットワークからリモート・サーバーへのファイルのアップロード (FTP、FTPS、または SFTP を使用)
- リモート・サーバーから MFT ネットワークへのファイルのダウンロード (FTP、FTPS、または SFTP を使用)

注 : プロトコル・ブリッジ・エージェントは、絶対ファイル・パスによってファイルへのアクセスを可能にする FTP、FTPS、または SFTP サーバーのみをサポートできます。 転送要求に相対ファイル・パスが指定されると、プロトコル・ブリッジ・エージェントは、プロトコル・サーバーへのログインに使用されたホーム・ディレクトリーが基準になっていると見なして、相対パスを絶対ファイル・パスに変換しようとします。 現行ディレクトリーに基づいたファイルへのアクセスのみが可能なプロトコル・サーバーは、プロトコル・ブリッジ・エージェントではサポートされません。



この図は、異なるロケーションにある 2 つの FTP サーバーを示しています。 FTP サーバーは、Managed File Transfer エージェントとファイルを交換するために使用されています。 プロトコル・ブリッジ・エージェントは、FTP サーバーと、MFT ネットワークの残りの部分との間にあり、両方の FTP サーバーと通信するように構成されています。

プロトコル・ブリッジ・エージェントに加え、MFT ネットワークに別のエージェントがあることを確認します。 プロトコル・ブリッジ・エージェントは、FTP、FTPS、または SFTP サーバーに対してのみのブリッジであり、転送されたファイルをローカル・ディスクに書き込むことはありません。 ファイルを FTP、FTPS、または SFTP サーバーとの間で転送する場合は、プロトコル・ブリッジ・エージェントを (FTP、FTPS、または SFTP サーバーを代表する) ファイル転送の宛先またはソースとして使用し、別の標準エージェントを対応するソースまたは宛先として使用する必要があります。

プロトコル・ブリッジを使用してファイルを転送する場合、ブリッジは、転送するファイルが格納されているソースまたは宛先ディレクトリーを読み取るための権限を持っている必要があります。 例えば、実行許可 (d--x--x--x --x) のみを持つディレクトリー `/home/fte/bridge` からファイルを転送する場合は、このディレクトリーからの転送は失敗し、次のエラー・メッセージが表示されます。

```
BFGBR0032E: Attempt to read filename from the protocol file server
  has failed with server error 550. Failed to open file.
```

プロトコル・ブリッジ・エージェントの構成

プロトコル・ブリッジ・エージェントは、標準的な MFT エージェントに類似しています。 **fteCreateBridgeAgent** コマンドを使用してプロトコル・ブリッジ・エージェントを作成します。 プロトコルブリッジエージェントの設定は、 プロトコルブリッジプロパティファイルフォーマットに記載されている `ProtocolBridgeProperties.xml` ファイルを用いて行います。 以前のバージョンを使用している場合は、 拡張エージェント・プロパティー: プロトコル・ブリッジ および 拡張エージェント・プロパティー: プロトコル・ブリッジ・エージェントのロギングで説明されている特定のプロトコル・ブリッジ・プロパティーを使用してエージェントを構成してください。 すべてのバージョンで、296 ページの『ファイル・サーバーの資格情報のマップ』での説明に従って資格情報マッピングを構成することもできます。 特定のプロトコル・ファイル・サーバー用にプロトコル・ブリッジ・エージェントを構成した後で、このエージェントをその他の目的で使用できません。

プロトコル・ブリッジのリカバリー

ファイル・サーバーが使用できないために、プロトコル・ブリッジ・エージェントがファイル・サーバーに接続できない場合、すべてのファイル転送要求は、 ファイル・サーバーが使用可能になるまで、キューに入れられます。 エージェントが誤った資格情報を使用しているために、プロトコル・ブリッジ・エージェントがファイル・サーバーに接続できない場合、転送は失敗し、転送ログ・メッセージにこのエラーが

反映されます。 何らかの理由によって プロトコル・ブリッジ・エージェントが終了した場合、要求済みのファイル転送はすべて保持され、 プロトコル・ブリッジが再始動すると、続きが処理されます。

ファイル転送中は、 ファイルは通常、一時ファイルとして転送先に書き込まれ、 転送が完了した時点でリネームされます。 ただし、転送先が、書き込みを制限して構成されている (ユーザーはプロトコル・ファイル・サーバーにファイルをアップロードできるが、それらのアップロードされたファイルは決して変更できず、ユーザーが書き込めるのは事実上一度だけである) プロトコル・ファイル・サーバーの場合、転送されるファイルは転送先に直接書き込まれます。 そのため、転送中に問題が発生した場合、一部しか書き込まれていないファイルが転送先プロトコル・ファイル・サーバーに残りますが、Managed File Transfer はこのようなファイルを削除も編集もできません。 このシチュエーションでは、 転送は失敗します。

関連タスク

301 ページの『例: UNIX SFTP サーバーで秘密鍵の資格情報を使用するようにプロトコル・ブリッジ・エージェントを構成する方法』
この例では、 ProtocolBridgeCredentials.xml ファイルを生成して構成する方法を示します。 この例は標準的な例であり、ご使用のプラットフォームに応じて詳細が異なることがありますが、原則は同じです。

289 ページの『ProtocolBridgeProperties.xml ファイルを使用したプロトコル・ファイル・サーバーのプロパティーの定義』
エージェント設定ディレクトリで Managed File Transfer 提供されるファイルを使用して、 ProtocolBridgeProperties.xml ファイル転送を行う 1 つ以上のプロトコルファイルサーバのプロパティを定義します。

関連資料

fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)

296 ページの『ファイル・サーバーの資格情報のマップ』
プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用するか、独自のユーザー出口を作成して、Managed File Transfer にあるユーザー資格情報をファイル・サーバーのユーザー資格情報にマップします。 Managed File Transfer には、ユーザー資格情報マッピングを実行するサンプルのユーザー出口が用意されています。

ProtocolBridgeCredentialExit.java インターフェース

プロトコル・ブリッジ資格情報ユーザー出口のサンプル

プロトコル・ブリッジによる FTPS サーバーのサポート

ProtocolBridgeProperties.xml ファイルを使用したプロトコル・ファイル・サーバーのプロパティーの定義

エージェント設定ディレクトリで Managed File Transfer 提供されるファイルを使用して、 ProtocolBridgeProperties.xml ファイル転送を行う 1 つ以上のプロトコルファイルサーバのプロパティを定義します。

このタスクについて

fteCreateBridgeAgent コマンドを使用すると、エージェント構成ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* に ProtocolBridgeProperties.xml ファイルが作成されます。 このコマンドの実行時にデフォルトが指定されている場合は、コマンドはこのファイル内にデフォルトのプロトコル・ファイル・サーバーの項目も作成します。

メッセージ BFGCL0392I は、 ProtocolBridgeProperties.xml ファイルの場所を示します。

```
<?xml version="1.0" encoding="IBM-1047"?>
<!--
This ProtocolBridgeProperties.xml file determines the protocol servers that will be accessed by
the
MQMFT protocol bridge agent.

Each protocol server is defined using either a <tns:ftpServer>, <tns:ftpsServer>, or
<tns:sftpServer>
element - depending on the protocol used to communicate with the server.  When the protocol
```

```
bridge agent participates in a managed file transfer it will determine which server to used
based on
the prefix (if any) present on the file path.  For example a file path of 'server1:/home/user/
file.txt' would
be interpreted as a request to transfer /home/user/file.txt using 'server1'.  The server name
is compared
to the 'name' attribute of each <tns:ftpServer>, <tns:ftpsServer> or <tns:sftpServer> element
in this
XML document and the first match is used to determine which protocol server the protocol bridge
agent will connect to.  If no match is found then the managed file transfer operation will fail.

If a file path is not prefixed with a server name, for example '/home/user/file.txt' then this
XML
document can specify a default server to use for the managed file transfer.  To specify a
default server use the <tns:defaultServer> element as the first element inside the
<tns:serverProperties>
element.  The default server will be used whenever the protocol bridge agent participates in
a managed file transfer for file names which do not specify a prefix.

An optional <tns:limits> element can be specified within each server definition. This element
contains
attributes that govern the amount of resources used by each defined server.

An optional <tns:credentialsFile> element can be specified within each serverProperties
definition. This
element contains a path to a file containing credentials to be used when connecting to defined
servers.

An example ProtocolBridgeProperties.xml file is as follows:

<?xml version="1.0" encoding="UTF-8"?>
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
ProtocolBridgeProperties.xsd">

    <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml" />

    <tns:defaultServer name="myFTPserver" />

    <tns:ftpServer name="myFTPserver" host="windows.hursley.ibm.com" port="1234"
platform="windows"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF-8"
        listFormat="unix" limitedWrite="false">

        <tns:limits maxListFileNames="100" maxListDirectoryLevels="999999999"
            maxReconnectRetry="2"  reconnectWaitPeriod="10"
            maxSessions="60" socketTimeout="30" />

    </tns:ftpServer>

    <tns:ftpsServer name="myFTPSserver" host="unix.hursley.ibm.com" platform="unix"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF8"
        listFormat="unix" limitedWrite="false" ftpsType="explicit"
        trustStore="C:\FTE\keystores\myFTPSserver\FTPSKeyStore.jks"
trustStorePassword="password">

        <tns:limits maxReconnectRetry="10" connectionTimeout="10"/>

    </tns:ftpsServer>

    <tns:sftpServer name="mySFTPserver" host="windows.hursley.ibm.com" platform="windows"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF-8"
        limitedWrite="false">

        <tns:limits connectionTimeout="60"/>

    </tns:sftpServer>

</tns:serverProperties>

This example shows the outermost <tns:serverProperties> element which must exist for the
document to
be valid, an optional <tns:defaultServer> element, as well as definitions for an FTP, FTPS and
SFTP server.

The attributes of the <tns:ftpServer>, <tns:ftpsServer> and <tns:sftpServer> elements determine
the
characteristics of the connection established to the server.  These attributes correspond to
the command
line parameters for the 'fteCreateBridgeAgent' command.
```

```
The following attributes are valid for all of the <tns:ftpServer>, <tns:ftpsServer> and
<tns:sftpServer>
elements: name, host, port, platform, fileEncoding, limitedWrite and controlEncoding.

The following attributes are valid for the <tns:ftpServer> and <tns:ftpsServer> elements:
timezone, locale,
listFormat, listFileRecentDateFormat, listFileOldDateFormat, and monthShortNames.

The following attributes are valid for the <tns:ftpServer> element only: passiveMode

The following attributes are valid for the <tns:ftpsServer> element only: ftpsType, trustStore,
trustStorePassword,
trustStoreType, keyStore, keyStorePassword, keyStoreType, ccc, protFirst, auth, and
connectTimeout.

The following attributes are valid for the <tns:limits> element within all of the
<tns:ftpServer>, <tns:ftpsServer>
and <tns:sftpServer> elements: maxListFileNames, maxListDirectoryLevels, maxReconnectRetry,
reconnectWaitPeriod,
maxSessions and socketTimeout

-->
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
ProtocolBridgeProperties.xsd">

    <!-- By default the location of the credentials file is in the home directory of the user
that started the -->
    <!-- protocol bridge agent. If you wish to specify a different location use the
credentialsFile element to -->
    <!-- do this. For
example:                                                                                    -->
    <!-- <tns:credentialsFile path="/test/
ProtocolBridgeCredentials.xml"/>                                    -->

    <tns:defaultServer name="WINMVSCA.HURSLEY.IBM.COM" />
    <tns:ftpServer name="WINMVSCA.HURSLEY.IBM.COM" host="WINMVSCA.HURSLEY.IBM.COM"
platform="unix"
                timeZone="Europe/London" locale="en-GB" fileEncoding="US-ASCII"
                listFormat="unix" limitedWrite="false"  />

    <!-- Define servers here -->
</tns:serverProperties>
```

このコマンドは、以下のメッセージ:BFGCL0532I を生成することがあります。

```
このエージェントが機能するためには、追加の資格情報ファイルを手動で作成する必要があります。
デフォルトでは、このファイルは ProtocolBridgeCredentials.xml という名前で、ホームにあります。
エージェントを開始するユーザーのディレクトリー。 例えば、このユーザーがエージェントを開始したとします。
場所は次のとおりです: $HOME/ProtocolBridgeCredentials.xml
```

資格情報ファイルを使用する場合、次の点に注意してください。

1. 作成方法の詳細については、以下のテキストを参照してください。
2. 資格情報ファイルは、アクセス権が制限されたディレクトリー内になければなりません。 例えば、他のユーザーの読み取りアクセスがないディレクトリーである必要があります。
3. エージェントを開始したユーザー ID の $HOME 環境変数で資格情報ファイルのディレクトリーの場所を指定するか、ProtocolBridgeProperties.xml ファイルを編集して以下の場所でロケーションを指定します。

   ```
   <tns:credentialsFile path="/test/ProtocolBridgeCredentials.xml"/>
   ```

デフォルト以外のプロトコル・サーバーをさらに追加する場合は、このファイルを編集して、プロトコル・サーバーのプロパティーを定義してください。 この例では、追加の FTP サーバーを加えます。

注 : プロトコル・ブリッジ・エージェントはファイル・ロックをサポートしていません。 これは、Managed File Transfer がファイル・サーバーのファイル・ロック・メカニズムをサポートしていないためです。

手順

1. 以下の行を <tns:serverProperties>の子エレメントとしてファイルに挿入して、プロトコル・ファイル・サーバーを定義します。

```
<tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234"
   platform="windows"
                           timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
                          listFormat="unix" limitedWrite="false" >
<tns:limits maxListFileNames="10" maxListDirectoryLevels="500"/>
```

2. 次に、属性の値を変更します。
   - name はプロトコル・ファイル・サーバーの名前です。
   - host はプロトコル・ファイル・サーバーのホスト名または IP アドレスです。
   - port はプロトコル・ファイル・サーバーのポート番号です。
   - platform はプロトコル・ファイル・サーバーが実行されるプラットフォームです。
   - timeZone はプロトコル・ファイル・サーバーを実行する時間帯です。
   - locale はプロトコル・ファイル・サーバーで使用される言語です。
   - fileEncoding はプロトコル・ファイル・サーバーの文字エンコードです。
   - listFormat はプロトコル・ファイル・サーバーから戻されるファイルのリスト形式です。
   - limitedWrite はファイル・サーバーに書き込みを行う際にデフォルト・モードに従うかどうかを判別します。デフォルト・モードでは、一時ファイルを作成し、転送が完了した後にそのファイルをリネームします。 書き込み専用として構成されたファイル・サーバーの場合、 ファイルは、最終的な名前をそのまま使用して作成されます。 このプロパティーの値は、true または false のいずれかになります。 limitedWrite 属性と doNotUseTempOutputFile エージェント・プロパティーは、プロトコル・ブリッジ・エージェントの場合に一緒に使用します。 一時ファイルを使用する場合は、doNotUseTempOutputFile の値を設定せず、limitedWrite の値を false に設定する必要があります。 これ以外の組み合わせで設定を行うと、一時ファイルは使用されません。
   - maxListFileNames はプロトコル・ファイル・サーバー上のディレクトリーでファイル名をスキャンする際に収集される名前の最大数です。
   - maxListDirectoryLevels はプロトコル・ファイル・サーバー上のディレクトリーでファイル名をスキャンする際に繰り返されるディレクトリー・レベルの最大数です。

   これらの属性のデフォルト値や、これらの属性が必須かオプションかなど、これらの属性に関する詳細情報については、プロトコル・ブリッジ・プロパティー・ファイルのフォーマットを参照してください。

関連資料

プロトコル・ブリッジ・プロパティー・ファイルのフォーマット

MFT が使用する正規表現

プロトコル・ファイル・サーバー・プロパティーの検索: ProtocolBridgePropertiesExit2

プロトコル・ファイル・サーバーが多数ある場合は、com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2 インターフェースを実装して、転送で参照されているプロトコル・ファイル・サーバー・プロパティーを検索できます。 このインターフェースは、 ProtocolBridgeProperties.xml ファイルを保守するように設定することができます。

このタスクについて

Managed File Transfer には、プロトコル・ファイル・サーバー・プロパティーを検索するサンプルのユーザー出口が用意されています。 詳しくは、293 ページの『サンプル・ユーザー出口を使用したプロトコル・ファイル・サーバー・プロパティーの検索』を参照してください。

プロトコル・ブリッジ・プロパティーを検索するユーザー出口は、インターフェース com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2 を実装する必要があります。 詳しくは、ProtocolBridgePropertiesExit.java インターフェースを参照してください。

他のユーザー出口と同じように、複数のプロトコル・サーバー・プロパティー出口をまとめてチェーニングできます。 出口は、エージェント・プロパティー・ファイルで protocolBridgePropertiesExitClasses プロパティーを使用して指定された順序で呼び出されます。 initialize メソッドはすべて個別に値を返します。1 つ以上のメソッドが値 false を返す場合は、エージェントは開始しません。 エージェントのイベント・ログにエラーが報告されます。

すべての出口の getProtocolServerProperties メソッドについては、1 つの全体的な結果のみが返されます。メソッドがプロパティー・オブジェクトを結果コードとして返す場合、この値は返された結果となり、後続の出口の getProtocolServerProperties メソッドは呼び出されません。 メソッドがヌル値を結果コードとして返す場合は、次の出口の getProtocolServerProperties メソッドが呼び出されます。 後続の出口がない場合は、ヌルの結果が返されます。 全体的な結果コードがヌルである場合は、プロトコル・ブリッジ・エージェントによる検索が失敗したとみなされます。

ProtocolBridgePropertiesExit2.java インターフェースの使用が推奨されますが、ProtocolBridgePropertiesExit.java インターフェースについては、『294 ページの『プロトコル・ファイル・サーバー・プロパティーの検索: ProtocolBridgePropertiesExit』』を参照してください。

出口を実行するには、以下のステップを実行します。

手順

1. プロトコル・サーバー・プロパティー・ユーザー出口をコンパイルします。
2. コンパイルした出口とそのパッケージ構造が含まれる Java アーカイブ (JAR) ファイルを作成します。
3. 出口クラスが含まれている JAR ファイルを、プロトコル・ブリッジ・エージェントの exits ディレクトリーに配置します。 このディレクトリーは、 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* ディレクトリーにあります。
4. プロパティー protocolBridgePropertiesExitClasses を含めるように、プロトコル・ブリッジ・エージェントのプロパティー・ファイルを編集します。 このプロパティーの値には、プロトコル・ブリッジ・サーバー・プロパティー・ユーザー出口を実装するクラスのコンマ区切りのリストを指定します。 出口クラスは、このリストで指定された順序で呼び出されます。 詳しくは、 MFT agent.properties ファイルを参照してください。
5. オプションで、protocolBridgePropertiesConfiguration プロパティーを指定できます。 このプロパティーに指定した値は、protocolBridgePropertiesExitClasses によって指定された出口クラスの initialize() メソッドにストリングとして渡されます。 詳しくは、MFT agent.properties ファイルを参照してください。

サンプル・ユーザー出口を使用したプロトコル・ファイル・サーバー・プロパティーの検索

Managed File Transfer には、プロトコル・ファイル・サーバー・プロパティーを検索するサンプルのユーザー出口が用意されています。

このタスクについて

プロトコル・ブリッジ・プロパティーを検索するサンプル・ユーザー出口は、*MQ_INSTALLATION_PATH*/mqft/samples/protocolBridge ディレクトリーおよびトピック「プロトコル・ブリッジ・プロパティー・ユーザー出口の」に用意されています。

SamplePropertiesExit2.java 出口は、プロトコル・サーバーのプロパティーが含まれているプロパティー・ファイルを読み取ります。 プロパティー・ファイル内の各項目の形式は、次のとおりです。

```
serverName=type://host:port
```

プロパティー・ファイルの場所は、プロトコル・ブリッジ・エージェント・プロパティーである protocolBridgePropertiesConfiguration から取得されます。

サンプル・ユーザー出口を実行するには、以下のステップを実行します。

手順

1. SamplePropertiesExit2.java ファイルをコンパイルします。
2. コンパイルした出口とそのパッケージ構造が含まれる JAR ファイルを作成します。
3. JAR ファイルを *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent*/exits ディレクトリーに配置します。
4. *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/agent.properties ファイルを編集して、以下の行を含めます。

```
protocolBridgePropertiesExitClasses=SamplePropertiesExit2
```

5. ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent* に、プロトコル・ブリッジ・プロパティー・ファイル (protocol_bridge_properties.properties など) を作成します。 このファイルを編集して、次の形式の項目を含めます。

```
serverName=type://host:port
```

6. *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent*/agent.properties ファイルを編集して、以下の行を含めます。

```
protocolBridgePropertiesConfiguration=MQ_DATA_PATH/mqft/config/coordination_queue_manager/
agents/bridge_agent/protocol_bridge_properties.properties
```

   protocol_bridge_properties.properties ファイルへの絶対パスを使用する必要があります。
7. **fteStartAgent** コマンドを使用してプロトコル・ブリッジ・エージェントを開始します。

関連概念

287 ページの『プロトコル・ブリッジ』
プロトコル・ブリッジを使用すれば、Managed File Transfer (MFT) ネットワークから、MFT ネットワークの外部 (ローカル・ドメインとリモート・ロケーションの両方) にあるファイル・サーバーに格納されているファイルにアクセスできます。 このファイル・サーバーでは、FTP、FTPS、または SFTP ネットワーク・プロトコルを使用できます。 それぞれのファイル・サーバーで少なくとも 1 つの専用エージェントが必要です。 この専用エージェントは、 プロトコル・ブリッジ・エージェントとして知られています。 ブリッジ・エージェントは、複数のファイル・サーバーと相互作用できます。

関連資料

ProtocolBridgePropertiesExit.java インターフェース
プロトコル・ブリッジ・プロパティー・ユーザー出口のサンプル
MFT agent.properties ファイル
fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)

プロトコル・ファイル・サーバー・プロパティーの検索: *ProtocolBridgePropertiesExit*

プロトコル・ファイル・サーバーが多数ある場合は、com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit インターフェースを実装して、転送で参照されているプロトコル・ファイル・サーバー・プロパティーを検索できます。

このタスクについて

ProtocolBridgeProperties.xml ファイルを保持するよりはむしろ、com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit インターフェースを実装することができます。 ProtocolBridgePropertiesExit2.java インターフェースを使用します。 ProtocolBridgePropertiesExit2.java 内の **getCredentialLocation** メソッドは、ProtocolBridgeCredentials.xml ファイルのデフォルト・ロケーション (ホーム・ディレクトリー) を使用します。

プロトコル・ブリッジ・プロパティーを検索するユーザー出口は、インターフェース com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit を実装する必要があります。

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     * @return {@code true} if the initialization is successful and {@code
     *         false} if unsuccessful. If {@code false} is returned from an exit
     *         the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Obtains a set of properties for the specified protocol server name.
     * <p>
     * The returned {@link Properties} must contain entries with key names
     * corresponding to the constants defined in
     * {@link ProtocolServerPropertyConstants} and in particular must include an
     * entry for all appropriate constants described as required.
     *
     * @param protocolServerName
     *            The name of the protocol server whose properties are to be
     *            returned. If a null or a blank value is specified, properties
     *            for the default protocol server are to be returned.
     * @return The {@link Properties} for the specified protocol server, or null
     *         if the server cannot be found.
     */
    public Properties getProtocolServerProperties(
            final String protocolServerName);

    /**
     * Invoked once when a protocol bridge agent is shut down. It is intended to
     * release any resources that were allocated by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     */
    public void shutdown(final Map<String, String> bridgeProperties);

}
```

他のユーザー出口と同じように、複数のプロトコル・サーバー・プロパティー出口をまとめてチェーニングできます。 出口は、エージェント・プロパティー・ファイルで protocolBridgePropertiesExitClasses プロパティーを使用して指定された順序で呼び出されます。 initialize メソッドはすべて個別に値を返します。1 つ以上のメソッドが値 false を返す場合は、エージェントは開始しません。 エージェントのイベント・ログにエラーが報告されます。

すべての出口の getProtocolServerProperties メソッドについては、1 つの全体的な結果のみが返されます。メソッドがプロパティー・オブジェクトを結果コードとして返す場合、この値は返された結果となり、後続の出口の getProtocolServerProperties メソッドは呼び出されません。 メソッドがヌル値を結果コード

として返す場合は、次の出口の getProtocolServerProperties メソッドが呼び出されます。 後続の出口がない場合は、ヌルの結果が返されます。 全体的な結果コードがヌルである場合は、プロトコル・ブリッジ・エージェントによる検索が失敗したとみなされます。

手順

出口を実行するには、以下のステップを実行します。

1. プロトコル・サーバー・プロパティー・ユーザー出口をコンパイルします。
2. コンパイルした出口とそのパッケージ構造が含まれる Java アーカイブ (JAR) ファイルを作成します。
3. 出口クラスが含まれている JAR ファイルを、プロトコル・ブリッジ・エージェントの `exits` ディレクトリーに配置します。

   このディレクトリーは、 *MQ_DATA_PATH*`/mqft/config/`*coordination_queue_manager*`/agents/`*bridge_agent_name* ディレクトリーにあります。
4. プロパティー `protocolBridgePropertiesExitClasses` を含めるように、プロトコル・ブリッジ・エージェントのプロパティー・ファイルを編集します。

   このプロパティーの値には、プロトコル・ブリッジ・サーバー・プロパティー・ユーザー出口を実装するクラスのコンマ区切りのリストを指定します。 出口クラスは、このリストで指定された順序で呼び出されます。 詳しくは、MFT `agent.properties` ファイルを参照してください。
5. オプションで、`protocolBridgePropertiesConfiguration` プロパティーを指定できます。

   このプロパティーに指定した値は、`protocolBridgePropertiesExitClasses` によって指定された出口クラスの initialize() メソッドにストリングとして渡されます。 詳しくは、 MFT `agent.properties` ファイルを参照してください。

ファイル・サーバーの資格情報のマップ

プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用するか、独自のユーザー出口を作成して、Managed File Transfer にあるユーザー資格情報をファイル・サーバーのユーザー資格情報にマップします。 Managed File Transfer には、ユーザー資格情報マッピングを実行するサンプルのユーザー出口が用意されています。

関連概念

287 ページの『プロトコル・ブリッジ』
プロトコル・ブリッジを使用すれば、Managed File Transfer (MFT) ネットワークから、MFT ネットワークの外部 (ローカル・ドメインとリモート・ロケーションの両方) にあるファイル・サーバーに格納されているファイルにアクセスできます。 このファイル・サーバーでは、FTP、FTPS、または SFTP ネットワーク・プロトコルを使用できます。 それぞれのファイル・サーバーで少なくとも 1 つの専用エージェントが必要です。 この専用エージェントは、 プロトコル・ブリッジ・エージェントとして知られています。 ブリッジ・エージェントは、複数のファイル・サーバーと相互作用できます。

関連タスク

297 ページの『ProtocolBridgeCredentials.xml ファイルを使用したファイル・サーバーの資格情報のマッピング』
プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用して、Managed File Transfer のユーザー資格情報をファイル・サーバーのユーザー資格情報にマップします。 Managed File Transfer で提供される XML ファイルを編集して、ユーザーの資格情報を組み込むことができます。

298 ページの『出口クラスを使用したファイル・サーバーの資格情報のマップ』
プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用しない場合は、独自のユーザー出口を作成して、Managed File Transfer のユーザー資格情報をファイル・サーバーのユーザー資格情報にマップできます。 資格情報マッピング・ユーザー出口を構成すると、デフォルトの資格情報マッピング機能の代わりになります。

301 ページの『例: UNIX SFTP サーバーで秘密鍵の資格情報を使用するようにプロトコル・ブリッジ・エージェントを構成する方法』
この例では、 `ProtocolBridgeCredentials.xml` ファイルを生成して構成する方法を示します。 この例は標準的な例であり、ご使用のプラットフォームに応じて詳細が異なることがありますが、原則は同じです。

関連資料
ProtocolBridgeCredentialExit.java インターフェース
プロトコル・ブリッジ資格情報ユーザー出口のサンプル
MFT agent.properties ファイル

ProtocolBridgeCredentials.xml ファイルを使用したファイル・サーバーの資格情報のマッピング

プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用して、Managed File Transfer のユーザー資格情報をファイル・サーバーのユーザー資格情報にマップします。 Managed File Transfer で提供される XML ファイルを編集して、ユーザーの資格情報を組み込むことができます。

このタスクについて

ProtocolBridgeCredentials.xml ファイルは、ユーザーが手動で作成する必要があります。 このファイルのデフォルトの場所は、プロトコル・ブリッジ・エージェントを始動したユーザーのホーム・ディレクトリーです。しかし、エージェントがアクセス可能なファイル・システム上の任意の場所に、このファイルを保管することができます。 別の場所を指定するには、 <credentialsFile> エレメントを ProtocolBridgeProperties.xml ファイルに追加します。 例:

```
<tns:credentialsFile path="/example/path/to/ProtocolBridgeCredentials.xml"/>
```

プロトコル・ブリッジ・エージェントを使用する前に、このファイルを編集してホスト、ユーザー、および資格情報を含めることによって、資格情報マッピングをセットアップします。 詳細およびサンプルについては、プロトコル・ブリッジの資格情報ファイルのフォーマットを参照してください。

手順

1. • <tns:server name="*server name*"> の行を編集して、name 属性の値を ProtocolBridgeProperties.xml ファイル内のサーバー名に変更します。

   ワイルドカードや正規表現を含むサーバー名を使用したことを指定するには、パターン属性を使用します。 例:

   ```
   <tns:server name="serverA*" pattern="wildcard">
   ```

2. <tns:server>の子エレメントとして、ユーザー ID および資格情報をファイルに挿入します。

   ファイルには、以下の 1 つ以上のエレメントを挿入できます。

   • プロトコル・ファイル・サーバーが FTP、FTPS または SFTP サーバーである場合は、パスワードを使用して、転送を要求しているユーザーを認証できます。 次の行をファイルに挿入します。

```
<tns:user name="FTE User ID"
  serverUserId="Server User ID"
  serverPassword="Server Password">
</tns:user>
```

次に、属性の値を変更します。

– name は、 MFT 転送要求に関連付けられた MQMD ユーザー ID と一致する Java 正規表現です。

– serverUserId は、プロトコル・ファイル・サーバーにログイン・ユーザー ID として渡される値です。 serverUserId 属性が指定されていない場合は、MFT 転送要求に関連付けられた MQMD ユーザー ID が代わりに使用されます。

– serverPassword は、serverUserId に関連付けられたパスワードです。

name 属性には Java 正規表現を含めることができます。 資格情報マッパーは、MFT 転送要求の MQMD ユーザー ID を、この正規表現と突き合わせようとします。 プロトコル・ブリッジ・エージェントは、エレメントがファイルに存在している順序で、<tns:user> エレメントの name 属性内の正規表現と MQMD ユーザー ID を突き合わせようとします。 一致が検出されると、プロトコル・ブリッ

ジ・エージェントはその他の一致を検索しません。 一致が検出されると、対応する serverUserId 値と serverPassword 値が、ログイン・ユーザー ID とパスワードとしてプロトコル・ファイル・サーバーに渡されます。 MQMD ユーザー ID の突き合わせでは大/小文字が区別されます。

- プロトコル・ファイル・サーバーが SFTP サーバーである場合は、転送を要求しているユーザーの認証に公開鍵と秘密鍵を使用できます。 次の行をファイルに挿入して、属性の値を変更します。 <tns:user> エレメントには、1 つ以上の <tns:privateKey> エレメントを含めることができます。

```
<tns:user name="FTE User ID"
    serverUserId="Server User ID"
    hostKey="Host Key">
    <tns:privateKey associationName="association"
      keyPassword="Private key password">
    Private key file text
    </tns:privateKey>
</tns:user>
```

  - name は、 MFT 転送要求に関連付けられた MQMD ユーザー ID と一致する Java 正規表現です。
  - serverUserId は、プロトコル・ファイル・サーバーにログイン・ユーザー ID として渡される値です。 serverUserId 属性が指定されていない場合は、MFT 転送要求に関連付けられた MQMD ユーザー ID が代わりに使用されます。
  - hostKey は、ログオン時にサーバーから返されることが予期される鍵です。
  - key は、serverUserId の秘密鍵です。
  - keyPassword は、公開鍵を生成するための鍵のパスワードです。
  - associationName は、トレースとロギングの目的で識別するために使用される値です。

  name 属性には Java 正規表現を含めることができます。 資格情報マッパーは、MFT 転送要求の MQMD ユーザー ID を、この正規表現と突き合わせようとします。 プロトコル・ブリッジ・エージェントは、エレメントがファイルに存在している順序で、<tns:user> エレメントの name 属性内の正規表現と MQMD ユーザー ID を突き合わせようとします。 一致が検出されると、プロトコル・ブリッジ・エージェントはその他の一致を検索しません。 一致が検出されると、対応する serverUserId 値と key 値が、プロトコル・ファイル・サーバーで MFT ユーザーを認証するために使用されます。 MQMD ユーザー ID の突き合わせでは大/小文字が区別されます。

  プロトコル・ブリッジ・エージェントでの秘密鍵の使用に関する詳細は、301 ページの『例: UNIX SFTP サーバーで秘密鍵の資格情報を使用するようにプロトコル・ブリッジ・エージェントを構成する方法』を参照してください。

注 : z/OS

転送要求がコマンド・キューに書き込まれると、ソース・エージェントのコマンド・キューが z/OS システムまたは IBM i システム上にある場合、MQMD ユーザー ID は大文字に変換されることがあります。この結果、同じ発信元ユーザーの MQMD ユーザー ID であっても、転送要求で指定されたソース・エージェントに応じて、元の大/小文字の形式かまたは変換された大文字の形式で資格情報出口に着信することになります。 デフォルトの資格情報マッピング出口は、提供された MQMD ユーザー ID に照らして大/小文字を区別する突き合わせを実行します。このことは、マッピング・ファイルにおいて考慮する必要がある場合があります。

関連資料

プロトコル・ブリッジの資格情報ファイルのフォーマット

プロトコル・ブリッジ・プロパティー・ファイルのフォーマット

MFT が使用する正規表現

出口クラスを使用したファイル・サーバーの資格情報のマップ

プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用しない場合は、独自のユーザー出口を作成して、Managed File Transfer のユーザー資格情報をファイル・サーバーのユーザー資格情報にマップできます。 資格情報マッピング・ユーザー出口を構成すると、デフォルトの資格情報マッピング機能の代わりになります。

このタスクについて

Managed File Transfer には、ユーザー資格情報マッピングを実行するサンプルのユーザー出口が用意されています。 詳しくは、300 ページの『プロトコル・ブリッジ資格情報ユーザー出口のサンプルの使用』を参照してください。

マッピング・プロトコル・ブリッジ資格情報のユーザー出口は、以下のいずれかのインターフェースを実装する必要があります。

- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit`。 プロトコル・ブリッジ・エージェントが 1 つのデフォルト・プロトコル・ファイル・サーバーとの間でファイルを転送できるようにします。
- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2`。複数のエンドポイントとの間でファイルを転送できるようにします。

`com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2` インターフェースには、`com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit` と同じ機能が含まれ、加えて拡張機能が含まれています。 詳しくは、ProtocolBridgeCredentialExit.java インターフェースおよびProtocolBridgeCredentialExit2.java インターフェースを参照してください。

資格情報出口は、他のユーザー出口と同じような方法でまとめてチェーニングできます。 出口は、エージェント・プロパティー・ファイルで `protocolBridgeCredentialConfiguration` プロパティーを使用して指定された順序で呼び出されます。 initialize メソッドはすべて個別に値を返します。1 つ以上のメソッドが値 false を返す場合は、エージェントは開始しません。 エージェントのイベント・ログにエラーが報告されます。

すべての出口の mapMQUserId メソッドについては、1 つの全体的な結果のみが、以下のように返されます。

- メソッドが値 USER_SUCCESSFULLY_MAPPED または USER_DENIED_ACCESS を結果コードとして返す場合、この値は返された結果となり、後続の出口の mapMQUserId メソッドは呼び出されません。
- メソッドが値 NO_MAPPING_FOUND を結果コードとして返す場合は、次の出口の mqMQUserId メソッドが呼び出されます。
- 後続の出口がない場合は、結果 NO_MAPPING_FOUND が返されます。
- USER_DENIED_ACCESS または NO_MAPPING_FOUND の全体的な結果コードは、ブリッジ・エージェントによる転送障害であるとみなされます。

出口を実行するには、以下のステップを実行します。

手順

1. プロトコル・ブリッジ資格情報ユーザー出口をコンパイルします。
2. コンパイルした出口とそのパッケージ構造が含まれる Java アーカイブ (JAR) ファイルを作成します。
3. 出口クラスが入っている JAR ファイルを、ブリッジ・エージェントの `exits` ディレクトリーに配置します。 このディレクトリーは、*MQ_DATA_PATH*`/mqft/config/`*coordination_queue_manager*`/agents/`*bridge_agent_name* ディレクトリーにあります。
4. プロパティー `protocolBridgeCredentialExitClasses` を含めるように、プロトコル・ブリッジ・エージェントのプロパティー・ファイルを編集します。 このプロパティーの値には、プロトコル・ブリッジ資格情報の出口ルーチンを実装するクラスのコンマ区切りのリストを指定します。 出口クラスは、このリストで指定された順序で呼び出されます。 詳しくは、 MFT agent.properties ファイルを参照してください。
5. プロトコル・ブリッジ・エージェントのプロパティー・ファイルを編集して、以下を含めます。

```
exitClassPath=IBM MQ
installation_directory\mqft\config\configuration_queue_manager\agents\protocol_bridge_agent_n
ame\exits\SampleCredentialExit.jar
```

エージェントの agent.properties ファイルは、ご使用の *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*bridge_agent_name* ディレクトリーにあります。

agent.properties ファイルを変更する場合は、変更を反映するためにエージェントを再始動する必要があります。

6. オプションで、protocolBridgeCredentialConfiguration プロパティーを指定できます。 このプロパティーに指定した値は、protocolBridgeCredentialExitClasses によって指定された出口クラスの initialize() メソッドにストリング・オブジェクトとして渡されます。 詳しくは、 MFT agent.properties ファイルを参照してください。
7. **fteStartAgent** コマンドを使用してプロトコル・ブリッジ・エージェントを開始します。

プロトコル・ブリッジ資格情報ユーザー出口のサンプルの使用
Managed File Transfer には、ユーザー資格情報マッピングを実行するサンプルのユーザー出口が用意されています。

このタスクについて

サンプル・プロトコル・ブリッジ資格情報出口は、 *MQ_INSTALLATION_PATH*/mqft/samples/protocolBridge ディレクトリーおよびトピック プロトコル・ブリッジ資格情報ユーザー出口のサンプル に用意されています。 このサンプルは、com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit インターフェースに基づいています。

SampleCredentialExit.java 出口は、転送要求に関連付けられている MQMD ユーザー ID をサーバー・ユーザー ID およびサーバー・パスワードにマップするプロパティー・ファイルを読み取ります。 プロパティー・ファイルの場所は、プロトコル・ブリッジ・エージェント・プロパティーである protocolBridgeCredentialConfiguration から取得されます。

サンプル・ユーザー出口を実行するには、以下のステップを実行します。

手順

1. SampleCredentialExit.java ファイルをコンパイルします。
2. コンパイルした出口とそのパッケージ構造が含まれる JAR ファイルを作成します。
3. JAR ファイルを *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/exits ディレクトリーに配置します。
4. *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/agent.properties ファイルを編集して、以下の行を含めます。

```
protocolBridgeCredentialExitClasses=SampleCredentialExit
```

5. プロトコル・ブリッジ・エージェントのプロパティー・ファイルを編集して、以下を含めます。

```
exitClassPath=IBM MQ
installation_directory\mqft\config\configuration_queue_manager\agents\protocol_bridge_agent_n
ame\exits\SampleCredentialExit.jar
```

エージェントの agent.properties ファイルは、ご使用の *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*agent_name* ディレクトリーにあります。

agent.properties ファイルを変更する場合は、変更を反映するためにエージェントを再始動する必要があります。

6. ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent* に資格情報プロパティー・ファイル (credentials.properties) を作成し、そのファイルを編集して以下の形式でエントリーを組み込みます。

```
mqUserId=serverUserId,serverPassword
```

7. *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/agent.properties ファイルを編集して、以下の行を含めます。

```
protocolBridgeCredentialConfiguration=MQ_DATA_PATH/mqft/
config/coordination_queue_manager/agents/bridge_agent_name/credentials.properties
```

   credentials.properties ファイルへの絶対パスを使用する必要があります。

8. **fteStartAgent** コマンドを使用してプロトコル・ブリッジ・エージェントを開始します。

関連概念

287 ページの『プロトコル・ブリッジ』
プロトコル・ブリッジを使用すれば、Managed File Transfer (MFT) ネットワークから、MFT ネットワークの外部 (ローカル・ドメインとリモート・ロケーションの両方) にあるファイル・サーバーに格納されているファイルにアクセスできます。 このファイル・サーバーでは、FTP、FTPS、または SFTP ネットワーク・プロトコルを使用できます。 それぞれのファイル・サーバーで少なくとも 1 つの専用エージェントが必要です。 この専用エージェントは、 プロトコル・ブリッジ・エージェントとして知られています。 ブリッジ・エージェントは、複数のファイル・サーバーと相互作用できます。

関連資料

ProtocolBridgeCredentialExit.java インターフェース
ProtocolBridgeCredentialExit2.java インターフェース
プロトコル・ブリッジ資格情報ユーザー出口のサンプル
MFT agent.properties ファイル
fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)

例: UNIX SFTP サーバーで秘密鍵の資格情報を使用するようにプロトコル・ブリッジ・エージェントを構成する方法

この例では、 ProtocolBridgeCredentials.xml ファイルを生成して構成する方法を示します。 この例は標準的な例であり、ご使用のプラットフォームに応じて詳細が異なることがありますが、原則は同じです。

このタスクについて

手順

1. SFTP サーバーでの認証に使用する公開鍵と秘密鍵を生成します。

   例えば、Linux ホスト・システムでは、'openssh' パッケージの一部として提供されているツール **ssh-keygen** を使用して、公開鍵と秘密鍵のペアを作成できます。

   デフォルトでは、引数を指定せずに **ssh-keygen** コマンドを実行すると、2 つの鍵ファイルの場所とパスフレーズの入力を求めるプロンプトが出されます。デフォルトの名前は以下のとおりです。

```
id_rsa      <-- Private key
id_rsa.pub  <-- Public key
```

   重要 : 最新バージョンの OpenSSH (RHEL 8 で提供されているものなど) から **ssh-keygen** コマンドを使用している場合、使用される鍵フォーマットはプロトコル・ブリッジ・エージェントと互換性がなく、SFTP サーバーへの転送は失敗し、以下のメッセージが表示されます。

```
BFGBR0216E: Authentication to protocol server 'sftp.host.address' failed
because of invalid private key.
```

これらの新しいバージョンの OpenSSH と互換性のある秘密鍵を作成するには、**ssh-keygen** コマンドに以下の引数を使用して鍵フォーマットを指定します。

```
ssh-keygen -m PEM
```

id_rsa 秘密鍵の内容には、以下の最初と最後の行が含まれます。

```
-----BEGIN RSA PRIVATE KEY-----
... ... ...
-----END RSA PRIVATE KEY-----
```

これは、プロトコル・ブリッジ・エージェントと互換性があります。

2. id_rsa.pub ファイルの内容全体を SFTP サーバー上の SFTP ユーザーの ~/.ssh/authorized_keys ファイルにコピーします。

   SFTP サーバーが鍵認証を許可するように、このファイルおよび ~/.ssh ディレクトリーのファイル許可が適切に設定されていることを確認してください。 通常、これらの権限は以下のとおりです

```
~/.ssh                    Mode 700
~/.ssh/authorized_keys    Mode 600
```

3. Managed File Transfer では、MD5 アルゴリズムを使用して生成されたホストの ssh 指紋が必要です。以下のいずれかのコマンドを実行して、SFTP サーバーのホストの ssh 指紋を取得します。

   - Red Hat® Enterprise Linux バージョン 6.x 以前、および Linux Ubuntu 14.04 の場合、以下のコマンドを実行します。

     ```
     ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
     ```

   - Red Hat Enterprise Linux 7.x、Linux Ubuntu 16.04、および SuSE Linux 12.4 以降では、 ssh-keygen コマンドにより、SHA256 アルゴリズムを使用して ssh 指紋がデフォルトで生成されます。 MD5 アルゴリズムを使用して ssh 指紋を生成するには、以下のコマンドを実行します。

     ```
     ssh-keygen -l -E MD5 -f /etc/ssh/ssh_host_rsa_key.pub
     ```

     コマンドの出力は、以下の例のようになります。

     ```
     2048 MD5:64:39:f5:49:41:10:55:d2:0b:81:42:5c:87:62:9d:27 no comment (RSA)
     ```

     ProtocolBridgeCredentials.xml ファイル内でホスト・キーとして使用する出力の 16 進数部分のみを抽出します (ステップ 302 ページの『4』を参照)。 したがって、この例では、64:39:f5:49:41:10:55:d2:0b:81:42:5c:87:62:9d:27 を抽出します。

4. プロトコル・ブリッジ・エージェント・システム上で、 ProtocolBridgeCredentials.xml ファイルを編集します。 以下の例でイタリックで示されている値を独自の値に置換します。

```
<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
 ProtocolBridgeCredentials.xsd ">

<tns:agent name="Agent_name">

<tns:server name="SFTP_name">

<tns:user name="mq_User_ID" serverUserId="SFTP_user_ID"
  hostKey="ssh_host_finger">
<tns:privateKey associationName="name" keyPassword="pass_phrase">
 Complete contents of the id_rsa file including the entries
 -----BEGIN RSA PRIVATE KEY-----

 -----END RSA PRIVATE KEY-----
</tns:privateKey>
</tns:user>

</tns:server>
```

```
</tns:agent>
</tns:credentials>
```

ここで、

- *Agent_name* は、プロトコル・ブリッジ・エージェントの名前です。
- *SFTP_host_name* は、 `ProtocolBridgeProperties.xml` ファイルに示されているように、 SFTP サーバーの名前です。
- *mq_User_ID* は、転送要求に関連付けられた MQMD ユーザー ID です。
- *SFTP_user_ID* は、ステップ 2 で使用されている SFTP ユーザー ID です。 これは、ログイン・ユーザー ID として SFTP 機能に渡される値です。
- *ssh_host_finger* は、ステップ 3 で収集した指紋です。
- *name* は、トレースとロギングを目的として使用するために指定できる名前です。
- *pass_phrase* は、ステップ 1 で ssh-keygen に指定したパスフレーズです。
- *id_rsa* ファイルの内容を完了します。 は、ステップ 1 から生成された `id_rsa` ファイルの完全な内容です。 接続エラーを回避するには、以下の両方の項目が含まれていることを確認してください。

```
-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----
```

<tns:privatekey> エレメントを複製することによって、鍵を追加できます。

5. プロトコル・ブリッジ・エージェントがまだ開始していない場合は、開始します。 代わりに、プロトコル・ブリッジ・エージェントは、定期的に `ProtocolBridgeCredentials.xml` ファイルをポーリングし、変更内容を選択します。

関連資料

プロトコル・ブリッジの資格情報ファイルのフォーマット

fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)

MFT `agent.properties` ファイル

296 ページの『ファイル・サーバーの資格情報のマップ』

プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用するか、独自のユーザー出口を作成して、Managed File Transfer にあるユーザー資格情報をファイル・サーバーのユーザー資格情報にマップします。 Managed File Transfer には、ユーザー資格情報マッピングを実行するサンプルのユーザー出口が用意されています。

FTPS サーバー用のプロトコル・ブリッジの構成

FTPS サーバーの構成は、FTP サーバーの構成と同様の方法で行います。つまり、サーバー用のブリッジ・エージェントを作成し、サーバー・プロパティーを定義し、ユーザー資格情報をマップします。

このタスクについて

FTPS サーバーを構成するには、以下のステップを実行します。

手順

1. **fteCreateBridgeAgent** コマンドを使用して、FTPS サーバー用のプロトコル・ブリッジ・エージェントを作成します。 FTP に適用できるパラメーターを FTPS にも適用できますが、それに加えて以下の 3 つの FTPS に固有な必須パラメーターがあります。
   a) **-bt** パラメーター。 このパラメーターの値として FTPS を指定してください。
   b) トラストストア・ファイルの **-bts** パラメーター。 コマンドはサーバー認証のみが必要だと想定するので、トラストストア・ファイルの場所を指定しなければなりません。

デフォルトで **fteCreateBridgeAgent** コマンドによって FTPS プロトコルの明示書式が構成されますが、暗黙書式はプロトコル・ブリッジ・プロパティー・ファイルを変更することによって構成できます。 常にプロトコル・ブリッジはパッシブ・モードで FTPS サーバーに接続します。

fteCreateBridgeAgent コマンドについて詳しくは、fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成) を参照してください。

トラストストア・ファイルの作成方法についての説明が必要な場合は、 Oracle keytool 資料で keytool に関する情報を参照してください。

2. プロトコル・ブリッジ・プロパティー・ファイル ProtocolBridgeProperties.xml 内の <ftpsServer> エレメント内に FTPS サーバー・プロパティーを定義します。 詳しくは、 289 ページの『ProtocolBridgeProperties.xml ファイルを使用したプロトコル・ファイル・サーバーのプロパティーの定義』を参照してください。 プロトコル・ブリッジ・プロパティー・ファイルを編集して、クライアント認証を使用可能にすることもできます。 すべての構成オプションについて詳しくは、プロトコル・ブリッジ・プロパティー・ファイルのフォーマットを参照してください。
3. プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用するか、独自のユーザー出口を作成して、Managed File Transfer にあるユーザー資格情報を FTPS サーバーのユーザー資格情報にマップします。 詳しくは、296 ページの『ファイル・サーバーの資格情報のマップ』を参照してください。
4. デフォルトでは、トラストストア・ファイルは JKS 形式になるように構成されます。この形式を変更する場合は、プロトコル・ブリッジ・プロパティー・ファイルを編集します。

例

プロトコル・ブリッジ・プロパティー・ファイル内の FTPS サーバーの項目の例を以下に示します。

```
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
       xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
       ProtocolBridgeProperties.xsd">
    <tns:defaultServer name="ftpsserver.mycompany.com" />

    <tns:ftpsServer name="ftpsserver.mycompany.com" host="ftpsserver.mycompany.com" port="990"
platform="windows"
       timeZone="Europe/London" locale="en_US" fileEncoding="UTF8"
       listFormat="unix" limitedWrite="false"
       trustStore="c:\mydirec\truststore.jks" />

    <!-- Define servers here -->
</tns:serverProperties>
```

次のタスク

FTPS プロトコルのサポートされている部分と、サポートされていない部分に関する情報は、プロトコル・ブリッジによる FTPS サーバーのサポートを参照してください。

関連概念

287 ページの『プロトコル・ブリッジ』
プロトコル・ブリッジを使用すれば、Managed File Transfer (MFT) ネットワークから、MFT ネットワークの外部 (ローカル・ドメインとリモート・ロケーションの両方) にあるファイル・サーバーに格納されているファイルにアクセスできます。 このファイル・サーバーでは、FTP、FTPS、または SFTP ネットワーク・プロトコルを使用できます。 それぞれのファイル・サーバーで少なくとも 1 つの専用エージェントが必要です。 この専用エージェントは、 プロトコル・ブリッジ・エージェントとして知られています。 ブリッジ・エージェントは、複数のファイル・サーバーと相互作用できます。

関連タスク

297 ページの『ProtocolBridgeCredentials.xml ファイルを使用したファイル・サーバーの資格情報のマッピング』
プロトコル・ブリッジ・エージェントのデフォルトの資格情報マッピング機能を使用して、Managed File Transfer のユーザー資格情報をファイル・サーバーのユーザー資格情報にマップします。 Managed File Transfer で提供される XML ファイルを編集して、ユーザーの資格情報を組み込むことができます。

289 ページの『ProtocolBridgeProperties.xml ファイルを使用したプロトコル・ファイル・サーバーのプロパティーの定義』
エージェント設定ディレクトリで Managed File Transfer 提供されるファイルを使用して、ProtocolBridgeProperties.xml ファイル転送を行う 1 つ以上のプロトコルファイルサーバのプロパティを定義します。

関連資料

fteCreateBridgeAgent (MFT プロトコル・ブリッジ・エージェントの作成および構成)
プロトコル・ブリッジの資格情報ファイルのフォーマット
プロトコル・ブリッジ・プロパティー・ファイルのフォーマット
プロトコル・ブリッジによる FTPS サーバーのサポート

個々のファイル・サーバーへのファイル転送数制限のシナリオおよび例

変更されたプロトコル・ブリッジ・エージェントでの **maxActiveDestinationTransfers** 属性および **failTransferWhenCapacityReached** 属性の使用方法と、その例をいくつか説明します。

maxActiveDestinationTransfers の値に基づいたプロトコル・ブリッジ・エージェントの振る舞いを示すシナリオ

シナリオ 1

プロトコル・ブリッジ・エージェントの ProtocolBridgeProperties.xml ファイルには、以下の 2 つのファイル・サーバー定義が含まれています。

- グローバル **maxActiveDestinationTransfers** 属性が設定されていません。
- fileServerA と FileServerB の両方に **maxActiveDestinationTransfers** 属性が設定されていません。
- プロトコル・ブリッジ・エージェントの **maxDestinationTransfers** 属性をデフォルト値に設定しました。

プロトコル・ブリッジ・エージェントの **maxDestinationTransfers** 属性をデフォルト値の 25 に設定した場合は、以下の動作が発生します。

- 宛先エージェントは、 fileServerA への 2 つの管理対象転送の処理を開始します。
- 両方の転送が完了します。

  この時点で、クライアントは fileServerA が失敗したことを認識し、ProtocolBridgeProperties.xml ファイル内の fileServerA に以下の値を設定します。

  maxActiveDestinationTransfers = 0
  failTransferWhenCapacityReached =true
- fileServerA 用に別の転送が到着し、 fileServerB 用にいくつかの転送が到着します。

  前のステップで設定したプロパティーに基づき、fileServerA への管理対象転送は拒否され、failed とマークが付くのに対し、fileServerB への転送は標準的な既存のフローで処理されます。
- 少し時間が経って、クライアントは fileServerA が再び稼働していることを検出し、そのため、先ほど ProtocolBridgeProperties.xml に追加した値を削除またはコメント化します。 新しい管理対象転送が fileServerA に到着し、標準的な既存のフローで処理されます。

シナリオ 2

- ファイル・サーバーの **maxActiveDestinationTransfers** 属性を設定したが、**failTransferWhenCapacityReached** 属性は設定しなかった。
- プロトコル・ブリッジ・エージェントは、ファイル・サーバーへのこの件数の管理対象転送に対して、宛先エージェントとして機能する。

- **maxActiveDestinationTransfers** 属性の値は 1 減る。

プロトコル・ブリッジ・エージェントはその構成を動的に更新し、アクティブのまま **maxActiveDestinationTransfers** を新しい値に設定します。 進行中の管理対象転送は、この更新による影響を受けずに完了します。

シナリオ 3

プロトコル・ブリッジ・エージェントの ProtocolBridgeProperties.xml ファイルには、以下の 2 つのファイル・サーバー定義が含まれます。

- グローバル **maxActiveDestinationTransfers** 属性が設定されていません。
- **failTransferWhenCapacityReached** 属性が設定されていません。
- fileServerA で **maxActiveDestinationTransfers** を 1 に設定しました。
- fileServerB で **maxActiveDestinationTransfers** 属性が設定されていません。

プロトコル・ブリッジ・エージェントの **maxDestinationTransfers** 属性が 5 に設定されていた場合は、以下の動作が発生します。

- プロトコル・ブリッジ・エージェントから fileServerA へのアクティブ宛先転送の最大数は 1 です (宛先エージェントに 5 個の宛先転送スロットがある場合でも、fileServerA への管理対象転送に使用されるスロットは 1 個のみです)。

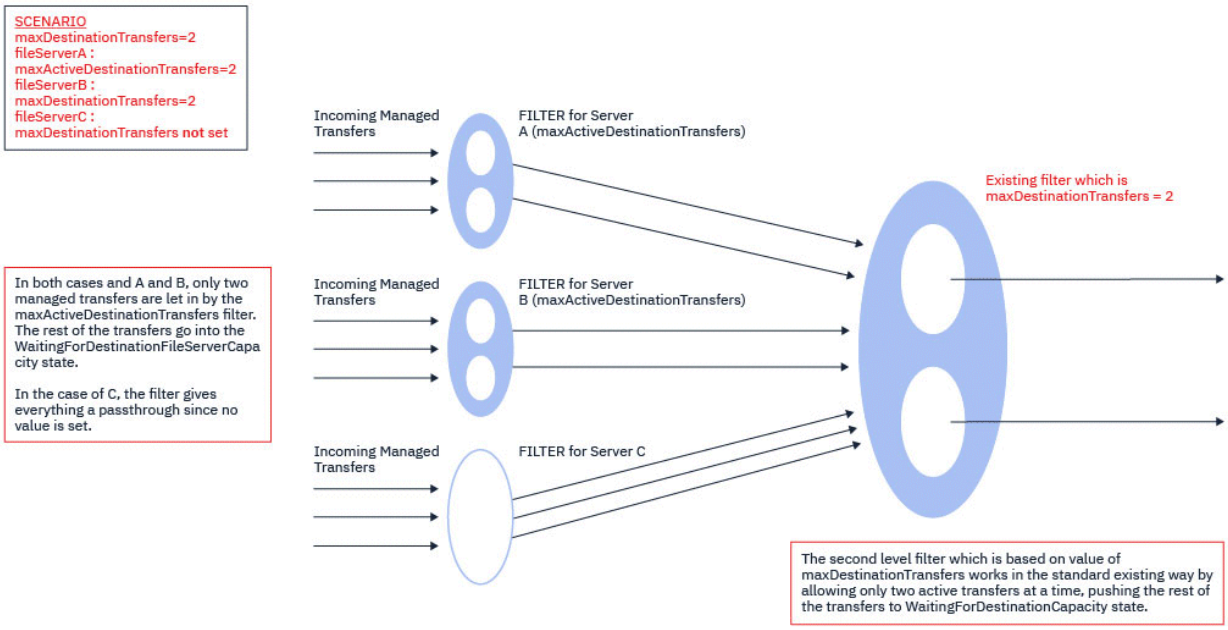
  これは、 fileServerA に障害が発生した場合に役立ちます。 fileServerA が再び稼働したら、 **maxActiveDestinationTransfers** の値を 5 に増やして、可能な宛先転送の全容量を許可することができます。
- プロトコル・ブリッジ・エージェントから fileServerB へのアクティブな宛先転送の最大数は 5 です。

  このファイル・サーバーの **maxActiveDestinationTransfers** は設定されていないため、プロトコル・ブリッジ・エージェントは、このサーバーへの管理対象転送に 5 つの宛先転送スロットをすべて使用できます。

シナリオ 4

次の図では、

- agent.properties ファイルで **maxDestinationTransfers** 属性を 2 に設定しました。
- fileServerA で **maxActiveDestinationTransfers** を 2 に設定しました。
- fileServerB で **maxActiveDestinationTransfers** 属性を 2 に設定しました。
- fileServerC で **maxActiveDestinationTransfers** 属性が設定されていません。



図が示すように、**maxActiveDestinationTransfers** 属性と **maxDestinationTransfers** 属性は互いに独立しています。

各サーバーの **maxActiveDestinationTransfers** の値がチェックされます。 この値に基づき、転送がさらに続行されるか、または **WaitingForDestinationFileServerCapacity** 状態にプッシュされます。

次に、許可された転送に、既存の標準的なフローで **maxDestinationTransfers** に対するチェックが実行されます。

シナリオ 5

重要 : maxActiveDestinationTransfers 属性の値を設定するときには、**maxDestinationTransfers** 属性の値を考慮する必要があります。設定は慎重に行ってください。

これを行わないと、以下のテキストで説明されている状態が発生する可能性があります。

- グローバル **maxActiveDestinationTransfers** 属性の値が設定されていません。
- `agent.properties` ファイルに **maxDestinationTransfers**= 2 の値を設定しました。
- `fileServerA` に値 **maxActiveDestinationTransfers**= 2 を設定しました。
- `fileServerB` 上の **maxActiveDestinationTransfers** の値が設定されていません。

以下の順序でイベントが発生したとします。

- プロトコル・ブリッジ・エージェントが `fileServerA` へのファイルの転送要求を受け取ります。 プロトコル・ブリッジ・エージェントは現在何も実行していないため、この管理対象転送要求を受諾します。

  転送スロットは次のようになります。

  - 宛先転送: 1
  - `fileServerA` の宛先転送: 1
  - `fileServerB` の宛先転送: 0
- 今度は、プロトコル・ブリッジ・エージェントは別の要求を受け取ります。この要求では、プロトコル・ブリッジ・エージェントは `fileServerA` が関与する管理対象転送の宛先エージェントとして機能します。 今回もまた、プロトコル・ブリッジ・エージェントはこの要求を受諾するため、転送スロットは次のようになります。

- Destination Transfers: 2
- fileServerA の宛先転送: 2
- fileServerB の宛先転送: 0

エージェント内の 2 つの Destination Transfer スロットが占有されているため、fileServerA への転送の 1 つが完了するまで、エージェントはそれ以上の管理対象転送に参加できません。

- 少し後になって、fileServerA に障害が発生します。これにより、2 件の管理対象転送はリカバリー処理に回されます。 これらの管理対象転送が使用している Destination transfer スロットは、この期間中は使用中のままです。
- 次に、プロトコル・ブリッジ・エージェントは fileServerB へのファイルの転送要求を受け取ります。この転送用のスペースが Destination Transfers for fileServerB スロットにありますが、エージェントのすべての Destination Transfer スロットが使用されているため、転送はバックログに入れられ、後で再試行できるようになります。

その結果、fileServerA への転送の少なくとも 1 つが完了して Destination Transfer スロットが解放されるまで、 fileServerB への転送はブロックされます。

この状況が発生しないようにするには、以下を実行します。

- ファイル・サーバー上の **maxActiveDestinationTransfers** の値を **maxDestinationTransfers** 値より小さく設定して、空きスロットが残るようにします。
- または、**maxActiveDestinationTransfers** 属性の値をすべてのエンドポイント・サーバーに均等に配分します。

maxActiveDestinationTransfers 属性の値に基づくプロトコル・ブリッジ・エージェントの動作

注 : 以下の表にリストしたすべてのエラー事例において、**maxActiveDestinationTransfers** 属性の値が、有効ではない値に設定された場合、プロトコル・ブリッジ・エージェントはこの属性が設定されていないと想定します。

maxActiveDestinationTransfers	サンプル値	説明
指定なし	指定なし	転送は通常どおり発生します。 *ftp* エンドポイントの転送数に制限は課せられません。
指定あり	0	この特定の *ftp* エンドポイントには転送が許可されません。
負の値	-1	output0.log にエラーが記録されます。値 -1 は、負でない整数ではないため有効ではありません。 プロトコル・ブリッジ・エージェントは、この属性が設定されていないと想定します。
整数ではない値	abc	output0.log にエラーが記録されます。値 abc は整数としては有効でありません。 プロトコル・ブリッジ・エージェントは、この属性が設定されていないと想定します。
空	""	属性 maxActiveDestinationTransfers の値 '' は、負ではない整数としては有効ではありません。

maxActiveDestinationTransfers	サンプル値	説明
指定あり	5	この *ftp* エンドポイントに対して、どの時点でも 5 件のアクティブ転送のみ実行されることを許可します。 failTransferWhenCapacityReached 属性の値に基づいて、過剰な転送は再試行されるか拒否されます。

maxActiveDestinationTransfers 属性と failTransferWhenCapacityReached 属性を組み合わせた場合のプロトコル・ブリッジ・エージェントの振る舞い

failTransferWhenCapacityReached の値	maxActiveDestinationTransfers の値	結果
False	3	このエンドポイント・サーバーへの 3 件のアクティブ転送が許可されます。 それを超える転送は再試行されます。
True	3	このエンドポイント・サーバーへの 3 件のアクティブ転送が許可されます。 それを超える転送は拒否され、failed とマークされます。
指定なし	3	failTransferWhenCapacityReached のデフォルト値 **false** が考慮されます。 結果は、このエンドポイント・サーバーへの 3 件のアクティブ転送が許可されます。 それを超える転送は再試行されます。
ブール値以外の値	指定あり	output.log にエラーが記録されます。 failTransferWhenCapacityReached に指定された値がブール値ではありません。 failTransferWhenCapacityReached のデフォルト値が考慮されます。

maxDestinationTransfers 属性と failTransferWhenCapacityReached 属性を組み合わせた場合のプロトコル・ブリッジ・エージェントの振る舞い

failTransferWhenCapacityReached の値	maxDestinationTransfers の値	結果
True	10	同時アクティブ転送の数が 10 に達すると、11 個の番目 管理対象転送はプロトコル・ブリッジ・エージェントによって失敗します。
False	10	既存の振る舞い。

failTransferWhenCapacityReached の値	maxDestinationTransfers の値	結果
		同時アクティブ転送の数が 10 に達すると、11 個の番目 管理対象転送は、スロットが解放されるのを待機してキューに入れられます。
指定なし	10	既存の振る舞い

エラー・メッセージ

既存のメッセージ:

BFGS0082I
このエラー・メッセージは、**maxDestinationTransfers** 属性に定義された転送最大回数をプロトコル・ブリッジ・エージェントが既に実行している場合、プロトコル・ブリッジ・エージェントが転送を拒否したときにソース・エージェントの output0.log ファイルに記録されます。

新規メッセージ:

BFGSS0085I
このエラー・メッセージは、プロトコル・ブリッジ・エージェントが管理対象転送を拒否し、再試行したときにソース・エージェントの output0.log ファイルに記録されます。

BFGSS0086I
このエラー・メッセージは、プロトコル・ブリッジ・エージェントが管理対象転送を拒否して再試行し、宛先アイテムにファイル・サーバー名が含まれていない場合に、ソース・エージェントの output0.log ファイルに記録されます。

BFGSS0084E
このエラー・メッセージは、**maxActiveDestinationTransfers** 属性に指定された最大同時転送数を超えたためにプロトコル・ブリッジ・エージェントが管理対象転送を拒否し、これを failed とマークした場合に Explorer および audit.xml ファイルに記録されます。

BFGSS0087E
このエラー・メッセージは、**maxActiveDestinationTransfers** 属性に指定された最大宛先転送数を超えたために、プロトコル・ブリッジ・エージェントが管理対象転送を拒否し、これを failed とマークした場合に、Explorer および audit.xml ファイルに記録されます。

BFGSS0088W
このエラー・メッセージは、**maxActiveDestinationTransfers** 属性の値が **maxDestinationTransfers** 属性の値を超えた場合に output0.log に記録されます。

BFGSS0089I
このメッセージは、プロトコル・ブリッジ・エージェントが IBM MQ 9.3.0 以降ではないソース・エージェントで稼働している場合に、宛先プロトコル・ブリッジ・エージェントの output0.log ファイルに記録されます。

関連概念

287 ページの『プロトコル・ブリッジ』
プロトコル・ブリッジを使用すれば、Managed File Transfer (MFT) ネットワークから、MFT ネットワークの外部 (ローカル・ドメインとリモート・ロケーションの両方) にあるファイル・サーバーに格納されているファイルにアクセスできます。 このファイル・サーバーでは、FTP、FTPS、または SFTP ネットワーク・プロトコルを使用できます。 それぞれのファイル・サーバーで少なくとも 1 つの専用エージェントが必要です。 この専用エージェントは、 プロトコル・ブリッジ・エージェントとして知られています。 ブリッジ・エージェントは、複数のファイル・サーバーと相互作用できます。

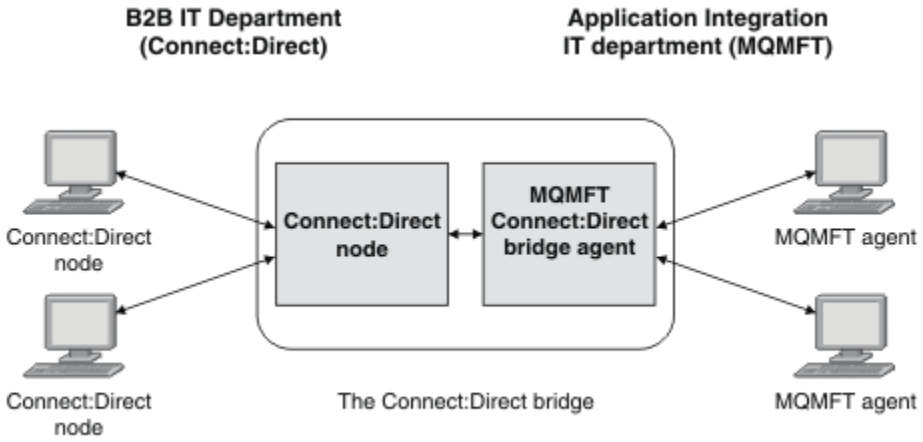
関連タスク

289 ページの『ProtocolBridgeProperties.xml ファイルを使用したプロトコル・ファイル・サーバーのプロパティーの定義』

エージェント設定ディレクトリで Managed File Transfer 提供されるファイルを使用して、`ProtocolBridgeProperties.xml` ファイル転送を行う 1 つ以上のプロトコルファイルサーバのプロパティを定義します。

Connect:Direct ブリッジ

既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、 MFT と IBM Sterling Connect:Direct の間でファイルを転送します。



この図は、 2 つの部門、 B2B IT 部門、およびアプリケーション統合 IT 部門の間の MFT Connect:Direct ブリッジを示しています。 B2B IT 部門は、 Connect:Direct を使用して、会社のビジネス・パートナーとの間でファイルを転送します。 アプリケーション統合 IT 部門では、IBM MQ をメッセージング・インフラストラクチャーとして使用するため、最近ファイル転送ソリューションとして Managed File Transfer を選択しました。

MFT Connect:Direct ブリッジを使用することにより、B2B IT 部門の Connect:Direct ネットワークと、アプリケーション統合 IT 部門の MFT ネットワークの間で相互にファイルを転送できるようになりました。 Connect:Direct ブリッジは Managed File Transfer のコンポーネントであり、 Connect:Direct ノードと通信する MFT エージェントが含まれています。 MFT エージェントは Connect:Direct ノードとの転送用の専用エージェントで、Connect:Direct ブリッジ・エージェントと呼ばれます。

Connect:Direct ブリッジは Managed File Transfer の Service コンポーネントおよび Agent コンポーネントの一部として入手可能で、以下のタスクで使用できます。

1. Managed File Transfer コマンドを使用して、MFT エージェントから Connect:Direct ノードへの単一ファイルまたは複数ファイルの転送を開始します。
2. Managed File Transfer コマンドを使用して、Connect:Direct ノードから MFT エージェントへの単一ファイルまたは複数ファイルの転送を開始します。
3. Managed File Transfer コマンドを使用して、ユーザー定義 Connect:Direct プロセスを開始するファイル転送を開始します。
4. Connect:Direct プロセスを使用して、MFT のファイル転送要求を送信します。

Connect:Direct ブリッジでは、Connect:Direct ノードを転送元または転送先とするファイルの転送のみが可能です。 Connect:Direct ブリッジでローカル・ファイル・システムを転送元または転送先とするファイル転送を実行できるのは、その転送が Connect:Direct プロセスによって実行される転送の一部になっている場合に限られます。

z/OS Connect:Direct ブリッジを使用して、z/OS システム上の Connect:Direct ノード上にあるデータ・セットとの間で転送することができます。 Managed File Transfer エージェントだけがかかわっているデータ・セット転送と比較すると、動作にいくらかの違いがあります。 詳しくは、 z/OS Connect:Direct ノードとの間のデータ・セット転送を参照してください。

サポートされているプラットフォーム

Connect:Direct ブリッジは、 MFT Connect:Direct ブリッジ・エージェントと Connect:Direct ノードで構成されています。 エージェントは、 Windows および x86-64 Linux サポートされます。 このノードは、「 for Windows および IBM Sterling Connect:Direct for UNIX IBM Sterling Connect:Direct サポートされているプラットフォームでサポートされています。 Connect:Direct ブリッジ・エージェントを作成し、そのエージェントと通信できるように Connect:Direct ノードを構成する方法については、Connect:Direct ブリッジの構成を参照してください。

Connect:Direct ブリッジは、 Connect:Direct (Windows の場合) または Connect:Direct (UNIX の場合)、あるいは Connect:Direct (z/OS Service インストールの場合) の一部として実行されている Connect:Direct ノードとの間でファイルを転送できます。 サポートされている Connect:Direct のバージョンの詳細については、Web ページ「System Requirements for IBM MQ」を参照してください。

Connect:Direct ブリッジを構成するエージェントとノードは、同じシステムに存在しているか、共用 NFS マウントなどによって同じファイル・システムにアクセスできる状態になっている必要があります。 このファイル・システムは、Connect:Direct ブリッジに関係するファイル転送中に、**cdTmpDir** パラメーターで定義されたディレクトリーにファイルを一時的に保管するために使用されます。 Connect:Direct ブリッジ・エージェントと Connect:Direct ブリッジ・ノードでは、同じパス名を使用してこのディレクトリーを指定できる必要があります。 例えば、エージェントとノードが別個の Windows システムにある場合、共有ファイル・システムをマウントするためにそれらのシステムで同じドライブ名が使用されている必要があります。 以下の構成を使用すると、エージェントとノードで同じパス名を使用できます。

- エージェントとノードが、Windows または Linux for x86-64 のいずれかを実行する同じシステム上にある
- エージェントが Linux for x86-64 上にあり、ノードが AIX 上にある
- エージェントが Windows システム上にあり、ノードがそれとは別の Windows システム上にある

以下の構成を使用すると、エージェントとノードで同じパス名を使用できません。

- エージェントが Linux for x86-64 上にあり、ノードが Windows 上にある
- エージェントが Windows 上にあり、ノードが UNIX 上にある

Connect:Direct ブリッジのインストールを計画する際には、これらの制約事項を考慮してください。

関連概念

320 ページの『Connect:Direct ノードを転送元および転送先とする転送のリカバリーおよび再始動』
転送中に、Managed File Transfer が IBM Sterling Connect:Direct ノードに接続できなくなる場合があります。例えば、ノードが使用不可になる場合です。 その場合、Managed File Transfer が転送のリカバリーを試行するか、転送が失敗してエラー・メッセージが生成されます。

321 ページの『ファイル転送要求からのユーザー定義 Connect:Direct プロセスの送信』
ファイル転送の一部としてユーザー定義 Connect:Direct プロセスを呼び出す Connect:Direct ブリッジ・エージェントを経由する転送の転送要求を送信できます。

325 ページの『Connect:Direct プロセスを使用して Managed File Transfer 転送要求を送信する操作』
Connect:Direct プロセスから Connect:Direct ブリッジ・エージェントに転送要求を送信できます。 Managed File Transfer には、Connect:Direct プロセスの **RUN TASK** ステートメントから呼び出すことができるコマンドが用意されています。

関連タスク

Connect:Direct ブリッジの構成

313 ページの『Connect:Direct ノードへのファイルの転送』
Connect:Direct ブリッジを使用して、 Managed File Transfer エージェントから Connect:Direct ノードにファイルを転送できます。 Connect:Direct ブリッジ・エージェントを宛先エージェントとして指定し、*connect_direct_node_name*:*file_path* という形式で宛先ファイルを指定することにより、転送の宛先として Connect:Direct ードを指定します。

314 ページの『Connect:Direct ノードからのファイルの転送』
Connect:Direct ブリッジを使用して、 Connect:Direct ノードから Managed File Transfer Agent にファイルを転送できます。 Connect:Direct ブリッジ・エージェントをソース・エージェントとして指定し、ソース

仕様を *connect_direct_node_name*:*file_path* 形式で指定することにより、転送のソースとして Connect:Direct ノードを指定することができます。

316 ページの『Connect:Direct ノードへの複数ファイルの転送』
Connect:Direct ブリッジを使用して、 Managed File Transfer Agent から Connect:Direct ノードに複数のファイルを転送できます。 複数ファイル転送の宛先として Connect:Direct ノードを使用するには、 Connect:Direct ブリッジ・エージェントを宛先エージェントとして指定し、 *connect_direct_node_name*:*directory_path* という形式で宛先ディレクトリーを指定します。

317 ページの『Transferring multiple files from a Connect:Direct node』
You can transfer multiple files from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the multiple file transfer by specifying the Connect:Direct bridge agent as the source agent and specifying one or more source specifications in the form *connect_direct_node_name*:*file_path*.

318 ページの『ワイルドカードを使用した Connect:Direct への複数ファイルの転送』
Managed File Transfer エージェントから Connect:Direct ノードへ複数のファイルを転送するには、 Connect:Direct ブリッジを使用します。 **fteCreateTransfer** コマンドに指定するソース指定内では、ワイルドカード文字を使用できます。 ワイルドカードを使用するすべての Managed File Transfer の転送と同様、ファイル・パスの最後の部分にのみワイルドカード文字を使用できます。 例えば、`/abc/def*` は有効なファイル・パスで、 `/abc*/def` は無効です。

Connect:Direct ブリッジのトラブルシューティング

関連資料

fteCreateCDAgent: Connect:Direct ブリッジ・エージェントの作成
Connect:Direct ブリッジ・エージェントの制約事項

Connect:Direct ノードへのファイルの転送

Connect:Direct ブリッジを使用して、 Managed File Transfer エージェントから Connect:Direct ノードにファイルを転送できます。 Connect:Direct ブリッジ・エージェントを宛先エージェントとして指定し、 *connect_direct_node_name*:*file_path* という形式で宛先ファイルを指定することにより、転送の宛先として Connect:Direct ードを指定します。

始める前に

ファイルを転送する前に、Managed File Transfer のコンポーネントである Connect:Direct ブリッジを構成する必要があります。 詳しくは、Connect:Direct ブリッジの構成を参照してください。

このタスクについて

この例では、Connect:Direct ブリッジ・エージェントは CD_BRIDGE という名前です。 ソース・エージェントは、FTE_AGENT という名前であり、WMQFTE のどのバージョンでもかまいません。 宛先 Connect:Direct ノードは CD_NODE1 という名前です。 転送されるファイルは、FTE_AGENT が配置されているシステム上のファイル・パス `/home/helen/file.log` にあります。 このファイルは、 CD_NODE1 が実行されているシステム上のファイル・パス`/files/data.log` に転送されます。

手順

1. **-df** (宛先ファイル) パラメーターに *connect_direct_node_name*:*file_path* 形式の値を使用し、**-da** (宛先エージェント) パラメーターの値として Connect:Direct ブリッジ・エージェントの名前を指定して、fteCreateTransfer コマンドを使用します。

   注: *connect_direct_node_name* によって指定される Connect:Direct ノードは、 Connect:Direct ブリッジの一部として作動する Connect:Direct ノードではなく、ファイルを転送するノードです。

   ```
   fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
     -df CD_NODE1:/files/data.log /home/helen/file.log
   ```

   詳しくは、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。

2. ソース・エージェント FTE_AGENT がファイルを Connect:Direct ブリッジ・エージェント CD_BRIDGE に転送します。 ファイルは、Connect:Direct ブリッジ・エージェントが実行されているシステム上の、cdTmpDir エージェント・プロパティーで定義されている場所に一時的に保管されます。
Connect:Direct ブリッジ・エージェントが Connect:Direct ノード CD_NODE1 にファイルを転送します。

関連概念

311 ページの『Connect:Direct ブリッジ』
既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、 MFT と IBM Sterling Connect:Direct の間でファイルを転送します。

関連タスク

314 ページの『Connect:Direct ノードからのファイルの転送』
Connect:Direct ブリッジを使用して、 Connect:Direct ノードから Managed File Transfer Agent にファイルを転送できます。 Connect:Direct ブリッジ・エージェントをソース・エージェントとして指定し、ソース仕様を *connect_direct_node_name*:*file_path* 形式で指定することにより、転送のソースとして Connect:Direct ノードを指定することができます。

関連資料

MFT `agent.properties` ファイル

Connect:Direct ノードからのファイルの転送

Connect:Direct ブリッジを使用して、 Connect:Direct ノードから Managed File Transfer Agent にファイルを転送できます。 Connect:Direct ブリッジ・エージェントをソース・エージェントとして指定し、ソース仕様を *connect_direct_node_name*:*file_path* 形式で指定することにより、転送のソースとして Connect:Direct ノードを指定することができます。

始める前に

ファイルを転送する前に、Managed File Transfer のコンポーネントである Connect:Direct ブリッジを構成する必要があります。 Connect:Direct ブリッジの構成を参照してください。

このタスクについて

この例では、Connect:Direct ブリッジ・エージェントは CD_BRIDGE という名前です。 宛先エージェントは、FTE_AGENT という名前であり、Managed File Transfer のどのバージョンでもかまいません。 ソース Connect:Direct ノードは CD_NODE1 という名前です。 転送されるファイルは、 CD_NODE1 が配置されているシステム上のファイル・パス `/home/brian/in.file` にあります。 このファイルは、 FTE_AGENT が実行されているシステム上のファイル・パス `/files/out.file` に転送されます。

手順

fteCreateTransfer コマンドは、ソース仕様の値を *connect_direct_node_name*:*file_path* の形式で指定し、 **-sa** パラメーターの値を Connect:Direct ブリッジ・エージェントの名前として指定して使用します。

注 : *connect_direct_node_name* によって指定される Connect:Direct ノードは、Connect:Direct ブリッジの一部として作動する Connect:Direct ノードではなく、ファイルの転送元とするノードです。 以下に例を示します。

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_AGENT
                  -df /files/out.file CD_NODE1:/home/brian/in.file
```

詳しくは、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。

タスクの結果

Connect:Direct ブリッジ・エージェント CD_BRIDGE が、Connect:Direct ノード CD_NODE1 からのファイルを要求します。 Connect:Direct ノードが Connect:Direct ブリッジにファイルを送信します。
Connect:Direct ノードからのファイルの転送中、Connect:Direct ブリッジは、`cdTmpDir` エージェント・

プロパティーで定義されている場所に一時的にそのファイルを保管します。 Connect:Direct ノードから Connect:Direct ブリッジへのファイル転送が完了すると、Connect:Direct ブリッジは、そのファイルを宛先エージェント FTE_AGENT に送信し、一時ロケーションからそのファイルを削除します。

関連概念

311 ページの『Connect:Direct ブリッジ』

既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、 MFT と IBM Sterling Connect:Direct の間でファイルを転送します。

関連資料

MFT `agent.properties` ファイル

z/OS 上の Connect:Direct ノードへのデータ・セットの転送

Windows システムまたは Linux システムにある Connect:Direct ブリッジを使用して、 z/OS 上の Managed File Transfer エージェントから z/OS 上の Connect:Direct ノードにデータ・セットを転送できます。

始める前に

ファイルを転送する前に、Managed File Transfer のコンポーネントである Connect:Direct ブリッジを構成する必要があります。 Connect:Direct ブリッジの構成を参照してください。

このタスクについて

この例では、**-df** パラメーターを使用して、転送の宛先を指定します。 **-df** パラメーターは、転送のソース・エージェントが Managed File Transfer のどのバージョンの場合でも有効です。 代わりに **-ds** パラメーターを使用できます。 ソース・エージェントは、FTE_ZOS1 という名前であり、Managed File Transfer のエージェントです。 Connect:Direct ブリッジ・エージェントは、CD_BRIDGE という名前であり、Linux システムにあります。 宛先 Connect:Direct ノードは CD_ZOS2 という名前です。 ソース・エージェントも宛先 Connect:Direct ノードも、z/OS システムにあります。 転送するデータ・セットは、FTE_ZOS1 が配置されているシステムの `//FTEUSER.SOURCE.LIB` にあります。 そのデータ・セットを、CD_ZOS2 が配置されているシステムのデータ・セット `//CDUSER.DEST.LIB` に転送します。

手順

1. **-df** パラメーターの値を *connect_direct_node_name*:*data_set_name*;*attributes* という形式で指定し、**-da** (宛先エージェント) パラメーターの値を Connect:Direct ブリッジ・エージェントの名前として指定して、fteCreateTransfer コマンドを使用します。

   connect_direct_node_name によって指定される Connect:Direct ノードは、 Connect:Direct ブリッジの一部として作動する Connect:Direct ノードではなく、データ・セットを転送するノードです。

   data_set_name で指定するデータ・セット名は、相対名ではなく絶対名でなければなりません。 Connect:Direct では、データ・セット名の接頭部としてユーザー名が追加されません。

   ```
   fteCreateTransfer -sa FTE_ZOS1 -sm QM_ZOS
                     -da CD_BRIDGE -dm QM_BRIDGE
                     -df CD_ZOS2://'CDUSER.DEST.LIB;BLKSIZE(8000);LRECL(80)'
                     //'FTEUSER.SOURCE.LIB'
   ```

   詳しくは、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。
2. ソース・エージェント FTE_ZOS1 がデータ・セット内のデータを Connect:Direct ブリッジ・エージェント CD_BRIDGE に転送します。 そのデータは、Connect:Direct ブリッジ・エージェントが稼働しているシステムでフラット・ファイルとして一時的に格納されます。格納場所は、cdTmpDir エージェント・プロパティーで定義されている場所になります。 Connect:Direct ブリッジ・エージェントが Connect:Direct ノード CD_ZOS2 にデータを転送します。 転送が完了すると、Connect:Direct ブリッジ・エージェントが稼働しているシステムからそのフラット・ファイルが削除されます。

関連概念

311 ページの『Connect:Direct ブリッジ』

既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、 MFT と IBM Sterling Connect:Direct の間でファイルを転送します。

関連タスク

Connect:Direct ノードとの間のデータ・セット転送

関連資料

MFT で使用できない BPXWDYN のプロパティー

Connect:Direct ノードへの複数ファイルの転送

Connect:Direct ブリッジを使用して、 Managed File Transfer Agent から Connect:Direct ノードに複数のファイルを転送できます。 複数ファイル転送の宛先として Connect:Direct ノードを使用するには、 Connect:Direct ブリッジ・エージェントを宛先エージェントとして指定し、 *connect_direct_node_name*:*directory_path* という形式で宛先ディレクトリーを指定します。

始める前に

ファイルを転送する前に、Managed File Transfer のコンポーネントである Connect:Direct ブリッジを構成する必要があります。 Connect:Direct ブリッジの構成を参照してください。

このタスクについて

この例では、ソース・エージェントは FTE_AGENT という名前です。 Connect:Direct ブリッジ・エージェントは CD_BRIDGE という名前です。 宛先 Connect:Direct ノードは CD_NODE1 という名前です。 転送されるファイルは、 FTE_AGENT が配置されているシステム上の `/home/jack/data.log`, `/logs/log1.txt`, および`/results/latest` です。 これらのファイルは、 CD_NODE1 が実行されているシステム上のディレクトリー `/in/files` に転送されます。

手順

-dd (宛先ディレクトリー) パラメーターに *connect_direct_node_name*:*directory_path* 形式の値を使用して、fteCreateTransfer コマンドを使用します。 **-da** (宛先エージェント) パラメーターの値を、Connect:Direct ブリッジ・エージェントの名前として指定します。

注 : *connect_direct_node_name* によって指定される Connect:Direct ノードは、Connect:Direct ブリッジの一部として作動する Connect:Direct ノードではなく、ファイルを転送するノードです。

```
fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
                  -dd CD_NODE1:/in/files /home/jack/data.log
                  /logs/log1.txt /results/latest
```

詳しくは、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。

タスクの結果

ソース・エージェント FTE_AGENT が最初のファイルを Connect:Direct ブリッジ・エージェント CD_BRIDGE に転送します。 Connect:Direct ブリッジ・エージェントがこのファイルを、cdTmpDir プロパティーで定義された場所に一時的に保管します。 ソース・エージェントから Connect:Direct ブリッジにファイルが完全に転送されると、Connect:Direct ブリッジ・エージェントがそのファイルを、cdNode エージェント・プロパティーで定義された Connect:Direct ノードに送信します。 このノードがファイルを、宛先 Connect:Direct ノード CD_NODE1 に送信します。 2 つの Connect:Direct ノード間で転送が完了すると、 Connect:Direct ブリッジ・エージェントが一時ロケーションからファイルを削除します。 指定されたすべてのソース・ファイルごとに、このプロセスが繰り返されます。

関連概念

311 ページの『Connect:Direct ブリッジ』

既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、 MFT と IBM Sterling Connect:Direct の間でファイルを転送します。

関連タスク

313 ページの『Connect:Direct ノードへのファイルの転送』
Connect:Direct ブリッジを使用して、 Managed File Transfer エージェントから Connect:Direct ノードにファイルを転送できます。 Connect:Direct ブリッジ・エージェントを宛先エージェントとして指定し、 *connect_direct_node_name*:*file_path* という形式で宛先ファイルを指定することにより、転送の宛先として Connect:Direct ードを指定します。

318 ページの『ワイルドカードを使用した Connect:Direct への複数ファイルの転送』
Managed File Transfer エージェントから Connect:Direct ノードへ複数のファイルを転送するには、 Connect:Direct ブリッジを使用します。 **fteCreateTransfer** コマンドに指定するソース指定内では、ワイルドカード文字を使用できます。 ワイルドカードを使用するすべての Managed File Transfer の転送と同様、ファイル・パスの最後の部分にのみワイルドカード文字を使用できます。 例えば、`/abc/def*` は有効なファイル・パスで、 `/abc*/def` は無効です。

314 ページの『Connect:Direct ノードからのファイルの転送』
Connect:Direct ブリッジを使用して、 Connect:Direct ノードから Managed File Transfer Agent にファイルを転送できます。 Connect:Direct ブリッジ・エージェントをソース・エージェントとして指定し、ソース仕様を *connect_direct_node_name*:*file_path* 形式で指定することにより、転送のソースとして Connect:Direct ノードを指定することができます。

317 ページの『Transferring multiple files from a Connect:Direct node』
You can transfer multiple files from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the multiple file transfer by specifying the Connect:Direct bridge agent as the source agent and specifying one or more source specifications in the form *connect_direct_node_name*:*file_path*.

関連資料

MFT `agent.properties` ファイル

z/OS Transferring multiple files from a Connect:Direct node

You can transfer multiple files from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the multiple file transfer by specifying the Connect:Direct bridge agent as the source agent and specifying one or more source specifications in the form *connect_direct_node_name*:*file_path*.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of Managed File Transfer. See Configuring the Connect:Direct bridge.

About this task

In this example, the Connect:Direct bridge agent is called CD_BRIDGE. The destination agent is called FTE_Z, and is running on a z/OS system. The source Connect:Direct node is called CD_NODE1. The files to be transferred are located at the file paths `/in/file1`, `/in/file2`, and `/in/file3` on the system where CD_NODE1 is located. The files are transferred to the partitioned data set `//OBJECT.LIB` on the system where FTE_Z is running.

Procedure

Use the fteCreateTransfer command with the values for the source specifications in the form *connect_direct_node_name*:*file_path* and the value of the **-sa** parameter specified as the name of the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the files to be transferred from, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_Z
                  -dp //'OBJECT.LIB' CD_NODE1:/in/file1
                  CD_NODE1:/in/file2 CD_NODE1:/in/file3
```

For more information, see **fteCreateTransfer**: start a new file transfer.

Results

The Connect:Direct bridge agent CD_BRIDGE requests the first file from the Connect:Direct node CD_NODE1. The Connect:Direct node sends the file to the Connect:Direct bridge. While the file is being transferred from the Connect:Direct node, the Connect:Direct bridge stores the file temporarily in the location defined by the `cdTmpDir` agent property. When the file has finished transferring from the Connect:Direct node to the Connect:Direct bridge, the Connect:Direct bridge sends the file to the destination agent FTE_Z and then deletes the file from the temporary location. This process is repeated for each specified source file.

Related concepts

“Connect:Direct ブリッジ” on page 311
既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、 MFT と IBM Sterling Connect:Direct の間でファイルを転送します。

Related reference

The MFT `agent.properties` file

ワイルドカードを使用した Connect:Direct への複数ファイルの転送

Managed File Transfer エージェントから Connect:Direct ノードへ複数のファイルを転送するには、 Connect:Direct ブリッジを使用します。 **fteCreateTransfer** コマンドに指定するソース指定内では、ワイルドカード文字を使用できます。 ワイルドカードを使用するすべての Managed File Transfer の転送と同様、ファイル・パスの最後の部分にのみワイルドカード文字を使用できます。 例えば、`/abc/def*` は有効なファイル・パスで、 `/abc*/def` は無効です。

始める前に

ファイルを転送する前に、Managed File Transfer のコンポーネントである Connect:Direct ブリッジを構成する必要があります。 詳しくは、Connect:Direct ブリッジの構成を参照してください。

このタスクについて

この例では、ソース・エージェントは FTE_AGENT という名前で、Connect:Direct ブリッジ・エージェントは CD_BRIDGE という名前です。 宛先 Connect:Direct ノードは CD_NODE1 という名前です。 転送されるファイルは、 FTE_AGENT が配置されているシステム上の`/reports` ディレクトリーにあります。 名前が `report` で始まり、その後に 2 つの文字と接尾部 `.log` が付いたファイルのみが転送されます。 例えば、ファイル `/reports/report01.log` は転送されますが、ファイル `/reports/report1.log` は転送されません。 これらのファイルは、 CD_NODE1 が実行されているシステム上のディレクトリー `/home/fred` に転送されます。

手順

1. **-dd** (宛先ディレクトリー) パラメーターに *connect_direct_node_name*:*directory_path* 形式の値を使用して、fteCreateTransfer コマンドを使用します。 **-da** (宛先エージェント) パラメーターには、Connect:Direct ブリッジ・エージェントを指定します。

注: *connect_direct_node_name* によって指定される Connect:Direct ノードは、Connect:Direct ブリッジの一部として作動する Connect:Direct ノードではなく、ファイルを転送するノードです。

```
fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
                  -dd CD_NODE1:/home/fred "/reports/report??.log"
```

詳しくは、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。

2. ソース・エージェント FTE_AGENT は、パターン /reports/report??.log に一致する最初のファイルを Connect:Direct ブリッジ・エージェント CD_BRIDGE に転送します。 Connect:Direct ブリッジ・エージェントがこのファイルを、cdTmpDir プロパティーで定義された場所に一時的に保管します。 ソース・エージェントから Connect:Direct ブリッジにファイルが完全に転送されると、Connect:Direct ブリッジ・エージェントがそのファイルを、cdNode エージェント・プロパティーで定義された Connect:Direct ノードに送信します。 このノードがファイルを、宛先 Connect:Direct ノード CD_NODE1 に送信します。 2 つの Connect:Direct ノード間で転送が完了すると、Connect:Direct ブリッジ・エージェントが一時ロケーションからファイルを削除します。このプロセスは、ワイルドカード・パターン/reports/report??.log に一致するソース・ファイルごとに繰り返されます。

   注: パターン/reports/report??.log に一致するファイルのリストは、ソース・エージェント FTE_AGENT が配置されているシステムのオペレーティング・システムによって異なります。

   - ソース・エージェントが Windows オペレーティング・システムを使用するシステム上にある場合、パターン・マッチングは大/小文字を区別しません。 このパターンは, 大文字小文字にかかわらず, ファイル名の後に 2 文字とサフィックス .log として , が続く report 形式の /reports ディレクトリ内のすべてのファイルにマッチします。 例えば、 は一致しています。 Report99.Log
   - ソース・エージェントが Linux または UNIX オペレーティング・システムを使用するシステム上にある場合、パターン・マッチングは大/小文字を区別します。 パターンが一致するのは、 /reports ディレクトリー内のファイルの後に、 report という形式のファイル名と、その後に続く 2 つの文字と接尾部 .log を持つものだけです。 例えば、 reportAB.log は一致していますが、 reportAB.LOG と Report99.Log は一致していません。

関連概念

311 ページの『Connect:Direct ブリッジ』
既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、 MFT と IBM Sterling Connect:Direct の間でファイルを転送します。

関連タスク

MFT でのワイルドカード文字の使用

313 ページの『Connect:Direct ノードへのファイルの転送』
Connect:Direct ブリッジを使用して、 Managed File Transfer エージェントから Connect:Direct ノードにファイルを転送できます。 Connect:Direct ブリッジ・エージェントを宛先エージェントとして指定し、 *connect_direct_node_name*:*file_path* という形式で宛先ファイルを指定することにより、転送の宛先として Connect:Direct ードを指定します。

316 ページの『Connect:Direct ノードへの複数ファイルの転送』
Connect:Direct ブリッジを使用して、 Managed File Transfer Agent から Connect:Direct ノードに複数のファイルを転送できます。 複数ファイル転送の宛先として Connect:Direct ノードを使用するには、 Connect:Direct ブリッジ・エージェントを宛先エージェントとして指定し、 *connect_direct_node_name*:*directory_path* という形式で宛先ディレクトリーを指定します。

317 ページの『Transferring multiple files from a Connect:Direct node』
You can transfer multiple files from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the multiple file transfer by specifying the Connect:Direct bridge agent as the source agent and specifying one or more source specifications in the form *connect_direct_node_name*:*file_path*.

関連資料

MFT agent.properties ファイル

Connect:Direct ノードを転送元および転送先とする転送のリカバリーおよび再始動

転送中に、Managed File Transfer が IBM Sterling Connect:Direct ノードに接続できなくなる場合があります。例えば、ノードが使用不可になる場合です。 その場合、Managed File Transfer が転送のリカバリーを試行するか、転送が失敗してエラー・メッセージが生成されます。

Connect:Direct ノードが使用不可になる場合

Connect:Direct ノードが、ネットワーク障害や電源異常などが原因で使用不可になると、Managed File Transfer は以下の方法でファイル転送をリカバリーします。

- Managed File Transfer がこの転送要求の一部として Connect:Direct ノードに以前に正常に接続されていない場合、 **cdMaxConnectionRetries** および **recoverableTransferRetryInterval properties** の値によって決定された時間だけ転送が再試行されます。 これらのプロパティーは、Connect:Direct ブリッジ・エージェントの `agent.properties` ファイルで指定されます。 試行の失敗回数が **cdMaxConnectionRetries property** の値に達すると、転送が失敗し、エラー・メッセージが生成されます。 デフォルトでは、転送は無限に試行され、試行の間隔は 60 秒です。
- この転送要求の一部として、Managed File Transfer がこれまでこの Connect:Direct ノードとの接続に成功している場合、**cdMaxPartialWorkConnectionRetries** プロパティーおよび **recoverableTransferRetryInterval** プロパティーの値によって決定される時間の間、転送が再試行されます。 失敗した試行の回数が **cdMaxPartialWorkConnectionRetries** プロパティーの値に達すると転送が失敗し、エラー・メッセージが生成されます。 デフォルトでは、転送は無限に試行され、試行の間隔は 60 秒です。
- 特定のタイプの Connect:Direct ノード障害 (例えば、強制的に停止されているノード) の場合、ノードがリカバリーすると、 Connect:Direct プロセスは `Held Due to Error (HE)` 状況になります。 ノードがリカバリーすると、 Managed File Transfer は、ファイル転送に関連し、状況が HE であるすべての Connect:Direct プロセスを自動的に再開します。
- 転送が失敗すると、転送に関連したすべての一時ファイルが、Connect:Direct ブリッジをホストするシステムから削除されます。 これらの一時ファイルの場所は、**cdTmpDir** プロパティーによって定義されています。
- Managed File Transfer から Connect:Direct への転送で、ソースの後処理として削除が指定されている場合、転送が失敗するとソース・ファイルは削除されません。

Connect:Direct ノードのユーザー資格情報が無効な場合

Managed File Transfer から Connect:Direct ノードへの接続で、ユーザーの資格情報がノードによって拒否されたために接続が失敗すると、転送が失敗し、エラー・メッセージが生成されます。 このシチュエーションでは、Connect:Direct ノードに対して、正しいユーザー資格情報が提供されていることを確認します。詳しくは、Connect:Direct の資格情報のマップを参照してください。

Connect:Direct ブリッジ・エージェントが使用不可になる場合

Connect:Direct ブリッジ・エージェントが使用不可になると、すべての進行中のファイル転送は、標準の Managed File Transfer 転送と同様にリカバリーされます。 詳しくは、327 ページの『MFT のリカバリーと再始動』を参照してください。

関連概念

311 ページの『Connect:Direct ブリッジ』
既存の IBM Sterling Connect:Direct ネットワークとの相互間で、ファイルを転送することができます。 Managed File Transfer のコンポーネントである Connect:Direct ブリッジを使用して、MFT と IBM Sterling Connect:Direct の間でファイルを転送します。

327 ページの『MFT のリカバリーと再始動』
エージェントまたはキュー・マネージャーが何らかの理由 (例えば、電源やネットワークの障害など) で使用できない場合、Managed File Transfer は、以下のシナリオで示すようにリカバリーを行います。

関連タスク
Connect:Direct ブリッジの構成
関連資料
MFT agent.properties ファイル

ファイル転送要求からのユーザー定義 Connect:Direct プロセスの送信

ファイル転送の一部としてユーザー定義 Connect:Direct プロセスを呼び出す Connect:Direct ブリッジ・エージェントを経由する転送の転送要求を送信できます。

Connect:Direct ブリッジを経由する転送のファイル転送要求を送信すると、デフォルトでは、Connect:Direct ブリッジ・エージェントがリモート Connect:Direct ノードとの間でファイルを転送するための Connect:Direct プロセスを生成します。

しかし、ConnectDirectProcessDefinition.xml ファイルを使用することで、代わりにユーザー定義プロセス Connect:Direct を呼び出すように Connect:Direct ブリッジエージェントを設定することができます。

ConnectDirectProcessDefinition.xml ファイル

fteCreateCDAgent コマンドを使用すると、エージェントの構成ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name* にファイル ConnectDirectProcessDefinitions.xml が作成されます。 Connect:Direct ブリッジ・エージェントからユーザー定義 Connect:Direct プロセスを呼び出すには、まずこのファイルを編集してプロセス定義をセットアップする必要があります。

このファイルでは、転送の一部として呼び出す 1 つ以上の Connect:Direct プロセスの場所を組み込んだプロセス・セットを 1 つ以上定義します。 それぞれのプロセス・セットには、いくつかの条件を組み込みます。 転送がプロセス・セットのすべての条件を満たす場合、そのプロセス・セットを使用して、転送で呼び出す Connect:Direct プロセスが指定されます。 詳しくは、322 ページの『ConnectDirectProcessDefinition.xml ファイルを使用して、開始する Connect:Direct プロセスを指定する操作』を参照してください。

組み込みシンボリック変数

Managed File Transfer が定義する組み込みシンボリック変数を使用して、値をユーザー定義 Connect:Direct プロセスに置換できます。 Connect:Direct の命名規則に合わせて、Managed File Transfer で使用するすべての組み込みシンボリック変数は、%FTE の後に 5 つの大文字英数字を付けた形式になっています。

Connect:Direct ノードから Connect:Direct ブリッジ・システムにファイルを転送するプロセスを作成する場合、Connect:Direct プロセスの TO FILE の値として組み込み変数 %FTETFILE を使用する必要があります。 Connect:Direct ブリッジ・システムから Connect:Direct ノードにファイルを転送するプロセスを作成する場合、Connect:Direct プロセスの FROM FILE の値として組み込み変数 %FTEFFILE を使用する必要があります。 これらの変数には、Connect:Direct ブリッジ・エージェントが Managed File Transfer ネットワークを転送先および転送元とする転送で使用する一時ファイル・パスが含まれます。

組み込みシンボリック変数の詳細については、Connect:Direct の製品資料を参照してください。

サンプル Connect:Direct プロセス

Managed File Transfer では、サンプル Connect:Direct プロセスが提供されています。 これらのサンプルは、 *MQ_INSTALLATION_PATH*/mqft/samples/ConnectDirectProcessTemplates というディレクトリーにあります。

関連タスク
322 ページの『ConnectDirectProcessDefinition.xml ファイルを使用して、開始する Connect:Direct プロセスを指定する操作』
Managed File Transfer 転送の一部として開始する Connect:Direct プロセスを指定します。 Managed File Transfer には、プロセス定義を指定するために編集できる XML ファイルが用意されています。

323 ページの『Managed File Transfer によって呼び出される Connect:Direct プロセスでの組み込みシンボリック変数の使用』
Managed File Transfer 転送からユーザー定義の Connect:Direct プロセスを呼び出し、プロセス定義内の組み込みシンボリック変数を使用して、転送から Connect:Direct プロセスに情報を渡すことができます。

関連資料

Connect:Direct プロセスの定義ファイルのフォーマット

ユーザー定義 Connect:Direct プロセスで使用する置換変数

ConnectDirectProcessDefinition.xml ファイルを使用して、開始する *Connect:Direct* プロセスを指定する操作

Managed File Transfer 転送の一部として開始する Connect:Direct プロセスを指定します。 Managed File Transfer には、プロセス定義を指定するために編集できる XML ファイルが用意されています。

このタスクについて

fteCreateCDAgent コマンドを使用すると、エージェントの構成ディレクトリー *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name* にファイル ConnectDirectProcessDefinitions.xml が作成されます。 Connect:Direct ブリッジ・エージェントからユーザー定義 Connect:Direct プロセスを呼び出すには、まずこのファイルを編集してプロセス定義をセットアップする必要があります。

Connect:Direct ブリッジを経由した転送の一部として呼び出すように指定するプロセスごとに、以下の手順を実行します。

手順

1. 転送の一部として Connect:Direct ブリッジ・エージェントから呼び出す Connect:Direct プロセスを定義し、プロセス・テンプレートをファイルに保存します。
2. *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name*/ConnectDirectProcessDefinitions.xml ファイルをテキスト・エディターで開きます。
3. <processSet> エレメントを作成します。
4. <processSet>エレメントの内部に、<condition>エレメントを作成します。
5. <condition>エレメント内で、ステップ 1 で定義した Connect:Direct プロセスを呼び出すために転送要求が一致する必要がある条件を定義するエレメントを 1 つ以上作成します。 これらのエレメントは、<match> エレメントまたは <defined> エレメントのいずれかです。
   - <match> エレメントを使用して、変数の値がパターンに一致する必要があることを指定します。 以下の属性を使用して <match> エレメントを作成します。
     - variable - 値を比較する変数の名前。 この変数は、組み込みシンボルです。 詳しくは、ユーザー定義 Connect:Direct プロセスで使用する置換変数を参照してください。
     - value - 指定した変数の値と比較するパターン。
     - オプション: pattern - value 属性の値で使用するパターンのタイプ。 このパターン・タイプは、wildcard または regex のいずれかになります。 この属性は任意指定であり、デフォルトは wildcard です。
   - <defined>エレメントを使用して、変数に値を定義する必要があることを指定します。 以下の属性を使用して<defined>エレメントを作成します。
     - variable - 値が定義されていなければならない変数の名前。 この変数は、組み込みシンボルです。 詳しくは、ユーザー定義 Connect:Direct プロセスで使用する置換変数を参照してください。

   <condition>エレメント内に指定された条件は、論理 AND と結合されます。 Connect:Direct ブリッジ・エージェントがこの <processSet>エレメントによって指定されたプロセスを呼び出すには、すべ

ての条件を満たす必要があります。 <condition>エレメントを指定しない場合、プロセス・セットはすべての転送に一致します。

6. <processSet>エレメントの内部に、<process>エレメントを作成します。
7. <process>エレメントの内部に、<transfer>エレメントを作成します。

   transfer エレメントでは、Connect:Direct ブリッジ・エージェントが転送の一部として呼び出す Connect:Direct プロセスを指定します。 以下の属性を使用して<transfer>エレメントを作成します。

   - process- -ステップ 1 で定義した Connect:Direct プロセスの場所です。 このファイルの場所は、絶対パスで指定されます。または、 *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name* ディレクトリーに対する相対パスで指定します。

タスクの結果

Connect:Direct ブリッジ・エージェントは、条件に合致する項目を検索するときに、ファイルの先頭から末尾に向かって検索します。 最初に見つかった一致が使用されます。

関連タスク

Connect:Direct ブリッジの構成

関連資料

Connect:Direct プロセスの定義ファイルのフォーマット

fteCreateCDAgent: Connect:Direct ブリッジ・エージェントの作成

Managed File Transfer によって呼び出される *Connect:Direct* プロセスでの組み込みシンボリック変数の使用

Managed File Transfer 転送からユーザー定義の Connect:Direct プロセスを呼び出し、プロセス定義内の組み込みシンボリック変数を使用して、転送から Connect:Direct プロセスに情報を渡すことができます。

このタスクについて

この例では、組み込みシンボリック変数を使用して、Managed File Transfer 転送からユーザー定義 Connect:Direct プロセスに情報を渡します。 Managed File Transfer で使用する組み込みシンボリック変数の詳細については、ユーザー定義 Connect:Direct プロセスで使用する置換変数を参照してください。

この例では、Managed File Transfer Agent から Connect:Direct ブリッジ・ノードにファイルを転送します。転送の第 1 部分を Managed File Transfer が実行します。 転送の第 2 部分をユーザー定義 Connect:Direct プロセスが実行します。

手順

1. 組み込みシンボリック変数を使用する Connect:Direct プロセスを作成します。

```
%FTEPNAME PROCESS
    SNODE=%FTESNODE
    PNODEID=(%FTEPUSER,%FTEPPASS)
    SNODEID=(%FTESUSER,%FTESPASS)

COPY001 COPY
    FROM (
        FILE=%FTEFFILE
        DISP=%FTEFDISP
    )
    TO (
        FILE=%FTETFILE
        DISP=%FTETDISP
    )
PEND
```

2. このプロセスをテキスト・ファイルに保存します。場所: *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent*/Example.cdp
3. ConnectDirectProcessDefinition.xml ファイルを編集して、ステップ 1 で作成した Connect:Direct プロセスを呼び出すルールを組み込みます。

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
               xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://wmqfte.ibm.com/
                ConnectDirectProcessDefinitions ConnectDirectProcessDefinitions.xsd">


    <tns:processSet>
        <tns:condition>
            <tns:match variable="%FTESNODE" value="TOBERMORY" pattern="wildcard" />
        </tns:condition>
        <tns:process>
            <tns:transfer process="Example.cdp" />
        </tns:process>
    </tns:processSet>

</tns:cdprocess>
```

この例では、転送要求がソースまたは宛先 Connect:Direct ノードとして TOBERMORY を持つ Connect:Direct ブリッジ・エージェントに実行依頼されると、 `Example.cdp` Connect:Direct プロセスが呼び出されます。

4. ステップ 3 の `ConnectDirectProcessDefinition.xml` ファイルで定義した条件を満たすファイル転送要求を実行依頼します。

   例:

   ```
   fteCreateTransfer -sa ORINOCO -da CD_BRIDGE
                     -sm QM_WIMBLEDON -dm QM_COMMON
                     -de overwrite -df TOBERMORY:/home/bulgaria/destination.txt
                     -sd leave c:\bungo\source.txt
   ```

   この例では、宛先 Connect:Direct ノードが TOBERMORY になっています。 このノードは、転送の 2 次ノードであり、%FTESNODE の値が TOBERMORY に設定されています。 このコマンドは、`ConnectDirectProcessDefinition.xml` ファイルに設定されている条件に一致します。

5. Managed File Transfer が Connect:Direct ブリッジ・エージェントと同じシステムの一時的な場所にソース・ファイルを転送します。
6. Connect:Direct ブリッジ・エージェントが転送要求と構成情報に含まれている情報に基づいて組み込みシンボリック変数の値を設定します。

   組み込みシンボリック変数は、以下の値に設定されます。

   - %FTEPNAME=*process_name* - この値は、Connect:Direct ブリッジ・エージェントによって生成される 8 文字のプロセス名です。
   - %FTESNODE=TOBERMORY - この値は、**fteCreateTransfer** コマンドの **-df** パラメーターから設定されます。
   - %FTEPUSER,=プライマリー・ノード・ユーザー - この情報は、`ConnectDirectCredentials.xml` ファイルから取得されます。
   - %FTEPPASS=*primary_node_user_password* - この情報は `ConnectDirectCredentials.xml` ファイルから取得されます。
   - %FTESUSER,=セカンダリー・ノード・ユーザー - この情報は `ConnectDirectCredentials.xml` ファイルから取得されます。
   - %FTESPASS=*2* 次ノード・ユーザー・パスワード - この情報は、`ConnectDirectCredentials.xml` ファイルから取得されます。
   - %FTEFFILE =*temporary_location* - この値は、Connect:Direct ブリッジ・エージェントと同じシステムの一時的なファイル保管場所です。
   - %FTEFDISP=leave - この値は、**fteCreateTransfer** コマンドの **-sd** パラメーターから設定されます。
   - %FTETFILE=/home/bulgaria/destination.txt - この値は、**fteCreateTransfer** コマンドの **-df** パラメーターから設定されます。

- %FTETDISP=overwrite - この値は、**fteCreateTransfer** コマンドの **-de** パラメーターから設定されます。

7. Connect:Direct ブリッジ・ノードで Connect:Direct プロセスが開始されます。 Connect:Direct は、Connect:Direct ブリッジ・システム上の一時ロケーションから、Connect:Direct ノード TOBERMORY が実行されているシステム上の宛先 `/home/bulgaria/destination.txt` にファイルを転送します。

関連概念

321 ページの『ファイル転送要求からのユーザー定義 Connect:Direct プロセスの送信』
ファイル転送の一部としてユーザー定義 Connect:Direct プロセスを呼び出す Connect:Direct ブリッジ・エージェントを経由する転送の転送要求を送信できます。

関連資料

ユーザー定義 Connect:Direct プロセスで使用する置換変数

Connect:Direct プロセスを使用して Managed File Transfer 転送要求を送信する操作

Connect:Direct プロセスから Connect:Direct ブリッジ・エージェントに転送要求を送信できます。 Managed File Transfer には、Connect:Direct プロセスの **RUN TASK** ステートメントから呼び出すことができるコマンドが用意されています。

Managed File Transfer には、Connect:Direct プロセスで使用できる以下のコマンドが用意されています。

ftetag
ftebxfer、**ftecxfer** の各コマンドの前のステップでこのコマンドを指定して、転送に関する必要な監査情報を作成します。 このコマンドでは、転送のソース指定をパラメーターとして使用します。 ソース仕様の形式については、 **fteCreateTransfer**: 新規ファイル転送の開始を参照してください。

ftebxfer
転送要求の実行依頼先のキュー・マネージャーが、コマンドを実行依頼する Connect:Direct ノードと同じシステム上にある場合に、このコマンドを指定してファイル転送要求を作成します。 このコマンドでは、**fteCreateTransfer** コマンドと同じパラメーターを使用します。 これらのパラメーターについては、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。 このコマンドには、以下の追加パラメーターもあります。

-qmgrname
必須。 コマンドの送信先のキュー・マネージャーの名前。

ftecxfer
転送要求の送信先のキュー・マネージャーが、コマンドを送信する Connect:Direct ノードとは別のシステムに存在する場合は、このコマンドを指定して、ファイル転送要求を作成します。 このコマンドでは、**fteCreateTransfer** コマンドと同じパラメーターを使用します。 パラメーターについては、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。 このコマンドには、さらに追加のパラメーターが 3 つあります。

-qmgrname
必須。 コマンドの送信先のキュー・マネージャーの名前。

-connname
必須。 コマンドの送信先のキュー・マネージャーのホストとポート。IBM MQ の CONNAME の形式で指定します。 例えば、 `host.example.com(1337)`. などです。

-channelname
オプション。 コマンドの送信先のキュー・マネージャーに接続するときに使用するチャネルの名前。 指定しない場合は、デフォルト値の SYSTEM.DEF.SVRCONN が使用されます。

関連タスク

326 ページの『Connect:Direct Requester を使用して、Managed File Transfer を呼び出す Connect:Direct プロセスを作成して送信する操作』
Connect:Direct 要求者は、 Managed File Transfer を呼び出す Connect:Direct プロセスを作成および実行依頼するために使用できるグラフィカル・ユーザー・インターフェースです。

関連資料
例: MFT コマンドを呼び出す Connect:Direct プロセス・ファイル

Connect:Direct Requester を使用して、*Managed File Transfer* を呼び出す *Connect:Direct* プロセスを作成して送信する操作

Connect:Direct 要求者は、 Managed File Transfer を呼び出す Connect:Direct プロセスを作成および実行依頼するために使用できるグラフィカル・ユーザー・インターフェースです。

このタスクについて

このタスクでは、 Managed File Transfer **ftecxfer** コマンドまたは **ftebxfer** コマンドを呼び出す Connect:Direct プロセスを作成する方法について説明します。 転送要求の実行依頼先のキュー・マネージャーが、コマンドを実行依頼する Connect:Direct ノードとは別のシステムにある場合は、 **ftecxfer** コマンドを使用します。 転送要求の実行依頼先のキュー・マネージャーが、コマンドを実行依頼する Connect:Direct ノードと同じシステム上にある場合は、 **ftebxfer** コマンドを使用します。 **ftecxfer** コマンドは、転送のソース・エージェントのエージェント・キュー・マネージャーに対するクライアント接続を確立します。 **ftecxfer** コマンドを呼び出す前に、**ftetag** コマンドを呼び出して、ソースの指定情報を渡す必要があります。 このようにすれば、Managed File Transfer から開始した転送の場合と同じ要領で、プロセスのログを記録して監査することが可能になります。

手順

1. Connect:Direct Requester を開始します。
2. パネルの「**ノード**」タブで、プロセスの 1 次ノードとして使用する Connect:Direct ノードを選択します。
3. 「**ファイル**」 > 「**新規**」 > 「**プロセス**」を選択します。 「**プロセス・プロパティー**」ウィンドウが開きます。
4. 「**名前:**」フィールドにプロセスの名前を入力します。
5. 「**Snode**」 > 「**名前:**」リストから 2 次ノードを選択します。
6. 「**Snode**」 > 「**オペレーティング・システム:**」リストから 2 次ノードのオペレーティング・システムを選択します。
7. オプション: このウィンドウで必要な情報をさらに入力します。
8. 「**OK**」をクリックします。 「**プロセス・プロパティー**」ウィンドウが閉じます。
9. Managed File Transfer の **ftetag** コマンドを実行するステートメントを作成します。
   a) 「**プロセス**」ウィンドウで **End** ステートメントを右クリックします。
   b) 「**挿入**」 > 「**タスクの実行**」を選択します。 「**タスク実行ステートメント**」ウィンドウが開きます。
   c) 「**ラベル:**」フィールドに Tag と入力します。
   d) 「**オプション・パラメーターまたはコマンド (Optional Parameters or Commands)**」 フィールドに、 pgm(*MQ_INSTALLATION_PATH*/bin/ftetag) args(*source_specification*)と入力します。 *source_specification* のフォーマットについて詳しくは、**fteCreateTransfer**: 新規ファイル転送の開始を参照してください。
   e) 「**OK**」をクリックします。 「**タスク実行ステートメント**」ウィンドウが閉じます。
10. Managed File Transfer の **ftecxfer** コマンドまたは **ftebxfer** コマンドを実行するステートメントを作成します。
    a) 「**プロセス**」ウィンドウで **End** ステートメントを右クリックします。
    b) 「**挿入**」 > 「**タスクの実行**」を選択します。 「**タスク実行ステートメント**」ウィンドウが開きます。
    c) 「**ラベル:**」フィールドに Transfer と入力します。
    d) 「**オプション・パラメーターまたはコマンド (Optional Parameters or Commands)**」 フィールドに、選択するコマンドに応じて pgm(*MQ_INSTALLATION_PATH*/bin/ftecxfer) args(*parameters*) または pgm(*MQ_INSTALLATION_PATH*/bin/ftebxfer) args(*parameters*) を入力します。 **ftecxfer** コマンドおよび **ftebxfer** コマンドで使用する

パラメーターは、**fteCreateTransfer** コマンドで使用するパラメーターと同じですが、**ftecxfer** および **ftebxfer** に特定のパラメーターもいくつかあります。 詳しくは、**fteCreateTransfer**: 新規ファイル転送の開始および 325 ページの『Connect:Direct プロセスを使用して Managed File Transfer 転送要求を送信する操作』を参照してください。

e)「**OK**」をクリックします。 「**タスク実行ステートメント**」ウィンドウが閉じます。

11. オプション: 必要なステートメントをさらに作成します。
12. プロセスを送信します。
    a)「**プロセス**」ウィンドウで右クリックします。
    b)「**実行依頼**」を選択します。 「**Connect:Direct 接続**」ウィンドウが開きます。
    c) プロセスを実行するために使用するユーザー名とパスワードを入力します。
    d)「**OK**」をクリックします。

関連概念

325 ページの『Connect:Direct プロセスを使用して Managed File Transfer 転送要求を送信する操作』
Connect:Direct プロセスから Connect:Direct ブリッジ・エージェントに転送要求を送信できます。 Managed File Transfer には、Connect:Direct プロセスの **RUN TASK** ステートメントから呼び出すことができるコマンドが用意されています。

IBM Integration Bus からの MFT の操作

FTEOutput ノードと FTEInput ノードを使用して、IBM Integration Bus から Managed File Transfer を操作できます。

- FTEInput ノードを使用すると、Managed File Transfer を使用してネットワークでファイルを転送し、そのファイルを Integration Bus フローの一部として処理できます。
- FTEOutput ノードを使用すると、Integration Bus フローで出力されたファイルをネットワーク内の別の場所に転送できます。

ブローカー・エージェントとの間でファイルを転送するエージェントは、Managed File Transfer のどのレベルでも構いません。

詳しくは、IBM Integration Bus 製品資料を参照してください。

MFT のリカバリーと再始動

エージェントまたはキュー・マネージャーが何らかの理由 (例えば、電源やネットワークの障害など) で使用できない場合、Managed File Transfer は、以下のシナリオで示すようにリカバリーを行います。

- 通常、ファイルの転送中に問題が発生すると、Managed File Transfer は、問題が修復された後にそのファイル転送をリカバリーおよび再開します。
- エージェントまたはキュー・マネージャーが使用できなくなっている間に、転送処理中のファイルが削除または変更されると、転送は失敗し、その失敗に関する詳細を示すメッセージが転送ログに記録されます。
- ファイル転送中にエージェント・プロセスが失敗しても、エージェントを再始動すると、転送が続行されます。
- エージェントがエージェント・キュー・マネージャーへの接続を失うと、エージェントはキュー・マネージャーへの再接続を試行する間、待機状態になります。 エージェントがキュー・マネージャーに正常に再接続すると、 現在の転送を続行します。
- エージェントが何らかの理由で停止した場合、エージェントに関連付けられているリソース・モニターはすべてポーリングを停止します。 エージェントがリカバリーすると、モニターも再始動されて、リソースのポーリングも再開します。
- ソースのファイル属性指定が `delete` に設定されたファイル転送の場合、ソース・エージェントから宛先エージェントにすべてのデータが送信された後にリカバリーが発生すると、ソース・ファイルは削除の前にアンロックされます。 このアンロックの影響で、ソース・ファイルが削除される前にファイルが変

更される可能性があります。 したがって、ソース・ファイルの削除は安全ではないと見なされるため、次の警告が表示されます。

```
BFGTR0075W: The source file has not been deleted because it is possible that the source file
was modified after the source file was transferred.
```

この場合は、ソース・ファイルの内容が変更されていないことを確認してから、手動でソース・ファイルを削除してください。

転送の状況については、IBM MQ Explorer で確認することができます。 いずれかの転送が Stalled として表示される場合、停止状況はエージェントの問題、または転送に関与する 2 つのエージェントの間の問題のいずれかを示すため、修正アクションを実行する必要がある場合があります。

関連タスク

328 ページの『停止した転送のリカバリーに対するタイムアウトの設定』
停止したファイル転送について、ソース・エージェントのすべての転送に適用される転送リカバリー・タイムアウトを設定できます。 また、転送ごとに転送リカバリー・タイムアウトを設定することもできます。停止したファイル転送のリカバリーをソース・エージェントで試行し続ける特定の期間を秒単位で設定した場合、転送が成功しないままエージェントがタイムアウトに達すると、転送は失敗します。

停止した転送のリカバリーに対するタイムアウトの設定

停止したファイル転送について、ソース・エージェントのすべての転送に適用される転送リカバリー・タイムアウトを設定できます。 また、転送ごとに転送リカバリー・タイムアウトを設定することもできます。停止したファイル転送のリカバリーをソース・エージェントで試行し続ける特定の期間を秒単位で設定した場合、転送が成功しないままエージェントがタイムアウトに達すると、転送は失敗します。

このタスクについて

エージェントの agent.properties ファイルに転送リカバリー・タイムアウト・パラメーターを追加することで、ソース・エージェントのすべての転送に適用される転送リカバリー・タイムアウトを設定できます。 また、コマンド行、IBM MQ Explorer、または Apache Ant タスクを使用して、転送ごとに転送リカバリー・タイムアウトを設定することもできます。 agent.properties ファイルに転送リカバリー・タイムアウト値が設定されている場合に、転送ごとに転送リカバリー・タイムアウトを設定すると、agent.properties ファイルの値がオーバーライドされます。

転送リカバリー・タイムアウトには、次の 3 つのオプションがあります。

- エージェントは、停止した転送が成功するまでリカバリーを試行し続ける。 これは、転送リカバリー・タイムアウトが設定されていない場合のエージェントのデフォルトの動作と同じです。
- エージェントは、リカバリーに入るとすぐに転送に失敗のマークを付ける。
- エージェントは、指定された期間、停止した転送の再試行を続けた後に、転送に失敗のマークを付ける。

ファイル転送リカバリー・タイムアウトの設定はオプションです。 これを設定しない場合、転送はデフォルトの動作に従います。この動作では、停止した転送が成功するまで、エージェントはその転送のリカバリーを試行し続けます。

関連概念

327 ページの『MFT のリカバリーと再始動』
エージェントまたはキュー・マネージャーが何らかの理由 (例えば、電源やネットワークの障害など) で使用できない場合、Managed File Transfer は、以下のシナリオで示すようにリカバリーを行います。

転送リカバリー・タイムアウトの概念

停止したファイル転送のリカバリーをソース・エージェントが試行し続ける時間を秒単位で設定できます。転送が成功しないままエージェントが再試行間隔のタイムアウトに達した場合、その転送は失敗します。

リカバリー・タイムアウトの優先順位

fteCreateTransfer、**fteCreateTemplate**、または **fteCreateMonitor** コマンドを使用して、または IBM MQ Explorer を使用して、あるいは **fte:filespec** ネスト・エレメントで指定された個々の転送の転送リカバリー・タイムアウト値は、ソース・エージェントの agent.properties ファイル内の **transferRecoveryTimeout** パラメーターに指定された値よりも優先されます。

例えば、**fteCreateTransfer** コマンドを **-rt** パラメーターと値のペアを指定せずに開始すると、ソース・エージェント AGENT1 は、リカバリー・タイムアウトの動作を決定するために、agent.properties ファイルで **transferRecoveryTimeout** 値を探します。

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

agent.properties ファイルの **transferRecoveryTimeout** パラメーターが設定されていないか、-1 に設定されている場合、エージェントはデフォルトの動作に従い、転送が成功するまでリカバリーを試行し続けます。

一方、**fteCreateTransfer** コマンドに **-rt** パラメーターを指定した場合は、そのパラメーターの値が agent.properties ファイルの値よりも優先され、転送のリカバリー・タイムアウト設定として使用されます。

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt 21600 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

リカバリー・タイムアウト・カウンター

転送がリカバリー状態に入ると、リカバリー・タイムアウト・カウンターが開始されます。 転送状況がリカバリーに変わったことと、その状況が変わったソース・エージェントのクロック時刻を示す転送ログ・メッセージが、Log/agent_name/transfer_ID というトピック・ストリングで SYSTEM.FTE トピックにパブリッシュされます。 設定された再試行間隔内で転送が再開されて、リカバリー・タイムアウトに達しなかった場合 (カウンター <= リカバリー・タイムアウト)、転送がリカバリーに入ったときに再び開始できるようにカウンターは 0 にリセットされます。

リカバリー・タイムアウトとして設定された最大値にカウンターが達すると (カウンター == リカバリー・タイムアウト)、転送のリカバリーは停止し、ソース・エージェントはその転送を失敗として報告します。転送がリカバリー・タイムアウトに達したために発生するこの種の転送失敗は、メッセージ・コード RECOVERY TIMEOUT (69) で示されます。 転送が失敗したことを示す別の転送ログ・メッセージが、Log/agent_name/transfer_ID というトピック・ストリングで SYSTEM.FTE トピックにパブリッシュされます。この転送ログ・メッセージには、メッセージ、戻りコード、およびソース・エージェントのイベント・ログが含まれます。 リカバリー中に以下のいずれかのイベントが発生すると、ソース・エージェントのイベント・ログがメッセージで更新されます。

- -1 よりも大きい値がリカバリー・タイムアウト・パラメーターに設定されると、転送はリカバリーに入ります。 エージェントのイベント・ログが更新されて、**TransferId** のリカバリー・タイマーの開始と、ソース・エージェントがリカバリー・タイムアウト処理を開始するまで待機する時間が示されます。
- リカバリー中の転送が再開されると、ソース・エージェントのイベント・ログが新しいメッセージで更新されて、リカバリー状態だった **TransferId** が再開されたことが示されます。
- リカバリー中の転送がタイムアウトになると、ソース・エージェントのイベント・ログが更新されて、リカバリー中にリカバリー・タイムアウトのために失敗した **TransferId** が示されます。

これらのログ・メッセージから、ユーザー (サブスクライバーおよびロガー) は、転送リカバリー・タイムアウトのために失敗した転送を特定できます。

リカバリー・タイムアウトのカウンターは、常にソース・エージェント側にあります。 ただし、ソース・エージェントからの情報を宛先エージェントがタイムリーに受信できなかった場合、宛先エージェントは転送のリカバリーを開始するように求める要求をソース・エージェントに送信できます。 リカバリー・タイムアウト・オプションが設定された転送の場合、ソース・エージェントは宛先エージェントからこの要求を受信すると、リカバリー・タイムアウト・カウンターを開始します。

リカバリー・タイムアウト・オプションを使用しない転送、失敗した転送、および部分的に完了した転送については、引き続き手動処理が必要になります。

複数のファイルに対して単一の転送要求が発行される転送セットの場合、正常に完了したファイルが複数あっても、部分的にしか完了しなかったファイルが 1 つあると、その転送には失敗のマークが付けられます。予期されるとおりには完了しなかったからです。 部分的に完了したファイルを転送中にソース・エージェントがタイムアウトになった可能性があります。

宛先エージェントとファイル・サーバーが作動可能でファイル転送を受け入れる状態にあることを確認してください。

セット全体の転送要求を再発行する必要があります。ただし、最初に転送を試行したときのファイルの一部が宛先に残っているために起きる問題を避けるために、「既存の場合は上書き」オプションを指定して新しい要求を発行してください。 これにより、宛先にファイルを再度書き込む前に、前回の転送試行で生じた不完全なファイル・セットを、新しい転送の一環としてクリーンアップします。

IBM MQ 9.1.5 以降、最初に転送の試行に失敗した後に宛先に残る部分ファイルを手動で削除する必要はなくなりました。 転送に転送リカバリー・タイムアウトが設定されている場合に、転送リカバリーがタイムアウトすると、ソース・エージェントが、その転送を RecoveryTimedOut 状態に移行します。 転送が再同期されると、宛先エージェントが、転送中に作成された部分ファイルを削除し、ソース・エージェントに完了メッセージを送信します。

トレースとメッセージ

診断の目的でトレース・ポイントが組み込まれています。 リカバリー・タイムアウト値、再試行間隔の開始、再開期間の開始とカウンターのリセット、および転送がタイムアウトになって失敗したのかどうかが、ログに記録されます。 問題や予期しない動作が発生した場合、IBM サポートから要求されたら、トラブルシューティングに役立つように、ソース・エージェントの出力ログとトレース・ファイルを収集して提供してください。

メッセージで通知される状況は次のとおりです。

- 転送のリカバリーが開始された (BFGTR0081I)
- リカバリーがタイムアウトになったために転送が終了された (BFGSS0081E)
- リカバリーの開始後に転送が再開された (BFGTR0082I)

関連概念

327 ページの『MFT のリカバリーと再始動』

エージェントまたはキュー・マネージャーが何らかの理由 (例えば、電源やネットワークの障害など) で使用できない場合、Managed File Transfer は、以下のシナリオで示すようにリカバリーを行います。

ソース・エージェントのすべての転送に対する転送リカバリー・タイムアウトの設定

transferRecoveryTimeout パラメーターを agent.properties ファイルに追加して、ソース・エージェントのすべての転送に適用される転送リカバリー・タイムアウトを設定できます。

このタスクについて

ソース・エージェントのすべての転送に適用される転送リカバリー・タイムアウトを設定するには、**transferRecoveryTimeout** パラメーターと値のペアを agent.properties ファイルに追加します。

transferRecoveryTimeout パラメーターには次の 3 つのオプションがあります。

-1
エージェントは、停止した転送のリカバリーを、転送が成功するまで試行し続けます。 このオプションを使用すると、このプロパティーを設定しない場合のエージェントのデフォルトの動作と同じになります。

0
エージェントは、リカバリーに入るとすぐにファイル転送を停止します。

>0
エージェントは、指定された正整数値で設定された時間 (秒単位) だけ、停止した転送のリカバリーを試行し続けます。

agent.properties ファイルに対して行った変更は、エージェントが再始動された後でのみ有効になります。

必要に応じて、個々の転送について、 agent.properties ファイル内の転送リカバリー・タイムアウト値をオーバーライドすることができます。 詳しくは、331 ページの『転送ごとの転送リカバリー・タイムアウトの設定』を参照してください。

手順

- 停止した転送が成功するまでリカバリーを試行し続けるようにエージェントに指定するには、以下の例に示すように、転送リカバリー・タイムアウト値 -1 を設定します。

```
transferRecoveryTimeout=-1
```

- リカバリーに入るとすぐに転送に失敗のマークを付けるようにエージェントに指定するには、以下の例に示すように、転送リカバリー・タイムアウト値を 0 に設定します。

```
transferRecoveryTimeout=0
```

- 指定された期間、停止した転送の再試行を続けた後に、転送に失敗のマークを付けるようにエージェントに指定するには、転送リカバリー・タイムアウト値として、エージェントが再試行を続ける期間を秒単位で設定します。

  例えば、転送リカバリー・タイムアウト値 21600 を設定すると、エージェントは、リカバリーを開始してから 6 時間にわたって転送のリカバリーを試行し続けます。

```
transferRecoveryTimeout=21600
```

  このパラメーターの最大値は 999999999 です。

転送ごとの転送リカバリー・タイムアウトの設定

コマンド行、IBM MQ Explorer、または Apache Ant タスクを使用して、転送ごとに転送リカバリー・タイムアウトを設定できます。 転送リカバリー・タイムアウト値が agent.properties ファイルに設定されている場合は、個々の転送の転送リカバリー・タイムアウトを設定すると、 agent.properties ファイルに設定されている値がオーバーライドされます。

このタスクについて

転送ごとに転送リカバリー・タイムアウト・パラメーターを設定できる状況は次のとおりです。

- **fteCreateTransfer** コマンド、または IBM MQ Explorer を使用して転送を作成するとき。
- **fteCreateTemplate** コマンド、または IBM MQ Explorer を使用して転送テンプレートを作成するとき。
- **fteCreateMonitor** コマンド、または IBM MQ Explorer を使用してリソース・モニターを作成するとき。
- fte:filecopy や fte:filemoveAnt タスクによるファイルのコピー

転送リカバリー・タイムアウト値を個別の転送に設定すると、この値は agent.properties ファイルに設定されている転送リカバリー・タイムアウト値をオーバーライドします (330 ページの『ソース・エージェントのすべての転送に対する転送リカバリー・タイムアウトの設定』を参照) 。

手順

- **fteCreateTransfer** コマンドまたは **fteCreateTemplate** コマンドを使用して転送リカバリー・タイムアウトを設定するには、 次のように **-rt** パラメーターに適切なオプションを指定します。

-1
エージェントは、停止した転送のリカバリーを、転送が成功するまで試行し続けます。 このオプションを使用すると、このプロパティーを設定しない場合のエージェントのデフォルトの動作と同じになります。

0
エージェントは、リカバリーに入るとすぐにファイル転送を停止します。

>0
エージェントは、指定された時間 (秒単位) だけ、停止した転送の復旧を試行し続けます。

fteCreateTransfer コマンドの例

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt -1 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt 0 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt 21600 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

fteCreateTemplate コマンドの例

```
fteCreateTemplate -tn "payroll accounts monthly report template" -rt -1 -sa PAYROLL -sm
QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

```
fteCreateTemplate -tn "payroll accounts monthly report template" -rt 0 -sa PAYROLL -sm
QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

```
fteCreateTemplate -tn "payroll accounts monthly report template" -rt 21600 -sa PAYROLL -sm
QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

fteCreateMonitor コマンドには **-rt** パラメーターはありません。 **fteCreateTransfer** コマンドに **-rt** パラメーターを設定すると同時に **-gt** パラメーターも設定すると、**fteCreateTransfer** コマンドの実行時に生成される転送定義の XML 文書に、リカバリー・タイムアウト・パラメーターが含められます。 その後、**fteCreateMonitor** コマンドを実行するときに、その XML 文書をリソース・モニターで使用します。 以下の例では、転送リカバリー・タイムアウトの詳細が task.xml ファイルに含まれています。

```
fteCreateMonitor -ma AgentName -md C:\mqmft\monitors -mn Monitor_Name -mt task.xml -tr
"fileSize>=5MB,*.zip"
```

- IBM MQ Explorer の「新規の転送」、「新規モニター」、または「テンプレートの新規作成」のウィザード・ページを使用して転送リカバリー・タイムアウトを設定するには、「**転送のリカバリーのタイムアウト**」(秒) フィールドで必要なオプションを選択します。

  ソース・エージェントで
  「**ソース・エージェントで**」を選択すると、agent.properties ファイルの **transferRecoveryTimeout** パラメーター値が設定されている場合にはその値が使用されます。 設定されていない場合には、転送リカバリー・タイムアウトのデフォルトの動作が適用されます。

  数値リスト・ボックス
  数値リスト・ボックスに秒単位で時間を入力すると、エージェントはその指定された時間、停止した転送のリカバリーを試行し続けます。

  なし
  「**なし**」を選択すると、転送リカバリー・タイムアウトは設定されず、停止した転送が成功するまでエージェントはリカバリーを試行し続けます。

- Ant タスクを使用してリカバリー・タイムアウトを設定します。 **transferRecoveryTimeout** オプションと値を、ファイルを移動またはコピーするための **fte:filecopy** エレメントまたは **fte:filemove** エレメントと共に含めます。以下に例を示します。

  fte:filecopy の例

  ```
  <fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
                src="agent1@qm1" dst="agent2@qm2"
                rcproperty="copy.result" transferRecoveryTimeout="0">

      <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

  </fte:filecopy>
  ```

  fte:filemove の例

  ```
  <fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
                src=agent1@qm1 dst="agent2@qm2"
                rcproperty="move.result" transferRecoveryTimeout="21600">

      <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

  </fte:filemove>
  ```

関連概念

327 ページの『MFT のリカバリーと再始動』
エージェントまたはキュー・マネージャーが何らかの理由 (例えば、電源やネットワークの障害など) で使用できない場合、Managed File Transfer は、以下のシナリオで示すようにリカバリーを行います。

関連タスク

216 ページの『新規ファイル転送の開始』
新規ファイル転送は、IBM MQ Explorer またはコマンド行から開始でき、単一ファイルまたは複数ファイルのグループのいずれかの転送を選択できます。

259 ページの『IBM MQ Explorer を使用したファイル転送テンプレートの作成』
ファイル転送テンプレートを IBM MQ Explorer またはコマンド行から作成することができます。 その後そのテンプレートを使用してそのテンプレート詳細を使用する新規ファイル転送を作成したり、そのテンプレートを送信してファイル転送を開始したりできます。

227 ページの『MFT リソースのモニター』
キューやディレクトリーなどの Managed File Transfer リソースをモニターできます。 そのリソースで条件が満たされると、リソース・モニターがファイル転送などのタスクを開始します。 IBM MQ Explorer 用 Managed File Transfer プラグインの **fteCreateMonitor** コマンドまたは 「**モニター**」 ビューを使用して、リソース・モニターを作成できます。

225 ページの『「転送ログ」のファイル転送の状況の表示』
IBM MQ Explorer の 「**転送ログ**」 を使用して、ファイル転送の詳細を表示できます。 対象にできるのは、コマンド行または IBM MQ Explorer のいずれかから開始された転送です。 また、「**転送ログ**」に表示される内容をカスタマイズすることもできます。

関連資料

MFT `agent.properties` ファイル

fteCreateTransfer: 新規ファイル転送の開始

fteCreateTemplate: 新規ファイル転送テンプレートの作成

fteCreateMonitor: MFT リソース・モニターの作成

fte:filecopy の Ant タスク

fte:filemove の Ant タスク

Windows Linux AIX MQ Telemetry の管理

MQ Telemetry の管理は、IBM MQ Explorer またはコマンド行で行います。 エクスプローラーを使用して、テレメトリー・チャネルの構成、テレメトリー・サービスの制御、および IBM MQ に接続されている MQTT

クライアントのモニターを行います。 MQ Telemetry のセキュリティーの構成には、JAAS、TLS、および IBM MQ オブジェクト権限マネージャーを使用します。

IBM MQ Explorer を使用した管理

エクスプローラーを使用して、テレメトリー・チャネルの構成、テレメトリー・サービスの制御、および IBM MQ に接続されている MQTT クライアントのモニターを行います。 MQ Telemetry のセキュリティーの構成には、JAAS、TLS、および IBM MQ オブジェクト権限マネージャーを使用します。

コマンド行を使用した管理

MQ Telemetry は、コマンド行で MQSC コマンドを使用して完全に管理できます。

MQ Telemetry の資料では、IBM MQ Telemetry Transport v3 クライアント・アプリケーションの基本的な使い方を示すサンプル・スクリプトも提供されています。

サンプルを使用する前に、IBM MQ Telemetry Transport のサンプル・プログラムに記載されている例を読み、理解してください。

関連概念

MQ Telemetry

関連資料

MQXR プロパティー

Linux AIX Linux および AIX でテレメトリーを行うためのキュー・マネージャーの構成

MQ Telemetry を手動で構成するには、以下の手順を実行します。 ゲスト・ユーザー ID を使用する単純な構成のみが必要な場合は、代わりに IBM MQ Explorer で MQ Telemetry サポート・ウィザードを実行できます。

始める前に

単純な構成のみが必要な場合は、IBM MQ Explorer で MQ Telemetry サポートを使用することを検討してください。 このサポートには、ウィザードとサンプル・コマンド・プロシージャー `sampleMQM` が含まれています。 これらのリソースは、ゲスト・ユーザー ID を使用して初期構成をセットアップします。 IBM MQ Explorer を使用した MQ Telemetry のインストールの検査 および IBM MQ Telemetry Transport サンプル・プログラムを参照してください。

別の認証方式を使用する、より複雑な構成が必要な場合は、このタスクのステップを使用します。 以下の初期ステップから開始します。

1. IBM MQ および MQ Telemetry フィーチャーのインストール方法については、「MQ Telemetry のインストールに関する注意点」を参照してください。
2. キュー・マネージャーを作成して開始します。 このタスクでは、キュー・マネージャーを *qMgr* で表します。
3. このタスクの一部として、テレメトリー (MQXR) サービスを構成します。 MQXR プロパティー設定は、プラットフォーム固有のプロパティー・ファイル `mqxr_win.properties` に保管されます。 通常、MQXR プロパティー・ファイルを直接編集する必要はありません。ほとんどすべての設定は、MQSC admin コマンドまたは IBM MQ Explorer によって構成できるからです。 ファイルを直接編集する場合、変更を行う前にキュー・マネージャーを停止してください。 MQXR プロパティーを参照してください。

このタスクについて

さまざまな許可スキームを使用して MQ Telemetry を手動で構成するには、このタスクのステップに従います。

手順

1. テレメトリーのサンプル・ディレクトリーでコマンド・ウィンドウを開きます。

テレメトリー・サンプル・ディレクトリーは /opt/mqm/mqxr/samples です。

2. テレメトリー伝送キューを作成します。

   SYSTEM.MQTT.TRANSMIT.QUEUE が存在しない場合は、テレメトリー (MQXR) サービスが最初に開始されたときに自動的に作成され、ゲスト・ユーザー ID を使用するように設定されます。 ただし、このタスクでは、別の許可スキームを使用するように MQ Telemetry を構成します。 このタスクでは、SYSTEM.MQTT.TRANSMIT.QUEUE を作成し、それに対するアクセス権限を構成してから、テレメトリー (MQXR) サービスを開始します。

   以下のコマンドを実行します。

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc
   qMgr
   ```

3. デフォルト伝送キューを設定します。

   SYSTEM.MQTT.TRANSMIT.QUEUE がデフォルト伝送キューである場合は、 MQTT クライアントにメッセージを直接送信する方が簡単です。 それ以外の場合は、IBM MQ メッセージを受信するすべてのクライアントにリモート・キュー定義を追加する必要があります。 339 ページの『クライアントへのメッセージの直接送信』を参照してください。 デフォルト伝送キューを変更すると、既存の構成が妨げられる可能性があることに注意してください。

   テレメトリー (MQXR) サービスは、最初に開始されるときに、 SYSTEM.MQTT.TRANSMIT.QUEUE をキュー・マネージャーのデフォルト伝送キューとして設定しません。 この設定を構成するには、デフォルト伝送キュー・プロパティーを変更します。 これを行うには、IBM MQ Explorer を使用するか、以下のコマンドを実行します。

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

4. 342 ページの『MQTT クライアントによる IBM MQ オブジェクトへのアクセスの許可』の手順に従ってユーザー ID を 1 つ以上作成します。 このユーザー ID には、パブリッシュ、サブスクライブ、および MQTT クライアントへのパブリケーション送信の権限があります。

5. installMQXRService_unix.mqsc ファイルを編集して、MQTT TLS チャネルのパスフレーズを暗号化するために使用する鍵ファイルを構成します。

   a) *WMQ program installation directory*/mqxr/samples/installMQXRService_unix.mqsc ファイルを開きます。

   b) **STARTARG** パラメーターが含まれている行を見つけ、 **-sf** オプションを編集して、資格情報鍵ファイルの場所を指定します。

   デフォルトでは、installMQXRService_unix.mqsc ファイルは、[DEFAULT]という名前のデフォルトの鍵ファイルを使用します。 デフォルトの鍵ファイルはすべての IBM MQ インストール済み環境で同じであるため、パスフレーズを暗号化するときには、ご使用のインストール済み環境に固有の鍵ファイルを指定する必要があります。

   336 ページの『SYSTEM.MQXR.SERVICE の作成』のコード例も参照してください。

6. 以下のコマンドを実行して、テレメトリー (MQXR) サービスをインストールします。

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

7. サービスを開始します。

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   キュー・マネージャーを開始すると、テレメトリー (MQXR) サービスが自動的に開始されます。 このタスクでは、キュー・マネージャーが既に実行されているので、手動で開始します。

8. IBM MQ Explorer を使用して、MQTT クライアントからの接続を受け入れるようにテレメトリー・チャネルを構成します。

テレメトリー・チャネルは、それらの ID がステップ 335 ページの『4』 で定義したユーザー ID の 1 つになるように構成する必要があります。

DEFINE CHANNEL (MQTT) も参照してください。

9. サンプル・クライアントを実行して、構成を検査します。

   サンプル・クライアントがテレメトリー・チャネルと連動するためには、クライアントがパブリッシュ、サブスクライブ、およびパブリケーション受信を行うことをチャネルが許可しなければなりません。 サンプル・クライアントは、デフォルトではポート 1883 でテレメトリー・チャネルに接続します。 IBM MQ Telemetry Transport サンプル・プログラムも参照してください。

SYSTEM.MQXR.SERVICE の作成

runMQXRService コマンドを使用して、 SYSTEM.MQXR.SERVICE を作成します。

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+" -sf "/home/keyFileLocation/
keyFile.txt"') +
STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')
```

Windows Windows 上のテレメトリー用キュー・マネージャーの構成

MQ Telemetry を手動で構成するには、以下の手順を実行します。 ゲスト・ユーザー ID を使用する単純な構成のみが必要な場合は、代わりに IBM MQ Explorer で MQ Telemetry サポート・ウィザードを実行できます。

始める前に

単純な構成のみが必要な場合は、IBM MQ Explorer で MQ Telemetry サポートを使用することを検討してください。 このサポートには、ウィザードとサンプル・コマンド・プロシージャー sampleMQM が含まれています。 これらのリソースは、ゲスト・ユーザー ID を使用して初期構成をセットアップします。 IBM MQ Explorer を使用した MQ Telemetry のインストールの検査 および IBM MQ Telemetry Transport サンプル・プログラムを参照してください。

別の認証方式を使用する、より複雑な構成が必要な場合は、このタスクのステップを使用します。 以下の初期ステップから開始します。

1. IBM MQ および MQ Telemetry フィーチャーのインストール方法については、「MQ Telemetry のインストールに関する注意点」を参照してください。
2. キュー・マネージャーを作成して開始します。 このタスクでは、キュー・マネージャーを *qMgr* で表します。
3. このタスクの一部として、テレメトリー (MQXR) サービスを構成します。 MQXR プロパティー設定は、プラットフォーム固有のプロパティー・ファイル mqxr_win.properties に保管されます。 通常、MQXR プロパティー・ファイルを直接編集する必要はありません。ほとんどすべての設定は、MQSC admin コマンドまたは IBM MQ Explorer によって構成できるからです。 ファイルを直接編集する場合、変更を行う前にキュー・マネージャーを停止してください。 MQXR プロパティーを参照してください。

このタスクについて

さまざまな許可スキームを使用して MQ Telemetry を手動で構成するには、このタスクのステップに従います。

手順

1. テレメトリーのサンプル・ディレクトリーでコマンド・ウィンドウを開きます。

   テレメトリー・サンプル・ディレクトリーは *WMQ program installation directory*\mqxr\samples です。

2. テレメトリー伝送キューを作成します。

   SYSTEM.MQTT.TRANSMIT.QUEUE が存在しない場合は、テレメトリー (MQXR) サービスが最初に開始されたときに自動的に作成され、ゲスト・ユーザー ID を使用するように設定されます。 ただし、このタスクでは、別の許可スキームを使用するように MQ Telemetry を構成します。 このタスクでは、SYSTEM.MQTT.TRANSMIT.QUEUE を作成し、それに対するアクセス権限を構成してから、テレメトリー (MQXR) サービスを開始します。

   以下のコマンドを実行します。

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

3. デフォルト伝送キューを設定します。

   SYSTEM.MQTT.TRANSMIT.QUEUE がデフォルト伝送キューである場合は、 MQTT クライアントにメッセージを直接送信する方が簡単です。 それ以外の場合は、IBM MQ メッセージを受信するすべてのクライアントにリモート・キュー定義を追加する必要があります。 339 ページの『クライアントへのメッセージの直接送信』を参照してください。 デフォルト伝送キューを変更すると、既存の構成が妨げられる可能性があることに注意してください。

   テレメトリー (MQXR) サービスは、最初に開始されるときに、 SYSTEM.MQTT.TRANSMIT.QUEUE をキュー・マネージャーのデフォルト伝送キューとして設定しません。 この設定を構成するには、デフォルト伝送キュー・プロパティーを変更します。 これを行うには、IBM MQ Explorer を使用するか、以下のコマンドを実行します。

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

4. 342 ページの『MQTT クライアントによる IBM MQ オブジェクトへのアクセスの許可』の手順に従ってユーザー ID を 1 つ以上作成します。 このユーザー ID には、パブリッシュ、サブスクライブ、および MQTT クライアントへのパブリケーション送信の権限があります。

5. installMQXRService_win.mqsc ファイルを編集して、MQTT TLS チャネルのパスフレーズを暗号化するために使用する鍵ファイルを構成します。

   a) *WMQ program installation directory*\mqxr\samples\installMQXRService_win.mqsc ファイルを開きます。

   b) **STARTARG** パラメーターが含まれている行を見つけ、 **-sf** オプションを編集して、資格情報鍵ファイルの場所を指定します。

   デフォルトでは、 installMQXRService_win.mqsc ファイルは、 [DEFAULT]という名前のデフォルトの鍵ファイルを使用します。 デフォルトの鍵ファイルはすべての IBM MQ インストール済み環境で同じであるため、パスフレーズを暗号化するときには、ご使用のインストール済み環境に固有の鍵ファイルを指定する必要があります。

   338 ページの『Creating SYSTEM.MQXR.SERVICE』のコード例も参照してください。

6. 以下のコマンドを実行して、テレメトリー (MQXR) サービスをインストールします。

   ```
   type installMQXRService_win.mqsc | runmqsc qMgr
   ```

7. サービスを開始します。

   ```
   echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
   ```

   キュー・マネージャーを開始すると、テレメトリー (MQXR) サービスが自動的に開始されます。 このタスクでは、キュー・マネージャーが既に実行されているので、手動で開始します。

8. IBM MQ Explorer を使用して、MQTT クライアントからの接続を受け入れるようにテレメトリー・チャネルを構成します。

   テレメトリー・チャネルは、それらの ID がステップ 337 ページの『4』 で定義したユーザー ID の 1 つになるように構成する必要があります。

   DEFINE CHANNEL (MQTT) も参照してください。
9. サンプル・クライアントを実行して、構成を検査します。

   サンプル・クライアントがテレメトリー・チャネルと連動するためには、クライアントがパブリッシュ、サブスクライブ、およびパブリケーション受信を行うことをチャネルが許可しなければなりません。 サンプル・クライアントは、デフォルトではポート 1883 でテレメトリー・チャネルに接続します。 IBM MQ Telemetry Transport サンプル・プログラムも参照してください。

Creating SYSTEM.MQXR.SERVICE

runMQXRService コマンドを使用して、 SYSTEM.MQXR.SERVICE を作成します。

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+\." -g "+MQ_DATA_PATH+\." -sf
"c:\keyFileLocation\keyFile.txt"') +
STOPCMD('+MQ_INSTALL_PATH+\mqxr\bin\endMQXRService.bat') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+\mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+\mqxr.stderr')
```

Windows Linux AIX メッセージを MQTT クライアントに送信するように分散キューイングを構成

IBM MQ アプリケーションは、クライアントによって作成されたサブスクリプションにパブリッシュするか、またはメッセージを直接送信することによって、 MQTT v3 クライアント・メッセージを送信できます。 いずれの方法を使用する場合でも、メッセージは SYSTEM.MQTT.TRANSMIT.QUEUE に配置され、テレメトリー (MQXR) サービスによってクライアントに送信されます。 SYSTEM.MQTT.TRANSMIT.QUEUE にメッセージを配置する方法はいくつかあります。

MQTT クライアントのサブスクリプションへの応答としてのメッセージのパブリッシュ

テレメトリー (MQXR) サービスは、MQTT クライアントのために、サブスクリプションを作成します。 このクライアントは、クライアントによって送信されたサブスクリプションに一致するパブリケーションの宛先です。 テレメトリー・サービスは、一致したパブリケーションをクライアントに転送します。

MQTT クライアントは、キュー・マネージャーとして IBM MQ に接続され、そのキュー・マネージャー名が ClientIdentifier に設定されます。 クライアントに送信されるパブリケーションの宛先は、伝送キュー SYSTEM.MQTT.TRANSMIT.QUEUE です。 テレメトリー・サービスは、ターゲット・キュー・マネージャー名を特定のクライアントへのキーとして使用して、 SYSTEM.MQTT.TRANSMIT.QUEUE 上のメッセージを MQTT クライアントに転送します。

テレメトリー (MQXR) サービスは、ClientIdentifier をキュー・マネージャー名として使用して、伝送キューを開きます。 テレメトリー (MQXR) サービスは、キューのオブジェクト・ハンドルを MQSUB 呼び出しに渡して、クライアント・サブスクリプションに一致するパブリケーションを転送します。 オブジェクト名の解決では、リモート・キュー・マネージャー名として ClientIdentifier が作成され、伝送キューは SYSTEM.MQTT.TRANSMIT.QUEUE に解決されます。 標準の IBM MQ オブジェクト名解決を使用して、*ClientIdentifier* は次のように解決されます。339 ページの表 16 を参照してください。

1. *ClientIdentifier* がいずれにも一致しない場合。

   ClientIdentifier はリモート・キュー・マネージャー名です。 ローカルのキュー・マネージャー、キュー・マネージャー別名、または伝送キュー名とは一致しません。

キュー名が定義されていません。 現在、テレメトリー (MQXR) サービスによって SYSTEM.MQTT.PUBLICATION.QUEUE がキューの名前として設定されています。 MQTT v3 クライアントではキューはサポートされないため、解決されたキュー名はそのクライアントでは無視されます。
ローカル・キュー・マネージャー・プロパティー、デフォルト伝送キューの名前を SYSTEM.MQTT.TRANSMIT.QUEUE に設定することで、パブリケーションが SYSTEM.MQTT.TRANSMIT.QUEUE に配置され、クライアントに送信されるようにする必要があります。

2. *ClientIdentifier* が、*ClientIdentifier* というキュー・マネージャー別名と一致する場合。

   ClientIdentifier はリモート・キュー・マネージャー名です。 キュー・マネージャー別名の名前と一致します。
   キュー・マネージャー別名は、リモート・キュー・マネージャー名として *ClientIdentifier* を使用して定義されている必要があります。
   キュー・マネージャー別名定義で伝送キュー名を設定すると、デフォルト伝送を SYSTEM.MQTT.TRANSMIT.QUEUE に設定する必要がなくなります。

表 *16. MQTT* キュー・マネージャー別名の名前解決

	入力		出力		
ClientIdentifier	**キュー・マネージャー名**	**キュー名**	**キュー・マネージャー名**	**キュー名**	**伝送キュー**
一致なし	*ClientIdentifier*	未定義	*ClientIdentifier*	未定義	デフォルト伝送キュー。 SYSTEM.MQTT.TRANSMIT.QUEUE
ClientIdentifier というキュー・マネージャー別名と一致	*ClientIdentifier*	未定義	*ClientIdentifier*	未定義	SYSTEM.MQTT.TRANSMIT.QUEUE

ネーム解決の詳細については、ネーム解決を参照してください。

あらゆる IBM MQ プログラムが、同じトピックにパブリッシュできます。 パブリケーションは、そのサブスクライバー (トピックをサブスクリプションしている MQTT v3 クライアントなど) に送信されます。

管理トピックが、属性 CLUSTER(*clusterName*) を使用してクラスターで作成されている場合、そのクラスター内のすべてのアプリケーションがクライアントにパブリッシュできます。次に例を示します。

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

注 : SYSTEM.MQTT.TRANSMIT.QUEUE にクラスター属性を指定しないでください。

MQTT クライアントのサブスクライバーとパブリッシャーは、異なる複数のキュー・マネージャーに接続できます。 これらのサブスクライバーとパブリッシャーは、同じクラスターの一部であるか、または、パブリッシュ/サブスクライブ階層で接続されている場合があります。 パブリケーションは、IBM MQ を使用してパブリッシャーからサブスクライバーに送達されます。

クライアントへのメッセージの直接送信

クライアントがサブスクリプションを作成してサブスクリプション・トピックに一致するパブリケーションを受け取る代わりに、MQTT v3 クライアントにメッセージを直接送信します。 MQTT V3 クライアント・アプリケーションはメッセージを直接送信することができませんが、IBM MQ アプリケーションなどの他のアプリケーションは直接送信できます。

IBM MQ アプリケーションは、 MQTT v3 クライアントの ClientIdentifier を認識している必要があります。 MQTT v3 クライアントにはキューがないので、ターゲット・キュー名はトピック名として MQTT v3 アプリケーション・クライアントの messageArrived メソッドに渡されます。 例えば、MQI プログラムでは、クライアントに関するオブジェクト記述子を ObjectQmgrName として次のように作成します。

```
MQOD.ObjectQmgrName = ClientIdentifier ;
MQOD.ObjectName = name ;
```

アプリケーションが JMS を使用して記述されている場合は、Point-to-Point 宛先を作成します。次に例を示します。

JM 3.0

```
jakarta.jms.Destination jmsDestination =
(jakarta.jms.Destination)jmsFactory.createQueue
("queue://ClientIdentifier/name");
```

JMS 2.0

```
javax.jms.Destination jmsDestination =
(javax.jms.Destination)jmsFactory.createQueue
("queue://ClientIdentifier/name");
```

非送信請求メッセージを MQTT クライアントに送信するには、リモート・キュー定義を使用します。 リモート・キュー・マネージャー名は、クライアントの ClientIdentifier に解決される必要があります。伝送キューは SYSTEM.MQTT.TRANSMIT.QUEUE に解決される必要があります。340 ページの表 17 を参照してください。 リモート・キュー名は任意の名前にすることができます。 クライアントは、それをトピック・ストリングとして受信します。

表 17. *MQTT* クライアント・リモート・キュー定義の名前解決

入力		出力		
キュー名	**キュー・マネージャー名**	**キュー名**	**キュー・マネージャー名**	**伝送キュー**
リモート・キュー定義の名前	ブランクまたはローカルのキュー・マネージャー名	トピック・ストリングとして使用されるリモート・キュー名	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

クライアントが接続されている場合、メッセージは MQTT クライアントに直接送信されます。このクライアントは messageArrived メソッドを呼び出します。 messageArrived メソッドを参照してください。

クライアントが持続セッションから切断した場合、メッセージは SYSTEM.MQTT.TRANSMIT.QUEUE に保管されます。 MQTT ステートレス・セッションおよびステートフル・セッションを参照してください。 クライアントがセッションに再接続すると、このメッセージがクライアントに転送されます。

非持続メッセージを送信する場合、そのメッセージは「最高 1 回」のサービスの品質、QoS=0 でクライアントに送信されます。 持続メッセージをクライアントに直接送信する場合、デフォルトでは、「正確に 1 回」のサービス品質、QoS=2 で送信されます。 持続メカニズムがないクライアントの場合は、直接送信されてくるメッセージのサービス品質を下げることができます。 クライアントに直接送信されてくるメッセージのサービス品質を下げるには、トピック DEFAULT.QoS へのサブスクリプションを行います。 クライアントがサポートできる最高のサービス品質を指定します。

Windows Linux AIX MQTT クライアントの識別、許可、および認証

テレメトリー (MQXR) サービスは、 MQTT チャネルを使用して、 MQTT クライアントの代わりに IBM MQ トピックをパブリッシュまたはサブスクライブします。 IBM MQ 管理者は、 IBM MQ 許可に使用される MQTT チャネル ID を構成します。 管理者はチャネルの共通 ID を定義すること、あるいはチャネルに接続したクライアントの Username または ClientIdentifier を使用することができます。

テレメトリー (MQXR) サービスは、クライアントが提供する Username を使用して、またはクライアント証明書を使用して、クライアントを認証できます。 Username は、クライアントが提供するパスワードを使用して認証されます。

要約すると、クライアント ID はクライアントの識別要素のセレクションです。 コンテキストに応じて、クライアントは ClientIdentifier、Username、管理者が作成した共通クライアント ID、またはクライアント証明書によって識別されます。 認証検査に使用されるクライアント ID は、許可に使用されるものと同じ ID である必要はありません。

MQTT クライアント・プログラムは、MQTT チャネルを使用してサーバーに送信される ユーザー名 と パスワード を設定します。 それらは接続の暗号化および認証に必要な TLS プロパティーも設定できます。 管理者は、MQTT チャネルを認証するかどうか、およびチャネルを認証する方法を決めます。

MQTT クライアントに IBM MQ オブジェクトにアクセスする権限を与えるには、クライアントの ClientIdentifier または Username を許可するか、または共通クライアント ID を許可します。 クライアントが IBM MQ に接続することを許可するには、Username を認証するか、クライアント証明書を使用します。 Username を認証するように JAAS を構成し、クライアント証明書を認証するように TLS を構成します。

Password をクライアントで設定する場合、VPN を使用して接続を暗号化するか、または TLS を使用するように MQTT チャネルを構成して、パスワードを秘密に保ちます。

クライアント証明書を管理することは困難です。 そのため、パスワード認証に関連したリスクが許容可能であれば、パスワード認証がクライアントの認証にしばしば使用されます。

クライアント証明書を管理および保管するためのセキュアな方法があれば、証明書の認証に依存することができます。 ただし、テレメトリーが使用されるタイプの環境で、証明書をセキュアに管理できることは稀です。 その代わりに、クライアント証明書を使用する装置の認証はサーバーでのクライアント・パスワードの認証によって補完されます。 クライアント証明書の使用はより複雑なものとなるので、機密性の高いアプリケーションに限定されます。 2 つの方式を使用する認証は、2 因子認証と呼ばれます。 一方の因子 (パスワードなど) を知っていると共に、他方の因子 (証明書など) を所持している必要があります。

chip-and-pin 装置などの機密性の高いアプリケーションでは、内部のハードウェアおよびソフトウェアが改ざんされないように、製造の際に装置がロックされます。 信頼できる、時間の限定されたクライアント証明書が装置にコピーされます。 装置は使用される場所にデプロイされます。 装置が使用されるたびに、パスワードまたはスマート・カードからの別の証明書を使用して追加の認証が行われます。

Windows Linux AIX MQTT クライアントの ID および許可

クライアント ID、Username、または共通クライアント ID を使用して、IBM MQ オブジェクトにアクセスする許可を得ます。

IBM MQ 管理者は、3 つの選択肢から MQTT チャネルの ID を選択できます。 管理者は、クライアントが使用する MQTT チャネルを定義または変更するときに選択を行います。 ID は、IBM MQ トピックへのアクセスを許可するために使用されます。 以下の順に選択されます。

1. クライアント ID (USECLNTID 参照)。
2. 管理者がチャネルに提供する ID。(チャネルの MCAUSER。 MCAUSER を参照してください。)
3. 上記のいずれの選択も該当しない場合、 MQTT クライアントから渡される Username (Username は MqttConnectOptions クラスの属性です。 これはクライアントがサービスに接続する前に設定する必要があります。 そのデフォルト値は NULL です)。

問題の回避 : このプロセスで選択された ID は、その後、DISPLAY CHSTATUS (MQTT) コマンドなどで、クライアントの MCAUSER として参照されます。 必ずしも選択肢 (2) で言及されているチャネルの MCAUSER と同じ ID でなければならないわけではないことに注意してください。

IBM MQ の **setmqaut** コマンドを使用して、MQTT チャネルに関連付けられた ID が使用を許可されるオブジェクトおよびアクションを選択します。 例えば、以下のコードは、キュー・マネージャー QM1 の管理者によって提供されたチャネル ID MQTTClient を許可します。

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

Windows Linux AIX *MQTT クライアントによる IBM MQ オブジェクトへのアクセスの許可*

IBM MQ オブジェクトにパブリッシュおよびサブスクライブする権限を MQTT クライアントに与えるには、以下の手順を実行します。 これらの手順は、4 つの代替アクセス制御パターンに従っています。

始める前に

MQTT クライアントは、テレメトリー・チャネルに接続するときに ID を割り当てられることによって、IBM MQ 内のオブジェクトへアクセスする権限が与えられます。 IBM MQ の管理者は IBM MQ エクスプローラーを使用して、次の 3 つのタイプのいずれかの ID をクライアントに付与するようにテレメトリー・チャネルを構成します。

1. ClientIdentifier
2. ユーザー名
3. 管理者がチャネルに割り当てた名前。

いずれのタイプが使用される場合でも、ID は、インストールされた許可サービスによってプリンシパルとして IBM MQ に定義されている必要があります。 Windows または Linux でのデフォルトの許可サービスは、オブジェクト権限マネージャー (OAM) と呼ばれます。 OAM を使用する場合、ID はユーザー ID として定義される必要があります。

ID を使用して、IBM MQ で定義されているトピックへのパブリッシュおよびサブスクライブを行うためのアクセス権を 1 つのクライアントまたはクライアントのコレクションに付与します。 MQTT クライアントがトピックにサブスクライブした場合は、ID を使用して、結果としてのパブリケーションを受信するためのアクセス権をクライアントに付与します。

何万もの MQTT クライアントが存在し、各クライアントが個別のアクセス権限を必要とする場合、システムの管理は困難です。 1 つの解決策は、共通の ID をいくつか定義し、それらの共通 ID のいずれかを個々の MQTT クライアントに関連付けることです。 さまざまなアクセス権の組み合わせを定義するために、共通 ID を必要なだけ定義できます。 別の解決策は、オペレーティング・システムよりも簡単に何千ものユーザーを処理できる独自の許可サービスを記述することです。

OAM を使用して、MQTT クライアントを次の 2 つの方法で共通 ID に結合できます。

1. 複数のテレメトリー・チャネルを定義し、管理者が IBM MQ エクスプローラーを使用してそれぞれのチャネルに異なるユーザー ID を割り当てる。 異なる TCP/IP ポート番号を使用して接続するクライアントは、異なるテレメトリー・チャネルに関連付けられ、異なる ID が割り当てられます。
2. 単一のテレメトリー・チャネルを定義し、各クライアントは少数のユーザー ID からユーザー名を選択する。 管理者は、クライアントのユーザー名を ID として選択するようにテレメトリー・チャネルを構成します。

このタスクでは、テレメトリー・チャネルの ID は、設定方法に関係なく *mqttUser* と呼ばれます。 クライアントのコレクションが異なる ID を使用する場合は、複数の *mqttUsers* を使用して、それぞれをクライアントの各コレクションに使用します。 タスクは OAM を使用するため、各 *mqttUser* はユーザー ID である必要があります。

このタスクについて

このタスクでは、特定の要件に合わせて調整できる、4 つのアクセス制御パターンを選択できます。 これらのパターンは、アクセス制御の細分度が異なります。

- 343 ページの『アクセス制御なし』

- 343 ページの『粗い細分度のアクセス制御』
- 343 ページの『中程度の細分度のアクセス制御』
- 343 ページの『細い細分度のアクセス制御』

モデルの結果は、 IBM MQ にパブリッシュおよびサブスクライブするための *mqttUsers* 許可セットを割り当て、 IBM MQ からパブリケーションを受け取ることです。

アクセス制御なし
MQTT クライアントは、IBM MQ 管理権限が付与され、任意のオブジェクトに対する任意のアクションを実行できます。

手順

1. すべての MQTT クライアントの ID として機能するユーザー ID *mqttUser* を作成します。
2. Add *mqttUser* to the mqm group; see Windows 上のグループにユーザーを追加する , or Linux でのグループの作成および管理

粗い細分度のアクセス制御
MQTT クライアントは、パブリッシュとサブスクライブ、および MQTT クライアントへのメッセージの送信を行う権限を持ちます。 その他のアクションを実行する権限や、他のオブジェクトにアクセスする権限はありません。

手順

1. すべての MQTT クライアントの ID として機能するユーザー ID *mqttUser* を作成します。
2. すべてのトピックにパブリッシュおよびサブスクライブする権限、および MQTT クライアントにパブリケーションを送信する権限を *mqttUser* に与えます。

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

中程度の細分度のアクセス制御
さまざまなトピック・セットへのパブリッシュとサブスクライブ、および MQTT クライアントへのメッセージの送信を行うために、MQTT クライアントがさまざまなグループに分けられます。

手順

1. パブリッシュ/サブスクライブのトピック・ツリーに複数のユーザー ID、*mqttUsers*、および複数の管理トピックを作成します。
2. さまざまなトピックへの権限を別々の *mqttUsers* に与えます。

```
setmqaut -m qMgr -t topic -n topic1 -p mqttUserA -all +pub +sub
setmqaut -m qMgr -t topic -n topic2 -p mqttUserB -all +pub +sub
```

3. グループ *mqtt* を作成し、すべての *mqttUsers* をこのグループに追加します。
4. MQTT クライアントにトピックを送信する権限を *mqtt* に与えます。

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

細い細分度のアクセス制御
MQTT クライアントは、オブジェクトに対するアクションを実行する権限をグループに与える、既存システムのアクセス制御に組み込まれます。

このタスクについて

ユーザー ID は、必要な権限に応じて、1 つ以上のオペレーティング・システム・グループに割り当てられます。 IBM MQ アプリケーションが、MQTT クライアントと同じトピック・スペースへのパブリッシュおよびサブスクライブを行う場合は、このモデルを使用します。 グループは、 Publish *X*、 Subscribe *Y*、および mqtt と呼ばれます。

Publish *X*
Publish *X* グループのメンバーは、 *topicX* に公開できます。

Subscribe *Y*
Subscribe *Y* グループのメンバーは、 *topicY* にサブスクライブできます。

mqtt
mqtt グループのメンバーは、MQTT クライアントにパブリケーションを送信できます。

手順

1. パブリッシュ/サブスクライブ・トピック・ツリー内の複数の管理トピックに割り振られる複数のグループ Publish *X* および Subscribe *Y* を作成します。
2. グループ mqtt を作成します。
3. 複数のユーザー ID、*mqttUsers* を作成し、ユーザーに何を行う実行権限を与えるかということに応じて、いずれかのグループにユーザーを追加します。
4. 異なるトピックに対して異なる Publish *X* グループおよび Subscribe *X* グループを許可し、 MQTT クライアントにメッセージを送信する権限を *mqtt* グループに与えます。

```
setmqaut -m qMgr -t topic -n topic1 -p Publish X -all +pub
setmqaut -m qMgr -t topic -n topic1 -p Subscribe X -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Windows Linux AIX パスワードを使用する MQTT クライアントの認証

クライアント・パスワードを使用して Username を認証します。 トピックのパブリッシュおよびサブスクライブをクライアントに許可するために使用した ID とは別の ID を使用して、クライアントを認証できます。

テレメトリー (MQXR) サービスは、JAAS を使用してクライアント Username を認証します。 JAAS は、MQTT クライアントによって提供される パスワード を使用します。

IBM MQ 管理者は、クライアントが接続する MQTT チャネルを構成することによって、 Username を認証するかどうかを決定します。 クライアントを別のチャネルに割り当てたり、各チャネルを別の方法でクライアント認証するように構成したりできます。 JAAS を使用して、クライアントを認証する必要のあるメソッド、およびオプションでクライアントを認証できるメソッドを構成できます。

認証のための ID の選択は、許可のための ID の選択に影響を与えません。 管理に役立つように許可のための共通 ID をセットアップすることができますが、その場合には、その ID を使用するように各ユーザーを認証することになります。 以下の手順は、共通 ID を使用するように個別のユーザーを認証する方法の概要を示しています。

1. IBM MQ 管理者は、 IBM MQ エクスプローラーを使用して、 MQTT チャネル ID を任意の名前 (MQTTClientUser など) に設定します。
2. IBM MQ 管理者は、MQTTClient が任意のトピックにパブリッシュおよびサブスクライブすることを許可します。

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

3. MQTT クライアント・アプリケーション開発者は、MqttConnectOptions オブジェクトを作成して、サーバーに接続する前に Username および Password を設定します。

4. セキュリティー開発者は JAAS LoginModule を作成して、Username を Password によって認証し、それを JAAS 構成ファイルに組み込みます。
5. IBM MQ 管理者は、JAAS を使用してクライアントの ユーザー名 を認証するように MQTT チャネルを構成します。

Windows Linux AIX TLS を使用した MQTT クライアント認証

MQTT クライアントとキュー・マネージャーとの間の接続は、常に MQTT クライアントによって開始されます。 MQTT クライアントは常に SSL クライアントです。 サーバーのクライアント認証および MQTT クライアントのサーバー認証は、どちらもオプションです。

クライアントにプライベート署名付きデジタル証明書を提供することで、MQTT クライアントを認証して IBM MQ。 IBM MQ 管理者は、MQTT クライアントが TLS を使用してキュー・マネージャーに対して認証するように強制できます。 クライアント認証は、相互認証の一部としてのみ要求できます。

SSL を使用する代わりに、例えば IPsec など、いくつかの種類の仮想プライベート・ネットワーク (VPN) は TCP/IP 接続のエンドポイントを認証します。 VPN は、ネットワーク上を流れる各 IP パケットを暗号化します。 そのような VPN 接続が確立されると、トラステッド・ネットワークが確立されます。 VPN ネットワーク上で TCP/IP を使用して、MQTT クライアントをテレメトリー・チャネルに接続できます。

TLS を使用するクライアント認証は、機密事項を持つクライアントに依存します。 機密事項は、クライアントの秘密鍵 (自己署名証明書の場合) または認証局によって提供される鍵です。 この鍵は、クライアントのデジタル証明書を署名するために使用されます。 対応する公開鍵を持つユーザーであれば、デジタル証明書を検証できます。 証明書は信頼できます。 あるいはチェーニングされた証明書の場合には、トラステッド・ルート証明書に至るまで証明書チェーンをトレースバックしてください。 クライアント検査は、クライアントによって提供される証明書チェーン内のすべての証明書をサーバーに送ります。 サーバーは信頼できる証明書が見つかるまで証明書チェーンを検査します。 信頼できる証明書は、自己署名証明書から生成されるパブリック証明書、または通常は認証局で発行されるルート証明書のいずれかです。 オプションの最終ステップとして、信頼できる証明書を「現時点で有効な」証明書失効リストと比較できます。

信頼できる証明書は、認証局によって発行されており、JRE 証明書ストアに既に含まれている場合があります。 これは自己署名証明書、またはテレメトリー・チャネルの鍵ストアに信頼できる証明書として追加されたいずれかの証明書である場合があります。

注 : テレメトリー・チャネルには、1 つ以上のテレメトリー・チャネルに対する秘密鍵と、クライアントの認証に必要なパブリック証明書の両方を保持する、結合された鍵ストア/トラストストアが含まれています。 SSL チャネルには鍵ストアが含まれている必要があり、鍵ストアはチャネルのトラストストアと同じファイルであるため、JRE 証明書ストアが参照されることはありません。 これは、クライアントの認証で CA ルート証明書が必要な場合、CA ルート証明書が JRE 証明書ストアに既に存在する場合でも、ルート証明書をチャネルの鍵ストアに配置する必要があることを意味します。 JRE 証明書ストアが参照されることはありません。

クライアント認証が対抗する予定の危険、およびその危険に対抗するためのクライアントとサーバーの役割について考えてください。 クライアント証明書を認証するだけでは、システムに対する無許可アクセスを防止するために不十分です。 他人がクライアント装置を操作するようになった場合、そのクライアント装置は証明書の所有者の権限で動作しているとは限らなくなります。 望まれない攻撃に対抗するには、単一の防御だけに依存しないでください。 少なくとも 2 因子認証アプローチを使用して、証明書を所有しているかどうかを秘密情報の知識があるかどうかで補足します。 例えば、JAAS を使用すると共に、サーバーから発行されたパスワードを使用してクライアントを認証します。

クライアント証明書に関する第 1 の危険は、それが他人の手に渡ってしまうことです。 証明書は、クライアントにあるパスワードで保護された鍵ストアに保管されています。 それはどのような方法で鍵ストアに入れられますか。 MQTT クライアントはどのようにしてパスワードを鍵ストアに入れますか。 パスワード保護はどれほどセキュアですか。 テレメトリー装置は簡単に取り外せることが多く、人目につかない場所でのハッキングが可能になります。 装置のハードウェアを不正な改造から保護する必要がありますか。 クライアント・サイドの証明書を配布して保護することは困難であることが知られています。 これは鍵管理の問題と呼ばれます。

2 次的な危険は、意図されない方法でサーバーにアクセスするために装置が誤用されることです。 例えば、MQTT アプリケーションが不正に改造された場合、認証されたクライアント ID を使用してサーバー構成の弱点を利用できるようになる可能性があります。

SSL を使用して MQTT クライアントを認証するには、テレメトリー・チャネルおよびクライアントを構成します。

関連概念

346 ページの『TLS を使用した MQTT クライアント認証のためのテレメトリー・チャネルの構成』
の IBM MQ 管理者はサーバーでテレメトリ チャネルを構成します。 各チャネルは、異なるポート番号上の TCP/IP 接続を受け入れるように構成されます。 TLS チャネルは、鍵ファイルに対するパスフレーズで保護されたアクセスによって構成されます。 TLS チャネルがパスフレーズも鍵ファイルも使用しないで定義されている場合、このチャネルは TLS 接続を受け入れません。

TLS を使用したクライアント認証のための MQTT クライアントの構成

Windows Linux AIX *TLS を使用した MQTT クライアント認証のためのテレメトリー・チャネルの構成*

の IBM MQ 管理者はサーバーでテレメトリ チャネルを構成します。 各チャネルは、異なるポート番号上の TCP/IP 接続を受け入れるように構成されます。 TLS チャネルは、鍵ファイルに対するパスフレーズで保護されたアクセスによって構成されます。 TLS チャネルがパスフレーズも鍵ファイルも使用しないで定義されている場合、このチャネルは TLS 接続を受け入れません。

TLS テレメトリー・チャネルのプロパティー `com.ibm.mq.MQTT.ClientAuth` を `REQUIRED` に設定して、そのチャネルに接続しているすべてのクライアントに対して、検証済みのデジタル証明書があることを証明するように強制します。 クライアント証明書は、認証局からの証明書を使用して認証され、トラステッド・ルート証明書になります。 クライアント証明書が自己署名されているか、または認証局からの証明書で署名されている場合、クライアントの公開署名証明書、または認証局の証明書は、サーバーで安全に保管されている必要があります。

クライアントの公開署名証明書または認証局の証明書をテレメトリー・チャネルの鍵ストアに配置します。 サーバーでは、公開署名証明書は、秘密署名証明書と別のトラストストアではなく、同じ鍵ファイルに保管されます。

サーバーは、所有している公開証明書および暗号スイートをすべて使用して、送信されてきたクライアント証明書の署名を検証します。 サーバーは、鍵チェーンを検証します。 証明書取り消しリストに照らして証明書をテストするようにキュー・マネージャーを構成できます。 キュー・マネージャーの取り消し名前リストのプロパティーは `SSLCRLNL` です。

クライアントが送信したいずれかの証明書がサーバーの鍵ストア内の証明書で検証された場合、そのクライアントは認証されます。

IBM MQ 管理者は、同じテレメトリー・チャネルを、JAAS を使用してクライアントのパスワードでクライアントのユーザー名または `ClientIdentfier` を確認するように構成できます。

複数のテレメトリー・チャネルに対して同じ鍵ストアを使用できます。

装置上のパスワードで保護されたクライアント鍵ストア内の少なくとも 1 つのデジタル証明書を検証することで、サーバーに対するクライアントの認証を実行します。 デジタル証明書は、IBM MQ による認証にのみ使用されます。 クライアントの TCP/IP アドレスの検証や、許可またはアカウンティングのためのクライアント ID の設定には使用されません。 サーバーにより使用されるクライアント ID は、クライアントの `Username` または `ClientIdentifier` のいずれかか、IBM MQ 管理者により作成された ID です。

また、TLS 暗号スイートをクライアント認証に使用することもできます。 SHA-2 暗号スイートを使用する予定の場合は、350 ページの『MQTT チャネルで SHA-2 暗号スイートを使用する場合のシステム要件』を参照してください。

関連概念

347 ページの『TLS を使用したチャネル認証のためのテレメトリー・チャネル構成』
の IBM MQ 管理者はサーバーでテレメトリ チャネルを構成します。 各チャネルは、異なるポート番号上の TCP/IP 接続を受け入れるように構成されます。 TLS チャネルは、鍵ファイルに対するパスフレーズで保護されたアクセスによって構成されます。 TLS チャネルがパスフレーズも鍵ファイルも使用しないで定義されている場合、このチャネルは TLS 接続を受け入れません。

CipherSpec および CipherSuite

関連資料
チャネル定義 (MQTT)
チャネルの変更 (MQTT)

Windows Linux AIX TLS を使用したテレメトリー・チャネルの認証

MQTT クライアントとキュー・マネージャーとの間の接続は、常に MQTT クライアントによって開始されます。 MQTT クライアントは常に SSL クライアントです。 サーバーのクライアント認証および MQTT クライアントのサーバー認証は、どちらもオプションです。

クライアントが匿名接続をサポートする CipherSpec を使用するように構成されていない限り、クライアントは常にサーバーの認証を試行します。 認証が失敗すると、接続は確立されません。

SSL を使用する代わりに、例えば IPsec など、いくつかの種類の仮想プライベート・ネットワーク (VPN) は TCP/IP 接続のエンドポイントを認証します。 VPN は、ネットワーク上を流れる各 IP パケットを暗号化します。 そのような VPN 接続が確立されると、トラステッド・ネットワークが確立されます。 VPN ネットワーク上で TCP/IP を使用して、MQTT クライアントをテレメトリー・チャネルに接続できます。

SSL を使用するサーバー認証は、機密情報の送信先となるサーバーを認証します。 クライアントは、サーバーから送信された証明書と、トラストストアまたは JRE に配置された証明書を照合するチェックを実行します。`cacerts` 店。

JRE 証明書ストアは JKS ファイルです。`cacerts`。 それは `JRE InstallPath\lib\security\`。 これはデフォルト・パスワードの `changeit` を使用してインストールされます。 信頼するストア証明書は、JRE 証明書ストアまたはクライアントのトラストストアのいずれかに保管することができます。 両方のストアを使用することはできません。 クライアントが信頼する公開証明書を他の証明書とは別に保管したい場合は、クライアントトラストストアを使用します。Java アプリケーションの使用。 すべての証明書に共通の証明書ストアを使用する場合は、JRE 証明書ストアを使用します。Java クライアント上で実行されているアプリケーション。 JRE 証明書ストアを使用することにした場合は、その証明書ストアに含まれている証明書を検討して、それらの証明書が信頼できるものであることを保証してください。

別のトラスト・プロバイダーを提供して JSSE 構成を変更できます。 トラスト・プロバイダーを、証明書に対して別の検査を行うようにカスタマイズできます。 MQTT クライアントを使用していた一部の OGSi 環境では、環境が別のトラスト・プロバイダーを提供します。

TLS を使用してテレメトリー・チャネルを認証するには、サーバーおよびクライアントを構成します。

Windows Linux AIX TLS を使用したチャネル認証のためのテレメトリー・チャネル構成

の IBM MQ 管理者はサーバーでテレメトリ チャネルを構成します。 各チャネルは、異なるポート番号上の TCP/IP 接続を受け入れるように構成されます。 TLS チャネルは、鍵ファイルに対するパスフレーズで保護されたアクセスによって構成されます。 TLS チャネルがパスフレーズも鍵ファイルも使用しないで定義されている場合、このチャネルは TLS 接続を受け入れません。

サーバーの秘密鍵を使用して署名されたデジタル証明書は、テレメトリー・チャネルがサーバーで使用する予定の鍵ストアに保管します。 サーバーの鍵チェーンをクライアントに送信する場合は、その鍵チェーン内の証明書はすべて、その鍵ストアに保管します。 IBM MQ エクスプローラーを使用するテレメトリー・チャネルは、TLS を使用するよう構成します。 そのようなテレメトリー・チャネルには、鍵ストアへのパスと、鍵ストアにアクセスするためのパスフレーズを与えます。 テレメトリー・チャネルの TCP/IP ポート番号を設定しない場合、TLS テレメトリー・チャネルのポート番号はデフォルトで 8883 に設定されます。

また、TLS 暗号スイートをチャネル認証に使用することもできます。 SHA-2 暗号スイートを使用する予定の場合は、350 ページの『MQTT チャネルで SHA-2 暗号スイートを使用する場合のシステム要件』を参照してください。

重要： V 9.4.0 V 9.4.0 IBM MQ 9.4.0 以降、 CMS 鍵リポジトリーおよび stash ファイルは、SSL/TLS を使用する AMQP チャネルおよび MQTT チャネルではサポートされません。 代わりに、PKCS

#12 鍵リポジトリーを使用し、 IBM MQ パスワード保護システムを使用して鍵リポジトリーのパスワードを保護します。 以下のコマンドを使用して、PKCS #12 鍵リポジトリーを作成できます。

```
runmqakm -keydb -create -db filename.p12 -pw password -type pkcs12
```

このコマンドは、指定されたパスワードで保護された *filename*.p12 という名前の PKCS #12 キー・リポジトリー・ファイルを作成します。

関連概念

346 ページの『TLS を使用した MQTT クライアント認証のためのテレメトリー・チャネルの構成』

の IBM MQ 管理者はサーバーでテレメトリ チャネルを構成します。 各チャネルは、異なるポート番号上の TCP/IP 接続を受け入れるように構成されます。 TLS チャネルは、鍵ファイルに対するパスフレーズで保護されたアクセスによって構成されます。 TLS チャネルがパスフレーズも鍵ファイルも使用しないで定義されている場合、このチャネルは TLS 接続を受け入れません。

CipherSpec および CipherSuite

関連資料

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

TLS 認証を使用した *MQTT* チャネル構成の例

この例では、TLS 認証を使用する MQTT チャネルの構成例について説明します。

この例では、 MQTT と Mosquitto の間のチャネルを構成します。

この例では、 Red Hat Enterprise Linux 上の IBM MQ と CentOS 上の Mosquitto の両方に Docker コンテナーを使用しますが、どのタイプのサーバーにも適用されます。 (レジストリー資格のために、CentOS が Mosquitto に使用されました。)

片方向 TLS 用の IBM MQ 鍵ストアおよびチャネルの構成

重要 : V 9.4.0 V 9.4.0 IBM MQ 9.4.0 以降、 CMS 鍵リポジトリーおよび stash ファイルは、SSL/TLS を使用する AMQP チャネルおよび MQTT チャネルではサポートされません。 代わりに、PKCS #12 鍵リポジトリーを使用し、 IBM MQ パスワード保護システムを使用して鍵リポジトリーのパスワードを保護します。

以下のステップを実行します。

1. V 9.4.0 V 9.4.0 IBM MQ 鍵ストアを作成します。

   ```
   runmqakm -keydb -create -db mqtt.p12 -pw "passw0rd" -type p12
   ```

2. V 9.4.0 V 9.4.0 個人証明書を作成します。

   ```
   runmqakm -cert -create -db mqtt.p12 -pw "passw0rd" -size 2048 -dn "CN= mqm, OU=MQTest,
   O=MQSuppor, C=US" -sig_alg SHA256_WITH_RSA -label ibmwebspheremqmqm
   ```

   以下のコマンドを使用して、証明書の作成を確認できます。

   ```
   runmqakm -cert -list -v -db mqtt.p12 -pw "passw0rd"
   ```

3. V 9.4.0 V 9.4.0 runmqsc プロンプトで以下のコマンドを入力して、 MQTT チャネルを作成します。

   ```
   DEFINE CHANNEL(MQTTDEMO) CHLTYPE(MQTT) BACKLOG(4096) PORT(8883) MCAUSER('mqm')
   PROTOCOL(MQTTV311,MQTTV3,HTTP) SSLCAUTH(OPTIONAL) SSLCIPH('SSL_RSA_WITH_AES_256_CBC_SHA256')
   SSLKEYR('/var/mqm/mqtt/mqtt.p12') SSLKEYP('passw0rd') TRPTYPE(TCP)
   ```

   チャネルは Java 暗号マッピングを使用することに注意してください。 IBM MQ classes for JMS の TLS CipherSpecs および CipherSuites を参照してください。

4. 証明書を抽出します。

```
runmqakm -cert -extract -db mqtt.kdb -stashed -label ibmwebspheremqmqm -target serverCert.pem
```

Docker コンテナー内の CentOS に Mosquitto をインストールする

CentOS で実行する Mosquitto を使用して Docker コンテナーを作成するには、以下の手順を実行します。

1. `docker pull centos`
2. `docker run -it centos /bin/bash`
3. `yum -y install epel-release`
4. `yum -y install mosquitto`

署名者証明書を Mosquitto に移動する

IBM MQ で作成した証明書を Mosquitto に移動するには、以下の手順を実行します。 これらのステップは、Docker ホスト・マシン上で実行されます。

1. Docker のコンテナー ID を表示します。

   ```
   docker container ls
   ```

2. Docker コンテナーからローカル・システム Docker にファイルをコピーします。

   ```
   cp MQ_Container_ID:/var/mqm/mqtt/serverCert.pem serverCert.pem
   ```

3. ローカル・マシンから centOS マシン上のルート・ディレクトリーにファイルをコピーします。

   ```
   docker cp serverCert.pem CentOS_ContainerID:/serverCert.pem
   ```

Mosquitto で公開

以下のコマンドを使用して、Mosquitto でテスト・メッセージをパブリッシュします。

```
mosquitto_pub -h 172.17.0.2 --cafile serverCert.pem --insecure -p 8883 -i mosquittoClient -t
test -m 'test message' -d
```

コマンド引数の意味は次のとおりです。

-h
Red Hat Enterprise Linux ホストの IP アドレス (**nslookup** を使用して見つけることができます)。

-- cafile
署名者証明書を含むファイル。

-- 非セキュア
この例では自己署名証明書を使用しているため、このオプションを指定します。 実際の CA 証明書を使用する場合は、このオプションを使用しないでください。

-p
ポート番号。

-i
クライアント ID。

-t
パブリッシュ先のトピック。

-m
パブリッシュされるメッセージ。

-d
デバッグ・メッセージを有効にします。

相互 TLS 認証のための MQTT チャネルの構成

以下のコマンドを入力して、 MQTT チャネルを SSLCAUTH (REQUIRED) として再構成します。

```
ALTER CHANNEL(MQTTDEMO) CHLTYPE(MQTT) SSLCAUTH(REQUIRED)
```

Mosquitto サーバー上に鍵と証明書のペアを作成し、 IBM MQ に追加します。

以下のコマンドを入力して、Mosquitto 上に鍵/証明書のペアを作成します。

1. **openSSL** を使用して、Mosquitto 用の鍵と証明書のペアを作成します。

   ```
   openssl req -x509 -newkey rsa:4096 -keyout mosquittoKey.pem -out mosquittoCert.pem -subj "/
   CN=Mosquitto"
   ```

2. コンテナーのコンテナー ID をリストします。

   ```
   docker container ls
   ```

3. 以下のようにして、Mosquitto 証明書をローカル・システムの Docker にコピーします。

   ```
   docker cp CentOS_ContainerID:mosquittoCert.pem .
   ```

4. Mosquitto 証明書を IBM MQ にコピーします。

   ```
   docker cp mosquittoCert.pem MQ_Container_ID:/var/mqm/mqtt
   ```

5. 証明書を IBM MQ 鍵ストアに追加します。

   ```
   runmqakm -cert -add -db mqtt.kdb -stashed -file mosquittoCert.pem
   ```

6. MQTT チャネルを再始動します。

Mosquitto と相互認証を使用した公開

相互認証を使用して Mosquitto で公開するには、以下の手順を実行します。

1. 以下のコマンドは、テスト・メッセージを正常にパブリッシュする必要があります。

   ```
   mosquitto_pub -h 172.17.0.2 --cafile serverCert.pem --insecure -p 8883 -i mosquittoClient -t
   test -m 'test message' -d --cert mosquittoCert.pem --key mosquittoKey.pem
   ```

2. 以下のコマンドは、Mosquitto から個人証明書を送信しないため、テスト・メッセージを公開してエラー・メッセージを生成することはできません。

   ```
   mosquitto_pub -h 172.17.0.2 --cafile serverCert.pem --insecure -p 8883 -i mosquittoClient -t
   test -m 'test message' -d /var/mqm/qmgrs/mqttDemoQM/errors/ mqxr_0.log
   ```

関連情報

鍵と証明書の管理

Windows Linux AIX MQTT チャネルで SHA-2 暗号スイートを使用する場合のシステム要件

SHA-2 暗号スイートをサポートする Java のバージョンを使用する場合、これらのスイートを使用して MQTT (テレメトリー) チャネルおよびクライアント・アプリケーションを保護できます。

テレメトリー (MQXR) サービスが組み込まれている IBM MQ 8.0 では、最小 Java バージョンは Java 7 (IBM 製) SR6 です。 SHA-2 暗号スイートは、Java 7 (IBM 製) SR4 以降でデフォルトでサポートされています。 そのため、テレメトリー (MQXR) サービスとともに SHA-2 暗号スイートを使用して MQTT (テレメトリー) チャネルを保護できます。

別の JRE で MQTT クライアントを実行する場合、SHA-2 暗号スイートもサポートしていることを確認する必要があります。

関連概念

遠隔測定 (MQXR) サービス

347 ページの『TLS を使用したチャネル認証のためのテレメトリー・チャネル構成』

の IBM MQ 管理者はサーバーでテレメトリ チャネルを構成します。 各チャネルは、異なるポート番号上の TCP/IP 接続を受け入れるように構成されます。 TLS チャネルは、鍵ファイルに対するパスフレーズで保護されたアクセスによって構成されます。 TLS チャネルがパスフレーズも鍵ファイルも使用しないで定義されている場合、このチャネルは TLS 接続を受け入れません。

関連資料

チャネル定義 (MQTT)

チャネルの変更 (MQTT)

Windows Linux AIX テレメトリー・チャネルでのパブリケーションのプライバシー

テレメトリー・チャネル間でいずれかの方向に送信される MQTT パブリケーションのプライバシーは、TLS を使用して接続上の伝送を暗号化することにより保護されます。

テレメトリー・チャネルに接続する MQTT クライアントは、TLS を使用して、対称鍵暗号方式を使用しているチャネル上を伝送されるパブリケーションのプライバシーを保護します。 エンドポイントは認証されないため、 チャネルの暗号化を単独で信用することはできません。 プライバシーの保護と、サーバー認証または相互認証とを結合します。

SSL を使用する代わりに、例えば IPsec など、いくつかの種類の仮想プライベート・ネットワーク (VPN) は TCP/IP 接続のエンドポイントを認証します。 VPN は、ネットワーク上を流れる各 IP パケットを暗号化します。 そのような VPN 接続が確立されると、トラステッド・ネットワークが確立されます。 VPN ネットワーク上で TCP/IP を使用して、MQTT クライアントをテレメトリー・チャネルに接続できます。

チャネルを暗号化し、サーバーを認証する標準的な構成の場合については、347 ページの『TLS を使用したテレメトリー・チャネルの認証』を参照してください。

サーバーを認証しないで TLS 接続を暗号化すると、接続は中間者攻撃にさらされます。 交換する情報は盗聴に対しては保護されますが、その情報を交換している相手が誰であるかは分かりません。 ネットワークを制御しない限り、IP 伝送を傍受し、エンドポイントになりすましている誰かにさらされます。

匿名 TLS をサポートする Diffie-Hellman 鍵交換 CipherSpec を使用することにより、サーバーを認証しなくても暗号化された TLS 接続を作成することができます。 非公開署名されたサーバー証明書を交換しなくても、クライアントとサーバーの間で共有され、TLS 伝送を暗号化するために使用されるマスター・シークレットが確立されます。

匿名接続は安全ではないので、ほとんどの TLS 実装では、デフォルトでは匿名の CipherSpec は使用されません。 テレメトリー・チャネルが TLS 接続を要求するクライアント要求を受け入れる場合、そのテレメトリー・チャネルは、鍵ストアをパスフレーズで保護する必要があります。 デフォルトでは、TLS 実装は匿名の CipherSpec を使用しないので、クライアントが認証できる、非公開署名された証明書を鍵ストアに含める必要があります。

匿名の CipherSpec を使用する場合は、サーバーの鍵ストアが存在している必要がありますが、非公開署名された証明書が含まれている必要はありません。

暗号化された接続を確立するための別の方法は、クライアント側にあるトラスト・プロバイダーを独自の実装で置き換えることです。 この独自の実装のトラスト・プロバイダーはサーバー証明書を認証しないでしょうが、接続は暗号化されます。

重要 : MQTT で TLS を使用する場合は、大きなメッセージを使用できますが、使用するとパフォーマンスに影響する可能性があります。 MQTT は、小さなメッセージ (通常は 1KB から 1MB までのサイズ) を処理するために最適化されています。

Windows Linux AIX MQTT Java クライアントおよびテレメトリー・チャネルの TLS 構成

テレメトリー・チャネルおよび MQTT Java クライアントを認証し、それらの間のメッセージ転送を暗号化するように TLS を構成します。MQTT Java クライアントは、 Java Secure Socket Extension (JSSE) を使用して、TLS を使用するテレメトリー・チャネルに接続します。 SSL を使用する代わりに、例えば IPsec など、いくつかの種類の仮想プライベート・ネットワーク (VPN) は TCP/IP 接続のエンドポイントを認証します。 VPN は、ネットワーク上を流れる各 IP パケットを暗号化します。 そのような VPN 接続が確立されると、トラステッド・ネットワークが確立されます。 VPN ネットワーク上で TCP/IP を使用して、MQTT クライアントをテレメトリー・チャネルに接続できます。

Java MQTT クライアントとテレメトリー・チャネルの間の接続は、TCP/IP 上での TLS プロトコルを使用するように構成できます。 保護対象は、JSSE を使用するために TLS がどのように構成されているのかによって異なります。 最も保護の高い構成から順になっている、以下の 3 つの異なるレベルのセキュリティーを構成できます。

1. 信頼できる MQTT クライアントのみに接続を許可します。 信頼できるテレメトリー・チャネルにのみ MQTT クライアントを接続します。 クライアントとキュー・マネージャーの間のメッセージを暗号化します。345 ページの『TLS を使用した MQTT クライアント認証』を参照してください。
2. 信頼できるテレメトリー・チャネルにのみ MQTT クライアントを接続します。 クライアントとキュー・マネージャーの間のメッセージを暗号化します。347 ページの『TLS を使用したテレメトリー・チャネルの認証』を参照してください。
3. クライアントとキュー・マネージャーの間のメッセージを暗号化します。351 ページの『テレメトリー・チャネルでのパブリケーションのプライバシー』を参照してください。

JSSE 構成パラメーター

JSSE パラメーターを変更して、TLS 接続の構成方法を変えます。 JSSE 構成パラメーターは次の 3 つのセットに配置されます。

1. MQ Telemetry チャネル
2. MQTT Java クライアント
3. JRE

IBM MQ エクスプローラーを使用して、テレメトリー・チャネル・パラメーターを構成します。 MQTT Java クライアント・パラメーターは `MqttConnectionOptions.SSLProperties` 属性で設定します。 クライアント側およびサーバー側の両方の JRE の security ディレクトリー内のファイルを編集して、JRE セキュリティー・パラメーターを変更します。

MQ Telemetry チャネル

IBM MQ エクスプローラーを使用して、テレメトリー・チャネルのすべての TLS パラメーターを設定します。

`ChannelName`

`ChannelName` は、すべてのチャネルで必須のパラメーターです。

チャネル名は、特定のポート番号に関連付けられているチャネルを識別します。 チャネルには、MQTT クライアント・セットを管理するのに役立つ名前を付けてください。

`PortNumber`

`PortNumber` は、すべてのチャネルのオプション・パラメーターです。 このパラメーターのデフォルトは、TCP チャネルの場合は 1883 で、TLS チャネルの場合は 8883 です。

このチャネルに関連付けられている TCP/IP ポート番号。 チャネルに対して定義されているポートを指定することにより、MQTT クライアントはそのチャネルに接続されます。 チャネルが TLS プロ

パティーを持っている場合、クライアントは TLS プロトコルを使用して接続する必要があります。以下に例を示します。

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

KeyFileName は、TLS チャネルに必須のパラメーターです。 TCP チャネルの場合は、このパラメーターを省略する必要があります。

KeyFileName は、提供されたデジタル証明書を含む Java 鍵ストアへのパスです。 サーバー側の鍵ストアのタイプとしては、JKS、JCEKS または PKCS12 を使用します。

鍵ストア・タイプを識別するには、以下に示したいずれかのファイル拡張子を使用します。

```
.jks
.jceks
.p12
.pkcs12
```

上記以外のファイル拡張子を持つ鍵ストアは、JKS 鍵ストアと見なされます。

サーバー側のあるタイプの鍵ストアと、クライアント側のそれ以外のタイプの鍵ストアを組み合わせることができます。

サーバーのプライベート証明書は、鍵ストアに配置します。 この証明書はサーバー証明書と呼ばれます。 この証明書は自己署名することも、署名権限によって署名される証明書チェーンの一部にすることもできます。

証明書チェーンを使用する場合は、サーバーの鍵ストア内に関連付けられている証明書を配置します。

サーバー証明書、およびサーバーの鍵チェーン内の証明書は、サーバーの ID を認証するためにクライアントに送信されます。

ClientAuth を Required に設定していた場合、鍵ストアには、クライアントを認証するのに必要な証明書が含まれていなければなりません。 クライアントは自己署名証明書、または証明書チェーンを送信し、クライアントは鍵ストア内の証明書に対するこの内容の最初の検証によって認証されます。 証明書チェーンを使用すると、たとえさまざまなクライアント証明書と一緒にクライアントが発行されていたとしても、1 つの証明書で複数のクライアントを検証することができます。

PassPhrase

PassPhrase は、TLS チャネルに必須のパラメーターです。 TCP チャネルの場合は、このパラメーターを省略する必要があります。

鍵ストアの保護には、パスフレーズが使用されます。

ClientAuth

ClientAuth はオプションの TLS パラメーターです。 このパラメーターのデフォルトは、クライアントを認証しない、です。 TCP チャネルの場合は、このパラメーターを省略する必要があります。

クライアントがテレメトリー・チャネルに接続することを許可する前に、テレメトリー (MQXR) サービスがクライアントを認証するようにするには、ClientAuth を設定します。

ClientAuth を設定した場合、クライアントは TLS を使用しているサーバーに接続し、そのサーバーを認証しなければなりません。 ClientAuth の設定に対する応答として、クライアントはそのデジタル証明書と、その鍵ストア内にあるその他の証明書をサーバーに送信します。 このデジタル証明書は、クライアント証明書と呼ばれます。 これらの証明書は、チャネル鍵ストアに保持されている証明書、および JRE cacerts ストアで認証されます。

CipherSuite

CipherSuite は、オプションの TLS パラメーターです。 このパラメーターのデフォルトは、有効な CipherSpec をすべて試行する、です。 TCP チャネルの場合は、このパラメーターを省略する必要があります。

特定の CipherSpec を使用する場合は、CipherSuite を TLS 接続の確立に使用する必要のある CipherSpec の名前に設定します。

テレメトリー・サービスと MQTT クライアントは、各端点で有効になっているすべての CipherSpec の中から共通の CipherSpec を折衝します。 特定の CipherSpec が接続の片端または両端で指定された場合、その CipherSpec はもう一方の端の CipherSpec に一致しなければなりません。

追加のプロバイダーを JSSE に追加することにより、追加の暗号をインストールします。

連邦情報処理標準 (FIPS)

FIPS は、オプションの設定です。 デフォルトでは、これは設定されていません。

キュー・マネージャーのプロパティー・パネルで、または **runmqsc** を使用して、SSLFIPS を設定します。 SSLFIPS は、FIPS によって証明されたアルゴリズムだけを使用するのかどうかを指定します。

取り消し名前リスト

取り消し名前リストは、オプションの設定です。 デフォルトでは、これは設定されていません。

キュー・マネージャーのプロパティー・パネルで、または **runmqsc** を使用して、SSLCRLNL を設定します。 SSLCRLNL は、証明書取り消し場所を提供するために使用される認証情報オブジェクトの名前リストを指定します。

TLS プロパティーを設定するその他のキュー・マネージャー・パラメーターは使用されません。

MQTT Java クライアント

Java クライアントの TLS プロパティーを MqttConnectionOptions.SSLProperties で設定します。以下に例を示します。

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

特定のプロパティーの名前と値は、 MqttConnectOptions クラスで説明されています。 MQTT クライアント・ライブラリーのクライアント API 資料へのリンクについては、 MQTT クライアント・プログラミング・リファレンスを参照してください。

Protocol

Protocol はオプションです。

このプロトコルは、テレメトリー・サーバーとの折衝で選択されます。 特定のプロトコルが必要な場合は、それを選択することができます。 テレメトリー・サーバーがそのプロトコルをサポートしていない場合、接続は失敗します。

ContextProvider

ContextProvider はオプションです。

KeyStore

KeyStore はオプションです。 クライアントの認証を強制的に行うために ClientAuth がサーバー側で設定されている場合は、これを構成します。

クライアントの秘密鍵を使用して署名されている、クライアントのデジタル証明書を鍵ストアに配置します。 鍵ストアのパスとパスワードを指定します。 タイプとプロバイダーはオプションです。 デフォルトのタイプは JKS、デフォルトのプロバイダーは IBMJCE です。

新規の鍵ストア・プロバイダーを追加するクラスを参照するには、別の鍵ストア・プロバイダーを指定します。 鍵ストア・プロバイダーが使用するアルゴリズムの名前を渡し、鍵マネージャー名を設定して KeyManagerFactory のインスタンスを生成します。

TrustStore

TrustStore はオプションです。 信頼できるすべての証明書を、 JRE cacerts ストアに配置することができます。

クライアント用に別のトラストストアが必要な場合には、このトラストストアを構成します。 既にルート証明書が cacerts に保管されている既知の CA によって発行された証明書をサーバーが使用している場合は、トラストストアを構成する必要がない可能性があります。

サーバーの公開署名された証明書またはルート証明書をトラストストアに追加し、トラストストアのパスとパスワードを指定します。 デフォルトのタイプは JKS、デフォルトのプロバイダーは IBMJCE です。

新規のトラストストア・プロバイダーを追加するクラスを参照するには、別のトラストストア・プロバイダーを指定します。 トラストストア・プロバイダーが使用するアルゴリズムの名前を渡し、トラスト・マネージャー名を設定して TrustManagerFactory のインスタンスを生成します。

JRE

クライアント側およびサーバー側の両方での TLS の動作に影響を与える Java セキュリティーの他の側面が JRE 内に構成されます。 Windows 上の構成ファイルは、 *Java Installation Directory*\jre\lib\security にあります。 IBM MQ に同梱されている JRE を使用している場合のパスは、以下の表に示すとおりです。

表 *18. JRE TLS* 構成ファイルのプラットフォーム別ファイル・パス

プラットフォーム	ファイル・パス
Windows	*WMQ Installation Directory*\java\jre\lib\security
AIX and Linux プラットフォーム	*WMQ Installation Directory*/java/jre64/jre/lib/security

既知の認証局

cacerts ファイルには、既知の認証局のルート証明書が含まれます。 トラストストアを指定しない限り、cacerts はデフォルトで使用されます。 cacerts ストアを使用する場合、またはトラストストアを提供しない場合は、セキュリティー要件を満たすように、 cacerts 内の署名者のリストを確認し、編集する必要があります。

V 9.4.0 V 9.4.0 **runmqktool** コマンドを使用して、 cacerts 証明書ファイルを管理できます。 cacerts は JKS ファイルです。 **runmqktool** コマンドを使用して証明書ファイルを管理する場合は、パラメーター **-storetype jks** を指定します。

V 9.4.0 V 9.4.0 cacerts ファイルのデフォルトのパスワードは changeit です。 **runmqktool -storepasswd** コマンドを使用してパスワードを変更し、ファイルを保護します。

セキュリティー・クラスの構成

java.security ファイルを使用して、追加のセキュリティー・プロバイダーおよびその他のデフォルト・セキュリティー・プロパティーを登録する

権限

java.policy ファイルを使用して、リソースに付与された許可を変更します。 javaws.policy は許可を javaws.jar に付与します。

暗号化の強度

一部の JRE は、暗号化の強度を下げて出荷されています。 鍵を鍵ストアにインポートできない場合は、暗号化の強度が下げられているのが原因である場合があります。 必要に応じて、 IBM

Developer Kit のセキュリティー情報から、強力で限定された管轄権ファイルをダウンロードします。

重要 : お住まいの国によっては、暗号化ソフトウェアの輸入、所持、使用、または別の国への再輸出に制限が課せられている場合があります。 お住まいの国の法律を確認してから、規制されていないポリシー・ファイルをダウンロードして使用するようにしてください。 暗号化ソフトウェアの輸入、所持、使用、および再輸出に関する国の規制および方針を確認して、それが許可されているのかどうかを決定してください。

任意のサーバーへの接続をクライアントに許可するようトラスト・プロバイダーを変更する

この例は、トラスト・プロバイダーを追加して、MQTT クライアント・コードからそのプロバイダーを参照する方法を示しています。 この例では、クライアントまたはサーバーの認証は実行されません。 その結果、TLS 接続は暗号化されますが、認証されません。

356 ページの図 16 のコード・スニペットは、 MQTT クライアントの `AcceptAllProviders` トラスト・プロバイダーとトラスト・マネージャーを設定します。

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

図 *16. MQTT* クライアントのコード・スニペット

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
private static final long serialVersionUID = 1L;
public AcceptAllProvider() {
super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
put("TrustManagerFactory.TrustAllCertificates",
AcceptAllTrustManagerFactory.class.getName());
}
```

図 *17. AcceptAllProvider.java*

```
protected static class AcceptAllTrustManagerFactory extends
javax.net.ssl.TrustManagerFactorySpi {
public AcceptAllTrustManagerFactory() {}
protected void engineInit(java.security.KeyStore keystore) {}
protected void engineInit(
javax.net.ssl.ManagerFactoryParameters parameters) {}
protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
}
```

図 *18. AcceptAllTrustManagerFactory.java*

```
protected static class AcceptAllX509TrustManager implements
javax.net.ssl.X509TrustManager {
public void checkClientTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Client authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public void checkServerTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Server authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public java.security.cert.X509Certificate[] getAcceptedIssuers() {
return new java.security.cert.X509Certificate[0];
}
private static void report(String string) {
System.out.println(string);
}
}
```

図 *19. AcceptAllX509TrustManager.java*

Windows Linux AIX テレメトリー・チャネルの JAAS 構成

クライアントから送られた Username を認証するように JAAS を構成します。

IBM MQ 管理者は、JAAS を使用するクライアント認証を必要と MQTT チャネルを構成する。 JAAS 認証を実行する各チャネルの JAAS 構成の名前を指定します。 すべてのチャネルが同じ JAAS 構成を使用することもでき、それぞれが異なる JAAS 構成を使用することもできます。 構成は、 *WMQData directory*\qmgrs*qMgrName*\mqxr\jaas.config で定義されます。

jaas.config ファイルは、 JAAS 構成名によって編成されています。 それぞれの構成名の下には、ログイン構成のリストがあります。 358 ページの『サンプル・ jaas.config ファイル』を参照してください。

JAAS は、4 つの標準ログイン・モジュールを提供します。 標準の NT および UNIX ログイン・モジュールの価値は、限られたものです。

JndiLoginModule
JNDI (Java Naming and Directory Interface) の下で構成されたディレクトリー・サービスに対して認証します。

Krb5LoginModule
Kerberos プロトコルを使用して認証します。

NTLoginModule
現行ユーザーの NT セキュリティー情報を使用して認証します。

UnixLoginModule
現行ユーザーの UNIX セキュリティー情報を使用して認証します。

NTLoginModule または UnixLoginModule を使用する場合の問題は、 MQTT チャネルの ID ではなく、mqm ID を使用してテレメトリー (MQXR) サービスが実行されることです。 mqm は、認証のために NTLoginModule または UnixLoginModule に渡される ID であり、クライアントの ID ではありません。

この問題を解決するには、独自のログイン・モジュールを記述するか、または他の標準ログイン・モジュールを使用します。 サンプルの JAASLoginModule.java が MQ Telemetry で提供されています。 これは、javax.security.auth.spi.LoginModule インターフェースのインプリメンテーションです。 これを使用して、独自の認証方式を開発します。

提供する新しい LoginModule クラスは、テレメトリー (MQXR) サービスのクラスパス上に存在しなければなりません。 そのクラスを、クラスパスに含まれる IBM MQ ディレクトリーに入れないでください。 独自のディレクトリーを作成して、テレメトリー (MQXR) サービスのためのクラスパス全体を定義します。

テレメトリー (MQXR) サービスによって使用されるクラスパスを拡張するには、 service.env ファイルにクラスパスを設定します。 CLASSPATH は大文字で記述する必要があり、クラスパス・ステートメントに含めることができるのはリテラルだけです。 CLASSPATH の変数を使用することはできません。例えば、CLASSPATH=%CLASSPATH% が正しくありません。 テレメトリー (MQXR) サービスは、独自のクラスパスを設定します。 service.env で定義されている CLASSPATH が追加されます。

テレメトリー (MQXR) サービスは、MQTT チャネルに接続されているクライアントの ユーザー名 と パスワード を返す 2 つのコールバックを提供します。 「ユーザー名」 および 「パスワード」 は、MqttConnectOptions オブジェクトで設定されます。 Username および Password にアクセスする方法について詳しくは、358 ページの『JAASLoginModule.Login() メソッドのサンプル』を参照してください。

サンプル・ jaas.config ファイル

指名された 1 つの構成 MQXRConfig がある JAAS 構成ファイルの例

```
MQXRConfig {
samples.JAASLoginModule required debug=true;
//com.ibm.security.auth.module.NTLoginModule required;
//com.ibm.security.auth.module.Krb5LoginModule required
//         principal=principal@your_realm
//         useDefaultCcache=TRUE
//         renewTGT=true;
//com.sun.security.auth.module.NTLoginModule required;
//com.sun.security.auth.module.UnixLoginModule required;
//com.sun.security.auth.module.Krb5LoginModule required
//         useTicketCache="true"
//         ticketCache="${user.home}${/}tickets";
};
```

JAASLoginModule.Login() メソッドのサンプル

MQTT クライアントによって提供される ユーザー名 および パスワード を受け取るようにコーディングされた JAAS ログイン・モジュールの例。

```
public boolean login()
throws javax.security.auth.login.LoginException {
javax.security.auth.callback.Callback[] callbacks =
new javax.security.auth.callback.Callback[2];
callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
callbacks[1] = new javax.security.auth.callback.PasswordCallback(
"PasswordCallback", false);
try {
callbackHandler.handle(callbacks);
String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
.getName();
char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
.getPassword();
 // Accept everything.
if (true) {
loggedIn = true;
} else
throw new javax.security.auth.login.FailedLoginException("Login failed");

principal= new JAASPrincipal(username);

} catch (java.io.IOException exception) {
throw new javax.security.auth.login.LoginException(exception.toString());
} catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
throw new javax.security.auth.login.LoginException(exception.toString());
}

return loggedIn;
}
```

関連タスク

問題の解決: JAAS ログイン・モジュールがテレメトリー・サービスによって呼び出されない

関連資料
AuthCallback MQXR クラス

AMQP クライアントの管理

AMQP クライアントは、 IBM MQ Explorer を使用して管理することも、コマンドラインで管理することもできます。 エクスプローラーを使用して、チャネルを構成し、 IBM MQ に接続されている AMQP クライアントをモニターします。 TLS および JAAS を使用して AMQP クライアントのセキュリティーを構成します。

始める前に

ご使用のプラットフォームでの AMQP のインストールについては、インストール内容の選択を参照してください。

IBM MQ Explorer を使用した管理

エクスプローラーを使用して、AMQP チャネルを構成し、 IBM MQ に接続されている AMQP クライアントをモニターします。 TLS および JAAS を使用して AMQP クライアントのセキュリティーを構成できます。

コマンド行を使用した管理

コマンド・ライン MQSC コマンドの使用で AMQP クライアントを管理できます。

Windows V 9.4.0 Linux V 9.4.0 AIX AMQP サービスがキュー・マネージャーの始動時に自動的に開始しない

IBM MQ 9.4.0 以降、AMQP サービスを開始するための **CONTROL** 属性の設定のデフォルトの動作が変更されました。 新しいキュー・マネージャーを作成して開始するときに、AMQP サービスがキュー・マネージャーの開始プロセスの一部として自動的に開始されることはありません。

IBM MQ 9.0.4 と IBM MQ 9.4.0 の間では、AMQP サービスを開始するための **CONTROL** 属性の設定のデフォルトの動作は QMGR です。

AMQP コンポーネントがインストールされている場合、AMQP サービスは、使用されていなくても自動的に開始されます。 AMQP Java 仮想マシン (JVM) のデフォルトの始動を回避するために、以下の 2 つのオプションがありました。

- AMQP コンポーネントがインストールされていない、または
- キュー・マネージャーの開始後に AMQP サービスの **CONTROL** 属性を MANUAL に変更する。

IBM MQ 9.4.0 以降、新しく作成されたキュー・マネージャーは、 SYSTEM.AMQP.SERVICE の **CONTROL** 属性の設定を 手動に戻しました。これは、 IBM MQ 9.0.4 より前のデフォルト設定です。

マイグレーションされたキュー・マネージャーは、AMQP を使用している場合、キュー・マネージャーの始動時にサービスの自動開始を続行します。 AMQP が使用されたかどうかを判別するために、以下が検査されます。

- 既存の AMQP チャネル
- AMQP エラー・ログ内のチャネル開始メッセージ。

重要 :

- これは 1 回だけ行われます。アップグレード後に初めてキュー・マネージャーが開始されます。
- マイグレーション中に **CONTROL** 属性が QMGR から MANUAL に変更された場合、変更を示す情報メッセージが IBM MQ エラー・ログに記録されます。 詳しくは、 AMQP ログ、エラー・ログ、および構成ファイルの場所 を参照してください。

AMQP サービスを自動開始する場合は、サービスの **CONTROL** 属性を QMGR に変更し、キュー・マネージャーを再始動します。 その後、キュー・マネージャーを再始動すると、AMQP サービスが開始されます。

AMQP クライアントで使用中の IBM MQ オブジェクトの表示

AMQP クライアントによって使用されているさまざまな IBM MQ リソース (接続やサブスクリプションなど) を表示できます。

接続

AMQP サービスが開始されると、複数の新規 Hconn が作成され、キュー・マネージャーに接続されます。この Hconn のプールは、AMQP クライアントがメッセージをパブリッシュするときに使用されます。**DISPLAY CONN** コマンドを使用して、Hconn を表示することができます。 以下に例を示します。

```
DISPLAY CONN(*) TYPE(CONN) WHERE (APPLDESC LK 'IBM MQ Advanced Message Queuing Protocol*')
```

このコマンドにより、クライアント固有の Hconn も示されます。 クライアント ID 属性がブランクの Hconn は、プールで使用されている Hconn です。

AMQP クライアントが AMQP チャネルに接続すると、新しい Hconn がキュー・マネージャーに接続されます。 この Hconn は、AMQP クライアントが作成したサブスクリプションのメッセージを非同期で消費するために使用されます。 **DISPLAY CONN** コマンドを使用して、特定の AMQP クライアントによって使用される Hconn を表示できます。 以下に例を示します。

```
DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_abcd1234')
```

クライアントによって作成されたサブスクリプション

AMQP クライアントがトピックにサブスクライブすると、新しい IBM MQ サブスクリプションが作成されます。 サブスクリプション名には以下の情報が含まれます。

- クライアントの名前。 クライアントが共有サブスクリプションを結合した場合は、共有の名前が使用されます。
- クライアントがサブスクライブしたトピック・パターン。
- 接頭部。 接頭部は、クライアントが非共有サブスクリプションを作成した場合は `private`、クライアントが共有サブスクリプションを結合した場合は `share` です

特定の AMQP クライアントによって使用されているサブスクリプションを表示するには、 **DISPLAY SUB** コマンドを実行し、 `private` 接頭部でフィルタリングします。

```
DISPLAY SUB(':private:*')
```

複数のクライアントで使用中の共有サブスクリプションを表示するには、以下のように **DISPLAY SUB** コマンドを実行し、`share` 接頭部でフィルターに掛けます。

```
DISPLAY SUB(':share:*')
```

共用サブスクリプションは複数の AMQP クライアントで使用できるため、共用サブスクリプションからのメッセージを現在消費しているクライアントを表示することができます。 これを行うには、サブスクリプション・キューで現在ハンドルがオープンしている Hconn をリストします。 共有を現在使用しているクライアントを表示するには、以下のステップを実行します。

1. 共有サブスクリプションで宛先として使用しているキュー名を見つけます。 以下に例を示します。

```
DISPLAY SUB(':private:recv_e298452:public') DEST
   5 : DISPLAY SUB(':private:recv_e298452:public') DEST
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120514D312020202020202020707E0A565C2D0020)
   SUB(:private:recv_e298452:public)
   DEST(SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
```

2. **DISPLAY CONN** コマンドを実行して、そのキューでオープンしているハンドルを見つけます。

```
DISPLAY CONN(*) TYPE(HANDLE) WHERE  (OBJNAME
EQ SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
   21 : DISPLAY CONN(*) TYPE(HANDLE) WHERE(OBJNAME EQ
SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)

AMQ8276: Display Connection details.
   CONN(707E0A56642B0020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(HANDLE)

   OBJNAME(SYSTEM.BASE.TOPIC)      OBJTYPE(TOPIC)

   OBJNAME(SYSTEM.MANAGED.DURABLE.560A7E7020002961)
   OBJTYPE(QUEUE)
```

3. ハンドルごとに、ハンドルが開いている AMQP クライアント ID を表示します。

```
DISPLAY CONN(707E0A56642B0020) CLIENTID
   23 : DISPLAY CONN(707E0A56642B0020) CLIENTID

AMQ8276: Display Connection details.
   CONN(707E0A56642B0020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(CONN)
   CLIENTID(recv_8f02c9d)
DISPLAY CONN(707E0A565F290020) CLIENTID
   24 : DISPLAY CONN(707E0A565F290020) CLIENTID
AMQ8276: Display Connection details.
   CONN(707E0A565F290020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(CONN)
   CLIENTID(recv_86d8888)
```

AMQP クライアントの識別、許可、および認証

他の IBM MQ クライアント・アプリケーションと同様に、いくつかの方法で AMQP 接続を保護することができます。

以下のセキュリティー機能を利用して、IBM MQ への AMQP 接続を保護できます。

- チャネル認証レコード
- 接続認証
- チャネル MCA ユーザー構成
- IBM MQ 権限定義
- TLS 接続

セキュリティー上の観点で、接続の確立は以下の 2 つのステップで構成されます。

- 接続を継続するかどうかの判別
- 後に行われる権限検査でアプリケーションに付与される IBM MQ ID の特定

さまざまな IBM MQ 構成について、また AMQP クライアントが接続の作成を試みるときに実行されるステップについて、以下で概説します。 これらの IBM MQ 構成の中には、ここで説明するステップのすべてを必ずしも使用しないものがあります。 例えば、企業ファイアウォール内での接続に TLS を使用しない構成もあれば、TLS を使用するもののクライアント証明書を認証に使用しない構成もあります。 多くの環境では、カスタム・モジュールやカスタム JAAS モジュールを使用しません。

接続の確立

以下のステップでは、AMQP クライアントによって接続が確立されるときの処理について説明します。 これらのステップでは、接続を継続するかどうかを判別し、権限検査でアプリケーションに付与される IBM MQ ID を特定します。

1. クライアントが IBM MQ への TLS 接続を開き、証明書を提供すると、キュー・マネージャーはクライアント証明書の検証を試みます。
2. クライアントがユーザー名およびパスワードの資格情報を提供すると、AMQP SASL フレームをキュー・マネージャーが受け取り、MQ CONNAUTH 構成が検査されます。
3. MQ チャネル認証規則が検査されます (例えば、IP アドレスおよび TLS 証明書 DN が有効かどうかなど)
4. チャネル MCAUSER が表明されます (チャネル認証規則によってそのように判別されない場合を除く)。
5. JAAS モジュールが構成されている場合は、それが呼び出されます。
6. MQ CONNECT 権限検査が、結果として得られた MQ ユーザー ID に適用されます。
7. 付与された IBM MQ ID を使用して接続が確立されます。

メッセージのパブリッシュ

以下のステップでは、AMQP クライアントによってメッセージがパブリッシュされるときの処理について説明します。 これらのステップでは、接続を継続するかどうかを判別し、権限検査でアプリケーションに付与される IBM MQ ID を特定します。

1. AMQP リンク・アタッチ・フレームをキュー・マネージャーが受け取ります。 接続時に確立された MQ ユーザー ID について、指定されたトピック・ストリングに対する IBM MQ パブリッシュ権限が検査されます。
2. 指定されたトピック・ストリングにメッセージがパブリッシュされます。

トピック・パターンのサブスクライブ

以下のステップでは、AMQP クライアントによってトピック・パターンのサブスクライブがなされるときの処理について説明します。 これらのステップでは、接続を継続するかどうかを判別し、権限検査でアプリケーションに付与される IBM MQ ID を特定します。

1. AMQP リンク・アタッチ・フレームをキュー・マネージャーが受け取ります。 接続時に確立された MQ ユーザー ID について、指定されたトピック・パターンに対する IBM MQ サブスクライブ権限が検査されます。
2. サブスクリプションが作成されます。

AMQP クライアントの ID と許可

IBM MQ オブジェクトへのアクセスを許可するには、AMQP クライアント ID、AMQP ユーザー名、またはチャネルまたはチャネル認証規則で定義された共通クライアント ID を使用します。

管理者は、AMQP チャネルを定義または変更するときに、キュー・マネージャーの CONNAUTH 設定を構成するか、またはチャネル認証規則を定義して選択を行います。 ID は、IBM MQ トピックへのアクセスを許可するために使用されます。 この選択は、以下の項目に基づいて行われます。

1. チャネル USECLNTID 属性。
2. キュー・マネージャー CONNAUTH 規則の ADOPTCTX 属性。
3. チャネルで定義された MCAUSER 属性。
4. 一致するチャネル認証規則の USERSRC 属性。

問題の回避 : このプロセスで選択された ID は、その後、DISPLAY CHSTATUS (AMQP) コマンドなどで、クライアントの MCAUSER として参照されます。 必ずしも選択肢 (2) で言及されているチャネルの MCAUSER と同じ ID でなければならないわけではないことに注意してください。

IBM MQ の **setmqaut** コマンドを使用して、AMQP チャネルに関連付けられた ID が使用を許可されるオブジェクトおよびアクションを選択します。 例えば、以下のコマンドは、キュー・マネージャー QM1 の管理者によって提供されるチャネル ID AMQPClient を許可します。

```
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPClient -all +pub +sub
```

および

```
setmqaut -m QM1 -t qmgr -p AMQPClient -all +connect
```

パスワードを使用した AMQP クライアント認証

クライアント・パスワードを使用して AMQP クライアント・ユーザー名を認証します。 トピックのパブリッシュおよびサブスクライブをクライアントに許可するために使用した ID とは別の ID を使用して、クライアントを認証できます。

AMQP サービスでは、MQ CONNAUTH または JAAS を使用してクライアント・ユーザー名を認証できます。 これらのいずれかを構成した場合、クライアントで指定されたパスワードは、MQ CONNAUTH 構成または JAAS モジュールによって検証されます。

以下の手順では、ローカル OS ユーザーとパスワードに対して個々のユーザーを認証し、成功した場合は共通 ID `AMQPUser` を採用するためのステップの例を概説します。

1. IBM MQ 管理者は、 IBM MQ エクスプローラーを使用して、AMQP チャネルの MCAUSER ID を任意の名前 (`AMQPUser` など) に設定します。
2. IBM MQ 管理者は、 `AMQPUser` が任意のトピックにパブリッシュおよびサブスクライブすることを許可します。

   ```
   setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPUser -all +pub +sub +connect
   ```

3. IBM MQ 管理者は、クライアントから提供されたユーザー名とパスワードを検査するための IDPWOS CONNAUTH 規則を構成します。 CONNAUTH 規則では、CHCKCLNT(REQUIRED) および ADOPTCTX(NO) を設定する必要があります。

注 : チャネル認証規則を使用し、MCAUSER チャネル属性を特権のないユーザーに設定して、キュー・マネージャーへの接続をいっそう制御できるようにすることをお勧めします。

チャネルでのパブリケーションのプライバシー

AMQP チャネル間でいずれかの方向に送信される AMQP パブリケーションのプライバシーは、TLS を使用して、接続を介する送信を暗号化することにより保護されます。

AMQP チャネルに接続する AMQP クライアントは、TLS を使用して、対称鍵暗号方式でチャネル上を伝送されるパブリケーションのプライバシーを保護します。 エンドポイントは認証されないため、 チャネルの暗号化を単独で信用することはできません。 プライバシーの保護と、サーバー認証または相互認証とを結合します。

TLS を使用する代わりとして、IPsec などの数種類の仮想プライベートネットワーク (VPN) は、TCP/IP 接続のエンドポイントを認証します。 VPN は、ネットワーク上を流れる各 IP パケットを暗号化します。 そのような VPN 接続が確立されると、トラステッド・ネットワークが確立されます。 VPN ネットワークで TCP/IP を使用して、AMQP クライアントを AMQP チャネルに接続できます。

サーバーを認証しないで TLS 接続を暗号化すると、接続は中間者攻撃にさらされます。 交換する情報は盗聴に対しては保護されますが、その情報を交換している相手が誰であるかは分かりません。 ネットワークを制御しない限り、IP 伝送を傍受し、エンドポイントになりすましている誰かにさらされます。

匿名 TLS をサポートする Diffie-Hellman 鍵交換 CipherSpec を使用することにより、サーバーを認証しなくても暗号化された TLS 接続を作成することができます。 非公開署名されたサーバー証明書を交換しなくても、クライアントとサーバーの間で共有され、TLS 伝送を暗号化するために使用されるマスター・シークレットが確立されます。

匿名接続は安全ではないので、ほとんどの TLS 実装では、デフォルトでは匿名の CipherSpec は使用されません。 AMQP チャネルが TLS 接続を要求するクライアント要求を受け入れる場合、そのチャネルは、鍵ストアをパスフレーズで保護する必要があります。 デフォルトでは、TLS 実装は匿名の CipherSpec を使用しないので、クライアントが認証できる、非公開署名された証明書を鍵ストアに含める必要があります。

匿名の CipherSpec を使用する場合は、サーバーの鍵ストアが存在している必要がありますが、非公開署名された証明書が含まれている必要はありません。

暗号化された接続を確立するための別の方法は、クライアント側にあるトラスト・プロバイダーを独自の実装で置き換えることです。 この独自の実装のトラスト・プロバイダーはサーバー証明書を認証しないでしょうが、接続は暗号化されます。

TLS を使用した AMQP クライアントの構成

TLS を使用してネットワークを流れるデータを保護し、クライアントが接続するキュー・マネージャーの ID を認証するように AMQP クライアントを構成できます。

AMQP クライアントから AMQP チャネルへの接続に TLS を使用するには、キュー・マネージャーが TLS に構成されていることを確認する必要があります。 キュー・マネージャーでの TLS の構成には、キュー・マネージャーによって TLS 証明書が読み取られる鍵ストアの構成方法が説明されています。

鍵ストアとともにキュー・マネージャーが構成されている場合、クライアントが接続する AMQP チャネルで TLS 属性を構成する必要があります。 AMQP チャネルには、TLS 構成に関連した、以下の 4 つの属性があります。

SSLCAUTH
SSLCAUTH 属性は、ID を検証するために AMQP クライアントがクライアント証明書を提示することをキュー・マネージャーが必要とするかどうかを指定するために使用されます。

SSLCIPH
SSLCIPH 属性は、TLS フローでデータをエンコードするためにチャネルが使用する必要がある暗号を指定します。

V 9.4.0 IBM MQ 9.4.0 以降、AMQP チャネルは ANY* 汎用 CipherSpecs をサポートするようになりました。 CipherSpecs について詳しくは、CipherSpec の有効化を参照してください。

SSLPEER
SSLPEER 属性は、接続を許可する必要がある場合にクライアント証明書と合致しなければならない識別名 (DN) を指定するために使用します。

CERTLABL
CERTLABL は、キュー・マネージャーがクライアントに提供する必要がある証明書を指定します。 キュー・マネージャーの鍵ストアには、複数の証明書を含めることができます。 この属性を使用すると、このチャネルへの接続で使用する証明書を指定できます。 CERTLABL が指定されていない場合は、キュー・マネージャーの鍵リポジトリーにある、キュー・マネージャーの CERTLABL 属性に対応するラベルを持つ証明書が使用されます。

TLS 属性を指定して AMQP チャネルを構成したら、以下のコマンドを使用して AMQP サービスを再開する必要があります。

```
STOP SERVICE(SYSTEM.AMQP.SERVICE) START SERVICE(SYSTEM.AMQP.SERVICE)
```

AMQP クライアントは、TLS によって保護されている AMQP チャネルに接続すると、キュー・マネージャーによって提示される証明書の ID を検証します。 これを行うには、キュー・マネージャーの証明書を含むトラストストアを使用して AMQP クライアントを構成する必要があります。 これを行う手順は、使用している AMQP クライアントによって異なります。 さまざまな AMQP クライアントおよび API については、それぞれの AMQP クライアントの資料を参照してください。

関連資料
DEFINE CHANNEL (新規チャネルの定義)
Multiplatforms での STOP SERVICE (サービスの停止)
Multiplatforms での START SERVICE (サービスの開始)

キュー・マネージャーからの AMQP クライアントの切断

AMQP クライアントをキュー・マネージャーから切断する場合は、PURGE CHANNEL コマンドを実行するか、AMQP クライアントへの接続を停止します。

- **PURGE CHANNEL** コマンドを実行します。 以下に例を示します。

```
PURGE CHANNEL(MYAMQP) CLIENTID('recv_28dbb7e')
```

- あるいは、以下のステップを実行して、AMQP クライアントがクライアントを切断するために使用している接続を停止します。

  1. **DISPLAY CONN** コマンドを実行して、クライアントが使用している接続を見つけます。 以下に例を示します。

     ```
     DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_28dbb7e')
     ```

     コマンドの出力は以下のようになります。

     ```
     DISPLAY CONN(*)  TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
       40 : DISPLAY CONN(*) TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
     AMQ8276: Display  Connection details.
       CONN(707E0A565F2D0020)
       EXTCONN(414D5143514D31202020202020202020)
       TYPE(CONN)
       CLIENTID(recv_28dbb7e)
     ```

  2. 接続を停止します。 以下に例を示します。

     ```
     STOP CONN(707E0A565F2D0020)
     ```

マルチキャストの管理

この情報は、マルチキャスト・メッセージのサイズの削減、およびデータ変換の使用可能化などの IBM MQ Multicast 管理タスクについて学習するのに使用します。

マルチキャストの概要

この情報は、IBM MQ Multicast のトピックと通信情報オブジェクトの概要を知るのに使用します。

このタスクについて

IBM MQ Multicast メッセージングはネットワークを使用して、グループ・アドレスにトピックをマッピングすることによりメッセージを送達します。 以下のタスクを実行すると、必要な IP アドレスとポートがマルチキャスト・メッセージング用に正しく構成されているかどうか短時間でテストできます。

マルチキャスト用の COMMINFO オブジェクトの作成
通信情報 (COMMINFO) オブジェクトには、マルチキャスト伝送に関連付けられた属性が含まれます。 COMMINFO オブジェクト・パラメーターの詳細については、DEFINE COMMINFO を参照してください。

以下のコマンド行の例を使用して、マルチキャスト用の COMMINFO オブジェクトを定義してください。

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

MC1 は COMMINFO オブジェクトの名前、*group address* はグループ・マルチキャストの IP アドレスまたは DNS 名、*port number* は伝送に使用するポート (デフォルト値は 1414) です。

MC1 という名前の新しい COMMINFO オブジェクトが作成されます。この名前は、次の例で TOPIC オブジェクトを定義する際に指定しなければならない名前です。

マルチキャスト用の TOPIC オブジェクトの作成
トピックとは、パブリッシュ/サブスクライブ・メッセージでパブリッシュされる情報のサブジェクトのことで、トピックを定義するには TOPIC オブジェクトを作成します。 TOPIC オブジェクトには、マ

ルチキャストと併用できるかどうかを定義する 2 つのパラメーターがあります。 これらのパラメーターは、**COMMINFO** と **MCAST** です。

- **COMMINFO** パラメーターは、マルチキャスト通信情報オブジェクトの名前を指定します。 COMMINFO オブジェクト・パラメーターの詳細については、DEFINE COMMINFO を参照してください。
- **MCAST** パラメーターは、トピック・ツリー内のこの位置でマルチキャストを許容するかどうかを指定します。

以下のコマンド行の例を使用して、マルチキャスト用に TOPIC オブジェクトを定義してください。

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

ALLSPORTS という名前の新しい TOPIC オブジェクトが作成されます。 このオブジェクトにはトピック・ストリング *Sports* があり、このオブジェクトに関連した通信情報オブジェクトは *MC1* (前の例で COMMINFO オブジェクトの定義時に指定した名前) と呼ばれ、マルチキャストが使用可能になります。

マルチキャスト・パブリッシュ/サブスクライブのテスト

TOPIC オブジェクトおよび COMMINFO オブジェクトが作成された後、 amqspubc サンプルおよび amqssubc サンプルを使用して、それらのオブジェクトをテストすることができます。 これらのサンプルについて詳しくは、パブリッシュ/サブスクライブのサンプル・プログラムを参照してください。

1. 2 つのコマンド行ウィンドウを開きます。最初のコマンド行は amqspubc パブリッシュ・サンプル用で、 2 番目のコマンド行は amqssubc サブスクライブ・サンプル用です。
2. コマンド行 1 で以下のコマンドを入力します。

   ```
   amqspubc Sports QM1
   ```

   Sports は前述の例で定義した TOPIC オブジェクトのトピック・ストリングで、*QM1* はキュー・マネージャーの名前です。
3. コマンド行 2 で以下のコマンドを入力します。

   ```
   amqssubc Sports QM1
   ```

   Sports と *QM1* はステップ 366 ページの『2』 で使用した値と同じです。
4. コマンド行 1 で Hello world と入力します。 COMMINFO オブジェクトで指定されたポートと IP アドレスが正しく構成されている場合、指定されたアドレスからのパブリケーションをポートで listen している amqssubc サンプルは、コマンド行 2 で Hello world を出力します。

IBM MQ Multicast のトピック・トポロジー

この例を利用して、IBM MQ Multicast のトピック・トポロジーについて理解を深めてください。

IBM MQ Multicast サポートでは、総階層内の各サブツリーに独自のマルチキャスト・グループとデータ・ストリームがあることが必要です。

クラスフル・ネットワーク IP アドレス指定スキームには、マルチキャスト・アドレス用の指定アドレス・スペースがあります。 IP アドレスのマルチキャスト範囲全体は 224.0.0.0 から 239.255.255.255 までですが、これらのアドレスの一部は予約済みです。 予約済みのアドレスのリストについては、システム管理者にお問い合わせください。または詳細について、https://www.iana.org/assignments/multicast-addresses を参照してください。 239.0.0.0 から 239.255.255.255 までの、ローカル側で有効範囲が設定されたマルチキャスト・アドレスを使用することをお勧めします。

以下の図には、2 つの可能なマルチキャスト・データ・ストリームがあります。

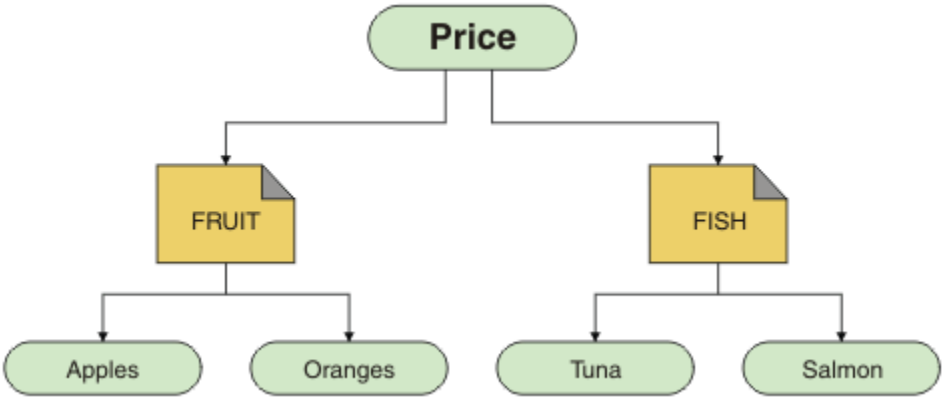
```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)
```

```
DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

239.XXX.XXX.XXX および *239.YYY.YYY.YYY* は有効なマルチキャスト・アドレスです。

これらのトピック定義は、次の図に示すトピック・ツリーを作成するために使用されます。

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)
DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



それぞれのマルチキャスト通信情報 (COMMINFO) オブジェクトは、そのグループ・アドレスが異なっているので、それぞれ異なるデータ・ストリームを表しています。 この例では、FRUIT トピックは COMMINFO オブジェクト MC1 を使用するように定義され、FISH トピックは COMMINFO オブジェクト MC2 を使用するように定義され、Price ノードにはマルチキャスト定義がありません。

IBM MQ Multicast には、トピック・ストリングを 255 文字までとする長さ制限があります。 この制限は、ツリー内のノードおよびリーフ・ノードの名前を使用して注意する必要があることを意味します。ノードおよびリーフ・ノードの名前が長すぎる場合、トピック・ストリングは 255 文字を超える可能性があり、2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR 理由コードが返される可能性があります。 トピック・ストリングが長いとパフォーマンスに不利な影響が及ぶ可能性があるため、トピック・ストリングはなるべく短くすることをお勧めします。

マルチキャスト・メッセージのサイズの制御

この情報は、IBM MQ メッセージ形式について学習したり、IBM MQ メッセージのサイズを小さくしたりするのに使用します。

IBM MQ メッセージには、多数の属性が関連付けられており、それらの属性はメッセージ記述子に含まれています。 小さなメッセージの場合、これらの属性がデータ・トラフィックのほとんどを占めてしまうことがあり、伝送速度に多大な悪影響を及ぼすおそれがあります。 IBM MQ Multicast では、これらの属性の内のどれをメッセージと共に送信するかをユーザーが構成できます。

トピック・ストリング以外のメッセージ属性があるかどうかは、COMMINFO オブジェクト状態でそれらの属性が送信されることになっているかどうかによります。 属性が送信されない場合、受信側のアプリケーションはデフォルト値を適用します。 デフォルトの MQMD 値は、必ずしも MQMD_DEFAULT 値と同じではなく、 368 ページの表 19 で説明されています。

COMMINFO オブジェクトには MCPROP 属性が含まれており、メッセージと共に流れる MQMD フィールドとユーザー・プロパティーの数を制御します。 以下のように、この属性の値を適切なレベルに設定すると、IBM MQ Multicast メッセージのサイズを制御できます。

MCPROP
このマルチキャスト・プロパティーの値では、メッセージと一緒に流れる MQMD プロパティーとユーザー・プロパティーの数を制御します。

ALL
すべてのユーザー・プロパティーと MQMD のすべてのフィールドが送信されます。

REPLY
ユーザー・プロパティーと、メッセージへの応答に関連する MQMD フィールドだけを送信します。以下のプロパティーが該当します。

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

USER
ユーザー・プロパティーのみが送信されます。

NONE
ユーザー・プロパティーも MQMD フィールドも送信されません。

COMPAT
この値を指定すると、互換モードで RMM へのメッセージの伝送が行われ、現行の XMS アプリケーションおよび IBM Integration Bus RMM アプリケーションとの相互協調処理の一部を行うことができるようになります。

Removed XMS .NET マルチキャストメッセージング（RMM）は廃止されました IBM MQ 9.2 そして削除されました IBM MQ 9.3。

マルチキャスト・メッセージの属性

メッセージ属性は、MQMD、MQRFH2 内のフィールド、メッセージ・プロパティーなど、さまざまな場所からのものである可能性があります。

以下の表は、MCPROP (このセクションで前述) の値に従ってメッセージが送信される場合に行われる操作と、属性が送信されない場合に使用されるデフォルト値を示しています。

表 *19.* メッセージング属性と、マルチキャストとの関係

属性	マルチキャスト使用時のアクション	伝送されない場合のデフォルト
TopicString	常に含まれる	適用外
MQMQ StrucId	伝送されない	適用外
MQMD Version	伝送されない	適用外
レポート	デフォルト以外の場合に含まれる	0
MsgType	デフォルト以外の場合に含まれる	MQMT_DATAGRAM
Expiry	デフォルト以外の場合に含まれる	0
Feedback	デフォルト以外の場合に含まれる	0
Encoding	デフォルト以外の場合に含まれる	MQENC_NORMAL(equiv)
CodedCharSetId	デフォルト以外の場合に含まれる	1208
Format	デフォルト以外の場合に含まれる	MQRFH2
優先順位	デフォルト以外の場合に含まれる	4
Persistence	デフォルト以外の場合に含まれる	MQPER_NOT_PERSISTENT
MsgId	デフォルト以外の場合に含まれる	NULL
CorrelId	デフォルト以外の場合に含まれる	NULL
BackoutCount	デフォルト以外の場合に含まれる	0

表 19. メッセージング属性と、マルチキャストとの関係 (続き)		
属性	**マルチキャスト使用時のアクション**	**伝送されない場合のデフォルト**
ReplyToQ	デフォルト以外の場合に含まれる	ブランク
ReplyToQMgr	デフォルト以外の場合に含まれる	ブランク
UserIdentifier	デフォルト以外の場合に含まれる	ブランク
AccountingToken	デフォルト以外の場合に含まれる	NULL
PutAppIType	デフォルト以外の場合に含まれる	MQAT_JAVA
PutAppIName	デフォルト以外の場合に含まれる	ブランク
PutDate	デフォルト以外の場合に含まれる	ブランク
PutTime	デフォルト以外の場合に含まれる	ブランク
ApplOriginData	デフォルト以外の場合に含まれる	ブランク
GroupID	除外	適用外
MsgSeqNumber	除外	適用外
オフセット	除外	適用外
MsgFlags	除外	適用外
OriginalLength	除外	適用外
UserProperties	含まれる	適用外

関連資料

Multi ALTER COMMINFO

DEFINE COMMINFO

Multicast メッセージングに関するデータ変換を使用可能にする

この情報は、IBM MQ Multicast メッセージングで、データ変換が行われる方法について理解するために役立ちます。

IBM MQ Multicast はコネクションレスの共有プロトコルであるため、各クライアントがデータ変換に関する特定の要求を行うことはできません。 同じマルチキャスト・ストリームにサブスクライブしているクライアントはすべて同じバイナリー・データを受け取ります。したがって、IBM MQ データ変換が必要な場合は、変換は各クライアントでローカルに実行されます。

プラットフォームが混用されているインストール済み環境では、ほとんどのクライアントで、送信元のアプリケーションのネイティブ・フォーマットではないフォーマットのデータが必要とされる可能性があります。 この状態の場合、効率化を図るために、マルチキャスト COMMINFO オブジェクトの **CCSID** 値と **ENCODING** 値を使用して、メッセージ伝送のエンコードを定義できます。

IBM MQ Multicast は、以下の組み込み形式のメッセージ・ペイロードのデータ変換をサポートします。

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

これらの形式に加えて、独自の形式を定義し、MQDXP - データ変換出口パラメーターのデータ変換出口を使用することもできます。

データ変換のプログラミングについては、マルチキャスト・メッセージング用の MQI でのデータ変換を参照してください。

データ変換の詳細については、データ変換を参照してください。

データ変換出口および ClientExitPath の詳細については、クライアント構成ファイルの ClientExitPath スタンザを参照してください。

マルチキャスト・アプリケーションのモニター

この情報は IBM MQ Multicast の管理とモニターについて学習するのに使用します。

マルチキャスト・トラフィックに関する現行のパブリッシャーとサブスクライバーの状況 (送受信されたメッセージの数や失われたメッセージの数など) は、クライアントからサーバーに定期的に伝送されます。 状況の受信時に、COMMINFO オブジェクトの COMMEV 属性は、キュー・マネージャーがイベント・メッセージを SYSTEM.ADMIN.PUBSUB.EVENT に書き込むかどうかを指定します。 イベント・メッセージには、受け取った状況情報が含まれます。 この情報は、問題の原因を調べるうえで、非常に貴重な補助的な診断情報になります。

MQSC コマンド **DISPLAY CONN** は、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関する接続情報を表示するために使用します。 **DISPLAY CONN** コマンドについて詳しくは、DISPLAY CONN を参照してください。

MQSC コマンド **DISPLAY TPSTATUS** は、パブリッシャーとサブスクライバーの状況を表示するために使用されます。 **DISPLAY TPSTATUS** コマンドについて詳しくは、DISPLAY TPSTATUS を参照してください。

COMMEV とマルチキャスト・メッセージ信頼性標識

信頼性標識 は、COMMINFO オブジェクトの **COMMEV** 属性と併用され、IBM MQ Multicast のパブリッシャーとサブスクライバーをモニターするうえで鍵となる要素です。 信頼性標識 (パブリッシュまたはサブスクライブ状況コマンドで返される **MSGREL** フィールド) は、エラーにならなかった伝送のパーセンテージを示す IBM MQ 標識です。伝送エラーのためにメッセージを再送する必要が生じることがあります。この状況は **MSGREL** の値に反映されます。 伝送エラーの原因としては、低速のサブスクライバー、過密なネットワーク、ネットワークの障害などの可能性があります。 **COMMEV** は、COMMINFO オブジェクトを使用して作成されるマルチキャスト・ハンドルに関するイベント・メッセージを生成するかどうかを制御し、以下の 3 つの有効値のいずれかに設定されます。

DISABLED
イベント・メッセージは書き込まれません。

ENABLED
COMMINFO **MONINT** パラメーターで定義されている頻度で、常にイベント・メッセージが書き込まれます。

EXCEPTION
メッセージ信頼性が信頼性しきい値を下回ると、イベント・メッセージが書き込まれます。 メッセージ信頼性のレベルが 90% 以下であることは、ネットワーク構成に問題があるか、または 1 つ以上のパブリッシュ/サブスクライブ・アプリケーションの実行速度が遅すぎる可能性があることを示します。

- 値 **MSGREL(100,100)** であれば、短期または長期のいずれの時間フレームでも問題がないことが分かります。
- 値 **MSGREL(80,60)** であれば、現在 20% のメッセージに問題があるが、長期の値 60 からは改善していることが分かります。

クライアントは、キュー・マネージャーへのユニキャスト接続が切断された場合もマルチキャスト・トラフィックの送受信を続行できることがあるため、データは古くなっている可能性もあります。

マルチキャスト・メッセージの信頼性

この情報は、IBM MQ Multicast のサブスクリプションとメッセージのヒストリーを設定する方法について学習するのに使用します。

マルチキャスト使用時の伝送の失敗を解決するうえで鍵となる要素は、IBM MQ による送信データのバッファリング (リンクの伝送側に保持されるメッセージのヒストリー) です。 このプロセスは、IBM MQ によって信頼性が提供されるので、書き込み側のアプリケーション・プロセスでメッセージのバッファリングが必要ないことを意味します。 後述するように、このヒストリーのサイズは、通信情報 (COMMINFO) オブジェクトによって構成されます。 伝送バッファーが大きいほど、必要に応じて再送されるヒストリーが増えることを意味しますが、マルチキャストの性質上、100% 保証された送達はサポートできません。

IBM MQ Multicast のメッセージ・ヒストリーは、通信情報 (COMMINFO) オブジェクト内で **MSGHIST** 属性によって制御されます。

MSGHIST
この値は、システムが NACK (否定応答) の場合の再送信を処理するために保持しておくメッセージ・ヒストリーの量 (キロバイト単位) です。

値が `0` の場合は、信頼性のレベルが最も低くなります。 デフォルト値は `100` KB です。

IBM MQ Multicast の新しいサブスクリプション・ヒストリーは、通信情報 (COMMINFO) オブジェクト内の **NSUBHIST** 属性によって制御されます。

NSUBHIST
この新規サブスクライバー・ヒストリーの値では、パブリケーション・ストリームに加わるサブスクライバーが現時点で入手できる限りの量のデータを受け取るのか、それともサブスクリプションの時点以降に実行されたパブリケーションだけを受け取るのかを制御します。

NONE
値が `NONE` の場合、送信側は、サブスクリプションの時点から作成されたパブリケーションのみを送信します。 `NONE` はデフォルト値です。

ALL
値 `ALL` を指定すると、送信側はトピックの既知の履歴を再送します。 状況によっては、この状態は、保存パブリケーションに対する動作が類似することがあります。

注 : `ALL` の値を使用すると、すべてのトピック・ヒストリーが再送されるため、大規模なトピック・ヒストリーがある場合にパフォーマンスに悪影響を及ぼす可能性があります。

関連資料
DEFINE COMMINFO
Multi ALTER COMMINFO

拡張マルチキャスト・タスク

この情報を使用して、 `.ini` ファイルの構成、および IBM MQ LLM とのインターオペラビリティーなどの、高度な IBM MQ Multicast 管理タスクについて学習します。

Multicast インストール済み環境でのセキュリティーの考慮事項については、マルチキャストのセキュリティーを参照してください。

マルチキャストと非マルチキャストのパブリッシュ/サブスクライブ・ドメイン間のブリッジング

この情報を使用して、非マルチキャスト・パブリッシャーが IBM MQ マルチキャスト対応トピックをパブリッシュした場合に何が行われるかを理解します。

非マルチキャスト・パブリッシャーが、**MCAST** および **BRIDGE** が有効であると定義されたトピックをパブリッシュする場合、キュー・マネージャーは、listen している可能性がある任意のサブスクライバーに、マルチキャストを通じてメッセージを直接送信します。 マルチキャスト・パブリッシャーは、マルチキャストが有効になっていないトピックをパブリッシュできません。

既存のトピックでは、トピック・オブジェクトの **MCAST** パラメーターおよび **COMMINFO** パラメーターを設定することによってマルチキャストを有効にできます。 これらのパラメーターの詳細については、 初期マルチキャストの概念 を参照

COMMINFO オブジェクトの **BRIDGE** 属性は、マルチキャストを使用していないアプリケーションからのパブリケーションを制御します。 **BRIDGE** が ENABLED に設定され、トピックの **MCAST** パラメーターも ENABLED に設定されている場合、マルチキャストを使用していないアプリケーションからのパブリケーションは、マルチキャストを使用しているアプリケーションにブリッジされます。 **BRIDGE** パラメーターの詳細については、DEFINE COMMINFO を参照してください。

マルチキャスト用に .ini ファイルを構成する

この情報を使用して、 .ini ファイル内の IBM MQ Multicast フィールドを理解します。

追加の IBM MQ マルチキャスト構成は、ini ファイルで行うことができます。 使用しなければならない特定の ini ファイルは、アプリケーションのタイプによって異なります。

- クライアント: *MQ_DATA_PATH* /mqclient.ini ファイルを構成します。
- キュー・マネージャー: *MQ_DATA_PATH* /qmgrs/*QMNAME*/qm.ini ファイルを構成します。

ここで、 *MQ_DATA_PATH* は IBM MQ データ・ディレクトリーの場所 (/var/mqm/mqclient.ini) であり、 *QMNAME* は .ini ファイルが適用されるキュー・マネージャーの名前です。

.ini ファイルには、 IBM MQ Multicast の動作を微調整するために使用されるフィールドが含まれてい 。

```
Multicast:
Protocol       = IP | UDP
IPVersion       = IPv4 | IPv6 | ANY | BOTH
LimitTransRate    = DISABLED | STATIC | DYNAMIC
TransRateLimit    = 100000
SocketTTL       = 1
Batch         = NO
Loop         = 1
Interface       = <IPaddress>
FeedbackMode     = ACK | NACK | WAIT1
HeartbeatTimeout   = 20000
HeartbeatInterval   = 2000
```

プロトコル

UDP
このモードでは、UDP プロトコルを使用してパケットが送信されます。 しかし、ネットワーク要素は、IP モードでは行っているマルチキャスト配布の支援を行うことができません。 パケットの形式は PGM との互換性が保たれます。 これがデフォルト値です。

IP
このモードでは、送信側は未加工の IP パケットを送信します。 PGM サポートのあるネットワーク要素は、信頼性の高いマルチキャスト・パケット配布を支援します。 このモードは、PGM 規格との完全な互換性があります。

IPVersion

IPv4
IPv4 プロトコルのみを使用して通信します。 これがデフォルト値です。

IPv6
IPv6 プロトコルのみを使用して通信します。

ANY
使用可能なプロトコルに応じて、IPv4 と IPv6 のいずれかまたは両方を使用して通信します。

BOTH
IPv4 と IPv6 の両方を使用する通信をサポートします。

LimitTransRate

DISABLED
伝送速度の制御はありません。 これがデフォルト値です。

STATIC
静的な伝送速度の制御を実装します。 送信側は、TransRateLimit パラメーターで指定された値を超える速度では伝送しません。

DYNAMIC
送信側は、受信側から取得するフィードバックに従って、伝送速度を適合させます。 この場合、伝送速度の制限は TransRateLimit パラメーターで指定された値を超えることはできません。 送信側は最適な伝送速度に達しようとします。

TransRateLimit
Kbps 単位の伝送速度の制限。

SocketTTL
SocketTTL の値は、マルチキャスト・トラフィックがルーターを通過できるかどうか、または通過できるルーターの数を判別します。

バッチ
メッセージをバッチ形式にするか、それとも即時に送信されるかを制御します。以下の 2 つの有効値があります。

- *NO*。メッセージはバッチ形式ではなく、即時に送信されます。
- *YES*。メッセージはバッチ形式になります。

Loop
この値を 1 に設定すると、マルチキャスト・ループが使用可能になります。 マルチキャスト・ループは、送信されるデータがホストにループバックされるかどうかを定義します。

インターフェース
マルチキャスト・トラフィックが流れるインターフェースの IP アドレス。 詳細およびトラブルシューティングについて詳しくは、非マルチキャスト・ネットワークでのマルチキャスト・アプリケーションのテストおよびマルチキャスト・トラフィック用の適切なネットワークの設定を参照してください。

FeedbackMode

NACK
否定応答によるフィードバック。 これがデフォルト値です。

ACK
肯定応答によるフィードバック。

WAIT1
肯定応答によるフィードバックで、送信側はいずれかの受信側からの ACK を 1 つだけ待ちます。

HeartbeatTimeout
ハートビートのタイムアウト (ミリ秒)。 0 の値は、トピックの 1 つ以上の受信側でハートビート・タイムアウト・イベントが発生しないことを示します。 デフォルト値は 20000 です。

HeartbeatInterval
ハートビート間隔 (ミリ秒)。 0 の値は、ハートビートが送信されないことを示します。 ハートビート間隔は、偽のハートビート・タイムアウト・イベントを回避するために、 **HeartbeatTimeout** 値よりもかなり小さくする必要があります。 デフォルト値は 2000 です。

IBM MQ Low Latency Messaging とのマルチキャスト相互運用性

この情報を利用して、IBM MQ Multicast と IBM MQ Low Latency Messaging (LLM) との間の相互運用性に関する理解を深めてください。

LLM を使用するアプリケーションと、マルチキャストを使用する別のアプリケーションとの間の両方向のメッセージ交換に、基本的なペイロード転送が可能です。 マルチキャストは LLM テクノロジーを使用しますが、LLM 製品自体は組み込まれていません。 したがって、LLM と IBM MQ Multicast を両方ともインストールし、2 つの製品を個別に操作したり保守したりできます。

マルチキャストと通信する LLM アプリケーションが、メッセージ・プロパティーを送受信する必要が生じることがあります。 以下の表のように、IBM MQ メッセージ・プロパティーと MQMD フィールドは、特定の LLM メッセージ・プロパティー・コードのある LLM メッセージ・プロパティーとして伝送されます。

表 *20. IBM MQ* メッセージ・プロパティーから *IBM MQ LLM* プロパティーへのマッピング

IBM MQ プロパティー	**IBM MQ LLM プロパティー・タイプ**	**LLM プロパティーの種類**	**LLM プロパティー・コード**
MQMD.Report	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1001
MQMD.MsgType	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1002
MQMD.Expiry	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1003
MQMD.Feedback	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1004
MQMD.Encoding	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1005
MQMD.CodedCharSetId	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1006
MQMD.Format	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1007
MQMD.Priority	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1008
MQMD.Persistence	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1009
MQMD.MsgId	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_ByteArray	-1010
MQMD.BackoutCount	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1012
MQMD.ReplyToQ	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1013
MQMD.ReplyToQMger	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1014
MQMD.PutDate	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1020
MQMD.PutTime	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1021
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

LLM について詳しくは、LLM 製品資料: IBM MQ Low Latency Messaging を参照してください。

IBM i IBM MQ for IBM i の管理

CL コマンドは、 IBM i 上の IBM MQ を管理するための推奨される方法です。 MQSC コマンド、PCF コマンド、制御コマンド、およびリモート管理を使用することもできます。

このタスクについて

管理タスクには、クラスター、プロセス、および IBM MQ オブジェクト (キュー・マネージャー、キュー、名前リスト、プロセス定義、チャネル、クライアント接続チャネル、リスナー、サービス、および認証情報オブジェクト) の作成、始動、変更、表示、停止、および削除があります。

IBM MQ for IBM i の管理方法について詳しくは、以下のリンクを参照してください。

関連概念

IBM MQ for IBM i キュー・マネージャー・ライブラリー名についての理解

IBM i でのインストール可能サービスとコンポーネント

関連タスク

Multiplatforms での IBM MQ 構成情報の変更

IBM i でのセキュリティーのセットアップ

165 ページの『IBM i での送達不能キュー・ハンドラーの呼び出し』

IBM MQ for IBM i では、 **STRMQMDLQ** コマンドを設定して DLQ ハンドラーを呼び出します。

IBM MQ for IBM i アプリケーションの問題判別

関連資料

システムおよびデフォルト・オブジェクト

IBM i CL コマンドを使用した IBM MQ for IBM i の管理

IBM MQ IBM i のコマンドを理解するために使用します。

キュー・マネージャー、キュー、トピック、チャネル、名前リスト、プロセス定義、および認証情報オブジェクトに関連するものを含め、多くのグループの IBM MQ コマンドには、関連する **WRK*** コマンドを使用してアクセスできます。

このセットの基本コマンドは、**WRKMQM** です。 このコマンドを使用すると、例えばシステム上のすべてのキュー・マネージャーのリストを、状況の情報と共に表示できます。 別の方法として、エントリーごとに各種オプションを使用して、キュー・マネージャー固有のすべてのコマンドを処理することもできます。

例えば、チャネル、トピック、またはキューを処理しながら、**WRKMQM** コマンドから各キュー・マネージャー固有の領域を選択して、そこからオブジェクトを個別に選択することができます。

IBM MQ アプリケーション定義の記録

IBM MQ アプリケーションを作成またはカスタマイズする際に、作成したすべての IBM MQ 定義の記録をとっておくと役立ちます。 この記録は以下に使用できます。

- 回復目的
- 保守
- IBM MQ アプリケーションのロールアウト

IBM MQ アプリケーション定義を、次の 2 つの方法のどちらかで記録できます。

1. 制御言語プログラムを作成して、サーバー用に IBM MQ 定義を生成する。
2. クロスプラットフォーム IBM MQ コマンド言語を使用して SRC メンバーとしての MQSC テキスト・ファイルを作成し、IBM MQ 定義を生成する。

キュー・オブジェクトの定義の詳細については、12 ページの『MQSC コマンドを使用した IBM MQ の管理』および 26 ページの『IBM MQ プログラマブル・コマンド・フォーマットの使用』を参照してください。

関連資料

IBM MQ for IBM i CL コマンドのリファレンス

IBM i CL コマンドを使用した IBM MQ for IBM i の使用を開始する前に

この情報を使用して、IBM MQ サブシステムを開始し、ローカル・キュー・マネージャーを作成します。

始める前に

IBM MQ サブシステムが稼働していることを (STRSBS QMQM/QMQM コマンドを使用して) 確認し、そのサブシステムに関連付けられているジョブ・キューが保留状態でないことを確認します。 デフォルトでは、IBM MQ サブシステムおよびジョブ・キューはどちらも、QMQM という名前でライブラリー QMQM にあります。

このタスクについて

IBM i コマンド行を使用したキュー・マネージャーの開始

手順

1. IBM i コマンド行から CRTMQM コマンドを発行して、ローカル・キュー・マネージャーを作成します。

キュー・マネージャーを作成する場合、そのキュー・マネージャーをデフォルトのキュー・マネージャーにするかどうかを任意に選択できます。 デフォルトのキュー・マネージャー (1 つのみ選択可能) は、キュー・マネージャー名パラメーター (MQMNAME) が省略されている場合は、CL コマンドが適用されるキュー・マネージャーです。

2. IBM i コマンド行から STRMQM コマンドを発行して、ローカル・キュー・マネージャーを開始します。

   キュー・マネージャーの開始に数秒より長くかかる場合、IBM MQ は開始状況の詳細を示す状況メッセージを断続的に表示します。 これらのメッセージの詳細については、メッセージおよび理由コードを参照

次のタスク

IBM i コマンド行から ENDMQM コマンドを発行してキュー・マネージャーを停止したり、 IBM i コマンド行から他の IBM MQ コマンドを発行してキュー・マネージャーを制御したりすることができます。

リモート・キュー・マネージャーはリモートで開始することはできません。 したがって、システム内でローカル・オペレーターが作成し開始する必要があります。 ただし、リモート操作を可能にするリモート操作機能が (IBM MQ for IBM i の外部に) 存在する場合は例外です。

ローカル・キュー管理者は、リモート・キュー・マネージャーを停止することはできません。

注 : IBM MQ システム静止の一環として、アクティブなキュー・マネージャーを静止させる必要があります。 これについては、445 ページの『IBM MQ for IBM i の静止』で説明されています。

IBM i IBM MQ for IBM i オブジェクトの作成

ここでは、 IBM i 用の IBM MQ オブジェクトを作成する方法について説明します。

始める前に

以下の作業は、コマンド・ラインから IBM MQ for IBM i を使用するさまざまな方法を示しています。

このタスクについて

オンラインで IBM MQ オブジェクトを作成する方法には、次の 2 つがあります。

手順

1. 作成コマンドを使用する。 例えば、**Create MQM Queue** コマンド: **CRTMQMQ**
2. MQM オブジェクト処理コマンドを使用し、その後に F6 を続ける (例: **Work with MQM Queues** コマンド: **WRKMQMQ**)

次のタスク

全コマンドのリストについては、 IBM MQ for IBM i CL コマンドを参照してください。

注 : MQM コマンドは、すべて「メッセージ・キュー・マネージャー・コマンド」メニューから実行依頼できます。 このメニューを表示するには、コマンド行に GO CMDMQM と入力してから、Enter キーを押します。

このメニューからコマンドを選択すると、プロンプト・パネルがシステムによって自動的に表示されます。 コマンド行から直接入力したコマンド用のプロンプト・パネルを表示するには、F4 キーを押してから Enter キーを押してください。

CRTMQMQ コマンドを使用するローカル・キューの作成

手順

1. コマンド行で CHGMQM と入力し、F4 キーを押します。

2.「**MQM キューの作成**」パネルで、作成するキューの名前を Queue name フィールドに入力します。 大文字と小文字が混合している名前を指定する場合は、名前をアポストロフィで囲んでください。
3. Queue type フィールドに *LCL と入力します。
4. デフォルトのキュー・マネージャーを使用しない場合は、キュー・マネージャー名を指定して、Enter キーを押します。 新しい値を使用して任意の値を上書きすることができます。 さらにフィールドを表示するためには、下方にスクロールします。 クラスターに使用するオプションは、オプションのリストの最後にあります。
5. 値の変更が終わったら、Enter キーを押して、新しいキューを作成します。

WRKMQMQ コマンドを使用するローカル・キューの作成

手順

1. コマンド・ラインに WRKMQMQ と入力します。
2. キュー・マネージャーの名前を入力します。
3. プロンプト・パネルを表示するには、F4 キーを押します。 プロンプト・パネルは、総称キュー名またはキュー・タイプを指定して、表示されるキューの数を減らすのに便利です。
4. Enter を押すと、「**MQM キューの処理**」**パネル** が表示されます。 これらの値に新しい値を上書き入力できます。 さらにフィールドを表示するためには、下方にスクロールします。 クラスターに使用するオプションは、オプションのリストの最後にあります。
5. 新しいキューを作成するために F6 キーを押します。 「**CRTMQMQ**」パネルが表示されます。 キューの作成手順については、376 ページの『CRTMQMQ コマンドを使用するローカル・キューの作成』を参照してください。 キューが作成されると、「**MQM キューの処理**」**パネル**が再び表示されます。 F5=Refresh キーを押すと、新しいキューがリストに追加されます。

キュー・マネージャーの属性の変更

このタスクについて

CHGMQM コマンドに指定されたキュー・マネージャーの属性を変更するには、変更したい属性および値を指定します。 例えば、jupiter.queue.manager の属性を変更するには、次のオプションを使用します。

手順

コマンド行で **CHGMQM** と入力し、F4 キーを押します。

タスクの結果

このコマンドにより、使用されている送達不能キューが変更され、禁止イベントが使用可能になります。

IBM i ローカル・キューの操作 (IBM i)

このセクションでは、ローカル・キューを管理するために使用できるコマンドの例をいくつか示します。 ここに示すコマンドは、すべて **WRKMQMQ コマンド・パネル**のオプションで使用することもできます。

ローカル・キューの定義

アプリケーションにとって、ローカル・キュー・マネージャーとは、アプリケーションが接続されているキュー・マネージャーです。 ローカル・キュー・マネージャーによって管理されるキューは、そのキュー・マネージャーに対してローカルであるといいます。

ローカル・キューの定義を作成するため、またキューと呼ばれるデータ構造を作成するためには、コマンド **CRTMQMQ QTYPE *LCL** を使用します。 デフォルト・ローカル・キューの特性からのキュー特性を修正することもできます。

この例では、定義するキュー orange.local.queue は、次のような特性を持つものとして指定します。

- 読み取りは可能、書き込みは不可、先入れ先出し法 (FIFO) で操作が行われる。
- 「通常の」キュー。つまり、開始キューや伝送キューではなく、トリガー・メッセージを生成しない。
- キューの最大サイズは、1000 個のメッセージで、最大メッセージ長は、2000 バイトである。

以下のコマンドは、これをデフォルトのキュー・マネージャーに対して実行します。

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL)
TEXT('Queue for messages from other systems')
PUTENBL(*NO)
GETENBL(*YES)
TRGENBL(*NO)
MSGDLYSEQ(*FIFO)
MAXDEPTH(1000)
MAXMSGLEN(2000)
USAGE(*NORMAL)
```

注 :

1. USAGE *NORMAL は、このキューが伝送キューではないことを示します。
2. 同じキュー・マネージャーに名前が orange.local.queue であるローカル・キューが既にある場合、このコマンドは失敗します。 既存のキューの定義を上書きする場合には、REPLACE *YES 属性を使用してください。ただし、379 ページの『ローカル・キュー属性の変更』も参照してください。

送達不能キューの定義

正しい宛先に送達できないメッセージを後で取り出すために保管することができるよう、各キュー・マネージャーは、送達不能キューとして使用されるローカル・キューを持っている必要があります。 送達不能キューについては、キュー・マネージャーに明示的に通知する必要があります。 このことは、送達不能キューを **CRTMQM** コマンドに指定することにより行えます。 あるいは **CHGMQM** コマンドを使用して後でそれを指定することができます。 送達不能キューを使用するためには、その前にそれを定義しておくことも必要です。

SYSTEM.DEAD.LETTER.QUEUE という名前のサンプル送達不能キューが、製品と共に提供されています。このキューは、キュー・マネージャーを作成すると、自動的に作成されます。 必要ならば、この定義を修正できます。 その名前を変更する必要はありません。 ただし、必要なら変更することもできます。

送達不能キューには、以下に示すものを除いて、特別な要件はありません。

- ローカル・キューでなければならない。
- その MAXMSGL (最大メッセージ長) 属性は、キュー・マネージャーが取り扱う最大メッセージ**および**送達不能ヘッダー (MQDLH) のサイズをキューに収容できるようにしておく必要があります。

IBM MQ には送達不能キュー・ハンドラーが用意されています。これを使用して、送達不能キュー上で見つかったメッセージの処理方法または除去方法を指定できます。 詳しくは、165 ページの『IBM i での送達不能キュー・ハンドラーの呼び出し』を参照してください。

デフォルト・オブジェクト属性の表示

IBM MQ オブジェクトを定義する際に、指定していない属性はデフォルト・オブジェクトから得られます。例えば、ローカル・キューを定義すると、このキューは、定義の中で省略された属性を、SYSTEM.DEFAULT.LOCAL.QUEUE と呼ばれるデフォルト・ローカル・キューから継承します。 これらの属性を正確に知りたい場合には、次のコマンドを使用します。

```
DSPMQMQ QNAME(SYSTEM.DEFAULT.LOCAL.QUEUE) MQMNAME(MYQUEUEMANAGER)
```

ローカル・キュー定義のコピー

CPYMQMQ コマンドを使用すると、キュー定義をコピーできます。 以下に例を示します。

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

このコマンドにより、システム・デフォルト・ローカル・キューの属性ではなく、コピー元のキュー orange.local.queue と同じ属性を持つキューが作成されます。

CPYMQMQ コマンドを使用して、キュー定義をコピーし、元のキューの属性を変更したものを 1 つ以上代わりに使用することもできます。 以下に例を示します。

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('third.queue') MQMNAME(MYQUEUEMANAGER)
MAXMSGLEN(1024)
```

このコマンドにより、キュー orange.local.queue の属性がキュー third.queue にコピーされ、新しいキューの最大メッセージ長は、2000 バイトではなく、1024 バイトになるように指定されます。

注 : CPYMQMQ コマンドを使用した場合、キューにあるメッセージではなく、キューの属性だけをコピーします。

ローカル・キュー属性の変更

キューの属性は 2 とおりの方法で変更できます。 つまり、**CHGMQMQ** コマンドを使用するか、あるいは **CPYMQMQ** コマンドに REPLACE *YES 属性を指定して使用するかです。 377 ページの『ローカル・キューの定義』で、キュー orange.local.queue を定義しました。 ここで、例えばこのキューの最大メッセージ長を 10,000 バイトに増やす必要があるとします。

- **CHGMQMQ** コマンドを使用する場合は、以下のようにします。

  ```
  CHGMQMQ QNAME('orange.local.queue') MQMNAME(MYQUEUEMANAGER) MAXMSGLEN(10000)
  ```

  このコマンドにより、1 つの属性、つまり最大メッセージ長の属性は変更されますが、他の属性はすべて変更されません。
- REPLACE *YES オプションを指定した **CRTMQMQ** コマンドを使用する場合は、例えば以下のようにします。

  ```
  CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL) MQMNAME(MYQUEUEMANAGER)
  MAXMSGLEN(10000) REPLACE(*YES)
  ```

  このコマンドにより、最大メッセージ長だけでなく、他のすべての属性も変更されます。 他のすべての属性にはデフォルト値が与えられます。 このキューは、以前は書き込み保護でしたが、これで書き込み可能になります。 キュー SYSTEM.DEFAULT.LOCAL.QUEUE で指定されているとおり、変更されていない限り、書き込み可能はデフォルト値です。

  既存のキューの最大メッセージ長を **短くしても** 、既存のメッセージは影響を受けません。 ただし、新しいメッセージはこの新しい基準に適合する必要があります。

ローカル・キューのクリア

magenta.queue という名前のローカル・キューからすべてのメッセージを削除するためには、次のコマンドを使用します。

```
CLRMQMQ QNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

次の場合には、キューの内容をクリアすることができません。

- 同期点でコミットされていないメッセージで、そのキューに書き込まれているものがある場合
- アプリケーションがそのキューを現在オープンしている場合

ローカル・キューの削除

ローカル・キューを削除するには、**DLTMQMQ** コマンドを使用します。

キュー上にコミットされていないメッセージがある場合、またはキューが使用中である場合、 そのキューは削除できません。

大規模キューの使用可能化

IBM MQ では、2 GB を超えるキューがサポートされます。 IBM i による大容量ファイルのサポートを可能にする方法については、オペレーティング・システムの資料を参照してください。

IBM i 製品資料は、IBM Documentation から参照できます。

一部のユーティリティーでは、2 GB を超えるファイルを処理できない場合があります。 大容量ファイルのサポートを使用可能化する前に、この種のサポートに対する制約事項について、オペレーティング・システムの資料を確認してください。

IBM i 別名キューの操作 (IBM i)

このセクションでは、別名キューを管理するために使用できるコマンドの例をいくつか示します。 ここに示すコマンドは、すべて **WRKMQMQ コマンド・パネル**のオプションで使用することもできます。

別名キュー (キュー別名と呼ばれることもあります) は、MQI 呼び出しのリダイレクトの方法を提供します。 別名キューは、実際のキューではなく、実際のキューに解決される定義です。 別名キュー定義は、TGTQNAME 属性で指定される宛先キュー名を含んでいます。

アプリケーションが MQI 呼び出しの中で別名キューを指定すると、キュー・マネージャーは実行時に実際のキュー名に解決します。

例えば、my.alias.queue という名前のキューにメッセージを入れるようなアプリケーションが開発されたとします。 このアプリケーションは、**MQOPEN** 要求を出すときにこのキューの名前を指定し、メッセージをこのキューに書き込む場合には、間接的にこのキューの名前を指定します。 アプリケーションは、キューが別名キューであることを認識しません。 この別名を使用した各 MQI 呼び出しについて、キュー・マネージャーは実際のキュー名に解決します。このキュー名はローカル・キューか、このキュー・マネージャーに定義されたリモート・キューのいずれかです。

TGTQNAME 属性の値を変更することにより、MQI 呼び出しを別のキュー (おそらく別のキュー・マネージャー上の別のキュー) にリダイレクトできます。 これは、保守、移行、および負荷平衡に役立ちます。

別名キューの定義

次のコマンドにより、別名キューが作成されます。

```
CRTMQMQ QNAME('my.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
MQMNAME(MYQUEUEMANAGER)
```

このコマンドは、MQI 呼び出し (my.alias.queue を指定している) をキュー yellow.queue にリダイレクトします。 このコマンドは、ターゲット・キューを作成しないので、キュー yellow.queue が実行時に存在しなければ、MQI 呼び出しは失敗します。

別名定義を変更すると、MQI 呼び出しを別のキューにリダイレクトできます。 以下に例を示します。

```
CHGMQMQ QNAME('my.alias.queue') TGTQNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

このコマンドは、MQI 呼び出しを別のキュー magenta.queue にリダイレクトします。

別名キューを使用すると、単一のキュー (ターゲット・キュー) が、異なるアプリケーションについては異なる属性を持っているよう見えるようにすることもできます。 これは、アプリケーションごとに 1 つの別名、つまり合計 2 つの別名を定義すると行えます。 2 つのアプリケーションがあるとします。

- アプリケーション ALPHA は、メッセージを yellow.queue に書き込むことができますが、そこからメッセージを読み取ることはできません。
- アプリケーション BETA は、yellow.queue からメッセージを読み取ることはできますが、そこにメッセージを書き込むことはできません。

これは、次のコマンドを使用して行います。

```
/* This alias is put enabled and get disabled for application ALPHA */

CRTMQMQ QNAME('alphas.alias.queue') QTYPE(*ALS)     TGTQNAME('yellow.queue')
PUTENBL(*YES) GETENBL(*NO) MQMNAME(MYQUEUEMANAGER)

/* This alias is put disabled and get enabled for application BETA */

CRTMQMQ QNAME('betas.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
PUTENBL(*NO) GETENBL(*YES) MQMNAME(MYQUEUEMANAGER)
```

ALPHA は、MQI 呼び出しの中でキュー名 alphas.alias.queue を使用しますが、BETA は、キュー名 betas.alias.queue を使用します。 これらはいずれも同じキューをアクセスしますが、その方法は異なっています。

別名キューを定義する際には、ローカル・キューの場合と同様にして、REPLACE *YES 属性を使用することができます。

キュー別名でのその他のコマンドの使用

該当のコマンドを使用すると、別名キューの属性を表示または変更することができます。 以下に例を示します。

```
* Display the alias queue's attributes */

DSPMQMQ QNAME('alphas.alias.queue') MQMNAME(MYQUEUEMANAGER)

/* ALTER the base queue name, to which the alias resolves. */
/* FORCE = Force the change even if the queue is open. */

CHQMQMQ QNAME('alphas.alias.queue') TGTQNAME('orange.local.queue') FORCE(*YES)
MQMNAME(MYQUEUEMANAGER)
```

IBM i モデル・キューの操作 (IBM i)

このセクションでは、モデル・キューを管理するために使用できるコマンドの例をいくつか示します。 ここに示すコマンドは、すべて **WRKMQMQ コマンド・パネル**のオプションで使用することもできます。

キュー・マネージャーは、モデル・キューとして定義されているキュー名を指定した MQI 呼び出しをアプリケーションから受け取ると、動的キュー を作成します。 新しい動的キューの名前は、そのキューの作成時にキュー・マネージャーによって生成されます。 モデル・キュー とは、動的キューの属性を指定しているテンプレートのことで、動的キューはこのモデル・キューから作成されます。

モデル・キューは、アプリケーションがキューを必要とするときにそのキューを作成するための便利な方法を提供します。

モデル・キューの定義

ローカル・キューを定義するのと同じ方法で、一連の属性を持つモデル・キューを定義します。 作成される動的キューが一時キューとなるか永続キューとなるかをモデル・キューには指定できるが、ローカル・キューにはできないこと以外は、モデル・キューとローカル・キューは同じ一連の属性を持っています。 (永続キューはキュー・マネージャーが再始動しても維持されますが、一時キューは維持されません。) 以下に例を示します。

```
CRTMQMQ QNAME('green.model.queue') QTYPE(*MDL) DFNTYPE(*PERMDYN)
```

このコマンドにより、モデル・キュー定義が作成されます。 DFNTYPE 属性により、このテンプレートから作成される実際のキューは、永続動的キューになります。 指定されていない属性は、SYSYTEM.DEFAULT.MODEL.QUEUE デフォルト・キューから自動的にコピーされます。

モデル・キューを定義する際には、ローカル・キューの場合と同様にして、REPLACE *YES 属性を使用することができます。

モデル・キューでのその他のコマンドの使用

該当のコマンドを使用すると、モデル・キューの属性を表示または変更することができます。 以下に例を示します。

```
/* Display the model queue's attributes */

DSPMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('green.model.queue')

/* ALTER the model queue to enable puts on any */
/* dynamic queue created from this model. */

CHGMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('blue.model.queue') PUTENBL(*YES)
```

IBM i トリガー操作 (IBM i)

この情報を使用して、トリガー操作およびプロセス定義について学習します。

IBM MQ には、キューで特定の条件が満たされると自動的にアプリケーションを開始する機能が用意されています。 このような条件の例としては、キュー上のメッセージ数が指定の数に達した場合があります。 この機能は「トリガー操作」と呼ばれています。詳細については、チャネルのトリガー操作に説明があります。

トリガー操作とは

キュー・マネージャーは、特定の条件を構成トリガー・イベントとして定義します。 トリガー操作がキューに対して有効になっている場合にトリガー・イベントが発生すると、キュー・マネージャーはトリガー・メッセージを開始キューに送信します。 開始キューにトリガー・メッセージがある場合、トリガー・イベントが発生したことを意味しています。

キュー・マネージャーが生成したトリガー・メッセージは、永続的ではありません。 これにより ロギングが減少し (したがってパフォーマンスが改善され)、再始動中の重複が最小限になります。 その結果、再始動の時間が短縮されます。

トリガー・モニターとは

開始キューを処理するプログラムは、トリガー・モニター・アプリケーションと呼ばれ、トリガー・メッセージを読み取り、トリガー・メッセージの情報に基づいて適切な処理を行います。 通常 この処理によって他のアプリケーションが開始され、トリガー・メッセージを生成する要因となったキューが処理されます。 キュー・マネージャーから見て、トリガー・モニター・アプリケーションは特別なものではなく、キュー (開始キュー) のメッセージを読み取る別の 1 つのアプリケーションです。

トリガー・モニターのジョブの実行依頼の属性の変更

コマンド **STRMQMTRM** として提供されているトリガー・モニターは、システムのデフォルトのジョブ記述 (QDFTJOBD) を使用して各トリガー・メッセージに対するジョブの実行依頼を行います。 これには、実行依頼されたジョブが常に QDFTJOBD と呼ばれ、ライブラリー・リスト *SYSVAL を含むデフォルトのジョブ記述の属性があるという点で制限があります。 IBM MQ には、これらの属性を指定変更する方法が用意されています。 例えば、ジョブ名をより分かりやすくするために、実行依頼されたジョブを次のようにカスタマイズすることができます。

1. ジョブ記述で、任意の記述 (例えば、ロギング値) を指定する。
2. トリガー操作処理で使用されるプロセス定義の環境データを指定する。

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOBD)')
```

トリガー・モニターは、指定された記述を用いて SBMJOB を実行します。

該当するキーワードおよび値をプロセス定義の環境データに指定することに よって、SBMJOB の他の属性を指定変更することができます。 唯一の例外は、CMD キーワードです。 これは、この属性がトリガー・モニターによって指定されるためです。 ジョブ名と記述の両方が変更されるプロセス定義の環境データを指定するコマンドの例を以下に示します。

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOB)
JOB(TRIGGER)')
```

トリガー操作のためのアプリケーション・キューの定義

アプリケーション・キューとは、アプリケーションが MQI を介してメッセージ用に使用するローカル・キューのことです。 トリガー操作では、いくつかのキュー属性をアプリケーション・キューに定義する必要があります。 トリガー操作自体は、TRGENBL 属性によって使用可能になります。

以下に示す例では、トリガー・イベントは、motor.insurance.queue というローカル・キューに優先度 5 以上のメッセージが 100 個入れられたときに生成されます。

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.insurance.queue') QTYPE(*LCL)
PRCNAME('motor.insurance.quote.process') MAXMSGLEN(2000)
DFTMSGPST(*YES) INITQNAME('motor.ins.init.queue')
TRGENBL(*YES) TRGTYPE(*DEPTH) TRGDEPTH(100) TRGMSGPTY(5)
```

パラメーターは、以下のとおりです。

MQMNAME(MYQUEUEMANAGER)
キュー・マネージャーの名前。

QNAME('motor.insurance.queue')
定義するアプリケーション・キューの名前

PRCNAME('motor.insurance.quote.process')
トリガー・モニター・プログラムによって開始するアプリケーションの名前

MAXMSGLEN(2000)
キューに入れる最大メッセージ長

DFTMSGPST(*YES)
デフォルトでメッセージをこのキュー上で存続させます。

INITQNAME('motor.ins.init.queue')
キュー・マネージャーがトリガー・メッセージを入れる開始キューの名前

TRGENBL(*YES)
トリガー属性値

TRGTYPE(*DEPTH)
要求した優先度 (**TRGMSGPTY**) を持つメッセージの数が **TRGDEPTH** で指定した数に達したときにトリガー・イベントを生成します。

TRGDEPTH(100)
トリガー・イベントを生成するのに必要なメッセージ数

TRGMSGPTY(5)
トリガー・イベントを生成するかどうかを決める際に、キュー・マネージャーが考慮に入れるメッセージの優先度。 優先度 5 以上のメッセージだけが考慮に入れられます。

開始キューの定義

トリガー・イベントが発生すると、キュー・マネージャーは、アプリケーション・キュー定義に指定された開始キューにトリガー・メッセージを入れます。 開始キューには特別の設定値はありませんが、参考として以下に示す motor.ins.init.queue というローカル・キューの定義を使用することができます。

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.ins.init.queue') QTYPE(*LCL)
GETENBL(*YES) SHARE(*NO) TRGTYPE(*NONE)
MAXMSGL(2000)
MAXDEPTH(1000)
```

プロセス定義の作成

プロセス定義を作成するには、**CRTMQMPRC** コマンドを使用します。 プロセス定義は、アプリケーション・キューを、キューからのメッセージを処理するアプリケーションと関連付けます。 この関連付けは、アプリケーション・キュー motor.insurance.queue の PRCNAME 属性によって行われます。 次のコマンドは、この例で識別されている必須プロセス motor.insurance.quote.process を作成します。

```
CRTMQMPRC MQMNAME(MYQUEUEMANAGER) PRCNAME('motor.insurance.quote.process')
TEXT('Insurance request message processing')
APPTYPE(*OS400) APPID(MQTEST/TESTPROG)
USRDATA('open, close, 235')
```

パラメーターは、以下のとおりです。

MQMNAME(MYQUEUEMANAGER)
キュー・マネージャーの名前。

PRCNAME('motor.insurance.quote.process')
プロセス定義の名前

TEXT('Insurance request message processing')
この定義を関連付けるアプリケーション・プログラムの記述。 このテキストは、**DSPMQMPRC** コマンドを使用すると表示されます。 これは、プロセスが行う事柄を識別するのに役立ちます。 ストリングの中でスペースを使用する場合は、ストリングを単一引用符で囲む必要があります。

APPTYPE(*OS400)
開始するアプリケーションのタイプ

APPID(MQTEST/TESTPROG)
アプリケーションの実行可能プログラムの名前で、完全修飾ファイル名として指定されます。

USRDATA('open, close, 235')
アプリケーションで使用できるユーザー定義のデータ

プロセス定義の表示

定義の結果を調べるには、**DSPMQMPRC** コマンドを使用します。 以下に例を示します。

```
MQMNAME(MYQUEUEMANAGER) DSPMQMPRC('motor.insurance.quote.process')
```

CHGMQMPRC コマンドを使用して既存のプロセス定義を変更したり、**DLTMQMPRC** を使用してプロセス定義を削除することもできます。

IBM i 2 つの IBM MQ システム間の通信 (IBM i)

CL コマンドで 2 つの IBM MQ for IBM i システムをセットアップして相互に通信できるようにするためのコーディング例を取り上げます。

2 つのシステムは SYSTEMA および SYSTEMB という名前で、使用する通信プロトコルは TCP/IP です。

以下の手順を実行します。

1. SYSTEMA 上にキュー・マネージャーを作成し、それを QMGRA1 と呼びます。

```
CRTMQM   MQMNAME(QMGRA1) TEXT('System A - Queue +
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

2. このキュー・マネージャーを開始します。

```
STRMQM   MQMNAME(QMGRA1)
```

3. SYSTEMB 上のキュー・マネージャーにメッセージを送信する必要がある SYSTEMA 上の IBM MQ オブジェクトを定義します。

```
/* Transmission queue */
CRTMQMQ  QNAME(XMITQ.TO.QMGRB1) QTYPE(*LCL) +
MQMNAME(QMGRA1) TEXT('Transmission Queue +
to QMGRB1') MAXDEPTH(5000) USAGE(*TMQ)

/* Remote queue that points to a queue called TARGETB        */
/* TARGETB belongs to queue manager QMGRB1 on SYSTEMB        */
CRTMQMQ  QNAME(TARGETB.ON.QMGRB1) QTYPE(*RMT) +
MQMNAME(QMGRA1) TEXT('Remote Q pointing +
at Q TARGETB on QMGRB1 on Remote System +
SYSTEMB') RMTQNAME(TARGETB) +
RMTMQMNAME(QMGRB1) TMQNAME(XMITQ.TO.QMGRB1)

/* TCP/IP sender channel to send messages to the queue manager on SYSTEMB*/
CRTMQMCHL CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*SDR) +
MQMNAME(QMGRA1) TRPTYPE(*TCP) +
TEXT('Sender Channel From QMGRA1 on +
SYSTEMA to QMGRB1 on SYSTEMB') +
CONNAME(SYSTEMB) TMQNAME(XMITQ.TO.QMGRB1)
```

4. SYSTEMB 上にキュー・マネージャーを作成し、それを QMGRB1 と呼びます。

```
CRTMQM   MQMNAME(QMGRB1) TEXT('System B - Queue +
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

5. SYSTEMB 上のキュー・マネージャーを開始します。

```
STRMQM   MQMNAME(QMGRB1)
```

6. SYSTEMA 上のキュー・マネージャーからメッセージを受信するために必要な IBM MQ オブジェクトを定義します。

```
/* Local queue to receive messages on */
CRTMQMQ  QNAME(TARGETB) QTYPE(*LCL) MQMNAME(QMGRB1) +
TEXT('Sample Local Queue for QMGRB1')

/* Receiver channel of the same name as the sender channel on SYSTEMA */
CRTMQMCHL CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*RCVR) +
MQMNAME(QMGRB1) TRPTYPE(*TCP) +
TEXT('Receiver Channel from QMGRA1 to +
QMGRB1')
```

7. 最後に、SYSTEMB 上の TCP/IP listener を開始して、チャネルを開始できるようにします。 この例では、デフォルトのポート 1414 を使用しています。

```
STRMQMLSR MQMNAME(QMGRB1)
```

これで、テスト・メッセージを SYSTEMA と SYSTEMB との間で送受信することができます。 提供されている例の 1 つを使用して、いくつかのメッセージに put を実行して、SYSTEMA 上のリモート・キューに書き込んでください。

SYSTEMA 上のチャネルを開始するには、コマンド **STRMQMCHL** を使用するか、またはコマンド **WRKMQMCHL** を使用して、送信側チャネルに対して開始要求 (オプション 14) を入力してください。

チャネルは RUNNING 状況になるはずであり、メッセージは SYSTEMB 上のキュー TARGETB に送信されます。

以下のコマンドを発行して、メッセージを検査してください。

```
WRKMQMMSG QNAME(TARGETB) MQMNAME(QMGRB1).
```

IBM i サンプル・リソース定義 (IBM i)

このサンプルには、AMQSAMP4 サンプル IBM i 制御言語プログラムが含まれています。

```
/*********************************************************************/
/*                                                                   */
/* Program name: AMQSAMP4                                            */
/*                                                                   */
/* Description: Sample CL program defining MQM queues                */
/*       to use with the sample programs                             */
/*       Can be run, with changes as needed, after                   */
/*       starting the MQM                                            */
/*                                                                   */
/* <N_OCO_COPYRIGHT>                                               */
/* Licensed Materials - Property of IBM                              */
/*                                                                   */
/* 63H9336                                                           */
/* (c) Copyright IBM Corp. 1993, 2024. All Rights Reserved.          */
/*                                                                   */
/* US Government Users Restricted Rights - Use, duplication or       */
/* disclosure restricted by GSA ADP Schedule Contract with           */
/* IBM Corp.                                                         */
/* <NOC_COPYRIGHT>                                                 */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/* Function:                                                         */
/*                                                                   */
/*                                                                   */
/*  AMQSAMP4 is a sample CL program to create or reset the           */
/*  MQI resources to use with the sample programs.                   */
/*                                                                   */
/*  This program, or a similar one, can be run when the MQM          */
/*  is started - it creates the objects if missing, or resets        */
/*  their attributes to the prescribed values.                       */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*  Exceptions signaled: none                                        */
/*  Exceptions monitored: none                                       */
/*                                                                   */
/*  AMQSAMP4 takes a single parameter, the Queue Manager name        */
/*                                                                   */
/*********************************************************************/
QSYS/PGM PARM(&QMGRNAME)

/*********************************************************************/
/*  Queue Manager Name Parameter                                     */
/*********************************************************************/
QSYS/DCL VAR(&QMGRNAME) TYPE(*CHAR)

/*********************************************************************/
/*    EXAMPLES OF DIFFERENT QUEUE TYPES                              */
/*                                                                   */
/*  Create local, alias and remote queues                            */
/*                                                                   */
/*  Uses system defaults for most attributes                         */
/*                                                                   */
/*********************************************************************/
/*  Create a local queue   */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.LOCAL')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
```

```
TEXT('Sample local queue') /* description */+
SHARE(*YES)        /* Shareable  */+
DFTMSGPST(*YES) /* Persistent messages OK */

/*  Create an alias queue  */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.ALIAS')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*ALS) REPLACE(*YES)         +
+
TEXT('Sample alias queue')         +
DFTMSGPST(*YES) /* Persistent messages OK */+
TGTQNAME('SYSTEM.SAMPLE.LOCAL')

/*  Create a remote queue - in this case, an indirect reference  */
/*   is made to the sample local queue on OTHER queue manager  */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.REMOTE')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*RMT) REPLACE(*YES)         +
+
TEXT('Sample remote queue')/* description */+
DFTMSGPST(*YES) /* Persistent messages OK */+
RMTQNAME('SYSTEM.SAMPLE.LOCAL')       +
RMTMQMNAME(OTHER)  /* Queue is on OTHER  */

/*  Create a transmission queue for messages to queues at OTHER  */
/*  By default, use remote node name                */
CRTMQMQ   QNAME('OTHER') /* transmission queue name */+
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES) +
TEXT('Transmision queue to OTHER') +
USAGE(*TMQ)  /* transmission queue   */

/*********************************************************************/
/*    SPECIFIC QUEUES AND PROCESS USED BY SAMPLE PROGRAMS           */
/*                                                                   */
/*  Create local queues used by sample programs                      */
/*  Create MQI process associated with sample initiation queue       */
/*                                                                   */
/*********************************************************************/
/*  General reply queue  */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.REPLY')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('General reply queue')         +
DFTMSGPST(*NO) /* Not Persistent     */

/*  Queue used by AMQSINQ4                     */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.INQ')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSINQ4')         +
SHARE(*YES)        /* Shareable  */+
DFTMSGPST(*NO) /* Not Persistent     */+
+
TRGENBL(*YES) /* Trigger control on   */+
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.INQPROCESS')     +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Queue used by AMQSSET4                     */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.SET')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSSET4')         +
SHARE(*YES)        /* Shareable */ +
DFTMSGPST(*NO)/* Not Persistent     */ +
+
TRGENBL(*YES) /* Trigger control on   */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.SETPROCESS')     +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Queue used by AMQSECH4                     */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.ECHO')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSECH4')         +
SHARE(*YES)        /* Shareable */ +
```

```
DFTMSGPST(*NO)/* Not Persistent     */ +
+
TRGENBL(*YES) /* Trigger control on   */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')    +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Initiation Queue used by AMQSTRG4, sample trigger process   */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.TRIGGER') +
MQMNAME(&QMGRNAME)              +
QTYPE(*LCL) REPLACE(*YES)  +
TEXT('Trigger queue for sample programs')

/*  MQI Processes associated with triggered sample programs    */
/*                                 */
/*****  Note - there are versions of the triggered samples ******/
/*****   in different languages - set APPID for these   ******/
/*****   process to the variation you want to trigger   ******/
/*                                 */
CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.INQPROCESS')     +
MQMNAME(&QMGRNAME)              +
REPLACE(*YES)                +
+
TEXT('Trigger process for AMQSINQ4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSINQ4')  /* C       +
/*    APPID('QMQM/AMQ0INQ4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3INQ4')  /* RPG - ILE */

CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.SETPROCESS')     +
MQMNAME(&QMGRNAME)              +
REPLACE(*YES)                +
+
TEXT('Trigger process for AMQSSET4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSSET4')  /* C   */    +
/*    APPID('QMQM/AMQ0SET4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3SET4')  /* RPG - ILE */

CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')    +
MQMNAME(&QMGRNAME)              +
REPLACE(*YES)                +
+
TEXT('Trigger process for AMQSECH4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSECH4')  /* C   */    +
/*    APPID('QMQM/AMQ0ECH4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3ECH4')  /* RPG - ILE */

/*********************************************************************/
/*                                                                   */
/*  Normal return.                                                   */
/*                                                                   */
/*********************************************************************/
SNDPGMMSG  MSG('AMQSAMP4 Completed creating sample +
objects for ' *CAT &QMGRNAME)
RETURN
ENDPGM

/*********************************************************************/
/*                                                                   */
/* END OF AMQSAMP4                                                   */
/*                                                                   */
/*********************************************************************/
```

IBM i IBM MQ for IBM i 管理の代替方法

IBM MQ for IBM i の管理には、CL コマンドを使用する方式を推奨します。 ただし、MQSC コマンド、PCF コマンド、制御コマンド、リモート管理など、さまざまな管理方法を使用できます。

このタスクについて

通常は、IBM i CL コマンドを使用して IBM MQ for IBM i を管理します。 これらのコマンドの概要については、375 ページの『CL コマンドを使用した IBM MQ for IBM i の管理』を参照してください。

サブトピックで説明されているように MQSC コマンドおよび PCF コマンドを使用することもできます。また、 10 ページの『制御コマンドによる IBM MQ for Multiplatforms の管理』で説明されているように、制御コマンドを使用することもできます。

キュー・マネージャーの操作をモニターするために、IBM MQ の観測イベントを使用することができます。IBM MQ インスツルメンテーション・イベントとその使用方法については、「インスツルメンテーション・イベント」を参照してください。

IBM i CL コマンドを使用する代わりに、以下のサブトピックで説明されているいずれかの管理方法を使用してください。

IBM i ローカル管理とリモート管理 (IBM i)

IBM MQ for IBM i オブジェクトをローカルまたはリモート側で管理します。

このタスクについて

ローカル管理 とは、ローカル・システムに定義したキュー・マネージャーで管理タスクを実行することです。 IBM MQ チャネルは無関係であり、通信はオペレーティング・システムによって管理されるため、IBM MQ ではこれをローカル管理と考えることができます。 この種のタスクを実行するためには、リモート・システムにログオンしてそこからコマンドを発行するか、あるいはユーザーの代わりにコマンドを発行するプロセスを作成する必要があります。

IBM MQ では、リモート管理 という管理方法による、単一ポイントからの管理がサポートされています。リモート管理は、プログラマブル・コマンド・フォーマット (PCF) の制御メッセージを、宛先キュー・マネージャー上の SYSTEM.ADMIN.COMMAND.QUEUE に送信することによって行われます。

PCF メッセージを生成する方法はいくつかあります。 これらについては、以下のステップで説明します。

手順

- PCF メッセージを使用してプログラムを作成します。 391 ページの『IBM i での PCF コマンドによる管理』 を参照してください。
- PCF メッセージを送信する MQAI を使用してプログラムを作成します。 37 ページの『MQAI を使用して PCF の使い方を単純化する』 を参照してください。
- IBM MQ for Windows で使用可能な IBM MQ エクスプローラーを使用します。これにより、グラフィカル・ユーザー・インターフェース (GUI) を使用できるようになり、正しい PCF メッセージが生成されます。 391 ページの『IBM MQ for IBM i での IBM MQ Explorer の使用』を参照してください。
- **STRMQMMQSC** を使用して、リモート・キュー・マネージャーにコマンドを間接的に送信する。 389 ページの『IBM i での MQSC コマンドを使用した管理』を参照してください。

  例えば、リモート・コマンドを発行して、リモート・キュー・マネージャー上のキュー定義を変更することができます。

  一部のコマンドは、このような方法では発行することができません。特に、キュー・マネージャーの作成や開始、およびコマンド・サーバーの開始などの際には、このような方法では発行できません。 このようなタスクを実行するためには、リモート・システムにログオンしてそこからコマンドを発行するか、あるいはユーザーの代わりにコマンドを発行するプロセスを作成する必要があります。

IBM i IBM i での MQSC コマンドを使用した管理

IBM i では、スクリプト・ファイル内にコマンドのリストを作成し、 **STRMQMMQSC** コマンドを使用してそのファイルを実行します。 MQSC コマンドを使用して、キュー・マネージャー自体、キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、トピック、および認証情報オブジェクトなどのキュー・マネージャー・オブジェクトを管理します。

このタスクについて

IBM MQ スクリプト (MQSC) コマンドは、人間が理解できる形式で EBCDIC テキストで記述されます。 キュー・マネージャーへの MQSC コマンドの発行には、**STRMQMMQSC** IBM MQ CL コマンドを使用します。 こ

の方式はバッチ方式のみで、サーバーのライブラリー・システムにあるソース物理ファイルから入力します。 このソース物理ファイルのデフォルト名は QMQSC です。

重要 : QTEMP ライブラリーを STRMQMMQSC へのソース・ライブラリーとして使用しないでください。 QTEMP ライブラリーの使用は限定されています。 このコマンドの入力ファイルとして別のライブラリーを使用する必要があります。

IBM MQ 環境間での移植性を考えて、MQSC コマンド・ファイルの行の長さは、最大 72 文字に制限します。 コマンドが次の行に続くことを示すには、正符号を使用します。

MQSC で指定されたオブジェクト属性は、このトピックでは大文字で示されています (RQMNAME など)。 ただし、大文字と小文字は区別されません。

注 :

1. MQSC ファイルの形式は、ファイル・システム内の場所には依存していません。
2. MQSC 属性名は 8 文字までに制限されています。
3. MQSC コマンドは、すべての IBM MQ プラットフォームで使用可能です。

各 MQSC コマンドおよびその構文についての説明は、MQSC コマンドを参照してください。

手順

1. QMQSC ソース・ファイルを作成します。

   IBM MQ for IBM i は、 QMQSC というソース・ファイルを提供しません。 MQSC コマンドを処理するには、以下のコマンドを発行して、選択したライブラリー内に QMQSC ソース・ファイルを作成する必要があります。

   ```
   CRTSRCPF FILE(MYLIB/QMQSC) RCDLEN(240) TEXT('IBM MQ - MQSC Source')
   ```

2. メンバーを処理します。

   MQSC ソースは、このソース・ファイル内にメンバーとして存在します。 メンバーを使用するには、次のコマンドを入力します。

   ```
   WRKMBRPDM MYLIB/QMQSC
   ```

   これで、新規メンバーを追加し、既存のメンバーを維持することができます。

   ```
   .
   .
   DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE +
   DESCR(' ') +
   PUT(ENABLED) +
   DEFPRTY(0) +
   DEFPSIST(NO) +
   GET(ENABLED) +
   MAXDEPTH(5000) +
   MAXMSGL(1024) +
   DEFSOPT(SHARED) +
   NOHARDENBO +
   USAGE(NORMAL) +
   NOTRIGGER;
   .
   .
   ```

   図 *20. MQSC* コマンド・ファイル *myprog.in* から抽出します。このファイルには、属性を持つ *MQSC* コマンド *(DEFINE QLOCAL)* が示されています。

関連情報

MQSC コマンドを使用した IBM MQ の管理

IBM i での PCF コマンドによる管理

IBM MQ プログラマブル・コマンド・フォーマット (PCF) コマンドの目的は、管理タスクを管理プログラムに組み込めるようにすることです。 このようにして、プログラムから、キューやプロセス定義を作成したり、キュー・マネージャーを変更したりすることができます。

PCF コマンドは、MQSC コマンドが対象とする範囲と同じ機能範囲をカバーします。 しかし、MQSC コマンドとは異なり、PCF コマンドおよびそれらの応答は、読み取りできるテキスト形式ではありません。

単一のノードから、ネットワーク内の任意のキュー・マネージャーへ PCF コマンドを発行するプログラムを作成することができます。 これにより、管理タスクを中央集中方式にすると同時に自動化することができます。

各 PCF コマンドは、IBM MQ メッセージのアプリケーション・データ部分に組み込まれたデータ構造です。各コマンドは、他のメッセージの場合と同様に、MQI 機能 MQPUT を使用して宛先キュー・マネージャーに送られます。 メッセージを受信するキュー・マネージャー上のコマンド・サーバーは、そのコマンドをコマンド・メッセージとして解釈し、実行します。 応答を入手するには、アプリケーションが MQGET 呼び出しを発行します。 すると、応答データが別のデータ構造に戻されます。 次に、アプリケーションはその応答を処理し、その応答に応じてアクションを実行します。

次に、PCF コマンド・メッセージを作成するためにアプリケーション・プログラマーが指定する必要のある事項のいくつかを簡単に示します。

メッセージ記述子
標準 IBM MQ メッセージ記述子です。これを使用して以下を行います。

- メッセージ・タイプ (*MsgType*) には、MQMT_REQUEST を指定します。
- メッセージ形式 (*Format*) には、MQFMT_ADMIN を指定します。

アプリケーション・データ
これには、PCF ヘッダーを含む PCF メッセージが入ります。このメッセージの中で、以下のように指定します。

- PCF メッセージ・タイプ (*Type*) には MQCFT_COMMAND を指定します。
- コマンド ID は、コマンドを指定します (例: *Change Queue* (MQCMD_CHANGE_Q))。

エスケープ PCF は、メッセージ・テキスト内に MQSC コマンドを含んでいる PCF コマンドです。 PCF を使用して、リモート・キュー・マネージャーにコマンドを送信することができます。 詳しくは、37 ページの『MQAI を使用して PCF の使い方を単純化する』を参照してください。

PCF データ構造とその実装方法の詳細説明については、コマンドおよび応答の構造を参照してください。

IBM i での IBM MQ for IBM i の使用

ここでは、 IBM MQ Explorer を使用して IBM MQ for IBM i を管理する方法について説明します。

IBM MQ for Windows (x86 プラットフォーム) および IBM MQ for Linux (x86 および x86-64 プラットフォーム) には、管理タスクを実行するため、CL コマンド、制御コマンド、MQSC コマンドを使用する代わりに IBM MQ エクスプローラーという管理インターフェースが用意されています。

IBM MQ ExplorerWindows (x86 プラットフォーム), または Linux (x86 および x86-64 プラットフォーム) を実行しているコンピュータから、関心のあるキュー・マネージャーやクラスタを指定して IBM MQ Explorer、ネットワークのローカルまたはリモート管理を実行することができます。

IBM MQ Explorer では、以下を実行できます。

- キュー・マネージャーの起動と停止 (ローカル・マシン上のキュー・マネージャーのみ)
- キュー、トピック、チャネルなどの IBM MQ オブジェクトの定義、定義の表示、および定義の変更
- キュー内のメッセージの表示
- チャネルの開始と停止
- チャネル状況の情報の表示
- クラスターを構成するキュー・マネージャーの表示

- 特定のキューがオープンしているアプリケーション、ユーザー、またはチャネルの検査
- 「**新しいクラスターの作成**」ウィザードによる、新しいキュー・マネージャー・クラスターの作成
- 「**クラスターへのキュー・マネージャーの追加**」ウィザードによる、クラスターへのキュー・マネージャーの追加
- Transport Layer Security (TLS) チャネル・セキュリティーで使用される、認証情報オブジェクトの管理。

オンライン・ガイダンスを使用して、以下を行うことができます。

- キュー・マネージャー、キュー、チャネル、プロセス定義、クライアント接続チャネル、リスナー、トピック、サービス、名前リスト、およびクラスターなどのさまざまなリソースの定義と管理
- キュー・マネージャーとその関連プロセスの起動/停止
- 使用ワークステーション上または他のワークステーションからの、キュー・マネージャーとその関連オブジェクトの表示
- キュー・マネージャー、クラスター、およびチャネルの状況の確認

サーバー・マシン上で IBM MQ Explorer を使用して IBM MQ を管理する前に、以下の要件が満たされていることを確認してください。 次の項目について確認します。

1. コマンド・サーバーは、サーバー上で CL コマンド **STRMQMCSVR** によって開始された、管理対象のすべてのキュー・マネージャーに対して動作している。
2. リモートのどのキュー・マネージャーについても、適切な TCP/IP listener が存在する。 これは、**STRMQMLSR** コマンドによって開始される IBM MQ リスナーです。
3. リモートのすべてのキュー・マネージャー上に、サーバー接続チャネル、SYSTEM.ADMIN.SVRCONN がある。 このチャネルはユーザー自身で作成する必要があります。 このチャネルは、管理対象のリモート・キュー・マネージャーに必須です。 このチャネルがないと、リモート管理はできません。
4. SYSTEM.MQEXPLORER.REPLY.MODEL キューが終了していることを確認する。

IBM i コマンド・サーバーのリモート管理 (IBM i)

この情報を使用して、IBM MQ for IBM i のコマンド・サーバーのリモート管理について学習します。

各キュー・マネージャーには、それぞれに関連付けられた 1 つのコマンド・サーバーがあります。 コマンド・サーバーは、リモート・キュー・マネージャーからの着信コマンド、またはアプリケーションからの PCF コマンドを処理します。 コマンド・サーバーは処理のために、そのコマンドをキュー・マネージャーに渡し、コマンドの発信元に応じて、完了コードやオペレーター・メッセージを戻します。

コマンド・サーバーは、PCF、MQAI に関するすべての管理およびリモート管理にも必須です。

注 : リモート管理では、宛先キュー・マネージャーを確実に実行しているようにする必要があります。 実行していないと、コマンドを含んだメッセージは、メッセージの発信元のキュー・マネージャーから出ていくことができません。 代わりに、それらのメッセージは、リモート・キュー・マネージャーが使用しているローカル伝送キューに保持されます。 可能な限り、この状態は避けてください。

コマンド・サーバーを開始および停止するための別々の制御コマンドがあります。 IBM MQ エクスプローラーを使用して、以下のセクションに記載された操作を実行できます。

コマンド・サーバーの開始と停止

コマンド・サーバーを開始するには、次の CL コマンドを使用します。

```
STRMQMCSVR MQMNAME('saturn.queue.manager')
```

ここで、saturn.queue.manager は、開始されるコマンド・サーバーに関連したキュー・マネージャーです。

コマンド・サーバーを停止するには、以下の CL コマンドのいずれかを使用します。

1.
```
ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*CNTRLD)
```

制御された停止を実行するためのものです。ここで、saturn.queue.manager は、停止されるコマンド・サーバーに関連したキュー・マネージャーです。 これはデフォルトのオプションであり、OPTION(*CNTRLD) が省略できることを意味しています。

2.
```
ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*IMMED)
```

即時停止を実行するためのものです。ここで、saturn.queue.manager は、停止されるコマンド・サーバーに関連したキュー・マネージャーです。

コマンド・サーバーの状況を表示する

リモート管理では、宛先キュー・マネージャー上でコマンド・サーバーが実行中であることを確認します。これが実行されていないと、リモート・コマンドを処理できません。 コマンドを含んだメッセージは、ターゲット・キュー・マネージャーのコマンド・キュー SYSTEM.ADMIN.COMMAND.QUEUE に入れられます。

キュー・マネージャー (ここでは saturn.queue.manager) のコマンド・サーバーの状況を表示するには、次の CL コマンドを出します。

```
DSPMQMCSVR MQMNAME('saturn.queue.manager')
```

ターゲット・マシンにこのコマンドを発行します。 コマンド・サーバーが実行されていると、393 ページの図 21 のパネルが表示されます。

```
Display MQM Command Server (DSPMQMCSVR)

Queue manager name . . . . . > saturn.queue.manager

MQM Command Server Status. . . . > RUNNING




F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

図 *21.* 「*MQ* コマンド・サーバーの表示」パネル

IBM i Web コンソール・コマンドの実行

Web コンソール関連の Qshell コマンドを IBM MQ for IBM i 上で正しく実行するには、以下のテキストの説明に従って環境を構成する必要があります。

このタスクについて

Qshell の開始時には、コマンド処理の内部テーブルはジョブの CCSID に基づいて初期化されます。 Web コンソール関連の Qshell コマンドを IBM i 上で正しく実行するには、環境を構成する必要があります。

ロケールを設定するには、LANG 環境変数をロケール・オブジェクトへのパス名に設定します。 例えば、US English のロケールを設定するには LANG 環境変数を次のように設定します。

```
LANG=/QSYS.LIB/EN_US.LOCALE
```

Qshell でコマンド・セットを発行してすべての環境変数をリストすることにより、設定を確認できます。通常、これはランタイム環境のロケールに影響する可能性がある LANG です。 これには、LC_ALL が含まれることもあります。

Qshell コマンドを正しく実行するには、ロケール環境設定がジョブ設定と整合している必要があります。

手順

CL コマンド DSPJOB JOB(JobNumber/USERProfile/JobName) の使用

a) オプション 2 を選択して、ジョブ定義属性を表示します。

b) 以下の属性は、LANG または LC_ALL 環境設定と整合している必要があります。

- 言語 ID
- 国または地域の ID
- コード化文字セット ID

例えば、

```
LANG=/QSYS.LIB/FR_FR.LOCALE
```

ジョブ属性は次のようになります。

- 言語 ID . . . . . FRA
- 国または地域 ID . . . FR
- コード化文字セット ID 297

次のタスク

各国語サポートについて詳しくは、IBM Documentation のトピック『各国語サポート (NLS) に関する考慮事項』を参照してください。

IBM i IBM i でのワーク・マネジメント

この情報は、IBM MQ がどのように実行要求を処理するかについて説明します。また IBM MQ に関連付けられたジョブを優先順位付けおよび制御するために使用可能なオプションの詳細についても説明します。

警告

IBM i および IBM MQ 実行管理機能の概念を完全に理解していない限り、IBM MQ 実行管理機能オブジェクトを変更しないでください。

サブシステムおよびジョブ記述に関する追加情報は、 IBM i 製品資料の「 実行管理機能 」に記載されています。 特に、Starting jobs と Batch jobs のセクションに注目してください。

IBM MQ for IBM i には、IBM i UNIX 環境と IBM i スレッドが組み込まれています。 統合ファイル・システム (IFS) のオブジェクトは変更**しないで**ください。

通常の操作時には、IBM MQ キュー・マネージャーは数多くのバッチ・ジョブを開始し、さまざまなタスクを実行します。 デフォルトでは、これらのバッチ・ジョブは、IBM MQ のインストール時に作成される QMQM サブシステムで実行されます。

ワーク・マネジメントとは、IBM MQ タスクを調整して、ご使用のシステムで最適のパフォーマンスが得られるようにすること、または管理をより簡単にすることを言います。

例えば、以下を行うことができます。

- ジョブの実行優先順位を変更して、特定のキュー・マネージャーによる対応性を、他のものより高くする。
- いくつかのジョブの出力を、特定の出力キューにリダイレクトする。
- 特定のタイプのすべてのジョブを、特定のサブシステム内で実行する。
- エラーとサブシステムを分離する。

ワーク・マネジメントは、IBM MQ ジョブに関連付けられたジョブ記述を作成または変更することによって実施されます。 ワーク・マネジメントは、次の対象について構成できます。

- IBM MQ インストール済み環境全体
- 個々のキュー・マネージャー

- 個々のキュー・マネージャーの個々のジョブ

IBM i IBM i の IBM MQ タスク

この表は、IBM MQ for IBM i ジョブとそれぞれを簡単に説明したものです。

キュー・マネージャーの実行時には、IBM MQ サブシステム内の QMQM ユーザー・プロファイルの下で実行される次のバッチ・ジョブの一部またはすべてが表示されます。 395 ページの表 21 にはそれらのジョブの概要が掲載されています。

「**キュー・マネージャーの処理 (WRKMQM)**」パネルのオプション 22 を使用して、キュー・マネージャーに接続されたすべてのジョブを表示できます。 listener は、WRKMQMLSR コマンドを使用して表示できます。

表 *21. IBM MQ* タスク。

ジョブ名	Function
AMQZMUC0	ユーティリティー・マネージャー。 このジョブは、例えばジャーナル・チェーン・ マネージャーのような、重要なキュー・マネージャー・ユーティリティーを実行します。
AMQZXMA0	実行制御プログラムは、キュー・マネージャーによって開始される最初のジョブです。 これは MQCONN 要求を処理し、また IBM MQ API 呼び出しを処理するエージェント・プロセスを開始します。
AMQZFUMA	オブジェクト権限マネージャー (OAM)。
AMQZLAA0	キュー・マネージャー・エージェントは、MQCNO_STANDARD_BINDING を使用してキュー・マネージャーに接続するほとんどのアプリケーションの作業を実行します。
AMQZLSA0	キュー・マネージャー・エージェント。
AMQZMUF0	ユーティリティー・マネージャー
AMQZMGR0	プロセス・コントローラー。 このジョブは、リスナーとサービスの開始および管理に使用されます。
AMQZMUR0	ユーティリティー・マネージャー。 このジョブは、例えばジャーナル・チェーン・ マネージャーのような、重要なキュー・マネージャー・ユーティリティーを実行します。
AMQFQPUB	キューに入れられたパブリッシュ/サブスクライブ・デーモン。
AMQFCXBA	ブローカー・ワーカー・ジョブ。
RUNMQBRK	ブローカー制御ジョブ。
AMQRMPPA	チャネル処理プール・ジョブ。
AMQCRSTA	TCP/IP 起動型チャネル・レスポンダー。
AMQCRS6B	LU62 受信側チャネルおよびクライアント接続 (注を参照)。
AMQRRMFA	クラスターのリポジトリー管理プログラム。
AMQCLMAA	非スレッド型 TCP/IP listener。
AMQPCSEA	PCF コマンド処理プログラムは、PCF およびリモート管理要求を処理します。
RUNMQTRM	トリガー・モニター。
RUNMQDLQ	送達不能キュー・ハンドラー。
RUNMQCHI	チャネル・イニシエーター。
RUNMQCHL	各送信側チャネルに対して開始される送信側チャネル・ジョブ。
RUNMQLSR	スレッド型 TCP/IP listener。
AMQRCMLA	チャネル MQSC および PCF コマンド・プロセッサー。

注 : LU62 受信側ジョブは、通信サブシステムで実行され、その実行時プロパティーは、ジョブの開始時に使用される経路指定項目および通信項目から取られます。 詳しくは、開始される側 (受信側) を参照してください。

IBM i ワーク・マネジメント・オブジェクト (IBM i)

IBM MQ のインストール時に、ワーク・マネジメントを支援する様々なオブジェクトが QMQM ライブラリーに提供されます。 これらのオブジェクトは、IBM MQ ジョブをその固有のサブシステムで実行するために必要なものです。

サンプルのジョブ記述が、2 つの IBM MQ バッチ・ジョブに用意されています。 特定のジョブ記述が用意されていない IBM MQ ジョブは、デフォルトのジョブ記述 QMQMJOBD で実行します。

IBM MQ のインストール時に提供されるワーク・マネジメント・オブジェクトは 396 ページの表 22 に、キュー・マネージャー用に作成されるオブジェクトは 396 ページの表 23 にリストされています。

注 : ワーク・マネジメント・オブジェクトは QMQM ライブラリーにあり、キュー・マネージャー・オブジェクトはキュー・マネージャー・ライブラリーにあります。

表 *22.* ワーク・マネジメント・オブジェクト

名前	タイプ	説明
AMQZLAA0	*JOBD	IBM MQ エージェント・プロセスで使用されるジョブ記述
AMQZLSA0	*JOBD	分離されたバインディング・キュー・マネージャー・エージェント
AMQZXMA0	*JOBD	IBM MQ 実行制御プログラムで使用されるジョブ記述
QMQM	*SBSD	すべての IBM MQ ジョブが実行されるサブシステム
QMQM	*JOBQ	提供されるサブシステムに付加されるジョブ・キュー
QMQMJOBD	*JOBD	ジョブに特定のジョブ記述がない場合に使用されるデフォルト IBM MQ ジョブ記述
QMQMMSG	*MSGQ	IBM MQ ジョブのデフォルト・メッセージ・キュー
QMQMRUN20	*CLS	優先度が高い IBM MQ ジョブのクラス記述
QMQMRUN35	*CLS	優先度が中程度の IBM MQ ジョブのクラス記述
QMQMRUN50	*CLS	優先度が低い IBM MQ ジョブのクラス記述

表 *23.* キュー・マネージャー用に作成されたワーク・マネジメント・オブジェクト

名前	タイプ	説明
AMQA000000	*JRNRCV	ローカル・ジャーナル・レシーバー
AMQAJRN	*JRN	ローカル・ジャーナル
AMQJRNINF	*USRSPC	開始とキュー・マネージャーのメディア・リカバリーに必要な、最新のジャーナル・ レシーバーで更新されるユーザー・スペース。 このユーザー・スペースは、アーカイブを必要とするジャーナル・レシーバーおよび安全に削除可能な ジャーナル・レシーバーを判別するために、アプリケーションから照会できます。
AMQAJRNMSG	*MSGQ	ローカル・ジャーナル・メッセージ・キュー
AMQCRC6B	*PGM	LU6.2 接続を開始するプログラム
AMQRFOLD	*FILE	移行済みのキュー・マネージャーのチャネル定義ファイル
QMQMMSG	*MSGQ	キュー・マネージャー・メッセージ・キュー

IBM i IBM i で IBM MQ がワーク・マネジメント・オブジェクトを使用する方法

この情報は、IBM MQ によるワーク・マネジメント・オブジェクトの使用法について説明し、構成例を提供しています。

重要 : 優先順位ごとに許可されるサブシステム内のジョブ数を制限するために、QMQM サブシステム内のジョブ・キューのエントリー設定を変更しないでください。 変更する場合には、重要な IBM MQ ジョブをサブミットのあとで実行を中止して、キュー・マネージャーの開始を失敗させます。

ワーク・マネジメントの構成方法を理解するには、まず IBM MQ でジョブ記述がどのように使用されるかを理解する必要があります。

ジョブを開始するために使用されるジョブ記述は、ジョブの多くの属性を制御します。 以下に例を示します。

- ジョブが入れられるジョブ・キュー。ジョブはこのジョブ・キューのサブシステムで実行される。
- ジョブを開始するために使用する経路指定データ。およびその実行時パラメーターに使用するジョブのクラス。
- ジョブが印刷ファイルに使用する出力キュー。

IBM MQ ジョブの開始プロセスには、次の 3 つのステップがあると考えることができます。

1. IBM MQ によってジョブ記述が選択されます。

   IBM MQ では次の技法を使用して、どのジョブ記述をバッチ・ジョブに使用するかが決定されます。

   a. キュー・マネージャー・ライブラリーから、ジョブと同じ名前のジョブ記述を探します。 キュー・マネージャー・ライブラリーについて詳しくは、 IBM MQ for IBM i キュー・マネージャー・ライブラリー名の理解 を参照してください。

   b. キュー・マネージャー・ライブラリーから、デフォルトのジョブ記述 QMQMJOBD を探します。

   c. QMQM ライブラリーから、ジョブと同じ名前のジョブ記述を探します。

   d. QMQM ライブラリー内のデフォルトのジョブ記述 QMQMJOBD を使用します。

2. ジョブをジョブ・キューに実行依頼する。

   IBM MQ に用意されたジョブ記述は、デフォルトではジョブをライブラリー QMQM 内のジョブ・キュー QMQM に置くようにセットアップされています。 QMQM ジョブ・キューは、提供される QMQM サブシステムに付加され、デフォルトでは、ジョブは QMQM サブシステムで実行を開始します。

3. ジョブはサブシステムに入り、経路指定ステップをたどります。

   ジョブがサブシステムに入ると、ジョブ記述で指定された経路指定データは、そのジョブの経路指定項目を検出するために使用されます。

   経路指定データは、QMQM サブシステムで定義されている経路指定項目の 1 つと一致しなければならず、これは、ジョブによって使用される、提供されるクラス (QMQMRUN20、QMQMRUN35、または QMQMRUN50) のいずれかを定義します。

注 : IBM MQ ジョブが開始していないと思われる場合は、サブシステムが実行されていること、またジョブ・キューが保留状態になっていないことを確認してください。

IBM MQ ワーク・マネジメント・オブジェクトを変更した場合は、すべてのオブジェクトが正しく関連付けられていることを確認してください。 例えば、ジョブ記述に QMQM/QMQM 以外のジョブ・キューを指定する場合は、サブシステム、つまり QMQM に対して ADDJOBQE が必ず実行されるようにします。

次のワークシートを例として使用すると、395 ページの表 21 で説明するジョブそれぞれにジョブ記述を作成できます。

```
What is the queue manager library name? __________
Does job description AMQZXMA0 exist in the queue manager library? Yes   No
Does job description QMQMJOBD exist in the queue manager library? Yes   No
Does job description AMQZXMA0 exist in the QMQM library?     Yes   No
Does job description QMQMJOBD exist in the QMQM library?     Yes   No
```

これらの質問の回答がすべて「いいえ」である場合、QMQM ライブラリーにグローバル・ジョブ記述 QMQMJOBD を作成します。

IBM MQ メッセージ・キュー

IBM MQ メッセージ・キュー QMQMMSG は、それぞれのキュー・マネージャー・ライブラリーで作成されます。 キュー・マネージャーのジョブが終了して IBM MQ がメッセージをキューに送信するとき、オペレーティング・システムのメッセージがこのキューに送信されます。 例えば、どのジャーナル・レシーバーが始動時に必要とされるかを報告します。 モニターを容易にするために、このメッセージ・キューに入るメッセージの数を管理可能な数に制限しておきます。

IBM i IBM i でのデフォルトのシステム例

次の例は、標準的ないくつかのジョブをキュー・マネージャーの起動時に実行依頼した場合に、未変更の IBM MQ インストール済み環境がどのように動作するかを示しています。

まず、AMQZXMA0 実行制御プログラム・ジョブが開始します。

1. **STRMQM** コマンドを、キュー・マネージャー TESTQM に対して発行します。
2. IBM MQ は、キュー・マネージャー・ライブラリー QMTESTQM から、まずジョブ記述 AMQZXMA0 を、次にジョブ記述 QMQMJOBD を探します。

   このどちらのジョブ記述も存在しないので、IBM MQ は製品ライブラリー QMQM から、ジョブ記述 AMQZXMA0 を探します。 このジョブ記述は存在するので、それがジョブの実行依頼に使用されます。
3. このジョブ記述は IBM MQ のデフォルト・ジョブ・キューを使用するので、ジョブはジョブ・キュー QMQM/QMQM に実行依頼されます。
4. AMQZXMA0 ジョブ記述上の経路指定データは QMQMRUN20 であるので、システムはこのデータと一致するものをサブシステムの経路指定項目から検索します。

   デフォルトでは、シーケンス番号 9900 の経路指定項目には、QMQMRUN20 と一致する比較データがあるので、ジョブはこの経路指定項目上で定義されている、これも QMQMRUN20 と呼ばれるクラスで開始されます。
5. QMQM/QMQMRUN20 クラスは、実行優先順位が 20 に設定されているので、AMQZXMA0 ジョブはサブシステム QMQM において、システム上の最も対話的なジョブと同じ優先順位で実行されます。

IBM i IBM i でのワーク・マネジメントの構成の例

この情報を使用して、IBM MQ ジョブ記述を変更および作成して、IBM MQ ジョブの実行時属性を変更する方法について学習します。

IBM MQ ワーク・マネジメントの柔軟性は、IBM MQ でジョブ記述を検索するための、以下の 2 層的な方法によるものです。

- キュー・マネージャー・ライブラリーでジョブ記述を作成または変更する場合、これらの変更は QMQM 内のグローバル・ジョブ記述を指定変更しますが、その変更はローカルであり、その特定のキュー・マネージャーだけに影響を与えます。
- グローバル・ジョブ記述を QMQM ライブラリーで作成または変更する場合、それらのジョブ記述は、 個々のキュー・マネージャーをローカルに指定変更していない限り、システム上のすべての キュー・マネージャーに影響を与えます。

1. 次の例では、個々のキュー・マネージャーに対するチャネル制御ジョブの優先順位を高くしています。

   リポジトリー管理プログラム AMQRRMFA およびチャネル開始プログラム RUNMQCHI を、キュー・マネージャー TESTQM に対してできるだけ速く実行するには、次のステップを実行します。

   a. QMQM/QMQMJOBD ジョブ記述のローカル複製を、キュー・マネージャー・ライブラリー内で制御する IBM MQ プロセスの名前で作成します。 以下に例を示します。

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
NEWOBJ(RUNMQCHI)
```

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
NEWOBJ(AMQRRMFA)
```

b. ジョブ記述上の経路指定データ・パラメーターを変更して、このジョブが必ず QMQMRUN20 クラスを使用するようにします。

```
CHGJOBD JOBD(QMTESTQM/RUNMQCHI) RTGDTA('QMQMRUN20')
CHGJOBD JOBD(QMTESTQM/AMQRRMFA) RTGDTA('QMQMRUN20')
```

ここまでで、キュー・マネージャー TESTQM に対する AMQRRMFA および RUNMQCHI は、次のようになります。

- キュー・マネージャー・ライブラリー内の新しいローカル・ジョブ記述を使用します。
- 優先順位 20 で実行します。 これは、ジョブがサブシステムに入ると、QMQMRUN20 クラスが使用されるからです。

2. 次の例では、QMQM サブシステムに新規の実行優先順位クラスを定義しています。

a. 次のコマンドを発行して、QMQM ライブラリーに複製クラスを作成し、他のキュー・マネージャーがこのクラスにアクセスできるようにします。

```
CRTDUPOBJ OBJ(QMQMRUN20) FROMLIB(QMQM) OBJTYPE(*CLS) TOLIB(QMQM)
NEWOBJ(QMQMRUN10)
```

b. 次のコマンドを発行して、クラスを変更し、新規の実行優先順位を持たせます。

```
CHGCLS CLS(QMQM/QMQMRUN10) RUNPTY(10)
```

c. 次のコマンドを発行して、新規のクラス定義をサブシステムに追加します。

```
ADDRTGE SBSD(QMQM/QMQM) SEQNBR(8999) CMPVAL('QMQMRUN10') PGM(QSYS/QCMD)
CLS(QMQM/QMQMRUN10)
```

注 : 経路指定シーケンス番号には任意の数値を指定できますが、この値は連続していなければなりません。 このシーケンス番号により、サブシステムが一致する経路指定データを探して経路指定項目を検索する順序が指定されます。

d. 次のコマンドを発行して、新規の優先順位クラスを使用するローカルまたはグローバル・ジョブ記述を変更します。

```
CHGJOBD JOBD(QMQMlibname/QMQMJOBD) RTGDTA('QMQMRUN10')
```

ここで、QMlibraryname に関連付けられたすべてのキュー・マネージャーが、実行優先順位 10 を使用します。

3. 次の例は、固有のサブシステムのキュー・マネージャーで実行されます。

キュー・マネージャー TESTQM に対するすべてのジョブが QBATCH サブシステムで実行されるようにするには、次のステップを実行します。

a. キュー・マネージャー・ライブラリー内に QMQM/QMQMJOBD ジョブ記述のローカル複製を、コマンドで作成します。

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
```

b. ジョブ記述上のジョブ・キュー・パラメーターを変更して、このジョブが必ず QBATCH ジョブ・キューを使用するようにします。

```
CHGJOBD JOBD(QMTESTQM/QMQMJOBD) JOBQ(*LIBL/QBATCH)
```

注 : ジョブ・キューは、サブシステム記述に関連付けられています。 このジョブがジョブ・キューにとどまっていることがわかった場合は、ジョブ・キュー定義が SBSD に定義されていることを確認してください。 サブシステム用の DSPSBSD コマンドを使用して、オプション 6 ジョブ・キュー項目を選択します。

ここまでで、キュー・マネージャー TESTQM に対するすべてのジョブは、次のようになっています。

- キュー・マネージャー・ライブラリー内の新しいデフォルトのローカル・ジョブ記述を使用します。
- ジョブ・キュー QBATCH に実行依頼されます。

ジョブが正しく経路指定されて優先順位付けされるようにするには、次のいずれかを実行できます。

- IBM MQ ジョブの経路指定項目をサブシステム QBATCH に作成します。
- どの経路指定データを使用するかに関係なく、QCMD を呼び出す全キャッチ経路指定項目を利用します。

  このオプションは、ジョブ・キュー QBATCH に対する最大活動ジョブ・オプションが *NOMAX に設定されている場合だけ有効です。 システム・デフォルトは 1 です。

4. 次の例では、もう 1 つの IBM MQ サブシステムが作成されます。
   a. 次のコマンドを発行して、QMQM ライブラリーに複製サブシステムを作成します。

   ```
   CRTDUPOBJ OBJ(QMQM) FROMLIB(QMQM) OBJTYPE(*SBSD) TOLIB(QMQM) NEWOBJ(QMQM2)
   ```

   b. 次のコマンドを発行して、QMQM ジョブ・キューを除去します。

   ```
   RMVJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM)
   ```

   c. 次のコマンドを発行して、新規のジョブ・キューをサブシステムに作成します。

   ```
   CRTJOBQ JOBQ(QMQM/QMQM2) TEXT('Job queue for IBM MQ Queue Manager')
   ```

   d. 次のコマンドを発行して、ジョブ・キュー項目をサブシステムに追加します。

   ```
   ADDJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM2) MAXACT(*NOMAX)
   ```

   e. 次のコマンドを発行して、キュー・マネージャー・ライブラリーに複製 QMQMJOBD を作成します。

   ```
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMlibraryname)
   ```

   f. 次のコマンドを発行して、新規のジョブ・キューを使用するジョブ記述を変更します。

   ```
   CHGJOBD JOBD(QMlibraryname/QMQMJOBD) JOBQ(QMQM/QMQM2)
   ```

   g. 次のコマンドを発行して、サブシステムを始動します。

   ```
   STRSBS SBSD(QMQM/QMQM2)
   ```

注 :

a. サブシステムは、任意のライブラリーに指定できます。 何らかの理由で製品が再インストールされた場合、または QMQM ライブラリーが置き換えられた場合、それまでに行った変更内容が除去されます。

b. ここで、QMlibraryname に関連付けられたすべてのキュー・マネージャーのジョブが、サブシステム QMQM2 の下で実行されます。

IBM i IBM i での可用性、バックアップ、回復、および再始動

この情報は、IBM MQ for IBM i がバックアップおよび復元計画を支援するために IBM i ジャーナル処理サポートを使用する方法について理解するのに使用します。

このセクションを読む前に、IBM i の標準的なバックアップおよび回復の方法と、IBM i でのジャーナルおよびその関連ジャーナル・レシーバーの使用について理解しておくことが必要です。 これらのトピックについては、バックアップおよび回復を参照してください。

バックアップおよび回復の方法を理解するためには、まず IBM MQ for IBM i がそのデータを IBM i ファイル・システムおよび統合ファイル・システム (IFS) 内で編成する方法について理解しておく必要があります。

IBM MQ for IBM i はそのデータを、各キュー・マネージャー・インスタンスの個々のライブラリーや、IFS ファイル・システム内のストリーム・ファイルに保持します。

キュー・マネージャーの固有ライブラリーには、キュー・マネージャーのワーク・マネージメントを 制御するために必要な、ジャーナル、ジャーナル・レシーバー、およびオブジェクトが入っています。 IFS ディレクトリーおよびファイルには、IBM MQ 構成ファイル、IBM MQ オブジェクトの記述、およびそれらに含まれるデータが入っています。

これらのオブジェクトへの変更で、システム障害の後で回復可能なものは、いずれも該当するオブジェクトに適用される前 にジャーナルに記録されます。 この方法には、ジャーナルに記録された情報を再生することによって、こうした変更を回復できるという効果があります。

異なるサーバー上で複数のキュー・マネージャー・インスタンスを使用するように IBM MQ for IBM i を構成することで、キュー・マネージャーの可用性を高め、サーバーまたはキュー・マネージャーで障害が発生した場合の回復を速めることができます。

IBM i キュー・マネージャー・ジャーナル (IBM i)

この情報は、IBM MQ for IBM i が、操作にジャーナルを使用し、ローカル・オブジェクトに対する更新を制御する方法について理解するために使用します。

各キュー・マネージャー・ライブラリーにはそのキュー・マネージャーのジャーナルが含まれ、ジャーナルの名前は QM *GRLIB*/AMQ *A* JRN となります。ここで、 QM *GRLIB* はキュー・マネージャー・ライブラリーの名前、 *A* はキュー・マネージャー・インスタンスに固有の文字 A (単一インスタンス・キュー・マネージャーの場合) です。

QM *GRLIB* は、 QM という名前の後に、固有の形式でキュー・マネージャーの名前を付けます。 例えば、TEST という名前のキュー・マネージャーは、QMTEST という名前のキュー・マネージャー・ライブラリーを持ちます。 **CRTMQM** コマンドを使用してキュー・マネージャーを作成するときに、キュー・マネージャー・ライブラリーを指定することができます。

ジャーナルには、ジャーナル処理される情報を入れるジャーナル・レシーバーが関連付けられます。 レシーバーは、情報が一方的に追加されるオブジェクトであり、最終的には満杯になります。

ジャーナル・レシーバーは、古くなった情報で貴重なディスク・スペースを浪費してしまいます。 ただし、情報を永続ストレージに入れることにより、この問題を最小限にとどめることができます。 どの時点でも、ジャーナルにはジャーナル・レシーバーが 1 つ接続されています。 ジャーナル・レシーバーが既定のしきい値に達した場合には、切り離して新しいジャーナル・レシーバーで置き換えます。 **CRTMQM** および **THRESHOLD** パラメーターを使用してキュー・マネージャーを作成するときには、ジャーナル・レシーバーのしきい値を指定できます。

ローカル IBM MQ for IBM i ジャーナルに関連付けられたジャーナル・レシーバーは、各キュー・マネージャー・ライブラリーに存在し、次のような命名規則が適用されます。

```
AMQ Arnnnnn
```

説明:

A
文字 A-Z を示します。 単一インスタンスのキュー・マネージャーの場合、これは A になります。 これは、複数インスタンスのキュー・マネージャーのインスタンスによって異なります。

nnnnn
シーケンス内の次のジャーナルに対して 1 ずつ増分される 10 進数 `00000 to 99999` です。

r
10 進数の `0 to 9` で、レシーバーが復元されるたびに 1 ずつ増分されます。

ジャーナルの順序は日付に基づいています。 ただし、次のジャーナルは次の規則に基づいて指定されます。

1. AMQArnnnnn は AMQAr(nnnnn+1) になり、99999 に達すると nnnnn は折り返します。 例えば、AMQA099999 は AMQA000000 になり、AMQA999999 は AMQA900000 になります。
2. 規則 1 によって生成された名前を持つジャーナルが既に存在する場合、メッセージ CPI70E3 が QSYSOPR メッセージ・キューに送信され、自動レシーバー切り替えは停止します。

   現在接続されているレシーバーは、問題を調べて新しいレシーバーを手動で接続するまで引き続き使用されます。
3. 新しい名前を順序どおりに使用できない (つまり、すべてのジャーナル名がシステム上にある) 場合は、次の両方を実行する必要があります。

   a. 必要ではないジャーナルを削除する (407 ページの『IBM i でのジャーナル管理』を参照)。

   b. (**RCDMQMIMG**) を使用してジャーナルの変更を最後のジャーナル・レシーバーに記録し、次に前の手順を繰り返す。 このようにすると、古いジャーナル・レシーバーの名前を再使用できます。

AMQAJRN ジャーナルは MNGRCV(*SYSTEM) オプションを使用して、しきい値に達した場合に、オペレーティング・システムがジャーナル・レシーバーを自動的に変更できるようにします。 システムによるレシーバー管理の詳細については、「*IBM i* バックアップおよび回復の手引き」を参照してください。

ジャーナル・レシーバーのデフォルトのしきい値は、100,000 KB です。 キュー・マネージャーを作成するときに、これより大きな値に設定できます。 **ログ受信側のサイズ** 属性の初期値は、 `mqs.ini` ファイルの **ログ・デフォルト** スタンザに書き込まれます。

ジャーナル・レシーバーが指定されたしきい値を超えて拡張されると、レシーバーは切り離され、新しいジャーナル・レシーバーが作成されて、前のレシーバーから属性を継承します。 キュー・マネージャーが作成された後の LogReceiverSize 属性または LogASP 属性の変更は、システムが自動的に新しいジャーナル・レシーバーを接続するときに無視されます。

システムの構成について詳しくは、 Multiplatforms での IBM MQ 構成情報の変更 を参照してください。

キュー・マネージャーの作成後にジャーナル・レシーバーのサイズを変更したい場合には、新しいジャーナル・レシーバーを作成して、次のコマンドを使用してその所有者を QMQM に設定する必要があります。

```
CRTJRNRCV JRNRCV(QM GRLIB/AMQ Arnnnnn) THRESHOLD(xxxxxx) +
TEXT('MQM LOCAL JOURNAL RECEIVER')
CHGOBJOWN OBJ(QM GRLIB/AMQ Arnnnnn) OBJTYPE(*JRNRCV) NEWOWN(QMQM)
```

説明:

QMGRLIB
ご使用のキュー・マネージャー・ライブラリーの名前

A
インスタンス ID です (通常は A)。

rnnnnn
上記の命名順序内の次のジャーナル・レシーバー。

xxxxxx
新しいレシーバーのしきい値 (KB 単位)

注: レシーバーの最大サイズは、オペレーティング・システムによって決まります。 この値を確認するには、**CRTJRNRCV** コマンドの THRESHOLD キーワードを調べます。

ここで、次のコマンドを使用して、新しいレシーバーを AMQAJRN ジャーナルに付加します。

```
CHGJRN JRN(QMGRLIB/AMQ A JRN) JRNRCV(QMGRLIB/AMQ Annnnnn)
```

これらのジャーナル・レシーバーの管理方法の詳細については、407 ページの『IBM i でのジャーナル管理』を参照してください。

IBM i キュー・マネージャー・ジャーナルの使用 (IBM i)

この情報は、IBM MQ for IBM i が、操作にジャーナルを使用し、ローカル・オブジェクトに対する更新を制御する方法について理解するために使用します。

メッセージ・キューに対する持続更新は、2 段階で行われます。 まず更新を表すレコードがログに書き込まれ、その後にキュー・ファイルが更新されます。

したがって、ジャーナル・レシーバーの方が、キュー・ファイルよりも最新のものになる可能性があります。 再始動処理が確実に整合点から開始されるようにするために、IBM MQ はチェックポイントを使用します。

チェックポイントとは、ジャーナルに記述されているレコードとキューのレコードが一致する時点 (ポイント) です。 チェックポイント自体は、キュー・マネージャーを再始動するために必要な一連のジャーナル・レコードです。 例えば、チェックポイントの時点でアクティブであったすべてのトランザクション (つまり作業単位) の状態です。

チェックポイントは、IBM MQ によって自動的に生成されます。 チェックポイントは、キュー・マネージャーの開始時、シャットダウン時、および特定回数の操作がログに記録されるたびに生成されます。

次のようにして、キュー・マネージャーのすべてのオブジェクトに対して RCDMQMIMG コマンドを発行し、その結果を表示することによって、キュー・マネージャーがチェックポイントを取るように強制できます。

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME) DSPJRNDTA(*YES)
```

キューが取り扱うメッセージが増えるにつれて、チェックポイント・レコードは、キューの現在の状態と一致しなくなります。

IBM MQ は再始動時に、ログ内の最新チェックポイント・レコードを見つけます。 その情報は、すべてのチェックポイントの最後に更新される チェックポイント・ファイルに保持されています。 チェックポイント・レコードは、ログとデータの間の最新の整合点を表すものです。 このチェックポイントのデータは、チェックポイント時に存在していたとおりのキューを再作成するのに使用されます。 キューが再作成されると、システムの障害時以前または停止時以前の状態にキューを戻すために、ログを作動させます。

IBM MQ でジャーナルがどのように使用されるかを理解するために、キュー・マネージャー TEST 内の TESTQ というローカル・キューについて考えてみましょう。 これは IFS ファイルで次のように表現されます。

```
/QIBM/UserData/mqm/qmgrs/TEST/queues
```

指定されたメッセージがこのキューに書き込まれた後でキューから取り出される場合に行われる処理を、404 ページの図 22 に示します。

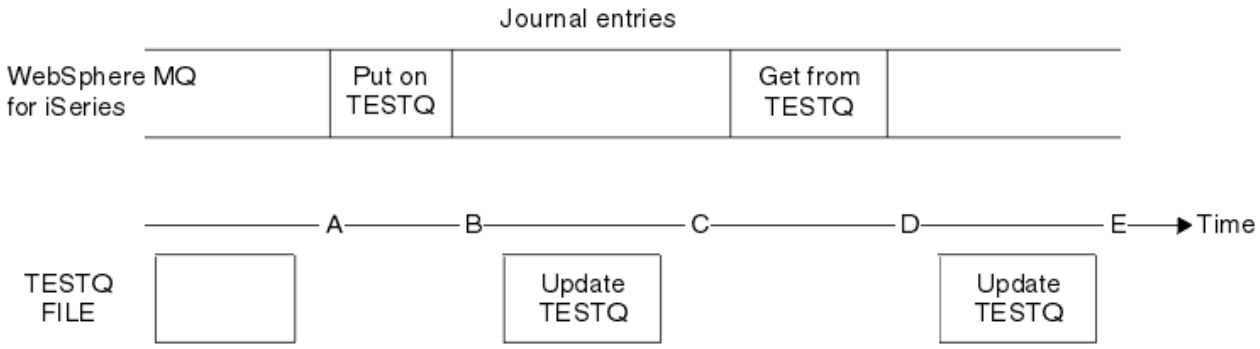


図 22. *MQM* オブジェクトの更新時のイベントのシーケンス

図の A から E までの 5 つの点は、次の状態を定義する時点を表します。

A
IFS ファイルによるキューの表現は、ジャーナルに入っている情報と整合性があります。

B
ジャーナル項目は、キュー上の Put 操作を定義するジャーナルに書き込まれます。

C
該当する更新がキューに行われます。

D
ジャーナル項目は、キューからの Get 操作を定義するジャーナルに書き込まれます。

E
該当する更新がキューに行われます。

IBM MQ for IBM i の主要な回復機能として、ユーザーは A の時点で TESTQ の IFS ファイル表現を保存し、後に E の時点で TESTQ の IFS ファイル表現を回復することができます (保存されたオブジェクトを復元し、ジャーナル内の A の時点以降の項目を再生することによって)。

この方法は、システム障害の後で持続メッセージを回復するときに、IBM MQ for IBM i で使用されます。 IBM MQ はジャーナル・レシーバー内の特定項目を記憶していて、始動時にこの時点以降のジャーナル内の項目を再生できるようにします。 この始動項目は定期的に再計算されるので、IBM MQ は次回の始動時に必要最小限の再生を行うだけですみます。

IBM MQ ではオブジェクトを個別に回復することができます。 オブジェクトに関連するすべての持続情報が、ローカル IBM MQ for IBM i ジャーナルに記録されます。 損傷または破損した IBM MQ オブジェクトはいずれも、ジャーナルに保持されている情報から完全に再作成できます。

システムによるレシーバー管理の詳細については、「401 ページの『IBM i での可用性、バックアップ、回復、および再始動』」を参照してください。

IBM i *IBM i でのメディア・イメージ*

IBM i では、メディア・イメージは、ジャーナルに記録されている IBM MQ オブジェクトの完全なコピーです。 破損または損傷している一部のオブジェクトは、メディア・イメージから自動的に回復できます。

長期間存続する IBM MQ オブジェクトでは、作成された時点までさかのぼると、ジャーナル項目が膨大な数になることがあります。 これを避けるために、IBM MQ for IBM i にはオブジェクトのメディア・イメージという概念があります。

このメディア・イメージは、ジャーナルに記録されている IBM MQ オブジェクトの完全なコピーです。 オブジェクトのイメージがとられている場合、そのオブジェクトは、このイメージ以降のジャーナル項目を再生して再作成できます。 各 IBM MQ オブジェクトの再生点を表すジャーナル項目をメディア回復項目と呼びます。 IBM MQ では、以下の追跡が行われます。

- 各キュー・マネージャー・オブジェクトのメディア回復項目。
- このセット内の最も古い項目 (詳細については、407 ページの『IBM i でのジャーナル管理』を参照してください)。

*CTLG オブジェクトおよび *MQM オブジェクトはキュー・マネージャーの再始動に必要とされるので、これらのイメージは定期的に取り込まれます。

その他のオブジェクトのイメージは適宜取り込まれます。 デフォルトでは、すべてのオブジェクトのイメージは、パラメーター ENDCCTJOB(*YES) 付きの **ENDMQM** コマンドを使用してキュー・マネージャーがシャットダウンされるときに取り込まれます。 この操作には、非常に大きなキュー・マネージャーの場合、かなりの時間がかかります。 迅速にシャットダウンする必要がある場合は、パラメーター RCDMQMIMG(*NO) を ENDCCTJOB(*YES) 付きで 指定します。 このような場合、キュー・マネージャーの再始動後に、完全なメディア・イメージをジャーナルに記録することをお勧めします。これには、次のコマンドを使用します。

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME)
```

IBM MQ は、ジャーナル内の小さな 1 つの項目でオブジェクトを簡潔に記述できる点が見つかると、自動的にオブジェクトのイメージを記録します。 しかし、多数のメッセージを常に含んでいるキューのように、このような点が決して見つからないオブジェクトもあります。

最も古いメディア回復項目の日付を不必要に長い期間継続させるのではなく、 IBM MQ コマンド RCDMQMIMG を使用して、選択したオブジェクトのイメージを手動で取ることができます。

メディア・イメージによる回復

IBM MQ は、いくつかのオブジェクトが壊れているか損傷していることを検出すると、メディア・イメージからそれらを自動的に回復します。 特にこれは、通常のキュー・マネージャーの始動の一部である、特殊 *MQM および *CTLG オブジェクトに適用されます。 キュー・マネージャーが最後に終了されたときに未完了の同期点トランザクションがあった場合は、始動操作を完了するために、影響を受けたキューも自動的に回復されます。

他のオブジェクトは、IBM MQ コマンド RCRMQMOBJ を使用して手動で回復する必要があります。 このコマンドにより、IBM MQ オブジェクトを再作成するために、ジャーナル内の項目が再生されます。 IBM MQ オブジェクトが損傷した場合、有効な処置は、オブジェクトを削除するか、またはこの方法で再作成するかしかありません。 ただし、非持続メッセージは、この方法では回復できないことに注意してください。

IBM i チェックポイント *(IBM MQ for IBM i)*

さまざまな時刻でチェックポイントは取られ、回復の場合に既知で整合性のある開始点を提供します。

以下のポイントで、プロセス AMQZMUC0 内のチェックポイント・スレッドがチェックポイントを取ります。

- キュー・マネージャーの始動 (STRMQM)。
- キュー・マネージャーのシャットダウン (ENDMQM)。
- 最後のチェックポイント以降で、ある時間が経過したとき (デフォルトの時間は 30 分) および直前のチェックポイント以降に最小ログ・レコード数 (デフォルト値は 100) が書き込まれたとき。
- ある数のログ・レコードが書き込まれた後。 デフォルト値は 10,000 です。
- ジャーナルのサイズがしきい値を超過し、新規ジャーナル・レシーバーが自動的に作成された後。
- 次の指定により、メディアの完全なイメージが取られたとき。

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME) DSPJRNDTA(*YES)
```

IBM i IBM MQ for IBM i データのバックアップ

この情報は、それぞれのキュー・マネージャーの 2 つのタイプの IBM MQ バックアップを理解するために使用します。

それぞれのキュー・マネージャーについて考慮すべき IBM MQ バックアップには、次の 2 種類があります。

- データおよびジャーナル・バックアップ。

データの両方のセットが一貫性を保つように、このバックアップはキュー・マネージャーを終了した後にのみ行います。

- ジャーナル・バックアップ。

  このバックアップは、キュー・マネージャーがアクティブになっているときに行います。

どちらの方法も、キュー・マネージャーの IFS ディレクトリーとキュー・マネージャー・ライブラリーの名前が必要です。 これらの名前は IBM MQ 構成ファイル (mqs.ini) にあります。 詳しくは、Multiplatforms での IBM MQ 構成情報の変更を参照してください。

どちらのタイプのバックアップも、次の手順を使用します。

特定のキュー・マネージャーのデータおよびジャーナル・バックアップ

注 : キュー・マネージャーの実行中は save-while-active 要求を使用しないでください。 保留中の変更があるコミットメント定義がすべてコミットまたはロールバックされなければ、この要求は完了できません。 キュー・マネージャーがアクティブになっているときにこのコマンドを使用すると、チャネル接続が異常終了することがあります。 必ず、以下の手順を使用してください。

1. 次のコマンドを使用して、空のジャーナル・レシーバーを作成します。

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. **RCDMQMIMG** コマンドを使用して、すべての IBM MQ オブジェクトの MQM イメージを記録し、次のコマンドを使用してチェックポイントを強制します。

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. チャネルを終了し、キュー・マネージャーが実行中でないことを確認します。 キュー・マネージャーが実行されている場合は、**ENDMQM** コマンドでそれを停止してください。
4. キュー・マネージャー・ライブラリーを、次のコマンドを発行してバックアップします。

```
SAVLIB LIB(QMTEST)
```

5. キュー・マネージャーの IFS ディレクトリーを、次のコマンドを発行してバックアップします。

```
SAV DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/test'))
```

特定のキュー・マネージャーのジャーナル・バックアップ

ある時点で全保管を実行すれば、関係情報はすべてジャーナルに保持されているので、ジャーナル・レシーバーを保管して部分バックアップを実行できます。 部分バックアップでは全バックアップ以降のすべての変更が記録されます。実行するには、次のコマンドを発行します。

1. 次のコマンドを使用して、空のジャーナル・レシーバーを作成します。

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. **RCDMQMIMG** コマンドを使用して、すべての IBM MQ オブジェクトの MQM イメージを記録し、次のコマンドを使用してチェックポイントを強制します。

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 次のコマンドを使用して、ジャーナル・レシーバーを保管します。

```
SAVOBJ OBJ(AMQ*) LIB(QMTEST) OBJTYPE(*JRNRCV) .........
```

シンプルなバックアップ方法としては、IBM MQ ライブラリーの全バックアップを毎週実行し、ジャーナル・バックアップを毎日実行する方法があります。 これは、言うまでもなく、貴社でセットアップしたバックアップ方法次第です。

IBM i *IBM i でのジャーナル管理*

バックアップ作業の一部として、ジャーナル・レシーバーを処理します。 以下のようなさまざまな理由により、IBM MQ ライブラリーからのジャーナル・レシーバーの削除が役立ちます。

- スペースの解放のため ; すべてのジャーナル・レシーバーに適用されます。
- 始動 (STRMQM) 時のパフォーマンスの向上のため
- オブジェクトの再作成 (RCRMQMOBJ) 時のパフォーマンスの向上のため

ジャーナル・レシーバーを削除する前に、バックアップ・コピーがあること、ジャーナル・レシーバーをもう必要としないことに注意する必要があります。

ジャーナル・レシーバーはジャーナルから切り離した後 で、キュー・マネージャー・ライブラリーから除去して、保存できます。 ただし、回復操作が必要になった場合に、復元に使用できるように保存することが必要です。

ジャーナル管理の概念を 407 ページの図 23 に示します。

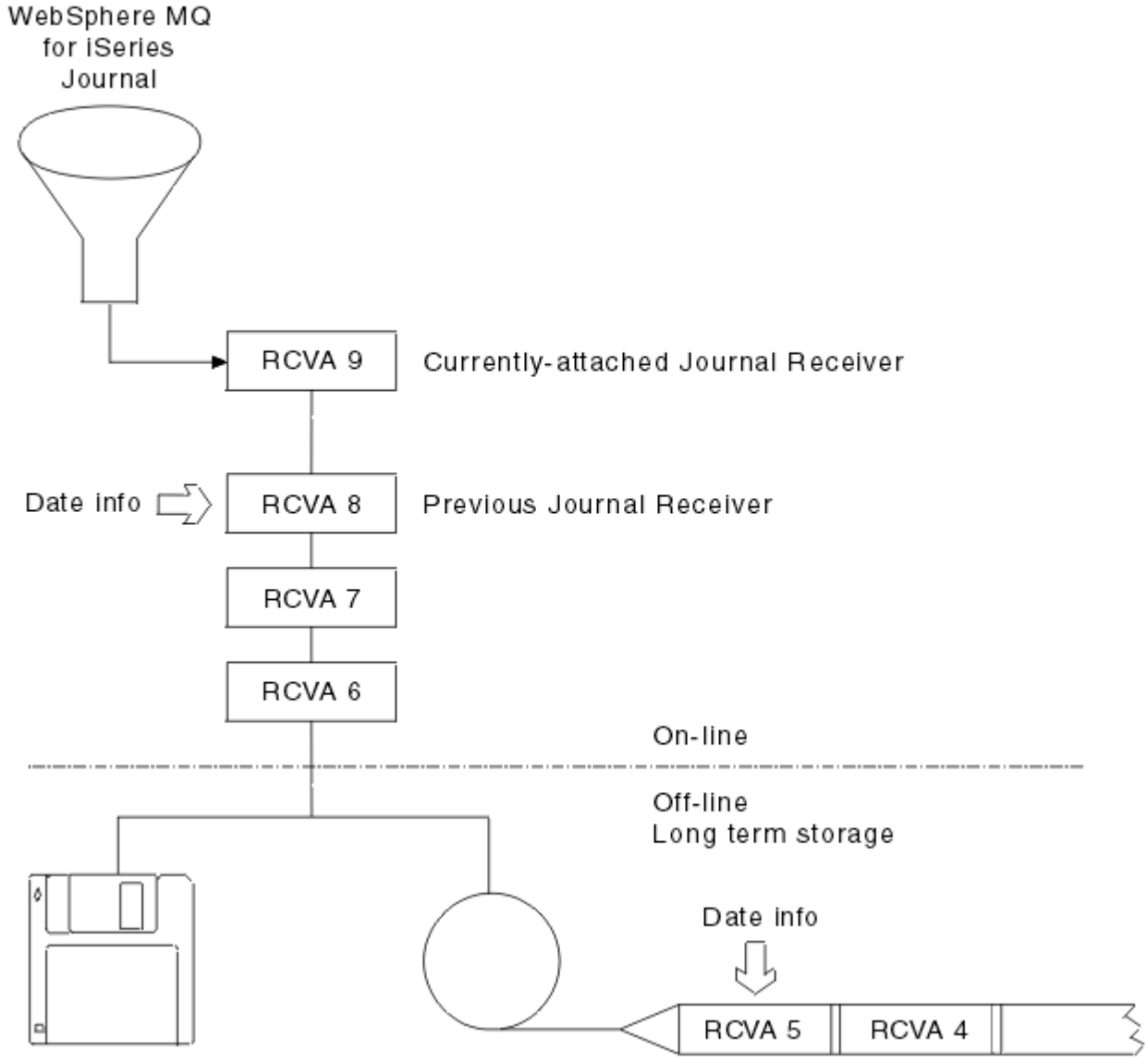


図 *23. ジャーナリング (IBM i)*

バックアップされたジャーナル・レシーバーをキュー・マネージャー・ライブラリーから除去できる時点と、バックアップ自体を廃棄できる時点を判別するには、IBM MQ がジャーナルをどの程度までさかのぼる必要があるかを知ることが重要です。

この時点を判別するために、IBM MQ は 2 つのメッセージをキュー・マネージャー・メッセージ・キュー (キュー・マネージャー・ライブラリー内の QMQMMSG) に対して出します。 これらのメッセージは、始動時、ローカル・ジャーナル・レシーバーの変更時、および RCDMQIMG を使用したチェックポイントの強制時に発行されます。 2 つのメッセージは次のとおりです。

AMQ7460
始動回復点。 このメッセージは、始動回復パスのイベントにおいて IBM MQ がジャーナルを再生する開始点となる始動項目の日時を定義します。 このレコードが入っているジャーナル・レシーバーが IBM MQ ライブラリーで使用できる場合は、このレコードが入っているジャーナル・レシーバーの名前もメッセージに示されます。

AMQ7462
最も古いメディア回復項目。 このメッセージは、メディア・イメージからオブジェクトを再作成するために使用できる最も古い項目の日時を定義します。

識別されるジャーナル・レシーバーは、必要なものの中の最も古いものです。 作成日がそれより古い他の IBM MQ ジャーナル・レシーバーは不要になります。 星印が表示される場合のみ、示される日付からどれが最も古いジャーナル・レシーバーかを判断して、バックアップを復元する必要があります。

これらのメッセージがログに記録されるとき、IBM MQ は、システム上に保持する必要のある最も古いジャーナル・レシーバーの名前のみを含むユーザー・スペース・オブジェクトもキュー・マネージャー・ライブラリーに書き込みます。 このユーザー・スペースは AMQJRNINF と呼ばれ、データは次の形式で書き込まれます。

```
JJJJJJJJJJLLLLLLLLLLYYYYMMDDHHMMSSmmm
```

ここで、

JJJJJJJJJJ
IBM MQ がまだ必要としている最も古いレシーバーの名前。

LLLLLLLLLL
ジャーナル・レシーバー・ライブラリーの名前。

YYYY
IBM MQ が必要とする最も古いジャーナル項目の年。

MM
IBM MQ が必要とする最も古いジャーナル項目の月。

DD
IBM MQ が必要とする最も古いジャーナル項目の日。

HH
IBM MQ が必要とする最も古いジャーナル項目の時刻。

SS
IBM MQ が必要とする最も古いジャーナル項目の秒。

mmm
IBM MQ が必要とする最も古いジャーナル項目のミリ秒。

最も古いジャーナル・レシーバーがシステムから削除されると、このユーザー・スペースのジャーナル・レシーバーにはアスタリスク (*) が付けられます。

注: RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) を定期的に実行すると、IBM MQ の起動時間を節約でき、回復のために保管および復元する必要があるローカル・ジャーナル・レシーバーの数を減らすことができます。

IBM MQ for IBM i は、始動、またはオブジェクトの再作成のいずれかのために回復パスを実行する場合以外は、ジャーナル・レシーバーを参照しません。 必要なジャーナルがないことが判明すると、これはメッセージ AMQ7432 を、キュー・マネージャー・メッセージ・キュー (QMQMMSG) に対して出して、回復パスを完了するために必要なジャーナル項目の日時を報告します。

これが起こった場合、この日付以後に切り離されたジャーナル・レシーバーをすべてバックアップから復元して、回復パスの続行を可能にしてください。

始動エントリーが入っているジャーナル・レシーバーと、それ以降のすべてのジャーナル・レシーバーは、キュー・マネージャー・ライブラリー内で使用可能にしてください。

最も古い Media Recovery Entry を含むジャーナル・レシーバー、およびそれ以降のすべてのジャーナル・レシーバーを常に使用可能にして、キュー・マネージャー・ライブラリーまたはバックアップに存在するようにします。

チェックポイントを強制するときは、次の操作をしてください。

- AMQ7460 というジャーナル・レシーバーが拡張されていない場合、これはコミットまたはロールバックの必要があるが、未完了の作業単位があることを示します。
- AMQ7462 というジャーナル・レシーバーが拡張されていない場合、損傷したオブジェクトが 1 つ以上あることを示します。

IBM i *IBM i でのキュー・マネージャー (データおよびジャーナル) 全体の復元*

この情報を使用して、1 つ以上のキュー・マネージャーをバックアップまたはリモート・マシンから復元します。

1 つ以上の IBM MQ キュー・マネージャーをバックアップから回復する必要がある場合は、以下の手順に従ってください。

1. IBM MQ キュー・マネージャーを静止します。
2. 最後のバックアップ・セット (最新の全バックアップと、その後でバックアップされたジャーナル・レシーバーからなる) を見つけます。
3. 次のコマンドを発行して、全バックアップからの RSTLIB 操作を実行し、IBM MQ データ・ライブラリーを全バックアップ時の状態に復元します。

   ```
   RSTLIB LIB(QMQRLIB1) .........
   RSTLIB LIB(QMQRLIB2) .........
   ```

   ジャーナル・レシーバーが 1 つのジャーナル・バックアップでは部分保管され、それ以降のバックアップで全保管されている場合は、全保管されているものだけを復元してください。 ジャーナルは、個別に、日時順で復元します。
4. RST 操作を実行して、IBM MQ IFS ディレクトリーを IFS ファイル・システムに復元するには、次のコマンドを使用します。

   ```
   RST DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/testqm')) ...
   ```

5. メッセージ・キュー・マネージャーを開始します。 これで、全バックアップ以降に書き込まれたジャーナル・レコードがすべて再生され、すべての IBM MQ オブジェクトがジャーナル・バックアップ時の整合状態に復元されます。

キュー・マネージャー全体を別のマシンに復元する場合、次の手順を使用して、キュー・マネージャー・ライブラリーの全内容を復元します。 (サンプルのキュー・マネージャー名として TEST を使用します。)

1. CRTMQM TEST
2. DLTLIB LIB(QMTEST)
3. RSTLIB SAVLIB(QMTEST) DEV(*SAVF) SAVF(QMGRLIBSAV)
4. 以下の IFS ファイルを削除します。

   ```
   /QIBM/UserData/mqm/qmgrs/TEST/QMQMCHKPT
   /QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMQMOBJCAT
   /QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMANAGER
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.AUTH.DATA.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CHANNEL.INITQ/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.COMMAND.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.REPOSITORY.QUEUE/q
   ```

```
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.TRANSMIT.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.PENDING.DATA.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.ADMIN.COMMAND.QUEUE/q
```

5. STRMQM TEST
6. RCRMQMOBJ OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(TEST)

IBM i *IBM i での特定のキュー・マネージャーに対するジャーナル・レシーバーの復元*

この情報は、ジャーナル・レシーバーを復元するための異なる方法を理解するために使用します。

除去されたレシーバーがその後の回復機能で再び必要になった場合、最も一般的に行われる処置は、バックアップされたジャーナル・レシーバーをキュー・マネージャー・ライブラリーに復元することです。

これは簡単な作業で、必要なことは次の標準 IBM i RSTOBJ コマンドを使用してジャーナル・レシーバーを復元することです。

```
RSTOBJ OBJ(QMQMDATA/AMQA000005) OBJTYPE(*JRNRCV) .........
```

1 つのレシーバーだけでなく一連のジャーナル・レシーバーの復元が必要な場合もあります。 例えば、AMQA000007 は IBM MQ ライブラリー内の最も古いレシーバーであり、 AMQA000005 と AMQA000006 の両方を復元する必要があります。

この場合、レシーバーを新しい順に個別に復元します。 これは必ず必要ではありませんが、役に立つ方法です。 厳しい条件のもとでは、復元されたジャーナル・レシーバーをジャーナルに関連付けるために IBM i のコマンド WRKJRNA が必要になる場合があります。

ジャーナルを復元するとき、システムは、ジャーナル・レシーバーの順序で名前が新しい接続済みジャーナル・レシーバーを自動的に作成します。 ただし、生成された新しい名前は、復元に必要なジャーナル・レシーバーと同じ名前である場合があります。 この問題を解決するには手動で介入する必要があります。順序どおりに新しい名前のジャーナル・レシーバーを作成したり、ジャーナル・レシーバーを復元する前に新しいジャーナルを作成することがあります。

例えば、保存したジャーナル AMQAJRN と次のジャーナル・レシーバーでの問題を考えてみましょう。

- AMQA000000
- AMQA100000
- AMQA200000
- AMQA300000
- AMQA400000
- AMQA500000
- AMQA600000
- AMQA700000
- AMQA800000
- AMQA900000

ジャーナル AMQAJRN をキュー・マネージャー・ライブラリーに復元するとき、システムは自動的にジャーナル・レシーバー AMQA000000 を作成します。 この自動的に生成したレシーバーは、復元したい既存のジャーナル・レシーバー (AMQA000000) の 1 つと対立します。したがって、復元できません。

解決方法は次のとおりです。

1. 手動で次のジャーナル・レシーバーを作成する (401 ページの『キュー・マネージャー・ジャーナル (IBM i)』 を参照)。

   ```
   CRTJRNRCV JRNRCV(QMQRLIB/AMQA900001) THRESHOLD(XXXXX)
   ```

2. ジャーナル・レシーバーを使用して手動でジャーナルを作成する。

```
CRTJRN JRN(QMGRLIB/AMQAJRN) MNGRCV(*SYSTEM) +
JRNRCV(QMGRLIB/AMQA9000001) MSGQ(QMGRLIB/AMQAJRNMSG)
```

3. ローカル・ジャーナル・レシーバー AMQA000000 - AMQA900000 を復元する。

IBM i IBM i での複数インスタンス・キュー・マネージャー

複数インスタンスのキュー・マネージャーでは、アクティブ・サーバーで障害が発生した場合にスタンバイ・サーバーに自動的に切り替えることで、可用性が向上します。 アクティブ・サーバーとスタンバイ・サーバーは同じキュー・マネージャーの複数インスタンスであり、同じキュー・マネージャー・データを共有します。 アクティブ・インスタンスで障害が発生する場合、引き継ぎ先のスタンバイにそのジャーナルを転送し、キュー・マネージャーがそのキューを再作成できるようにする必要があります。

複数インスタンスのキュー・マネージャーを実行している IBM i システムを構成し、アクティブ・キュー・マネージャー・インスタンスで障害が発生した場合に、使用しているジャーナルを引き継ぎ先のスタンバイ・インスタンスで使用できるようにします。 アクティブ・インスタンスからのジャーナルを引き継ぎ先のインスタンスで使用できるように構成タスクと管理タスクを設計することができます。 メッセージを発行しない場合、スタンバイ・ジャーナルが障害発生時点のアクティブ・ジャーナルと整合するように設計する必要があります。 設計を行う際には、整合性を維持する後続のトピックにある例で説明されている 2 つの構成のうちの 1 つを作りかえて使うことができます。

1. アクティブ・キュー・マネージャー・インスタンスを実行しているシステムからスタンバイ・インスタンスを実行しているシステムにジャーナルをミラーリングします。
2. アクティブ・インスタンスを実行しているシステムからスタンバイ・インスタンスへの転送が可能な独立補助ストレージ・プール (IASP) にジャーナルを配置します。

最初のソリューションでは、基本 ASP を使用するので、追加のハードウェアまたはソフトウェアは必要ありません。 2 番目のソリューションでは、IBM i クラスタリング・サポート (別料金の IBM i ライセンス製品 5761-SS1 オプション 41 として入手可能) を必要とする切り替え可能 IASP が必要です。

IBM i *IBM i での信頼性と可用性*

複数インスタンスのキュー・マネージャーは、アプリケーションの可用性を向上させることを目的としています。 技術的または物理的な制約は、災害復旧、バックアップ・キュー・マネージャー、および連続稼働の要求を満たすためにさまざまなソリューションが必要であることを意味します。

信頼性と可用性のために構成すると、多数の要因が取引されるため、以下の 4 つの異なる設計ポイントになります。

災害時リカバリー
すべてのローカル資産を破壊する大災害の後の復旧のために最適化されます。

IBM i の災害復旧は大抵、IASP の地理的ミラーリングに基づきます。

バックアップ
局所的な障害 (通常、人為的なエラーや予期しない技術的な問題) の後の回復のために最適化されます。

IBM MQ には、キュー・マネージャーを定期的にバックアップするバックアップ・キュー・マネージャーが備えられています。 キュー・マネージャー・ジャーナルの非同期複製を使用することで、バックアップの現行性を向上させることもできます。

可用性
予測可能な技術的障害 (サーバーやディスクの障害など) の後、ほとんど中断を感じさせることなくサービスを提供し、迅速に操作を復元するために最適化されます。

回復は通常、分単位で測定されます。回復プロセスより検出に長い時間がかかることもあります。 複数インスタンスのキュー・マネージャーは、可用性 の構成を支援します。

連続稼働
中断なしのサービスを提供するために最適化されます。

連続稼働ソリューションでは、検出の問題を解決する必要があり、ほぼ毎回、複数のシステムでの同じ作業の実行依頼が伴います。その際、最初の結果が使用されるか、正確さが最も重要な考慮事項である場合は少なくとも 2 つの結果が比較されます。

複数インスタンスのキュー・マネージャーは、可用性 の構成を支援します。 アクティブになるキュー・マネージャーのインスタンスは、一度に 1 つです。 スタンバイ・インスタンスへの切り替えは、ほんの 10 秒程度の場合もあれば、15 分以上かかることもあります。これは、システムの構成、ロード、および調整の方法によって異なります。

マルチインスタンスキューマネージャは、再接続可能なキューマネージャと併用することで IBM MQ MQI clients、アプリケーションプログラムがキューマネージャの停止を認識することなく処理を継続できるため、ほぼ無停止のサービスのように見せることができます。 自動化されたクライアントの再接続を参照してください。

IBM i *IBM i の高可用性ソリューションのコンポーネント*

複数インスタンスのキュー・マネージャーを使用する高可用性ソリューションを構成するには、キュー・マネージャー・データ用の堅固なネットワーク・ストレージ、キュー・マネージャー・ジャーナル用のジャーナル複製または堅固な IASP ストレージを提供するとともに、再始動可能キュー・マネージャー・サービスとして構成されるアプリケーションの再接続可能クライアントを使用します。

複数インスタンスのキュー・マネージャーは、キュー・マネージャーの障害が検出されると、それに反応して、別のサーバー上で別のキュー・マネージャー・インスタンスの開始を再開します。 その開始を完了するために、インスタンスは、ネットワーク・ストレージの共有キュー・マネージャー・データ、およびローカル・キュー・マネージャー・ジャーナルのコピーにアクセスできなければなりません。

高可用性ソリューションを作成するには、キュー・マネージャー・データの可用性およびローカル・キュー・マネージャー・ジャーナルの現行性を管理し、再接続可能クライアント・アプリケーションを作成するか、アプリケーションをキュー・マネージャー・サービスとしてデプロイして、キュー・マネージャーの再開時に自動的に再始動する必要があります。 IBM MQ classes for Java は自動クライアント再接続をサポートしていません。

キュー・マネージャー・データ

通常は RAID レベル 1 以上のディスクを使用して、可用性と信頼性が高い共有ネットワーク・ストレージにキュー・マネージャー・データを配置します。 ファイル・システムは、複数インスタンス・キュー・マネージャーの共有ファイル・システムに関する要件を満たしている必要があります。共有ファイル・システムの要件についての詳細は、ファイル共有システムの要件を参照してください。 ネットワーク・ファイル・システム 4 (NFS4) は、これらの要件を満たすプロトコルです。

キュー・マネージャー・ジャーナル

スタンバイ・インスタンスがそのキュー・マネージャー・データを整合した状態に復元できるように、キュー・マネージャー・インスタンスによって使用される IBM i ジャーナルを構成する必要もあります。 これは、サービスが中断されないようにするために、アクティブ・インスタンスで障害が発生した時点の状態にジャーナルを復元する必要があることを意味しています。 バックアップまたは災害復旧のソリューションとは異なり、ジャーナルを以前のチェックポイントに復元するだけでは不十分です。

ネットワーク・ストレージ上の複数の IBM i システム間でジャーナルを物理的に共有することはできません。 キュー・マネージャー・ジャーナルを障害発生時点の整合した状態に復元するには、アクティブ化された新しいインスタンスに、障害発生時にアクティブ・キュー・マネージャー・インスタンスに対してローカルだった物理ジャーナルを転送するか、または実行中のスタンバイ・インスタンスにジャーナルのミラーを保持する必要があります。 ミラーリングされたジャーナルは、障害が発生したインスタンスに属するローカル・ジャーナルとの同期が正確に維持されているリモート・ジャーナルのレプリカです。

複数インスタンスのキュー・マネージャーのジャーナルを管理する方法を設計する上で、次の 3 つの構成は開始点となります。

1. アクティブ・インスタンス ASP からスタンバイ・インスタンス ASP への同期ジャーナル複製 (ジャーナル・ミラーリング) の使用。
2. アクティブ・インスタンスからスタンバイ・インスタンス (アクティブ・インスタンスとして引き継ぐ) への、キュー・マネージャー・ジャーナルを保持するように構成された IASP の転送。
3. 同期 2 次 IASP ミラーの使用。

IBM MQ IBM i CRTMQM コマンドの iASP へのキュー・マネージャー・データの書き込みについて詳しくは、ASP オプションを参照してください。

また、 IBM Documentation の IBM i 情報の 高可用性 も参照してください。

アプリケーション

クライアントを作成し、スタンバイ・キュー・マネージャーの再開時にキュー・マネージャーに自動再接続するには、MQCONNX を使用してキュー・マネージャーにアプリケーションを接続し、**MQCNO** の「オプション」フィールドに MQCNO_RECONNECT_Q_MGR を指定します。 再接続可能クライアントを使用する 3 つのサンプル・プログラムについては高可用性のサンプル・プログラムを、クライアント・アプリケーションの回復の設計について詳しくはアプリケーションの復旧を参照してください。

IBM i *IBM i での NetServer 使用によるキュー・マネージャー・データ用ネットワーク共有の作成*

IBM i サーバー上に、キュー・マネージャー・データ保管のためのネットワーク共有を作成します。 キュー・マネージャー・インスタンスをホストすることになる 2 つのサーバーから、ネットワーク共有にアクセスするための接続をセットアップします。

始める前に

- このタスクには 3 つの IBM i サーバーが必要です。 ネットワーク共有は、サーバーのうちの 1 つ (GAMMA) において定義されています。 他の 2 つのサーバー (ALPHA および BETA) は、GAMMA に接続されています。
- 3 つのサーバーのすべてに IBM MQ をインストールします。
- System i® ナビゲーターをインストールします。System i Navigator を参照してください。

このタスクについて

- GAMMA 上にキュー・マネージャー・ディレクトリーを作成し、ユーザー・プロファイル QMQM および QMQMADM の所有権と許可を正しく設定します。 GAMMA 上に IBM MQ をインストールするなら、ディレクトリーと許可は容易に作成できます。
- System i ナビゲーターを使用することにより、GAMMA 上のキュー・マネージャー・データ・ディレクトリーの共有を作成します。
- ALPHA および BETA 上に、その共有を指すディレクトリーを作成します。

手順

1. GAMMA 上で、QMQM ユーザー・プロファイルを所有者とし、QMQMADM を 1 次グループとするキュー・マネージャー・データをホストするためのディレクトリーを作成します。

   ヒント:

   正しい許可を使用して短時間でディレクトリーを作成する信頼性の高い方法の 1 つは、GAMMA 上に IBM MQ をインストールすることです。

   GAMMA 上で IBM MQ を実行することが望ましくない場合は、後で IBM MQ をアンインストールしてください。 アンインストールの後も、/QIBM/UserData/mqm/qmgrs は、QMQM ユーザー・プロファイルを所有者とし、QMQMADM を 1 次グループとするディレクトリーとして GAMMA 上にそのまま残ります。

   このタスクでは、GAMMA 上の /QIBM/UserData/mqm/qmgrs ディレクトリーを使用して共有します。
2. System i ナビゲーターの「**接続の追加**」ウィザードを開始し、GAMMA システムに接続します。
   a) Windows デスクトップ上の **System i Navigator** ・アイコンをダブルクリックします。
   b)「**はい**」をクリックして、接続を作成します。
   c)「**接続の追加**」ウィザードの指示に従って、IBM i システムから GAMMA への接続を作成します。

      GAMMA への接続が「**マイ・コネクション**」に追加されます。

3. GAMMA 上に新規ファイル共有を追加します。
   a)「**システム Navigator** 」ウィンドウで、「My Connections/GAMMA/File Systems」の File Shares フォルダー」をクリックします。
   b)「**マイ・タスク**」ウィンドウで「**IBM i NetServer 共有の管理**」をクリックします。

      新しいウィンドウ「**IBM i NetServer - GAMMA**」がデスクトップに表示され、そこに共有オブジェクトが表示されます。
   c) Shared Objects フォルダー＞ **ファイル**＞ **ニュー**＞ **ファイル**を右クリックする。

      新しいウィンドウ、「**IBM i NetServer ファイル共有 - GAMMA**」が表示されます。
   d) 共有の名前を指定します (例えば WMQ)。
   e) アクセス制御を Read/Write に設定します。
   f) **パス名** を選択するには、前に作成した /QIBM/UserData/mqm/qmgrs ディレクトリーを参照して、**オク**をクリックします。

      IBM i NetServer ファイル共用 - GAMMA ウィンドウが閉じ、「共有オブジェクト」ウィンドウに WMQ がリストされます。
4. 共有オブジェクトのウィンドウで **WMQ** を右クリックします。 「**ファイル**」>「**許可**」をクリックします。

   オブジェクト /QIBM/UserData/mqm/qmgrs のウィンドウが開きます **Qmgrs 権限 - GAMMA**
   a) QMQM について、以下の許可がまだ設定されていない場合、それらにチェック・マークを付けます。

      Read
      Write
      Execute
      Management
      Existence
      Alter
      Reference
   b) QMQMADM について、以下の許可がまだ設定されていない場合、それらにチェック・マークを付けます。

      Read
      Write
      Execute
      Reference
   c) 権限を付与する他のユーザー・プロファイルを /QIBM/UserData/mqm/qmgrs に追加します。

      例えば、デフォルトのユーザー・プロファイル (Public) Read および Execute 権限を /QIBM/UserData/mqm/qmgrs に付与することができます。
5. GAMMA 上の /QIBM/UserData/mqm/qmgrs へのアクセス権限を付与されているすべてのユーザー・プロファイルが、 GAMMA にアクセスするサーバー上で実行されるのと同じパスワードを持っていることを確認します。

   特に、共有にアクセスすることになる他のサーバー上の QMQM ユーザー・プロファイルのパスワードが、GAMMA 上の QMQM ユーザー・プロファイルと同じであることを確認します。

   ヒント: パスワードを設定するには、 System i Navigator の My Connections/GAMMA/Users and Groups フォルダーをクリックします。 あるいは、**CHFUSRPRF** および **CHGPWD** のコマンドを使用します。

タスクの結果

共有を使用する他のサーバーから GAMMA にアクセス可能であることを確認します。 他のタスクを実行している場合は、パス /QNTC/GAMMA/WMQ を使用して、 ALPHA および BETA から GAMMA にアクセスできることを確認してください。 /QNTC/GAMMA ディレクトリーが ALPHA または BETA 上に存在しない場合は、ディレクトリーを作成する必要があります。 NetServer のドメインによっては、そのディレクトリーを作成する前に、IPL ALPHA または BETA が必要になる場合があります。

```
CRTDIR DIR('/QNTC/GAMMA')
```

ALPHA または BETA から /QNTC/GAMMA/WMQ にアクセス可能であることを確認したら、CRTMQM MQMNAME('QM1') MQMDIRP('/QNTC/GAMMA/WMQ') コマンドを発行することにより、GAMMA 上に /QIBM/UserData/mqm/qmgrs/QM1 が作成されます。

次のタスク

426 ページの『IBM i でのジャーナル・ミラーリングおよび NetServer 使用による複数インスタンス・キュー・マネージャーの作成』または 430 ページの『IBM i での NetServer およびジャーナル・ミラーリングを使用した単一インスタンス・キュー・マネージャーから複数インスタンス・キュー・マネージャーへの変換』のいずれかのタスクのステップを実行することにより、複数インスタンス・キュー・マネージャーを作成します。

IBM i フェイルオーバー・パフォーマンス *(IBM i)*

キュー・マネージャー・インスタンスで障害が発生したことを検出し、スタンバイで処理を再開するまでにかかる時間は、構成によって異なります。数十秒の場合もあれば、15 分、あるいはそれ以上の場合もあります。 高可用性ソリューションを設計およびテストする際には、パフォーマンスを最重要考慮事項とする必要があります。

ジャーナル複製、または IASP を使用するように複数インスタンスのキュー・マネージャーを構成するかどうかを決定する際、比較評価するべき利点と欠点があります。 ミラーリングにおいて、キュー・マネージャーは同期的にリモート・ジャーナルに対して書き込みを行う必要があります。 ハードウェアの観点からすればこれは必ずしもパフォーマンスに影響を及ぼしませんが、ソフトウェアの観点からすれば、リモート・ジャーナルへの書き込みは、単なるローカル・ジャーナルへの書き込みと比べて、関係するパス長さが長くなるので実行中のキュー・マネージャーのパフォーマンスがある程度低下する可能性があります。 ただし、スタンバイ・キュー・マネージャーが引き継ぐ際、障害が発生する前にアクティブ・インスタンスが保持していたリモート・ジャーナルからそのローカル・ジャーナルに同期化するときの遅延は通常、IBM i が IASP を検出し、キュー・マネージャーのスタンバイ・インスタンスを実行するサーバーに転送するためにかかる時間ほど長くはありません。 IASP の転送時間は、 10 分から 15 分ほどではなく、秒単位で完了することができます。 IASP 転送時間は、IASP がスタンバイ・システムに転送される際にオンに変更する 必要があるオブジェクトの数と、マージする必要があるアクセス・パスまたは索引のサイズによって異なります。

スタンバイ・キュー・マネージャーが引き継ぐ際、障害が発生する前にアクティブ・インスタンスが保持していたリモート・ジャーナルからそのローカル・ジャーナルに同期化するときの遅延は通常、IBM i が独立 ASP を検出し、キュー・マネージャーのスタンバイ・インスタンスを実行するサーバーに転送するためにかかる時間ほど長くはありません。 独立 ASP 転送時間は数秒では完了せず、10 分から 15 分程度かかる場合があります。 独立 ASP 転送時間は、独立 ASP がスタンバイ・システムに転送される際にオンに変更する 必要があるオブジェクトの数と、マージする必要があるアクセス・パスまたは索引のサイズによって異なります。

ただし、ジャーナルの転送だけが、スタンバイ・インスタンスが完全に再開するためにかかる時間に影響を与える要因ではありません。 開始の続行を試みるようスタンバイ・インスタンスに信号を送るキュー・マネージャー・データのロックを解放するためにネットワーク・ファイル・システムでかかる時間も考慮に入れる必要があります。 さらに、インスタンスがメッセージの処理を再開できるよう、ジャーナルからキューを回復するためにかかる時間も考慮に入れる必要があります。 これらのその他の遅延ソースは、スタンバイ・インスタンスを開始するのにかかる時間にすべて加算されます。 切り替えにかかる合計時間を構成する要素は次のとおりです。

障害検出時間
NFS がキュー・マネージャー・データのロックを解放し、スタンバイ・インスタンスがその開始プロセスを続行するためにかかる時間。

転送時間
HA クラスターの場合、アクティブ・インスタンスをホストするシステムからスタンバイ・インスタンスに IBM i が IASP を転送するためにかかる時間。ジャーナル複製の場合、リモート・レプリカのデータでスタンバイのローカル・ジャーナルを更新するためにかかる時間。

再開時間
新しくアクティブになったキュー・マネージャー・インスタンスがその復元されたジャーナルの最新チェックポイントからそのキューを再作成し、メッセージの処理を再開するためにかかる時間。

注 :

テークオーバーしたスタンバイ・インスタンスが、以前アクティブだったインスタンスと同期複製するように構成されている場合、始動において遅延が発生する可能性があります。 以前アクティブだったインスタンスをホストしていたサーバー上にリモート・ジャーナルがあり、その状態でサーバーで障害が発生した場合、新たにアクティブ化されたインスタンスは、そのリモート・ジャーナルへの複製を実行できない可能性があります。

同期応答を待機するデフォルトの待ち時間は 1 分です。 複製がタイムアウトになる前に、最大遅延時間を構成することができます。 あるいは、障害の発生したアクティブ・インスタンスへの非同期複製を使用して始動するよう、スタンバイ・インスタンスを構成することもできます。 その場合、障害の発生したインスタンスが再びスタンバイ側で実行されるようになった時点で、同期複製に切り替えます。 同じ考慮事項が、同期独立 ASP ミラーの使用にも当てはまります。

これらの構成要素に関する基本的な測定を個々に作成すると、フェイルオーバー全体にかかる時間を評価する上で役に立ち、使用する構成アプローチを決定する際に考慮に入れることができます。 最良の構成の決定を行うには、同じサーバー上の他のアプリケーションがフェイルオーバーする方法や、バックアップまたはすでに IASP を使用している災害復旧プロセスがあるかどうかも考慮に入れる必要があります。

IASP 転送時間は、クラスター構成を調整することで短縮することができます。

1. オンに変更するプロセスで UID および GID を変更しなくてもすむように、クラスター内のシステム全体のユーザー・プロファイルの GID と UID を同じにする必要があります。
2. システム内のデータベース・オブジェクトの数および基本ユーザー・ディスク・プールを最小限に抑えます。これらは、ディスク・プール・グループ用の相互参照表を作成するためにマージする必要があるからです。
3. パフォーマンスのヒントについて詳しくは、IBM Redbook「*Implementing PowerHA® for IBM i (SG24-7405)*」を参照してください。

基本 ASP を使った構成、ジャーナル・ミラーリング、および小さな構成は、数十秒程度で切り替えられます。

IBM i *IBM i のクラスタリング機能と IBM MQ のクラスタリングの組み合わせの概要*

IBM i 上で IBM MQ を実行し、IBM i クラスタリング機能を活用することで、IBM MQ クラスタリングのみを使用するよりも包括的な高可用性ソリューションを実現できます。

この機能を使用するには、以下の項目をセットアップする必要があります。

1. IBM i マシン上のクラスター。417 ページの『IBM i クラスター』を参照してください。
2. 独立補助記憶域プール (IASP)。このプールにキュー・マネージャーを移動します。417 ページの『独立補助記憶域プール (IASP)』を参照してください。
3. クラスター・リソース・グループ (CRG)。このグループで以下の項目を定義します。417 ページの『装置クラスター・リソース・グループ』を参照してください。
   - リカバリー・ドメイン
   - IASP
   - 出口プログラム。417 ページの『装置 CRG 出口プログラム』を参照してください。

IBM i クラスター

IBM i クラスターは、論理的に相互にリンクされたインスタンス (つまり、IBM i のコンピューターやパーティション) の集合です。

このグループ化の目的は、各インスタンスのバックアップを可能にして、Single Point of Failure をなくし、アプリケーションとデータの回復力を高めることです。 クラスターを作成すれば、クラスター内のアプリケーションやデータや装置を管理するために、さまざまなタイプのクラスター・リソース・グループ (CRG) を構成できます。

詳細については、Creating a cluster と Create Cluster (CRTCLU) コマンドを参照してください。

独立補助記憶域プール (IASP)

IASP は、単一レベル・ストレージの拡張としての役割を果たすユーザー ASP の一種です。 これは、ストレージの一部分であり、システム・ストレージから独立しているのでシステムに IPL を実行する必要もなく簡単に操作できます。

IASP は、別のオペレーティング・システム・インスタンスに簡単に切り替えたり、別のオペレーティング・システム・インスタンス上のターゲット IASP に簡単に複製したりできます。 インスタンス間で IASP を切り替えるには、2 つの方法があります。

- 最初の方法では、高速リンク (HSL) ループを使用して、クラスター内のすべてのコンピューターと、IASP が含まれている切り替え可能ディスク・タワーを接続する必要があります。
- 2 番目の方法では、複数のオペレーティング・システム・インスタンスを同じ IBM i コンピューター上の複数のパーティションにして、パーティション間で入出力プロセッサー (IOP) を切り替えるようにしなければなりません。 IASP を複製するための特別なハードウェアは必要ありません。 ネットワークで TCP/IP を使用して複製を実行します。

詳細については、Configure Device ASP (CFGDEVASP) コマンドを参照してください。

装置クラスター・リソース・グループ

クラスター・リソース・グループ (CRG) にはいくつかのタイプがあります。 各種の CRG の詳細については、Cluster resource group を参照してください。

このトピックでは、装置 CRG を特に取り上げます。 装置 CRG は以下のような働きをします。

- 独立補助ストレージ・プール (IASP) などの装置リソースを記述して管理します。
- クラスター・ノードのリカバリー・ドメインを定義します。
- 装置を割り当てます。
- クラスター・イベントを処理する出口プログラムを割り当てます。

リカバリー・ドメインでは、どのクラスター・ノードを 1 次ノードと見なすかを指定します。 残りのノードはバックアップと見なします。 バックアップ・ノードについても、リカバリー・ドメイン内で順序を付け、リカバリー・ドメイン内にあるノードの数に応じて、1 番目のバックアップ、2 番目のバックアップなどと指定します。

1 次ノードで障害が発生すると、リカバリー・ドメイン内のすべてのノードで出口プログラムが実行されます。 最初のバックアップで実行される出口プログラムが、そのノードを新しい 1 次ノードにするために必要な初期化を行います。

詳細については、Creating device CRGs と Create Cluster Resource Group (CRTCRG) コマンドを参照してください。

装置 CRG 出口プログラム

オペレーティング・システムのクラスター・リソース・サービスは、リカバリー・ドメインで定義されているいずれかのノードでフェイルオーバーや切り替えなどのイベントが発生すると、装置 CRG 出口プログラムを呼び出します。

フェイルオーバー・イベントが発生するのは、クラスターの 1 次ノードで障害が発生し、管理対象のすべてのリソースで CRG が切り替えられた時です。切り替えイベントが発生するのは、特定の CRG が手動で 1 次ノードからバックアップ・ノードに切り替えられた時です。

どちらの場合も、出口プログラムが、前の 1 次ノードで実行されていたすべてのプログラムの初期化と開始を担当します。これにより、最初のバックアップ・ノードが新しい 1 次ノードに変換されます。

例えば、IBM MQ では、出口プログラムが IBM MQ サブシステム (QMQM) とキュー・マネージャーの開始を担当する必要があります。 キュー・マネージャーで、リスナー、およびトリガー・モニターなどのサービスを自動的に開始するように構成しておくことも必要です。

サンプル出口プログラム AMQSCRG4 は、 IBM i から入手できます。

切り替え可能 IASP の構成

IBM MQ で、IBM i のクラスタリング機能を利用するためのセットアップを実行できます。 そのためには、次のようにします。

1. データ・センター・システム間で IBM i クラスターを作成します。
2. キュー・マネージャーを IASP に移動します。

   419 ページの『独立補助ストレージ・プールへのキュー・マネージャーの移動、独立補助ストレージ・プールからのキュー・マネージャーの削除』には、この操作の実行に役立つサンプル・コードが含まれています。
3. リカバリー・ドメインや IASP や出口プログラムを定義する CRG を作成する必要があります。

   418 ページの『装置クラスター・リソース・グループの構成』には、この操作の実行に役立つサンプル・コードが含まれています。

関連概念

439 ページの『独立 ASP および高可用性』
独立 ASP では、アプリケーションとデータをサーバー間で移動させることができます。 独立 ASP に柔軟性があるということは、それがいくつかの IBM i 高可用性ソリューションの基本であることを意味します。キュー・マネージャー・ジャーナルで ASP を使用するか独立 ASP を使用するかを検討する際には、独立 ASP に基づくその他の高可用性の構成を検討する必要があります。

IBM i 装置クラスター・リソース・グループの構成
装置クラスター・リソース・グループ (CRG) をセットアップするためのサンプル・プログラム。

このタスクについて

以下の例に関する注記をまとめておきます。

- [PRIMARY SITE NAME] と [BACKUP SITE NAME] は、8 文字以下のストリングです。2 つの区別が可能なストリングであれば何でも構いません。
- [PRIMARY IP] と [BACKUP IP] は、ミラーリングのために使用する IP です。

手順

1. クラスターの名前を指定します。
2. CRG の出口プログラムの名前とライブラリーを指定します。
3. この CRG で定義する 1 次ノードとバックアップ・ノードの名前を指定します。
4. この CRG で管理する IASP を指定し、その IASP が 1 次ノードで作成されていることを確認します。
5. 以下のコマンドを使用して、バックアップ・ノードで装置記述を作成します。

   ```
   CRTDEVASP DEVD([IASP NAME]) RSRCNAME([IASP NAME])
   ```
6. 以下のコマンドを使用して、すべてのノードにテークオーバー IP アドレスを追加します。

```
ADDTCPIFC INTNETADR(' [TAKEOVER IP]') LIND([LINE DESC])
SUBNETMASK('[SUBNET MASK]') AUTOSTART(*NO)
```

7. 以下のコマンドを使用して、1 次ノードだけでテークオーバー IP アドレスを開始します。

```
STRTCPIFC INTNETADR('[TAKEOVER IP')
```

8. オプション: IASP を切り替え可能にする場合は、以下のコマンドを呼び出します。

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG( [CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT
NAME])
USRPRF([EXIT PROFILE]) RCYDMN(( [PRIMARY NODE] *PRIMARY) ([BACKUP NAME] *BACKUP))
EXITPGMFMT(EXTP0200) CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]')
```

9. オプション: IASP をミラーリング構成にする場合は、以下のコマンドを呼び出します。

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG([CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT NAME])
USRPRF([EXIT PROFILE]) RCYDMN(([PRIMARY NODE] *PRIMARY *LAST [PRIMARY SITE NAME] ('[PRIMARY
IP]'))
[BACKUP NAME] *BACKUP *LAST [BACKUP SITE NAME] ('[BACKUP IP]'))) EXITPGMFMT(EXTP0200)
CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]'))
```

IBM i 独立補助ストレージ・プールへのキュー・マネージャーの移動、独立補助ストレージ・プールからのキュー・マネージャーの削除

キュー・マネージャーを独立補助ストレージ・プール (IASP) に移動するコマンドとキュー・マネージャーを IASP から削除するコマンドの例。

このタスクについて

以下の例に関する注記をまとめておきます。

- [MANAGER NAME] は、キュー・マネージャーの名前です。
- [IASP NAME] は、IASP の名前です。
- [MANAGER LIBRARY] は、キュー・マネージャー・ライブラリーの名前です。
- [MANAGER DIRECTORY] は、キュー・マネージャー・ディレクトリーの名前です。

手順

1. 1 次ノードとバックアップ・ノードを指定します。
2. 1 次ノードで以下の手順を実行します。
   a) キュー・マネージャーが終了していることを確認します。
   b) 次のコマンドを使用して、IASP が vary on であることを確認します。

```
VRYCFG CFGOBJ([IASP NAME]) CFGTYPE(*DEV) STATUS(*ON)
```

   c) IASP の下にキュー・マネージャー・ディレクトリーを作成します。
   以下のように、ルートの下に IASP 名のディレクトリーがあります。

```
QSH CMD('mkdir -p /[IASP_NAME]/QIBM/UserData/mqm/qmgrs/')
```

   d) 以下のコマンドを使用して、マネージャーの IFS オブジェクトを、IASP の下に作成したキュー・マネージャー・ディレクトリーに移動します。

```
QSH CMD('mv /QIBM/UserData/mqm/qmgrs/[MANAGER NAME]
/[IASP NAME]/QIBM/UserData/mqm/qmgrs')
```

   e) 以下のコマンドを使用して、MGRLIB という名前の一時保存ファイルを作成します。

```
CRTSAVF QGPL/MGRLIB
```

f) 以下のコマンドを使用して、キュー・マネージャー・ライブラリーを MGRLIB 保存ファイルに保存します。

```
SAVLIB LIB([MANGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
```

g) 以下のコマンドを使用して、キュー・マネージャー・ライブラリーを削除します。照会メッセージはすべて無視してください。

```
DLTLIB [MANAGER LIBRARY]
```

h) 以下のコマンドを使用して、キュー・マネージャー・ライブラリーを IASP にリストアします。

```
RSTLIB SAVLIB([MANAGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
RSTASPDEV([IASP NAME])
```

i) 以下のコマンドを使用して、一時保存ファイルを削除します。

```
 DLTF FILE(QGPL/MGRLIB)
```

j) 以下のコマンドを使用して、IASP の下にキュー・マネージャー IFS オブジェクトのシンボリック・リンクを作成します。

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

k) 以下のコマンドを使用して、IASP に接続します。

```
SETASPGRP [IASP NAME]
```

l) 以下のコマンドを使用して、キュー・マネージャーを開始します。

```
STRMQM [MANAGER NAME]
```

3. バックアップ・ノードで以下の手順を実行します。

a) 以下のコマンドを使用して、一時キュー・マネージャー・ディレクトリーを作成します。

```
QSH CMD('mkdir -p /[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

b) 以下のコマンドを使用して、キュー・マネージャーの一時ディレクトリーへのシンボリック・リンクを作成します。

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

c) 以下のコマンドを使用して、一時ディレクトリーを削除します。

```
QSH CMD('rm -r /[IASP NAME]')
```

d) /QIBM/UserData/mqm/mqs.ini ファイルの末尾に以下のコードを追加します。

```
QueueManager:
Name=[MANAGER NAME]
Prefix=/QIBM/UserData/mqm
Library=[MANAGER LIBRARY]
Directory=[MANAGER DIRECTORY]
```

4. IASP からキュー・マネージャーを削除するには、以下のコマンドを実行します。

a) VRYCFG CFGOBJ([IASP NAME]) CFGTYPE(*DEV) STATUS(*ON)

b) SETASPGRP [IASP NAME]

c) ENDMQM [MANAGER NAME]

d) DLTMQM [MANAGER NAME]

IBM i *IBM i での ASP のミラーリングされたジャーナル構成*

ミラーリングされたジャーナル間の同期複製を使用して、堅固な複数インスタンスのキュー・マネージャーを構成します。

ミラーリングされたキュー・マネージャー構成では、基本ストレージ・プールまたは独立した補助ストレージ・プール (ASP) 内に作成されたジャーナルを使用します。

IBM i では、キュー・マネージャーのデータはジャーナルとファイル・システムに書き込まれます。 ジャーナルには、キュー・マネージャー・データのマスター・コピーが格納されます。 ジャーナルは、同期または非同期のジャーナル複製を使用して、システム間で共有されます。 キュー・マネージャー・インスタンスを再始動するには、ローカル・ジャーナルとリモート・ジャーナルの両方が必要です。 キュー・マネージャーの再始動では、サーバー上のローカル・ジャーナルとリモート・ジャーナルの組み合わせからジャーナル・レコードが読み込まれるとともに、共有ネットワーク・ファイル・システム上のキュー・マネージャー・データが読み込まれます。 ファイル・システム内のデータによって、キュー・マネージャーの再始動が高速化されます。 ファイル・システムには、ファイル・システムとジャーナルの間の同期点にマークを付けるチェックポイントが保管されます。 通常のキュー・マネージャーの再始動には、チェックポイントより前に保管されたジャーナル・レコードは必要ありません。 ただし、ファイル・システム内のデータが最新のものではない可能性があるため、チェックポイント後のジャーナル・レコードを使用して、キュー・マネージャーの再始動を完了します。 インスタンスに接続されたジャーナル内のデータは最新の状態に維持されることから、再始動は正常に完了します。

一方、スタンバイ・サーバー上のリモート・ジャーナルが非同期で複製されていて、ジャーナルが同期化される前に障害が発生した場合には、ジャーナル・レコードでさえも最新のものではない可能性があります。 同期化されていないリモート・ジャーナルを使用してキュー・マネージャーを再始動することにした場合、スタンバイ・キュー・マネージャー・インスタンスが、アクティブ・インスタンスでの障害発生前に削除されたメッセージを再処理しないか、あるいはアクティブ・インスタンスでの障害発生前に受信したメッセージを処理しない可能性があります。

また、まれなことですが、最新のチェックポイント・レコードが、ファイル・システムには格納されていて、スタンバイ側の同期化されていないリモート・ジャーナルには格納されていないことがあります。 この場合、キュー・マネージャーは自動的に再始動しません。 選択肢としては、リモート・ジャーナルが同期化されるまで待機するか、ファイル・システムからスタンバイ・キュー・マネージャーのコールド・スタートを実行するという方法があります。 この場合、ファイル・システムに、リモート・ジャーナルよりも新しいキュー・マネージャー・データのチェックポイントが格納されているとしても、アクティブ・インスタンスでの障害発生前に処理されたすべてのメッセージが格納されているとは限りません。 したがって、ジャーナルと同期化されていないコールド・リスタートを実行すると、一部のメッセージは再処理され、一部のメッセージは処理されない可能性があります。

複数インスタンス・キュー・マネージャーでは、キュー・マネージャーのどのインスタンスをアクティブにし、どのインスタンスをスタンバイにするかを制御するために、ファイル・システムも使用されます。 キュー・マネージャー・データに対するロックは、アクティブ・インスタンスが獲得します。 スタンバイ・インスタンスはロックを獲得するまで待機し、ロックを獲得した時点でアクティブ・インスタンスになります。 アクティブ・インスタンスは、正常に終了するとロックを解放します。 ファイル・システムがアクティブ・インスタンスで障害が発生したか、またはアクティブ・インスタンスがファイル・システムにアクセスできないことを検出した場合は、ファイル・システムによってロックが解放されます。 ファイル・システムは、障害検出に関する要件を満たしていなければなりません。 ファイル共有システムの要件を参照してください。

IBM i での複数インスタンス・キュー・マネージャーのアーキテクチャーでは、サーバーまたはキュー・マネージャーでの障害発生後に自動再始動が行われます。 また、キュー・マネージャー・データが保管されているファイル・システムで障害が発生した後のキュー・マネージャー・データの回復もサポートされます。

422 ページの図 24 では、ALPHA で障害が発生した場合、ミラーリングされたジャーナルを使用して BETA 上の QM1 を手動で再始動できます。 QM1 に複数インスタンス・キュー・マネージャー機能を追加すると、ALPHA 上のアクティブ・インスタンスで障害が発生した場合、QM1 のスタンバイ・インスタンスが BETA 上で自動的に再開します。 QM1 は、QM1 のアクティブ・インスタンスだけでなく、サーバー ALPHA で障害が発生した場合にも、自動的に再開します。 BETA がアクティブ・キュー・マネージャー・インスタンスのホストになった後、ALPHA でスタンバイ・インスタンスを開始できます。

422 ページの図 24 は、キュー・マネージャーの 2 つのインスタンスの間でジャーナルをミラーリングする構成を示しています。この構成では、NetServer を使用してキュー・マネージャー・データを保管します。パターンを拡張して追加のジャーナルを組み込むことにより、インスタンスを増やすことができます。 その場合には、トピック 401 ページの『キュー・マネージャー・ジャーナル (IBM i)』で説明されているジャーナル命名規則に従ってください。 現在のところ、キュー・マネージャーの実行インスタンスの数は 2 つに制限されています。1 つはアクティブ・インスタンス、もう 1 つはスタンバイ・インスタンスです。

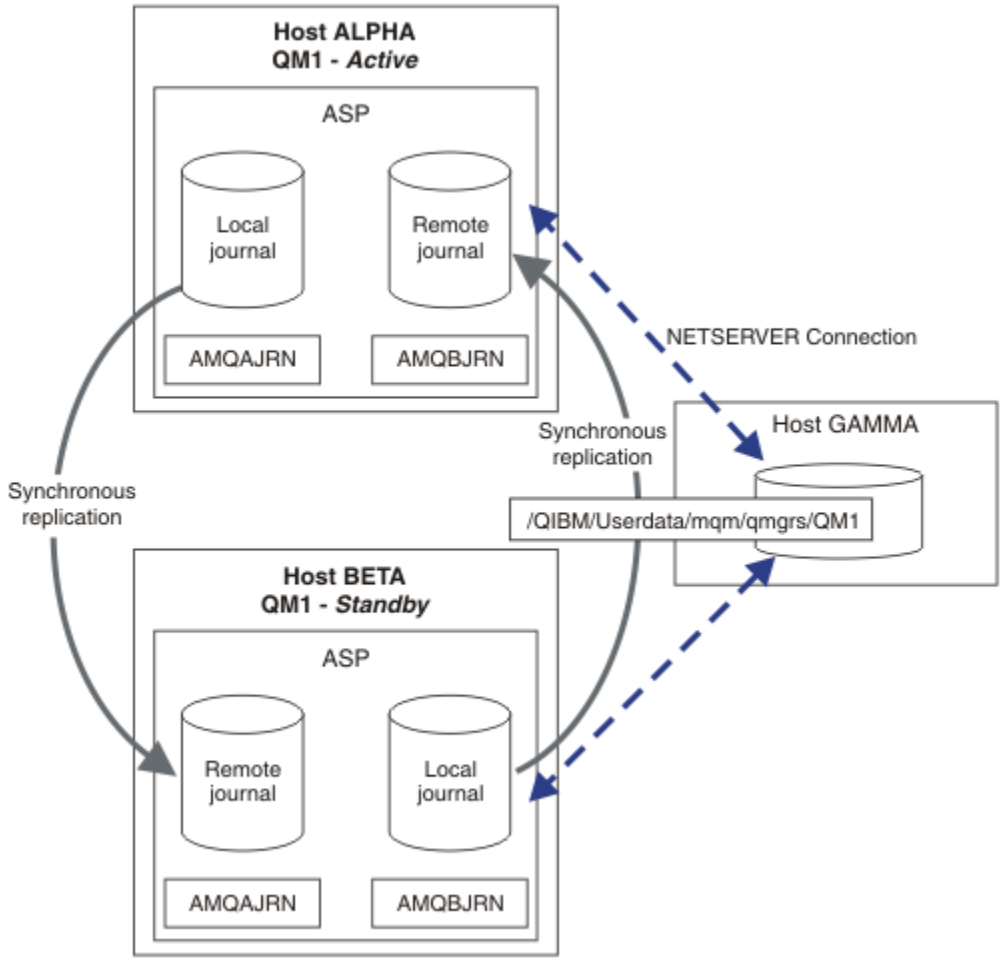


図 *24.* キュー・マネージャー・ジャーナルのミラーリング

ホスト ALPHA 上の QM1 のローカル・ジャーナルの名前は AMQAJRN (厳密には QMQM1/AMQAJRN) であり、BETA 上のジャーナルは QMQM1/AMQBJRN です。 各ローカル・ジャーナルは、キュー・マネージャーの他のすべてのインスタンス上のリモート・ジャーナルに複製します。 キュー・マネージャーが 2 つのインスタンスで構成されている場合、ローカル・ジャーナルは 1 つのリモート・ジャーナルに複製されます。

*SYNC または *ASYNC リモート・ジャーナルの複製

IBM i ジャーナルをミラーリングするには、同期 (*SYNC) ジャーナリングまたは非同期 (*ASYNC) ジャーナリングのいずれかが使用されます。『遠隔ジャーナル管理』を参照してください。

422 ページの図 24 のレプリケーション・モードは *非同期では *同期ません。 *非同期 ではありませんが、リモート・ジャーナルの状態が *非同期のときに障害が発生した場合、ローカル・ジャーナルとリモート・ジャーナルの整合性がありません。 リモート・ジャーナルは、ローカル・ジャーナルからの後れを取り戻さなければなりません。 *SYNC を選択する場合、ローカル・システムは、完了済みの書き込みを必要とする呼び出しから戻る前にリモート・ジャーナルを待ちます。 ローカル・ジャーナルとリモート・ジャーナルは相互に整合した状態を保ちます。 *同期 操作に指定された時間より長い時間がかかる場合のみ [1] リモート・ジャーナリングは非活動化されています。ジャーナルの同期が切れなくなります。 エラーがジ

ャーナル・メッセージ待ち行列および QSYSOPR に記録されます。 キュー・マネージャーはこのメッセージを検出し、エラーをキュー・マネージャー・エラー・ログに書き込み、キュー・マネージャーのジャーナルのリモート複製を非アクティブ化します。 アクティブ・キュー・マネージャー・インスタンスは、このジャーナルに対するリモート・ジャーナリングなしで再開します。 リモート・サーバーが再び使用可能になったら、同期リモート・ジャーナル複製を再びアクティブ化する必要があります。 すると、ジャーナルが再同期化されます。

422 ページの図 24 に示した *SYNC/*SYNC 構成での問題は、BETA 上のスタンバイ・キュー・マネージャー・インスタンスがどのように制御を取得するかです。 BETA 上のキュー・マネージャー・インスタンスは、最初の持続メッセージを書き込むと同時に、ALPHA 上のリモート・ジャーナルを更新しようとします。 ALPHA から BETA に制御が渡された原因が ALPHA の障害であり、ALPHA がまだ停止しているとしたら、ALPHA に対するリモート・ジャーナリングは失敗します。 BETA は ALPHA が応答するまで待機した後、リモート・ジャーナリングを非アクティブ化し、1 つのローカル・ジャーナリングだけでメッセージの処理を再開します。 BETA は ALPHA の停止を検出するまで待機しなければならないため、非活動期間が発生します。

リモート・ジャーナリングを *SYNC に設定するか、または *ASYNC に設定するかは、トレードオフが伴う選択です。 423 ページの表 24 に、キュー・マネージャーのペアの間で *SYNC ジャーナリングを使用した場合と *ASYNC ジャーナリングを使用した場合のトレードオフを要約します。

表 *24.* リモート・ジャーナリングのオプション

アクティブ	スタンバイ	***SYNC**	***ASYNC**
***SYNC**		1. 一貫性のある切り替えおよびフェイルオーバー 2. フェイルオーバー後、スタンバイ・インスタンスは即時に再開しません。 3. 常にリモート・ジャーナリングが使用可能ではなければなりません。 4. キュー・マネージャーのパフォーマンスはリモート・ジャーナリングに依存します。	1. 一貫性のある切り替えおよびフェイルオーバー 2. スタンバイ・サーバーが使用可能になった時点で、リモート・ジャーナリングを *SYNC に切り替える必要があります。 3. 再始動後にも、リモート・ジャーナリングが引き続き使用可能でなければなりません。 4. キュー・マネージャーのパフォーマンスはリモート・ジャーナリングに依存します。
***ASYNC**		1. 賢明な組み合わせではありません。	1. フェイルオーバーまたは切り替え後に、一部のメッセージが失われるか、複製される可能性があります。 2. スタンバイ・インスタンスが常に使用可能でなくても、遅延なしでアクティブ・インスタンスを継続させることができます。 3. パフォーマンスはリモート・ジャーナリングに依存しません。

***SYNC / *SYNC**
アクティブ・キュー・マネージャー・インスタンスは *SYNC ジャーナリングを使用し、スタンバイ・キュー・マネージャー・インスタンスが開始すると同時に *SYNC ジャーナリングを使用しようと試みます。

[1] 指定された時間は、 IBM i 5 では 60 秒、 IBM i 6.1 上では 1 から 3600 秒の範囲で指定されます。

1. リモート・ジャーナルのトランザクションは、アクティブ・キュー・マネージャーのローカル・ジャーナルと整合します。 キュー・マネージャーがスタンバイ・インスタンスに切り替えられた場合、キュー・マネージャーは即時に再開できます。 通常、スタンバイ・インスタンスはメッセージの損失や重複なしで再開します。 メッセージの損失または重複が発生するのは、最終チェックポイント以降にリモート・ジャーナリングが失敗し、以前にアクティブであったキュー・マネージャーを再始動できない場合のみです。
2. キュー・マネージャーがスタンバイ・インスタンスに切り替えられた場合、即時に開始できない場合があります。 スタンバイ・キュー・マネージャー・インスタンスは、*SYNC ジャーナリングを使用してアクティブ化されます。 フェイルオーバーの原因が、スタンバイ・インスタンスをホストするサーバーに対するリモート・ジャーナリングの妨げとなる場合もあります。 キュー・マネージャーは、永続メッセージを処理する前に問題が検出されるまで待機します。 エラーがジャーナル・メッセージ待ち行列および QSYSOPR に記録されます。 キュー・マネージャーはこのメッセージを検出し、エラーをキュー・マネージャー・エラー・ログに書き込み、キュー・マネージャーのジャーナルのリモート複製を非アクティブ化します。 アクティブ・キュー・マネージャー・インスタンスは、このジャーナルに対するリモート・ジャーナリングなしで再開します。 リモート・サーバーが再び使用可能になったら、同期リモート・ジャーナル複製を再びアクティブ化する必要があります。 すると、ジャーナルが再同期化されます。
3. リモート・ジャーナルを維持するために、リモート・ジャーナリングの複製先サーバーが常に使用可能でなければなりません。 リモート・ジャーナルは通常、スタンバイ・キュー・マネージャーをホストするサーバーに複製されます。 サーバーが使用不可になる可能性が エラーがジャーナル・メッセージ待ち行列および QSYSOPR に記録されます。 キュー・マネージャーはこのメッセージを検出し、エラーをキュー・マネージャー・エラー・ログに書き込み、キュー・マネージャーのジャーナルのリモート複製を非アクティブ化します。 アクティブ・キュー・マネージャー・インスタンスは、このジャーナルに対するリモート・ジャーナリングなしで再開します。 リモート・サーバーが再び使用可能になったら、同期リモート・ジャーナル複製を再びアクティブ化する必要があります。 すると、ジャーナルが再同期化されます。
4. サーバー間の距離がかなり離れている場合、リモート・ジャーナリングには、ローカル・ジャーナリングよりも時間がかかります。 キュー・マネージャーはリモート・ジャーナリングを待機しなければならないため、キュー・マネージャーのパフォーマンスが低下します。

サーバーのペアの間での *SYNC/*SYNC 構成には、フェイルオーバー後にスタンバイ・インスタンスを再開する際に遅延が発生するという欠点があります。 *SYNC/*ASYNC 構成には、この問題はありません。

*SYNC/*SYNC 構成では、リモート・ジャーナルが使用可能である限り、切り替えまたはフェイルオーバー後にメッセージ損失が発生することはありません。 フェイルオーバーまたは切り替え後のメッセージ損失のリスクを低くする必要がある場合には、2 つの選択肢があります。 1 つは、リモート・ジャーナルが非活動状態になった場合にアクティブ・インスタンスを停止すること、もう 1 つは、複数のサーバーにリモート・ジャーナルを作成することです。

***SYNC / *ASYNC**

アクティブ・キュー・マネージャー・インスタンスは *SYNC ジャーナリングを使用し、スタンバイ・キュー・マネージャー・インスタンスが開始すると、*ASYNC ジャーナリングを使用します。 システム・オペレーターは、新しいスタンバイ・インスタンスをホストするサーバーが使用可能になった直後に、アクティブ・インスタンス上のリモート・ジャーナルを *SYNC に切り替える必要があります。 オペレーターがリモート・ジャーナルを *ASYNC から *SYNC に切り替えると、アクティブ・インスタンスは、リモート・ジャーナルの状態が *ASYNCPEND であれば、一時停止します。 アクティブ・キュー・マネージャー・インスタンスは、残りのジャーナル・エントリーがリモート・ジャーナルに転送されるまで待機します。 リモート・ジャーナルがローカル・ジャーナルと同期された時点で、新しいスタンバイ・インスタンスのトランザクションは、新しいアクティブ・インスタンスとの整合性を取り戻します。 複数インスタンス・キュー・マネージャーの管理という観点からすると、*SYNC/*ASYNC 構成では、IBM i システム・オペレーターのタスクが追加されます。 オペレーターは、障害が発生したキュー・マネージャー・インスタンスを再始動するだけでなく、リモート・ジャーナリングを *SYNC に切り替える必要もあります。

1. リモート・ジャーナルのトランザクションは、アクティブ・キュー・マネージャーのローカル・ジャーナルと整合します。 アクティブ・キュー・マネージャーのインスタンスが切り替えられた場合、またはスタンバイ・インスタンスにフェイルオーバーした場合は、スタンバイ・インスタンスが即

時に再開されます。 通常、スタンバイ・インスタンスはメッセージの損失や重複なしで再開します。メッセージの損失または重複が発生するのは、最終チェックポイント以降にリモート・ジャーナリングが失敗し、以前にアクティブであったキュー・マネージャーを再始動できない場合のみです。

2. アクティブ・インスタンスをホストするシステムが再び使用可能になった後、システム・オペレーターはすぐにリモート・ジャーナルを *ASYNC から *SYNC に切り替える必要があります。 オペレーターは、リモート・ジャーナルが後れを取り戻すまで待機してから、リモート・ジャーナルを *SYNC に切り替えなければならない場合もあります。 あるいは、オペレーターは即時にリモート・インスタンスを *SYNC に切り替え、スタンバイ・インスタンスのジャーナルが後れを取り戻すまで、アクティブ・インスタンスを強制的に待機させることもできます。 リモート・ジャーナリングが *同期に設定されている場合、スタンバイ・インスタンスは通常、アクティブ・インスタンスとトランザクションの整合性が保たれます。 メッセージの損失または重複が発生するのは、最終チェックポイント以降にリモート・ジャーナリングが失敗し、以前にアクティブであったキュー・マネージャーを再始動できない場合のみです。
3. 切り替えまたはフェイルオーバーによって構成が復元された場合、リモート・ジャーナルがホストされているサーバーは常時、使用可能である必要があります。

フェイルオーバー後にすぐにスタンバイ・キュー・マネージャーが再開できるようにするには、*SYNC/*ASYNC を選択してください。 この場合、新しいアクティブ・インスタンスでのリモート・ジャーナルの設定を手動で *SYNC に復元する必要があります。 *SYNC/*ASYNC 構成は、複数インスタンス・キュー・マネージャーの一般的な管理パターンと一致します。 1 つのインスタンスで障害が発生した後、スタンバイ・インスタンスが再始動されるまでには時間があり、その間はアクティブ・インスタンスのフェイルオーバーが不可能になります。

***ASYNC / *ASYNC**

アクティブ・キュー・マネージャーをホストするサーバーとスタンバイ・キュー・マネージャーをホストするサーバーの両方が、*ASYNC リモート・ジャーナリングを使用するように構成されます。

1. 切り替えまたはフェイルオーバーが発生すると、キュー・マネージャーは新しいサーバー上のジャーナルを使用して処理を続けます。 切り替えまたはフェイルオーバーの発生時に、ジャーナルが同期化されていない場合も考えられます。 その場合には、メッセージの損失または重複が発生する可能性があります。
2. アクティブ・インスタンスは、スタンバイ・キュー・マネージャーをホストするサーバーが使用可能でないとしても稼働します。 スタンバイ・サーバーが使用可能になると、ローカル・ジャーナルがスタンバイ・サーバーで非同期に複製されます。
3. ローカル・キュー・マネージャーのパフォーマンスは、リモート・ジャーナリングには影響されません。

パフォーマンスが主要な要件である場合には、*ASYNC/*ASYNC を選択し、フェイルオーバーまたは切り替え後のメッセージの損失または重複に備えてください。

***ASYNC / *SYNC**

このオプションの組み合わせを使用する理由は何もありません。

リモート・ジャーナルからのキュー・マネージャーのアクティブ化

ジャーナルは、同期的または非同期的に複製されます。 リモート・ジャーナルがアクティブでないか、ローカル・ジャーナルからの後れを取り戻そうとしている可能性があります。 同期的に複製される場合でも、リモート・ジャーナルが後れを取り戻そうとする可能性はあります。アクティブ化されてからまだ間もない可能性があるためです。 キュー・マネージャーは始動時に、リモート・ジャーナルの状態に次の規則を適用します。

1. スタンドバイをそのリモート・ジャーナルから再生する必要があり、ジャーナルの状況が *FAILED または *INACTPEND である場合、スタンバイの開始は失敗します。
2. スタンバイのアクティブ化が開始する際、スタンバイのリモート・ジャーナルの状況は、*ACTIVE または *INACTIVE でなければなりません。 状態が *INACTIVE になっていると、すべてのジャーナル・データが複製されなかった場合にアクティブ化が失敗する可能性があります。

   ネットワーク・ファイル・システム上のキュー・マネージャーのデータに、リモート・ジャーナルに存在するチェックポイント・レコードよりも新しいチェックポイント・レコードがある場合は、失敗しま

す。 チェックポイントのデフォルト最大間隔である 30 分以内にリモート・ジャーナルがアクティブ化される限り、この失敗が起こることはないはずです。 スタンバイ・キュー・マネージャーがそれよりも新しいチェックポイント・レコードをファイル・システムから読み取った場合、スタンバイ・キュー・マネージャーは始動しません。

選択肢は、アクティブ・サーバー上のローカル・ジャーナルを復元可能になるまで待機するか、スタンバイ・キュー・マネージャーのコールド・スタートを行うかのいずれかです。 コールド・スタートを選択する場合、キュー・マネージャーはジャーナル・データなしで始動し、ファイル・システム内のキュー・マネージャー・データの整合性および完全性に依存することになります。

注: キュー・マネージャーのコールド・スタートを行う場合には、最終チェックポイント後のメッセージが失われるか、複製されるリスクがあります。 メッセージ・トランザクションはジャーナルに書き込まれますが、トランザクションの一部がファイル・システム内のキュー・マネージャー・データに書き込まれていない可能性があります。 キュー・マネージャーのコールド・スタートを行うと、新規ジャーナルが開始され、ファイル・システム内のキュー・マネージャー・データに書き込まれていないトランザクションは失われます。

3. スタンバイ・キュー・マネージャーのアクティブ化は、スタンバイのリモート・ジャーナルの状況が *ASYNCPEND または *SYNCPEND から *ASYNC または *SYNC に変わるのを待ちます。 メッセージは定期的に、実行コントローラーのジョブ・ログに書き込まれます。

   注: この場合、活動化は、活動化されているスタンバイ・キュー・マネージャーに対してローカルのリモート・ジャーナルで待機しています。 リモート・ジャーナルなしで続行する前にも、キュー・マネージャーが待機する時間があります。 リモート・ジャーナル (複数の場合もあり) に同期的に書き込みを試みるときにジャーナルが使用できないと、待機するためです。
4. ジャーナルの状況が *FAILED または *INACTPEND に変更されると、アクティブ化は停止します。

アクティブ化で使用されるローカルおよびリモートのジャーナルの名前と状態は、キュー・マネージャーのエラー・ログに書き込まれます。

IBM i *IBM i* でのジャーナル・ミラーリングおよび *NetServer* 使用による複数インスタンス・キュー・マネージャーの作成

2 つの IBM i サーバー上で実行される複数インスタンス・キュー・マネージャーを作成します。 キュー・マネージャー・データは、NetServer を使用して第 3 の IBM i サーバー上に保管されます。 キュー・マネージャー・ジャーナルは、リモート・ジャーナリングを使用することにより、2 つのサーバーの間でミラーリングされます。 **ADDMQMJRN** コマンドを使用すると、リモート・ジャーナルの作成作業が簡素化されます。

始める前に

1. このタスクには、3 つの IBM i サーバーが必要です。 そのうちの 2 つ (この例では ALPHA と BETA) に IBM MQ をインストールします。 この製品は、IBM WebSphere MQ 7.0.1 Fix Pack 1 以上でなければなりません。
2. 第 3 のサーバーは、NetServer により ALPHA および BETA と接続された IBM i サーバーです。 これは、キュー・マネージャー・データの共有のために使用されます。 これに IBM MQ をインストールする必要はありません。 ただし、一時的なステップとしてこのサーバーに IBM MQ をインストールするなら、キュー・マネージャーのディレクトリーおよび許可をセットアップするのに役立ちます。
3. QMQM ユーザー・プロファイルのパスワードが、3 つのサーバーのすべてにおいて同じであることを確認してください。
4. IBM i NetServer をインストールします。 i5/OS NetServer を参照してください。

このタスクについて

以下の手順を実行することにより、429 ページの図 25 で示されている構成を作成します。 キュー・マネージャーのデータは、IBM i NetServer を使用して接続されます。

- ALPHA および BETA から、GAMMA 上でキュー・マネージャー・データの保管先となるディレクトリー共有への接続を作成します。 このタスクでは、必要な許可、ユーザー・プロファイル、およびパスワードもセットアップされます。

- キュー・マネージャー・インスタンスを実行予定の IBM i システムにリレーショナル・データベース・エントリー (RDBE) を追加します。 RDBE エントリーは、リモート・ジャーナリングのために使用される IBM i システムとの接続用に使用されます。
- IBM i サーバー ALPHA 上にキュー・マネージャー QM1 を作成します。
- もう一方の IBM i サーバー BETA 上の QM1 のキュー・マネージャー制御情報を追加します。
- 2 つの IBM i サーバー上で、2 つのキュー・マネージャー・インスタンスのためのリモート・ジャーナルを作成します。 各キュー・マネージャーは、それぞれローカル・ジャーナルに書き込みます。 そのローカル・ジャーナルがリモート・ジャーナルに複製されます。 **ADDMQMJRN** コマンドを使用すれば、ジャーナルの追加や接続の作業が簡素化されます。
- キュー・マネージャーを始動して、スタンバイ・インスタンスが可能になるようにします。

手順

1. 413 ページの『IBM i での NetServer 使用によるキュー・マネージャー・データ用ネットワーク共有の作成』のタスクを実行します。

   結果として、 ALPHA および BETA には、 GAMMA 上の /QIBM/UserData/mqm/qmgrs を指す共有が /QNTC/GAMMA/WMQ ます。 ユーザー・プロファイル QMQM および QMQMADM には必要な許可が付与されており、3 つのシステムすべてにおいて QMQM のパスワードは一致しています。
2. キュー・マネージャー・インスタンスをホストする予定の IBM i システムにリレーショナル・データベース・エントリー (RDBE) を追加します。

   a) ALPHA において BETA への接続を作成します。

   ```
   ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
   ```

   b) BETA において ALPHA への接続を作成します。

   ```
   ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
   ```
3. ALPHA 上にキュー・マネージャー QM1 を作成し、キュー・マネージャー・データが GAMMA 上に保存されるようにします。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   パスは、 NetServer を使用してキュー・マネージャー・データを作成します。
4. ALPHA 上で実行します。 このコマンドは、BETA 上にリモート・ジャーナルを追加します。

   ```
   ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
   ```

   アクティブ・インスタンスが ALPHA 上にある場合に、ALPHA 上のローカル・ジャーナルにジャーナル項目を作成します。 ALPHA 上のローカル・ジャーナルは、BETA 上のリモート・ジャーナルに複製されます。
5. ALPHA 上で作成された IBM MQ 構成データを検査するには、コマンドを使用します。

   この情報は次のステップで必要になります。

   この例では、以下の構成が ALPHA 上に作成されます。

   ```
   Name=QM1
   Prefix=/QIBM/UserData/mqm
   Library=QMQM1
   Directory=QM1
   DataPath= /QNTC/GAMMA/WMQ /QM1
   ```

6. 以下のコマンドを使用して、BETA 上に QM1 のキュー・マネージャー・インスタンスを作成します。 BETA 上で以下のコマンドを実行して、BETA 上のキュー・マネージャー制御情報を変更します。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

ヒント : 構成情報をコピーして貼り付けます。 キュー・マネージャー・スタンザは、ALPHA と BETA で同じです。

7. BETA 上で実行します。 このコマンドは, BETA 上にローカル・ジャーナルを追加し, ALPHA 上にリモート・ジャーナルを追加します。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
```

のアクティブ・インスタンスが BETA 上にあるときに、BETA 上のローカル・ジャーナルにジャーナル項目を作成します。 BETA 上のローカル・ジャーナルは、ALPHA 上のリモート・ジャーナルに複製されます。

注 : 別の方法として、非同期ジャーナリングを使用することにより、BETA から ALPHA へのリモート・ジャーナリングをセットアップすることもできます。

BETA から ALPHA への非同期ジャーナリングをセットアップするには、428 ページの『7』 のステップに示されているコマンドの代わりに、このコマンドを使用します。

```
ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
```

ALPHA のサーバーまたはジャーナリングが障害の発生源である場合、新規ジャーナル項目が ALPHA に複製されるのを待たずに BETA が開始します。

ALPHA が再びオンラインになったときに、コマンドを使用して複製モードを *SYNC に切り替えます。

ジャーナルのミラーリングを同期にするか、非同期にするか、その混合にするかを決定するには、421 ページの『IBM i での ASP のミラーリングされたジャーナル構成』の情報を使用してください。 デフォルトは、リモート・ジャーナルからの応答待ち時間 60 秒の同期複製です。

8. ALPHA および BETA 上のジャーナルが使用可能になっていること、およびリモート・ジャーナル複製の状況が使用可能になっていることを確認してください。

   a) ALPHA 上で、

   ```
   WRKMQMJRN MQMNAME(QM1)
   ```

   b) BETA 上で、

   ```
   WRKMQMJRN MQMNAME(QM1)
   ```

9. ALPHA および BETA 上でキュー・マネージャー・インスタンスを始動します。

   a) ALPHA 上で最初のインスタンスを始動し、それをアクティブ・インスタンスにします。 スタンバイ・インスタンスへの切り替えを使用可能にします。

   ```
   STRMQM MQMNAME(QM1) STANDBY(*YES)
   ```

   b) BETA 上で 2 番目のインスタンスを始動し、それをスタンバイ・インスタンスにします。

   ```
   STRMQM MQMNAME(QM1) STANDBY(*YES)
   ```

タスクの結果

キュー・マネージャーの状況を確認するために使用します。

1. ALPHA 上のキュー・マネージャー・インスタンスの状況は、次のようになります。
2. BETA 上のキュー・マネージャー・インスタンスの状況は、以下のようになっている必要があります。

例

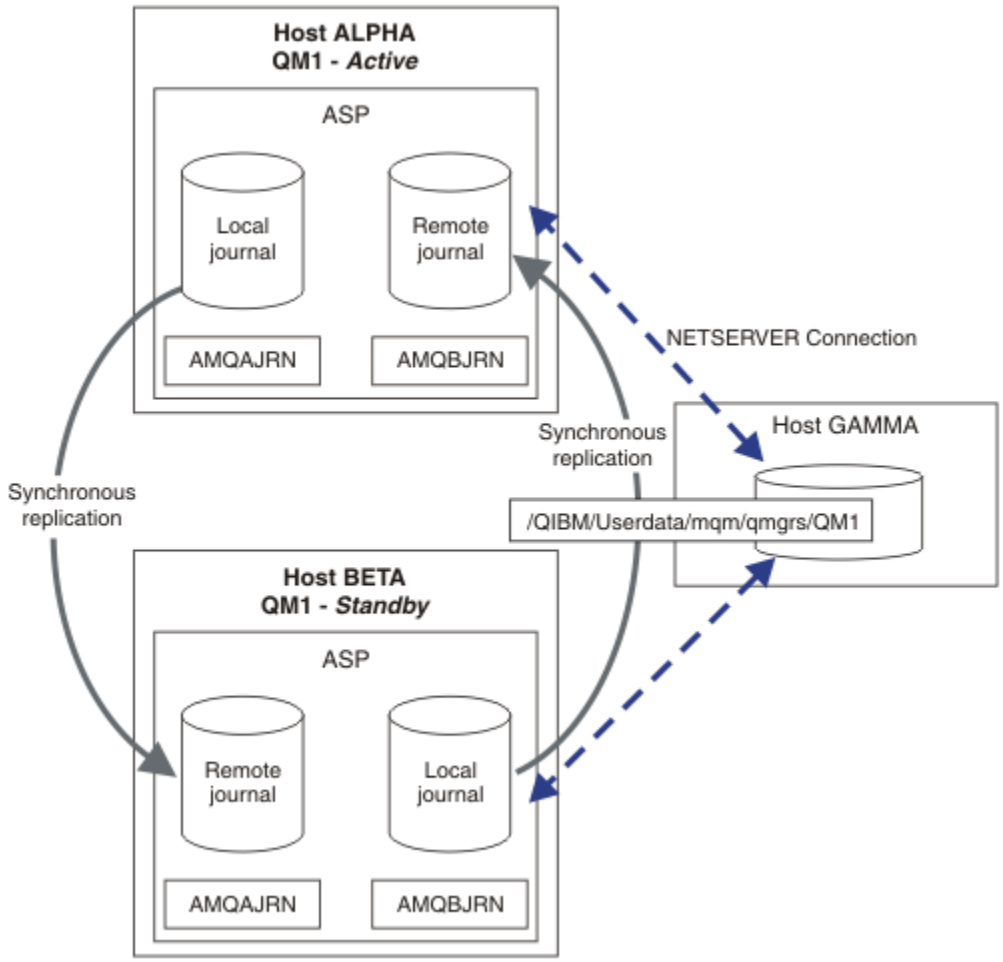


図 *25.* ミラーリングされたジャーナル構成

次のタスク

- アクティブ・インスタンスとスタンバイ・インスタンスが自動的に切り替えられることを確認します。高可用性サンプル・プログラムを実行することにより、切り替えのテストを実施できます。『高可用性のサンプル・プログラム』を参照してください。 サンプル・プログラムは 'C' クライアントです。 それらは、Windows または UNIX プラットフォームから実行できます。
  1. 高可用性サンプル・プログラムを開始します。
  2. ALPHA 上で、キュー・マネージャーを終了して切り替えを要求します。

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. ベータ版のインスタンスがアクティブであることを確認してください。

4. ALPHA 上で再始動します。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

- 代替高可用性構成を確認します。
  1. NetServer を使用することにより、Windows サーバー上にキュー・マネージャー・データを配置します。
  2. リモート・ジャーナリングを使用してキュー・マネージャー・ジャーナルをミラーリングする代わりに、独立 ASP 上にジャーナルを保管します。 IBM i クラスタリングを使用して、独立 ASP を ALPHA から BETA に転送します。

IBM i *IBM i での NetServer およびジャーナル・ミラーリングを使用した単一インスタンス・キュー・マネージャーから複数インスタンス・キュー・マネージャーへの変換*

単一インスタンス・キュー・マネージャーを複数インスタンス・キュー・マネージャーに変換します。 キュー・マネージャー・データを、NetServer によって接続されているネットワーク共有に移動します。 リモート・ジャーナリングを使用することにより、キュー・マネージャー・ジャーナルを 2 番目の IBM i サーバーにミラーリングします。

始める前に

1. このタスクには、3 つの IBM i サーバーが必要です。 この例のサーバー ALPHA 上にある既存の IBM MQ インストール済み環境は、少なくとも IBM WebSphere MQ 7.0.1 Fix Pack 1 でなければなりません。 この例の場合、ALPHA では、QM1 というキュー・マネージャーが稼働中です。
2. 2 番目の IBM i サーバー (例では BETA) に IBM MQ をインストールします。
3. 第 3 のサーバーは、NetServer により ALPHA および BETA と接続された IBM i サーバーです。 これは、キュー・マネージャー・データの共有のために使用されます。 これに IBM MQ をインストールする必要はありません。 ただし、一時的なステップとしてこのサーバーに IBM MQ をインストールするなら、キュー・マネージャーのディレクトリーおよび許可をセットアップするのに役立ちます。
4. QMQM ユーザー・プロファイルのパスワードが、3 つのサーバーのすべてにおいて同じであることを確認してください。
5. IBM i NetServer をインストールします。 i5/OS NetServer を参照してください。

このタスクについて

以下のステップを実行することにより、単一インスタンス・キュー・マネージャーを複数インスタンス・キュー・マネージャーに変換します (434 ページの図 26 を参照)。 このタスクでは単一インスタンス・キュー・マネージャーが削除された後、再作成され、キュー・マネージャー・データが、NetServer により接続されているネットワーク共有上に保管されます。 **CPY** コマンドを使用してキュー・マネージャーのディレクトリーとファイルをネットワーク共有に移動する方法と比較した場合、この手順は信頼性の点で勝っています。

- ALPHA および BETA から、GAMMA 上でキュー・マネージャー・データの保管先となるディレクトリー共有への接続を作成します。 このタスクでは、必要な許可、ユーザー・プロファイル、およびパスワードもセットアップされます。
- キュー・マネージャー・インスタンスを実行予定の IBM i システムにリレーショナル・データベース・エントリー (RDBE) を追加します。 RDBE エントリーは、リモート・ジャーナリングのために使用される IBM i システムとの接続用に使用されます。
- キュー・マネージャーのログと定義を保存し、キュー・マネージャーを停止した後、それを削除します。
- キュー・マネージャーを再作成し、キュー・マネージャー・データを GAMMA 上のネットワーク共有に保管します。
- もう一方のサーバーにキュー・マネージャーの 2 番目のインスタンスを追加します。
- 2 つの IBM i サーバー上で、2 つのキュー・マネージャー・インスタンスのためのリモート・ジャーナルを作成します。 各キュー・マネージャーは、それぞれローカル・ジャーナルに書き込みます。 そのロー

カル・ジャーナルがリモート・ジャーナルに複製されます。 **ADDMQMJRN** コマンドを使用すれば、ジャーナルの追加や接続の作業が簡素化されます。

- キュー・マネージャーを始動して、スタンバイ・インスタンスが可能になるようにします。

注 :

このタスクのステップ 431 ページの『4』 で、単一インスタンス・キュー・マネージャー QM1 を削除します。 キュー・マネージャーを削除すると、キュー上の持続メッセージがすべて削除されます。 そのため、キュー・マネージャーを変換する前に、そのキュー・マネージャーによって保管されたすべてのメッセージの処理を完了してください。 すべてのメッセージを処理することが不可能な場合は、ステップ 431 ページの『4』 の前にキュー・マネージャー・ライブラリーのバックアップを取ってください。 ステップ 431 ページの『5』 の後で、キュー・マネージャー・ライブラリーを復元します。

注 :

このタスクのステップ 431 ページの『5』 で、QM1 を再作成します。 キュー・マネージャーの名前は同じですが、キュー・マネージャー ID は異なります。 キュー・マネージャー・クラスタリングでは、キュー・マネージャー ID が使用されます。 クラスター内のキュー・マネージャーを削除して再作成するには、まず、クラスターからキュー・マネージャーを除去する必要があります。「クラスターからのキュー・マネージャーの削除 : 代替方法」または「クラスターからのキュー・マネージャーの除去」を参照してください。 キュー・マネージャーを再作成したら、それをクラスターに追加します。 その名前は以前と同じですが、クラスター内の他のキュー・マネージャーからは、新しいキュー・マネージャーとして認識されます。

手順

1. 413 ページの『IBM i での NetServer 使用によるキュー・マネージャー・データ用ネットワーク共有の作成』 のタスクを実行します。

   結果として、ALPHA および BETA には、GAMMA 上の /QIBM/UserData/mqm/qmgrs を指す共有が /QNTC/GAMMA/WMQ ます。 ユーザー・プロファイル QMQM および QMQMADM には必要な許可が付与されており、3 つのシステムすべてにおいて QMQM のパスワードは一致しています。

2. キュー・マネージャー・インスタンスをホストする予定の IBM i システムにリレーショナル・データベース・エントリー (RDBE) を追加します。

   a) ALPHA において BETA への接続を作成します。

   ```
   ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
   ```

   b) BETA において ALPHA への接続を作成します。

   ```
   ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
   ```

3. キュー・マネージャー・オブジェクトを再作成するスクリプトを作成します。

   ```
   QSAVEQMGR LCLQMGRNAM(QM1) FILENAME('*CURLIB/QMQSC(QM1)')
   OUTPUT(*REPLACE) MAKEAUTH(*YES) AUTHFN('*CURLIB/QMAUT(QM1)')
   ```

4. キュー・マネージャーを停止し、それを削除します。

   ```
   ENDMQM MQMNAME(QM1) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*YES) TIMEOUT(15)
   DLTMQM MQMNAME(QM1)
   ```

5. ALPHA 上にキュー・マネージャー QM1 を作成し、キュー・マネージャー・データが GAMMA 上に保存されるようにします。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

パスは、 NetServer を使用してキュー・マネージャー・データを作成します。

6. 保存しておいた定義から、QM1 のキュー・マネージャー・オブジェクトを再作成します。

```
STRMQMMQSC SRCMBR(QM1) SRCFILE(*CURLIB/QMQSC) MQMNAME(QM1)
```

7. 保存しておいた情報に基づいて、許可を適用します。
   a) 保存されている許可プログラムをコンパイルします。

```
CRTCLPGM PGM(*CURLIB/QM1) SRCFILE(*CURLIB/QMAUT)
SRCMBR(QM1) REPLACE(*YES)
```

   b) プログラムを実行して、許可を適用します。

```
CALL PGM(*CURLIB/QM1)
```

   c) QM1 のセキュリティー情報をリフレッシュします。

```
RFRMQMAUT MQMNAME(QM1)
```

8. ALPHA 上で実行します。 このコマンドは、BETA 上にリモート・ジャーナルを追加します。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
```

   アクティブ・インスタンスが ALPHA 上にある場合に、ALPHA 上のローカル・ジャーナルにジャーナル項目を作成します。 ALPHA 上のローカル・ジャーナルは、BETA 上のリモート・ジャーナルに複製されます。

9. ALPHA 上で作成された IBM MQ 構成データを検査するには、コマンドを使用します。

   この情報は次のステップで必要になります。

   この例では、以下の構成が ALPHA 上に作成されます。

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QMQM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

10. 以下のコマンドを使用して、BETA 上に QM1 のキュー・マネージャー・インスタンスを作成します。 BETA 上で以下のコマンドを実行して、BETA 上のキュー・マネージャー制御情報を変更します。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

    ヒント: 構成情報をコピーして貼り付けます。 キュー・マネージャー・スタンザは、ALPHA と BETA で同じです。

11. BETA 上で実行します。 このコマンドは, BETA 上にローカル・ジャーナルを追加し, ALPHA 上にリモート・ジャーナルを追加します。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
```

    のアクティブ・インスタンスが BETA 上にあるときに、BETA 上のローカル・ジャーナルにジャーナル項目を作成します。 BETA 上のローカル・ジャーナルは、ALPHA 上のリモート・ジャーナルに複製されます。

注 : 別の方法として、非同期ジャーナリングを使用することにより、BETA から ALPHA へのリモート・ジャーナリングをセットアップすることもできます。

BETA から ALPHA への非同期ジャーナリングをセットアップするには、428 ページの『7』 のステップに示されているコマンドの代わりに、このコマンドを使用します。

```
ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
```

ALPHA のサーバーまたはジャーナリングが障害の発生源である場合、新規ジャーナル項目が ALPHA に複製されるのを待たずに BETA が開始します。

ALPHA が再びオンラインになったときに、コマンドを使用して複製モードを *SYNC に切り替えます。

ジャーナルのミラーリングを同期にするか、非同期にするか、その混合にするかを決定するには、421 ページの『IBM i での ASP のミラーリングされたジャーナル構成』の情報を使用してください。 デフォルトは、リモート・ジャーナルからの応答待ち時間 60 秒の同期複製です。

12. ALPHA および BETA 上のジャーナルが使用可能になっていること、およびリモート・ジャーナル複製の状況が使用可能になっていることを確認してください。
    a) ALPHA 上で、

    ```
    WRKMQMJRN MQMNAME(QM1)
    ```

    b) BETA 上で、

    ```
    WRKMQMJRN MQMNAME(QM1)
    ```

13. ALPHA および BETA 上でキュー・マネージャー・インスタンスを始動します。
    a) ALPHA 上で最初のインスタンスを始動し、それをアクティブ・インスタンスにします。 スタンバイ・インスタンスへの切り替えを使用可能にします。

    ```
    STRMQM MQMNAME(QM1) STANDBY(*YES)
    ```

    b) BETA 上で 2 番目のインスタンスを始動し、それをスタンバイ・インスタンスにします。

    ```
    STRMQM MQMNAME(QM1) STANDBY(*YES)
    ```

タスクの結果

キュー・マネージャーの状況を確認するために使用します。

1. ALPHA 上のキュー・マネージャー・インスタンスの状況は、次のようになります。
2. BETA 上のキュー・マネージャー・インスタンスの状況は、以下のようになっている必要があります。

例

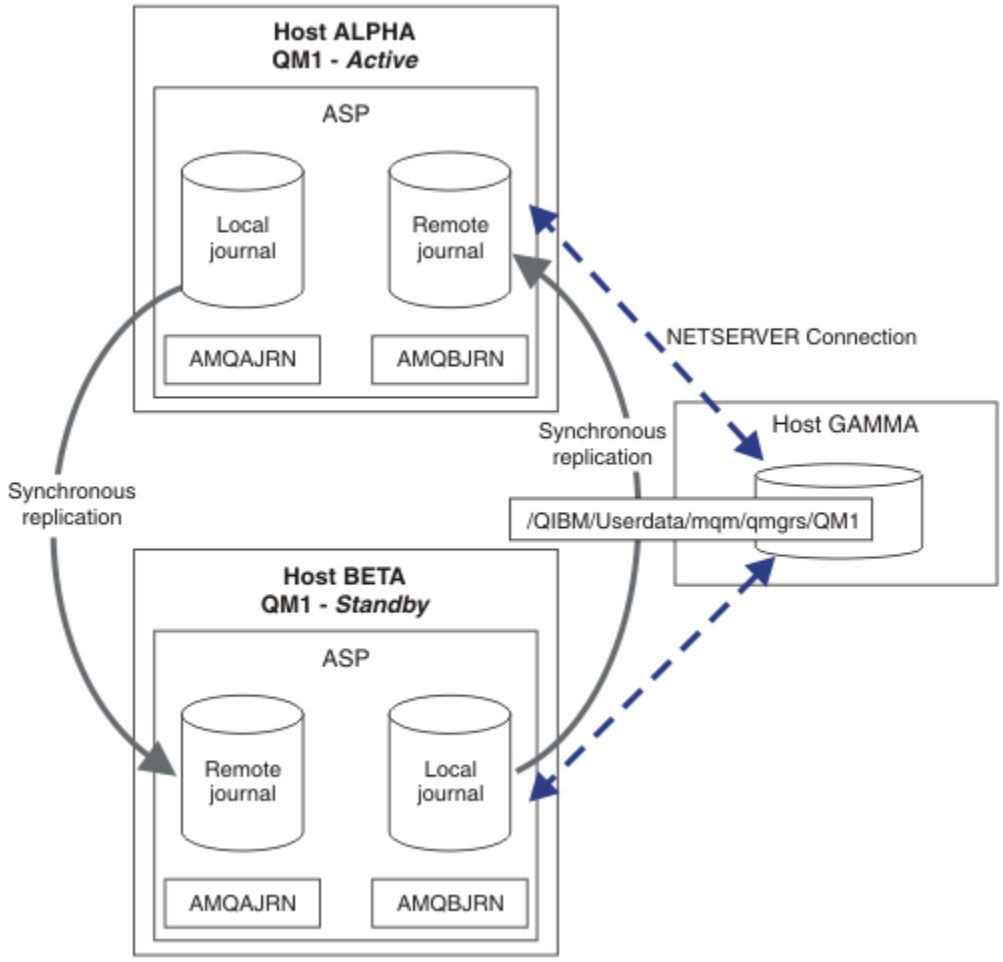


図 *26.* ミラーリングされたジャーナル構成

次のタスク

- アクティブ・インスタンスとスタンバイ・インスタンスが自動的に切り替えられることを確認します。高可用性サンプル・プログラムを実行することにより、切り替えのテストを実施できます。『高可用性のサンプル・プログラム』を参照してください。 サンプル・プログラムは 'C' クライアントです。 それらは、Windows または UNIX プラットフォームから実行できます。
  1. 高可用性サンプル・プログラムを開始します。
  2. ALPHA 上で、キュー・マネージャーを終了して切り替えを要求します。

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. ベータ版のインスタンスがアクティブであることを確認してください。
  4. ALPHA 上で再始動します。

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 代替高可用性構成を確認します。
  1. NetServer を使用することにより、Windows サーバー上にキュー・マネージャー・データを配置します。

2. リモート・ジャーナリングを使用してキュー・マネージャー・ジャーナルをミラーリングする代わりに、独立 ASP 上にジャーナルを保管します。 IBM i クラスタリングを使用して、独立 ASP を ALPHA から BETA に転送します。

IBM i *IBM i での切り替え独立 ASP ジャーナル構成*

複数インスタンスのキュー・マネージャーの構成を作成するために独立 ASP ジャーナルを複製する必要はありません。 アクティブ・キュー・マネージャーからスタンバイ・キュー・マネージャーに独立 ASP を転送する手段を自動化する必要があります。 独立 ASP を使用する代わりの高可用性ソリューションが存在します。すべての IASP で複数インスタンスのキュー・マネージャーの使用が必要なわけではありません。

独立 ASP を使用する際、キュー・マネージャー・ジャーナルをミラーリングする必要はありません。 クラスター管理がインストール済みであり、キュー・マネージャー・インスタンスをホストするサーバーが同じクラスター・リソース・グループ内にある場合、アクティブ・インスタンスを実行するホストで障害が発生したときに、アクティブ・サーバーから短距離内にある別のサーバーにキュー・マネージャー・ジャーナルを自動転送することができます。 計画済みの切り替えの一部としてジャーナルを手動で転送することも、コマンド・プロシージャーを作成して独立 ASP をプログラマチックに転送することもできます。

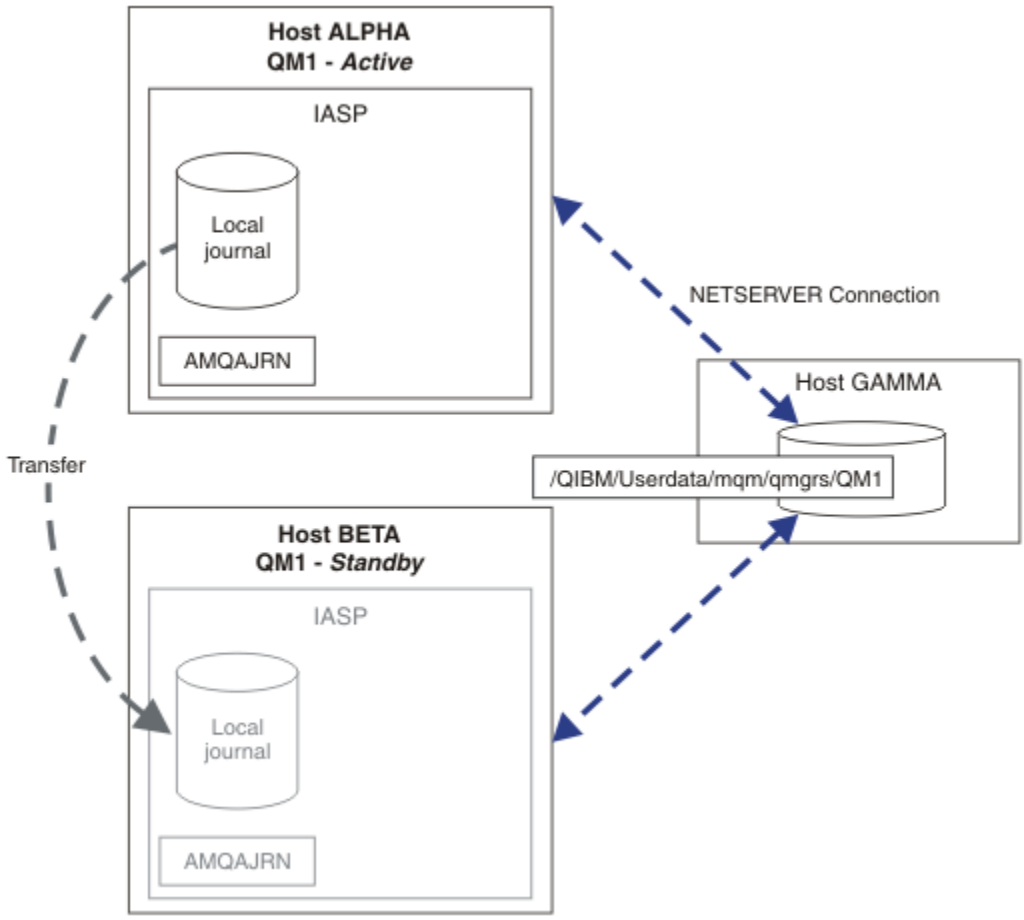


図 *27.* 独立 *ASP* を使ったキュー・マネージャー・ジャーナルの転送

複数インスタンスのキュー・マネージャーの操作では、キュー・マネージャーのデータを共有ファイル・システムに保管する必要があります。ファイル・システムは、さまざまなプラットフォームでホストすることができます。 複数インスタンスのキュー･マネージャーのデータを ASP または独立 ASP に保管することはできません。

共有ファイル・システムは、構成において 2 つの役割を果たします。 キュー・マネージャーのすべてのインスタンスの間で同じキュー・マネージャー・データが共有されます。 ファイル・システムには、開始後にキュー・マネージャーの 1 つのインスタンスしかキュー・マネージャー・データにアクセスできないようにする堅固なロック・プロトコルが必要です。 キュー・マネージャーで障害が発生するか、ファイル・サーバーとの通信が中断されると、ファイル・システムは、ファイル・システムとの通信が中断されたア

クティブ・インスタンスによって保持されているキュー・マネージャー・データに対するロックを解放します。 これにより、スタンバイ・キュー・マネージャー・インスタンスは、キュー・マネージャー・データに読み取り/書き込みアクセスを行えるようになります。 複数インスタンス・キュー・マネージャーと正しく連動するために、ファイル・システム・プロトコルが従わなければならない一連の規則があります。 412 ページの『IBM i の高可用性ソリューションのコンポーネント』を参照してください。

ロック機構は、キュー・マネージャーの開始コマンドを直列化し、アクティブなキュー・マネージャーのインスタンスを制御します。 キュー・マネージャーは、アクティブになった後、ユーザーまたは HA クラスターによってスタンバイ・サーバーに転送されたローカル・ジャーナルからそのキューを再作成します。 同じキュー・マネージャーへの再接続を待っている再接続可能クライアントは再接続され、未完了トランザクションはすべてバックアウトされます。 キュー・マネージャー・サービスとして開始するように構成されているアプリケーションは開始されます。

独立 ASP 上の障害が発生したアクティブ・キュー・マネージャー・インスタンスのローカル・ジャーナルが、新たにアクティブ化されるスタンバイ・キュー・マネージャー・インスタンスをホストするサーバーに転送されるようにする必要があります。これは、クラスター・リソース・マネージャーを構成するか、独立 ASP を手動で転送することによって行います。 バックアップおよび災害復旧で独立 ASP を使用し、複数インスタンスのキュー・マネージャー構成でリモート・ジャーナル・ミラーリングを使用することに決めた場合、独立 ASP を使用することでリモート・ジャーナルとミラーリングの構成が不要になるわけではありません。

独立 ASP を使用することにした場合、代わりの高可用性構成を検討することもできます。 これらのソリューションのバックグラウンドについては、439 ページの『独立 ASP および高可用性』を参照してください。

1. 複数インスタンスのキュー・マネージャーを使用するのではなく、単一インスタンスのキュー・マネージャーをもっぱら 1 つの独立 ASP にインストールして構成し、IBM i ハイ・アベイラビリティー・サービスを使用してキュー・マネージャーをフェイルオーバーします。 おそらく、サーバーと関係なくキュー・マネージャーで障害が発生したかどうかを検出するために、キュー・マネージャー・モニターでソリューションを補強する必要があります。 これは、「*Supportpac MC41: Configuring IBM MQ for iSeries for High Availability*」で説明されているソリューションの基礎となります。
2. 独立 ASP サイト間ミラーリング (XSM) を使用して、ローカル・バスで独立 ASP を切り替えるのではなく独立 ASP をミラーリングします。 これにより、独立 ASP ソリューションの地理的な範囲は、長距離でログ・レコードの書き込みにかかる時間が許す限り拡張されます。

IBM i *IBM i での独立 ASP および NetServer 使用による複数インスタンス・キュー・マネージャーの作成*

2 つの IBM i サーバー上で実行される複数インスタンス・キュー・マネージャーを作成します。 キュー・マネージャー・データは、NetServer を使用して IBM i サーバー上に保管されます。 キュー・マネージャーのジャーナルは、独立 ASP に保管されます。 IBM i クラスタリングまたは手動による手順を使用して、キュー・マネージャーのジャーナルを格納する独立 ASP をもう一方の IBM i サーバーに転送します。

始める前に

1. このタスクには、3 つの IBM i サーバーが必要です。 そのうちの 2 つ (この例では ALPHA と BETA) に IBM MQ をインストールします。 この製品は、IBM WebSphere MQ 7.0.1 Fix Pack 1 以上でなければなりません。
2. 第 3 のサーバーは、NetServer により ALPHA および BETA と接続された IBM i サーバーです。 これは、キュー・マネージャー・データの共有のために使用されます。 これに IBM MQ をインストールする必要はありません。 ただし、一時的なステップとしてこのサーバーに IBM MQ をインストールするなら、キュー・マネージャーのディレクトリーおよび許可をセットアップするのに役立ちます。
3. QMQM ユーザー・プロファイルのパスワードが、3 つのサーバーのすべてにおいて同じであることを確認してください。
4. IBM i NetServer をインストールします。 i5/OS NetServer を参照してください。
5. 障害が発生したキュー・マネージャーから引き継ぎ先スタンバイに独立 ASP を転送するためのプロシージャーを作成します。 「*SupportPac MC41: Configuring IBM MQ for iSeries for High Availability*」に記載されているいくつかの技法が、独立 ASP 転送プロシージャーを設計する上で役に立つかもしれません。

このタスクについて

以下の手順を実行することにより、438 ページの図 28 で示されている構成を作成します。 キュー・マネージャーのデータは、IBM i NetServer を使用して接続されます。

- ALPHA および BETA から、GAMMA 上でキュー・マネージャー・データの保管先となるディレクトリー共有への接続を作成します。 このタスクでは、必要な許可、ユーザー・プロファイル、およびパスワードもセットアップされます。
- IBM i サーバー ALPHA 上にキュー・マネージャー QM1 を作成します。
- もう一方の IBM i サーバー BETA 上の QM1 のキュー・マネージャー制御情報を追加します。
- キュー・マネージャーを始動して、スタンバイ・インスタンスが可能になるようにします。

手順

1. 413 ページの『IBM i での NetServer 使用によるキュー・マネージャー・データ用ネットワーク共有の作成』 のタスクを実行します。

   結果として、ALPHA および BETA には、 GAMMA 上の /QIBM/UserData/mqm/qmgrs を指す共有が /QNTC/GAMMA/WMQ ます。 ユーザー・プロファイル QMQM および QMQMADM には必要な許可が付与されており、3 つのシステムすべてにおいて QMQM のパスワードは一致しています。
2. ALPHA 上にキュー・マネージャー QM1 を作成し、キュー・マネージャー・データが GAMMA 上に保存されるようにします。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   パスは、 NetServer を使用してキュー・マネージャー・データを作成します。
3. ALPHA 上で作成された IBM MQ 構成データを検査するには、コマンドを使用します。

   この情報は次のステップで必要になります。

   この例では、以下の構成が ALPHA 上に作成されます。

   ```
   Name=QM1
   Prefix=/QIBM/UserData/mqm
   Library=QMQM1
   Directory=QM1
   DataPath= /QNTC/GAMMA/WMQ /QM1
   ```
4. 以下のコマンドを使用して、BETA 上に QM1 のキュー・マネージャー・インスタンスを作成します。 BETA 上で以下のコマンドを実行して、BETA 上のキュー・マネージャー制御情報を変更します。

   ```
   ADDMQMINF MQMNAME(QM1)
   PREFIX('/QIBM/UserData/mqm')
   MQMDIR(QM1)
   MQMLIB(QMQM1)
   DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
   ```

   ヒント : 構成情報をコピーして貼り付けます。 キュー・マネージャー・スタンザは、ALPHA と BETA で同じです。
5. ALPHA および BETA 上でキュー・マネージャー・インスタンスを始動します。

   a) ALPHA 上で最初のインスタンスを始動し、それをアクティブ・インスタンスにします。 スタンバイ・インスタンスへの切り替えを使用可能にします。

   ```
   STRMQM MQMNAME(QM1) STANDBY(*YES)
   ```

   b) BETA 上で 2 番目のインスタンスを始動し、それをスタンバイ・インスタンスにします。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

タスクの結果

キュー・マネージャーの状況を確認するために使用します。

1. ALPHA 上のキュー・マネージャー・インスタンスの状況は、次のようになります。
2. BETA 上のキュー・マネージャー・インスタンスの状況は、以下のようになっている必要があります。

例

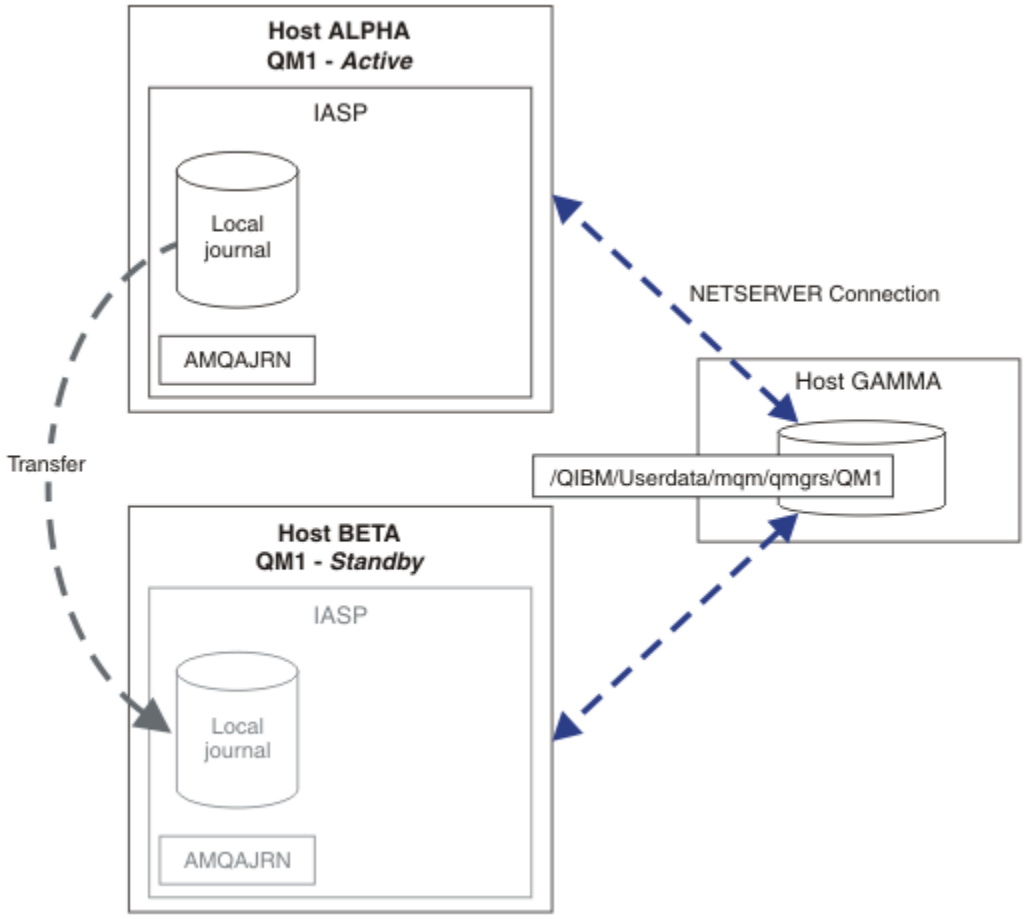


図 *28.* 独立 *ASP* を使ったキュー・マネージャー・ジャーナルの転送

次のタスク

- アクティブ・インスタンスとスタンバイ・インスタンスが自動的に切り替えられることを確認します。高可用性サンプル・プログラムを実行することにより、切り替えのテストを実施できます。『高可用性のサンプル・プログラム』を参照してください。 サンプル・プログラムは 'C' クライアントです。 それらは、Windows または UNIX プラットフォームから実行できます。
  1. 高可用性サンプル・プログラムを開始します。
  2. ALPHA 上で、キュー・マネージャーを終了して切り替えを要求します。

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. ベータ版のインスタンスがアクティブであることを確認してください。

4. ALPHA 上で再始動します。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

- 代替高可用性構成を確認します。
  1. NetServer を使用することにより、IBM i サーバー上にキュー・マネージャー・データを配置します。
  2. 独立 ASP を使用してキュー・マネージャーのジャーナルをスタンバイ・サーバーに転送する代わりに、リモート・ジャーナリングを使用して、ジャーナルをスタンバイ・サーバーにミラーリングします。

IBM i 独立 *ASP* および高可用性

独立 ASP では、アプリケーションとデータをサーバー間で移動させることができます。 独立 ASP に柔軟性があるということは、それがいくつかの IBM i 高可用性ソリューションの基本であることを意味します。 キュー・マネージャー・ジャーナルで ASP を使用するか独立 ASP を使用するかを検討する際には、独立 ASP に基づくその他の高可用性の構成を検討する必要があります。

補助ストレージ・プール (ASP) は、IBM i アーキテクチャーのビルディング・ブロックです。 複数のディスク装置がグループ化され、1 つの ASP を形成します。 異なる ASP にオブジェクトを配置することで、特定の ASP のデータを別の ASP でのディスク障害による影響から保護することができます。

どの IBM i サーバーにも、システム ASP と呼ばれる基本 ASP が少なくとも 1 つあります。 これは ASP1 として指定されます。*SYSBAS という名前の場合もあります。 追加基本ユーザー ASP を最大 31 個構成することができます。これは、アプリケーションの視点からシステム ASP と区別できません。それらは同じ名前空間を共有するからです。 複数の基本 ASP を使用してアプリケーションを多数のディスクに分散することで、パフォーマンスを向上させるとともに、回復時間を短縮することができます。 複数の基本 ASP を使用することである程度ディスク障害から分離されることにもなりますが、これによって全体的な信頼性が向上するわけではありません。

独立 ASP は、特殊なタイプの ASP です。 これはよく、独立ディスク・プールと呼ばれます。 独立ディスク・プールは、IBM i 高可用性のキー・コンポーネントです。 接続先の現行システムから独立していると見なされるデータおよびアプリケーションを独立したディスク・ストレージ・ユニットに保管することができます。 独立 ASP は、切り替え可能または切り替え不可能のものを構成できます。 可用性の観点からすれば、関係があるのは通常、切り替え可能の独立 ASP だけです。切り替え可能の独立 ASP は、サーバー間で自動転送することができます。 結果的に、独立 ASP 上のアプリケーションおよびデータをサーバー間で移動することができます。

基本ユーザー独立 ASP とは異なり、IASP はシステム ASP と同じ名前空間を共有しません。 ユーザー ASP を使って作業するアプリケーションが独立 ASP で作業するためには、変更を行う必要があります。 ソフトウェア、および使用するサード・パーティー・ソフトウェアが独立 ASP 環境で機能することを確認する必要があります。

独立 ASP が異なるサーバーに接続する場合、独立 ASP の名前空間をシステム ASP の名前空間と結合する必要があります。 このプロセスを、独立 ASP をオンに変更する といいます。 サーバーの IPL を実行せずに、独立 ASP をオンに変更することができます。 サーバー間で独立 ASP を自動転送するには、クラスタリング・サポートが必要です。

独立 ASP を使用した信頼できるソリューションの作成

障害が発生したキュー・マネージャー・インスタンスからのローカル・ジャーナルのコピーをスタンバイ・キュー・マネージャーに提供するための代替手段は、ASP へのジャーナリングおよびジャーナル複製の使用ではなく、独立 ASP へのジャーナリングです。 サーバー間の独立 ASP の自動転送を行うには、クラスタリング・サポートをインストールし、構成しておく必要があります。 独立 ASP には、クラスター・サポートと低レベルのディスク・ミラーリングに基づく、高可用性ソリューションが多数存在します。それらは、複数インスタンスのキュー・マネージャーと併用、または代用できます。

以下のリストでは、独立 ASP に基づく信頼できるソリューションを作成するために必要なコンポーネントを説明しています。

ジャーナリング
キュー・マネージャーおよびその他のアプリケーションはローカル・ジャーナルを使用して、サーバー障害に起因するメモリー内のデータの喪失から保護するために、持続データを安全にディスクに書き込みます。 これは、特定時点の整合性とも呼ばれます。 一定の期間に発生する複数の更新の整合性を保証するものではありません。

コミットメント制御
グローバル・トランザクションを使用することで、ジャーナルに書き込まれるデータが整合するようにメッセージおよびデータベースに対する更新を調整することができます。 これにより、2 フェーズ・コミット・プロトコルを使用することで一定期間、整合性が保たれます。

切り替えディスク

切り替えディスクは、HA クラスター内の装置クラスター・リソース・グループ (CRG) によって管理されます。 CRG は、計画外の停止の発生時に、独立 ASP を自動的に新しいサーバーに切り替えます。 CRG は地理的に、ローカル IO バスの範囲に限られます。

切り替え可能独立 ASP でローカル・ジャーナルを構成することで、別のサーバーにジャーナルを転送し、メッセージの処理を再開することができます。 独立 ASP が失敗しない限り、持続メッセージに対する変更は、同期点制御なしで行われる場合にも、同期点制御ありでコミットされる場合にも失われません。

切り替え可能独立 ASP に対してジャーナリング制御とコミットメント制御の両方を使用する場合、データベース・ジャーナルとキュー・マネージャー・ジャーナルを別のサーバーに転送し、整合性またはコミット済みトランザクションを失うことなくトランザクションの処理を再開します。

サイト間ミラーリング (XSM)
XSM は 1 次独立 ASP を地理的にリモートにある 2 次独立 ASP に TCP/IP ネットワークを介してミラーリングし、障害発生時に制御を自動的に転送します。 同期ミラーリングを構成することも、非同期ミラーリングを構成することも可能です。 同期ミラーリングの場合、実動システムでの書き込み操作が完了する前にデータがミラーリングされるためにキュー・マネージャーのパフォーマンスが低下しますが、2 次独立 ASP は確実に最新に保たれます。 一方、非同期ミラーリングを使用する場合、2 次独立 ASP は必ずしも最新に保たれません。 非同期ミラーリングでは、2 次独立 ASP の整合性が維持されます。

XSM には 3 つのテクノロジーがあります。

地理的ミラーリング
地理的ミラーリングはクラスタリングを拡張したもので、広範囲の独立 ASP の切り替えを可能にします。 これには同期モードと非同期モードの両方があります。 高可用性は同期モードでのみ保証されます。ただし、独立 ASP が分離されていることでパフォーマンスに多大な影響が及ぶこともあります。 地理的ミラーリングと切り替えディスクを併用することで、ローカルの高可用性とリモートの災害復旧を実現できます。

メトロ・ミラーリング
メトロ・ミラーリングは、ローカル・バスより長い距離において高速なローカル同期ミラーリングを提供するデバイス・レベルのサービスです。 これを複数インスタンスのキュー・マネージャーと併用することでキュー・マネージャーの高可用性を実現することができ、独立 ASP の 2 つのコピーを持つことで、キュー・マネージャー・ジャーナルの高可用性を実現できます。

グローバル・ミラーリング
グローバル・ミラーリングは非同期ミラーリングを提供する装置レベルのサービスであり、長距離のバックアップと災害復旧に適しています。 しかし、このミラーリングは特定時点の整合性を維持するに過ぎず、現行性は維持しないので、高可用性を望む場合の選択としては適していません。

決定の際に考慮すべき重要な点は、次のとおりです。

ASP と独立 ASP の間の選択
複数インスタンスのキュー・マネージャーを使用するために IBM i HA クラスターを実行する必要はありません。 すでに独立 ASP を使用している場合、あるいは独立 ASP を必要とする他のアプリケーションに関して可用性の要件がある場合には、独立 ASP を選択することができます。 独立 ASP を複数インスタンスのキュー・マネージャーと併用することで、キュー・マネージャー・モニターをキュー・マネージャーの障害を検出するための手段として代用することも有効かもしれません。

可用性
目標復旧時間 (RTO) はどれほどですか。 ほとんど中断を感じさせない動作が必要な場合、回復時間の最も短いソリューションはどれですか。

ジャーナルの可用性
Single Point of Failure としてジャーナルを除去する方法。 RAID 1 デバイス (またはそれより高性能のもの) を使用したハードウェア・ソリューションを採用する場合もあれば、レプリカ・ジャーナルまたはディスク・ミラーリングを使用したソフトウェア・ソリューションを使用または併用する場合もあります。

距離
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・インスタンスの間の距離。 ユーザーは、約 250 メートルを超える距離での同期的複製によって生じるパフォーマンスの低下を許容できますか。

スキル
ソリューションの定期的な維持および実行に関係する管理タスクを自動化するために行うべきタスクがあります。 自動化を行うために必要なスキルは、ASP に基づくソリューションか、独立 ASP に基づくソリューションかによって異なります。

IBM i *IBM i での複数インスタンス・キュー・マネージャーの削除*

複数インスタンス・キュー・マネージャーを削除する前に、リモート・ジャーナリングを停止し、キュー・マネージャー・インスタンスを除去してください。

始める前に

1. この例では、QM1 キュー・マネージャーの 2 つのインスタンスが、ALPHA および BETA というサーバー上に定義されています。 ALPHA がアクティブ・インスタンス、BETA がスタンバイです。 キュー・マネージャー QM1 に関連するキュー・マネージャー・データが、NetServer を使用することにより、GAMMA という IBM i サーバー上に保管されています。 426 ページの『IBM i でのジャーナル・ミラーリングおよび NetServer 使用による複数インスタンス・キュー・マネージャーの作成』を参照してください。
2. 定義されているリモート・ジャーナルのすべてを IBM MQ によって削除するには、ALPHA と BETA が接続されていなければなりません。
3. システム・コマンド **EDTF** または **WRKLNK** を使用することにより、/QNTC ディレクトリーとサーバー・ディレクトリー・ファイル共有がアクセス可能であることを検証します。

このタスクについて

DLTMQM コマンドを使用してサーバーから複数インスタンス・キュー・マネージャーを削除する前に、**RMVMQMINF** コマンドを使用することにより、他のサーバー上にあるキュー・マネージャー・インスタンスをすべて削除します。

RMVMQMINF コマンドを使用してキュー・マネージャー・インスタンスを削除すると、ローカルおよびリモートのジャーナルのうち、接頭部が AMQ でそのインスタンスに関連するものが削除されます。 ローカルからサーバーへのキュー・マネージャー・インスタンスに関する構成情報も削除されます。

キュー・マネージャーの残りのインスタンスが保持されているサーバー上では、**RMVMQMINF** コマンドを実行しないようにしてください。 そのようにすると、**DLTMQM** が正しく動作しません。

DLTMQM コマンドを使用してキュー・マネージャーを削除します。 キュー・マネージャー・データがネットワーク共有から削除されます。 ローカルおよびリモートのジャーナルのうち接頭部が AMQ でそのインスタンスに関連するものが削除されます。 さらに、**DLTMQM** により、ローカルからサーバーへのキュー・マネージャー・インスタンスに関する構成情報も削除されます。

この例の場合、キュー・マネージャー・インスタンスは 2 個のみです。 IBM MQ では、1 個のアクティブ・キュー・マネージャー・インスタンスと 1 個のスタンバイ・インスタンスによる複数インスタンスの実行構成がサポートされています。 実行構成で使用するために、さらに付加的なキュー・マネージャー・インスタンスを作成した場合は、残りのインスタンスを削除する前に、**RMVMQMINF** コマンドを使用することによってそれらを削除してください。

手順

1. 各サーバーで **CHGMQMJRN RMTJRNSTS** (*INACTIVE) コマンドを実行して、キュー・マネージャー・インスタンス間のリモート・ジャーナリングを非アクティブにします。

   a) ALPHA 上で、

   ```
   CHGMQMJRN MQMNAME('QM1')
   RMTJRNRDB('BETA') RMTJRNSTS(*INACTIVE)
   ```

   b) BETA 上で、

   ```
   CHGMQMJRN MQMNAME('QM1')
   RMTJRNRDB('ALPHA') RMTJRNSTS(*INACTIVE)
   ```

2. アクティブ・キュー・マネージャー・インスタンスである ALPHA 上で **ENDMQM** コマンドを実行することにより、QM1 の 2 つのインスタンスを停止します。

   ```
   ENDMQM MQMNAME(QM1) OPTION(*IMMED) INSTANCE(*ALL) ENDCCTJOB(*YES)
   ```

3. ALPHA 上で **RMVMQMINF** コマンドを実行することにより、ALPHA から BETA へのインスタンスのキュー・マネージャー・リソースを削除します。

   ```
   RMVMQMINF MQMNAME(QM1)
   ```

   RMVMQMINF により、QM1 のキュー・マネージャー構成情報が ALPHA から削除されます。 ジャーナル名の接頭部が AMQ の場合、QM1 に関連するローカル・ジャーナルが ALPHA から削除されます。 ジャーナル名の接頭部が AMQ の場合にリモート・ジャーナルが作成されていたなら、BETA のリモート・ジャーナルも削除されます。

4. BETA 上で **DLTMQM** コマンドを実行することにより、QM1 を削除します。

   ```
   DLTMQM MQMNAME(QM1)
   ```

   DLTMQM により、GAMMA 上のネットワーク共有からキュー・マネージャー・データが削除されます。 それにより QM1 のキュー・マネージャー構成情報が BETA から削除されます。 ジャーナル名の接頭部が AMQ の場合、QM1 に関連するローカル・ジャーナルが BETA から削除されます。 ジャーナル名の接頭部が AMQ の場合にリモート・ジャーナルが作成されていたなら、ALPHA のリモート・ジャーナルも削除されます。

タスクの結果

DLTMQM および **RMVMQMINF** により、**CRTMQM** および **ADDMQJRN** によって作成されたローカル・ジャーナルとリモート・ジャーナルが削除されます。 また、これらのコマンドによりジャーナル・レシーバーも削除されます。 ジャーナルおよびジャーナル・レシーバーは、名前が AMQ で始まるという命名規則に従うものでなければなりません。 **DLTMQM** および **RMVMQMINF** により、キュー・マネージャー・オブジェクト、キュー・マネージャー・データ、およびキュー・マネージャー構成情報が mqs.ini から削除されます。

次のタスク

別の方法として、ステップ 442 ページの『1』 でジャーナリングを非アクティブ化した後、キュー・マネージャー・インスタンスを終了する前に、以下のコマンドを発行することもできます。 あるいは、命名規則に従っていない場合には、名前を指定してジャーナルおよびジャーナル・レシーバーを削除する必要があります。

1. ALPHA 上で、

```
RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('BETA')
```

2. BETA 上で、

```
RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('ALPHA')
```

ジャーナルを削除した後、残りのステップを続行します。

IBM i *IBM i での複数インスタンス・キュー・マネージャーのバックアップ*

この手順は、ローカル・サーバー上にあるキュー・マネージャーのさまざまなオブジェクト、およびネットワーク・ファイル・サーバー上にあるキュー・マネージャー・データをバックアップする方法を示します。 他のキュー・マネージャーについてデータをバックアップする場合は、それに応じてサンプルを修正してください。

始める前に

この例では、キュー・マネージャー QM1 に関連するキュー・マネージャー・データが、NetServer を使用することにより、GAMMA という IBM i サーバー上に保管されています。 426 ページの『IBM i でのジャーナル・ミラーリングおよび NetServer 使用による複数インスタンス・キュー・マネージャーの作成』を参照してください。 サーバー ALPHA および BETA には、IBM MQ がインストールされています。 ALPHA および BETA 上にキュー・マネージャー QM1 が構成されています。

このタスクについて

IBM i では、リモート・ディレクトリーからのデータの保存はサポートされていません。 キュー・マネージャー・データをリモート・ファイル・システムに保存するには、ファイル・システム・サーバーにローカルなバックアップ・プロシージャーを使用します。 このタスクでは、ネットワーク・ファイル・システムは IBM i サーバー (GAMMA) 上にあります。 キュー・マネージャー・データは、GAMMA 上の保存ファイルの中にバックアップされます。

ネットワーク・ファイル・システムが Windows または Linux 上にある場合については、キュー・マネージャー・データを圧縮ファイル中に保管した後、それを保存することも可能です。 Tivoli Storage Manager などのバックアップ・システムを使用している場合は、それを使用してキュー・マネージャー・データのバックアップを実行してください。

手順

1. ALPHA 上に、QM1 に関連するキュー・マネージャー・ライブラリーに対応する保存ファイルを作成します。

   保存ファイルの名前には、キュー・マネージャー・ライブラリー名を使用します。

   ```
   CRTSAVF FILE(QGPL/QMQM1)
   ```

2. キュー・マネージャー・ライブラリーを ALPHA 上の保存ファイルに保存します。

   ```
   SAVLIB LIB(QMQM1) DEV(*SAVF) SAVF(QGPL/QMQM1)
   ```

3. GAMMA 上で、キュー・マネージャー・データ・ディレクトリーのための保存ファイルを作成します。

   保存ファイルの名前には、キュー・マネージャーの名前を使用します。

   ```
   CRTSAVF FILE(QGPL/QMDQM1)
   ```

4. GAMMA 上のローカル・ディレクトリーから、キュー・マネージャー・データのコピーを保存します。

```
SAV DEV('/QSYS.LIB/QGPL.LIB/QMDQM1.FILE') OBJ('/QIBM/Userdata/mqm/qmgrs/QM1')
```

IBM i 複数インスタンスのキュー・マネージャーをセットアップするためのコマンド

IBM MQ には、ジャーナル複製の構成、新しいキュー・マネージャー・インスタンスの追加、および独立 ASP の使用のためのキュー・マネージャーの構成を単純化するコマンドがあります。

ローカルとリモートのジャーナルを作成および管理するためのジャーナル・コマンドは、次のとおりです。

ADDMQMJRN
このコマンドを使用して、キュー・マネージャー・インスタンス用の指定されたローカルおよびリモートのジャーナルを作成し、複製が同期か非同期か、同期タイムアウトはどのくらいか、リモート・ジャーナルを直ちにアクティブ化するかどうかに関して構成することができます。

CHGMQMJRN
このコマンドでは、レプリカ・ジャーナルに影響を与えるタイムアウト、状況、および送達パラメーターを変更します。

RMVMQMJRN
指定されたリモート・ジャーナルをキュー・マネージャー・インスタンスから除去します。

WRKMQMJRN
ローカルのキュー・マネージャー・インスタンスに関するローカルおよびリモートのジャーナルの状況をリストします。

キューマネージャーインスタンスの追加と管理は、`mqs.ini` ファイルを修正する以下のコマンドを使用します。

ADDMQMINF
このコマンドは、 `mqs.ini` ファイルから抽出した情報を使用して、 DSPMQMINF コマンドを使用して、別の IBM i サーバー上に新しいキュー・マネージャー・インスタンスを追加します。

RMVMQMINF
キュー・マネージャー・インスタンスを除去します。 このコマンドは、既存のキュー・マネージャーのインスタンスを除去するか、別のサーバーから削除されたキュー・マネージャーの構成情報を除去するために使用します。

CRTMQM コマンドには、複数インスタンスのキュー・マネージャーの構成を支援する以下の 3 つのパラメーターがあります。

MQMDIRP(*DFT| *directory-prefix*)
このパラメーターは、ネットワーク・ストレージ上のキュー・マネージャー・データにマップされるマウント・ポイントを選択するために使用します。

ASP(*SYSTEM|*ASPDEV|*auxiliary-storage-pool-number*)
キュー・マネージャー・ジャーナルをシステムまたは基本ユーザー ASP に配置するには、*SYSTEM または *auxiliary-storage-pool-number* を指定します。 キュー・マネージャー・ジャーナルを独立 ASP に配置するには、*ASPDEV オプションを選択し、さらに **ASPDEV** パラメーターを使用して装置名も設定します。

ASPDEV(*ASP|*device-name*)
1 次独立 ASP 装置または 2 次独立 ASP 装置の *device-name* を指定します。 *ASP を選択した場合、**ASP** (*SYSTEM) を指定した場合と同じ結果が得られます。

IBM i IBM i でのパフォーマンスおよびディスク・フェイルオーバーの考慮事項

異なる補助ストレージ・プールを使用して、パフォーマンスと信頼性を向上させます。

多数の持続メッセージや、サイズの大きなメッセージをアプリケーションで使用する場合、それらのメッセージをディスクに書き込む際に費やされる時間は、システムのパフォーマンスにおける大きな要因になります。

この可能性を処理するのに十分なディスクの活動量を確保するか、またはキュー・マネージャーのジャーナル・レシーバーを独立した補助ストレージ・プール ASP に置くことを検討してください。

ASP パラメーター **CRTMQM** を使用してキュー・マネージャーを作成するときに、キュー・マネージャーのライブラリーとジャーナルをどの ASP に保管するかを指定できます。 デフォルトでは、キュー・マネージャーのライブラリーとジャーナル、および IFS データは、システム ASP に保管されます。

ASP により、1 つ以上の特定のディスク装置にあるオブジェクトを分離できます。 また、これによりディスク・メディアの障害によるデータの喪失を削減できます。 多くの場合、影響を受けた ASP のディスク装置に保管されていたデータが失われるだけです。

フェイルオーバーを提供し、ディスクの競合を削減するためには、キュー・マネージャーのライブラリーとジャーナル・データを、異なるユーザー ASP のルート IFS ファイル・システムに保管することをお勧めします。

詳しくは、 IBM i 資料の バックアップおよびリカバリー を参照してください。

IBM i SAVLIB を使用した IBM i 上の IBM MQ ライブラリーの保管

SAVLIB LIB(*ALLUSR) を使用して IBM MQ ライブラリーを保管することはできません。これらのライブラリーの名前は Q で始まるためです。

SAVLIB LIB(QM*) を使用すると、すべてのキュー・マネージャー・ライブラリーを保管できますが、*SAVF 以外の保管装置を使用している場合に限ります。 DEV(*SAVF) の場合は、システム上のキュー・マネージャー・ライブラリーごとに SAVLIB コマンドを使用する必要があります。

IBM i IBM MQ for IBM i の静止

このセクションでは、IBM MQ for IBM i を静止 (穏やかに終了) する方法について説明します。

IBM MQ for IBM i を静止するには、以下を行います。

1. 新しいインタラクティブ IBM MQ for IBM i セッションにサインオンし、オブジェクトにアクセスしていないことを確認します。
2. 以下のことを確認します。
   - *ALLOBJ 権限、または QMQM ライブラリーについてのオブジェクト管理権限
   - ENDSBS コマンドを使用するのに十分な権限
3. すべてのユーザーに対して、IBM MQ for IBM i を停止しようとしていることを通知します。
4. 次の処理方法は、シャットダウン (静止) の対象が (他のキュー・マネージャーが存在する場合に) 単一のキュー・マネージャーである (446 ページの『IBM MQ for IBM i の単一のキュー・マネージャーのシャットダウン』を参照) か、すべてのキュー・マネージャーである (447 ページの『IBM MQ for IBM i のすべてのキュー・マネージャーのシャットダウン』を参照) かによって異なります。
5. qshell で次のコマンドを入力して、mqweb サーバーをシャットダウンします。

   ```
   /QIBM/ProdData/mqm/bin/endmqweb
   ```

ENDMQM パラメーター ENDCCTJOB(*YES)

IBM MQ for IBM i V6.0 以降では、ENDMQM パラメーター ENDCCTJOB(*YES) は、前バージョンとは異なる働きをします。

前バージョンでは、ENDCCTJOB(*YES) を指定すると、MQ はアプリケーションを 強制終了させます。

IBM MQ for IBM i V6.0 以降では、ENDCCTJOB(*YES) を指定すると、アプリケーションは終了されませんが、キュー・マネージャーから切断されます。

ENDCCTJOB(*YES) が指定されていて、キュー・マネージャーが終了しつつあることを検出するようアプリケーションが作られていない場合は、次回新しい MQI 呼び出しが行われると、その呼び出しに対して MQRC_CONNECTION_BROKEN (2009) エラーが戻されます。

ENDCCTJOB(*YES) の代替方法としては、パラメーター ENDCCTJOB(*NO) を使用します。 そして、WRKMQM オプション 22 (ジョブの取り扱い) を使用して、キュー・マネージャーの再始動を妨げるアプリケーション・ジョブをすべて手動で終了させます。

IBM i IBM MQ for IBM i の単一のキュー・マネージャーのシャットダウン

この情報は、3 つのタイプのシャットダウンについて理解するために使用します。

次の手順では、QMgr1 というサンプル・キュー・マネージャー名と、SUBX というサブシステム名を使用しています。 これらの名前は、必要に応じて、独自の値と置き換えてください。

計画的シャットダウン

IBM i でのキュー・マネージャーの計画的シャットダウン

1. シャットダウンの前に、以下を実行します。

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

2. キュー・マネージャーをシャットダウンするには、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

   QMgr1 が終了しない場合は、チャネルまたはアプリケーションが使用中である可能性があります。
3. QMgr1 を即時にシャットダウンする必要がある場合は、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

計画外シャットダウン

1. キュー・マネージャーをシャットダウンするには、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

   QMgr1 が終了しない場合は、チャネルまたはアプリケーションが使用中である可能性があります。
2. QMgr1 を即時にシャットダウンする必要がある場合は、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

異常条件でのシャットダウン

1. キュー・マネージャーをシャットダウンするには、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

   QMgr1 が終了しない場合、次に該当する場合はステップ 3 に進んでください。

   - QMgr1 がそれ自体のサブシステムに含まれている。
   - QMgr1 を同一のサブシステムとして共有しているすべてのキュー・マネージャーを終了できる。 これらすべてのキュー・マネージャーに対し、計画外シャットダウン手順を使用します。
2. サブシステム (上記の例では SUBX) を共有するすべてのキュー・マネージャーについて上記の手順の全ステップを完了した後、次のコマンドを実行します。

```
ENDSBS SUBX *IMMED
```

このコマンドが完了に失敗した場合は、計画外シャットダウン手順を使用して、すべてのキュー・マネージャーをシャットダウンし、ご使用のマシンで IPL を実行します。

警告 : 直後にマシンで IPL を実行する準備ができていない限り、 ENDJOB または ENDSBS の結果として終了しない IBM MQ ジョブには、 ENDJOBABN を使用しないでください。

3. 次のコマンドを実行して、サブシステムを開始します。

```
STRSBS SUBX
```

4. 次のコマンドを実行して、キュー・マネージャーを即時にシャットダウンします。

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(10)
```

5. 次のコマンドを実行して、サブシステムを再始動します。

```
STRMQM MQMNAME(QMgr1)
```

これが正常に行われず、次のいずれかに該当する場合は、

- IPL を実行してマシンを再始動している。
- キュー・マネージャーが 1 つのみ存在する。

次のコマンドを実行して、IBM MQ の共有メモリーをタイディアップします。

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

上記のコマンドは、ステップ 5 を繰り返す前に実行します。

キュー・マネージャーの再始動に数秒程度より多くの時間がかかる場合、IBM MQ は開始状況の詳細を示す状況メッセージを、断続的にジョブ・ログに追加します。

それでもキュー・マネージャーの再始動に問題がある場合は、IBM サポート担当までご連絡ください。上記以外の処置を行うと、キュー・マネージャーを損傷し、IBM MQ が回復できなくなる可能性があります。

IBM i IBM MQ for IBM i のすべてのキュー・マネージャーのシャットダウン

この情報は、3 つのタイプのシャットダウンについて理解するために使用します。

手順は単一キュー・マネージャーの場合とほぼ同じですが、該当箇所でキュー・マネージャー名ではなく *ALL を使用します。または、それぞれのキュー・マネージャー名を順番に使用しながらコマンドを繰り返し使用します。 次の手順では、QMgr1 というサンプル・キュー・マネージャー名と、SUBX というサブシステム名を使用しています。 この部分をご使用のものと置き換えてください。

計画的シャットダウン

1. シャットダウンの 1 時間前に、次のコマンドを実行します。

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

シャットダウンをしたいキュー・マネージャーごとに、上記のコマンドを繰り返します。

2. キュー・マネージャーをシャットダウンするには、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

シャットダウンしたいキュー・マネージャーごとに、上記のコマンドを繰り返します。異なるコマンドは、並行して実行できます。

妥当な時間 (10 分など) 内に終了しないキュー・マネージャーがある場合は、ステップ 3 へ進みます。

3. すべてのキュー・マネージャーを即時にシャットダウンするには、次のコマンドを実行します。

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

計画外シャットダウン

1. キュー・マネージャーをシャットダウンするには、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

シャットダウンしたいキュー・マネージャーごとに、上記のコマンドを繰り返します。異なるコマンドは、並行して実行できます。

キュー・マネージャーが終了しない場合は、チャネルまたはアプリケーションが使用中である可能性があります。

2. キュー・マネージャーを即時にシャットダウンする必要がある場合は、次のコマンドを実行します。

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

異常条件でのシャットダウン

1. キュー・マネージャーをシャットダウンするには、次のコマンドを実行します。

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

シャットダウンしたいキュー・マネージャーごとに、上記のコマンドを繰り返します。異なるコマンドは、並行して実行できます。

2. 次のコマンドを実行して、サブシステム (この例では SUBX) を終了します。

```
ENDSBS SUBX *IMMED
```

シャットダウンしたいサブシステムごとに、上記のコマンドを繰り返します。異なるコマンドは、並行して実行できます。

このコマンドが正常に完了できなかった場合は、ご使用のシステムで IPL を実行します。

警告 : ENDJOB または ENDSBS を実行すると正常に終了できなくなるジョブに対しては、ご使用のシステムで直後に IPL を実行する用意ができている場合を除き、ENDJOBABN を使用しないでください。

3. 次のコマンドを実行して、サブシステムを開始します。

```
STRSBS SUBX
```

開始したいサブシステムごとに、上記のコマンドを繰り返します。

4. 次のコマンドを実行して、キュー・マネージャーを即時にシャットダウンします。

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

5. 次のコマンドを実行して、キュー・マネージャーを再始動します。

```
STRMQM MQMNAME(QMgr1)
```

開始したいキュー・マネージャーごとに、上記のコマンドを繰り返します。

どのキュー・マネージャーの再始動にも数秒以上かかる場合、IBM MQ は開始状況の詳細を示す状況メッセージを定期的に表示します。

それでもキュー・マネージャーの再始動に問題がある場合は、IBM サポート担当までご連絡ください。上記以外の処置を行うと、キュー・マネージャーを損傷し、MQSeries® または IBM MQ が回復できなくなる可能性があります。

z/OS Administering IBM MQ for z/OS

IBM MQ for z/OS can be controlled and managed by MQSC and PCF commands, by a set of utilities and programs provided with the product, and by authorized applications.

For details of how to administer IBM MQ for z/OS and the different administrative tasks you might have to undertake, see the following links.

You can also administer IBM MQ for z/OS using the IBM MQ Explorer running in a Linux shell. For more information, see “IBM MQ Explorer を使用した管理” on page 118.

Related concepts

IBM MQ for z/OS concepts

Related tasks

“IBM MQ の管理” on page 7
IBM MQ キュー・マネージャーと関連リソースを管理する時には、そうしたリソースをアクティブ化したり管理したりするための一連のタスクから好みの方法を選択できます。

Planning your IBM MQ environment on z/OS

Configuring queue managers on z/OS

z/OS Issuing queue manager commands on z/OS

You can control most of the operational environment of IBM MQ by using control commands. You can issue MQSC and PCF commands from the IBM MQ for z/OS console, the initialization input data sets, the batch utility CSQUTIL, or authorized applications.

About this task

You use MQSC commands, in batch or interactive mode, to administer queue managers directly. You use PCF commands to help you create applications that administer queue managers. MQSC commands are in human-readable text form, whereas PCF commands let applications create requests and read the replies without having to parse text strings. Like MQSC commands, applications issue PCF commands by sending them as messages to the command input queue.

The following topics describe how you issue queue manager commands from the IBM MQ for z/OS console, the initialization input data sets, the batch utility CSQUTIL, or from authorized applications.

Not all commands can be issued from all sources. See “Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS” on page 449.

Related tasks

Preparing sample applications for the TSO environment on z/OS

Related information

Administering IBM MQ using MQSC commands

z/OS Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS

You can issue MQSC and PCF commands from the IBM MQ for z/OS console, the initialization input data sets, the batch utility CSQUTIL, or from authorized applications. Not all commands can be issued from all these sources.

Which MQSC and PCF commands can control each IBM MQ object

Table 1 in "Command summary for IBM MQ for z/OS" maps which MQSC and PCF commands can be used on IBM MQ for z/OS to alter, define, delete and display each IBM MQ object. See also MQSC commands reference and “IBM MQ プログラマブル・コマンド・フォーマットの使用” on page 26.

List of sources from which commands can be issued

If you are a suitably authorized user, you can issue IBM MQ commands from the following sources:

- The z/OS console or equivalent (such as SDSF/TSO).

  See also “Using the operations and control panels on z/OS” on page 463.

  Note: When using the z/OS console, you need to add `/cpf` to the start of a command, where `cpf` is the command prefix for the queue manager subsystem.
- The initialization input data sets CSQINP1, CSQINP2, CSQINPT and CSQINPX.

  See “Initialization commands for IBM MQ for z/OS” on page 460.
- The z/OS master get command routine, MGCRE (SVC 34).
- The IBM MQ batch utility programs such as CSQUTIL, which processes a list of commands in a sequential data set.

  See “Using the IBM MQ for z/OS utilities” on page 471.
- Suitably authorized applications, sending commands as messages to the SYSTEM.COMMAND.INPUT queue.

  The application can be any of the following:

  – A batch region program
  – A CICS application
  – An IMS application
  – A TSO application
  – An application program or utility on another IBM MQ system

  See “Writing programs to administer IBM MQ for z/OS” on page 479 and Preparing sample applications for the TSO environment on z/OS.

Not all commands can be issued from all sources

Commands are classified according to where they can be issued from:

1
CSQINP1

2
CSQINP2

C
The z/OS console

R
The command server and command queue, by means of CSQUTIL, CSQINPT, CSQINPX, or authorized applications.

Within the command descriptions in MQSC commands reference, these sources are identified by the use of the characters 1, 2, C, and R in each command description. Table 2 in "Command summary for IBM MQ for z/OS" summarizes the MQSC commands and the sources from which they can be issued.

Related tasks
Preparing sample applications for the TSO environment on z/OS

Related information

Administering IBM MQ using MQSC commands

z/OS Command summary for IBM MQ for z/OS

A summary of the main MQSC and PCF commands, and of the sources from which you can run MQSC commands on IBM MQ for z/OS.

Table 25 on page 451 maps which MQSC and PCF commands can be used on IBM MQ for z/OS to alter, define, delete and display each IBM MQ object.

Table 25. Summary of the main MQSC and PCF commands by object type

MQSC command	**ALTER**	**DEFINE**	**DISPLAY**	**DELETE**
PCF command	**Change**	**Create/Copy**	**Inquire**	**Delete**
AUTHINFO	X	X	X	X
CFSTATUS			X	
CFSTRUCT	X	X	X	X
CHANNEL	X	X	X	X
CHSTATUS			X	
NAMELIST	X	X	X	X
PROCESS	X	X	X	X
QALIAS	M	M	M	M
QCLUSTER			M	
QLOCAL	M	M	M	M
QMGR	X		X	
QMODEL	M	M	M	M
QREMOTE	M	M	M	M
QUEUE	P	P	X	P
QSTATUS			X	
STGCLASS	X	X	X	X

Key to table symbols:

- M = MQSC only
- P = PCF only
- X = both

There are many other MQSC and PCF commands which allow you to manage other IBM MQ resources, and carry out other actions in addition to those summarized in Table 25 on page 451.

Table 26 on page 452 shows every MQSC command, and where each command can be issued from.

- CSQINP1 initialization input data set
- CSQINP2 initialization input data set
- z/OS console (or equivalent)
- SYSTEM.COMMAND.INPUT queue and command server (from applications, CSQUTIL, or the CSQINPX initialization input data set)

Table 26. Sources from which to run MQSC commands

Command	CSQINP1	CSQINP2	z/OS console	Command input queue and server
ALTER AUTHINFO		X	X	X
ALTER BUFFPOOL		X	X	X
ALTER CFSTRUCT		X	X	X
ALTER CHANNEL		X	X	X
ALTER NAMELIST		X	X	X
ALTER PROCESS		X	X	X
ALTER PSID			X	X
ALTER QALIAS		X	X	X
ALTER QLOCAL		X	X	X
ALTER QMGR		X	X	X
ALTER QMODEL		X	X	X
ALTER QREMOTE		X	X	X
ALTER SECURITY	X	X	X	X
ALTER SMDS		X	X	X
ALTER STGCLASS		X	X	X
ALTER SUB			X	X
ALTER TOPIC		X	X	X
ALTER TRACE	X	X	X	X
ARCHIVE LOG	X	X	X	X
BACKUP CFSTRUCT			X	X
CLEAR QLOCAL		X	X	X
CLEAR TOPICSTR			X	X
DEFINE AUTHINFO		X	X	X
DEFINE BUFFPOOL	X			
DEFINE CFSTRUCT		X	X	X
DEFINE CHANNEL		X	X	X
DEFINE LOG			X	X
DEFINE MAXSMSGS		X	X	X
DEFINE NAMELIST		X	X	X
DEFINE PROCESS		X	X	X
DEFINE PSID	X		X	X
DEFINE QALIAS		X	X	X
DEFINE QLOCAL		X	X	X

Table 26. Sources from which to run MQSC commands (continued)

Command	CSQINP1	CSQINP2	z/OS console	Command input queue and server
DEFINE QMODEL		X	X	X
DEFINE QREMOTE		X	X	X
DEFINE STGCLASS		X	X	X
DEFINE SUB			X	X
DEFINE TOPIC		X	X	X
DELETE AUTHINFO		X	X	X
DELETE BUFFPOOL		X	X	X
DELETE CFSTRUCT		X	X	X
DELETE CHANNEL			X	X
DELETE NAMELIST		X	X	X
DELETE PROCESS		X	X	X
DELETE PSID			X	X
DELETE QALIAS		X	X	X
DELETE QLOCAL		X	X	X
DELETE QMODEL		X	X	X
DELETE QREMOTE		X	X	X
DELETE STGCLASS		X	X	X
DELETE SUB			X	X
DELETE TOPIC		X	X	X
DISPLAY ARCHIVE	X	X	X	X
DISPLAY AUTHINFO		X	X	X
DISPLAY CFSTATUS			X	X
DISPLAY CFSTRUCT		X	X	X
DISPLAY CHANNEL		X	X	X
DISPLAY CHINIT			X	X
DISPLAY CHLAUTH		X	X	X
DISPLAY CHSTATUS			X	X
DISPLAY CLUSQMGR			X	X
DISPLAY CMDSERV	X	X	X	X
DISPLAY CONN		X	X	X
DISPLAY GROUP		X	X	X
DISPLAY LOG	X	X	X	X
DISPLAY MAXSMSGS		X	X	X

Table 26. Sources from which to run MQSC commands (continued)

Command	CSQINP1	CSQINP2	z/OS console	Command input queue and server
DISPLAY NAMELIST		X	X	X
DISPLAY PROCESS		X	X	X
DISPLAY PUBSUB		X	X	X
DISPLAY QALIAS		X	X	X
DISPLAY QCLUSTER		X	X	X
DISPLAY QLOCAL		X	X	X
DISPLAY QMGR		X	X	X
DISPLAY QMODEL		X	X	X
DISPLAY QREMOTE		X	X	X
DISPLAY QSTATUS		X	X	X
DISPLAY QUEUE		X	X	X
DISPLAY SBSTATUS			X	X
DISPLAY SECURITY			X	X
DISPLAY SMDS		X	X	X
DISPLAY SMDSCONN		X	X	X
DISPLAY STGCLASS		X	X	X
DISPLAY SUB			X	X
DISPLAY SYSTEM	X	X	X	X
DISPLAY TCLUSTER		X	X	X
DISPLAY THREAD		X	X	X
DISPLAY TOPIC		X	X	X
DISPLAY TPSTATUS		X	X	X
DISPLAY TRACE	X	X	X	X
DISPLAY USAGE		X	X	X
MOVE QLOCAL		X	X	X
PING CHANNEL			X	X
RECOVER BSDS			X	X
RECOVER CFSTRUCT			X	X
REFRESH CLUSTER			X	X
REFRESH QMGR		X	X	X
REFRESH SECURITY			X	X
RESET CFSTRUCT			X	X
RESET CHANNEL			X	X

Table 26. Sources from which to run MQSC commands (continued)				
Command	**CSQINP1**	**CSQINP2**	**z/OS console**	**Command input queue and server**
RESET CLUSTER			X	X
RESET QMGR		X	X	X
RESET QSTATS		X	X	X
RESET SMDS			X	X
RESET TPIPE			X	X
RESOLVE CHANNEL			X	X
RESOLVE INDOUBT		X	X	X
RESUME QMGR			X	X
RVERIFY SECURITY		X	X	X
SET ARCHIVE	X	X	X	X
SET CHLAUTH		X	X	X
SET LOG	X	X	X	X
SET SYSTEM	X	X	X	X
START CHANNEL			X	X
START CHINIT		X	X	X
START CMDSERV	X	X	X	
START LISTENER			X	X
START QMGR			X	
START SMDSCONN		X	X	X
START TRACE	X	X	X	X
STOP CHANNEL			X	X
STOP CHINIT			X	X
STOP CMDSERV	X	X	X	
STOP LISTENER			X	X
STOP QMGR			X	X
STOP SMDSCONN		X	X	X
STOP TRACE	X	X	X	X
SUSPEND QMGR			X	X

In MQSC commands, each command description identifies the sources from which that command can be run.

z/OS Using MQSC to start and stop a queue manager on z/OS

An introduction to using control commands on IBM MQ for z/OS: After you have installed IBM MQ, use MQSC commands to start and stop a queue manager.

Before you begin

After you have installed IBM MQ, it is defined as a formal z/OS subsystem. This message appears during any initial program load (IPL) of z/OS:

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm INITIALIZATION COMPLETE
```

where *ssnm* is the IBM MQ subsystem name.

From now on, you can start the queue manager for that subsystem *from any z/OS console that has been authorized to issue system control commands* ; that is, a z/OS SYS command group. You must issue the START command from the authorized console, you cannot issue it through JES or TSO.

If you are using queue sharing groups, you must start RRS first, and then Db2®, before you start the queue manager.

About this task

When a queue manager stops under normal conditions, its last action is to take a termination checkpoint. This checkpoint, and the logs, give the queue manager the information it needs to restart.

The following steps contain information about the START and STOP commands, and contain a brief overview of start-up after an abnormal termination has occurred.

Procedure

1. Start a queue manager

   You start a queue manager by issuing a START QMGR command. However, you cannot successfully use the START command unless you have appropriate authority. See the Setting up security on z/OS for information about IBM MQ security. The following code shows examples of the START command. Note that you must prefix an MQSC command with a command prefix string (CPF).

   ```
   +CSQ1  START QMGR
   +CSQ1  START QMGR PARM(NEWLOG)
   ```

   See START QMGR for information about the syntax of the START QMGR command.

   You cannot run the queue manager as a batch job or start it using a z/OS command START. These methods are likely to start an address space for IBM MQ that then ends abnormally. Nor can you start a queue manager from the CSQUTIL utility program or a similar user application.

   You can, however, start a queue manager from an APF-authorized program by passing a START QMGR command to the z/OS MGCRE (SVC 34) service.

   If you are using queue sharing groups, the associated Db2 systems and RRS must be active when you start the queue manager.

   Start options

   When you start a queue manager, a system parameter module is loaded. You can specify the name of the system parameter module in one of two ways:

   - With the PARM parameter of the /cpf START QMGR command, for example

     ```
     /cpf START QMGR PARM(CSQ1ZPRM)
     ```

   - With a parameter in the startup procedure, for example, code the JCL EXEC statement as

     ```
     //MQM  EXEC PGM=CSQYASCP,PARM='ZPARM(CSQ1ZPRM)'
     ```

A system parameter module provides information specified when the queue manager was customized.

You can use the **QMGRPROD** option to specify the product against which the queue manager usage is to be recorded, and the **AMSPROD** option to specify the equivalent for AMS if that is used. See the MQSC START QMGR command for details of the permitted values.

An example JCL EXEC statement follows:

```
//MQM EXEC  PGM=CSQYASCP,PARM='QMGRPROD(MQ)'
```

See z/OS MVS Product Management for more information on measured usage and product registration.

You can also use the ENVPARM option to substitute one or more parameters in the JCL procedure for the queue manager.

For example, you can update your queue manager startup procedure, so that the **DDname** CSQINP2 is a variable. This means that you can change the CSQINP2 **DDname** without changing the startup procedure. This is useful for implementing changes, providing back-outs for operators, and queue manager operations.

Suppose your start-up procedure for queue manager CSQ1 looked like this:

```
//CSQ1MSTR PROC INP2=NORM
//MQMESA  EXEC PGM=CSQYASCP
//STEPLIB DD  DISP=SHR,DSN=thlqual.SCSQANLE
//     DD  DISP=SHR,DSN=thlqual.SCSQAUTH
//     DD  DISP=SHR,DSN=db2qual.SDSNLOAD
//BSDS1  DD  DISP=SHR,DSN=myqual.BSDS01
//BSDS2  DD  DISP=SHR,DSN=myqual.BSDS02
//CSQP0000 DD  DISP=SHR,DSN=myqual.PSID00
//CSQP0001 DD  DISP=SHR,DSN=myqual.PSID01
//CSQP0002 DD  DISP=SHR,DSN=myqual.PSID02
//CSQP0003 DD  DISP=SHR,DSN=myqual.PSID03
//CSQINP1 DD  DISP=SHR,DSN=myqual.CSQINP(CSQ1INP1)
//CSQINP2 DD  DISP=SHR,DSN=myqual.CSQINP(CSQ1&INP2.)
//CSQOUT1 DD  SYSOUT=*
//CSQOUT2 DD  SYSOUT=*
```

If you then start your queue manager with the following command:

```
+CSQ1  START QMGR
```

then the CSQINP2 used is a member called CSQ1NORM.

However, suppose you are putting a new suite of programs into production so that the next time you start queue manager CSQ1, the CSQINP2 definitions are to be taken from member CSQ1NEW. To do this, you would start the queue manager with this command:

```
+CSQ1  START QMGR ENVPARM('INP2=NEW')
```

and CSQ1NEW would be used instead of CSQ1NORM. Note: z/OS limits the KEYWORD=value specifications for symbolic parameters (as in INP2=NEW) to 255 characters.

Starting after an abnormal termination

IBM MQ automatically detects whether restart follows a normal shutdown or an abnormal termination.

Starting a queue manager after it ends abnormally is different from starting it after the STOP QMGR command has been issued. After STOP QMGR, the system finishes its work in an orderly way and takes a termination checkpoint before stopping. When you restart the queue manager, it uses information from the system checkpoint and recovery log to determine the system status at shutdown.

However, if the queue manager ends abnormally, it terminates without being able to finish its work or take a termination checkpoint. When you restart a queue manager after an abend, it refreshes its knowledge of its status at termination using information in the log, and notifies you of the status of various tasks. Normally, the restart process resolves all inconsistent states. But, in some cases, you must take specific steps to resolve inconsistencies.

User messages on start-up

When you start a queue manager successfully, the queue manager produces a set of startup messages.

2. Stop a queue manager.

Before stopping a queue manager, all IBM MQ-related write-to-operator-with-reply (WTOR) messages must receive replies, for example, getting log requests. Each of the following commands terminates a running queue manager.

```
+CSQ1  STOP QMGR
+CSQ1  STOP QMGR MODE(QUIESCE)
+CSQ1  STOP QMGR MODE(FORCE)

+CSQ1  STOP QMGR MODE(RESTART)
```

The command STOP QMGR defaults to STOP QMGR MODE(QUIESCE).

In QUIESCE mode, IBM MQ does not allow any new connection threads to be created, but allows existing threads to continue; it terminates only when all threads have ended. Applications can request to be notified in the event of the queue manager quiescing. Therefore, use the QUIESCE mode where possible so that applications that have requested notification have the opportunity to disconnect. See What happens during termination for details.

If the queue manager does not terminate in a reasonable time in response to a STOP QMGR MODE(QUIESCE) command, use the DISPLAY CONN command to determine whether any connection threads exist, and take the necessary steps to terminate the associated applications. If there are no threads, issue a STOP QMGR MODE(FORCE) command.

The STOP QMGR MODE(QUIESCE) and STOP QMGR MODE(FORCE) commands deregister IBM MQ from the MVS Automatic Restart Manager (ARM), preventing ARM from restarting the queue manager automatically. The STOP QMGR MODE(RESTART) command works in the same way as the STOP QMGR MODE(FORCE) command, except that it does not deregister IBM MQ from ARM. This means that the queue manager is eligible for immediate automatic restart.

If the IBM MQ subsystem is not registered with ARM, the STOP QMGR MODE(RESTART) command is rejected and the following message is sent to the z/OS console:

```
CSQY205I ARM element arm-element is not registered
```

If this message is not issued, the queue manager is restarted automatically. For more information about ARM, see “Using the z/OS Automatic Restart Manager (ARM)” on page 538.

Only cancel the queue manager address space if STOP QMGR MODE(FORCE) does not terminate the queue manager.

If a queue manager is stopped by either canceling the address space or by using the command STOP QMGR MODE(FORCE), consistency is maintained with connected CICS or IMS systems. Resynchronization of resources is started when a queue manager restarts and is completed when the connection to the CICS or IMS system is established.

Note: When you stop your queue manager, you might find message IEF352I is issued. z/OS issues this message if it detects that failing to mark the address space as unusable would lead to an integrity exposure. You can ignore this message.

Stop messages

After issuing a STOP QMGR command, you get the messages CSQY009I and CSQY002I, for example:

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM
USER(userid), STOP MODE(FORCE)
CSQY002I +CSQ1 QUEUE MANAGER STOPPING
```

Where `userid` is the user ID that issued the STOP QMGR command, and the MODE parameter depends on that specified in the command.

When the STOP command has completed successfully, the following messages are displayed on the z/OS console:

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

If you are using ARM, and you did not specify MODE(RESTART), the following message is also displayed:

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type
arm-element-type successful
```

You cannot restart the queue manager until the following message has been displayed:

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```

z/OS Issuing commands from a z/OS console or equivalent

You can issue IBM MQ MQSC and PCF commands from a z/OS console or its equivalent. You can also issue IBM MQ commands from anywhere where you can issue z/OS commands, such as SDSF or by a program using the MGCRE macro. When using the z/OS console, you add `/cpf` to the start of a command.

Before you begin

Not all commands can be issued by the z/OS console. Within the command description topics (the children of MQSC commands reference), each command that can be issued by the console is identified by the character 'C'. Table 2 in "Command summary for IBM MQ for z/OS" summarizes the MQSC commands and the sources from which they can be issued.

You cannot issue IBM MQ commands using the IMS/SSR command format from an IMS terminal. This function is not supported by the IMS adapter.

The input field provided by SDSF might not be long enough for some commands, particularly those commands for channels.

The maximum amount of data that can be displayed as a result of a command typed in at the console is 32 KB.

About this task

If you are a suitably authorized user, you can issue IBM MQ commands from the z/OS console or equivalent (such as SDSF/TSO).

When using the z/OS console, you need to add `/cpf` to the start of a command, where `cpf` is the command prefix for the queue manager subsystem.

The following steps refer to commands and attributes using their MQSC command names rather than their PCF names.

Procedure

- Use command prefix strings

  Each IBM MQ command must be prefixed with a command prefix string (CPF).

  Because more than one IBM MQ subsystem can run under z/OS, the CPF is used to indicate which IBM MQ subsystem processes the command.

  For example, to start the queue manager for a subsystem called CSQ1, where CPF is ' +CSQ1 ', you issue the following command from the operator console:

  ```
  +CSQ1 START QMGR
  ```

  This CPF must be defined in the subsystem name table (for the subsystem CSQ1), as described in Defining command prefix strings (CPFs). In the examples, the string +CSQ1 is used as the command prefix.
- Use the z/OS console to issue commands

  You can type simple commands from the z/OS console, for example:

  ```
  +CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLOCAL)
  ```

  However, for complex commands or for sets of commands that you issue frequently, the other methods of issuing commands are better.
- Receive command responses

  Direct responses to commands are sent to the console that issued the command. IBM MQ supports the *Extended Console Support* (EMCS) function available in z/OS, and therefore consoles with 4 byte IDs can be used. Additionally, all commands except START QMGR and STOP QMGR support the use of Command and Response Tokens (CARTs) when the command is issued by a program using the MGCRE macro.

Related tasks

“Using the operations and control panels on z/OS” on page 463
You use these panels for defining, displaying, altering, or deleting IBM MQ objects. Use the panels for day-to-day administration and for making small changes to objects.

Preparing sample applications for the TSO environment on z/OS

z/OS Initialization commands for IBM MQ for z/OS

Initialization commands can be used to control the queue manager startup.

Commands in the initialization input data sets are processed when IBM MQ is initialized on queue manager startup. Three types of command can be issued from the initialization input data sets:

- Commands to define IBM MQ entities that cannot be defined elsewhere, for example DEFINE BUFFPOOL.

  These commands must reside in the data set identified by the DD name CSQINP1. They are processed before the restart phase of initialization. They cannot be issued through the console, operations and control panels, or an application program. The responses to these commands are written to the sequential data set that you refer to in the CSQOUT1 statement of the started task procedure.
- Commands to define IBM MQ objects that are recoverable after restart. These definitions must be specified in the data set identified by the DD name CSQINP2. They are stored in page set zero. CSQINP2 is processed after the restart phase of initialization. The responses to these commands are written to the sequential data set that you refer to in the CSQOUT2 statement of the started task procedure.
- Commands to manipulate IBM MQ objects. These commands must also be specified in the data set identified by the DD name CSQINP2. For example, the IBM MQ-supplied sample contains an ALTER

QMGR command to specify a dead-letter queue for the subsystem. The response to these commands is written to the CSQOUT2 output data set.

Note: If IBM MQ objects are defined in CSQINP2, IBM MQ attempts to redefine them each time the queue manager is started. If the objects already exist, the attempt to define them fails. If you need to define your objects in CSQINP2, you can avoid this problem by using the REPLACE parameter of the DEFINE commands, however, this overrides any changes that were made during the previous run of the queue manager.

Sample initialization data set members are supplied with IBM MQ for z/OS. They are described in Sample definitions supplied with IBM MQ.

Initialization commands for distributed queuing

You can also use the CSQINP2 initialization data set for the START CHINIT command. If you need a series of other commands to define your distributed queuing environment (for example, starting listeners), IBM MQ provides a third initialization input data set, called CSQINPX, that is processed as part of the channel initiator started task procedure.

The MQSC commands contained in the data set are executed at the end of channel initiator initialization, and output is written to the data set specified by the CSQOUTX DD statement. You might use the CSQINPX initialization data set to start listeners for example.

A sample channel initiator initialization data set member is supplied with IBM MQ for z/OS. It is described in Sample definitions supplied with IBM MQ.

Initialization commands for publish/Subscribe

If you need a series of commands to define your publish/subscribe environment (for example, when defining subscriptions), IBM MQ provides a fourth initialization input data set, called CSQINPT.

The MQSC commands contained in the data set are executed at the end of publish/subscribe initialization, and output is written to the data set specified by the CSQOUTT DD statement. You might use the CSQINPT initialization data set to define subscriptions for example.

A sample publish/subscribe initialization data set member is supplied with IBM MQ for z/OS. It is described in Sample definitions supplied with IBM MQ.

z/OS Private and global definitions on IBM MQ for z/OS

When you define an object on IBM MQ for z/OS, you can choose whether you want to share that definition with other queue managers (a *global* definition), or whether the object definition is to be used by one queue manager only (a *private* definition). This is called the object *disposition*.

Global definition

If your queue manager belongs to a queue sharing group, you can choose to share any object definitions you make with the other members of the group. This means that you have to define an object once only, reducing the total number of definitions required for the whole system.

Global object definitions are held in a *shared repository* (a Db2 shared database), and are available to all the queue managers in the queue sharing group. These objects have a disposition of GROUP.

Private definition

If you want to create an object definition that is required by one queue manager only, or if your queue manager is not a member of a queue sharing group, you can create object definitions that are not shared with other members of a queue sharing group.

Private object definitions are held on page set zero of the defining queue manager. These objects have a disposition of QMGR.

You can create private definitions for all types of IBM MQ objects except CF structures (that is, channels, namelists, process definitions, queues, queue managers, storage class definitions, and authentication information objects), and global definitions for all types of objects except queue managers.

IBM MQ automatically copies the definition of a group object to page set zero of each queue manager that uses it. You can alter the copy of the definition temporarily if you want, and IBM MQ allows you to refresh the page set copies from the repository copy if required.

IBM MQ always tries to refresh the page set copies from the repository copy at startup (for channel commands, this is done when the channel initiator restarts), or if the group object is changed.

Note: The copy of the definition is refreshed from the definition of the group, only if the definition of the group has changed after you created the copy of the definition.

This ensures that the page set copies reflect the version on the repository, including any changes that were made when the queue manager was inactive. The copies are refreshed by generating DEFINE REPLACE commands, therefore there are circumstances under which the refresh is not performed, for example:

- If a copy of the queue is open, a refresh that changes the usage of the queue fails.
- If a copy of a queue has messages on it, a refresh that deletes that queue fails.
- If a copy of a queue would require ALTER with FORCE to change it.

In these circumstances, the refresh is not performed on that copy, but is performed on the copies on all other queue managers.

If the queue manager is shut down and then restarted stand-alone, any local copies of objects are deleted, unless for example, the queue has associated messages.

There is a third object disposition that applies to local queues only. This allows you to create shared queues. The definition for a shared queue is held on the shared repository and is available to all the queue managers in the queue sharing group. In addition, the messages on a shared queue are also available to all the queue managers in the queue sharing group. This is described in Shared queues and queue sharing groups. Shared queues have an object disposition of SHARED.

The following table summarizes the effect of the object disposition options for queue managers started stand-alone, and as a member of a queue sharing group.

Disposition	Stand-alone queue manager	Member of a queue sharing group
QMGR	Object definition held on page set zero.	Object definition held on page set zero.
GROUP	Not allowed.	Object definition held in the shared repository. Local copy held on page set zero of each queue manager in the group.
SHARED	Not allowed.	Queue definition held in the shared repository. Messages available to any queue manager in the group.

Manipulating global definitions

If you want to change the definition of an object that is held in the shared repository, you need to specify whether you want to change the version on the repository, or the local copy on page set zero. Use the object disposition as part of the command to do this.

z/OS Directing commands to different queue managers on z/OS

You can use the *command scope* to control on which queue manager the command runs.

You can choose to execute a command on the queue manager where it is entered, or on a different queue manager in the queue sharing group. You can also choose to issue a particular command in parallel on all the queue managers in a queue sharing group. This is possible for both MQSC commands and PCF commands.

This is determined by the *command scope*. The command scope is used with the object disposition to determine which version of an object you want to work with.

For example, you might want to alter some of the attributes of an object, the definition of which is held in the shared repository.

- You might want to change the version on one queue manager only, and not make any changes to the version on the repository or those in use by other queue managers.
- You might want to change the version in the shared repository for future users, but leave existing copies unchanged.
- You might want to change the version in the shared repository, but also want your changes to be reflected immediately on all the queue managers in the queue sharing group that hold a copy of the object on their page set zero.

Use the command scope to specify whether the command is executed on this queue manager, another queue manager, or all queue managers. Use the object disposition to specify whether the object you are manipulating is in the shared repository (a group object), or is a local copy on page set zero (a queue manager object).

You do not have to specify the command scope and object disposition to work with a shared queue because every queue manager in the queue sharing group handles the shared queue as a single queue.

z/OS Using the operations and control panels on z/OS

You use these panels for defining, displaying, altering, or deleting IBM MQ objects. Use the panels for day-to-day administration and for making small changes to objects.

Before you begin

The IBM MQ for z/OS operations and controls panels (CSQOREXX) might not support all new function and parameters added from version 7 onwards. For example, there are no panels for the direct manipulation of topic objects or subscriptions. Use one of the following supported mechanisms to administer publish/subscribe definitions and other system controls that are not directly available from other panels:

1. IBM MQ Explorer
2. z/OS console
3. Programmable Command Format (PCF) messages
4. COMMAND function of CSQUTIL
5. IBM MQ Console

Note that the generic **Command** action in the CSQOREXX panels allows you to issue any valid MQSC command, including SMDS related commands. You can use all the commands that the COMMAND function of CSQUTIL issues.

You cannot issue the IBM MQ commands directly from the command line in the panels.

To use the operations and control panels, you must have the correct security authorization; this is described in the User IDs for command security and command resource security.

You cannot provide a user ID and password using CSQUTIL, or the CSQOREXX panels. Instead, if you user ID has UPDATE authority to the BATCH profile in MQCONN, you can bypass the **CHCKLOCL**(*REQUIRED* setting. See Using **CHCKLOCL** on locally bound applications for more information.

If you are setting up or changing many objects, use the COMMAND function of the CSQUTIL utility program. See “Using the CSQUTIL utility for IBM MQ for z/OS” on page 473.

About this task

The operations and control panels support the controls for the channel initiator (for example, to start a channel or a TCP/IP listener), for clustering, and for security. They also enable you to display information about threads and page set usage.

The panels work by sending MQSC type IBM MQ commands to a queue manager, through the system command input queue.

Example

This is the panel that displays when you start a panel session:

```
                  IBM MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . 1     0. List with filter   4. Manage
                                  1. List or Display    5. Perform
                                  2. Define like        6. Start
                                  3. Alter              7. Stop
                                  8. Command
Object type . . . . . . . . CHANNEL       +
Name  . . . . . . . . . . . *
Disposition . . . . . . . . A  Q=Qmgr, C=Copy, P=Private, G=Group,
                               S=Shared, A=All

Connect name  . . . . . . . MQ1C  - local queue manager or group
Target queue manager  . . . MQ1C
        - connected or remote queue manager for command input
Action queue manager  . . . MQ1C  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993, 2024. All rights reserved.

Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

From this panel you can perform actions such as these:

- Choose the local queue manager you want and whether you want the commands issued on that queue manager, on a remote queue manager, or on another queue manager in the same queue sharing group as the local queue manager. Over type the queue manager name if you need to change it.
- Select the action you want to perform by typing in the appropriate number in the **Action** field.
- Specify the object type that you want to work with. Press function key F1 for help about the object types if you are not sure what they are.
- Specify the disposition of the object type that you want to work with.
- Display a list of objects of the type specified. Type in an asterisk (*) in the **Name** field and press **Enter** to display a list of objects (of the type specified) that have already been defined on the action queue manager. You can then select one or more objects to work with in sequence. All the actions are available from the list.

Note: You are recommended to make choices that result in a list of objects being displayed, and then work from that list. Use the **Display** action, because that is allowed for all object types.

z/OS Invocation and rules for the operations and control panels

You can control IBM MQ and issue control commands through the ISPF panels.

How to access the IBM MQ operations and control panels

If the ISPF/PDF primary options menu has been updated for IBM MQ, you can access the IBM MQ operations and control panels from that menu. For details about updating the menu, see the Task 20: Set up the operations and control panels.

You can access the IBM MQ operations and control panels from the TSO command processor panel (typically option 6 on the ISPF/PDF primary options menu). The name of the exec that you run to do this is CSQOREXX. It has two parameters; `thlqual` is the high-level qualifier for the IBM MQ libraries to be used, and `langletter` is the letter identifying the national language libraries to be used (for example, E for U.S. English). The parameters can be omitted if the IBM MQ libraries are permanently installed in your ISPF setup. Alternatively, you can issue CSQOREXX from the TSO command line.

These panels are designed to be used by operators and administrators with a minimum of formal training. Read these instructions with the panels running and try out the different tasks suggested.

Note: While using the panels, temporary dynamic queues with names of the form SYSTEM.CSQOREXX.* are created.

Rules for the operations and control panels

See Rules for naming IBM MQ objects about the general rules for IBM MQ character strings and names. However, there are some rules that apply only to the operations and control panels:

- Do not enclose strings, for example descriptions, in single or double quotation marks.
- If you include an apostrophe or quotation mark in a text field, you do not have to repeat it or add an escape character. The characters are saved exactly as you type them; for example:

  ```
  This is Maria's queue
  ```

  The panel processor doubles them for you to pass them to IBM MQ. However, if it has to truncate your data to do this, it does so.
- You can use uppercase or lowercase characters in most fields, and they are folded to uppercase characters when you press Enter. The exceptions are:
  - Storage class names and coupling facility structure names, which must start with uppercase A through Z and be followed by uppercase A through Z or numeric characters.
  - Certain fields that are not translated. These include:
    - Application ID
    - Description
    - Environment data
    - Object names (but if you use a lowercase object name, you might not be able to enter it at a z/OS console)
    - Remote system name
    - Trigger data
    - User data
- In names, leading blanks and leading underscores are ignored. Therefore, you cannot have object names beginning with blanks or underscores.
- Underscores are used to show the extent of blank fields. When you press Enter, trailing underscores are replaced by blanks.
- Many description and text fields are presented in multiple parts, each part being handled by IBM MQ independently. This means that trailing blanks are retained and the text is not contiguous.

Blank fields

When you specify the **Define** action for an IBM MQ object, each field on the define panel contains a value. See the general help (extended help) for the display panels for information about where IBM MQ gets the values. If you type over a field with blanks, and blanks are not allowed, IBM MQ puts the installation default value in the field or prompts you to enter the required value.

When you specify the **Alter** action for an IBM MQ object, each field on the alter panel contains the current value for that field. If you type over a field with blanks, and blanks are not allowed, the value of that field is unchanged.

z/OS Objects and actions on z/OS

The operations and control panels offer you many different types of object and a number of actions that you can perform on them.

The actions are listed on the initial panel and enable you to manipulate the objects and display information about them. These objects include all the IBM MQ objects, together with some extra ones. The objects fall into the following categories.

- Queues, processes, authentication information objects, namelists, storage classes and CF structures
- Channels
- Cluster objects
- Queue manager and security
- Connections
- System

Refer to Actions for a cross-reference table of the actions which can be taken with the IBM MQ objects.

Queues, processes, authentication information objects, namelists, storage classes and CF structures

These are the basic IBM MQ objects. There can be many of each type. They can be listed, listed with filter, defined, and deleted, and have attributes that can be displayed and altered, using the LIST or DISPLAY, LIST with FILTER, DEFINE LIKE, MANAGE, and ALTER actions. (Objects are deleted using the MANAGE action.)

This category consists of the following objects:

QLOCAL	Local queue
QREMOTE	Remote queue
QALIAS	Alias queue for indirect reference to a queue
QMODEL	Model queue for defining queues dynamically
QUEUE	Any type of queue
QSTATUS	Status of a local queue
PROCESS	Information about an application to be started when a trigger event occurs
AUTHINFO	Authentication information: definitions required to perform Certificate Revocation List (CRL) checking using LDAP servers
NAMELIST	List of names, such as queues or clusters
STGCLASS	Storage class
CFSTRUCT	coupling facility (CF) structure
CFSTATUS	Status of a CF structure

Channels

Channels are used for distributed queuing. There can be many of each type, and they can be listed, listed with filter, defined, deleted, displayed, and altered. They also have other functions available using the START, STOP and PERFORM actions. PERFORM provides reset, ping, and resolve channel functions.

This category consists of the following objects:

CHANNEL	Any type of channel
SENDER	Sender channel
SERVER	Server channel

RECEIVER	Receiver channel
REQUESTER	Requester channel
CLUSRCVR	Cluster-receiver channel
CLUSSDR	Cluster-sender channel
SVRCONN	Server-connection channel
CLNTCONN	Client-connection channel
CHSTATUS	Status of a channel connection

Cluster objects

Cluster objects are created automatically for queues and channels that belong to a cluster. The base queue and channel definitions can be on another queue manager. There can be many of each type, and names can be duplicated. They can be listed, listed with filter, and displayed. PERFORM, START, and STOP are also available through the LIST actions.

This category consists of the following objects:

CLUSQ	Cluster queue, created for a queue that belongs to a cluster
CLUSCHL	Cluster channel, created for a channel that belongs to a cluster
CLUSQMGR	Cluster queue manager, the same as a cluster channel but identified by its queue manager name

Cluster channels and cluster queue managers do have the PERFORM, START and STOP actions, but only indirectly through the DISPLAY action.

Queue manager and security

Queue manager and security objects have a single instance. They can be listed, and have attributes that can be displayed and altered (using the LIST or DISPLAY, and ALTER actions), and have other functions available using the PERFORM action.

This category consists of the following objects:

MANAGER	Queue manager: the PERFORM action provides suspend and resume cluster functions
SECURITY	Security functions: the PERFORM action provides refresh and reverify functions

Connection

Connections can be listed, listed with filter and displayed.

This category consists only of the connection object, CONNECT.

System

A collection of other functions. This category consists of the following objects:

SYSTEM	System functions
CONTROL	Synonym for SYSTEM

The functions available are:

LIST or DISPLAY	Display queue sharing group, distributed queuing, page set, or data set usage information.
PERFORM	Refresh or reset clustering
START	Start the channel initiator or listeners

STOP Stop the channel initiator or listeners

Actions

The actions that you can perform for each type of object are shown in the following table:

Table 27. Valid operations and control panel actions for IBM MQ objects

Object	Alter	Define like	Manage (1)	List or Display	List with Filter	Perform	Start	Stop
AUTHINFO	X	X	X	X	X			
CFSTATUS				X				
CFSTRUCT	X	X	X	X	X			
CHANNEL	X	X	X	X	X	X	X	X
CHSTATUS				X	X			
CLNTCONN	X	X	X	X	X			
CLUSCHL				X	X	X(2)	X(2)	X(2)
CLUSQ				X	X			
CLUSQMGR				X	X	X(2)	X(2)	X(2)
CLUSRCVR	X	X	X	X	X	X	X	X
CLUSSDR	X	X	X	X	X	X	X	X
CONNECT				X	X			
CONTROL				X		X	X	X
MANAGER	X			X		X		
NAMELIST	X	X	X	X	X			
PROCESS	X	X	X	X	X			
QALIAS	X	X	X	X	X			
QLOCAL	X	X	X	X	X			
QMODEL	X	X	X	X	X			
QREMOTE	X	X	X	X	X			
QSTATUS				X	X			
QUEUE	X	X	X	X	X			
RECEIVER	X	X	X	X	X	X	X	X
REQUESTER	X	X	X	X	X	X	X	X
SECURITY	X			X		X		
SENDER	X	X	X	X	X	X	X	X
SERVER	X	X	X	X	X	X	X	X
SVRCONN	X	X	X	X	X		X	X
STGCLASS	X	X	X	X	X			
SYSTEM				X		X	X	X

Note:

1. Provides `Delete` and other functions
2. Using the `List or Display` action

z/OS Object dispositions on z/OS

You can specify the *disposition* of the object with which you need to work. The disposition signifies where the object **definition** is kept, and how the object behaves.

The disposition is significant only if you are working with any of the following object types:

- queues
- channels
- processes
- namelists
- storage classes
- authentication information objects

If you are working with other object types, the disposition is disregarded.

Permitted values are:

Q
QMGR. The object definitions are on the page set of the queue manager and are accessible only by the queue manager.

C
COPY. The object definitions are on the page set of the queue manager and are accessible only by the queue manager. They are local copies of objects defined as having a disposition of GROUP.

P
PRIVATE. The object definitions are on the page set of the queue manager and are accessible only by the queue manager. The objects have been defined as having a disposition of QMGR or COPY.

G
GROUP. The object definitions are in the shared repository, and are accessible by all queue managers in the queue sharing group.

S
SHARED. This disposition applies only to local queues. The queue definitions are in the shared repository, and are accessible by all queue managers in the queue sharing group.

A
ALL. If the action queue manager is either the target queue manager, or *, objects of **all** dispositions are included; otherwise, objects of QMGR and COPY dispositions only are included. This is the default.

z/OS Selecting a queue manager, defaults, and levels using the ISPF control panel on z/OS

You can use the CSQOREXX exec in ISPF to control your queue managers.

While you are viewing the initial panel, you are not connected to any queue manager. However, as soon as you press Enter, you are connected to the queue manager, or a queue manager in the queue sharing group named in the **Connect name** field. You can leave this field blank; this means that you are using the default queue manager for batch applications. This is defined in CSQBDEFV (see Task 19: Set up Batch, TSO, and RRS adapters for information about this).

Use the **Target queue manager** field to specify the queue manager where the actions you request are to be performed. If you leave this field blank, it defaults to the queue manager specified in the **Connect name** field. You can specify a target queue manager that is not the one you connect to. In this case, you would normally specify the name of a remote queue manager object that provides a queue manager alias

definition (the name is used as the *ObjectQMgrName* when opening the command input queue). To do this, you must have suitable queues and channels set up to access the remote queue manager.

The **Action queue manager** field allows you to specify a queue manager that is in the same queue sharing group as the queue manager specified in the **Target queue manager** field to be the queue manager where the actions you request are to be performed. If you specify * in this field, the actions you request are performed on all queue managers in the queue sharing group. If you leave this field blank, it defaults to the value specified in the **Target queue manager** field. The **Action queue manager** field corresponds to using the CMDSCOPE command modifier described in The MQSC commands.

Queue manager defaults

If you leave any queue manager fields blank, or choose to connect to a queue sharing group, a secondary window opens when you press **Enter**. This window confirms the names of the queue managers you will be using. Press **Enter** to continue. When you return to the initial panel after having made some requests, you find fields completed with the actual names.

Queue manager levels

If the action queue manager is not at IBM MQ 8.0.0 or later, some fields are not displayed, and some values cannot be entered. A few objects and actions are disallowed. In such cases, a secondary window opens asking for you to confirm that you want to proceed.

z/OS Using the function keys and command line with the ISPF control panels on z/OS

To use the panels, you must use the function keys or enter the equivalent commands in the ISPF control panel command area.

- Function keys
  - Processing your actions
  - “Displaying IBM MQ user messages” on page 471
  - Canceling your actions
  - Getting help
- Using the command line

Function keys

The function keys have special settings for IBM MQ. (This means that you cannot use the ISPF default values for the function keys; if you have previously used the `KEYLIST OFF ISPF` command anywhere, you must type `KEYLIST ON` in the command area of any operations and control panel and then press Enter to enable the IBM MQ settings.)

These function key settings can be displayed on the panels, as shown in “Using the operations and control panels on z/OS” on page 463. If the settings are not shown, type `PFSHOW` in the command area of any operations and control panel and then press **Enter**. To remove the display of the settings, use the command `PFSHOW OFF`.

The function key settings in the operations and control panels conform to CUA standards. Although you can change the key setting through normal ISPF procedures (such as the **KEYLIST** utility), you are not recommended to do so.

Note: Using the **PFSHOW** and **KEYLIST** commands affects any other logical ISPF screens that you have, and their settings remain when you leave the operations and control panels.

Processing your actions

Press **Enter** to carry out the action requested on a panel. The information from the panel is sent to the queue manager for processing.

Each time you press **Enter** in the panels, IBM MQ generates one or more operator messages. If the operation was successful, you get confirmation message CSQ9022I, otherwise you get some error messages.

Displaying IBM MQ user messages
Press function key F10 in any panel to see the IBM MQ user messages.

Canceling your actions
On the initial panel, both F3 and F12 exit the operations and control panels and return you to ISPF. No information is sent to the queue manager.

On any other panel, press function keys F3 or F12 to leave the current panel **ignoring any data you have typed since last pressing Enter**. Again, no information is sent to the queue manager.

- F3 takes you straight back to the initial panel.
- F12 takes you back to the previous panel.

Getting help

Each panel has help panels associated with it. The help panels use the ISPF protocols:

- Press function key F1 on any panel to see general help (extended help) about the task.
- Press function key F1 with the cursor on any field to see specific help about that field.
- Press function key F5 from any field help panel to get the general help.
- Press function key F3 to return to the base panel, that is, the panel from which you pressed function key F1.
- Press function key F6 from any help panel to get help about the function keys.

If the help information carries on into a second or subsequent pages, a **More** indicator is displayed in the upper-right of the panel. Use these function keys to navigate through the help pages:

- F11 to get to the next help page (if there is one).
- F10 to get back to the previous help page (if there is one).

Using the command line

You never need to use the command line to issue the commands used by the operations and control panels because they are available from function keys. The command line is provided to allow you to enter normal ISPF commands (like **PFSHOW**).

The ISPF command PANELID ON displays the name of the current CSQOREXX panel.

The command line is initially displayed in the default position at the bottom of the panels, regardless of what ISPF settings you have. You can use the SETTINGS ISPF command from any of the operations and control panels to change the position of the command line. The settings are remembered for subsequent sessions with the operations and control panels.

z/OS Using the IBM MQ for z/OS utilities

IBM MQ for z/OS provides a set of utility programs that you can use to help with system administration.

IBM MQ for z/OS supplies a set of utility programs to help you perform various administrative tasks, including the following:

- Manage message security policies.
- Perform backup, restoration, and reorganization tasks.
- Issue commands and process object definitions.
- Generate data-conversion exits.
- Modify the bootstrap data set.

- List information about the logs.
- Print the logs.
- Set up Db2 tables and other Db2 utilities.
- Process messages on the dead-letter queue.

The message security policy utility

The message security policy utility (CSQ0UTIL) runs as a stand-alone utility to manage message security policies. See The message security policy utility (CSQ0UTIL) for more information.

The CSQUTIL utility

This is a utility program provided to help you with backup, restore and reorganize tasks. See "Using the CSQUTIL utility for IBM MQ for z/OS" on page 473.

The data conversion exit utility

The IBM MQ for z/OS data conversion exit utility (**CSQUCVX**) runs as a stand-alone utility to create data conversion exit routines.

The change log inventory utility

The IBM MQ for z/OS change log inventory utility program (**CSQJU003**) runs as a stand-alone utility to change the bootstrap data set (BSDS). You can use the utility to perform the following functions:

- Add or delete active or archive log data sets.
- Supply passwords for archive logs.

The print log map utility

The IBM MQ for z/OS print log map utility program (**CSQJU004**) runs as a stand-alone utility to list the following information:

- Log data set name and log RBA association for both copies of all active and archive log data sets. If dual logging is not active, there is only one copy of the data sets.
- Active log data sets available for new log data.
- Contents of the queue of checkpoint records in the bootstrap data set (BSDS).
- Contents of the archive log command history record.
- System and utility time stamps.

The log print utility

The log print utility program (**CSQ1LOGP**) is run as a stand-alone utility. You can run the utility specifying:

- A bootstrap data set (BSDS)
- Active logs (with no BSDS)
- Archive logs (with no BSDS)

The queue sharing group utility

The queue sharing group utility program (**CSQ5PQSG**) runs as a stand-alone utility to set up Db2 tables and perform other Db2 tasks required for queue sharing groups.

The active log preformat utility

The active log preformat utility (**CSQJUFMT**) formats active log data sets before they are used by a queue manager. If the active log data sets are preformatted by the utility, log write performance is improved on the queue manager's first pass through the active logs.

The dead-letter queue handler utility

The dead-letter queue handler utility program (**CSQUDLQH**) runs as a stand-alone utility. It checks messages that are on the dead-letter queue and processes them according to a set of rules that you supply to the utility.

The queue load and unload utility

The queue load and unload utility copies or moves the contents of a queue, or its messages, to a file. The utility was originally shipped as the **QLOAD** utility in IBM MQ Supportpac MO03. From IBM MQ 8.0 it is integrated into the product as executable module **CSQUDMSG** in the SCSQLOAD library, with an alias of QLOAD for compatibility. Sample JCL is provided as member CSQ4QLOD in SCSQPROC.

The equivalent utility for Multiplatforms is called **dmpmqmsg**. For details of the available options, including the differences for z/OS, see dmpmqmsg (queue load and unload).

You can also reload messages as described in Restoring messages from a data set to a queue (LOAD) on z/OS and Restoring messages from a data set to a queue (SLOAD) on z/OS.

z/OS Using the CSQUTIL utility for IBM MQ for z/OS

The CSQUTIL utility program is provided with IBM MQ for z/OS to help you perform backup, restoration, and reorganization tasks, and to issue commands and process object definitions.

About this task

Use this utility program to invoke the following functions. For example, you can issue commands from a sequential data set using the COMMAND function of the CSQUTIL utility. This function transfers the commands, as messages, to the *system-command input queue* and waits for the response, which is printed together with the original commands in SYSPRINT.

For more information about the CSQUTIL utility program, see IBM MQ utility program (CSQUTIL).

Procedure

- COMMAND

  Use this function to issue MQSC commands, to record object definitions, and to make client-channel definition files.
- COPY

  Use this function to read the contents of a named IBM MQ for z/OS message queue or the contents of all the queues of a named page set, and put them into a sequential file and retain the original queue.
- COPYPAGE

  Use this function to copy whole page sets to larger page sets.
- EMPTY

  Use this function to delete the contents of a named IBM MQ for z/OS message queue or the contents of all the queues of a named page set, retaining the definitions of the queues.
- FORMAT

Use this function to format IBM MQ for z/OS page sets.

- Restoring messages from a data set to a queue (LOAD) on z/OS

  Use this function to restore the contents of a named IBM MQ for z/OS message queue or the contents of all the queues of a named page set from a sequential file created by the COPY function.
- PAGEINFO

  Use this function to extract page set information from one or more page sets.
- RESETPAGE

  Use this function to copy whole page sets to other page set data sets and reset the log information in the copy.
- SCOPY

  Use this function to copy the contents of a queue to a data set while the queue manager is offline.
- SDEFS

  Use this function to produce a set of define commands for objects while the queue manager is offline.
- SLOAD

  Use this function to restore messages from the destination data set of an earlier COPY or SCOPY operation. SLOAD processes a single queue.
- SWITCH

  Use this function to switch or query the transmission queue associated with cluster-sender channels.

z/OS Using the Command Facility on z/OS

Use the editor to enter or amend MQSC commands to be passed to the queue manager.

From the primary panel, CSQOPRIA, select option **8 Command**, to start the Command Facility.

You are presented with an edit session of a sequential file, *prefix*.CSQUTIL.COMMANDS, used as input to the CSQUTIL COMMAND function; see Issuing commands to IBM MQ.

You do not need to prefix commands with the command prefix string (CPF).

You can continue MQSC commands on subsequent lines by terminating the current line with the continuation characters **+** or **-**. Alternatively, use line edit mode to provide long MQSC commands or the values of long attribute values within the command.

line edit

To use line edit, move the cursor to the appropriate line in the edit panel and use **F4** to display a single line in a scrollable panel. A single line can be up to 32 760 bytes of data.

To leave line edit:

- **F3 exit** saves changes made to the line and exits
- **F12 cancel** returns to the edit panel discarding changes made to the line.

To discard changes made in the edit session, use **F12 cancel** to terminate the edit session leaving the contents of the file unchanged. Commands are not executed.

Executing commands

When you have finished entering MQSC commands, terminate the edit session with **F3 exit** to save the contents of the file and invoke CSQUTIL to pass the commands to the queue manager. The output from command processing is held in file *prefix*.CSQUTIL.OUTPUT. An edit session opens automatically on this file so that you can view the responses. Press **F3 exit** to exit this session and return to the main menu.

z/OS Working with IBM MQ objects on z/OS

Many of the tasks described in this documentation involve manipulating IBM MQ objects. The object types are queue managers, queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects.

- Defining simple queue objects
- Defining other types of objects
- Working with object definitions
- Working with namelists

Defining simple queue objects

To define a new object, use an existing definition as the basis for it. You can do this in one of three ways:

- By selecting an object that is a member of a list displayed as a result of options selected on the initial panel. You then enter action type 2 (**Define like**) in the action field next to the selected object. Your new object has the attributes of the selected object, except the disposition. You can then change any attributes in your new object as you require.
- On the initial panel, select the **Define like** action type, enter the type of object that you are defining in the **Object type** field, and enter the name of a specific existing object in the **Name** field. Your new object has the same attributes as the object you named in the **Name** field, except the disposition. You can then change any attributes in your new object definition as you require.
- By selecting the **Define like** action type, specifying an object type and then leaving the **Name** field blank. You can then define your new object and it has the default attributes defined for your installation. You can then change any attributes in your new object definition as you require.

Note: You do not enter the name of the object you are defining on the initial panel, but on the **Define** panel you are presented with.

The following example demonstrates how to define a local queue using an existing queue as a template.

Defining a local queue

To define a local queue object from the operations and control panels, use an existing queue definition as the basis for your new definition. There are several panels to complete. When you have completed all the panels and you are satisfied that the attributes are correct, press Enter to send your definition to the queue manager, which then creates the actual queue.

Use the **Define like** action either on the initial panel or against an object entry in a list displayed as a result of options selected on the initial panel.

For example, starting from the initial panel, complete these fields:

Action	2 (Define like)
Object type	QLOCAL
Name	QUEUE.YOU.LIKE. This is the name of the queue that provides the attributes for your new queue.

Press Enter to display the **Define a Local Queue** panel. The queue name field is blank so that you can supply the name for the new queue. The description is that of the queue upon which you are basing this new definition. Over type this field with your own description for the new queue.

The values in the other fields are those of the queue upon which you are basing this new queue, except the disposition. You can over type these fields as you require. For example, type Y in the **Put enabled** field (if it is not already Y) if suitably authorized applications can put messages on this queue.

You get field help by moving the cursor into a field and pressing function key F1. Field help provides information about the values that can be used for each attribute.

When you have completed the first panel, press function key F8 to display the second panel.

Hints:

1. Do not press Enter at this stage, otherwise the queue will be created before you have a chance to complete the remaining fields. (If you do press Enter prematurely, do not worry; you can always alter your definition later on.)
2. Do not press function keys F3 or F12, or the data you typed will be lost.

Press function key F8 repeatedly to see and complete the remaining panels, including the trigger definition, event control, and backout reporting panels.

When your local queue definition is complete

When your definition is complete, press Enter to send the information to the queue manager for processing. The queue manager creates the queue according to the definition you have supplied. If you do not want the queue to be created, press function key F3 to exit and cancel the definition.

Defining other types of objects

To define other types of object, use an existing definition as the base for your new definition as explained in Defining a local queue.

Use the **Define like** action either on the initial panel or against an object entry in a list displayed as a result of options selected on the initial panel.

For example, starting from the initial panel, complete these fields:

Action	2 (Define like)
Object type	QALIAS, NAMELIST, PROCESS, CHANNEL, and other resource objects.
Name	Leave blank or enter the name of an existing object of the same type.

Press Enter to display the corresponding DEFINE panels. Complete the fields as required and then press Enter again to send the information to the queue manager.

Like defining a local queue, defining another type of object generally requires several panels to be completed. Defining a namelist requires some additional work, as described in "Working with namelists" on page 477.

Working with object definitions

When an object has been defined, you can specify an action in the **Action** field, to alter, display, or manage it.

In each case, you can either:

- Select the object you want to work with from a list displayed as a result of options selected on the initial panel. For example, having entered 1 in the **Action** field to display objects, Queue in the **Object type** field, and * in the **Name** field, you are presented with a list of all queues defined in the system. You can then select from this list the queue with which you need to work.
- Start from the initial panel, where you specify the object you are working with by completing the **Object type** and **Name** fields.

Altering an object definition

To alter an object definition, specify action 3 and press Enter to see the ALTER panels. These panels are very similar to the DEFINE panels. You can alter the values you want. When your changes are complete, press Enter to send the information to the queue manager.

Displaying an object definition
If you want to see the details of an object without being able to change them, specify action 1 and press Enter to see the DISPLAY panels. Again, these panels are similar to the DEFINE panels except that you cannot change any of the fields. Change the object name to display details of another object.

Deleting an object
To delete an object, specify action 4 (Manage) and the **Delete** action is one of the actions presented on the resulting menu. Select the **Delete** action.

You are asked to confirm your request. If you press function key F3 or F12, the request is canceled. If you press Enter, the request is confirmed and passed to the queue manager. The object you specified is then deleted.

Note: You cannot delete most types of channel object unless the channel initiator is started.

Working with namelists

When working with namelists, proceed as you would for other objects.

For the actions DEFINE LIKE or ALTER, press function key F11 to add names to the list or to change the names in the list. This involves working with the ISPF editor and all the normal ISPF edit commands are available. Enter each name in the namelist on a separate line.

When you use the ISPF editor in this way, the function key settings are the normal ISPF settings, and **not** those used by the other operations and control panels.

If you need to specify lowercase names in the list, specify CAPS(OFF) on the editor panel command line. When you do this, all the namelists that you edit in the future are in lowercase until you specify CAPS(ON).

When you have finished editing the namelist, press function key F3 to end the ISPF edit session. Then press Enter to send the changes to the queue manager.

Attention: If you do not press Enter at this stage but press function key F3 instead, you lose any updates that you have typed in.

z/OS Implementing the system using multiple cluster transmission queues

It makes no difference if the channel is used in a single cluster, or an overlapping cluster. When the channel is selected and started, the channel selects the transmission queue depending on the definitions.

Procedure

- If you are using the DEFCLXQ option, see “Using the automatic definition of queues and switching” on page 477.
- If you are using a staged approach, see “Changing your cluster-sender channels using a phased approach” on page 478.

z/OS Using the automatic definition of queues and switching

Use this option if you are planning on using the DEFCLXQ option. There will be a queue created for every channel, and every new channel.

Procedure

1. Review the definition of the SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE and change the attributes if required.

   This queue is defined in member SCSQPROC(`csq4insx`).
2. Create the SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE model queue.
3. Apply security policies for this model queue, and the SYSTEM.CLUSTER.TRANSMIT.** queues.

For z/OS the channel initiator started task user ID needs:

- Control access to CLASS(MQADMIN) for

```
ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channelname
```

- Update access to CLASS(MQQUEUE) for

```
ssid.SYSTEM.CLUSTER.TRANSMIT.channelname
```

z/OS Changing your cluster-sender channels using a phased approach

Use this option if you are planning on using a staged approach. This process allows you to move to the new cluster-sender channels at various times to suit the needs of your enterprise.

Before you begin

- Identify your business applications, and which channels are used.
- For the queues you use, display the clusters they are in.
- Display the channels to show the connection names, the names of the remote queue managers, and which clusters the channel supports.

About this task

- Create a transmission queue. On z/OS you might want to consider which page set you use for the queue.
- Set up security policy for the queue.
- Change any queue monitoring to include this queue name.
- Decide which channels are to use this transmission queue. The channels should have a similar name, so generic characters ' * ' in the CLCHNAME identify the channel.
- When you are ready to use the new function, alter the transmission queue to specify the name of the channels to use this transmission queue. For example CLUSTER1.TOPARIS, or CLUSTER1.* or *.TOPARIS
- Start the channels

Procedure

1. Use the `DIS CLUSQMGR(xxxx) XMITQ` command to display the cluster sender channels defined in the cluster, where *xxxx* is the name of the remote queue manager.
2. Set up the security profile for the transmission queue and give the queue access to the channel initiator.
3. Define the transmission queue to be used, and specify USAGE(XMITQ) INDXTYPE(CORRELID) SHARE and CLCHNAME(*value*)

   The channel initiator started task user ID needs the following access:

   ```
   alter class(MQADMIN) ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channel
   update class(MQQUEUE ssid.SYSTEM.CLUSTER.TRANSMIT.channel
   ```

   and the user ID using the SWITCH command needs the following access:

   ```
   alter cl(MQADMIN) ssid.QUEUE.queuename
   ```

4. Stop and restart the channels.

   The channel change occurs when the channel starts using an MQSC command, or you use CSQUTIL. You can identify which channels need to be restarted using the `SWITCH CHANNEL(*)` STATUS of CSQUTIL

If you have problems when the channel is started, stop the channel, resolve the problems, and restart the channel.

Note that you can change the CLCHNAME attribute as often as you need to.

The value of CLCHNAME used is the one when the channel is started, so you can change the CLCHNAME definition while the channel continues to use the definitions from the time that it started. The channel uses the new definition when it is restarted.

z/OS Undoing a change to a transmission queue on z/OS

You need to have a process to backout a change if it the results are not as you expect.

What can go wrong?

If the new transmission queue is not what you expect:

1. Check the CLCHNAME is as you expect
2. Review the job log to check if the switch process has finished. If not, wait and check the new transmission queue of the channel later.

If you are using multiple cluster transmission queues, it is important that you design the transmission queues definitions explicitly and avoid complicated overlapping configuration. In this way, you can make sure that if there are problems, you can go back to the original queues and configuration.

If you encounter problems during the move to using a different transmission queue, you must resolve any problems before you can proceed with the change.

An existing change request must complete before a new change request can be made. For example, you:

1. Define a new transmission queue with a maximum depth of one and there are 10 messages waiting to be sent.
2. Change the transmission queue to specify the channel name in the CLCHNAME parameter.
3. Stop and restart the channel. The attempt to move the messages fails and reports the problems.
4. Change the CLCHNAME parameter on the transmission queue to be blank.
5. Stop and restart the channel. The channel continues to try and complete the original request, so the channel continues to use the new transmission queue.
6. Need to resolve the problems and restart the channel so the moving of messages completes successfully.

Next time the channel is restarted it picks up any changes, so if you had set CLCHNAME to blanks, the channel will not use the specified transmission queue.

In this example, changing the CLCHNAME on the transmission queue to blanks does not necessarily mean that the channel uses the SYSTEM.CLUSTER.TRANSMIT queue, as there might be other transmission queues whose CLCHNAME parameter match the channel name. For example, a generic name, or the queue manager attribute DEFCLXQ might be set to channel, so the channel uses a dynamic queue instead of the SYSTEM.CLUSTER.TRANSMIT queue.

z/OS Writing programs to administer IBM MQ for z/OS

You can write your own application programs to administer a queue manager. Use this topic to understand the requirements for writing your own administration programs.

Start of General-use programming interface information

This set of topics contains hints and guidance to enable you to issue IBM MQ commands from an IBM MQ application program.

Note: In this topic, the MQI calls are described using C-language notation. For typical invocations of the calls in the COBOL, PL/I, and assembler languages, see Function calls.

Understanding how it all works

In outline, the procedure for issuing commands from an application program is as follows:

1. Build an IBM MQ command into a type of IBM MQ message called a *request message*. The command can be in MQSC or PCF format.
2. Send (use MQPUT) this message to a special queue called the system-command input queue. The IBM MQ command processor runs the command.
3. Retrieve (use MQGET) the results of the command as *reply messages* on the reply-to queue. These messages contain the user messages that you need to determine whether your command was successful and, if it was, what the results were.

Then it is up to your application program to process the results.

This set of topics contains:

z/OS Preparing queues for administration programs

Administration programs require a number of predefined queues for system command input and receiving responses.

This section applies to commands in the MQSC format. For the equivalent in PCF, see “IBM MQ プログラマブル・コマンド・フォーマットの使用” on page 26.

Before you can issue any MQPUT or MQGET calls, you must first define, and then open, the queues you are going to use.

Defining the system-command input queue

The system-command input queue is a local queue called SYSTEM.COMMAND.INPUT. The supplied CSQINP2 initialization data set, thlqual.SCSQPROC(CSQ4INSG), contains a default definition for the system-command input queue. For compatibility with IBM MQ on other platforms, an alias of this queue, called SYSTEM.ADMIN.COMMAND.QUEUE is also supplied. See Sample definitions supplied with IBM MQ for more information.

Defining a reply-to queue

You must define a reply-to queue to receive reply messages from the IBM MQ command processor. It can be any queue with attributes that allow reply messages to be put on it. However, for normal operation, specify these attributes:

- USAGE(NORMAL)
- NOTRIGGER (unless your application uses triggering)

Avoid using persistent messages for commands, but if you choose to do so, the reply-to queue must not be a temporary dynamic queue.

The supplied CSQINP2 initialization data set, thlqual.SCSQPROC(CSQ4INSG), contains a definition for a model queue called SYSTEM.COMMAND.REPLY.MODEL. You can use this model to create a dynamic reply-to queue.

Note: Replies generated by the command processor can be up to 15 000 bytes in length.

If you use a permanent dynamic queue as a reply-to queue, your application should allow time for all PUT and GET operations to complete before attempting to delete the queue, otherwise MQRC2055 (MQRC_Q_NOT_EMPTY) can be returned. If this occurs, try the queue deletion again after a few seconds.

Opening the system-command input queue

Before you can open the system-command input queue, your application program must be connected to your queue manager. Use the MQI call MQCONN or MQCONNX to do this.

Then use the MQI call MQOPEN to open the system-command input queue. To use this call:

1. Set the **Options** parameter to MQOO_OUTPUT
2. Set the MQOD object descriptor fields as follows:

   ObjectType
   MQOT_Q (the object is a queue)

   ObjectName
   SYSTEM.COMMAND.INPUT

   ObjectQMgrName
   If you want to send your request messages to your local queue manager, leave this field blank. This means that your commands are processed locally.

   If you want your IBM MQ commands to be processed on a remote queue manager, put its name here. You must also have the correct queues and links set up, as described in Distributed queuing and clusters.

Opening a reply-to queue

To retrieve the replies from an IBM MQ command, you must open a reply-to queue. One way of doing this is to specify the model queue, SYSTEM.COMMAND.REPLY.MODEL in an MQOPEN call, to create a permanent dynamic queue as the reply-to queue. To use this call:

1. Set the **Options** parameter to MQOO_INPUT_SHARED
2. Set the MQOD object descriptor fields as follows:

   ObjectType
   MQOT_Q (the object is a queue)

   ObjectName
   The name of the reply-to queue. If the queue name you specify is the name of a model queue object, the queue manager creates a dynamic queue.

   ObjectQMgrName
   To receive replies on your local queue manager, leave this field blank.

   DynamicQName
   Specify the name of the dynamic queue to be created.

z/OS Using the command server

The command server is an IBM MQ component that works with the command processor component. You can send formatted messages to the command server which interprets the messages, runs the administration requests, and sends responses back to your administration application.

The command server reads request messages from the system-command input queue, verifies them, and passes the valid ones as commands to the command processor. The command processor processes the commands and puts any replies as reply messages on to the reply-to queue that you specify. The first reply message contains the user message CSQN205I. See “Interpreting the reply messages from the command server” on page 485 for more information. The command server also processes channel initiator and queue sharing group commands, wherever they are issued from.

Identifying the queue manager that processes your commands

The queue manager that processes the commands you issue from an administration program is the queue manager that owns the system-command input queue that the message is put onto.

Starting the command server

Normally, the command server is started automatically when the queue manager is started. It becomes available as soon as the message CSQ9022I 'START QMGR' NORMAL COMPLETION is returned from the START QMGR command. The command server is stopped when all the connected tasks have been disconnected during the system termination phase.

You can control the command server yourself using the START CMDSERV and STOP CMDSERV commands. To prevent the command server starting automatically when IBM MQ is restarted, you can add a STOP CMDSERV command to your CSQINP1 or CSQINP2 initialization data sets. However, this is not recommended as it prevents any channel initiator or queue sharing group commands being processed.

The STOP CMDSERV command stops the command server as soon as it has finished processing the current message, or immediately if no messages are being processed.

If the command server has been stopped by a STOP CMDSERV command in the program, no other commands from the program can be processed. To restart the command server, you must issue a START CMDSERV command from the z/OS console.

If you stop and restart the command server while the queue manager is running, all the messages that are on the system-command input queue when the command server stops are processed when the command server is restarted. However, if you stop and restart the queue manager after the command server is stopped, only the persistent messages on the system-command input queue are processed when the command server is restarted. All nonpersistent messages on the system-command input queue are lost.

Sending commands to the command server

For each command, you build a message containing the command, then put it onto the system-command input queue.

Building a message that includes IBM MQ commands

You can incorporate IBM MQ commands in an application program by building request messages that include the required commands. For each such command you:

1. Create a buffer containing a character string representing the command.
2. Issue an MQPUT call specifying the buffer name in the **buffer** parameter of the call.

The simplest way to do this in C is to define a buffer using 'char'. For example:

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

When you build a command, use a null-terminated character string. Do not specify a command prefix string (CPF) at the start of a command defined in this way. This means that you do not have to alter your command scripts if you want to run them on another queue manager. However, you must take into account that a CPF is included in any response messages that are put onto the reply-to queue.

The command server folds all lowercase characters to uppercase unless they are inside quotation marks.

Commands can be any length up to a maximum 32 762 characters.

Putting messages on the system-command input queue

Use the MQPUT call to put request messages containing commands on the system-command input queue. In this call you specify the name of the reply-to queue that you have already opened.

To use the MQPUT call:

1. Set these MQPUT parameters:

Hconn
The connection handle returned by the MQCONN or MQCONNX call.

Hobj
The object handle returned by the MQOPEN call for the system-command input queue.

BufferLength
The length of the formatted command.

Buffer
The name of the buffer containing the command.

2. Set these MQMD fields:

MsgType
MQMT_REQUEST

Format
MQFMT_STRING or MQFMT_NONE

If you are not using the same code page as the queue manager, set *CodedCharSetId* as appropriate and set MQFMT_STRING, so that the command server can convert the message. Do not set MQFMT_ADMIN, as that causes your command to be interpreted as PCF.

ReplyToQ
Name of your reply-to queue.

ReplyToQMgr
If you want replies sent to your local queue manager, leave this field blank. If you want your IBM MQ commands to be sent to a remote queue manager, put its name here. You must also have the correct queues and links set up, as described in Distributed queuing and clusters.

3. Set any other MQMD fields, as required. You should normally use nonpersistent messages for commands.
4. Set any *PutMsgOpts* options, as required.

   If you specify MQPMO_SYNCPOINT (the default), you must follow the MQPUT call with a syncpoint call.

Using MQPUT1 and the system-command input queue

If you want to put just one message on the system-command input queue, you can use the **MQPUT1** call. This call combines the functions of an **MQOPEN**, followed by an **MQPUT** of one message, followed by an **MQCLOSE**, all in one call. If you use this call, modify the parameters accordingly. See Putting one message on a queue using the MQPUT1 call for details.

z/OS Retrieving replies to your commands

The command server sends a response to a reply queue for each request message it receives. Any administration application must receive, and handle the reply messages.

When the command processor processes your commands, any reply messages are put onto the reply-to queue specified in the MQPUT call. The command server sends the reply messages with the same persistence as the command message it received.

Waiting for a reply

Use the MQGET call to retrieve a reply from your request message. One request message can produce several reply messages. For details, see "Interpreting the reply messages from the command server" on page 485.

You can specify a time interval that an MQGET call waits for a reply message to be generated. If you do not get a reply, use the checklist beginning in topic "If you do not receive a reply" on page 486.

To use the MQGET call:

1. Set these parameters:

Hconn
The connection handle returned by the MQCONN or MQCONNX call.

Hobj
The object handle returned by the MQOPEN call for the reply-to queue.

Buffer
The name of the area to receive the reply.

BufferLength
The length of the buffer to receive the reply. This must be a minimum of 80 bytes.

2. To ensure that you only get the responses from the command that you issued, you must specify the appropriate *MsgId* and *CorrelId* fields. These depend on the report options, MQMD_REPORT, you specified in the MQPUT call:

MQRO_NONE
Binary zero, '00...00' (24 nulls).

MQRO_NEW_MSG_ID
Binary zero, '00...00' (24 nulls).

This is the default if none of these options has been specified.

MQRO_PASS_MSG_ID
The *MsgId* from the **MQPUT**.

MQRO_NONE
The *MsgId* from the MQPUT call.

MQRO_COPY_MSG_ID_TO_CORREL_ID
The *MsgId* from the MQPUT call.

This is the default if none of these options has been specified.

MQRO_PASS_CORREL_ID
The *CorrelId* from the MQPUT call.

For more details on report options, see Report options and message flags.

3. Set the following *GetMsgOpts* fields:

Options
MQGMO_WAIT

If you are not using the same code page as the queue manager, set MQGMO_CONVERT, and set *CodedCharSetId* as appropriate in the MQMD.

WaitInterval
For replies from the local queue manager, try 5 seconds. Coded in milliseconds, this becomes 5 000. For replies from a remote queue manager, and channel control and status commands, try 30 seconds. Coded in milliseconds, this becomes 30 000.

Discarded messages

If the command server finds that a request message is not valid, it discards this message and writes the message CSQN205I to the named reply-to queue. If there is no reply-to queue, the CSQN205I message is put onto the dead-letter queue. The return code in this message shows why the original request message was not valid:

00D5020F It is not of type MQMT_REQUEST.

00D50210 It has zero length.

00D50212 It is longer than 32 762 bytes.

00D50211 It contains all blanks.

00D5483E It needed converting, but *Format* was not MQFMT_STRING.

Other See Command server codes

The command server reply message descriptor

For any reply message, the following MQMD message descriptor fields are set:

MsgType	MQMT_REPLY
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
Priority	As for the MQMD in the message you issued.
Persistence	As for the MQMD in the message you issued.
CorrelId	Depends on the MQPUT report options.
ReplyToQ	None.

The command server sets the *Options* field of the MQPMO structure to MQPMO_NO_SYNCPOINT. This means that you can retrieve the replies as they are created, rather than as a group at the next syncpoint.

z/OS Interpreting the reply messages from the command server

Each request message correctly processed by IBM MQ produces at least two reply messages. Each reply message contains a single IBM MQ user message.

The length of a reply depends on the command that was issued. The longest reply you can get is from a **DISPLAY NAMELIST** command, and that can be up to 15 000 bytes in length.

The first user message, CSQN205I, always contains:

- A count of the replies (in decimal), which you can use as a counter in a loop to get the rest of the replies. The count includes this first message.
- The return code from the command preprocessor.
- A reason code, which is the reason code from the command processor.

This message does not contain a CPF.

For example:

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
```

The COUNT field is 8 bytes long and is right-justified. It always starts at position 18, that is, immediately after COUNT=. The RETURN field is 8 bytes long in character hexadecimal and is immediately after RETURN= at position 35. The REASON field is 8 bytes long in character hexadecimal and is immediately after REASON= at position 52.

If the RETURN= value is 00000000 and the REASON= value is 00000004, the set of reply messages is incomplete. After retrieving the replies indicated by the CSQN205I message, issue a further MQGET call to wait for a further set of replies. The first message in the next set of replies is again CSQN205I, indicating how many replies there are, and whether there are still more to come.

See the IBM MQ for z/OS のメッセージ、完了コード、および理由コード documentation for more details about the individual messages.

If you are using a non-English language feature, the text and layout of the replies are different from those shown here. However, the size and position of the count and return codes in message CSQN205I are the same.

z/OS If you do not receive a reply

There are a series of steps you can take if you do not receive a response to request to the command server.

If you do not receive a reply to your request message, work through this checklist:

- Is the command server running?
- Is the *WaitInterval* long enough?
- Are the system-command input and reply-to queues correctly defined?
- Were the MQOPEN calls to these queues successful?
- Are both the system-command input and reply-to queues enabled for MQPUT and MQGET calls?
- Have you considered increasing the MAXDEPTH and MAXMSGL attributes of your queues?
- Are you are using the *CorrelId* and *MsgId* fields correctly?
- Is the queue manager still running?
- Was the command built correctly?
- Are all your remote links defined and operating correctly?
- Were the MQPUT calls correctly defined?
- Has the reply-to queue been defined as a temporary dynamic queue instead of a permanent dynamic queue? (If the request message is persistent, you must use a permanent dynamic queue for the reply.)

When the command server generates replies but cannot write them to the reply-to queue that you specify, it writes them to the dead-letter queue.

z/OS Passing commands using MGCRE

With appropriate authorization, an application program can make requests to multiple queue managers using a z/OS service routine.

If you have the correct authorization, you can pass IBM MQ commands from your program to multiple queue managers by the MGCRE (SVC 34) z/OS service. See the *z/OS MVS Programming: Authorized Assembler Services Guide* for more information.

The value of the CPF identifies the particular queue manager to which the command is directed. For information about CPFs, see User IDs for command security and command resource security and “Issuing queue manager commands on z/OS” on page 449.

If you use MGCRE, you can use a Command and Response Token (CART) to get the direct responses to the command.

z/OS Examples of commands and their replies

Use this topic as a series of examples of commands to the command server and the responses from the command server.

Here are some examples of commands that could be built into IBM MQ messages, and the user messages that are the replies. Unless otherwise stated, each line of the reply is a separate message.

- Messages from a DEFINE command
- Messages from a DELETE command
- Messages from DISPLAY commands
- Messages from commands with CMDSCOPE
- Messages from commands that generate commands with CMDSCOPE

Messages from a DEFINE command

The following command:

```
DEFINE QLOCAL(Q1)
```

produces these messages:

```
CSQN205I    COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

These reply messages are produced on normal completion.

Messages from a DELETE command

The following command:

```
DELETE QLOCAL(Q2)
```

produces these messages:

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
CSQM125I +CSQ1 CSQMUQLC QLOCAL (Q2) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E +CSQ1 CSQMUQLC FAILURE REASON CODE X'00D44002'
CSQ9023E +CSQ1 CSQMUQLC ' DELETE QLOCAL' ABNORMAL COMPLETION
```

These messages indicate that a local queue called Q2 does not exist.

Messages from DISPLAY commands

The following examples show the replies from some DISPLAY commands.

Finding out the name of the dead-letter queue

If you want to find out the name of the dead-letter queue for a queue manager, issue this command from an application program:

```
DISPLAY QMGR DEADQ
```

The following three user messages are returned, from which you can extract the required name:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE          )
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

Messages from the DISPLAY QUEUE command

The following examples show how the results from a command depend on the attributes specified in that command.

Example 1

You define a local queue using the command:

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

If you issue the following command from an application program:

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

these three user messages are returned:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(Q1                      ) TYPE(
QLOCAL ) QSGDISP(QMGR   )
DESCR(A sample queue
) SHARE  GET(ENABLED  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

Note: The second message, CSQM401I, is shown here occupying four lines.

Example 2

Two queues have names beginning with the letter A:

- A1 is a local queue with its PUT attribute set to DISABLED.
- A2 is a remote queue with its PUT attribute set to ENABLED.

If you issue the following command from an application program:

```
DISPLAY QUEUE(A*) PUT
```

these four user messages are returned:

```
CSQN205I    COUNT=    4, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(A1                      ) TYPE(
QLOCAL ) QSGDISP(QMGR   )
PUT(DISABLED )
CSQM406I +CSQ1 QUEUE(A2                      ) TYPE(
QREMOTE ) PUT(ENABLED  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

Note: The second and third messages, CSQM401I and CSQM406I, are shown here occupying three and two lines.

Messages from the DISPLAY NAMELIST command

You define a namelist using the command:

```
DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)
```

If you issue the following command from an application program:

```
DISPLAY NAMELIST(N1) NAMES NAMCOUNT
```

the following three user messages are returned:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM407I +CSQ1 NAMELIST(N1                        ) QS
GDISP(QMGR   ) NAMCOUNT(     2) NAMES(Q1
,SAMPLE_QUEUE                  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY NAMELIST' NORMAL COMPLETION
```

Note: The second message, CSQM407I, is shown here occupying three lines.

Messages from commands with CMDSCOPE

The following examples show the replies from commands that have been entered with the CMDSCOPE attribute.

Messages from the ALTER PROCESS command

The following command:

```
ALT PRO(V4) CMDSCOPE(*)
```

produces the following messages:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'ALT PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ26
CSQM125I !MQ26 CSQMMSGP PROCESS(V4) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E !MQ26 CSQMMSGP FAILURE REASON CODE X'00D44002'
CSQ9023E !MQ26 CSQMMSGP ' ALT PRO' ABNORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' ALT PRO' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=0000000C, REASON=00000008
CSQN123E !MQ25 'ALT PRO' command for CMDSCOPE(*) abnormal completion
```

These messages tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). The command was successful on MQ25 but the process definition did not exist on MQ26, so the command failed on that queue manager.

Messages from the DISPLAY PROCESS command

The following command:

```
DIS PRO(V*) CMDSCOPE(*)
```

produces the following messages:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ26
CSQM408I !MQ26 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ26 PROCESS(V3) QSGDISP(QMGR)
CSQ9022I !MQ26 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    7, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ25
CSQM408I !MQ25 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ25 PROCESS(V2) QSGDISP(GROUP)
CSQM408I !MQ25 PROCESS(V3) QSGDISP(QMGR)
CSQM408I !MQ25 PROCESS(V4) QSGDISP(QMGR)
CSQ9022I !MQ25 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS PRO' command for CMDSCOPE(*) normal completion
```

These messages tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Information is displayed about all the processes on each queue manager with names starting with the letter V.

Messages from the DISPLAY CHSTATUS command

The following command:

```
DIS CHS(VT) CMDSCOPE(*)
```

produces the following messages:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS CHS' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ25
CSQM422I !MQ25 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ25 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=    4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ26
CSQM422I !MQ26 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ26 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS CHS' command for CMDSCOPE(*) normal completion
```

These messages tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Information is displayed about channel status on each queue manager.

Messages from the STOP CHANNEL command

The following command:

```
STOP CHL(VT) CMDSCOPE(*)
```

produces these messages:

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'STOP CHL' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQM134I !MQ25 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
SQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQM134I !MQ26 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQ9022I !MQ26 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQ9022I !MQ25 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'STOP CHL' command for CMDSCOPE(*) normal completion
```

These messages tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Channel VT was stopped on each queue manager.

Messages from commands that generate commands with CMDSCOPE

The following command:

```
DEF PRO(V2) QSGDISP(GROUP)
```

produces these messages:

```
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQM122I !MQ25 CSQMMSGP ' DEF PRO' COMPLETED FOR QSGDISP(GROUP)
CSQN138I !MQ25 'DEFINE PRO' command generated for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ26
CSQ9022I !MQ26 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DEFINE PRO' command for CMDSCOPE(*) normal completion
```

These messages tell you that the command was entered on queue manager MQ25. When the object was created on the shared repository, another command was generated and sent to all the active queue managers in the queue sharing group (MQ25 and MQ26).

z/OS Managing IBM MQ resources on z/OS

Use the links in this topic to find out how to manage the resources used by IBM MQ for z/OS, for example, managing log files, data sets, page sets, buffer pools, and coupling facility structures.

Use the following links for details of the different administrative tasks you might have to complete while using IBM MQ for z/OS:

- “Managing the logs” on page 492
- “Managing the bootstrap data set (BSDS)” on page 501
- “Managing page sets” on page 508
- “How to back up and recover page sets” on page 515
- “How to back up and restore queues using CSQUTIL” on page 518
- “Managing buffer pools” on page 518

- “Managing queue sharing groups and shared queues on z/OS” on page 520

Related concepts

IBM MQ for z/OS concepts

“Administering IBM MQ for z/OS” on page 449
IBM MQ for z/OS can be controlled and managed by MQSC and PCF commands, by a set of utilities and programs provided with the product, and by authorized applications.

“Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS” on page 449
You can issue MQSC and PCF commands from the IBM MQ for z/OS console, the initialization input data sets, the batch utility CSQUTIL, or from authorized applications. Not all commands can be issued from all these sources.

“Recovery and restart on z/OS” on page 529
Use this topic to understand the recovery and restart mechanisms used by IBM MQ.

Related tasks

Planning your IBM MQ environment on z/OS

Configuring queue managers on z/OS

IBM MQ utilities on z/OS reference

Related reference

“Using the IBM MQ for z/OS utilities” on page 471
IBM MQ for z/OS provides a set of utility programs that you can use to help with system administration.

Programmable command formats reference

z/OS Managing the logs

Use this topic to understand how to manage your IBM MQ log files, including the log archiving process, using log record compression, log record recovery, and printing log records.

This topic describes the tasks involved in managing the IBM MQ logs. It contains these sections:

z/OS *Archiving logs with the ARCHIVE LOG command*

An authorized operator can archive the current IBM MQ active log data sets whenever required using the **ARCHIVE LOG** command.

When you issue the ARCHIVE LOG command, IBM MQ truncates the current active log data sets, then runs an asynchronous offload process, and updates the BSDS with a record of the offload process.

The **ARCHIVE LOG** command has a **MODE(QUIESCE)** option. With this option, IBM MQ jobs and users are quiesced after a commit point, and the resulting point of consistency is captured in the current active log before it is offloaded.

Consider using the **MODE(QUIESCE)** option when planning a backup strategy for off site recovery. It creates a system-wide point of consistency, which minimizes the number of data inconsistencies when the archive log is used with the most current backup page set copy during recovery. For example:

```
ARCHIVE LOG MODE(QUIESCE)
```

If you issue the **ARCHIVE LOG** command without specifying a **TIME** parameter, the quiesce time period defaults to the value of the **QUIESCE** parameter of the CSQ6ARVP macro. If the time required for the `ARCHIVE LOG MODE(QUIESCE)` to complete is less than the time specified, the command completes successfully; otherwise, the command fails when the time period expires. You can specify the time period explicitly by using the **TIME** option, for example:

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

This command specifies a quiesce period of up to 60 seconds before **ARCHIVE LOG** processing occurs.

Attention: Using the **TIME** option when time is critical can significantly disrupt IBM MQ availability for all jobs and users that use IBM MQ resources.

By default, the command is processed asynchronously from the time you submit the command. (To process the command synchronously with other IBM MQ commands use the WAIT(YES) option with **QUIESCE**, but be aware that the z/OS console is locked from IBM MQ command input for the entire **QUIESCE** period.)

During the quiesce period:

- Jobs and users on the queue manager are allowed to go through commit processing, but are suspended if they try to update any IBM MQ resource after the commit.
- Jobs and users that only read data can be affected, since they might be waiting for locks held by jobs or users that were suspended.
- New tasks can start, but they cannot update data.

The output from the **DISPLAY LOG** command uses the message CSQV400I to indicate that a quiesce is in effect.

For example:

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     400
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
MAXCNOFF    0
WRTHRSH     20
DEALLCT     0
COMPLOG     NONE
ZHYWRITE    NO
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full zHyperWrite Encrypted DSName
 1     68  NO          NO        VICY.CSQ1.LOGCOPY1.DS01
 2     68  NO          NO        VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2019-08-15 09:49:30 using RBA=000000000891B000
Latest RBA=000000000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
```

When all updates are quiesced, the quiesce history record in the BSDS is updated with the date and time that the active log data sets were truncated, and with the last-written RBA in the current active log data sets. IBM MQ truncates the current active log data sets, switches to the next available active log data sets, and issues message CSQJ311I stating that the offload process started.

If updates cannot be quiesced before the quiesce period expires, IBM MQ issues message CSQJ317I, and **ARCHIVE LOG** processing terminates. The current active log data sets are not truncated, nor switched to the next available log data sets, and the offload process is not started.

Whether the quiesce was successful or not, all suspended users and jobs are then resumed, and IBM MQ issues message CSQJ312I, stating that the quiesce is ended and update activity is resumed.

If **ARCHIVE LOG** is issued when the current active log is the last available active log data set, the command is not processed, and IBM MQ issues the following message:

```
CSQJ319I - csect-name CURRENT ACTIVE LOG DATA SET IS THE LAST
AVAILABLE ACTIVE LOG DATA SET. ARCHIVE LOG PROCESSING
WILL BE TERMINATED
```

If **ARCHIVE LOG** is issued when another **ARCHIVE LOG** command is already in progress, the new command is not processed, and IBM MQ issues the following message:

```
CSQJ318I - ARCHIVE LOG COMMAND ALREADY IN PROGRESS
```

For information about the messages issued during archiving, see Messages for IBM MQ for z/OS.

Restarting the log archive process after a failure

If there is a problem during the log archive process (for example, a problem with allocation or tape mounts), the archiving of the active log might be suspended. You can cancel the archive process and restart it by using the following command:

```
ARCHIVE LOG CANCEL OFFLOAD
```

This command cancels any offload processing currently in progress, and restarts the archive process. It starts with the oldest log data set that has not been archived, and proceeds through all active log data sets that need offloading. Any log archive operations that have been suspended are restarted.

Use this command only if you are sure that the current log archive task is no longer functioning, or if you want to restart a previous attempt that failed. This is because the command might cause an abnormal termination of the offload task, which might result in a dump.

z/OS *Controlling archiving and logging*

You can control compression, printing, archiving, recovery and logging with using the CSQ6LOGP, CSQ6ARVP, and CSQ6SYSP macros. Note, that changes to private objects only are logged in IBM MQlogs. Changes to GROUP objects (like shared inbound channels) are also logged, because the definitions are propagated around the group and held locally.

Many aspects of archiving and logging are controlled by parameters set using the CSQ6LOGP, CSQ6ARVP and CSQ6SYSP macros of the system parameter module when the queue manager is customized. See Tailor your system parameter module for details of these macros.

Some of these parameters can be changed while a queue manager is running using the IBM MQ MQSC SET LOG, SET SYSTEM and SET ARCHIVE commands. They are shown in Table 28 on page 494:

Table 28. Archiving and logging parameters that can be changed while a queue manager is running

SET command	**Parameters**
LOG	WRTHRSH, MAXARCH, DEALLCT, MAXRTU, COMPLOG
ARCHIVE	All
SYSTEM	LOGLOAD

You can display the settings of all the parameters using the MQSC DISPLAY LOG, DISPLAY ARCHIVE and DISPLAY SYSTEM commands. These commands also show status information about archiving and logging.

Controlling log compression

You can enable and disable the compression of log records using either

- The SET and DISPLAY LOG commands in MQSC; see The MQSC commands
- Invoking PCF interface. See “IBM MQ プログラマブル・コマンド・フォーマットの概要” on page 25
- Using the CSQ6LOGP macro in the system parameter module; see Using CSQ6LOGP

Printing log records

You can extract and print log records using the CSQ1LOGP utility. For instructions, see The log print utility.

Recovering logs

Normally, you do not need to back up and restore the IBM MQ logs, especially if you are using dual logging. However, in rare circumstances, such as an I/O error on a log, you might need to recover the logs. Use Access Method Services to delete and redefine the data set, and then copy the corresponding dual log into it.

z/OS *Discarding archive log data sets*

You can discard your archive log data sets and choose to discard the logs automatically or manually.

You must keep enough log data to be able to perform unit of work recovery, page set media recovery if a page set is lost, or CF structure media recovery if a CF structure is lost. Do not discard archive log data sets that might be required for recovery; if you discard these archive log data sets you might not be able to perform required recovery operations.

If you have confirmed that your archive log data sets can be discarded, you can do this in either of the following ways:

- Automatic archive log data set deletion
- Manually deleting archive log data sets

Automatic archive log data set deletion

You can use a DASD or tape management system to delete archive log data sets automatically. The retention period for IBM MQ archive log data sets is specified by the retention period field ARCRETN in the CSQ6ARVP installation macro (see the Using CSQ6ARVP for more information).

The default for the retention period specifies that archive logs are to be kept for 9999 days (the maximum).

Important: You can change the retention period but you must ensure that you can accommodate the number of backup cycles that you have planned for.

.

IBM MQ uses the retention period value as the value for the JCL parameter RETPD when archive log data sets are created.

The retention period set by the MVS™/DFP storage management subsystem (SMS) can be overridden by this IBM MQ parameter. Typically, the retention period is set to the smaller value specified by either IBM MQ or SMS. The storage administrator and IBM MQ administrator must agree on a retention period value that is appropriate for IBM MQ.

Note: IBM MQ does not have an automated method to delete information about archive log data sets from the BSDS, because some tape management systems provide external manual overrides of retention periods. Therefore, information about an archive log data set can still be in the BSDS long after the data set retention period has expired and the data set has been scratched by the tape management system. Conversely, the maximum number of archive log data sets might have been exceeded and the data from the BSDS might have been dropped before the data set has reached its expiration date.

If archive log data sets are deleted automatically, remember that the operation does not update the list of archive logs in the BSDS. You can update the BSDS with the change log inventory utility, as described in “Changing the BSDS” on page 502. The update is not essential. Recording old archive logs wastes space in the BSDS, but does no other harm.

Manually deleting archive log data sets

You must keep all the log records as far back as the lowest RBA identified in messages CSQI024I and CSQI025I. This RBA is obtained using the DISPLAY USAGE command that you issued when creating a point of recovery using Method 1: Full backup.

Read Creating a point of recovery for non-shared resources before discarding any logs.

Locate and discard archive log data sets

Having established the minimum log RBA required for recovery, you can find archive log data sets that contain only earlier log records by performing the following procedure:

1. Use the print log map utility to print the contents of the BSDS. For an example of the output, see The print log map utility.
2. Find the sections of the output titled `ARCHIVE LOG COPY n DATA SETS`. If you use dual logging, there are two sections. The columns labeled STARTRBA and ENDRBA show the range of RBAs contained in each volume. Find the volumes with ranges that include the minimum RBA you found with messages CSQI024I and CSQI025I. These are the earliest volumes you need to keep. If you are using dual-logging, there are two such volumes.

   If no volumes have an appropriate range, one of the following cases applies:

   - The minimum RBA has not yet been archived, and you can discard all archive log volumes.
   - The list of archive log volumes in the BSDS wrapped around when the number of volumes exceeded the number allowed by the MAXARCH parameter of the CSQ6LOGP macro. If the BSDS does not register an archive log volume, that volume cannot be used for recovery. Therefore, consider adding information about existing volumes to the BSDS. For instructions, see “Changes for archive logs” on page 505.

     Also consider increasing the value of MAXARCH. For information, see the Using CSQ6LOGP.
3. Delete any archive log data set or volume with an ENDRBA value that is less than the STARTRBA value of the earliest volume you want to keep. If you are using dual logging, delete both such copies.

   Because BSDS entries wrap around, the first few entries in the BSDS archive log section might be more recent than the entries at the end. Look at the combination of date and time and compare their ages. Do not assume that you can discard all entries before the entry for the archive log containing the minimum LOGRBA.

   Delete the data sets. If the archives are on tape, erase the tapes. If they are on DASD, run a z/OS utility to delete each data set. Then, if you want the BSDS to list only existing archive volumes, use the change log inventory utility (CSQJU003) to delete entries for the discarded volumes. See “Changes for archive logs” on page 505 for an example.

z/OS *The effect of log shunting*

Long running transactions can cause unit of work log records which span log data sets. IBM MQ handles this scenario by using log shunting, a technique which moves the log records to optimize the quantity of log data retained, and queue manager restart time.

When a unit of work is considered to be long, a representation of each log record is written further down the log. This is known as *log shunting*. It is described more fully in Log files.

The queue manager uses these shunted log records instead of the originals after a failure, to ensure unit of work integrity. There are two benefits to this:

- the quantity of log data which must be retained for unit of work coordination is reduced
- less log data must be traversed at queue manager restart time, so the queue manager is restarted more quickly

Shunted log records do not contain sufficient information for media recovery operations.

Data held in the log is used for two distinct purposes; media recovery and unit of work coordination. If a media failure occurs which affects either a CF structure or page set, the queue manager can recover the media to the point of failure by restoring a prior copy and updating this using data contained in the log.

Persistent activity performed in a unit of work is recorded on the log so that in the event of a failure, it can either be backed out or locks can be recovered on changed resources. The quantity of log data you need to retain to enable queue manager recovery is affected by these two elements.

For media recovery, you must retain sufficient log data to be able to perform media recovery from at least the most recent media copy and to be able to back out. (Your site may stipulate the ability to recover from older backups.) For unit of work integrity, you must retain the log data for your oldest in flight or indoubt units of work.

To assist you with managing the system, the queue manager detects old units of work at each log archive and reports them in messages CSQJ160 and CSQJ161. An internal task reads unit of work log information for these old units of work and rewrites it in a more succinct form to the current position in the log. Message CSQR026 indicates when this has happened. The MQSC command DISPLAY USAGE TYPE(DATASET) can also help you to manage the retention of log data. The command reports the following three pieces of recovery information:

1. How much of the log must be retained for unit of work recovery.
2. How much of the log must be retained for media recovery of page sets.
3. For a queue manager in a queue sharing group, how much of the log must be retained for media recovery of CF structures.

For each of these pieces of information, an attempt is made to map the oldest log data required into a data set. As new units of work start and stop, (1) would be expected to move to a more recent position in the log. If it is not moving, the long running UOW messages warn you that there is an issue. (2) relates to page set media recovery if the queue manager were to be shut down now and restarted. It does not know about when you last backed up your page sets, or which backup you might have to use if there was a page set failure. It normally moves to a more recent position in the log during checkpoint processing as changes held in the buffer pools are written to the page sets. In (3), the queue manager does know about CF structure backups taken either on this queue manager or on other queue managers in the queue sharing group. However, CF structure recovery requires a merge of log data from all queue managers in the queue sharing group which have interacted with the CF structure since the last backup. This means that the log data is identified by a log record sequence number, (or LRSN), which is timestamp based and so applicable across the entire queue sharing group rather than an RBA which would be different on different queue managers in the queue sharing group. It normally moves to a more recent position in the log as BACKUP CFSTRUCT commands are performed on either this or other queue managers in the queue sharing group.

z/OS *Resetting the queue manager's log*

Use this topic to understand how to reset the queue manager's log.

You must not allow the queue manager log RBA to wrap around from the end of the log RBA range to 0, as this leads to a queue manager outage and all persistent data will become unrecoverable. The end of the log RBA is either a value of FFFFFFFFFFFF (if 6-byte RBAs as in use), or FFFFFFFFFFFFFFFF (if 8-byte RBAs are in use).

The queue manager issues messages CSQI045I, CSQI046E, CSQI047E, CSQJ031D, and CSQJ032E to indicate that the used log range is significant and that you should plan to take action to avoid an unplanned outage.

The queue manager terminates with reason code 00D10257 when the RBA value reaches FFF800000000 (if 6-byte log RBAs are in use) or FFFFFFC000000000 (if 8-byte log RBAs are in use).

If 6-byte log RBAs are in use, consider converting the queue manager to use 8-byte log RBAs rather than resetting the queue manager's log, following the process described in “Implementing the larger log Relative Byte Address” on page 500. Converting a queue manager to use 8-byte log RBAs requires a shorter outage than resetting the log, and increases the period of time before you have to reset the log.

Message CSQJ034I, issued during queue manager initialization, indicates the end of the log RBA range for the queue manager as configured, and can be used to determine whether 6-byte or 8-byte log RBAs are in use.

The procedure to follow to reset the queue manager's log is as follows:

1. Resolve any unresolved units of work. The number of unresolved units of work is displayed at queue manager startup in message CSQR005I as the INDOUBT count. At each checkpoint, and at queue manager shutdown, the queue manager automatically issues the command

   DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED) to provide information about unresolved units of work.

   See How in-doubt units of recovery are resolved for information on resolving units of recovery. The ultimate recourse is to use the **RESOLVE INDOUBT** MQSC command to manually resolve indoubt units of recovery.
2. Shut down the queue manager cleanly.

   You can use either **STOP QMGR** or **STOP QMGR MODE(FORCE)** as both these commands flush any changed pages from bufferpools to the page sets.
3. If a queue manager is part of a queue sharing group, take CFSTRUCT backups on other queue managers for all structures in the queue sharing group. This ensures that the most recent backups are not in this queue manager's log, and that this queue manager's log is not required for CFSTRUCT recovery.
4. Define new logs and BSDS using CSQJU003 (see The change log inventory utility for more information on using the change log inventory utility).
5. Run **CSQUTIL RESETPAGE** against all the page sets for this queue manager (see Copying a page and resetting the log for more information on using this function). Note that page set RBAs can be reset independently, so multiple concurrent jobs (for example, one per page set) can be submitted to reduce the elapsed time for this step.
6. Restart the queue manager

Warning messages

When IBM MQ detects that the end of the log is approaching, it issues console messages in the following order, which indicate that a log reset should be planned. In this section the messages show 6-byte log RBA values. The same console messages are issued when IBM MQ is running in 8-byte log RBA mode but with different values; see “Warning thresholds” on page 499 for the 8-byte log RBA thresholds.

1. When IBM MQ detects that the end of the log is approaching in the near future, (approximately 94% full) IBM MQ issues console message CSQI045I, as in the following example:

   ```
   CSQI045I -CSQ7 CSQILCUR Log RBA has reached 0000F00000000000.
   Plan a log reset
   ```
2. IBM MQ issues the following CSQI046E error console message when the end of the log is near (approximately 97% full). This informs the IBM MQ administrator to take action soon.

   ```
   CSQI046E -CSQ7 CSQILCUR Log RBA has reached 0000F80000000000.
   Perform a log reset
   ```
3. After the CSQI046E message is issued, at the next log switch, IBM MQ issues the following CSQJ032E console message with the word WARNING:

   ```
   CSQJ032E -CSQ7 CSQJW307 WARNING - APPROACHING END OF
   THE LOG RBA RANGE OF 0000FFFFFFFFFFFF. CURRENT LOG RBA IS 0000F80000022000.
   ```
4. After the CSQI046E and CSQJ032E console messages are issued, IBM MQ issues one more error message, which does not require immediate IBM MQ administrator intervention. IBM MQ issues console message CSQI047E (when the log is approximately 99% full):

   ```
   CSQI047E -CSQ7 CSQILCUR Log RBA has reached 0000FF0000000000.
   Stop queue manager and reset logs
   ```

5. When the log RBA reaches FF8000000000, IBM MQ increases the urgency of the situation and issues console message CSQJ032E with the word CRITICAL:

```
CSQJ032E -CSQ7 CSQJW009 CRITICAL - APPROACHING END OF THE LOG RBA RANGE OF 0000FFFFFFFFFFFF.
CURRENT LOG RBA IS 0000FFF7FFFFDFFF.
```

6. If the queue manager is started when the log RBA is almost at the maximum, the following CSQJ031D console message is issued. This stage requires the input of the IBM MQ administrator:.

```
CSQJ031D -CSQ7 CSQYSTRT THE LOG RBA RANGE MUST BE RESET.
REPLY 'Y' TO CONTINUE STARTUP OR 'N' TO SHUTDOWN
```

7. IBM MQ startup remains suspended until a reply is given to message CSQJ031D.

The purpose of these messages is to give the IBM MQ administrator time to plan for a system outage to reset the logs. In an ideal configuration, there are at least two queue managers, possibly in a queue sharing group (QSG), sharing the workload. When one is down for maintenance the other can continue to receive work.

The severity of console messages that IBM MQ issues becomes greater as the RBA gets closer to the end. Ideally your IBM MQ administrator should plan to reset the log RBA when the first console message is seen.

If the warning and error console messages are ignored, IBM MQ terminates with reason code 5C6-00D10257 when the log RBA reaches FFF800000000, at which point IBM MQ determines that the available range is too small for the queue manager to continue. When this point is reached, the only option is to take an outage and either reset the log or extend the size of the log RBA.

Note: When the end of the log is reached it is not possible to resolve any in-flight units of work (UOW); these are lost during the log reset process. Enough of the RBA range should be left to start the queue manager and resolve any UOW. Because IBM MQ issues console messages several times to inform that the end of the log is approaching, a log reset should be planned.

The preferred option to avoid losing any in-flight UOW is to extend the log RBA to use 8 bytes. This means that a log RBA reset will not be necessary for a long period.

Warning thresholds

The following table lists the thresholds, based on the length of the log RBA.

Console message	6-byte log RBA	8-byte log RBA
CSQI045I	0000F00000000000	FFFF800000000000
CSQI046E	0000F80000000000	FFFFC00000000000
CSQI047E	0000FF8000000000	FFFFFC0000000000
CSQJ032E	0000FF8000000000 0000FF8000000000	FFFFFC0000000000 FFFFFC0000000000
CSQJ031D	0000FF8000000000	FFFFFC0000000000

Notes:

1. For message CSQJ032E, the first number applies to the WARNING text and the second number applies to the CRITICAL text in the console message.
2. Message CSQJ031D is issued at IBM MQ initialization only.

Related concepts

“Implementing the larger log Relative Byte Address” on page 500

Before IBM MQ for z/OS 8.0, IBM MQ for z/OS used a 6 byte log RBA to identify the location of data within the log. From IBM MQ for z/OS 8.0, the log RBA can be 8 bytes long, increasing the period of time before you have to reset the log.

z/OS *Implementing the larger log Relative Byte Address*

Before IBM MQ for z/OS 8.0, IBM MQ for z/OS used a 6 byte log RBA to identify the location of data within the log. From IBM MQ for z/OS 8.0, the log RBA can be 8 bytes long, increasing the period of time before you have to reset the log.

Attention: You only have to carry out the following procedure to enable this feature if your queue managers were created before IBM MQ 9.3.0, as queue managers created at IBM MQ 9.3.0 and later already have this feature enabled.

See Planning to increase the maximum addressable log range for considerations when planning to enable 8 byte log RBA.

Perform these instructions, in the order shown, to enable 8 byte log RBA on a single IBM MQ for z/OS queue manager. For queue managers in a queue sharing group, perform the steps on each queue manager in turn.

1. Allocate new BSDS data sets with similar attributes to the current BSDS. You can tailor sample CSQ4BSDS and delete any irrelevant statement, or you can use your existing JCL, but change the BSDS name to something like `++HLQ++.NEW.BSDS01`.

   Notes:

   a. Check the attributes of your new BSDS before submitting the job to allocate the new BSDS. The only attribute that might change is the size of the BSDS.

   b. The new BSDS contains more data that the current BSDS, therefore, you must ensure that the new data sets are allocated with sufficient available space. The sample JCL in thlqual.SCSQPROC(CSQ4BSDS) contains the recommended values when defining a new BSDS.

2. Shut down the queue manager cleanly.
3. Run the BSDS conversion utility (CSQJUCNV) to convert the existing BSDS to the new BSDS data sets. This usually takes a few seconds to run.

   Your existing BSDS will not be changed during this process, and you can use that for the initialization of the queue manager in the case of an unsuccessful conversion.

4. Rename the current BSDS to become the old BSDS, and the new BSDS to become the current BSDS, so that the new data sets are used when you next restart the queue manager. You can use the DFSMS Access Method Services ALTER command, for example:

   ```
   ALTER '++HLQ++.BSDS01' NEWNAME('++HLQ++.OLD.BSDS01')
   ALTER '++HLQ++.NEW.BSDS01' NEWNAME('++HLQ++.BSDS01')
   ```

   Ensure that you also issue commands to rename both the data and index portions of the VSAM cluster.

5. Restart the queue manager. It should start in the same amount of time as it would have done when using 6 byte log RBA.

   If the queue manager does not restart successfully due to a failure to access the converted BSDS, attempt to identify the cause of the failure, resolve the problem and retry the operation. If required, contact your IBM support center for assistance.

   If necessary, the change can be backed out at this point by:

   a. Renaming the current BSDS to become the new BSDS.

   b. Renaming the old BSDS to become the current BSDS.

   c. Restarting the queue manager.

6. Once the queue manager has been successfully restarted with the converted BSDS, do not attempt to start the queue manager using the old BSDS.

7. Message CSQJ034I is issued during queue manager initialization to indicate the end of the log RBA for the queue manager as configured. Confirm that the end of the log RBA range displayed is FFFFFFFFFFFFFFFF. This indicates that 8 byte log RBA is in use.

Related tasks

Planning to increase the maximum addressable log range

Related reference

Larger log Relative Byte Address

The BSDS conversion utility (CSQJUCNV)

z/OS Managing the bootstrap data set (BSDS)

The bootstrap data set (BSDS) is used to reference log data sets, and log records. Use this topic to understand how you can examine, change, and recover the BSDS.

For more information, see The bootstrap data set.

This topic describes the tasks involved in managing the bootstrap data set. It contains these sections:

- “Finding out what the BSDS contains” on page 501
- “Changing the BSDS” on page 502
- “Recovering the BSDS” on page 506

z/OS *Finding out what the BSDS contains*

You can use the print log map utility (CSQJU004) to examine the contents of the BSDS.

The print log map utility (CSQJU004) is a batch utility that lists the information stored in the BSDS. For instructions on running it, see The print log map utility.

The BSDS contains:

- Time stamps
- Active log data set status

Time stamps in the BSDS

The output of the print log map utility shows the time stamps, which are used to record the date and time of various system events, that are stored in the BSDS.

The following time stamps are included in the header section of the report:

SYSTEM TIMESTAMP

Reflects the date and time the BSDS was last updated. The BSDS time stamp can be updated when:

- The queue manager starts.
- The write threshold is reached during log write activities. Depending on the number of output buffers you have specified and the system activity rate, the BSDS might be updated several times a second, or might not be updated for several seconds, minutes, or even hours. For details of the write threshold, see the WRTHRSH parameter of the CSQ6LOGP macro in Using CSQ6LOGP.
- IBM MQ drops into a single BSDS mode from its normal dual BSDS mode due to an error. This can occur when a request to get, insert, point to, update, or delete a BSDS record is unsuccessful. When this error occurs, IBM MQ updates the time stamp in the remaining BSDS to force a time stamp mismatch with the disabled BSDS.

UTILITY TIMESTAMP
The date and time the contents of the BSDS were altered by the change log inventory utility (CSQJU003).

The following time stamps are included in the active and archive log data sets portion of the report:

Active log date
The date the active log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done.

Active log time
The time the active log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done.

Archive log date
The date the archive log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done or the archive itself was done.

Archive log time
The time the archive log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done or the archive itself was done.

Active log data set status

The BSDS records the status of an active log data set as one of the following:

NEW
The data set has been defined but never used by IBM MQ, or the log was truncated to a point before the data set was first used. In either case, the data set starting and ending RBA values are reset to zero.

REUSABLE
Either the data set has been defined but never used by IBM MQ, or the data set has been offloaded. In the print log map output, the start RBA value for the last REUSABLE data set is equal to the start RBA value of the last archive log data set.

NOT REUSABLE
The data set contains records that have not been offloaded.

STOPPED
The offload processor encountered an error while reading a record, and that record could not be obtained from the other copy of the active log.

TRUNCATED
Either:

- An I/O error occurred, and IBM MQ has stopped writing to this data set. The active log data set is offloaded, beginning with the starting RBA and continuing up to the last valid record segment in the truncated active log data set. The RBA of the last valid record segment is lower than the ending RBA of the active log data set. Logging is switched to the next available active log data set, and continues uninterrupted.

or

- An ARCHIVE LOG function has been called, which has truncated the active log.

The status appears in the output from the print log map utility.

z/OS *Changing the BSDS*

You do not have to take special steps to keep the BSDS updated with records of logging events because IBM MQ does that automatically.

However, you might want to change the BSDS if you do any of the following:

- Add more active log data sets.
- Copy active log data sets to newly allocated data sets, for example, when providing larger active log allocations.

- Move log data sets to other devices.
- Recover a damaged BSDS.
- Discard outdated archive log data sets.

You can change the BSDS by running the change log inventory utility (CSQJU003). Only run this utility when the queue manager is inactive, or you might get inconsistent results. The action of the utility is controlled by statements in the SYSIN data set. This section shows several examples. For complete instructions, see The change log inventory utility.

You can copy an active log data set only when the queue manager is inactive because IBM MQ allocates the active log data sets as exclusive (DISP=OLD) at queue manager startup.

z/OS *Changes for active logs*

Use this topic to understand how you can change the active logs using the BSDS.

You can add to, delete from, and record entries in the BSDS for active logs using the change log utility. Examples only are shown here; replace the data set names shown with the ones you want to use. For more details of the utility, see The change log inventory utility.

See these sections for more information:

- Adding record entries to the BSDS
- Deleting information about the active log data set from the BSDS
- Recording information about the log data set in the BSDS
- Increasing the size of the active log
- The use of CSQJUFMT

Adding record entries to the BSDS

If an active log has been flagged as "stopped", it is not reused for logging; however, it continues to be used for reading. Use the access method services to define new active log data sets, then use the change log inventory utility to register the new data sets in the BSDS. For example, use:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

If you are copying the contents of an old active log data set to the new one, you can also give the RBA range and the starting and ending time stamps on the NEWLOG function.

Deleting information about the active log data set from the BSDS

To delete information about an active log data set from the BSDS, you could use:

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

Recording information about the log data set in the BSDS

To record information about an existing active log data set in the BSDS, use:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

You might need to insert a record containing this type of information in the BSDS because:

- The entry for the data set has been deleted, but is needed again.

- You are copying the contents of one active log data set to another data set.
- You are recovering the BSDS from a backup copy.

Increasing the size of the active log

There are two methods of achieving this process.

1. When the queue manager is active:
   a. Define new larger log data sets using JCL.
   b. Add the new log data sets to the active queue manager using the MQSC DEFINE LOG command.
   c. Use the MQSC ARCHIVE LOG command to move the current active log, to be a new larger log.
   d. Wait for the archive of the smaller active log data set to complete.
   e. Shut down the queue manager, using the CSQJU003 utility to remove the old small active logs.
   f. Restart the queue manager.
2. When the queue manager is inactive:
   a. Stop the queue manager. This step is required because IBM MQ allocates all active log data sets for its exclusive use when it is active.
   b. Use Access Method Services ALTER with the NEWNAME option to rename your active log data sets.
   c. Use Access Method Services DEFINE to define larger active log data sets.

      By reusing the old data set names, you do not have to run the change log inventory utility to establish new names in the BSDSs. The old data set names and the correct RBA ranges are already in the BSDSs.
   d. Use Access Method Services REPRO to copy the old (renamed) data sets into their appropriate new data sets.

      Note: This step can take a long time, so your enterprise could be out of action for this period.
   e. Start the queue manager.

If all your log data sets are the same size, your system will be operationally more consistent and efficient. If the log data sets are not the same size, it is more difficult to track your system's logs, and so space can be wasted.

The use of CSQJUFMT

Do not run a CSQJUFMT format when increasing the size of an active log.

If you run CSQJUFMT (in order to provide a performance advantage the first time the queue manager writes to the new active log) you receive messages:

```
IEC070I 203-204,XS95GTLX,REPRO02,OUTPUT,B857,SPMG02, 358
IEC070I MG.W.MG4E.LOGCOPY1.DS02,MG.W.MG4E.LOGCOPY1.DS02.DATA,
IDC3302I ACTION ERROR ON MG.W.MG4E.LOGCOPY1.DS02
IDC3351I ** VSAM I/O RETURN CODE IS 28 - RPLFDBWD = X'2908001C'
IDC31467I MAXIMUM ERROR LIMIT REACHED.

IDC0005I NUMBER OF RECORDS PROCESSED WAS 0
```

In addition, if you use the Access Method Services REPRO, ensure that you define a new empty log.

If you use REPRO to copy the old (renamed) data set into its respective new data set, the default is NOREPLACE.

This means that REPRO does not replace a record that is already on the designated data set. When formatting is done on the data set, the RBA value is reset. The net result is a data set that is not empty after formatting.

z/OS *Changes for archive logs*

Use this topic to understand how to change the archive logs.

You can add to, delete from, and change the password of, entries in the BSDS for archive logs. Examples only are shown here; replace the data set names shown with the ones you want to use. For more details of the utility, see The change log inventory utility.

- Adding an archive log
- Deleting an archive log
- Changing the password of an archive log

Adding an archive log

When the recovery of an object depends on reading an existing archive log data set, the BSDS must contain information about that data set so that IBM MQ can find it. To register information about an existing archive log data set in the BSDS, use:

```
NEWLOG DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,
UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

Deleting an archive log

To delete an entire archive log data set on one or more volumes, use:

```
DELETE DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

Changing the password of an archive log

If you change the password of an existing archive log data set, you must also change the information in the BSDS.

1. List the BSDS, using the print log map utility.
2. Delete the entry for the archive log data set with the changed password, using the DELETE function of the CSQJU003 utility (see topic The change log inventory utility).
3. Name the data set as for a new archive log data set. Use the NEWLOG function of the CSQJU003 utility (see topic The change log inventory utility), and give the new password, the starting and ending RBAs, and the volume serial numbers (which can be found in the print log map utility output, see The print log map utility).

To change the password for new archive log data sets, use:

```
ARCHIVE PASSWORD= password
```

To stop placing passwords on new archive log data sets, use:

```
ARCHIVE NOPASSWD
```

Note: Only use the ARCHIVE utility function if you do not have an external security manager.

z/OS *Changing the high-level qualifier (HLQ) for the logs and BSDS*

Use this topic to understand the procedure required to change the high-level qualifier (HLQ).

Before you begin

You must end the queue manager normally before copying any of the logs or data sets to the new data sets. This is to ensure that the data is consistent and no recovery is needed during restart.

About this task

This task provides information about how to change the HLQ for the logs and BSDS. To do this, follow these steps:

Procedure

1. Run the log print utility CSQJU004 to record the log data set information. This information is needed later.
2. You can either:
   a) run DSS backup and restore with rename on the log and BSDS data sets to be renamed, or
   b) use AMS DEFINE and REPRO to create the HLQ data sets and copy the data from the old data sets.
3. Modify the MSTR and CHIN procedures to point to the new data sets.
4. Delete the old log information in the new copy of the BSDS using CSQJU003.
5. Define the new log data sets to the new BSDS using the NEWLOG function of CSQJU003.
   Keep all information about each log the same, apart from the HLQ.
6. The new BSDS should reflect the same information that was recorded for the old logs in the old BSDS.
   The HLQ should be the only thing that has changed.

What to do next

Compare the CSQJU004 output for the old and new BSDS to ensure that they look EXACTLY the same (except for the HLQs) before starting the queue manager.

Note: Care must be taken when performing these operations. Incorrect actions might lead to unrecoverable situations. Check the PRINT LOG MAP UTILITY output and make sure that all the information needed for recovery or restart has been included.

z/OS ***Recovering the BSDS***

If IBM MQ is operating in dual BSDS mode and one BSDS becomes damaged, forcing IBM MQ into single BSDS mode, IBM MQ continues to operate without a problem (until the next restart).

To return the environment to dual BSDS mode:

1. Use Access Method Services to rename or delete the damaged BSDS and to define a new BSDS with the same name as the damaged BSDS. Example control statements can be found in job CSQ4BREC in thlqual.SCSQPROC.
2. Issue the IBM MQ command RECOVER BSDS to make a copy of the valid BSDS in the newly allocated data set and to reinstate dual BSDS mode.

If IBM MQ is operating in single BSDS mode and the BSDS is damaged, or if IBM MQ is operating in dual BSDS mode and both BSDSs are damaged, the queue manager stops and does not restart until the BSDS data sets are repaired. In this case:

1. Locate the BSDS associated with the most recent archive log data set. The data set name of the most recent archive log appears on the job log in the last occurrence of message CSQJ003I, which indicates that offload processing has been completed successfully. In preparation for the rest of this procedure, it is a good practice to keep a log of all successful archives noted by that message:

- If archive logs are on DASD, the BSDS is allocated on any available DASD. The BSDS name is like the corresponding archive log data set name; change only the first letter of the last qualifier, from A to B, as in this example:

  Archive log name
  CSQ.ARCHLOG1. ***A*** 0000001

  BSDS copy name
  CSQ.ARCHLOG1. ***B*** 0000001
- If archive logs are on tape, the BSDS is the first data set of the first archive log volume. The BSDS is not repeated on later volumes.

2. If the most recent archive log data set has no copy of the BSDS (for example, because an error occurred when offloading it), locate an earlier copy of the BSDS from earlier offload processing.
3. Rename *damaged* BSDSs using the Access Method Services ALTER command with the NEWNAME option. If you want to delete a damaged BSDS, use the Access Method Services DELETE command. For each damaged BSDS, use Access Method Services to define a new BSDS as a replacement data set. Job CSQ4BREC in thlqual.SCSQPROC contains Access Method Services control statements to define a new BSDS.
4. Use the Access Method Services REPRO command to copy the BSDS from the archive log to one of the replacement BSDSs you defined in step “3” on page 507. Do not copy any data to the second replacement BSDS, you do that in step “5” on page 508.

   a. Print the contents of the replacement BSDS.

      Use the print log map utility (CSQJU004) to print the contents of the replacement BSDS. This enables you to review the contents of the replacement BSDS before continuing your recovery work.

   b. Update the archive log data set inventory in the replacement BSDS.

      Examine the output from the print log map utility and check that the replacement BSDS does not contain a record of the archive log from which the BSDS was copied. If the replacement BSDS is an old copy, its inventory might not contain all archive log data sets that were created more recently. The BSDS inventory of the archive log data sets must be updated to reflect the current subsystem inventory.

      Use the change log inventory utility (CSQJU003) NEWLOG statement to update the replacement BSDS, adding a record of the archive log from which the BSDS was copied. If the archive log data set is password-protected, use the PASSWORD option of the NEWLOG function. Also, if the archive log data set is cataloged, ensure that the CATALOG option of the NEWLOG function is properly set to CATALOG=YES. Use the NEWLOG statement to add any additional archive log data sets that were created later than the BSDS copy.

   c. Update passwords in the replacement BSDS.

      The BSDS contains passwords for the archive log data sets and for the active log data sets. To ensure that the passwords in the replacement BSDS reflect the current passwords used by your installation, use the change log inventory ARCHIVE utility function with the PASSWORD option.

   d. Update the active log data set inventory in the replacement BSDS.

      In unusual circumstances, your installation might have added, deleted, or renamed active log data sets since the BSDS was copied. In this case, the replacement BSDS does not reflect the actual number or names of the active log data sets your installation currently has in use.

      If you need to delete an active log data set from the replacement BSDS log inventory, use the change log inventory utility DELETE function.

      If you need to add an active log data set to the replacement BSDS log inventory, use the change log inventory utility NEWLOG function. Ensure that the RBA range is specified correctly on the NEWLOG function. If the active log data set is password-protected, use the PASSWORD option.

      If you need to rename an active log data set in the replacement BSDS log inventory, use the change log inventory utility DELETE function, followed by the NEWLOG function. Ensure that the RBA range

is specified correctly on the NEWLOG function. If the active log data set is password-protected, use the PASSWORD option.

e. Update the active log RBA ranges in the replacement BSDS.

Later, when the queue manager restarts, it compares the RBAs of the active log data sets listed in the BSDS with the RBAs found in the actual active log data sets. If the RBAs do not agree, the queue manager does not restart. The problem is magnified when an old copy of the BSDS is used. To solve this problem, use the change log inventory utility (CSQJU003) to adjust the RBAs found in the BSDS using the RBAs in the actual active log data sets. You do this by:

- Using the print log records utility (CSQ1LOGP) to print a summary report of the active log data set. This shows the starting and ending RBAs.
- Comparing the actual RBA ranges with the RBA ranges you have just printed, when the RBAs of all active log data sets are known.

  If the RBA ranges are equal for all active log data sets, you can proceed to the next recovery step without any additional work.

  If the RBA ranges are not equal, adjust the values in the BSDS to reflect the actual values. For each active log data set that needs to have the RBA range adjusted, use the change log inventory utility DELETE function to delete the active log data set from the inventory in the replacement BSDS. Then use the NEWLOG function to redefine the active log data set to the BSDS. If the active log data sets are password-protected, use the PASSWORD option of the NEWLOG function.

f. If only two active log data sets are specified for each copy of the active log, IBM MQ can have difficulty during queue manager restart. The problem can arise when one of the active log data sets is full and has not been offloaded, while the second active log data set is close to filling. In this case, add a new active log data set for each copy of the active log and define each new active log data set in the replacement BSDS log inventory.

Use the Access Method Services DEFINE command to define a new active log data set for each copy of the active log and use the change log inventory utility NEWLOG function to define the new active log data sets in the replacement BSDS. You do not need to specify the RBA ranges on the NEWLOG statement. However, if the active log data sets are password-protected, use the PASSWORD option of the NEWLOG function. Example control statements to accomplish this task can be found in job CSQ4LREC in thlqual.SCSQPROC.

5. Copy the updated BSDS to the second new BSDS data set. The BSDSs are now identical.

   Use the print log map utility (CSQJU004) to print the contents of the second replacement BSDS at this point.
6. See Active log problems for information about what to do if you have lost your current active log data set.
7. Restart the queue manager using the newly constructed BSDS. IBM MQ determines the current RBA and what active logs need to be archived.

z/OS Managing page sets

Use this topic to understand how to manage the page sets associated with a queue manager.

This topic describes how to add, copy, and generally manage the page sets associated with a queue manager. It contains these sections:

- “How to reduce a page set” on page 513
- “How to reintroduce a page set” on page 514
- “How to back up and recover page sets” on page 515
- “How to delete page sets” on page 518
- “How to back up and restore queues using CSQUTIL” on page 518

See Page sets for a description of page sets, storage classes, buffers, and buffer pools, and some of the performance considerations that apply.

How to change the high-level qualifier (HLQ) for the page sets

This task gives information on how to change the HLQ for the page sets. To perform this task, do the following:

1. Define the new HLQ page sets.
2. If the size allocation is the same as the old page sets, copy the existing page set using REPRO to the empty new HLQ page sets.
3. If you are increasing the size of the page sets, use the FORMAT function of CSQUTIL to format the destination pages, and then the COPYPAGE function of CSQUTIL to copy all the messages from the source page set to the destination page set.

   For more information, see Formatting page sets (FORMAT), and Expanding a page set (COPYPAGE).
4. Change the CSQP00xx DD statement in the queue manager procedure to point to the new HLQ page sets.

Restart the queue manager and verify the changes to the page sets.

How to add a page set to a queue manager

This description assumes that you have a queue manager that is already running. You might need to add a page set if, for example, your queue manager has to cope with new applications using new queues.

To add a new page set, use the following procedure:

1. Define and format the new page set. You can use the sample JCL in thlqual.SCSQPROC(CSQ4PAGE) as a basis. For more information, see Formatting page sets (FORMAT).

   Take care not to format any page sets that are in use, unless this is what you intend. If so, use the FORCE option of the FORMAT utility function.
2. Use the DEFINE PSID command with the DSN option to associate the page set with a buffer pool.
3. Add the appropriate storage class definitions for your page set by issuing DEFINE STGCLASS commands.
4. Optionally, to document how your queue manager is configured:
   a. Add the new page set to the started task procedure for your queue manager.
   b. Add a definition for the new page set to your CSQINP1 initialization data set.
   c. Add a definition for the new storage class to your CSQ4INYR initialization data set member.

For details of the DEFINE PSID and DEFINE STGCLASS commands, see DEFINE PSID and DEFINE STGCLASS.

What to do when one of your page sets becomes full

You can find out about the utilization of page sets by using the IBM MQ command DISPLAY USAGE. For example, the command:

```
DISPLAY USAGE PSID(03)
```

displays the current state of the page set 03. This tells you how many free pages this page set has.

If you have defined secondary extents for your page sets, they are dynamically expanded each time they fill up. Eventually, all secondary extents are used, or no further disk space is available. If this happens, an application receives the return code MQRC_STORAGE_MEDIUM_FULL.

If an application receives a return code of MQRC_STORAGE_MEDIUM_FULL from an MQI call, this is a clear indication that there is not enough space remaining on the page set. If the problem persists or is likely to recur, you must do something to solve it.

You can approach this problem in a number of ways:

- Balance the load between page sets by moving queues from one page set to another.
- Expand the page set. See "How to increase the size of a page set" on page 512 for instructions.
- Redefine the page set so that it can expand beyond 4 GB to a maximum size of 64 GB. See Defining a page set to be larger than 4 GB for instructions.

How to balance loads on page sets

Load balancing on page sets means moving the messages associated with one or more queues from one page set to another, less used, page set. Use this technique if it is not practical to expand the page set.

To identify which queues are using a page set, use the appropriate IBM MQ commands. For example, to find out which queues are mapped to page set 02, first, find out which storage classes map to page set 02, by using the command:

```
DISPLAY STGCLASS(*) PSID(02)
```

Then use the following command to find out which queues use which storage class:

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

Moving a non-shared queue

To move queues and their messages from one page set to another, use the MQSC MOVE QLOCAL command (described in MOVE QLOCAL). When you have identified the queue or queues that you want to move to a new page set, follow this procedure for each of these queues:

1. Ensure that the queue you want to move is not in use by any applications (that is, IPPROCS and OPPROCS values from the DISPLAY QSTATUS command are zero) and that it has no uncommitted messages (the UNCOM value from the DISPLAY QSTATUS command is NO).

   Note: The only way to ensure that this state continues is to change the security authorization of the queue temporarily. See Profiles for queue security for more information.

   If you cannot do this, later stages in this procedure might fail if applications start to use the queue despite precautionary steps such as setting PUT(DISABLED). However, messages can never be lost by this procedure.
2. Prevent applications from putting messages on the queue being moved by altering the queue definition to disable MQPUT s. Change the queue definition to PUT(DISABLED).
3. Define a temporary queue with the same attributes as the queue that is being moved, using the command:

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
```

Note: If this temporary queue already exists from a previous run, delete it before doing the define.

4. Move the messages to the temporary queue using the following command:

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. Delete the queue you are moving, using the command:

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. Define a new storage class that maps to the required page set, for example:

```
DEFINE STGCLASS(NEW) PSID(nn)
```

Add the new storage class definition to the CSQINP2 data sets ready for the next queue manager restart.

7. Redefine the queue that you are moving, by changing the storage class attribute:

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
```

When the queue is redefined, it is based on the temporary queue created in step “3” on page 510.

8. Move the messages back to the new queue, using the command:

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

9. The queue created in step “3” on page 510 is no longer required. Use the following command to delete it:

```
DELETE QL(TEMP_QUEUE)
```

10. If the queue being moved was defined in the CSQINP2 data sets, change the STGCLASS attribute of the appropriate DEFINE QLOCAL command in the CSQINP2 data sets. Add the REPLACE keyword so that the existing queue definition is replaced.

Figure 29 on page 512 shows an extract from a load balancing job.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE STGCLASS(NEW) PSID(2)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 29. Extract from a load balancing job for a page set

How to increase the size of a page set

You can initially allocate a page set larger than 4 GB, See Defining a page set to be larger than 4 GB

A page set can be defined to be automatically expanded as it becomes full by specifying EXPAND(SYSTEM) or EXPAND(USER). If your page set was defined with EXPAND(NONE), you can expand it in either of two ways:

- Alter its definition to allow automatic expansion. See Altering a page set to allow automatic expansion
- Create a new, larger page set and copy the messages from the old page set to the new one. See Moving messages to a new, larger page set

Defining a page set to be larger than 4 GB

IBM MQ can use a page set up to 64 GB in size, provided the data set is defined with 'extended addressability' to VSAM. Extended addressability is an attribute which is conferred by an SMS data class.

Note: Page sets and active log data sets are eligible to reside in the extended addressing space (EAS) part of an extended address volumes (EAV) and, from z/OS V1.12, an archive log dataset can also reside in the EAS.

In the example shown in the following sample JCL, the management class 'EXTENDED' is defined to SMS with 'Extended addressability'. If your existing page set is not currently defined as having extended addressability, use the following method to migrate to an extended addressability format data set.

1. Stop the queue manager.
2. Use Access Method Services to rename the existing page set.
3. Define a destination page set, the same size as the existing page set, but with DATACLAS(EXTENDED).

   Note: Extended-format data sets must be SMS managed. These are the mechanisms for requesting extended format for VSAM data sets:

   - Using a data class that has a DSNTYPE value of EXT and the subparameter R or P to indicate required or preferred.
   - Coding DSNTYPE=EXTREQ (extended format is required) or DSNTYPE=EXTPREF (extended format is preferred) on the DD statement.

- Coding the LIKE= parameter on the DD statement to refer to an existing extended format data set.

For more information, see Restrictions on Defining Extended-Format Data Sets.

4. Use the COPYPAGE function of CSQUTIL to copy all the messages from the source page set to the destination page set. See Expanding a page set (COPYPAGE) for more details.
5. Restart the queue manager.
6. Alter the page set to use system expansion, to allow it to continue growing beyond its current allocation.

The following JCL shows example Access Method Services commands:

```
//S1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN   DD  *
ALTER 'VICY.CSQ1.PAGE01' -
NEWNAME('VICY.CSQ1.PAGE01.OLD')
ALTER 'VICY.CSQ1.PAGE01.DATA' -
NEWNAME('VICY.CSQ1.PAGE01.DATA.OLD')
DEFINE CLUSTER (NAME('VICY.CSQ1.PAGE01') -
MODEL('VICY.CSQ1.PAGE01.OLD') -
DATACLAS(EXTENDED))
/*
```

Altering a page set to allow automatic expansion

Use the ALTER PSID command with the EXPAND(USER) or EXPAND(SYSTEM) options. See ALTER PSID and Expanding a page set (COPYPAGE) for general information on expanding page sets.

Moving messages to a new, larger page set

This technique involves stopping and restarting the queue manager. This deletes any nonpersistent messages that are not on shared queues at restart time. If you have nonpersistent messages that you do not want to be deleted, use load balancing instead. For more details, see “How to balance loads on page sets” on page 510. In this description, the page set that you want to expand is referred to as the *source* page set; the new, larger page set is referred to as the *destination* page set.

Follow these steps:

1. Stop the queue manager.
2. Define the destination page set, ensuring that it is larger than the source page set, with a larger secondary extent value.
3. Use the FORMAT function of CSQUTIL to format the destination page set. See Formatting page sets (FORMAT) for more details.
4. Use the COPYPAGE function of CSQUTIL to copy all the messages from the source page set to the destination page set. See Expanding a page set (COPYPAGE) for more details.
5. Restart the queue manager using the destination page set by doing one of the following:
   - Change the queue manager started task procedure to reference the destination page set.
   - Use Access Method Services to delete the source page set and then rename the destination page set, giving it the same name as that of the source page set.

Attention:

Before you delete any IBM MQ page set, be sure that you have made the required backup copies.

How to reduce a page set

Prevent all users, other than the IBM MQ administrator, from using the queue manager. For example; by changing the access security settings.

If you have a large page set that is mostly empty (as shown by the DISPLAY USAGE command), you might want to reduce its size. The procedure to do this involves using the COPY, FORMAT, and LOAD functions of CSQUTIL (see IBM MQ utility program). This procedure does not work for page set zero (0), as it is not practical to reduce the size of this page set; the only way to do so is by reinitializing your queue manager (see “Reinitializing a queue manager” on page 536). The prerequisite of this procedure is to try and remove all users from the system so that all UOWs are complete and the page sets are consistent.

1. Use the STOP QMGR command with the QUIESCE or FORCE attribute to stop the queue manager.
2. Run the SCOPY function of CSQUTIL with the PSID option, to copy all message data from the large page set and save them in a sequential data set.
3. Define a new smaller page set data set to replace the large page set.
4. Run the FORMAT TYPE(NEW) function of CSQUTIL against the page set that you created in step “3” on page 514.
5. Restart the queue manager using the page set created in step “3” on page 514.
6. Run the LOAD function of CSQUTIL to load back all the messages saved during step “2” on page 514.
7. Allow all users access to the queue manager.
8. Delete the old large page set.

How to reintroduce a page set

In certain scenarios it is useful to be able to bring an old page set online again to the queue manager. Unless specific action is taken, when the old page set is brought online the queue manager will recognize that the page set recovery RBA stored in the page set itself and in the checkpoint records is old, and will therefore automatically start media recovery of the page set to bring it up to date.

Such media recovery can only be performed at queue manager restart, and is likely to take a considerable length of time, especially if archive logs held on tape must be read. However, normally in this circumstance, the page set has been offline for the intervening period and so the log contains no information pertinent to the page set recovery.

The following three choices are available:

Allow full media recovery to be performed.

1. Stop the queue manager.
2. Ensure definitions are available for the page set in both the started task procedure for the queue manager and in the CSQINP1 initialization data set.
3. Restart the queue manager.

Allow any messages on the page set to be destroyed.

This choice is useful where a page set has been offline for a long time (some months, for example) and it has now been decided to reuse it for a different purpose.

1. Format the page set using the FORMAT function of CSQUTIL with the TYPE(NEW) option.
2. Add definitions for the page set to both the started task procedure for the queue manager and the CSQINP1 initialization data set.
3. Restart the queue manager.

Using the TYPE(NEW) option for formatting clears the current contents of the page set and tells the queue manager to ignore any historical information in the checkpoint about the page set.

Bring the page set online avoiding the media recovery process.

Use this technique only if you are sure that the page set has been offline since a clean shutdown of the queue manager. This choice is most appropriate where the page set has been offline for a short period, typically due to operational issues such as a backup running while the queue manager is being started.

1. Format the page set using the FORMAT function of CSQUTIL with the TYPE(REPLACE) option.

2. Either add the page set back into the queue manager dynamically using the DEFINE PSID command with the DSN option or allow it to be added at a queue manager restart.

Using the TYPE(REPLACE) option for formatting checks that the page set was cleanly closed by the queue manager, and marks it so that media recovery will not be performed. No other changes are made to the contents of the page set.

z/OS How to back up and recover page sets

There are different mechanisms available for back up and recovery. Use this topic to understand these mechanisms.

This section describes the following topics:

- “Creating a point of recovery for non-shared resources” on page 515
- “Backing up page sets” on page 516
- “Recovering page sets” on page 517
- How to delete page sets

For information about how to create a point of recovery for shared resources, see “Recovering shared queues” on page 523.

Creating a point of recovery for non-shared resources

IBM MQ can recover objects and non-shared persistent messages to their current state if both:

1. Copies of page sets from an earlier point exist.
2. All the IBM MQ logs are available to perform recovery from that point.

These represent a point of recovery for non-shared resources.

Both objects and messages are held on page sets. Multiple objects and messages from different queues can exist on the same page set. For recovery purposes, objects and messages cannot be backed up in isolation, so a page set must be backed up as a whole to ensure the correct recovery of the data.

The IBM MQ recovery log contains a record of all persistent messages and changes made to objects. If IBM MQ fails (for example, due to an I/O error on a page set), you can recover the page set by restoring the backup copy and restarting the queue manager. IBM MQ applies the log changes to the page set from the point of the backup copy.

There are two ways of creating a point of recovery:

Full backup
Stop the queue manager, which forces all updates on to the page sets.

This allows you to restart from the point of recovery, using only the backed up page set data sets and the logs from that point on.

Fuzzy backup
Take *fuzzy* backup copies of the page sets without stopping the queue manager.

If you use this method, and your associated logs later become damaged or lost, you cannot to use the fuzzy page set backup copies to recover. This is because the fuzzy page set backup copies contain an inconsistent view of the state of the queue manager and are dependent on the logs being available. If the logs are not available, you need to return to the last set of backup page set copies taken while the subsystem was inactive (Method 1) and accept the loss of data from that time.

Method 1: Full backup

This method involves shutting the queue manager down. This forces all updates on to the page sets so that the page sets are in a consistent state.

1. Stop all the IBM MQ applications that are using the queue manager (allowing them to complete first). This can be done by changing the access security or queue settings, for example.
2. When all activity has completed, display and resolve any in-doubt units of recovery. (Use the commands DISPLAY CONN and RESOLVE INDOUBT, as described in DISPLAY CONN and RESOLVE INDOUBT.)

   This brings the page sets to a consistent state; if you do not do this, your page sets might not be consistent, and you are effectively doing a fuzzy backup.
3. Issue the ARCHIVE LOG command to ensure that the latest log data is written out to the log data sets.
4. Issue the STOP QMGR MODE(QUIESCE) command. Record the lowest RBA value in the CSQI024I or CSQI025I messages (see CSQI024I and CSQI025I for more information). You should keep the log data sets starting from the one indicated by the RBA value up to the current log data set.
5. Take backup copies of all the queue manager page sets (see "Backing up page sets" on page 516).

Method 2: Fuzzy backup

This method does not involve shutting the queue manager down. Therefore, updates might be in virtual storage buffers during the backup process. This means that the page sets are not in a consistent state, and can only be used for recovery with the logs.

1. Issue the DISPLAY USAGE TYPE(ALL) command, and record the RBA value in the CSQI024I or CSQI025I messages (see CSQI024I and CSQI025I for more information).
2. Take backup copies of the page sets (see "Backing up page sets" on page 516).
3. Issue the ARCHIVE LOG command, to ensure that the latest log data is written out to the log data sets. To restart from the point of recovery, you must keep the log data sets starting from the log data set indicated by the RBA value up to the current log data set.

Backing up page sets

To recover a page set, IBM MQ needs to know how far back in the log to go. IBM MQ maintains a log RBA number in page zero of each page set, called the *recovery log sequence number* (LSN). This number is the starting RBA in the log from which IBM MQ can recover the page set. When you back up a page set, this number is also copied.

If the copy is later used to recover the page set, IBM MQ must have access to all the log records from this RBA value to the current RBA. That means you must keep enough of the log records to enable IBM MQ to recover from the oldest backup copy of a page set you intend to keep.

Use ADRDSSU COPY function to copy the page sets.

For more information, see the COPY DATASET Command Syntax for Logical Data Set documentation .

For example:

```
//STEP2 EXEC PGM=ADRDSSU,REGION=6M
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
 COPY -
 DATASET(INCLUDE(SCENDATA.MQPA.PAGESET.*)) -
 RENAMEU(SCENDATA.MQPA.PAGESET.**,SCENDATA.MQPA.BACKUP1.**) -
 SPHERE -
 REPUNC -
 FASTREPLICATION(PREF )-
 CANCELERROR -
 TOL(ENQF)
/*
//
```

If you copy the page set while the queue manager is running you must use a copy utility that copies page zero of the page set first. If you do not do this you could corrupt the data in your page set.

If the process of dynamically expanding a page set is interrupted, for example by power to the system being lost, you can still use ADRDSSU to take a backup of a page set.

If you perform an Access Method Services IDCAMS `LISTCAT ENT('page set data set name') ALLOC`, you will see that the `HI-ALLOC-RBA` is higher than the `HI-USED-RBA`.

The next time this page set fills up it is extended again, if possible, and the pages between the high used RBA and the highest allocated RBA are used, along with another new extent.

Backing up your object definitions

You should also back up copies of your object definitions. To do this, use the MAKEDEF feature of the CSQUTIL COMMAND function (described in Issuing commands to IBM MQ (COMMAND)).

Back up your object definitions whenever you take a backup copy of your queue manager, and keep the most current version.

Recovering page sets

If the queue manager has terminated due to a failure, the queue manager can normally be restarted with all recovery being performed during restart. However, such recovery is not possible if any of your page sets or log data sets are not available. The extent to which you can now recover depends on the availability of backup copies of page sets and log data sets.

To restart from a point of recovery you must have:

- A backup copy of the page set that is to be recovered.
- If you used the "fuzzy" backup process described in “Method 2: Fuzzy backup” on page 516, the log data set that included the recorded RBA value, the log data set that was made by the ARCHIVE LOG command, and all the log data sets between these.
- If you used full backup, but you do not have the log data sets following that made by the ARCHIVE LOG command, you do **not** need to run the FORMAT TYPE(REPLACE) function of the CSQUTIL utility against all your page sets.

To recover a page set to its current state, you must also have all the log data sets and records since the ARCHIVE LOG command.

There are two methods for recovering a page set. To use either method, the queue manager must be stopped.

Simple recovery

This is the simpler method, and is appropriate for most recovery situations.

1. Delete the page set you want to restore from backup.
2. Use the ADRDSSU COPY function to recover your page set from the backup copy..

   Alternatively, you can rename your backup copy to the original name, or change the CSQP00xx DD statement in your queue manager procedure to point to your backup page set. However, if you then lose or corrupt the page set, you will no longer have a backup copy to restore from.
3. Restart the queue manager.
4. When the queue manager has restarted successfully, you can restart your applications
5. Reinstate your normal backup procedures for the restored page.

Advanced recovery

This method provides performance advantages if you have a large page set to recover, or if there has been much activity on the page set since the last backup copy was taken. However, it requires more manual intervention than the simple method, which might increase the risk of error and the time taken to perform the recovery.

1. Delete and redefine the page set you want to restore from backup.
2. Use ADRDSSU to copy the backup copy of the page set into the new page set. Define your new page set with a secondary extent value so that it can be expanded dynamically.

   Alternatively, you can rename your backup copy to the original name, or change the CSQP00xx DD statement in your queue manager procedure to point to your backup page set. However, if you then lose or corrupt the page set, you will no longer have a backup copy to restore from.
3. Change the CSQINP1 definitions for your queue manager to make the buffer pool associated with the page set being recovered as large as possible. By making the buffer pool large, you might be able to keep all the changed pages resident in the buffer pool and reduce the amount of I/O to the page set.
4. Restart the queue manager.
5. When the queue manager has restarted successfully, stop it (using quiesce) and then restart it using the normal buffer pool definition for that page set. After this second restart completes successfully, you can restart your applications
6. Reinstate your normal backup procedures for the restored page.

What happens when the queue manager is restarted

When the queue manager is restarted, it applies all changes made to the page set that are registered in the log, beginning at the restart point for the page set. IBM MQ can recover multiple page sets in this way. The page set is dynamically expanded, if required, during media recovery.

During restart, IBM MQ determines the log RBA to start from by taking the lowest value from the following:

- Recovery LSN from the checkpoint log record for each page set.
- Recovery LSN from page zero in each page set.
- The RBA of the oldest incomplete unit of recovery in the system at the time the backup was taken.

All object definitions are stored on page set zero. Messages can be stored on any available page set.

Note: The queue manager cannot restart if page set zero is not available.

How to delete page sets

You delete a page set by using the DELETE PSID command; see DELETE PSID for details of this command.

You cannot delete a page set that is still referenced by any storage class. Use DISPLAY STGCLASS to find out which storage classes reference a page set.

The data set is deallocated from IBM MQ but is not deleted. It remains available for future use, or can be deleted using z/OS facilities.

Remove the page set from the started task procedure for your queue manager.

Remove the definition of the page set from your CSQINP1 initialization data set.

z/OS How to back up and restore queues using CSQUTIL

Use this topic as a reference for further information about back up and restore using CSQUTIL.

You can use the CSQUTIL utility functions for backing up and restoring queues. To back up a queue, use the COPY or SCOPY function to copy the messages from a queue onto a data set. To restore the queue, use the complementary function LOAD or SLOAD. For more information, see IBM MQ utility program.

z/OS Managing buffer pools

Use this topic if you want to change or delete your buffer pools.

This topic describes how to alter and delete buffer pools. It contains these sections:

- “How to change the number of buffers in a buffer pool” on page 519
- “How to delete a buffer pool” on page 519

Buffer pools are defined during queue manager initialization, using DEFINE BUFFPOOL commands issued from the initialization input data set CSQINP1. Their attributes can be altered in response to business requirements while the queue manager is running, using the processes detailed in this topic. The queue manager records the current buffer pool attributes in checkpoint log records. These are automatically restored on subsequent queue manager restart, unless the buffer pool definition in CSQINP1 includes the REPLACE attribute.

Use the DISPLAY USAGE command to display the current buffer attributes.

You can also define buffer pools dynamically using the DEFINE PSID command with the DSN option.

If you change buffer pools dynamically, you should also update their definitions in the initialization data set CSQINP1.

See z/OS での計画 for a description of page sets, storage classes, buffers, and buffer pools, and some of the performance considerations that apply.

Note: Buffer pools use significant storage. When you increase the size of a buffer pool or define a new buffer pool ensure that sufficient storage is available. For more information, see Address space storage.

How to change the number of buffers in a buffer pool

If a buffer pool is too small, the condition can result in message CSQP020E on the console, you can allocate more buffers to it using the ALTER BUFFPOOL command as follows:

1. Determine how much space is available for new buffers by looking at the CSQY220I messages in the log. The available space is reported in MB. As a buffer has a size of 4 KB, each MB of available space allows you to allocate 256 buffers. Do not allocate all the free space to buffers, as some is required for other tasks.

   If the buffer pool uses fixed 4 KB pages, that is, its PAGECLAS attribute is FIXED4KB, ensure that there is sufficient real storage available on the LPAR.
2. If the reported free space is inadequate, release some buffers from another buffer pool using the command

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   where *buf-pool-id* is the buffer pool from which you want to reclaim space and *integer* is the new number of buffers to be allocated to this buffer pool, which must be smaller than the original number of buffers allocated to it.
3. Add buffers to the buffer pool you want to expand using the command

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   where *buf-pool-id* is the buffer pool to be expanded and *integer* is the new number of buffers to be allocated to this buffer pool, which must be larger than the original number of buffers allocated to it.

How to delete a buffer pool

When a buffer pool is no longer used by any page sets, delete it to release the virtual storage allocated to it.

You delete a buffer pool using the DELETE BUFFPOOL command. The command fails if any page sets are using this buffer pool.

See “How to delete page sets” on page 518 for information about how to delete page sets.

z/OS Managing queue sharing groups and shared queues on z/OS

IBM MQ can use different types of shared resources, for example queue sharing groups, shared queues, and the coupling facility. Use this topic to review the procedures needed to manage these shared resources.

This section contains information about the following topics:

z/OS *Managing queue sharing groups*

You can add or remove a queue manager to a queue sharing group (QSG), and manage the associated Db2 tables.

This topic has sections about the following tasks:

Setting up a queue sharing group

Each queue sharing group has a name of up to four characters. The name must be unique in your network, and must be different from any queue manager names.

Follow these steps to set up a queue sharing group:

1. If this is the first queue sharing group to use the Db2 data-sharing group, set up the Db2 environment.
2. Set up the coupling facility.
3. Add the queue sharing group to the Db2 tables. Use the ADD QSG function of the queue sharing group utility (CSQ5PQSG). This program is described in The queue sharing group utility. A sample is provided in thlqual.SCSQPROC(CSQ45AQS).
4. Add a queue manager to the queue sharing group by following the steps in “Adding a queue manager to a queue sharing group” on page 521
5. Define application structures to IBM MQ by following the steps in “Adding a coupling facility structure” on page 528.
6. If required, migrate non-shared queues to shared queues.
7. For availability, create shared channels into and out of the queue sharing group.
   - For connections into the queue sharing group:
     - Set up a VIPA socket or hardware router to distribute workload between the available queue managers in the QSG.
     - Define a receiver channel with QSGDISP(GROUP), to ensure the channel definition is available on all queue managers in the QSG.
     - Start a listener with INDISP(GROUP), on each queue manager, for MCA channel connections into the QSG. Client connections into the QSG should still connect to a listener started with INDISP(QMGR).
     - Change applications to connect using the QSG name, rather than a specific queue manager name.

  - Ensure that the channel authentication rules on all queue managers in the QSG are the same, to allow applications to connect to any queue manager in the QSG.
- For connections out of the queue sharing group:
  - Define a shared transmission queue.
  - Define the outbound channel with QSGDISP(GROUP) and DEFCDISP(SHARED).

If you convert an existing channel to a shared channel, you might need to issue the RESET CHANNEL command before starting the channel as the synchronization queue used by the channel will have changed.

Adding a queue manager to a queue sharing group

A queue manager can be added to an existing queue sharing group.

Note that:

- The queue sharing group must exist before you can add queue managers to it.
- A queue manager can be a member of only one queue sharing group.

Follow these steps to add a queue manager to a queue sharing group:

1. Perform the tasks in implement ESM security controls for the queue sharing group to grant the appropriate access to the queue manager and channel initiator user IDs.
2. If the queue sharing group has CF structures configured to offload data to SMDS, perform the tasks in set up the SMDS environment.
3. Stop the queue manager.
4. Use the ADD QMGR function of the queue sharing group utility (CSQ5PQSG). This program is described in the queue sharing group utility. A sample is provided in thlqual.SCSQPROC(CSQ45AQM).
5. Change your system parameter module to add queue sharing group data:
   a. Modify CSQ6SYSP to specify the QSGDATA parameter. See using CSQ6SYSP for more information.
   b. Assemble and link the system parameter module. You might want to use a different name for the load module.
   c. Change your startup process to use the new module.
6. Copy and tailor sample member thlqual.SCSQPROC(CSQ4INSS), which defines required CF structures and SYSTEM queues. Add the customized member to the CSQINP2 DD in the queue manager startup JCL.
7. Restart your queue manager using the queue sharing group system parameter module.
8. Optionally, migrate to security profiles prefixed by the queue sharing group name, instead of the queue manager name.
9. If shared channels are used for connections into the QSG, create channel authentication rules that mirror those on the other queue managers in the QSG, to allow applications to connect to any queue manager in the QSG.
10. 10. Optionally, do either of the following to allow applications connected to the queue manager in the QSG to put messages to queues hosted by other queue managers in the QSG:
    - Turn on intra-group queuing by issuing the command ALTER QMGR IGQ(ENABLED).
    - Define transmission queues and channels to the other queue managers in the QSG. Defining transmission queues with the same name as the target queue managers avoids the need to define remote queues and queue manager aliases.

Note: To add a queue manager to an existing queue sharing group containing queue managers running earlier versions of IBM MQ, you must first apply the coexistence PTF for the highest version of IBM MQ in the group to every earlier version queue manager in the group.

Removing a queue manager from a queue sharing group

You can only remove a queue manager from a queue sharing group if the queue manager's logs are not needed by another process, and all SMDS owned by the queue manager are empty.

See Deleting shared message data sets and DELETE CFSTRUCT for more information.

The logs are needed if they contain:

- The latest backup of one of the coupling facility (CF) application structures used by the queue sharing group
- Data needed by a future restore process, that is, the queue manager has used a recoverable structure since the time described by the last backup exclusion interval value.

If either or both of these points apply, or an SMDS owned by the queue manager contains messages, the queue manager cannot be removed. To determine which queue managers' logs are needed for a future restore process, use the MQSC DISPLAY CFSTATUS command with the TYPE(BACKUP) option (for details of this command, see DISPLAY CFSTATUS).

Use the following steps to remove a queue manager from a queue sharing group:

1. Stop any applications connected to the queue manager that put messages to shared queues.
2. Resolve any indoubt units of work involving this queue manager.
3. Determine if there are any messages in any SMDS owned by the queue manager by issuing the command DISPLAY USAGE TYPE(SMDS).
4. If there are offloaded messages for any application structure, wait until those messages have been retrieved from the queue. The number of offloaded messages reported by DISPLAY USAGE TYPE(SMDS) should be zero before proceeding.
5. Shut the queue manager down cleanly using STOP QMGR MODE(QUIESCE).
6. Wait for an interval at least equivalent to the value of the EXCLINT parameter you will specify in the BACKUP CFSTRUCT command in the next step.
7. On another queue manager, run a CF structure backup for each recoverable CF structure by using the MQSC BACKUP CFSTRUCT command and specifying an EXCLINT value as required in the previous step.
8. Confirm that the queue manager's logs are not needed to restore any CF structures, by inspecting the output from the command DISPLAY CFSTATUS(*) TYPE(BACKUP).
9. Use the REMOVE QMGR function of the CSQ5PQSG utility to remove the queue manager from the queue sharing group. This program is described in The queue sharing group utility. A sample is provided in thlqual.SCSQPROC(CSQ45RQM).
10. Before restarting the queue manager, reset the QSGDATA system parameter to its default value, and recreate the system parameter module. See Using CSQ6SYSP for information about how to tailor your system parameters.

Note, that when removing the last queue manager in a queue sharing group, you must use the FORCE option, rather than REMOVE. This removes the queue manager from the queue sharing group, while not performing the consistency checks of the queue manager logs being required for recovery. You should only perform this operation if you are deleting the queue sharing group.

Removing a queue sharing group from the Db2 tables

To remove a queue sharing group from the Db2 tables, use the REMOVE QSG function of the queue sharing group utility (CSQ5PQSG). This program is described in The queue sharing group utility. A sample is provided in thlqual.SCSQPROC(CSQ45RQS).

You can only remove a queue sharing group from the common Db2 data-sharing group tables after you have removed all the queue managers from the queue sharing group (as described in “Removing a queue manager from a queue sharing group” on page 522).

When the queue sharing group record is deleted from the queue sharing group administration table, all objects and administrative information relating to that queue sharing group are deleted from other IBM MQ Db2 tables. This includes shared queue and group object information.

Validating the consistency of Db2 definitions

Problems for shared queues within a queue sharing group can occur if the Db2 object definitions have, for any reason, become inconsistent.

To validate the consistency of the Db2 object definitions for queue managers, CF structures, and shared queues, use the VERIFY QSG function of the queue sharing group utility (CSQ5PQSG). This program is described in The queue sharing group utility.

z/OS *Managing shared queues*

Use this topic to understand how to recover, move, and migrate shared queues.

This section describes the following tasks:

- “Recovering shared queues” on page 523
- “Moving shared queues” on page 524
- “Migrating non-shared queues to shared queues” on page 526
- Suspending a Db2 connection

Recovering shared queues

IBM MQ can recover persistent messages on shared queues if all:

- Backups of the CF structures containing the messages have been performed.
- All the logs for all queue managers in the queue sharing group are available, to perform recovery from the point the backups are taken.
- Db2 is available and the structure backup table is more recent that the most recent CF structure backup.

The messages on a shared queue are stored in a coupling facility (CF) structure. Persistent messages can be put onto shared queues, and like persistent messages on non-shared queues, they are copied to the queue manager log. The MQSC BACKUP CFSTRUCT and RECOVER CFSTRUCT commands are provided to allow the recovery of a CF structure in the unlikely event of a coupling facility failure. In such circumstances, any nonpersistent messages stored in the affected structure are lost, but persistent messages can be recovered. Any further application activity using the structure is prevented until the structure has been recovered.

To enable recovery, you must back up your coupling facility list structures frequently using the MQSC BACKUP CFSTRUCT command. The messages in the CF structure are written onto the active log data set of the queue manager making the backup. It writes a record of the backup to Db2: the name of the CF structure being backed up, the name of the queue manager doing the backup, the RBA range for this backup on that queue manager log, and the backup time. Back up CF list structures even if you are not actively using shared queues, for example, if you have set up a queue sharing group intending to use it in the future.

You can recover a CF structure by issuing an MQSC RECOVER CFSTRUCT command to the queue manager that can perform the recovery; you can use any queue manager in the queue sharing group. You can specify a single CF structure to be recovered, or you can recover several CF structures simultaneously.

As noted previously, it is important that you back up your CF list structures frequently, otherwise recovering a CF structure can take a long time. Moreover, the recovery process cannot be canceled.

The definition of a shared queue is kept in a Db2 database and can therefore be recovered if necessary using standard Db2 database procedures. See Shared queues and queue sharing groups for more information.

Moving shared queues

This section describes how to perform load balancing by moving a shared queue from one coupling facility structure to another. It also describes how to move a non-shared queue to a shared queue, and how to move a shared queue to a non-shared queue.

When you move a queue, you need to define a temporary queue as part of the procedure. This is because every queue must have a unique name, so you cannot have two queues of the same name, even if the queues have different queue dispositions. IBM MQ tolerates having two queues with the same name (as in step "2" on page 524), but you cannot use the queues.

- Moving a queue from one coupling facility structure to another
- Moving a non-shared queue to a shared queue
- Moving a shared queue to a non-shared queue

Moving a queue from one coupling facility structure to another

To move queues and their messages from one CF structure to another, use the MQSC MOVE QLOCAL command. When you have identified the queue or queues that you want to move to a new CF structure, use the following procedure to move each queue:

1. Ensure that the queue you want to move is not in use by any applications, that is, the queue attributes IPPROCS and OPPROCS are zero on all queue managers in the queue sharing group.
2. Prevent applications from putting messages on the queue being moved by altering the queue definition to disable MQPUT s. Change the queue definition to PUT(DISABLED).
3. Define a temporary queue with the same attributes as the queue that is being moved using the following command:

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
```

   Note: If this temporary queue exists from a previous run, delete it before doing the define.
4. Move the messages to the temporary queue using the following command:

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. Delete the queue you are moving, using the command:

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. Redefine the queue that is being moved, changing the CFSTRUCT attribute, using the following command:

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
```

   When the queue is redefined, it is based on the temporary queue created in step "3" on page 524.
7. Move the messages back to the new queue using the command:

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

8. The queue created in step “3” on page 524 is no longer required. Use the following command to delete it:

```
DELETE QL(TEMP_QUEUE)
```

9. If the queue being moved was defined in the CSQINP2 data sets, change the CFSTRUCT attribute of the appropriate DEFINE QLOCAL command in the CSQINP2 data sets. Add the REPLACE keyword so that the existing queue definition is replaced.

Figure 30 on page 525 shows a sample job for moving a queue from one CF structure to another.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 30. Sample job for moving a queue from one CF structure to another

Moving a non-shared queue to a shared queue

The procedure for moving a non-shared queue to a shared queue is like the procedure for moving a queue from one CF structure to another (see “Moving a queue from one coupling facility structure to another” on page 524). Figure 31 on page 525 gives a sample job to do this.

Note: Remember that messages on shared queues are subject to certain restrictions on the maximum message size, message persistence, and queue index type, so you might not be able to move some non-shared queues to a shared queue.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 31. Sample job for moving a non-shared queue to a shared queue

Moving a shared queue to a non-shared queue

The procedure for moving a shared queue to a non-shared queue is like the procedure for moving a queue from one CF structure to another (see “Moving a queue from one coupling facility structure to another” on page 524).

Figure 32 on page 526 gives a sample job to do this.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW) QSGDISP(QMGR)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 32. Sample job for moving a shared queue to a non-shared queue

Migrating non-shared queues to shared queues

There are two stages to migrating non-shared queues to shared queues:

- Migrating the first (or only) queue manager in the queue sharing group
- Migrating any other queue managers in the queue sharing group

Migrating the first (or only) queue manager in the queue sharing group

Figure 31 on page 525 shows an example job for moving a non-shared queue to a shared queue. Do this for each queue that needs migrating.

Note:

1. Messages on shared queues are subject to certain restrictions on the maximum message size, message persistence, and queue index type, so you might not be able to move some non-shared queues to a shared queue.
2. You must use the correct index type for shared queues. If you migrate a transmission queue to be a shared queue, the index type must be MSGID.

If the queue is empty, or you do not need to keep the messages that are on it, migrating the queue is simpler. Figure 33 on page 527 shows an example job to use in these circumstances.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 33. Sample job for moving a non-shared queue without messages to a shared queue

Migrating any other queue managers in the queue sharing group

1. For each queue that does not have the same name as an existing shared queue, move the queue as described in Figure 31 on page 525 or Figure 33 on page 527.
2. For queues that have the same name as an existing shared queue, move the messages to the shared queue using the commands shown in Figure 34 on page 527.

```
MOVE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR)
```

Figure 34. Moving messages from a non-shared queue to an existing shared queue

Suspending a connection to Db2

If you want to apply maintenance or service to the Db2 tables or package related to shared queues without stopping your queue manager, you must temporarily disconnect queue managers in the data sharing group (DSG) from Db2.

To do this:

1. Use the MQSC command SUSPEND QMGR FACILITY(Db2).
2. Do the binds.
3. Reconnect to Db2 using the MQSC command RESUME QMGR FACILITY(Db2)

Note that there are restrictions on the use of these commands.

Attention: While the Db2 connection is suspended, the following operations will not be available. Therefore, you need to do this work during a time when your enterprise is at its least busy.

- Access to Shared queue objects for administration (define, delete,alter)
- Starting shared channels
- Storing messages in Db2
- Backup or recover CFSTRUCT

z/OS *Managing group objects*

Use this topic to understand how to work with group objects.

IBM MQ automatically copies the definition of a group object to page set zero of each queue manager that uses it. You can alter the copy of the definition temporarily, and IBM MQ allows you to refresh the page set copies from the repository copy. IBM MQ always tries to refresh the page set copies from the repository

copy on start-up (for channel objects, this is done when the channel initiator restarts). This ensures that the page set copies reflect the version on the repository, including any changes that were made when the queue manager was inactive.

There are circumstances under which the refresh is not performed, for example:

- If a copy of the queue is open, a refresh that would change the usage of the queue fails.
- If a copy of a queue has messages on it, a refresh that would delete that queue fails.

In these circumstances, the refresh is not performed on that copy, but is performed on the copies on all other queue managers. Check for and correct any problems with copy objects after adding, changing, or deleting a group object, and at queue manager or channel initiator restart.

z/OS *Managing the coupling facility*

Use this topic to understand how to add or remove coupling facility (CF) structures.

This section describes the following tasks:

- “Adding a coupling facility structure” on page 528
- “Removing a coupling facility structure” on page 528

Adding a coupling facility structure

To add a coupling facility structure, use the following procedure:

1. Define the CF structure in the CFRM policy data set. The information about setting up the coupling facility in Set up the coupling facility describes the rules for naming coupling facility structures, and how to define structures in the CFRM policy data set.
2. If you want to configure the structure to offload message data to SMDS, allocate and preformat data sets. See creating a shared message data set for details.
3. Define the structure to IBM MQ using the DEFINE CFSTRUCT command.

Removing a coupling facility structure

To remove a coupling facility structure, use the following procedure:

1. Use the following command to get a list of all the queues using the coupling facility structure that you want to delete:

```
DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)
```

2. Delete all the queues that use the structure.
3. Delete the CF structure from IBM MQ using the DELETE CFSTRUCT command.
4. If the structure was configured to offload message data to SMDS, delete the SMDS.
5. Remove the structure definition from your CFRM policy data set and run the IXCMIAPU utility. (This is the reverse of the customization task set up the coupling facility, described in Set up the coupling facility.)

z/OS *Tuning coupling facility list monitoring*

Use this topic to understand coupling facility list monitoring

Coupling facility (CF) list monitoring is used to monitor the state of list structures containing IBM MQ shared queues. When a message is added to a shared queue, and the queue's depth transitions from zero to non-zero, the CF notifies all queue managers in the queue sharing group. When notified the queue

managers might perform a number of actions, including notifying trigger monitors that are using TRIGGER(FIRST), or applications which are performing a get-wait.

By default, the CF notifies all queue managers in the queue sharing group at the same time. In certain configurations this can cause problems, such as:

- Skewed workload distribution, where a large percentage of messages go to a particular queue manager in the queue sharing group, often the queue manager running on the fastest LPAR, or which is closest to the CF, or
- A large number of failed gets, resulting in wasted CPU time.

z/OS V2R3 introduces a new coupling facility resource manager (CFRM) attribute called **KEYRNOTIFYDELAY**, which can be used with list structures containing shared queues (that is, application structures, and not the admin structure), and which can, for certain workloads, minimize the effects of workload skewing and empty MQGET calls, or empty MQGET calls.

KEYRNOTIFYDELAY can only be set on structures in a CF, running at CFLEVEL 22 or higher.

Its value must be one to seven decimal digits, in a range from 0 to 1,000,000 microseconds. If set to a non-zero value and the depth of a queue transitions from zero to non-zero, the CF selects a single queue manager from the queue sharing group, and notifies that queue manager before all the other queue managers in the group.

The queue manager is selected in a round-robin manner. If the selected queue manager does not process the message inside the time interval described by **KEYRNOTIFYDELAY** all the other queue managers in the queue sharing group will also be notified.

More information on **KEYRNOTIFYDELAY** is available here: Understanding Keyrange Monitoring Notification Delay.

Note that there are two similar CFRM attributes called **LISTNOTIFYDELAY** and **SUBNOTIFYDELAY**. Neither of these has any measurable effect on IBM MQ workload.

z/OS Recovery and restart on z/OS

Use this topic to understand the recovery and restart mechanisms used by IBM MQ.

z/OS Restarting IBM MQ

After a queue manager terminates there are different restart procedures needed depending on how the queue manager terminated. Use this topic to understand the different restart procedures that you can use.

This topic contains information about how to restart your queue manager in the following circumstances:

- “Restarting after a normal shutdown” on page 529
- “Restarting after an abnormal termination” on page 530
- “Restarting if you have lost your page sets” on page 530
- “Restarting if you have lost your log data sets” on page 530
- Restarting if you have lost your CF structures

Restarting after a normal shutdown

If the queue manager was stopped with the STOP QMGR command, the system finishes its work in an orderly way and takes a termination checkpoint before stopping. When you restart the queue manager, it uses information from the system checkpoint and recovery log to determine the system status at shutdown.

To restart the queue manager, issue the START QMGR command as described in “Using MQSC to start and stop a queue manager on z/OS” on page 455.

Restarting after an abnormal termination

IBM MQ automatically detects whether restart follows a normal shutdown or an abnormal termination.

Starting the queue manager after it has terminated abnormally is different from starting it after the STOP QMGR command has been issued. If the queue manager terminates abnormally, it terminates without being able to finish its work or take a termination checkpoint.

To restart the queue manager, issue the START QMGR command as described in "Using MQSC to start and stop a queue manager on z/OS" on page 455. When you restart a queue manager after an abnormal termination, it refreshes its knowledge of its status at termination using information in the log, and notifies you of the status of various tasks.

Normally, the restart process resolves all inconsistent states. But, in some cases, you must take specific steps to resolve inconsistencies. This is described in "Recovering units of work manually" on page 542.

Restarting if you have lost your page sets

If you have lost your page sets, you need to restore them from your backup copies before you can restart the queue manager. This is described in "How to back up and recover page sets" on page 515.

The queue manager might take a long time to restart under these circumstances because of the length of time needed for media recovery.

Restarting if you have lost your log data sets

If, after stopping a queue manager (using the STOP QMGR command), both copies of the log are lost or damaged, you can restart the queue manager providing you have a consistent set of page sets (produced using Method 1: Full backup).

Follow this procedure:

1. Define new page sets to correspond to each existing page set in your queue manager. See Task 15: Define your page sets for information about page set definition.

   Ensure that each new page set is larger than the corresponding source page set.
2. Use the FORMAT function of CSQUTIL to format the destination page set. See Formatting page sets for more details.
3. Use the RESETPAGE function of CSQUTIL to copy the existing page sets or reset them in place, and reset the log RBA in each page. See Copying a page set and resetting the log for more information about this function.
4. Redefine your queue manager log data sets and BSDS using CSQJU003 (see The change log inventory utility).
5. Restart the queue manager using the new page sets. To do this, you do one of the following:
   - Change the queue manager started task procedure to reference the new page sets. See Task 6: Create procedures for the IBM MQ queue manager for more information.
   - Use Access Method Services to delete the old page sets and then rename the new page sets, giving them the same names as the old page sets.

Attention: Before you delete any IBM MQ page set, ensure that you have made the required backup copies.

If the queue manager is a member of a queue sharing group, GROUP and SHARED object definitions are not normally affected by lost or damaged logs. However, if any shared-queue messages are involved in a unit of work that was covered by the lost or damaged logs, the effect on such uncommitted messages is unpredictable.

Note: If logs are damaged and the queue manager is a member of a queue sharing group, the ability to recover shared persistent messages might be lost. Issue a BACKUP CFSTRUCT command immediately on another active queue manager in the queue sharing group for all CF structures with the RECOVER(YES) attribute.

Restarting if you have lost your CF structures

You do not need to restart if you lose your CF structures, because the queue manager does not terminate.

z/OS *Alternative site recovery on z/OS*

You can recover a single queue manager or a queue sharing group, or consider disk mirroring.

See the following sections for more details:

- Recovering a single queue manager at an alternative site
- Recovering a queue sharing group.
  - CF structure media recovery
  - Backing up the queue sharing group at the prime site
  - Recovering a queue sharing group at the alternative site
- Using disk mirroring

Recovering a single queue manager at an alternative site

If a total loss of an IBM MQ computing center occurs, you can recover on another queue manager or queue sharing group at a recovery site. (See “Recovering a queue sharing group at the alternative site” on page 534 for the alternative site recovery procedure for a queue sharing group.)

To recover on another queue manager at a recovery site, you must regularly back up the page sets and the logs. As with all data recovery operations, the objectives of disaster recovery are to lose as little data, workload processing (updates), and time, as possible.

At the recovery site:

- The recovery queue managers **must** have the same names as the lost queue managers.
- The system parameter module (for example, CSQZPARM) used on each recovery queue manager must contain the same parameters as the corresponding lost queue manager.

When you have done this, reestablish all your queue managers as described in the following procedure. This can be used to perform disaster recovery at the recovery site for a single queue manager. It assumes that all that is available are:

- Copies of the archive logs and BSDSs created by normal running at the primary site (the active logs will have been lost along with the queue manager at the primary site).
- Copies of the page sets from the queue manager at the primary site that are the same age or older than the most recent archive log copies available.

You can use dual logging for the active and archive logs, in which case you need to apply the BSDS updates to both copies:

1. Define new page set data sets and load them with the data in the copies of the page sets from the primary site.
2. Define new active log data sets.
3. Define a new BSDS data set and use Access Method Services REPRO to copy the most recent archived BSDS into it.
4. Use the print log map utility CSQJU004 to print information from this most recent BSDS. At the time this BSDS was archived, the most recent archived log you have would have just been truncated as an active log, and does not appear as an archived log. Record the STARTRBA and ENDRBA of this log.

5. Use the change log inventory utility, CSQJU003, to register this latest archive log data set in the BSDS that you have just restored, using the STARTRBA and ENDRBA recorded in Step “4” on page 531.
6. Use the DELETE option of CSQJU003 to remove all active log information from the BSDS.
7. Use the NEWLOG option of CSQJU003 to add active logs to the BSDS, do not specify STARTRBA or ENDRBA.
8. Use CSQJU003 to add a restart control record to the BSDS. Specify CRESTART CREATE,ENDRBA=*highrba*, where *highrba* is the high RBA of the most recent archive log available (found in Step “4” on page 531), plus 1.

   The BSDS now describes all active logs as being empty, all the archived logs you have available, and no checkpoints beyond the end of your logs.
9. Restart the queue manager with the START QMGR command. During initialization, an operator reply message such as the following is issued:

   ```
   CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
   REPLY Y TO CONTINUE, N TO CANCEL
   ```

   Type Y to start the queue manager. The queue manager starts, and recovers data up to ENDRBA specified in the CRESTART statement.

See IBM MQ utilities on z/OS reference for information about using CSQJU003 and CSQJU004.

The following example shows sample input statements for CSQJU003 for steps 6, 7, and 8:

```
* Step 6
DELETE DSNAME=MQM2.LOGCOPY1.DS01
DELETE DSNAME=MQM2.LOGCOPY1.DS02
DELETE DSNAME=MQM2.LOGCOPY1.DS03
DELETE DSNAME=MQM2.LOGCOPY1.DS04
DELETE DSNAME=MQM2.LOGCOPY2.DS01
DELETE DSNAME=MQM2.LOGCOPY2.DS02
DELETE DSNAME=MQM2.LOGCOPY2.DS03
DELETE DSNAME=MQM2.LOGCOPY2.DS04

* Step 7
NEWLOG DSNAME=MQM2.LOGCOPY1.DS01,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS02,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS03,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS04,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY2.DS01,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS02,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS03,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS04,COPY2

* Step 8
CRESTART CREATE,ENDRBA=063000
```

The things you need to consider for restarting the channel initiator at the recovery site are like those faced when using ARM to restart the channel initiator on a different z/OS image. See “Using ARM in an IBM MQ network” on page 540 for more information. Your recovery strategy should also cover recovery of the IBM MQ product libraries and the application programming environments that use IBM MQ (CICS , for example).

Other functions of the change log inventory utility (CSQJU003) can also be used in disaster recovery scenarios. The HIGHRBA function allows the update of the highest RBA written and highest RBA offloaded values within the bootstrap data set. The CHECKPT function allows the addition of new checkpoint queue records or the deletion of existing checkpoint queue records in the BSDS.

Attention: These functions might affect the integrity of your IBM MQ data. Only use them in disaster recovery scenarios under the guidance of IBM service personnel.

Fast copy techniques

If copies of all the page sets and logs are made while the queue manager is frozen, the copies will be a consistent set that can be used to restart the queue manager at an alternative site. They typically enable a much faster restart of the queue manager, as there is little media recovery to be performed.

Use the SUSPEND QMGR LOG command to freeze the queue manager. This command flushes buffer pools to the page sets, takes a checkpoint, and stops any further log write activity. Once log write activity has been suspended, the queue manager is effectively frozen until you issue a RESUME QMGR LOG command. While the queue manager is frozen, the page sets and logs can be copied.

By using copying tools such as FLASHCOPY or SNAPSHOT to rapidly copy the page sets and logs, the time during which the queue manager is frozen can be reduced to a minimum.

Within a queue sharing group, however, the SUSPEND QMGR LOG command might not be such a good solution. To be effective, the copies of the logs must all contain the same point in time for recovery, which means that the SUSPEND QMGR LOG command must be issued on all queue managers within the queue sharing group simultaneously, and therefore the entire queue sharing group will be frozen for some time.

Recovering a queue sharing group

In the event of a prime site disaster, you can restart a queue sharing group at a remote site using backup data sets from the prime site. To recover a queue sharing group you need to coordinate the recovery across all the queue managers in the queue sharing group, and coordinate with other resources, primarily Db2. This section describes these tasks in detail.

- CF structure media recovery
- Backing up the queue sharing group at the prime site
- Recovering a queue sharing group at the alternative site

CF structure media recovery

Media recovery of a CF structure used to hold persistent messages on a shared queue, relies on having a backup of the media that can be forward recovered by the application of logged updates. Take backups of your CF structures periodically using the MQSC BACKUP CFSTRUCT command. All updates to shared queues (MQGETs and MQPUTs) are written on the log of the queue manager where the update is performed. To perform media recovery of a CF structure you must apply logged updates to that backup from the logs of all the queue managers that have used that CF structure. When you use the MQSC RECOVER CFSTRUCT command, IBM MQ automatically merges the logs from relevant queue managers, and applies the updates to the most recent backup.

The CF structure backup is written to the log of the queue manager that processed the BACKUP CFSTRUCT command, so there are no additional data sets to be collected and transported to the alternative site.

Backing up the queue sharing group at the prime site

At the prime site you need to establish a consistent set of backups on a regular basis, which can be used in the event of a disaster to rebuild the queue sharing group at an alternative site. For a single queue manager, recovery can be to an arbitrary point in time, typically the end of the logs available at the remote site. However, where persistent messages have been stored on a shared queue, the logs of all the queue managers in the queue sharing group must be merged to recover shared queues, as any queue manager in the queue sharing group might have performed updates (MQPUT s or MQGET s) on the queue.

For recovery of a queue sharing group, you need to establish a point in time that is within the log range of the log data of all queue managers. However, as you can only **forward** recover media from the log, this point in time must be after the BACKUP CFSTRUCT command has been issued and after any page set backups have been performed. (Typically, the point in time for recovery might correspond to the end of a business day or week.)

The following diagram shows time lines for two queue managers in a queue sharing group. For each queue manager, fuzzy backups of page sets are taken (see Method 2: Fuzzy backup). On queue manager A, a BACKUP CFSTRUCT command is issued. Subsequently, an ARCHIVE LOG command is issued on each queue manager to truncate the active log, and copy it to media offline from the queue manager, which can be transported to the alternative site. End of log identifies the time at which the ARCHIVE LOG command was issued, and therefore marks the extent of log data typically available at the alternative site. The point in time for recovery must lie between the end of any page set or CF structure backups, and the earliest end of log available at the alternative site.

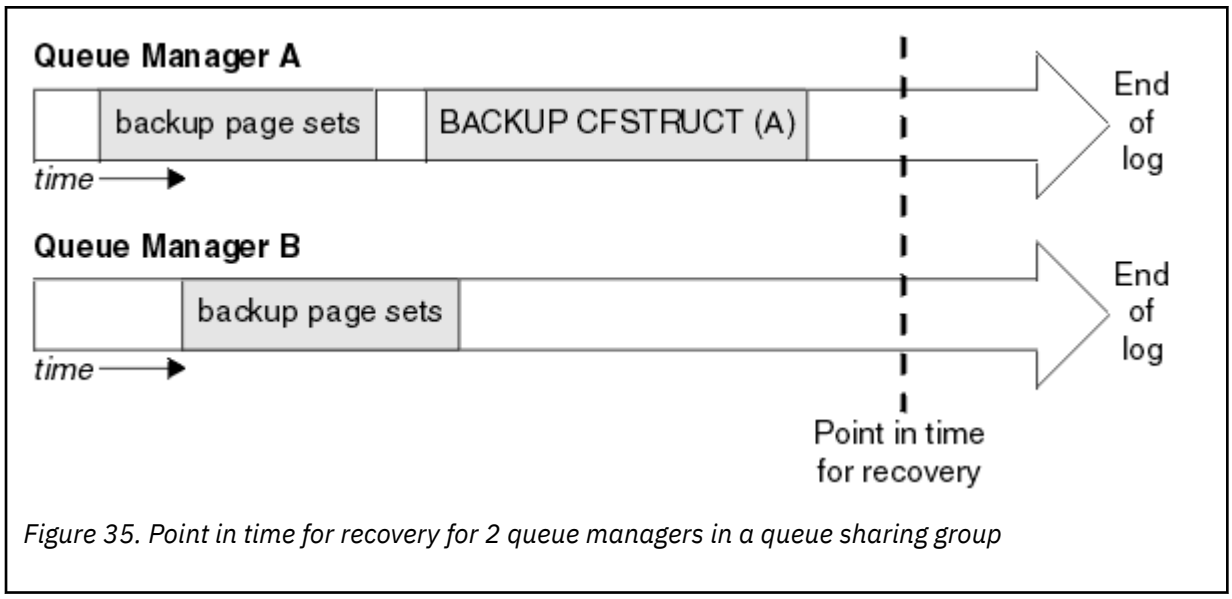


Figure 35. Point in time for recovery for 2 queue managers in a queue sharing group

IBM MQ records information associated with the CF structure backups in a table in Db2. Depending on your requirements, you might want to coordinate the point in time for recovery of IBM MQ with that for Db2, or it might be sufficient to take a copy of the IBM MQ CSQ.ADMIN_B_STRBACKUP table after the BACKUP CFSTRUCT commands have finished.

To prepare for a recovery:

1. Create page set backups for each queue manager in the queue sharing group.
2. Issue a BACKUP CFSTRUCT command for each CF structure with the RECOVER(YES) attribute. You can issue these commands from a single queue manager, or from different queue managers within the queue sharing group to balance the workload.
3. Once all the backups have completed, issue an ARCHIVE LOG command to switch the active log and create copies of the logs and BSDSs of each queue manager in the queue sharing group.
4. Transport the page set backups, the archived logs, the archived BSDS of all the queue managers in the queue sharing group, and your chosen Db2 backup information, off-site.

Recovering a queue sharing group at the alternative site

Before you can recover the queue sharing group, you need to prepare the environment:

1. If you have old information in your coupling facility from practice startups when you installed the queue sharing group, you need to clean this out first:

   Note: If you do not have old information in the coupling facility, you can omit this step.

   a. Enter the following z/OS command to display the CF structures for this queue sharing group:

```
D XCF,STRUCTURE,STRNAME= qsgname
```

b. For all structures that start with the queue sharing group name, use the z/OS command SETXCF FORCE CONNECTION to force the connection off those structures:

```
SETXCF FORCE,CONNECTION,STRNAME= strname,CONNAME=ALL
```

c. Delete all the CF structures using the following command for each structure:

```
SETXCF FORCE,STRUCTURE,STRNAME= strname
```

2. Restore Db2 systems and data-sharing groups.
3. Recover the CSQ.ADMIN_B_STRBACKUP table so that it contains information about the most recent structure backups taken at the prime site.

   Note: It is important that the STRBACKUP table contains the most recent structure backup information. Older structure backup information might require data sets that you have discarded as a result of the information given by a recent DISPLAY USAGE TYPE(DATASET) command, which would mean that your recovered CF structure would not contain accurate information.
4. Run the ADD QMGR command of the CSQ5PQSG utility for every queue manager in the queue sharing group. This will restore the XCF group entry for each queue manager.

   When you run the utility in this scenario, the following messages are normal:

```
CSQU566I Unable to get attributes for admin structure, CF not found
or not allocated
CSQU546E Unable to add QMGR queue_manager_name entry,
already exists in DB2 table CSQ.ADMIN_B_QMGR
CSQU148I CSQ5PQSG Utility completed, return code=4
```

To recover the queue managers in the queue sharing group:

1. Define new page set data sets and load them with the data in the copies of the page sets from the primary site.
2. Define new active log data sets.
3. Define a new BSDS data set and use Access Method Services REPRO to copy the most recent archived BSDS into it.
4. Use the print log map utility CSQJU004 to print information from this most recent BSDS. At the time this BSDS was archived, the most recent archived log you have would have just been truncated as an active log, and does not appear as an archived log. Record the STARTRBA, STARTLRSN, ENDRBA, and ENDLRSN values of this log.
5. Use the change log inventory utility, CSQJU003, to register this latest archive log data set in the BSDS that you have just restored, using the values recorded in Step “4” on page 535.
6. Use the DELETE option of CSQJU003 to remove all active log information from the BSDS.
7. Use the NEWLOG option of CSQJU003 to add active logs to the BSDS, do not specify STARTRBA or ENDRBA.
8. Calculate the *recoverylrsn* for the queue sharing group. The *recoverylrsn* is the lowest of the ENDLRSNs across all queue managers in the queue sharing group (as recorded in Step “4” on page 535), minus 1. For example, if there are two queue managers in the queue sharing group, and the ENDLRSN for one of them is B713 3C72 22C5, and for the other is B713 3D45 2123, the *recoverylrsn* is B713 3C72 22C4.
9. Use CSQJU003 to add a restart control record to the BSDS. Specify:

   ```
   CRESTART CREATE,ENDLRSN= recoverylrsn
   ```

   where *recoverylrsn* is the value you recorded in Step “8” on page 535.

   The BSDS now describes all active logs as being empty, all the archived logs you have available, and no checkpoints beyond the end of your logs.

You must add the CRESTART record to the BSDS for each queue manager within the queue sharing group.

10. Restart each queue manager in the queue sharing group with the START QMGR command. During initialization, an operator reply message such as the following is issued:

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
REPLY Y TO CONTINUE, N TO CANCEL
```

    Reply Y to start the queue manager. The queue manager starts, and recovers data up to ENDRBA specified in the CRESTART statement.

    The first IBM MQ queue manager started can rebuild the admin structure partitions for other members of the queue sharing group as well as its own, and it is no longer necessary to restart each queue manager in the queue sharing group at this stage.

11. When the admin structure data for all queue managers has been rebuilt, issue a RECOVER CFSTRUCT command for each CF application structure.

    If you issue the RECOVER CFSTRUCT command for all structures on a single queue manager, the log merge process is only performed once, so is quicker than issuing the command on a different queue manager for each CF structure, where each queue manager has to perform the log merge step.

When conditional restart processing is used in a queue sharing group, IBM MQ queue managers, performing peer admin rebuild, check that peers BSDS contain the same CRESTART LRSN as their own. This is to ensure the integrity of the rebuilt admin structure. It is therefore important to restart other peers in the QSG, so they can process their own CRESTART information, before the next unconditional restart of any member of the group.

Using disk mirroring

Many installations now use disk mirroring technologies such as IBM Metro Mirror (formerly PPRC) to make synchronous copies of data sets at an alternative site. In such situations, many of the steps detailed become unnecessary as the IBM MQ page sets and logs at the alternative site are effectively identical to those at the prime site. Where such technologies are used, the steps to restart a queue sharing group at an alternative site may be summarized as:

- Clear IBM MQ CF structures at the alternative site. (These often contain residual information from any previous disaster recovery exercise).
- Restore Db2 systems and all tables in the database used by the IBM MQ queue sharing group.
- Restart queue managers. Before IBM WebSphere MQ 7.0.1, it is necessary to restart each queue manager defined in the queue sharing group as each queue manage recovers its own partition of the admin structure during queue manager restart. After each queue manager has been restarted, those not on their home LPAR can be shut down again. The first IBM MQ queue manager started rebuilds the admin structure partitions for other members of the queue sharing group as well as its own, and it is no longer necessary to restart each queue manager in the queue sharing group.
- After the admin structure has been rebuilt, recover the application structures.

IBM MQ for z/OS supports use of zHyperWrite when writing to active logs mirrored using Metro Mirror. zHyperWrite can help reduce the performance impact of using Metro Mirror; see Using Metro Mirror with IBM MQ for more information.

z/OS *Reinitializing a queue manager*

If the queue manager has terminated abnormally you might not be able to restart it. This could be because your page sets or logs have been lost, truncated, or corrupted. If this has happened, you might have to reinitialize the queue manager (perform a cold start).

Attention

Only perform a cold start if you cannot restart the queue manager any other way. Performing a cold start enables you to recover your queue manager and your object definitions; you will **not** be able to recover your message data. Check that none of the other restart scenarios described in this topic work for you before you do this.

When you have restarted, all your IBM MQ objects are defined and available for use, but there is no message data.

Note: Do not reinitialize a queue manager while it is part of a cluster. You must first remove the queue manager from the cluster (using RESET CLUSTER commands on the other queue managers in the cluster), then reinitialize it, and finally reintroduce it to the cluster as a new queue manager.

This is because during reinitialization, the queue manager identifier (QMID) is changed, so any cluster object with the old queue manager identifier must be removed from the cluster.

For further information see the following sections:

- Reinitializing a queue manager that is not in a queue sharing group
- Reinitializing queue managers in a queue sharing group

Reinitializing a queue manager that is not in a queue sharing group

To reinitialize a queue manager, follow this procedure:

1. Prepare the object definition statements that to be used when you restart the queue manager. To do this, either:
   - If page set zero is available, use the CSQUTIL SDEFS function (see Producing a list of IBM MQ define commands). You must get definitions for all object types (authentication information objects, CF structures, channels, namelists, processes, queues, and storage classes).
   - If page set zero is not available, use the definitions from the last time you backed up your object definitions.
2. Redefine your queue manager data sets (do not do this until you have completed step “1” on page 537).

   See creating the bootstrap and log data sets and defining your page sets for more information.
3. Restart the queue manager using the newly defined and initialized log data sets, BSDS, and page sets. Use the object definition input statements that you created in step “1” on page 537 as input in the CSQINP2 initialization input data set.

Reinitializing queue managers in a queue sharing group

In a queue sharing group, reinitializing a queue manager is more complex. It might be necessary to reinitialize one or more queue managers because of page set or log problems, but there might also be problems with Db2 or the coupling facility to deal with. Because of this, there are a number of alternatives:

Cold start
Reinitializing the entire queue sharing group involves forcing all the coupling facilities structures, clearing all object definitions for the queue sharing group from Db2, deleting or redefining the logs and BSDS, and formatting page sets for all the queue managers in the queue sharing group.

Shared definitions retained
Delete or redefine the logs and BSDS, format page sets for all queue managers in the queue sharing group, and force all the coupling facilities structures. On restart, all messages will have been deleted. The queue managers re-create COPY objects that correspond to GROUP objects that still exist in the Db2 database. Any shared queues still exist and can be used.

Single queue manager reinitialized
Delete or redefine the logs and BSDS, and format page sets for the single queue manager (this deletes all its private objects and messages). On restart, the queue manager re-creates COPY objects that correspond to GROUP objects that still exist in the Db2 database. Any shared queues still exist, as do the messages on them, and can be used.

Point in time recovery of a queue sharing group
This is the alternative site disaster recovery scenario.

Shared objects are recovered to the point in time achieved by Db2 recovery (described in A Db2 system fails). Each queue manager can be recovered to a point in time achievable from the backup copies available at the alternative site.

Persistent messages can be used in queue sharing groups, and can be recovered using the MQSC RECOVER CFSTRUCT command. Note that this command recovers to the time of failure. However, there is no recovery of nonpersistent shared queue messages; they are lost unless you have made backup copies independently using the COPY function of the CSQUTIL utility program.

It is not necessary to try to restore each queue manager to the same point in time because there are no interdependencies between the local objects on different queue managers (which are what is actually being recovered), and the queue manager resynchronization with Db2 on restart creates or deletes COPY objects as necessary on a queue manager by queue manager basis.

z/OS Using the z/OS Automatic Restart Manager (ARM)

Use this topic to understand how you can use ARM to automatically restart your queue managers.

This section contains information about the following topics:

- “What is the ARM?” on page 538
- “ARM policies” on page 539
- “Using ARM in an IBM MQ network” on page 540

What is the ARM?

The z/OS Automatic Restart Manager (ARM) is a z/OS recovery function that can improve the availability of your queue managers. When a job or task fails, or the system on which it is running fails, ARM can restart the job or task without operator intervention.

If a queue manager or a channel initiator has failed, ARM restarts it on the same z/OS image. If z/OS, and hence a whole group of related subsystems and applications have failed, ARM can restart all the failed systems automatically, in a predefined order, on another z/OS image within the sysplex. This is called a *cross-system restart*.

Restart the channel initiator by ARM only in exceptional circumstances. If the queue manager is restarted by ARM, restart the channel initiator from the CSQINP2 initialization data set (see “Using ARM in an IBM MQ network” on page 540).

You can use ARM to restart a queue manager on a different z/OS image within the sysplex in the event of z/OS failure. The network implications of IBM MQ ARM restart on a different z/OS image are described in “Using ARM in an IBM MQ network” on page 540.

To enable automatic restart:

- Set up an ARM couple data set.
- Define the automatic restart actions that you want z/OS to perform in an *ARM policy*.
- Start the ARM policy.

Also, IBM MQ must register with ARM at startup (this happens automatically).

Note: If you want to restart queue managers in different z/OS images automatically, you must define every queue manager as a subsystem in each z/OS image on which that queue manager might be restarted, with a sysplex wide unique four character subsystem name.

ARM couple data sets

Ensure that you define the couple data sets required for ARM, and that they are online and active before you start any queue manager for which you want ARM support. IBM MQ automatic ARM registration fails if the couple data sets are not available at queue manager startup. In this situation, IBM MQ assumes that the absence of the couple data set means that you do not want ARM support, and initialization continues.

See *z/OS MVS Setting up a Sysplex* for information about ARM couple data sets.

z/OS *ARM policies*

The Automatic Restart Manager policies are user-defined rules that control ARM functions that can control any restarts of a queue manager.

ARM functions are controlled by a user-defined *ARM policy*. Each z/OS image running a queue manager instance that is to be restarted by ARM must be connected to an ARM couple data set with an active ARM policy.

IBM provides a default ARM policy. You can define new policies, or override the policy defaults by using the *administrative data utility* (IXCMIAPU) provided with z/OS. *z/OS MVS Setting up a Sysplex* describes this utility, and includes full details of how to define an ARM policy.

Figure 36 on page 539 shows an example of an ARM policy. This sample policy restarts any queue manager within a sysplex, if either the queue manager failed, or a whole system failed.

```
//IXCMIAPU EXEC PGM=IXCMIAPU,REGION=2M
//SYSPRINT DD SYSOUT=*
//SYSIN  DD *
DATA TYPE(ARM)
DEFINE POLICY NAME(ARMPOL1) REPLACE(YES)
RESTART_GROUP(DEFAULT)
ELEMENT(*)
RESTART_ATTEMPTS(0)  /* Jobs not to be restarted by ARM  */
RESTART_GROUP(GROUP1)
ELEMENT(SYSMQMGRMQ*)  /* These jobs to be restarted by ARM */
/*
```

Figure 36. Sample ARM policy

For more information see:

- Defining an ARM policy
- Activating an ARM policy
- Registering with ARM

Defining an ARM policy

Set up your ARM policy as follows:

- Define RESTART_GROUPs for each queue manager instance that also contain any CICS or IMS subsystems that connect to that queue manager instance. If you use a subsystem naming convention, you might be able to use the '?' and '*' wild-card characters in your element names to define RESTART_GROUPs with minimum definition effort.
- Specify TERMTYPE(ELEMTERM) for your channel initiators to indicate that they will be restarted only if the channel initiator has failed and the z/OS image has not failed.

- Specify TERMTYPE(ALLTERM) for your queue managers to indicate that they will be restarted if either the queue manager has failed or the z/OS image has failed.
- Specify RESTART_METHOD(BOTH, PERSIST) for both queue managers and channel initiators. This tells ARM to restart using the JCL it saved (after resolution of system symbols) during the last startup. It tells ARM to do this irrespective of whether the individual element failed, or the z/OS image failed.
- Accept the default values for all the other ARM policy options.

Activating an ARM policy

To start your automatic restart management policy, issue the following z/OS command:

```
SETXCF START,POLICY,TYPE=ARM,POLNAME= mypol
```

When the policy is started, all systems connected to the ARM couple data set use the same active policy.

Use the SETXCF STOP command to disable automatic restarts.

Registering with ARM

IBM MQ registers automatically as an *ARM element* during queue manager startup (subject to ARM availability). It deregisters during its shutdown phase, unless requested not to.

At startup, the queue manager determines whether ARM is available. If it is, IBM MQ registers using the name SYSMQMGR *ssid*, where *ssid* is the four character queue manager name, and SYSMQMGR is the element type.

The STOP QMGR MODE(QUIESCE) and STOP QMGR MODE(FORCE) commands deregister the queue manager from ARM (if it was registered with ARM at startup). This prevents ARM restarting this queue manager. The STOP QMGR MODE(RESTART) command does not deregister the queue manager from ARM, so it is eligible for immediate automatic restart.

Each channel initiator address space determines whether ARM is available, and if so registers with the element name SYSMQCH *ssid*, where *ssid* is the queue manager name, and SYSMQCH is the element type.

The channel initiator is always deregistered from ARM when it stops normally, and remains registered only if it ends abnormally. The channel initiator is always deregistered if the queue manager fails.

z/OS *Using ARM in an IBM MQ network*

You can set up your queue manager so that the channel initiators and associated listeners are started automatically when the queue manager is restarted.

To ensure fully automatic queue manager restart on the same z/OS image for both LU 6.2 and TCP/IP communication protocols:

- Start your listeners automatically by adding the appropriate START LISTENER command to the CSQINPX data set.
- Start your channel initiator automatically by adding the appropriate START CHINIT command to the CSQINP2 data set.

For restarting a queue manager with TCP/IP or LU6.2, see

- “Restarting on a different z/OS image with TCP/IP” on page 541
- “Restarting on a different z/OS image with LU 6.2” on page 542

See Task 13: Customize the initialization input data sets for information about the CSQINP2 and CSQINPX data sets.

Restarting on a different z/OS image with TCP/IP

If you are using TCP/IP as your communication protocol, and you are using virtual IP addresses, you can configure these to recover on other z/OS images, allowing channels connecting to that queue manager to reconnect without any changes. Otherwise, you can reallocate a TCP/IP address after moving a queue manager to a different z/OS image only if you are using clusters or if you are connecting to a queue sharing group using a WLM dynamic Domain Name System (DNS) logical group name.

- When using clustering
- When connecting to a queue sharing group

When using clustering

z/OS ARM responds to a system failure by restarting the queue manager on a different z/OS image in the same sysplex; this system has a different TCP/IP address to the original z/OS image. The following explains how you can use IBM MQ clusters to reassign a queue manager's TCP/IP address after it has been moved by ARM restart to a different z/OS image.

When a client queue manager detects the queue manager failure (as a channel failure), it responds by reallocating suitable messages on its cluster transmission queue to a different server queue manager that hosts a different instance of the target cluster queue. However, it cannot reallocate messages that are bound to the original server by affinity constraints, or messages that are in doubt because the server queue manager failed during end-of-batch processing. To process these messages, do the following:

1. Allocate a different cluster-receiver channel name and a different TCP/IP port to each z/OS queue manager. Each queue manager needs a different port so that two systems can share a single TCP/IP stack on a z/OS image. One of these is the queue manager originally running on that z/OS image, and the other is the queue manager that ARM will restart on that z/OS image following a system failure. Configure each port on each z/OS image, so that ARM can restart any queue manager on any z/OS image.
2. Create a different channel initiator command input file (CSQINPX) for each queue manager and z/OS image combination, to be referenced during channel initiator startup.

   Each CSQINPX file must include a START LISTENER PORT(port) command specific to that queue manager, and an ALTER CHANNEL command for a cluster-receiver channel specific to that queue manager and z/OS image combination. The ALTER CHANNEL command needs to set the connection name to the TCP/IP name of the z/OS image on which it is restarted. It must include the port number specific to the restarted queue manager as part of the connection name.

   The start-up JCL of each queue manager can have a fixed data set name for this CSQINPX file, and each z/OS image must have a different version of each CSQINPX file on a non-shared DASD volume.

If an ARM restart occurs, IBM MQ advertises the changed channel definition to the cluster repository, which in turn publishes it to all the client queue managers that have expressed an interest in the server queue manager.

The client queue manager treats the server queue manager failure as a channel failure, and tries to restart the failed channel. When the client queue manager learns the new server connection-name, the channel restart reconnects the client queue manager to the restarted server queue manager. The client queue manager can then resynchronize its messages, resolve any in-doubt messages on the client queue manager's transmission queue, and normal processing can continue.

When connecting to a queue sharing group

When connecting to a queue sharing group through a TCP/IP dynamic Domain Name System (DNS) logical group name, the connection name in your channel definition specifies the logical group name of your queue sharing group, not the host name or IP address of a physical machine. When this channel starts, it connects to the dynamic DNS and is then connected to one of the queue managers in the queue sharing group. This process is explained in Setting up communication for IBM MQ for z/OS using queue sharing groups.

In the unlikely event of an image failure, one of the following occurs:

- The queue managers on the failing image de-register from the dynamic DNS running on your sysplex. The channel responds to the connection failure by entering RETRYING state and then connects to the dynamic DNS running on the sysplex. The dynamic DNS allocates the inbound request to one of the remaining members of the queue sharing group that is still running on the remaining images.
- If no other queue manager in the queue sharing group is active and ARM restarts the queue manager and channel initiator on a different image, the group listener registers with dynamic DNS from this new image. This means that the logical group name (from the connection name field of the channel) connects to the dynamic DNS and is then connected to the same queue manager, now running on a different image. No change was required to the channel definition.

For this type of recovery to occur, the following points must be noted:

- On z/OS, the dynamic DNS runs on one of the z/OS images in the sysplex. If this image were to fail, the dynamic DNS needs to be configured so that there is a secondary name server active in the sysplex, acting as an alternative to the primary name server. Information about primary and secondary dynamic DNS servers can be found in the *OS/390® SecureWay CS IP Configuration* manual
- The TCP/IP group listener might have been started on a particular IP address that might not be available on this z/OS image. If so, the listener might need to be started on a different IP address on the new image. If you are using virtual IP addresses, you can configure these to recover on other z/OS images so that no change to the START LISTENER command is required.

Restarting on a different z/OS image with LU 6.2

If you use only LU 6.2 communication protocols, carry out the following procedure to enable network reconnect after automatic restart of a queue manager on a different z/OS image within the sysplex:

- Define each queue manager within the sysplex with a unique subsystem name.
- Define each channel initiator within the sysplex with a unique LUNAME. This is specified in both the queue manager attributes and in the START LISTENER command.

  Note: The LUNAME names an entry in the APPC side table, which in turn maps this to the actual LUNAME.
- Set up a shared APPC side table, which is referenced by each z/OS image within the sysplex. This should contain an entry for each channel initiator's LUNAME. See *z/OS MVS Planning: APPC/MVS Management* for information about this.
- Set up an APPCPM *xx* member of SYS1.PARMLIB for each channel initiator within the sysplex to contain an LUADD to activate the APPC side table entry for that channel initiator. These members should be shared by each z/OS image. The appropriate SYS1.PARMLIB member is activated by a z/OS command SET APPC= *xx*, which is issued automatically during ARM restart of the queue manager (and its channel initiator) on a different z/OS image, as described in the following text.
- Use the LU62ARM queue manager attribute to specify the *xx* suffix of this SYS1.PARMLIB member for each channel initiator. This causes the channel initiator to issue the required z/OS command SET APPC= *xx* to activate its LUNAME.

Define your ARM policy so that it restarts the channel initiator only if it fails while its z/OS image stays up; the user ID associated with the XCFAS address space must be authorized to issue the IBM MQ command START CHINIT. Do not restart the channel initiator automatically if its z/OS image also fails, instead use commands in the CSQINP2 and CSQINPX data sets to start the channel initiator and listeners.

z/OS Recovering units of work manually

You can manually recover units of work CICS, IMS, RRS, or other queue managers in a queue sharing group. You can use queue manager commands to display the status of the units of work associated with each connection to the queue manager.

This topic contains information about the following subjects:

Displaying connections and threads

You can use the DISPLAY CONN command to get information about connections to queue managers and their associated units of work. You can display active units of work to see what is currently happening, or to see what needs to be terminated to allow the queue manager to shut down, and you can display unresolved units of work to help with recovery.

Active units of work

To display only active units of work, use

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ ACTIVE)
```

Unresolved units of work

An unresolved unit of work, also known as an "in-doubt thread", is one that is in the second pass of the two-phase commit operation. Resources are held in IBM MQ on its behalf. To display unresolved units of work, use

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

External intervention is needed to resolve the status of unresolved units of work. This might only involve starting the recovery coordinator (CICS, IMS, or RRS) or might involve more, as described in the following sections.

z/OS *Recovering CICS units of recovery manually*

Use this topic to understand what happens when the CICS adapter restarts, and then explains how to deal with any unresolved units of recovery that arise.

What happens when the CICS adapter restarts

Whenever a connection is broken, the adapter has to go through a *restart phase* during the *reconnect process*. The restart phase resynchronizes resources. Resynchronization between CICS and IBM MQ enables in-doubt units of work to be identified and resolved.

Resynchronization can be caused by:

- An explicit request from the distributed queuing component
- An implicit request when a connection is made to IBM MQ

If the resynchronization is caused by connecting to IBM MQ, the sequence of events is:

1. The connection process retrieves a list of in-doubt units of work (UOW) IDs from IBM MQ.
2. The UOW IDs are displayed on the console in CSQC313I messages.
3. The UOW IDs are passed to CICS.
4. CICS initiates a resynchronization task (CRSY) for each in-doubt UOW ID.

5. The result of the task for each in-doubt UOW is displayed on the console.

You need to check the messages that are displayed during the connect process:

CSQC313I
Shows that a UOW is in doubt.

CSQC400I
Identifies the UOW and is followed by one of these messages:

- CSQC402I or CSQC403I shows that the UOW was resolved successfully (committed or backed out).
- CSQC404E, CSQC405E, CSQC406E, or CSQC407E shows that the UOW was not resolved.

CSQC409I
Shows that all UOWs were resolved successfully.

CSQC408I
Shows that not all UOWs were resolved successfully.

CSQC314I
Warns that UOW IDs highlighted with a * are not resolved automatically. These UOWs must be resolved explicitly by the distributed queuing component when it is restarted.

Figure 37 on page 544 shows an example set of restart messages displayed on the z/OS console.

```
CSQ9022I +CSQ1 CSQYASCP ' START QMGR' NORMAL COMPLETION
+CSQC323I VICIC1 CSQCQCON CONNECT received from TERMID=PB62 TRANID=CKCN
+CSQC303I VICIC1 CSQCCON CSQCSERV loaded. Entry point is 850E8918
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F0E2178D25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F055B2AC25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6EFFD60D425 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F07AB56D22 is in doubt
+CSQC307I VICIC1 CSQCCON Successful connection to subsystem VC2
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAD18) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAA10) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BA708) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAE88) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAB80) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA878) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA570) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA268) connect
successful
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F0E2178D25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F055B2AC25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F07AB56D22
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6EFFD60D425
+CSQC409I VICIC1 CSQCTRUE Resynchronization completed successfully
```

Figure 37. Example restart messages

The total number of CSQC313I messages should equal the total number of CSQC402I plus CSQC403I messages. If the totals are not equal, there are UOWs that the connection process cannot resolve. Those UOWs that cannot be resolved are caused by problems with CICS (for example, a cold start) or with IBM MQ, or by distributing queuing. When these problems have been fixed, you can initiate another resynchronization by disconnecting and then reconnecting.

Alternatively, you can resolve each outstanding UOW yourself using the RESOLVE INDOUBT command and the UOW ID shown in message CSQC400I. You must then initiate a disconnect and a connect to clean up

the *unit of recovery descriptors* in CICS. You need to know the correct outcome of the UOW to resolve UOWs manually.

All messages that are associated with unresolved UOWs are locked by IBM MQ and no Batch, TSO, or CICS task can access them.

If CICS fails and an emergency restart is necessary, *do not* vary the GENERIC APPLID of the CICS system. If you do and then reconnect to IBM MQ, data integrity with IBM MQ cannot be guaranteed. This is because IBM MQ treats the new instance of CICS as a different CICS (because the APPLID is different). In-doubt resolution is then based on the wrong CICS log.

How to resolve CICS units of recovery manually

If the adapter ends abnormally, CICS and IBM MQ build in-doubt lists either dynamically or during restart, depending on which subsystem caused the abend.

Note: If you use the DFH$INDB sample program to show units of work, you might find that it does not always show IBM MQ UOWs correctly.

When CICS connects to IBM MQ, there might be one or more units of recovery that have not been resolved.

One of the following messages is sent to the console:

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E
- CSQC408I

For details of what these messages mean, see the CICS adapter and Bridge messages messages.

CICS retains details of units of recovery that were not resolved during connection startup. An entry is purged when it no longer appears on the list presented by IBM MQ.

Any units of recovery that CICS cannot resolve must be resolved manually using IBM MQ commands. This manual procedure is rarely used within an installation, because it is required only where operational errors or software problems have prevented automatic resolution. *Any inconsistencies found during in-doubt resolution must be investigated.*

To resolve the units of recovery:

1. Obtain a list of the units of recovery from IBM MQ using the following command:

```
+CSQ1  DISPLAY CONN( * ) WHERE(UOWSTATE EQ UNRESOLVED)
```

   You receive the following message:

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC85772CBE3E0001)
EXTCONN(C3E2D8C3C7D9F0F94040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2005-02-04)
UOWLOGTI(10.17.44)
UOWSTDA(2005-02-04)
UOWSTTI(10.17.44)
UOWSTATE(UNRESOLVED)
NID(IYRCSQ1 .BC8571519B60222D)
EXTURID(BC8571519B60222D)
QMURID(0000002BDA50)
URTYPE(CICS)
USERID(MQTEST)
APPLTAG(IYRCSQ1)
ASID(0000)
APPLTYPE(CICS)
TRANSID(GP02)
TASKNO(0000096)
END CONN DETAILS
```

For CICS connections, the NID consists of the CICS applid and a unique number provided by CICS at the time the syncpoint log entries are written. This unique number is stored in records written to both the CICS system log and the IBM MQ log at syncpoint processing time. This value is referred to in CICS as the *recovery token*.

2. Scan the CICS log for entries related to a particular unit of recovery.

   Look for a PREPARE record for the task-related installation where the recovery token field (JCSRMTKN) equals the value obtained from the network ID. The network ID is supplied by IBM MQ in the DISPLAY CONN command output.

   The PREPARE record in the CICS log for the units of recovery provides the CICS task number. All other entries on the log for this CICS task can be located using this number.

   You can use the CICS journal print utility DFHJUP when scanning the log. For details of using this program, see the *CICS Operations and Utilities Guide*.

3. Scan the IBM MQ log for records with the NID related to a particular unit of recovery. Then use the URID from this record to obtain the rest of the log records for this unit of recovery.

   When scanning the IBM MQ log, note that the IBM MQ startup message CSQJ001I provides the start RBA for this session.

   The print log records program (CSQ1LOGP) can be used for that purpose.

4. If you need to, do in-doubt resolution in IBM MQ.

   IBM MQ can be directed to take the recovery action for a unit of recovery using an IBM MQ RESOLVE INDOUBT command.

   To recover all threads associated with a specific *connection-name*, use the NID(*) option.

   The command produces one of the following messages showing whether the thread is committed or backed out:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

When performing in-doubt resolution, CICS and the adapter are not aware of the commands to IBM MQ to commit or back out units of recovery, because only IBM MQ resources are affected. However, CICS keeps details about the in-doubt threads that could not be resolved by IBM MQ. This information is purged

either when the list presented is empty, or when the list does not include a unit of recovery of which CICS has details.

z/OS *Recovering IMS units of recovery manually*

Use this topic to understand what happens when the IMS adapter restarts, and then explains how to deal with any unresolved units of recovery that arise.

What happens when the IMS adapter restarts

Whenever the connection to IBM MQ is restarted, either following a queue manager restart or an IMS / START SUBSYS command, IMS initiates the following resynchronization process:

1. IMS presents the list of unit of work (UOW) IDs that it believes are in doubt to the IBM MQ IMS adapter one at a time with a resolution parameter of Commit or Backout.
2. The IMS adapter passes the resolution request to IBM MQ and reports the result back to IMS.
3. Having processed all the IMS resolution requests, the IMS adapter gets from IBM MQ a list of all UOWs that IBM MQ still holds in doubt that were initiated by the IMS system. These are reported to the IMS master terminal in message CSQQ008I.

Note: While a UOW is in doubt, any associated IBM MQ message is locked by IBM MQ and is not available to any application.

How to resolve IMS units of recovery manually

When IMS connects to IBM MQ, IBM MQ might have one, or more in-doubt units of recovery that have not been resolved.

If IBM MQ has in-doubt units of recovery that IMS did not resolve, the following message is issued at the IMS master terminal:

```
CSQQ008I nn units of recovery are still in doubt in queue manager qmgr-name
```

If this message is issued, IMS was either cold-started or it was started with an incomplete log tape. This message can also be issued if IBM MQ or IMS terminates abnormally because of a software error or other subsystem failure.

After receiving the CSQQ008I message:

- The connection remains active.
- IMS applications can still access IBM MQ resources.
- Some IBM MQ resources remain locked out.

If the in-doubt thread is not resolved, IMS message queues can start to build up. If the IMS queues fill to capacity, IMS terminates. You must be aware of this potential difficulty, and you must monitor IMS until the in-doubt units of recovery are fully resolved.

Recovery procedure

Use the following procedure to recover the IMS units of work:

1. Force the IMS log closed, using /SWI OLDS, and then archive the IMS log. Use the utility, DFSERA10, to print the records from the previous IMS log tape. Type X'3730' log records indicate a phase-2 commit request and type X'38' log records indicate an abort request. Record the requested action for the last transaction in each dependent region.
2. Run the DL/I batch job to back out each PSB involved that has not reached a commit point. The process might take some time because transactions are still being processed. It might also lock up

a number of records, which could affect the rest of the processing and the rest of the message queues.

3. Produce a list of the in-doubt units of recovery from IBM MQ using the following command:

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

You receive the following message:

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3E2C5C3F24040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2005-02-15)
UOWLOGTI(16.39.43)
UOWSTDA(2005-02-15)
UOWSTTI(16.39.43)
UOWSTATE(UNRESOLVED)
NID(IM8F  .BC45A794D3810344)
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0000)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

For IMS, the NID consists of the IMS connection name and a unique number provided by IMS. The value is referred to in IMS as the *recovery token*. For more information, see the IMS documentation.

4. Compare the NIDs (IMSID plus OASN in hexadecimal) displayed in the DISPLAY THREAD messages with the OASNs (4 bytes decimal) shown in the DFSERA10 output. Decide whether to commit or back out.
5. Perform in-doubt resolution in IBM MQ with the RESOLVE INDOUBT command, as follows:

```
RESOLVE INDOUBT( connection-name )
ACTION(COMMIT|BACKOUT)
NID( network-id )
```

To recover all threads associated with *connection-name*, use the NID(*) option. The command results in one of the following messages to indicate whether the thread is committed or backed out:

```
CSQV414I  THREAD network-id COMMIT SCHEDULED
CSQV415I  THREAD network-id BACKOUT SCHEDULED
```

When performing in-doubt resolution, IMS and the adapter are not aware of the commands to IBM MQ to commit or back out in-doubt units of recovery because only IBM MQ resources are affected.

z/OS *Recovering RRS units of recovery manually*

Use this topic to understand the how to determine if there are in-doubt RRS units of recovery, and how to manually resolve those units of recovery.

When RRS connects to IBM MQ, IBM MQ might have one, or more in-doubt units of recovery that have not been resolved. If IBM MQ has in-doubt units of recovery that RRS did not resolve, one of the following messages is issued at the z/OS console:

- CSQ3011I
- CSQ3013I

- CSQ3014I
- CSQ3016I

Both IBM MQ and RRS provide tools to display information about in-doubt units of recovery, and techniques for manually resolving them.

In IBM MQ, use the DISPLAY CONN command to display information about in-doubt IBM MQ threads. The output from the command includes RRS unit of recovery IDs for those IBM MQ threads that have RRS as a coordinator. This can be used to determine the outcome of the unit of recovery.

Use the RESOLVE INDOUBT command to resolve the IBM MQ in-doubt thread manually. This command can be used to either commit or back out the unit of recovery after you have determined what the correct decision is.

z/OS *Recovering units of recovery on another queue manager in the queue sharing group*

Use this topic to identify, and manually recover units of recovery on other queue managers in a queue sharing group.

If a queue manager that is a member of a queue sharing group fails and cannot be restarted, other queue managers in the group can perform peer recovery, and take over from it. However, the queue manager might have in-doubt units of recovery that cannot be resolved by peer recovery because the final disposition of that unit of recovery is known only to the failed queue manager. These units of recovery are resolved when the queue manager is eventually restarted, but until then, they remain in doubt.

This means that certain resources (for example, messages) might be locked, making them unavailable to other queue managers in the group. In this situation, you can use the DISPLAY THREAD command to display these units of work on the inactive queue manager. If you want to resolve these units of recovery manually to make the messages available to other queue managers in the group, you can use the RESOLVE INDOUBT command.

When you issue the DISPLAY THREAD command to display units of recovery that are in doubt, you can use the QMNAME keyword to specify the name of the inactive queue manager. For example, if you issue the following command:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

You receive the following messages:

```
CSQV436I +CSQ1 INDOUBT THREADS FOR QM01 -
NAME   THREAD-XREF    URID NID
USER1  00000000000000000000000000 CSQ:0001.0
USER2  00000000000000000000000000 CSQ:0002.0
DISPLAY THREAD REPORT COMPLETE
```

If the queue manager specified is active, IBM MQ does not return information about in-doubt threads, but issues the following message:

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

Use the IBM MQ command RESOLVE INDOUBT to resolve the in-doubt threads manually. Use the QMNAME keyword to specify the name of the inactive queue manager in the command.

This command can be used to commit or back out the unit of recovery. The command resolves the shared portion of the unit of recovery only; any local messages are unaffected and remain locked until the queue manager restarts, or reconnects to CICS, IMS, or RRS batch.

z/OS IBM MQ and IMS

IBM MQ provides two components to interface with IMS, the IBM MQ - IMS adapter, and the IBM MQ - IMS bridge. These components are commonly called the IMS adapter, and the IMS bridge.

z/OS Operating the IMS adapter

Use this topic to understand how to operate the IMS adapter, which connects IBM MQ to IMS systems.

Note: The IMS adapter does not incorporate any operations and control panels.

This topic contains the following sections:

z/OS *Controlling IMS connections*

Use this topic to understand the IMS operator commands which control and monitor the connection to IBM MQ.

IMS provides the following operator commands to control and monitor the connection to IBM MQ:

/CHANGE SUBSYS
Deletes an in-doubt unit of recovery from IMS.

/DISPLAY OASN SUBSYS
Displays outstanding recovery elements.

/DISPLAY SUBSYS
Displays connection status and thread activity.

/START SUBSYS
Connects the IMS control region to a queue manager.

/STOP SUBSYS
Disconnects IMS from a queue manager.

/TRACE
Controls the IMS trace.

For more information about these commands, see the *IMS/ESA® Operator's Reference* manual for the level of IMS that you are using.

IMS command responses are sent to the terminal from which the command was issued. Authorization to issue IMS commands is based on IMS security.

z/OS *Connecting from the IMS control region*

Use this topic to understand the mechanisms available to connect from IMS to IBM MQ.

IMS makes one connection from its control region to each queue manager that uses IMS. IMS must be enabled to make the connection in one of these ways:

- Automatically during either:
  - A cold start initialization.
  - A warm start of IMS, if the IBM MQ connection was active when IMS was shut down.
- In response to the IMS command:

```
/START SUBSYS sysid
```

where *sysid* is the queue manager name.

The command can be issued regardless of whether the queue manager is active.

The connection is not made until the first IBM MQ API call to the queue manager is made. Until that time, the IMS command /DIS SUBSYS shows the status as 'NOT CONN'.

The order in which you start IMS and the queue manager is not significant.

IMS cannot re-enable the connection to the queue manager automatically if the queue manager is stopped with a STOP QMGR command, the IMS command /STOP SUBSYS, or an abnormal end. Therefore, you must make the connection by using the IMS command /START SUBSYS.

If an IMS command is seen in the queue manager console log similar to this:

```
MODIFY IMS*,SS*
```

check the IMS master log and ensure that IBM MQ has RACF authority to issue IMS Adapter MODIFY commands.

Initializing the adapter and connecting to the queue manager

The adapter is a set of modules loaded into the IMS control and dependent regions, using the IMS external Subsystem Attach Facility.

This procedure initializes the adapter and connects to the queue manager:

1. Read the subsystem member (SSM) from IMS.PROCLIB. The SSM chosen is an IMS EXEC parameter. There is one entry in the member for each queue manager to which IMS can connect. Each entry contains control information about an IBM MQ adapter.
2. Load the IMS adapter.

   Note: IMS loads one copy of the adapter modules for each IBM MQ instance that is defined in the SSM member.
3. Attach the external subsystem task for IBM MQ.
4. Run the adapter with the CTL EXEC parameter (IMSID) as the connection name.

The process is the same whether the connection is part of initialization or a result of the IMS command / START SUBSYS.

If the queue manager is active when IMS tries to make the connection, the following messages are sent:

- to the z/OS console:

```
DFS3613I ESS TCB INITIALIZATION COMPLETE
```

- to the IMS master terminal:

```
CSQQ000I IMS/TM imsid connected to queue manager ssnm
```

When IMS tries to make the connection and *the queue manager is not active*, the following messages are sent to the IMS master terminal each time an application makes an MQI call:

```
CSQQ001I IMS/TM imsid not connected to queue manager ssnm.
Notify message accepted
DFS3607I MQM1   SUBSYSTEM ID EXIT FAILURE, FC = 0286, RC = 08,
JOBNAME = IMSEMPR1
```

If you get DFS3607I messages when you start the connection to IMS or on system startup, this indicates that the queue manager is not available. To prevent a large number of messages being generated, you must do one of the following:

1. Start the relevant queue manager.
2. Issue the IMS command:

   ```
   /STOP SUBSYS
   ```

   so that IMS does not expect to connect to the queue manager.

If you do neither, a DFS3607I message and the associated CSQQ001I message are issued each time a job is scheduled in the region and each time a connection request to the queue manager is made by an application.

Thread attachment

In an MPP or IFP region, IMS makes a thread connection when the first application program is scheduled into that region, even if that application program does not make an IBM MQ call. In a BMP region, the thread connection is made when the application makes its first IBM MQ call (`MQCONN` or `MQCONNX`). This thread is retained for the duration of the region or until the connection is stopped.

For both the message driven and non-message driven regions, the recovery thread cross-reference identifier, *`Thread-xref`*, associated with the thread is:

```
PSTid + PSBname
```

where:

PSTid
: Partition specification table region identifier

PSBname
: Program specification block name

You can use connection IDs as unique identifiers in IBM MQ commands, in which case IBM MQ automatically inserts these IDs into any operator message that it generates.

z/OS *Displaying in-doubt units of recovery*

You can display in-doubt of units of recovery and attempt to recover them.

The operational steps used to list and recover in-doubt units of recovery in this topic are for relatively simple cases only. If the queue manager ends abnormally while connected to IMS, IMS might commit or back out work without IBM MQ being aware of it. When the queue manager restarts, that work is termed *in doubt*. A decision must be made about the status of the work.

To display a list of in-doubt units of recovery, issue the command:

```
+CSQ1  DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

IBM MQ responds with a message like the following:

```
CSQM201I +CSQ1 CSQMDRTC DIS CONN DETAILS
CONN(BC0F6125F5A30001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-11-02)
UOWLOGTI(12.27.58)
UOWSTDA(2004-11-02)
UOWSTTI(12.27.58)
UOWSTATE(UNRESOLVED)
NID(CSQ1CHIN.BC0F5F1C86FC0766)
EXTURID(000000000000001F000000007472616E5F6964547565204E6F762020...)
QMURID(000000026232)
URTYPE(XA)
USERID( )
APPLTAG(CSQ1CHIN)
ASID(0000)
APPLTYPE(CHINIT)
CHANNEL( )
CONNAME( )
END CONN DETAILS
```

For an explanation of the attributes in this message, see the description of the DISPLAY CONN command.

Recovering in-doubt units of recovery

To recover in-doubt units of recovery, issue this command:

```
+CSQ1  RESOLVE INDOUBT( connection-name ) ACTION(COMMIT|BACKOUT)
NID( net-node.number )
```

where:

connection-name
The IMS system ID.

ACTION
Indicates whether to commit (COMMIT) or back out (BACKOUT) this unit of recovery.

net-node.number
The associated net-node.number.

When you have issued the RESOLVE INDOUBT command, one of the following messages is displayed:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED

CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

Resolving residual recovery entries

At given times, IMS builds a list of residual recovery entries (RREs). RREs are units of recovery about which IBM MQ might be in doubt. They arise in several situations:

- If the queue manager is not active, IMS has RREs that cannot be resolved until the queue manager is active. These RREs are not a problem.
- If the queue manager is active and connected to IMS, and if IMS backs out the work that IBM MQ has committed, the IMS adapter issues message CSQQ010E. If the data in the two systems must be consistent, there is a problem. For information about resolving this problem, see "Recovering IMS units of recovery manually" on page 547.
- If the queue manager is active and connected to IMS, there might still be RREs even though no messages have informed you of this problem. After the IBM MQ connection to IMS has been established, you can issue the following IMS command to find out if there is a problem:

```
/DISPLAY OASN SUBSYS sysid
```

To purge the RRE, issue one of the following IMS commands:

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

where *nnnn* is the originating application sequence number listed in response to your +CSQ1 DISPLAY command. This is the schedule number of the program instance, giving its place in the sequence of invocations of that program since the last IMS cold start. IMS cannot have two in-doubt units of recovery with the same schedule number.

These commands reset the status of IMS ; they do not result in any communication with IBM MQ.

z/OS *Controlling IMS dependent region connections*

You can control, monitor, and, when necessary, terminate connections between IMS and IBM MQ.

Controlling IMS dependent region connections involves the following activities:

- Connecting from dependent regions
- Region error options
- Monitoring the activity on connections
- Disconnecting from dependent regions

Connecting from dependent regions

The IMS adapter used in the control region is also loaded into dependent regions. A connection is made from each dependent region to IBM MQ. This connection is used to coordinate the commitment of IBM MQ and IMS work. To initialize and make the connection, IMS does the following:

1. Reads the subsystem member (SSM) from IMS.PROCLIB.

   A subsystem member can be specified on the dependent region EXEC parameter. If it is not specified, the control region SSM is used. If the region is never likely to connect to IBM MQ, to avoid loading the adapter, specify a member with no entries.
2. Loads the IBM MQ adapter.

For a batch message program, the load is not done until the application issues its first messaging command. At that time, IMS tries to make the connection.

For a message-processing program region or IMS fast-path region, the attempt is made when the region is initialized.

Region error options

If the queue manager is not active, or if resources are not available when the first messaging command is sent from application programs, the action taken depends on the error option specified on the SSM entry. The options are:

R
The appropriate return code is sent to the application.

Q
The application ends abnormally with abend code U3051. The input message is re-queued.

A
The application ends abnormally with abend code U3047. The input message is discarded.

Monitoring the activity on connections

A thread is established from a dependent region when an application makes its first successful IBM MQ request. You can display information about connections and the applications currently using them by issuing the following command from IBM MQ:

```
+CSQ1 DISPLAY CONN(*) ALL
```

The command produces a message like the following:

```
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-12-15)
UOWLOGTI(16.39.43)
UOWSTDA(2004-12-15)
UOWSTTI(16.39.43)
UOWSTATE(ACTIVE)
NID( )
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0049)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

For the control region, *thread-xref* is the special value CONTROL. For dependent regions, it is the PSTid concatenated with the PSBname. *auth-id* is either the user field from the job card, or the ID from the z/OS started procedures table.

For an explanation of the displayed list, see the description of message CSQV402I in the IBM MQ for z/OS のメッセージ、完了コード、および理由コード documentation.

IMS provides a display command to monitor the connection to IBM MQ. It shows which program is active on each dependent region connection, the LTERM user name, and the control region connection status. The command is:

```
/DISPLAY SUBSYS name
```

The status of the connection between IMS and IBM MQ is shown as one of:

```
CONNECTED
NOT CONNECTED
CONNECT IN PROGRESS
STOPPED
STOP IN PROGRESS
INVALID SUBSYSTEM NAME= name
SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING
```

The thread status from each dependent region is one of the following:

```
CONN
CONN, ACTIVE (includes LTERM of user)
```

Disconnecting from dependent regions

To change values in the SSM member of IMS.PROCLIB, you disconnect a dependent region. To do this, you must:

1. Issue the IMS command:

   ```
   /STOP REGION
   ```

2. Update the SSM member.
3. Issue the IMS command:

   ```
   /START REGION
   ```

z/OS *Disconnecting from IMS*

The connection is ended when either IMS or the queue manager terminates. Alternatively, the IMS master terminal operator can explicitly break the connection.

To terminate the connection between IMS and IBM MQ, use the following IMS command:

```
/STOP SUBSYS sysid
```

The command sends the following message to the terminal that issued it, typically the master terminal operator (MTO):

```
DFS058I STOP COMMAND IN PROGRESS
```

The IMS command:

```
/START SUBSYS sysid
```

is required to reestablish the connection.

Note: The IMS command /STOP SUBSYS is not completed if an IMS trigger monitor is running.

z/OS *Controlling the IMS trigger monitor*

You can use the CSQQTRMN transaction to stop, and start the IMS trigger monitor.

The IMS trigger monitor (the CSQQTRMN transaction) is described in the Setting up the IMS trigger monitor.

To control the IMS trigger monitor see:

- Starting CSQQTRMN
- Stopping CSQQTRMN

Starting CSQQTRMN

1. Start a batch-oriented BMP that runs the program CSQQTRMN for each initiation queue you want to monitor.
2. Modify your batch JCL to add a DDname of CSQQUT1 that points to a data set containing the following information:

```
QMGRNAME=q_manager_name    Comment: queue manager name
INITQUEUENAME=init_q_name   Comment: initiation queue name
LTERM=lterm          Comment: LTERM to remove error messages
CONSOLEMESSAGES=YES      Comment: Send error messages to console
```

   where:

q_manager_name	The name of the queue manager (if this is blank, the default nominated in CSQQDEFV is assumed)
init_q_name	The name of the initiation queue to be monitored
lterm	The IMS LTERM name for the destination of error messages (if this is blank, the default value is MASTER).
CONSOLEMESSAGES=YES	Requests that messages sent to the nominated IMS LTERM are also sent to the z/OS console. If this parameter is omitted or misspelled, the default is NOT to send messages to the console.

3. Add a DD name of CSQQUT2 if you want a printed report of the processing of CSQQUT1 input.

Note:

1. The data set CSQQUT1 is defined with LRECL=80. Other DCB information is taken from the data set. The DCB for data set CSQQUT2 is RECFM=VBA and LRECL=125.

2. You can put only one keyword on each record. The keyword value is delimited by the first blank following the keyword; this means that you can include comments. An asterisk in column 1 means that the whole input record is a comment.
3. If you misspell either of the QMGRNAME or LTERM keywords, CSQQTRMN uses the default for that keyword.
4. Ensure that the subsystem is started in IMS (by the /START SUBSYS command) before submitting the trigger monitor BMP job. If it is not started, your trigger monitor job terminates with abend code U3042.

Stopping CSQQTRMN

Once started, CSQQTRMN runs until either the connection between IBM MQ and IMS is broken due to one of the following events:

- the queue manager ending
- IMS ending

or a z/OS STOP **jobname** command is entered.

z/OS Controlling the IMS bridge

Use this topic to understand the IMS commands that you can use to control the IMS bridge.

There are no IBM MQ commands to control the IBM MQ-IMS bridge. However, you can stop messages being delivered to IMS in the following ways:

- For non-shared queues, by using the ALTER QLOCAL(xxx) GET(DISABLED) command for all bridge queues.
- For clustered queues, by using the SUSPEND QMGR CLUSTER(xxx) command. This is effective only when another queue manager is also hosting the clustered bridge queue.
- For clustered queues, by using the SUSPEND QMGR FACILITY(IMSBRIDGE) command. No further messages are sent to IMS, but the responses for any outstanding transactions are received from IMS.

  To start sending messages to IMS again, issue the RESUME QMGR FACILITY(IMSBRIDGE) command.

You can also use the MQSC command DISPLAY SYSTEM to display whether the bridge is suspended.

See MQSC commands for details of these commands.

For further information see:

- “Starting and stopping the IMS bridge” on page 558
- “Controlling IMS connections” on page 559
- Controlling bridge queues
- “Resynchronizing the IMS bridge” on page 560
- Working with tpipe names
- Deleting messages from IMS
- Deleting tpipes
- “IMS Transaction Expiration” on page 562

Starting and stopping the IMS bridge

Start the IBM MQ bridge by starting OTMA. Either use the IMS command:

```
/START OTMA
```

or start it automatically by specifying OTMA=YES in the IMS system parameters. If OTMA is already started, the bridge starts automatically when queue manager startup has completed. An IBM MQ event message is produced when OTMA is started.

Use the IMS command:

```
/STOP OTMA
```

to stop OTMA communication. When this command is issued, an IBM MQ event message is produced.

Controlling IMS connections

IMS provides these operator commands to control and monitor the connection to IBM MQ:

/DEQUEUE TMEMBER *tmember* TPIPE *tpipe*
Removes messages from a Tpipe. Specify PURGE to remove all messages or PURGE1 to remove the first message only.

/DISPLAY OTMA
Displays summary information about the OTMA server and clients, and client status.

/DISPLAY TMEMBER *name*
Displays information about an OTMA client.

/DISPLAY TRACE TMEMBER *name*
Displays information about what is being traced.

/SECURE OTMA
Sets security options.

/START OTMA
Enables communications through OTMA.

/START TMEMBER *tmember* TPIPE *tpipe*
Starts the named Tpipe.

/STOP OTMA
Stops communications through OTMA.

/STOP TMEMBER *tmember* TPIPE *tpipe*
Stops the named Tpipe.

/TRACE
Controls the IMS trace.

For more information about these commands, see the *IMS/ESA Operators Reference* manual for the level of IMS that you are using.

IMS command responses are sent to the terminal from which the command was issued. Authorization to issue IMS commands is based on IMS security.

Controlling bridge queues

To stop communicating with the queue manager with XCF member name *tmember* through the bridge, issue the following IMS command:

```
/STOP TMEMBER tmember TPIPE ALL
```

To resume communication, issue the following IMS command:

```
/START TMEMBER tmember TPIPE ALL
```

The Tpipes for a queue can be displayed using the MQ DISPLAY QUEUE command.

To stop communication with the queue manager on a single Tpipe, issue the following IMS command:

```
/STOP TMEMBER tmember TPIPE tpipe
```

One or two Tpipes are created for each active bridge queue, so issuing this command stops communication with the IBM MQ queue. To resume communication, use the following IMS command :

```
/START TMEMBER tmember TPIPE tpipe
```

Alternatively, you can alter the attributes of the IBM MQ queue to make it get inhibited.

Resynchronizing the IMS bridge

The IMS bridge is automatically restarted whenever the queue manager, IMS, or OTMA are restarted.

The first task undertaken by the IMS bridge is to resynchronize with IMS. This involves IBM MQ and IMS checking sequence numbers on every synchronized Tpipe. A synchronized Tpipe is used when persistent messages are sent to IMS from an IBM MQ - IMS bridge queue using commit mode zero (commit-then-send).

If the bridge cannot resynchronize with IMS, the IMS sense code is returned in message CSQ2023E and the connection to OTMA is stopped. If the bridge cannot resynchronize with an individual IMS Tpipe, the IMS sense code is returned in message CSQ2025E and the Tpipe is stopped. If a Tpipe has been cold started, the recoverable sequence numbers are automatically reset to 1.

If the bridge discovers mismatched sequence numbers when resynchronizing with a Tpipe, message CSQ2020E is issued. Use the IBM MQ command RESET TPIPE to initiate resynchronization with the IMS Tpipe. You need to provide the XCF group and member name, and the name of the Tpipe; this information is provided by the message.

You can also specify:

- A new recoverable sequence number to be set in the Tpipe for messages sent by IBM MQ, and to be set as the partner's receive sequence number. If you do not specify this, the partner's receive sequence number is set to the current IBM MQ send sequence number.
- A new recoverable sequence number to be set in the Tpipe for messages received by IBM MQ, and to be set as the partner's send sequence number. If you do not specify this, the partner's send sequence number is set to the current IBM MQ receive sequence number.

If there is an unresolved unit of recovery associated with the Tpipe, this is also notified in the message. Use the IBM MQ command RESET TPIPE to specify whether to commit the unit of recovery, or back it out. If you commit the unit of recovery, the batch of messages has already been sent to IMS, and is deleted from the bridge queue. If you back the unit of recovery out, the messages are returned to the bridge queue, to be later sent to IMS.

Commit mode 1 (send-then-commit) Tpipes are not synchronized.

Considerations for Commit mode 1 transactions

In IMS, commit mode 1 (CM1) transactions send their output replies before sync point.

A CM1 transaction might not be able to send its reply, for example because:

- The Tpipe on which the reply is to be sent is stopped
- OTMA is stopped
- The OTMA client (that is, the queue manager) has gone away
- The reply-to queue and dead-letter queue are unavailable

For these reasons, the IMS application sending the message pseudo-abends with code U0119. The IMS transaction and program are not stopped in this case.

These reasons often prevent messages being sent into IMS, as well as replies being delivered from IMS. A U0119 abend can occur if:

- The Tpipe, OTMA, or the queue manager is stopped while the message is in IMS
- IMS replies on a different Tpipe to the incoming message, and that Tpipe is stopped
- IMS replies to a different OTMA client, and that client is unavailable.

Whenever a U0119 abend occurs, both the incoming message to IMS and the reply messages to IBM MQ are lost. If the output of a CM0 transaction cannot be delivered for any of these reasons, it is queued on the Tpipe within IMS.

Working with tpipe names

Many of the commands used to control the IBM MQ - IMS bridge require the *tpipe* name. Use this topic to understand how you can find further details of the tpipe name.

You need *tpipe* names for many of the commands that control the IBM MQ - IMS bridge. You can get the tpipe names from DISPLAY QUEUE command and note the following points:

- tpipe names are assigned when a local queue is defined
- a local queue is given two tpipe names, one for sync and one for non-sync
- tpipe names will not be known to IMS until after some communication between IMS and IBM MQ specific to that particular local queue takes place
- For a tpipe to be available for use by the IBM MQ - IMS bridge its associated queue must be assigned to a Storage Class that has the correct XCF group and member name fields completed

Deleting messages from IMS

A message that is destined for IBM MQ through the IMS bridge can be deleted if the Tmember/Tpipe is stopped. To delete one message for the queue manager with XCF member name *tmember*, issue the following IMS command:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

To delete all the messages on the Tpipe, issue the following IMS command:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

Deleting tpipes

You cannot delete IMS tpipes yourself. They are deleted by IMS at the following times:

- Synchronized tpipes are deleted when IMS is cold started.

- Non-synchronized tpipes are deleted when IMS is restarted.

IMS Transaction Expiration

An expiration time is associated with a transaction; any IBM MQ message can have an expiration time associated with it. The expiration interval is passed from the application, to IBM MQ, using the MQMD.Expiry field. The time is the duration of a message before it expires, expressed as a value in tenths of a second. An attempt to perform the MQGET of a message, later than it has expired, results in the message being removed from the queue and expiry processing performed. The expiration time decreases as a message flows between queue managers on an IBM MQ network. When an IMS message is passed across the IMS bridge to OTMA, the remaining message expiry time is passed to OTMA as a transaction expiration time.

If a transaction has an expiration time specified, OTMA expires the input transactions in three different places in IMS:

- input message receiving from XCF
- input message enqueuing time
- application GU time

No expiration is performed after the GU time.

The transaction EXPRTIME can be provided by:

- IMS transaction definition
- IMS OTMA message header
- IMS DFSINSX0 user exit
- IMS CREATE or UPDATE TRAN commands

IMS indicates that it has expired a transaction by abending a transaction with 0243, and issuing a message. The message issued is either DFS555I in the non-shared-queues environment, or DFS2224I in the shared-queues environment.

z/OS Operating Advanced Message Security on z/OS

The Advanced Message Security address space accepts commands using the z/OS MODIFY command.

Procedure

- Modify Advanced Message Security on z/OS.

  To enter commands for the Advanced Message Security (AMS) address space, use the z/OS MODIFY command.

  For example:

  ```
  F qmgrAMSM, cmd
  ```

  where *qmgr* is the prefix of the started task name.

  The following table describes the MODIFY commands that are accepted:

Table 29. Advanced Message Security address space MODIFY commands

Command	Option	Description
DISPLAY		Display version information

Table 29. Advanced Message Security address space MODIFY commands (continued)		
Command	**Option**	**Description**
REFRESH	KEYRING POLICY ALL	Refresh the key ring certificates, security policies, or both.
SMFAUDIT	SUCCESS FAILURE ALL	Set whether SMF auditing is required when AMS successfully protects or unprotects messages, when AMS fails to protect or unprotect messages, or both.
SMFTYPE	0 - 255	Set the SMF record type to be generated when AMS protects or unprotects messages. To disable SMF auditing specify a record type of 0.

Note: To specify an option it must be separated by a comma. For example:

```
F qmgrAMSM,REFRESH KEYRING
F qmgrAMSM,SMFAUDIT ALL
F qmgrAMSM,SMFTYPE 180
```

- Refresh Advanced Message Security on z/OS.

  Changes that are made effective by issuing the **REFRESH** command apply to applications that issue MQOPEN after the **REFRESH** command has completed. Existing applications that have a queue open, continue to use the options from when the application opened the queue. To use the new values, the application has to close and reopen the queue.
- Start and stop AMS on z/OS.

  You do not need to enter a command to start or stop the Advanced Message Security address space. The AMS address space is started automatically when the queue manager is started if AMS has been enabled with the **SPLCAP** parameter of CSQ6SYSP, and is stopped when the queue manager is stopped.

IBM MQ Internet Pass-Thru の管理

このセクションでは、IBM MQ Internet Pass-Thru (MQIPT) の管理方法について説明します。

コンフィギュレーション IBM MQ Internet Pass-Thru を行うには MQIPT、「コンフィギュレーション.NET」の説明に従って `mqipt.conf` コンフィギュレーションファイルの変更を行います。 MQIPT を再始動することなく構成変更を有効にするために MQIPT をリフレッシュする操作を含め、MQIPT を管理するには、**mqiptAdmin** コマンドを使用します。 **mqiptAdmin** コマンドを使用した MQIPT の管理については、566 ページの『コマンド行を使用した MQIPT の管理』を参照してください。

MQIPT の開始と停止

MQIPT は、コマンド行から開始することも、システムの開始時に自動的に開始させることもできます。**mqiptAdmin** コマンドを使用して、 MQIPT を停止できます。

コマンド行からの MQIPT の開始

MQIPT は以下のようなインストール・ディレクトリーにインストールされています。

- Windows `C:\MQIPT` Windows システムでは、実行可能スクリプトが `C:\MQIPT\bin` にある

- Linux AIX /opt/mqipt AIX and Linux システムでは、実行可能スクリプトが /opt/mqipt/bin にある

MQIPT は、ホーム・ディレクトリーも使用します。ホーム・ディレクトリーには、構成ファイル mqipt.conf と、 MQIPT が実行中に出力されるファイルが含まれています。 MQIPT ホーム・ディレクトリーの以下のサブディレクトリーは、MQIPT の初回起動時に自動的に作成されます。

- First Failure Support Technology (FFST) およびトレース・ファイルが書き込まれる errors ディレクトリー
- 接続ログが保持される logs ディレクトリー

MQIPT を実行するユーザー ID にこれらのディレクトリーを作成する権限があるか、またはディレクトリーが既に存在している必要があり、ユーザー ID にはこれらのディレクトリーのファイルの作成、読み取り、書き込みのための権限が必要です。 また、Java security manager ポリシーを使用している場合は、セキュリティー・ポリシーによってこれらのディレクトリーに必要な権限が付与されている必要があります。 Security Manager ポリシー設定の詳細については、Java security manager を参照してください。

インストール・ディレクトリーをホーム・ディレクトリーとして使用できます。 このディレクトリーを使用する場合、MQIPT を実行するユーザー ID に適切な権限があり、Security Manager ポリシーが正しく構成されていることを必ず確認してください。

MQIPT を開始するには、 MQIPT インストール・ディレクトリーの bin ディレクトリーにある **mqipt** コマンドを使用します。 例えば、以下のコマンドは、ホーム・ディレクトリーとして C:\mqiptHome というディレクトリーを使用する MQIPT のインスタンスを開始します。

```
mqipt C:\mqiptHome
```

mqipt コマンドの詳細については、mqipt (MQIPT の開始) を参照してください。

mqipt コマンドを使用して、開始する MQIPT インスタンスに付ける名前を指定することができます。 MQIPT インスタンスの名前は、コマンド・ポートを使用せずに **mqiptAdmin** コマンドで MQIPT のローカル・インスタンスを管理するために使用されます。 このパラメーターを指定しない場合は、MQIPT のホーム・ディレクトリーの名前が MQIPT インスタンスの名前として使用されます。

コンソール・メッセージに MQIPT の状況が表示されます。 エラーが発生した場合は、IBM MQ Internet Pass-Thru のトラブルシューティングを参照してください。 以下のメッセージは、MQIPT が正常に開始した場合の出力例です。

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.2.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is C:\mqiptHome
MQCPI021 Password checking has been enabled on the command port
MQCPI144 MQ Advanced capabilities not enabled
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1414 is starting and will forward messages to :
MQCPI034 ....examplehost(1414)
MQCPI035 ....using MQ protocols
MQCPI057 ....trace level 5 enabled
MQCPI078 Route 1414 ready for connection requests
```

MQIPT の自動開始

システムの始動時に自動的に開始されるシステム・サービスとして、MQIPT をインストールすることができます。 **mqiptService** コマンドを使用して、 MQIPT サービスをインストールおよびアンインストールします。

- Windows Windows システムでは、 **mqiptService** コマンドは MQIPT を Windows サービスとしてインストールします。
- Linux AIX AIX and Linux システムでは、 **mqiptService** コマンドは、システムのブート時に開始する System V init サービスとして MQIPT をインストールします。 System V init をサポート

しない Linux システムでは、別の方法 (systemd など) を使用して、 MQIPT をサービスとして管理します。

MQIPT サービスが開始されると、すべてのアクティブな MQIPT 経路が開始されます。 サービスが停止すると、すべての経路が即時シャットダウンの対象になります。

システムに MQIPT のインストール済み環境が複数ある場合でも、1 つのシステムにインストールできる MQIPT サービスは 1 つのみです。

mqiptService コマンドの詳細については、mqiptService (MQIPT サービスの管理) を参照してください。

停止 MQIPT

-stop パラメーターを指定した **mqiptAdmin** コマンドを使用して、 MQIPT を停止できます。

例えば、以下のコマンドは、 **mqiptAdmin** コマンドと同じユーザー ID でローカルに実行されている mqipt1 という名前の MQIPT のインスタンスを停止します。

```
mqiptAdmin -stop -n ipt1
```

mqiptAdmin コマンドは、以下のいずれかの方式を使用して、管理する MQIPT のアクティブ・インスタンスに接続します。

- コマンド・ポートを使用せずに MQIPT のローカル・インスタンスに接続することによって
- コマンド・ポートに対してネットワーク接続を行う。

コマンドをコマンド・ポートに送信して MQIPT を停止するために **mqiptAdmin** コマンドを使用する前に、**RemoteShutDown** プロパティーを true に設定してリモート・シャットダウンを有効にする必要があります。

mqiptAdmin コマンドを使用した MQIPT の管理について詳しくは、 566 ページの『コマンド行を使用した MQIPT の管理』を参照してください。

パスワード暗号鍵の指定

MQIPT 構成に、デフォルト鍵以外の暗号鍵を使用して暗号化されたパスワードが含まれている場合は、MQIPT が開始時に読み取ることができるファイルにパスワード暗号鍵を指定する必要があります。

パスワード暗号鍵のファイル

MQIPT で保管および使用するために暗号化するパスワードを、ユーザーが指定した暗号鍵を使用して暗号化することができます。 ユーザーが暗号鍵を指定しない場合は、デフォルトの暗号鍵が使用されます。 パスワード暗号鍵の指定は必須ではありませんが、指定するとセキュリティーが向上します。 独自の暗号鍵を指定しない場合は、デフォルトの暗号鍵が使用されます。

パスワード暗号鍵を指定する場合には、パスワードを暗号化するために使用する **mqiptPW** コマンドおよび MQIPT からアクセスできるファイルに、パスワード暗号鍵を保管しなければなりません。 このファイルの内容に関する制限は、1 文字以上が含まれていること、そしてテキストが 1 行しか含まれていないことだけです。

注 : 必ず、無許可のユーザーが暗号鍵を読み取れないように、パスワード暗号鍵のファイルに適切なファイル許可を設定する必要があります。 パスワード暗号鍵の読み取り権限を必要とするのは、**mqiptPW** コマンドを実行するユーザーと、 MQIPT を実行するユーザーだけです。

MQIPT のインスタンスのために保管されるすべてのパスワードの暗号化と復号に、同じパスワード暗号鍵が使用されます。 そのため、必要なパスワード暗号鍵ファイルは、MQIPT インストール環境ごとに 1 つだけです。

MQIPT インストール環境のパスワード暗号鍵を変更した場合は、すべての暗号化パスワードを、新しい暗号鍵を使用して再暗号化する必要があります。

MQIPT を開始する

パスワード暗号鍵ファイルのデフォルト名は *MQIPT_HOME_DIR*/mqipt_cred.key です。ここで、*MQIPT_HOME_DIR* は、 mqipt.conf 構成ファイルが保管されているディレクトリーです。 自動的に開始されるサービスとして MQIPT を実行する場合は、パスワード暗号鍵ファイルをデフォルトの名前で作成する必要があります。

パスワード暗号鍵ファイルをデフォルト以外の名前で作成した場合は、開始時に MQIPT にそのファイル名を渡す必要があります。 パスワード暗号鍵ファイルの名前は、以下のいずれかの方式で指定できます。優先される順序で記載しています。

1. MQIPT を開始するために使用される **mqipt** コマンドの **-sf** パラメーター。
2. MQS_MQIPTCRED_KEYFILE 環境変数。
3. com.ibm.mq.ipt.cred.keyfile Java プロパティー。

パスワード暗号鍵ファイルの名前を指定しない場合、デフォルトのファイルが存在すれば、そのファイル名が使用されます。 デフォルトのパスワード暗号鍵ファイルが存在しなければ、デフォルトのパスワード暗号鍵が使用されます。

コマンド行を使用した MQIPT の管理

コマンド行で **mqiptAdmin** コマンドを使用して MQIPT を管理できます。

mqiptAdmin コマンドを使用して、以下の管理機能を実行できます。

- MQIPT のアクティブなローカル・インスタンスをリストする。
- 構成ファイルに変更を加えた後に、MQIPT のインスタンスをリフレッシュする。
- MQIPT のインスタンスを停止する。

mqiptAdmin コマンドは、 MQIPT インストール・ディレクトリーの bin サブディレクトリーにあります。

mqiptAdmin コマンドは、以下のいずれかの方式を使用して、管理する MQIPT のアクティブ・インスタンスに接続します。

- コマンド・ポートに対してネットワーク接続を行う。
- コマンド・ポートを使用せずに MQIPT のローカル・インスタンスに接続することによって

mqiptAdmin コマンドは、以前のバージョンの MQIPT と互換性がありますが、このコマンドを使用して、**mqiptAdmin** コマンドのバージョンより高いバージョンの MQIPT を管理することはできません。 複数の異なるバージョンの MQIPT が存在する環境では、最も新しいバージョンの **mqiptAdmin** コマンドを使用する必要があります。

mqiptAdmin コマンドの構文の詳細については、mqiptAdmin (MQIPT の管理) を参照してください。

コマンド・ポートを使用しないローカル管理

MQIPT のローカル・インスタンスは、コマンド・ポートを使用せずに管理できます。 ローカル管理では、管理対象の MQIPT インスタンスと同じシステムで実行されている場合にのみ、 **mqiptAdmin** コマンドを使用して MQIPT を管理できます。

コマンド・ポートを使用せずに MQIPT のローカル・インスタンスを管理する権限を **mqiptAdmin** に付与するには、MQIPT インスタンスが **mqiptAdmin** と同じシステム上で同じユーザー ID の下で実行されている必要があります。 また、AIX and Linux の場合は、**mqiptAdmin** を root として実行することもできます。

ローカル管理は、デフォルトで有効になっています。 ローカル管理を無効にするには、**LocalAdmin** 構成プロパティーを使用します。 **LocalAdmin** プロパティーの詳細については、LocalAdmin を参照してください。

MQIPT のローカル・インスタンスを管理するには、各インスタンスに名前を付ける必要があります。**mqipt** コマンドで MQIPT を開始するときに **-n** パラメーターを使用して、 MQIPT のインスタンスに名前

を割り当てることができます。 MQIPT の開始時に名前を指定しない場合は、MQIPT インスタンスの名前として、ホーム・ディレクトリーの名前が使用されます。 例えば、以下のコマンドは MQIPT を開始し、そのインスタンスに名前 ipt1 を割り当てます。

```
mqipt /opt/mqipt1 -n ipt1
```

インスタンスに名前を割り当てておけば、その名前を **mqiptAdmin** コマンドの **-n** パラメーターに指定して、そのインスタンスを管理することができます。 例えば、次のコマンドは、 ipt1 という名前の MQIPT のローカル・インスタンスを停止します。

```
mqiptAdmin -stop -n ipt1
```

-list パラメーターを指定した **mqiptAdmin** コマンドを使用することにより、コマンド・ポートを使用せずに、 **mqiptAdmin** コマンドが管理を許可されている MQIPT のすべてのローカル・アクティブ・インスタンスをリストできます。 例えば、次のコマンドは、**mqiptAdmin** コマンドを開始したユーザーが管理を許可されている MQIPT のすべてのローカル・アクティブ・インスタンスをリストします。

```
mqiptAdmin -list
```

コマンド・ポートを使用した管理

保護されていない 1 つのコマンド・ポートと TLS で保護された 1 つのコマンド・ポートを使用して MQIPT を構成できます。 管理対象の MQIPT インスタンスと同じシステムのユーザーでも、リモート・システムのユーザーでも、これらのコマンド・ポートを使用して MQIPT を管理できます。

これまでのバージョンの MQIPT は、保護されていないコマンド・ポートに対して発行された管理コマンドのみを受け入れていました。

注 : 保護されていないコマンド・ポートへの接続は暗号化されないので、保護されていないコマンド・ポートにネットワーク経由で送信されたデータは、 MQIPT のアクセス・パスワードを含め、ネットワーク上の他のユーザーに見られる可能性があります。

MQIPT が **mqiptAdmin** コマンドによって発行されたコマンドをコマンド・ポートで listen するためには、mqipt.conf 構成ファイルのグローバル・セクション内の **CommandPort** プロパティーまたは **SSLCommandPort** プロパティーのいずれかに値を指定する必要があります。

いずれかの MQIPT コマンド・ポートを有効にする前に、「その他のセキュリティーに関する考慮事項」でセキュリティーに関する考慮事項を確認してください。 コマンド・ポートで受け取るコマンドに対して認証を有効にすることを検討してください。 コマンド・ポートの認証の詳細については、571 ページの『コマンド・ポートの認証』を参照してください。

コマンド・ポートを使用して MQIPT のインスタンスを管理するには、 **mqiptAdmin** コマンドのパラメーターとして、 MQIPT が実行されているホストのネットワーク・アドレスとコマンド・ポート番号を指定します。 例えば、mqipt.example.com で実行されており、ポート 1890 で listen するように構成された非セキュア・コマンド・ポートを持つ MQIPT インスタンスをリフレッシュするには、以下のコマンドを発行します。

```
mqiptAdmin -refresh -r mqipt.example.com:1890
```

ホスト名とポート番号を指定しない場合、**mqiptAdmin** は、localhost、ポート 1881 への接続を試みます。

TLS コマンド・ポートを使用して MQIPT を管理する方法の詳細については、567 ページの『TLS コマンド・ポートを使用した MQIPT の管理』を参照してください。

TLS コマンド・ポートを使用した MQIPT の管理

MQIPT は、TLS コマンド・ポートを使用して、 **mqiptAdmin** コマンドによって発行された管理コマンドを listen するように構成できます。 TLS コマンド・ポートを使用すると、**mqiptAdmin** と MQIPT の間のネットワーク上の MQIPT アクセス・パスワードなどの機密データが保護されます。 TLS コマンド・ポートを構成し、TLS コマンド・ポートを使用して MQIPT を管理するには、以下の手順を使用します。

このタスクについて

TLS コマンド・ポートは、PKCS #12 鍵ストアまたは PKCS #11 暗号トークン・インターフェースをサポートする暗号ハードウェアのいずれかに保管されているサーバー証明書を使用して構成する必要があります。 そのコマンド・ポートのサーバー証明書が、TLS ハンドシェーク中に **mqiptAdmin** コマンドに送信されます。 このタスクでは、信頼されている認証局 (CA) に対してお客様が新しいサーバー証明書を要求し、ファイル形式の証明書が返されたものと想定しています。 **mqiptAdmin** コマンドは、サーバー証明書に署名した CA の CA 証明書を使用して、コマンド・ポートの証明書を検証します。 CA 証明書は、**mqiptAdmin** コマンドでアクセスできる PKCS #12 鍵ストアに保管する必要があります。

クライアントの証明書認証は、TLS コマンド・ポートではサポートされていません。 コマンド・ポートに発行される管理コマンドに対して認証を有効にするには、571 ページの『コマンド・ポートの認証』を参照してください。

この手順では、 V 9.4.0 V 9.4.0 **mqiptKeytool** コマンドを使用して TLS コマンド・ポートを使用するために必要な鍵ストアおよびデジタル証明書を管理する方法について説明します。 MQIPT が使用する鍵ストアの管理について詳しくは、 MQIPT 鍵ストアの管理を参照してください。

手順

1. 以下の手順に従って、MQIPT のインスタンスに TLS コマンド・ポートを構成します。
   a) 公開鍵と秘密鍵のペア、および関連する TLS コマンド・ポート・サーバー証明書を PKCS #12 鍵ストアに作成します。

   V 9.4.0 V 9.4.0 TLS コマンド・ポート・サーバー証明書を含む鍵ストアを作成するには、次のコマンドを入力します。

   ```
   mqiptKeytool -genkeypair -keystore filename -storetype pkcs12 -storepass password
                -dname distinguished_name -alias label
                -keyalg key_algorithm -keysize key_size -sigalg sig_algorithm
   ```

   ここで、

   -keystore ファイル名
   鍵ストア名を指定します。

   -storepass パスワード
   鍵ストア・パスワードを指定します。

   -alias ラベル
   証明書ラベルを指定します。

   -keyalg *key_algorithm*
   鍵ペアの作成に使用されるアルゴリズムを指定します。

   -keysize キー・サイズ
   鍵のサイズを指定します。

   -sigalg アルゴリズム
   証明書の署名に使用されるアルゴリズムを指定します。

   -dname *distinguished_name*
   二重引用符で囲んだ X.500 識別名を指定します。

   b) CA の署名付きの TLS コマンド・ポートのサーバー証明書を求める証明書要求を作成します。

   V 9.4.0 V 9.4.0 認証要求を作成するには、次のコマンドを入力します。

   ```
   mqiptKeytool -certreq -keystore filename -storetype pkcs12 -storepass password
                -alias label -file certreq_filename
   ```

   ここで、

   -keystore ファイル名
   鍵ストア名を指定します。

-storepass *パスワード*
鍵ストア・パスワードを指定します。

-alias *ラベル*
証明書ラベルを指定します。

-file *certreq_filename*
認証要求のファイル名を指定します。

c) 手順 568 ページの『1.b』 で作成した証明書要求のファイルを CA に送信して署名してもらいます。

d) CA が署名済み証明書を送信した後、署名済み証明書を鍵ストアに受け取ります。

V 9.4.0 V 9.4.0 署名済み証明書を鍵ストアに受け取るには、次のコマンドを入力します。

```
mqiptKeytool -importcert -keystore cert_filename -storetype pkcs12 -storepass password
             -file cert_filename
```

ここで、 *cert_filename* は証明書を含むファイルの名前、 *filename* は鍵ストアの名前、 *password* は鍵ストアのパスワードです。

e) **mqiptPW** コマンドを使用して、鍵ストア・パスワードを暗号化します。

以下のコマンドを入力します。

```
mqiptPW -sf encryption_key_file
```

ここで、*encryption_key_file* は、ご使用の MQIPT インストールのパスワード暗号鍵を含むファイルの名前です。 MQIPT インストール済み環境でデフォルトのパスワード暗号鍵を使用している場合は、 **-sf** パラメーターを指定する必要はありません。 プロンプトが出されたら、暗号化する鍵ストア・パスワードを入力します。

mqiptPW コマンドについて詳しくは、鍵リング・パスワードの暗号化を参照してください。

f) `mqipt.conf` 構成ファイルを編集して、 TLS コマンド・ポートを構成するための以下のプロパティーを指定します。

i) **SSLCommandPort** プロパティーの値を TLS コマンド・ポート番号に設定します。

ii) **SSLCommandPortKeyRing** プロパティーの値を、ステップ 568 ページの『1.a』で作成した鍵ストアのファイル名に設定します。

iii) **SSLCommandPortKeyRingPW** の値として、手順 569 ページの『1.e』 で **mqiptPW** コマンドから出力された文字列を設定します。

iv) **SSLCommandPortSiteLabel** プロパティーの値を、ステップ 568 ページの『1.b』 で認証要求を作成するときに指定した TLS コマンド・ポート証明書のラベル名に設定します。

v) TLS コマンド・ポートへのインバウンド接続を特定のネットワーク・インターフェースからのものに制限する場合は、 **SSLCommandPortListenerAddress** プロパティーの値を、MQIPT が実行されているシステム上のいずれかのネットワーク・インターフェースに属するネットワーク・アドレスに設定します。 例えば、TLS コマンド・ポートへのインバウンド接続をローカル・マシンからの接続のみに制限するには、 **SSLCommandPortListenerAddress** プロパティーの値を `localhost` に設定します。

g) TLS コマンド・ポートを有効にするために、MQIPT を開始またはリフレッシュします。

MQIPT から、有効になっている TLS コマンド・ポートの構成を示す以下のようなコンソール・メッセージが出力されます。

```
MQCPI155 Listening for control commands on port 1882 on local address * using TLS
MQCPI139 ......secure socket protocols <NULL>
MQCPI031 ......cipher suites <NULL>
MQCPI032 ......key ring file c:\\iptHome\\ssl\\commandport.p12
MQCPI072 ......and certificate label mqiptadmin
```

2. **mqiptAdmin** コマンドを使用して MQIPT を管理するシステムで、以下のステップを実行して、 **mqiptAdmin** が TLS コマンド・ポートに接続できるようにします。

a) TLS コマンド・ポート証明書に署名した CA の CA 証明書を PKCS #12 鍵ストアにインポートし、 **mqiptAdmin** コマンドでトラストストアとして使用できるようにします。

V 9.4.0 V 9.4.0 CA 証明書をインポートするには、次のコマンドを入力します。

```
mqiptKeytool -importcert -keystore filename -storetype pkcs12 -storepass password
             -file cert_filename -alias certlabel
```

ここで、

filename
作成する鍵ストアの名前を指定します。

パスワード
鍵ストア・パスワードを指定します。

certlabel
CA 証明書に付与するラベルを指定します

cert_filename
CA 証明書を含むファイルの名前を指定します。

b) **mqiptPW** コマンドを使用して、鍵ストア・パスワードを暗号化します。

以下のコマンドを入力します。

```
mqiptPW -sf encryption_key_file
```

ここで、*encryption_key_file* は、パスワード暗号鍵を含むファイルの名前です。 このパスワード暗号鍵ファイルは、 MQIPT 構成のパスワードの暗号化に使用するファイルとは異なるファイルにすることができます。 **-sf** パラメーターを使用して暗号鍵ファイルを指定しない場合は、デフォルトのパスワード暗号鍵が使用されます。 プロンプトが出されたら、暗号化する鍵ストア・パスワードを入力します。

mqiptPW コマンドについて詳しくは、鍵リング・パスワードの暗号化を参照してください。

c) **mqiptAdmin** コマンドで使用するプロパティー・ファイルを作成し、以下のプロパティーを指定します。

```
SSLClientCAKeyRing=key_ring_file_name
SSLClientCAKeyRingPW=key_ring_password
PasswordProtectionKeyFile=encryption_key_file
```

ここで、

key_ring_file_name
ステップ 569 ページの『2.a』で作成した鍵ストアの名前です。

key_ring_password
ステップ 570 ページの『2.b』で **mqiptPW** コマンドによって出力された暗号化パスワードです。

encryption_key_file
パスワード暗号鍵が含まれているファイルの名前です。 **PasswordProtectionKeyFile** プロパティーを指定する必要があるのは、ステップ 570 ページの『2.b』で鍵ストア・パスワードの暗号化に暗号鍵ファイルが使用された場合のみです。

d) **mqiptAdmin** コマンドを発行して MQIPT を管理します。その際、 **-s** パラメーターを指定して TLS 接続が必要であることを示し、 **-p** パラメーターを指定してステップ 570 ページの『2.c』で作成したプロパティー・ファイルの名前を指定します。

例えば、リフレッシュ・コマンドを TLS コマンド・ポートに送信して MQIPT のインスタンスをリフレッシュするには、次のコマンドを入力します。

```
mqiptAdmin -refresh -r hostname:port -s -p properties_file
```

mqiptAdmin コマンドから、MQIPT への接続が TLS で保護されることを確認するための以下のようなメッセージが出力されます。

```
MQCAI109 The connection to MQIPT is secured with TLSv1.2.
```

次のタスク

TLS コマンド・ポートで受信するコマンドに対して認証を有効にするには、571 ページの『コマンド・ポートの認証』の手順に従ってください。

コマンド・ポートの認証

保護されていないコマンド・ポートおよび TLS コマンド・ポートで受信したコマンドを、パスワードを使用して認証するように、MQIPT を構成することができます。 コマンド・ポートの認証を有効にするには、次の手順を使用します。

このタスクについて

mqiptAdmin コマンドは、コマンド・ポートの認証が有効になっている MQIPT のインスタンスのコマンド・ポートに接続するときに、ユーザーに対してパスワードの入力を求めるプロンプトを出します。 MQIPT は、**mqiptAdmin** コマンドで入力されたパスワードを、MQIPT 構成に指定されているアクセス・パスワードと照合します。

コマンド・ポートの認証について設定したプロパティーは、TLS コマンド・ポートおよび保護されていないコマンド・ポートの両方に適用されます。

手順

1. **mqiptPW** コマンドを使用して、 MQIPT アクセス・パスワードを暗号化します。

   以下のコマンドを入力します。

   ```
   mqiptPW -sf encryption_key_file
   ```

   ここで、*encryption_key_file* は、ご使用の MQIPT インストールのパスワード暗号鍵を含むファイルの名前です。 MQIPT インストール済み環境でデフォルトのパスワード暗号鍵を使用している場合は、 **-sf** パラメーターを指定する必要はありません。 プロンプトが表示されたら、暗号化するアクセス・パスワードを入力します。

   MQIPT 構成のパスワードの暗号化方法について詳しくは、保管されるパスワードの暗号化を参照してください。
2. mqipt.conf 構成ファイルを編集して、以下のプロパティーを指定します。

   ```
   AccessPW=encrypted_password
   RemoteCommandAuthentication=auth_setting
   ```

   ここで、

   encrypted_password
   手順 571 ページの『1』 で **mqiptPW** コマンドから出力された、暗号化されたパスワードです。

   auth_setting
   認証の要件です。 このプロパティーに以下のいずれかの値を設定すると、コマンド・ポートの認証が有効になります。

   オプション
   パスワードは必須ではありませんが、指定した場合は、有効なパスワードでなければなりません。 このオプションは、例えばマイグレーションの際に役立つ場合があります。

   required
   コマンド・ポートでコマンドを受信するたびに、有効なパスワードを指定する必要があります。

   これらのプロパティーについて詳しくは、MQIPT グローバル・プロパティーを参照してください。
3. 変更を有効にするために、MQIPT を開始またはリフレッシュします。

   MQIPT から、コマンド・ポートの認証が有効かどうかを示すメッセージが出力されます。 例えば、**mqiptAdmin** コマンドが実行されるたびに有効なパスワードの入力を要求するように MQIPT が構成されている場合、以下のメッセージが発行されます。

   ```
   MQCPI021 Password checking has been enabled on the command port
   ```

バックアップの作成

通常のバックアップ手順の一環としてバックアップする必要がある MQIPT ファイルがいくつかあります。

以下のファイルを定期的にバックアップします。

- 構成ファイル、 mqipt.conf
- mqipt.conf 内の以下のプロパティーによって指定される SSL/TLS 鍵リング・ファイル。
  - **SSLClientKeyRing**
  - **SSLClientCAKeyRing**
  - **SSLServerKeyRing**
  - **SSLServerCAKeyRing**
  - **SSLCommandPortKeyRing**
- mqipt.conf 内の以下のプロパティーによって指定される SSL/TLS 鍵リング・パスワード・ファイル。
  - **SSLClientKeyRingPW**
  - **SSLClientCAKeyRingPW**
  - **SSLServerKeyRingPW**
  - **SSLServerCAKeyRingPW**
- MQIPT の構成に、デフォルト鍵以外の暗号鍵で暗号化されたパスワードが含まれている場合は、そのパスワード暗号鍵のファイル。
- **SecurityManagerPolicy** プロパティーが設定されている場合に、そのプロパティーで指定されるポリシー・ファイル。
- mqipt.conf 内の以下のプロパティーによって指定されるセキュリティー出口ファイルおよび証明書出口ファイル。
  - **SecurityExitName**
  - **SSLExitName**
- MQIPT ホーム・ディレクトリーの log サブディレクトリーにある接続ログ・ファイル (監査のためにこれらのファイルが必要な場合) 。

パフォーマンスの調整

スレッド・プールとアイドル・タイムアウト指定の組み合わせを使用して、各 MQIPT 経路の相対パフォーマンスを調整できます。

接続スレッド

各 MQIPT 経路 には、着信コミュニケーション要求を処理する同時実行スレッドの作業プールが割り当てられています。 初期化時に、スレッドのプールが作成され (サイズは経路の MinConnectionThreads 属性で指定)、最初の着信要求を処理するためのスレッドが割り当てられます。 この要求を受信すると、別のスレッドが割り当てられて次の着信要求に備えます。 すべてのスレッドが作業用に割り当てられたら、新しいスレッドが作成され、作業プールに追加されて作業に割り当てられます。

この方法で、プールはスレッドが最大数 (**MaxConnectionThreads** で指定) に達するまで増大します。 会話が終了するか指定したアイドル・タイムアウト期間が経過すると、スレッドは解放され、プールに戻されます。 作業スレッドが最大数に達すると、次の受信要求はスレッドが解放されて作業プールに戻されるまで待機します。

使用可能なスレッドの数を増加させることで、 要求が待機する必要がある時間を短縮できます。 ただし、この増加は使用可能なシステム・リソースとバランスを取る必要があります。

アイドル・タイムアウト

デフォルトでは、作業スレッドが非アクティブであるために終了されることはありません。 スレッドが会話に割り当てられると、そのスレッドは会話の正常終了、経路の非アクティブ化、または MQIPT のシャットダウンが行われるまで割り当てが 維持されます。 オプションで、アイドル・タイムアウト間隔 (分単位) を **IdleTimeout** プロパティーで設定して、指定された期間非アクティブであったスレッドがリサイクルされるようにすることができます。 スレッドは作業プールに戻され、 リサイクルして使用されます。

IBM MQ のアクティビティーが断続的である場合、このハートビート間隔を MQIPT タイムアウトの値より短い時間に設定して、スレッドが頻繁にリサイクルされないようにすることができます。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

IBM 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。 本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510
東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
日本アイ・ビー・エム株式会社
法務・知的財産
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
〒 103-8510
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 INTERNATIONAL BUSINESS MACHINES CORPORATION は、法律上の瑕疵担保責任、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 "" 国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。 それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他の ライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で 決定されたものです。 そのため、他の操作環境で得られた結果は、異なる可能性があります。 一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。 さらに、一部の測定値が、推定値である可能性があります。 実際の結果は、異なる可能性があります。 お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、 もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。 したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、 名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報 (提供されている場合) は、このプログラムで使用するアプリケーション・ソフトウェアの作成を支援することを目的としています。

本書には、プログラムを作成するユーザーが IBM MQ のサービスを使用できるようにするためのプログラミング・インターフェースに関する情報が記載されています。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。 診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

重要 : この診断、修正、およびチューニング情報は、変更される可能性があるため、プログラミング・インターフェースとして使用しないでください。

商標

IBM、 IBM ロゴ、 ibm.com®は、世界の多くの国で登録された IBM Corporation の商標です。 現時点での IBM の商標リストについては、 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml をご覧ください。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds 氏の米国およびその他の国における登録商標です。

この製品には、 Eclipse Project (https://www.eclipse.org/) により開発されたソフトウェアが含まれています。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

部品番号:

(1P) P/N: