9.2

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、545 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 2、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、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通常の権利を IBM に付与します。

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。

詳しくは、121 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。

詳しくは、185 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』
- 9 ページの『プログラマブル・コマンド・フォーマット (PCF)』
- 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』
- 10 ページの『IBM MQ Console』
- Windows 10 ページの『Microsoft Cluster Service (MSCS)』

z/OS IBM MQ for z/OS® で使用できる管理インターフェースとオプションについては、415 ページの『IBM MQ for z/OS の管理』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、23 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 キュー・マネージャーおよび関連リソースを管理するには、いくつかの異なる方法があります。

IBM MQ の制御コマンド

ALW

制御コマンドを使用して、キュー・マネージャーそのものに対する管理タスクを実行できます。

IBM MQ for AIX®, Linux®, and Windows システムでは、システムのコマンド行で実行する制御コマンド が用意されています。

制御コマンドの説明については、Multiplatforms でのキュー・マネージャーの作成と管理を参照してください。 制御コマンドのコマンド・リファレンスについては、 IBM MQ 制御コマンド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。

キュー・マネージャーへの MQSC コマンドの発行には、**runmqsc** コマンドを使用します。 この場合、キーボードからコマンドを発行することによって対話式に実行するか、または ASCII テキスト・ファイルの一連のコマンドを実行するよう標準入力装置 (stdin) をリダイレクトします。 いずれの場合も、コマンドの形式は同じです。

コマンドに設定したフラグによって、**runmqsc** コマンドを次の 3 とおりのモードで実行できます。

- 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。

MQSC コマンドは、すべてのプラットフォームの、 IBM i を含む、および z/OS で使用可能です。 MQSC コマンドについては、コマンド・セットの比較に要約されています。

ALW AIX, Linux, and Windows では、システム・コマンド・ラインで MQSC を単一コマンドとして実行できます。 複雑なコマンドや複数のコマンドを実行する場合は、コマンド・ラインから実行するファイルとして MQSC を作成できます。 MQSC コマンドをリモート・キュー・マネージャーに送信することも可能です。 詳細については、17 ページの『テキスト・ファイルからの MQSC コマンドの実行』 を参照してください。

IBM i IBM i サーバーでコマンドを実行するために、コマンドのリストを組み込んだスクリプト・ファイルを作成し、STRMQMMQSC コマンドを使用してそのファイルを実行します。

注 : IBM i

1. QTEMP ライブラリーの使用は制限されているため、QTEMP ライブラリーを STRMQMMQSC の入力ライブラリーとして使用しないでください。 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。
2. IBM i では、スクリプト・ファイルから実行されるコマンドへの MQSC 応答がスプール・ファイルとして返されます。

MQSC コマンドの使用法の詳細については、11 ページの『MQSC コマンドによる管理』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、24 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。

詳しくは、administrative REST API を参照してください。

IBM i の制御言語 (CL)

IBM i

この言語を使用して、IBM MQ for IBM i に対する管理コマンドを実行できます。 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 の制御言語 (CL) の詳細については、 IBM MQ for IBM i CL コマンド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 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 を開始できます。

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。

詳しくは、112 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 を使用すると、z/OS を含む他のプラットフォーム上にあるリモート・キュー・マネージャーを管理できます。

IBM MQ Explorer は、製品インストールの一部としてインストールすることも (IBM MQ のインストールおよびアンインストールを参照)、Fix Central から入手できるスタンドアロン IBM MQ Explorer ダウンロードからインストールすることもできます (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)。

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。

詳しくは、88 ページの『Web コンソールを使用した管理』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。

ALW 制御コマンドによる IBM MQ の管理

制御コマンドは、AIX, Linux, and Windows 上でいくつかの IBM MQ 管理タスクを実行する方法を提供します。

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。

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。

制御コマンドの完全なリストについては、IBM MQ 制御コマンドを参照してください。

Windows システムでの制御コマンドの使用

Windows

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。

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。

AIX and Linux システムでの制御コマンドの使用

Linux AIX

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。

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。

関連資料

IBM MQ 制御コマンド・リファレンス

MQSC コマンドによる管理

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

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、キュー・マネージャーへの MQSC コマンドの発行には、**runmqsc** コマンドを使用します。 **runmqsc** コマンドは、以下のいくつかの方法で実行できます。
  - キーボードから対話式にコマンドを発行する。 13 ページの『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。 17 ページの『テキスト・ファイルからの MQSC コマンドの実行』を参照してください。
  - リモート・キュー・マネージャーで。 185 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。
- 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、20 ページの『z/OS 上で MQSC コマンドを発行できるソース』を参照してください。

MQSC コマンドについては、MQSC コマンドのセクションで詳しく説明されています。

手順

- 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。
- 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)。
- 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。
- 以下のいずれかが該当しなければ、ブランク、小文字、または特殊文字を含んだストリングは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 (.)
    - 順方向斜線 (/)
    - 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。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。

- 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。
- 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 ( )
```

- 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。
- 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。
- 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、22 ページの『総称値および特別な意味を持つ文字』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 コマンドの対話式実行

コマンド・ウィンドウまたはシェルを使用して、対話式で MQSC コマンドを使用できます。

始める前に

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。 15 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 UNIX and Linux プラットフォーム上で MQSC コマンドを対話式に実行する場合、**runmqsc** コマンド行で、コマンド再呼び出し、コマンド完了、および Emacs コマンド・キーがサポートされます。 16 ページの『AIX および Linux での runmqsc コマンド』を参照してください。

このタスクについて

手順

1. 対話式で MQSC コマンドを使用するには、コマンド・ウィンドウまたはシェルをオープンして、次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ブランクでない文字に続くことを示します。

   コマンドの入力は、連結文字でなくブランクでもない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
   ```

   また、使用しているオペレーティング・システムの EOF 文字を使用しても終了できます。

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、標準出力装置に送られます。 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 コマンドを参照してください。

関連タスク

17 ページの『テキスト・ファイルからの MQSC コマンドの実行』

MQSC コマンドを対話式に実行することはクイック・テストに適していますが、非常に長いコマンドがある場合や、特定の一連のコマンドを繰り返し使用する場合は、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

runmqsc

MQSC コマンド・プロンプトの設定

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。

手順

- **MQPROMPT** 環境変数を任意のプロンプトに設定します。

  このプロンプトは、 **runmqsc** コンソール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、176 ページの『サービス定義での置き換え可能な挿入』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 システムでは、MQ_FILE_SEP は / です。 – Windows Windows システムでは、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 システムでは、MQ_PATH_SEP は : です。 – Windows Windows システムでは、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、挿入が展開されたときの文字数で最大 256 文字に制限されます。 MQPROMPT を展開するとこの値を超える場合には、展開が行われずに MQPROMPT のストリング全体が切り捨てられます。

  例えば、プロンプトを MQSC に設定するには、以下のいずれかのコマンドを入力します。

  - Windows

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

- Linux AIX

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

例

以下の例は、 AIX システムでの **MQPROMPT** 変数の設定を示しています。 ユーザー名 (関連付けられたシステム環境変数から得られた値)、キュー・マネージャー名、および IBM MQ ホスト名 (MQ の置き換え可能な挿入から得られた値) を表示するようにプロンプトが設定されます。

```
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 AIX および Linux での runmqsc コマンド

AIX および Linux の **runmqsc** コマンド行は、コマンド再呼び出し、コマンド完了、および Emacs コマンド・キーをサポートします。

以下のコマンド行エディター機能を使用できます。

- 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** にコマンド行エディター機能がないので、**runmqsc** コマンド行が開始されると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 では、curses をインストールします。
- Linux では、ncurses をインストールします。

AIX での ncurses または curses のインストール

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 キーのバインディングのカスタマイズ

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・キーの無効化

環境変数を設定することによって、コマンド再呼び出し、コマンド完了、および Emacs コマンド・キーを無効にできます。 環境変数 **MQ_OVERRIDE_LIBEDIT_LOAD** を TRUE に設定します。

runmqsc で以下の通知メッセージが表示された場合に、この環境変数を回避策として使用できます。

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

テキスト・ファイルからの MQSC コマンドの実行

MQSC コマンドを対話式に実行することはクイック・テストに適していますが、非常に長いコマンドがある場合や、特定の一連のコマンドを繰り返し使用する場合は、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。

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。 stdin は、システムへの入力が行われる装置です。 通常、これはキーボードですが、入力をシリアル・ポートやディスク・ファイルなどに指定することができます。

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。 stdout は、システムからの出力の送信先の装置です。 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。

以下を使用する場合に、MQSC コマンドからスクリプトを作成する必要が生じることがあります。

- 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
- IBM i IBM i では **STRMQM** コマンドを使用します。

- ALW AIX, Linux, and Windows では **runmqsc** コマンドを使用します。

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、15 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。

手順

1. 実行する MQSC コマンドを収めたテキスト・ファイルを作成します。
   - 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。
   - 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に続くことを意味します。 + 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、24 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。
   - MQSC コマンド構文については、MQSC コマンドを参照してください。
   - テキスト・ファイルの作成に役立てるため、サンプル MQSC コマンド・ファイルを使用することができます。

     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

     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。

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。

2. ローカル・キュー・マネージャーで、コマンド構文が正しいことをコマンドを実行せずに検証します。 **runmqsc** コマンドで **-v** フラグを使用します。

   V 9.2.0

   - IBM MQ 9.2.0 以降では、 **-f** オプションを使用して、入力テキスト・ファイル名を識別します。 以下に例を示します。

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

   - IBM MQ 9.2.0 の前の Long Term Support リリース、および IBM MQ 9.1.4 の前の Continuous Delivery リリースでは、 < 演算子を使用して、入力テキスト・ファイルからコマンドに MQSC コマンドを送信します。 以下に例を示します。

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

   戻されるレポートは、20 ページの図 2 に示されているものと同様です。

   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。

3. 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。

   V 9.2.0

   - IBM MQ 9.2.0 以降では、(例えば) 以下のコマンドを実行します。

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

   - IBM MQ 9.2.0 より前の Long Term Support リリース、および IBM MQ 9.1.4 より前の Continuous Delivery リリースの場合は、以下のいずれかのコマンドを使用します。

     – < 演算子はテキスト・ファイルからの入力をダイレクトします。 例えば、以下のコマンドは、テキスト・ファイル `myprog.in` に含まれている一連のコマンドを実行します。

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

     – > 演算子は出力をテキスト・ファイルにダイレクトします。 例えば、以下のコマンドは、テキスト・ファイル `myprog.in` に含まれる一連のコマンドを実行し、それを `results.out` というファイルに出力します。

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

   20 ページの図 1 は、MQSC コマンド・ファイル `myprog.in` および 20 ページの図 2 からの抽出で、`results.out` の出力に対応する抽出です。

例

MQSC コマンドは、ユーザーが理解できる形式、つまり ASCII テキストで書きます。 以下の例は、MQSC コマンド・ファイルからの抜粋で、MQSC コマンド **DEFINE QLOCAL** を示しています。

```
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 *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

関連タスク

15 ページの『MQSC コマンド・プロンプトの設定』
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。

13 ページの『MQSC コマンドの対話式実行』
コマンド・ウィンドウまたはシェルを使用して、対話式で MQSC コマンドを使用できます。

関連資料

runmqsc

z/OS z/OS 上で MQSC コマンドを発行できるソース

MQSC 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。

コマンドは次のソースから発行できます。

- z/OS コンソールまたは同等のコンソール
- 初期設定入力データ・セット CSQINP1、CSQINP2、CSQINPT、および CSQINPX
- CSQUTIL バッチ・ユーティリティー
- 適切な権限が付与されたアプリケーション (コマンドをメッセージとして SYSTEM.COMMAND.INPUT キューに送信する)

z/OS 詳細については、415 ページの『IBM MQ for z/OS へのコマンドの実行』を参照してください。

ただし、これらすべてのソースからすべてのコマンドを発行できるわけではありません。 コマンドは、その発行元に従って、次のように分類できます。

1
CSQINP1

2
CSQINP2

C
z/OS コンソール

R
コマンド・サーバーおよびコマンド・キュー (CSQUTIL、CSQINPT、CSQINPX、またはアプリケーションによる)

MQSC コマンドの説明では、これらの発行元をそれぞれ 1、2、C、R の文字で表しています。

V 9.2.0 Multi 開始時の MQSC スクリプトからの自動構成

IBM MQ 9.2.0 以降では、キュー・マネージャーが開始されるたびに、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。

開始前に

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

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・マネージャーの開始時に、AutoConfig スタンザ属性の **MQSCConfig** によって識別される構成が **runmqsc** 検証を介して渡されて、有効な構文であることが確認されます。その後、キュー・マネージャーのデータ・ツリーの 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。

V 9.2.2 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が接続するために使用可能になります。 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 autocfgmqsc.LOG 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。

総称値および特別な意味を持つ文字

MQSC コマンドで使用する場合、バックスラッシュ（\）やダブルクォート（"）などの一部の文字が特別な意味を持ちます。 パラメーターに使用できる特殊文字の中には、総称値を使用することもできますが、正しく指定することも

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。

表 1. 特別な意味を持つ文字の説明 (続き)	
文字	**説明**
'	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。

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、24 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、以下によってサポートされています。

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

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。

- 25 ページの『PCF コマンド・メッセージ』
- 28 ページの『IBM MQ の PCF 応答』
- z/OS 30 ページの『拡張応答』
- IBM MQ オブジェクトの命名規則
- 31 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

Priority
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、z/OS を除くすべてのプラットフォームでサポートされます。 旧アプリケーションは、z/OS の PCF をサポートしません。 IBM MQ Windows エクスプローラーは、こ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 の PCF をサポートします)。

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 および他のいくつかのプラットフォームでサポートされます。 z/OS 上で発行されたコマンドは、拡張応答のみを生成します。 他のプラットフォームでは、 どちらのタイプの応答も生成できます。

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。

Priority
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。

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、z/OS を除くすべてのプラットフォームでサポートされます。

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 拡張応答

z/OS 上で発行されたコマンドは、拡張応答のみを生成します。

拡張応答には、以下の 3 つのタイプがあります。

- タイプ MQCFT_XR_MSG のメッセージ応答
- タイプ MQCFT_XR_ITEM の項目応答
- タイプ MQCFT_XR_SUMMARY の要約応答

各コマンドは、1 つ以上の応答セットを生成することができます。 各応答セットは、1 つ以上のメッセージで構成され、各メッセージの PCF ヘッダーの *MsgSeqNumber* フィールドには、 1 から始まる番号が順次割り振られます。 各セットの最後の (または唯一の) 応答 の *Control* フィールドの値は、MQCFC_LAST です。 セット内のそれ以外のすべての応答に関しては、この値は MQCFC_NOT_LAST です。

応答には、*Parameter* フィールドが MQBACF_RESPONSE_SET に設定されていて、値が応答セット ID である、 オプションの MQCFBS 構造体が 1 つ以上含まれている可能性があります。 この ID は固有で、応答の属する応答セットを識別するものです。 応答セットごとに、その応答セットを識別する MQCFBS 構造体が あります。

拡張応答には、少なくとも以下の 2 つのパラメーター構造体があります。

- *Parameter* フィールドが MQBACF_RESPONSE_ID に設定された MQCFBS 構造体。 このフィールドの値は、 応答が属する応答セットの ID です。 最初のセットの ID は、 任意です。 後続のセットの ID は、MQBACF_RESPONSE_SET 構造体で先に通知されている ID です。
- *Parameter* フィールドが MQCACF_RESPONSE_Q_MGR_NAME に設定されていて、値が、応答セットを出力したキュー・マネージャーの名前である MQCFST 構造体。

多くの応答には、追加のパラメーター構造体があります。これらの構造体については、 以下の各セクションで説明します。

1 つのセットに含まれている応答の数を事前に調べることはできません。 MQCFC_LAST の応答が検出されるまで応答を取り出していく以外に方法はありません。 また、応答セットの数を事前に調べることもできません。 これは、どのセットにも、追加のセットが生成されたことを示す MQBACF_RESPONSE_SET 構造体が含まれている可能性があるからです。

照会コマンドに対する拡張応答

照会コマンドは、 通常、指定された検索条件と一致することが検出された項目ごとに、 項目応答 (タイプ MQCFT_XR_ITEM) を生成します。 この項目応答のヘッダーの *CompCode* フィールドの値は MQCC_OK であり、*Reason* フィールドの値は MQRC_NONE です。 これには、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、 項目および要求された属性についての説明を含む、他のパラメーター構造体も入っています。

項目にエラーがある場合、ヘッダーの *CompCode* フィールドの値は MQCC_FAILED となり、*Reason* フィールドにはその特定のエラーが 示されます。 組み込まれる追加のパラメーター構造体によって、その項目が示されます。

特定の照会コマンドは、 項目応答に加えて、汎用 (名前に固有ではない) メッセージ応答を返すことができます。 これらの応答は、タイプ MQCFT_XR_MSG の通知応答またはエラー応答です。

照会コマンドが成功した場合、 オプションで、*CompCode* 値が MQCC_OK で、 *Reason* フィールド値が MQRC_NONE である 要約応答 (タイプ MQCFT_XR_SUMMARY) を返すことができます。

照会コマンドが失敗した場合、 項目応答が返された後に、オプションで、*CompCode* 値が MQCC_FAILED で、 *Reason* フィールド値が MQRCCF_COMMAND_FAILED である 要約応答 (タイプ MQCFT_XR_SUMMARY) を返すことができます。

照会以外のコマンドに対する拡張応答

コマンドが成功すると、メッセージ応答が生成されます。 このヘッダーの *CompCode* フィールドの値は MQCC_OK であり、 *Reason* フィールドの値は MQRC_NONE です。 少なくとも 1 つのメッセージ (通常、通知 (MQCFT_XR_MSG) または要約 (MQCFT_XR_SUMMARY) メッセージ) が必ず存在します。 オプションで、追加の通知 (タイプ MQCFT_XR_MSG) メッセージを 返すことができます。 各通知メッセージには、コマンドに関する情報が入った 追加のパラメーター構造体がいくつか含まれている可能性があります。含まれる可能性がある 構造体については、個々のコマンドの説明を参照してください。

コマンドが失敗すると、エラー・メッセージ応答 (タイプ MQCFT_XR_MSG) が生成されます。 この応答のヘッダーの *CompCode* フィールドの値は MQCC_FAILED で、 *Reason* フィールドにはその特定のエラーが示されます。 各メッセージには、エラーに関する情報が入った 追加のパラメーター構造体がいくつか含まれている可能性があります。含まれる可能性がある 構造体については、個々のエラーの説明を参照してください。 通知メッセージ応答を生成することができます。 オプションで、*CompCode* 値が MQCC_FAILED で、 *Reason* フィールド値が MQRCCF_COMMAND_FAILED である 要約応答 (タイプ MQCFT_XR_SUMMARY) を返すことができます。

CommandScope を使用するコマンドに対する拡張応答

コマンドで **CommandScope** パラメーターを使用したり、コマンドによって **CommandScope** パラメーターを使用するコマンドが生成されたりする場合は、 そのコマンドを受け取ったキュー・マネージャーからの初期応答セットがあります。 そして、コマンドの送信先のキュー・マネージャーごとに、(個別に複数のコマンドが発行された場合のように) 別個の応答セット (複数の場合あり) が生成されます。 最後に、受信側キュー・マネージャーからの応答セットがあります。 これには、包括的な要約応答 (タイプ MQCFT_XR_SUMMARY) が含まれます。 MQCACF_RESPONSE_Q_MGR_NAME パラメーター構造体は、 各セットを生成したキュー・マネージャーを示します。

初期応答セットには、以下の追加のパラメーター構造体が 入っています。

- MQIACF_COMMAND_INFO (MQCFIN)。 この構造体に入っている可能性がある値は、MQCMDI_CMDSCOPE_ACCEPTED または MQCMDI_CMDSCOPE_GENERATED です。
- MQIACF_CMDSCOPE_Q_MGR_COUNT (MQCFIN)。 この構造体は、コマンドが送信されるキュー・マネージャーの数を示します。

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、33 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

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、33 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
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

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 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
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

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)**
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
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

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)**
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、37 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

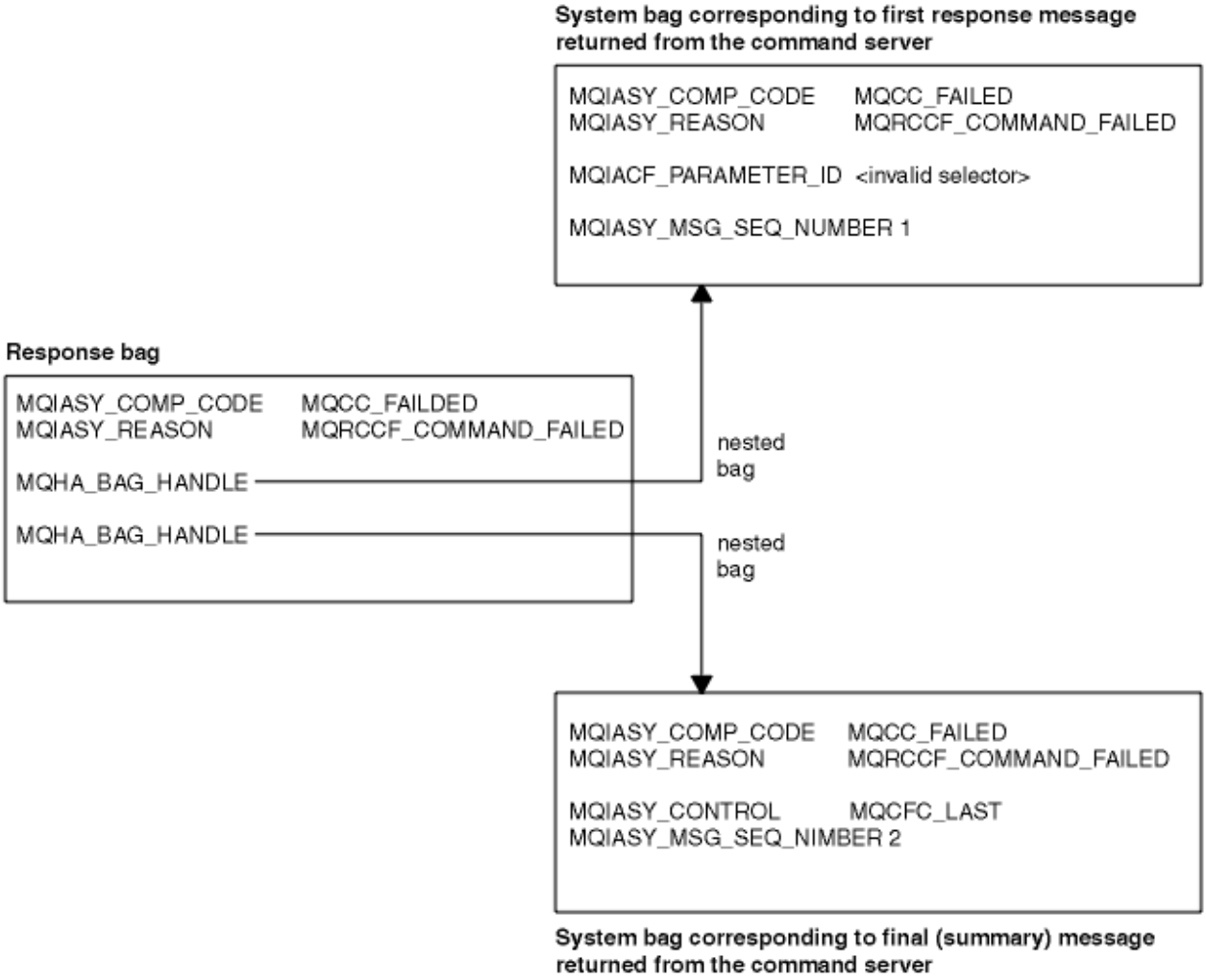
Multi MQAI で IBM MQ を構成するためのヒント

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。
- ストリングの属性値をクリアするには、空ストリングにするのではなく、単一ブランクに設定します。
- 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、71 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、39 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。

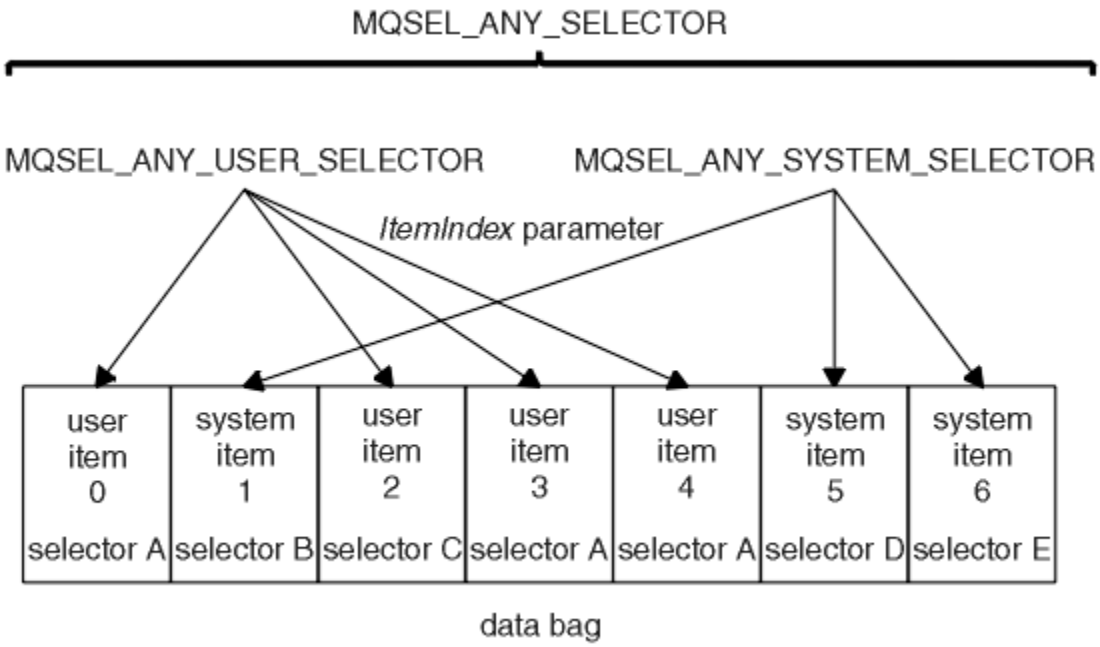


図 *3.* 索引付け

39 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、67 ページの『バッグ内の情報の変更』および 70 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。

40 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
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

193 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。

195 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、41 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
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、41 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、41 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	41 ページの表 5 を参照
Expiry	30 秒 (注 42 ページの『1』)
Feedback	MQFB_NONE

表 6. メッセージ記述子値 (続き)	
パラメーター	**値**
Encoding	MQENC_NATIVE
CodedCharSetId	バッグ CCSID により異なる (注 42 ページの『2』)
Format	41 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 42 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. 39 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、64 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、38 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。

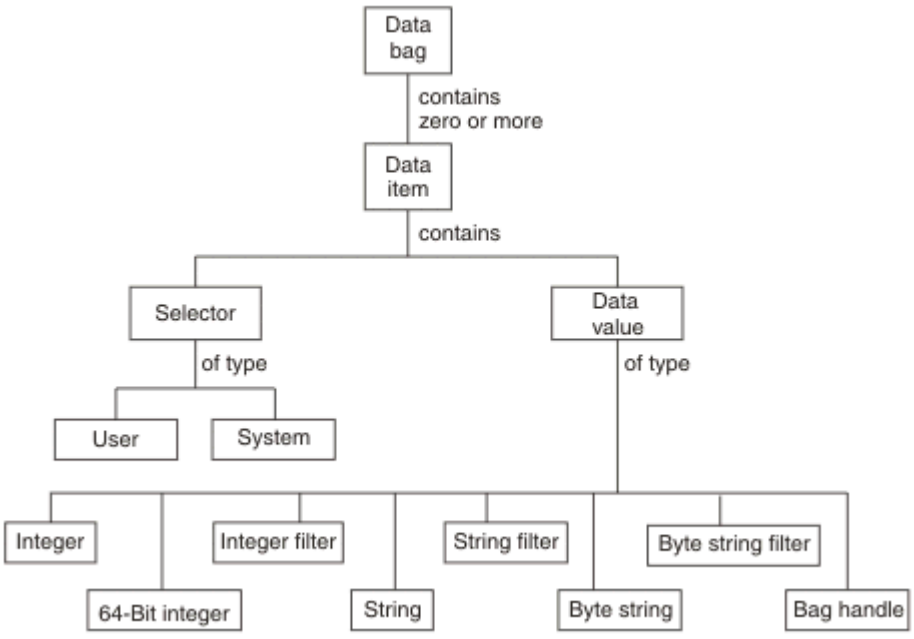


図 *4. MQAI* 概念の階層

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。 63 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、63 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: 63 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: 64 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: 65 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: 66 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: 67 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: 69 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: 67 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: 68 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: 69 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: 69 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、64 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、71 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、27 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、65 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。

- 69 ページの『データ項目のカウント』.
- 70 ページの『データ項目の削除』.
- 65 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』.
- 66 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 (67 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。 (70 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、64 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 66 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。

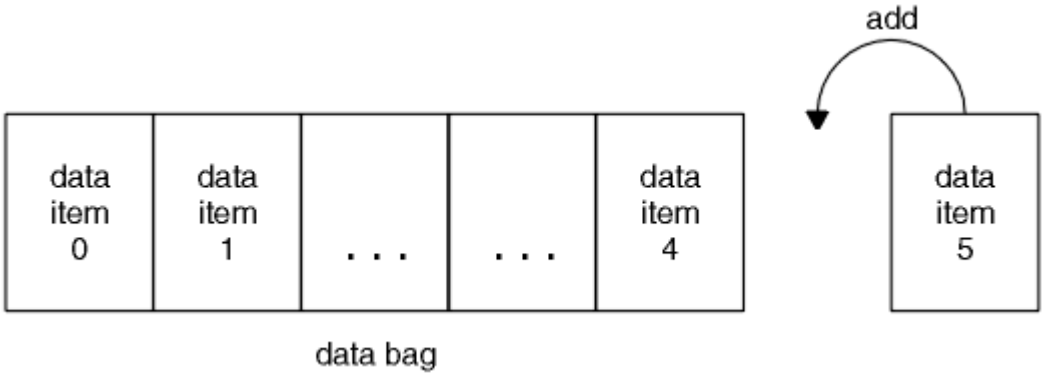


図 5. データ項目の追加

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 65 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、24 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 (68 ページの図 6 を参照)。

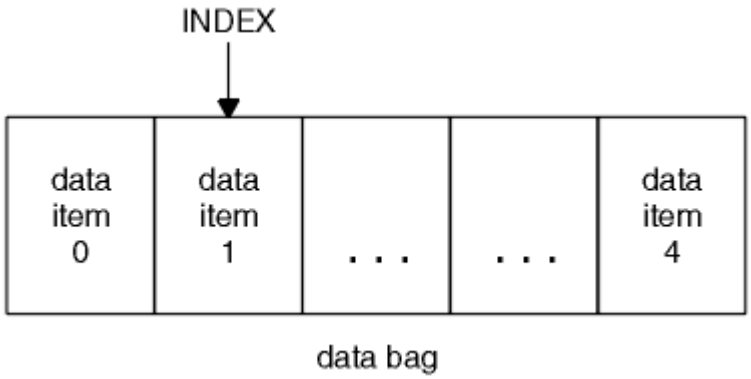


図 6. 単一データ項目の変更

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。 (68 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。

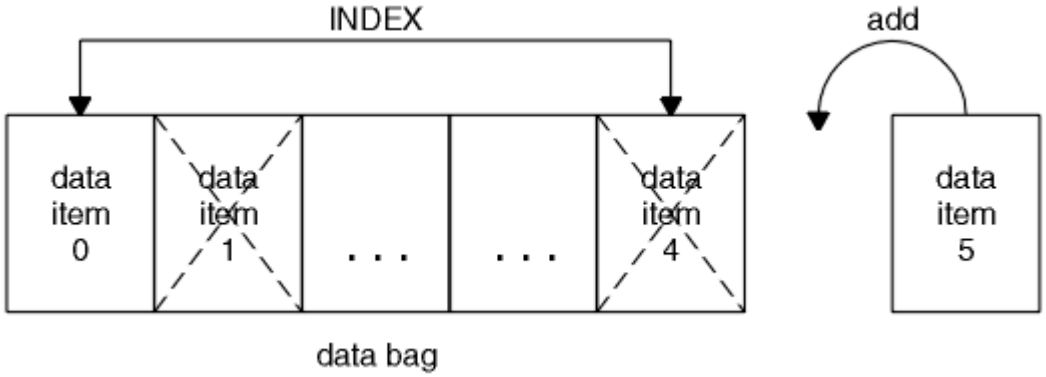


図 7. すべてのデータ項目の変更

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。

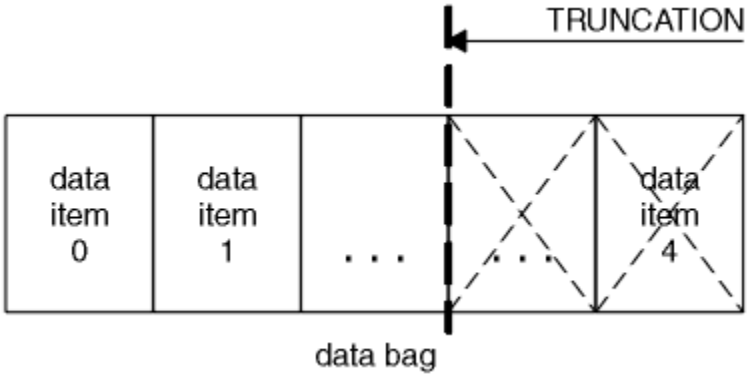


図 *8.* バッグの切り捨て

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 69 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。

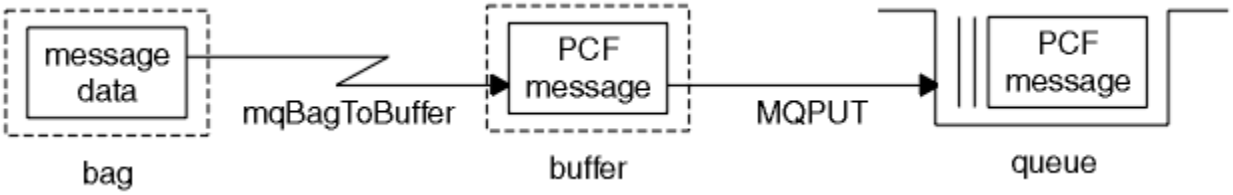


図 *9.* バッグの *PCF* メッセージへの変換

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 69 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。

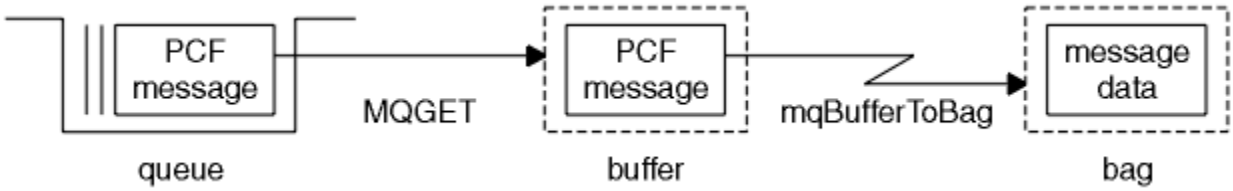


図 *10. PCF* メッセージのバッグ形式への変換

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( バッグ、 7、 ...)`は、バッグ内の項目の数を、セレクターが 7 の場合に戻します。 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。 詳細については、70 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、68 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。 詳細については、69 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。 (70 ページの図 11 を参照してください。)

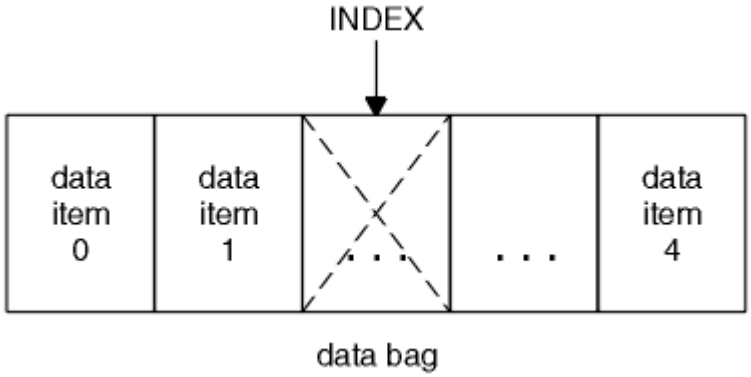


図 *11.* 単一データ項目の削除

または

2. 指定されたセレクターのすべてのオカレンス。 (70 ページの図 12 を参照してください。)

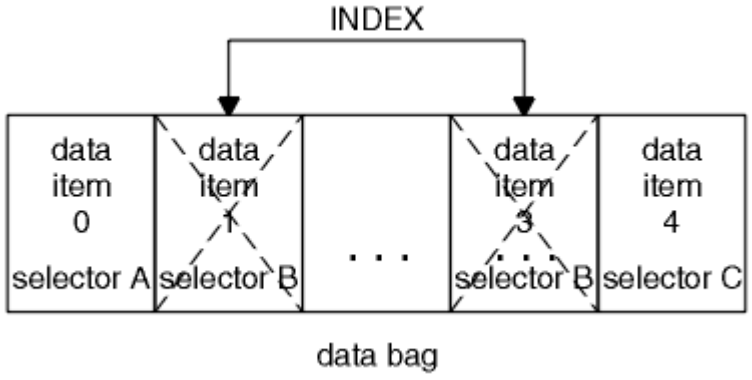


図 *12.* すべてのデータ項目の削除

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。

71 ページの図 13 にネスト の概念を示します。

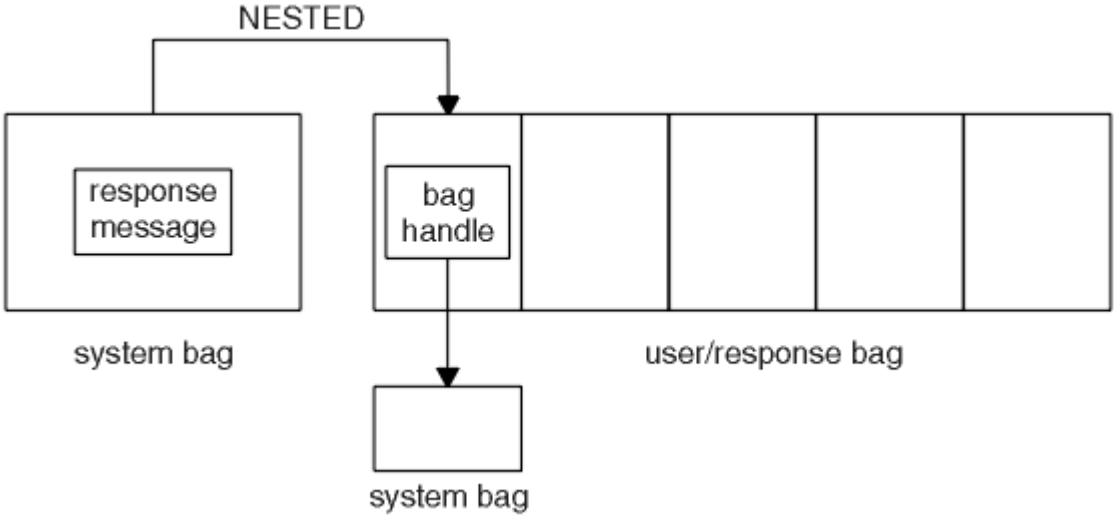


図 *13.* ネスト

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。 これがデフォルトです。

- 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

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

- 72 ページの『administrative REST API の使用開始』
- 76 ページの『administrative REST API の使用』
- 77 ページの『REST API によるリモート管理』
- 81 ページの『REST API タイム・スタンプ』
- 81 ページの『REST API エラー処理』
- 84 ページの『REST API ディスカバリー』
- 85 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 サーバーを構成します。

   基本レジストリーを使用した mqweb サーバーの基本構成の作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. 次のコマンドを入力して、 REST API URL を確認します。

   ```
   dspmqweb status
   ```

   以下のステップの例では、 REST API URL がデフォルト URL であることを前提としています 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 73 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、76 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、77 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、74 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、88 ページの『Web コンソールを使用した管理』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 サーバーを構成します。基本レジストリーを使用した mqweb サーバーの基本構成の作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. 次のコマンドを入力して、 REST API URL を確認します。

   ```
   dspmqweb status
   ```

   以下のステップの例では、 REST API URL がデフォルト URL であることを前提としています `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。 サーバーが再起動されたかどうかは、`console.log` and `messages.log` ファイルを見るか、ファイルを見るか、あるいは z/OS, スタートしたタスクの出力を見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、76 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、88 ページの『Web コンソールを使用した管理』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。 これらの認証方式を使用する方法について詳しくは、IBM MQ コンソールと 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 76 ページの『1.a』 と 76 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/queue
- /admin/subscription
- /admin/channel
- /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/agent
- /admin/transfer
- /admin/monitor

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、72 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、186 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。

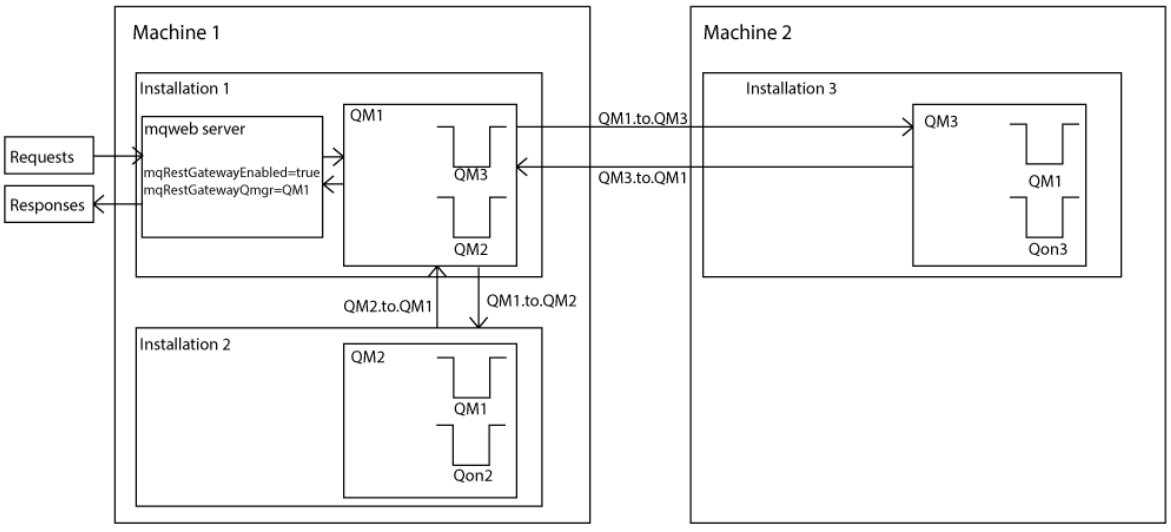


図 *14. REST API* を使用したリモート管理のための構成例の図。

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。

explanation
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。

exceptions
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 の例外の配列。 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。

V 9.2.0 IBM MQ 9.1.2 以降、このフィールドは返されなくなりました。

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。 mqweb サーバーで API ディスカバリー機能を有効にすると、REST API の Swagger 資料を表示できます。

始める前に

重要 : API Discovery フィーチャーは安定化されました。このフィーチャー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。

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。

86 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` 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。

V 9.2.0 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/
```

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。

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。

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。

Web コンソールを使用した管理

基本的な管理タスクは、Web コンソールを使用して実行できます。

V 9.2.0 IBM MQ 9.2.0 以降では、New Web Console と呼ばれる新しい Web コンソールを使用できます (90 ページの『新規 Web Console のクイック・ツアー』を参照)。

必要であれば、Multiplatforms でも Dashboard Web Console の使用を継続できます (111 ページの『コンソール・タイプの切り替え』を参照)。

注 : Web コンソールを使用する際は、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。 キュー・マネージャーでコマンド・サーバーが無効になっていると、Web コンソールは反応しなくなり、コマンドの処理に長時間の遅延が発生します。 コマンド・サーバーが無効になっているキュー・マネージャーに発行されるコマンドはすべて、タイムアウトになります。

関連タスク

V 9.2.0 新規 Web Console のトレース

Dashboard Web Console のトレース

V 9.2.0 Web コンソールの概要

Web コンソールの使用をすぐに開始します。

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。

手順

1. Web コンソールで使用するように 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 サーバーの基本構成の作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、Web コンソールの 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/
   ```

   Web コンソールの URI の末尾は、`console/` という接尾部になります。
4. 89 ページの『3』 で得られた URL をブラウザーで入力して、Web コンソールに接続します。

   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。 Web コンソールに進むことを選択してください。
5. Web コンソールにログインします。 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、Web コンソールはトークン・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 Web コンソール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。

z/OS z/OS での制約事項

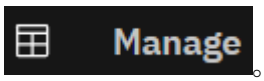
z/OS で IBM MQ Console を使用してキュー・マネージャーを管理する場合、以下の制約事項が適用されます。

- 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。
- 新規オブジェクトは、QSGDISP(QMGR) によってのみ作成できます。
- 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。

関連概念

88 ページの『Web コンソールを使用した管理』
基本的な管理タスクは、Web コンソールを使用して実行できます。

関連タスク

94 ページの『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 

。

Multi IBM MQ for Multiplatforms での制約事項

IBM MQ for Multiplatforms で IBM MQ Console を使用してキュー・マネージャーを管理する場合、以下の制約事項が適用されます。

- 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。

関連概念

88 ページの『Web コンソールを使用した管理』
基本的な管理タスクは、Web コンソールを使用して実行できます。

関連タスク

94 ページの『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 

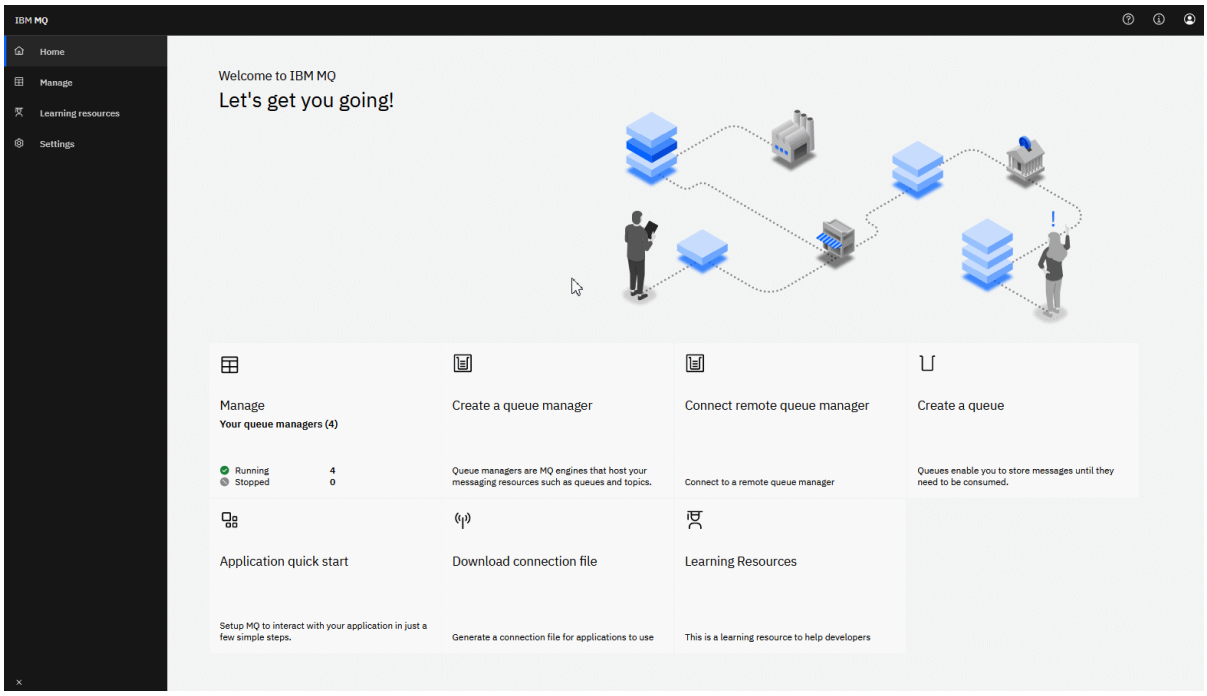
。

V 9.2.0 新規 Web Console のクイック・ツアー

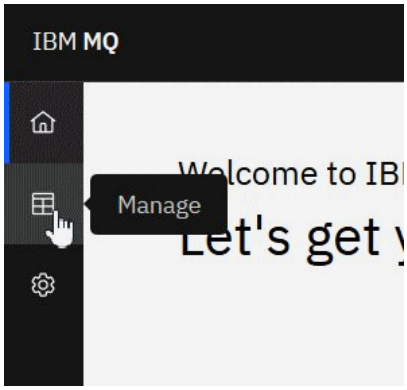
クイック・ツアーを利用すると、New Web Console をすぐに使用することができます。 以下のトピックでは、その使用法について詳しく説明します。

New Web Console がお勧めする Web UI ですが、既存のコンソール (Dashboard Web Console) を引き続き使用したい場合は、そのコンソールに戻すこともできます (111 ページの『コンソール・タイプの切り替え』を参照)。

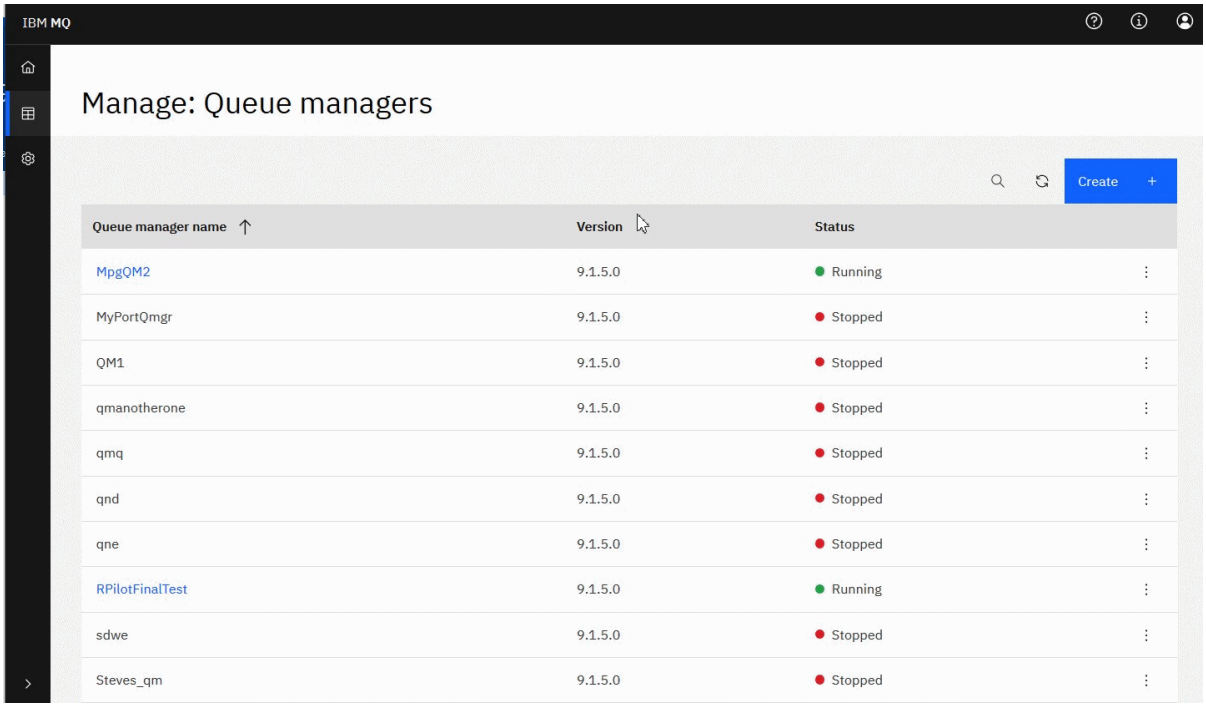
New Web Console に初めてログインすると、ランディング・ページが表示されます。 ここから、既存のキュー・マネージャーを管理するか、キュー・マネージャーまたはキューを作成するか、いくつかの教育トピックにナビゲートするか、IBM Documentation で IBM MQ 製品情報を開くかを選択できます。 また、アプリケーションのクイック・スタートを起動して、新規/既存のキュー・マネージャーやアプリケーションの間でメッセージングを短時間で簡単にセットアップするプロセスを示すガイドを利用することもできます。



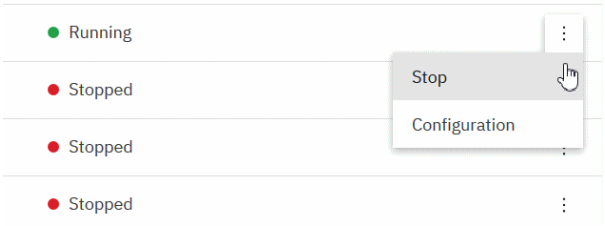
別の方法として、管理アイコンをクリックして、すぐに IBM MQ オブジェクトの管理を始めることもできます。



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。 新しいキュー・マネージャーを作成して、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。

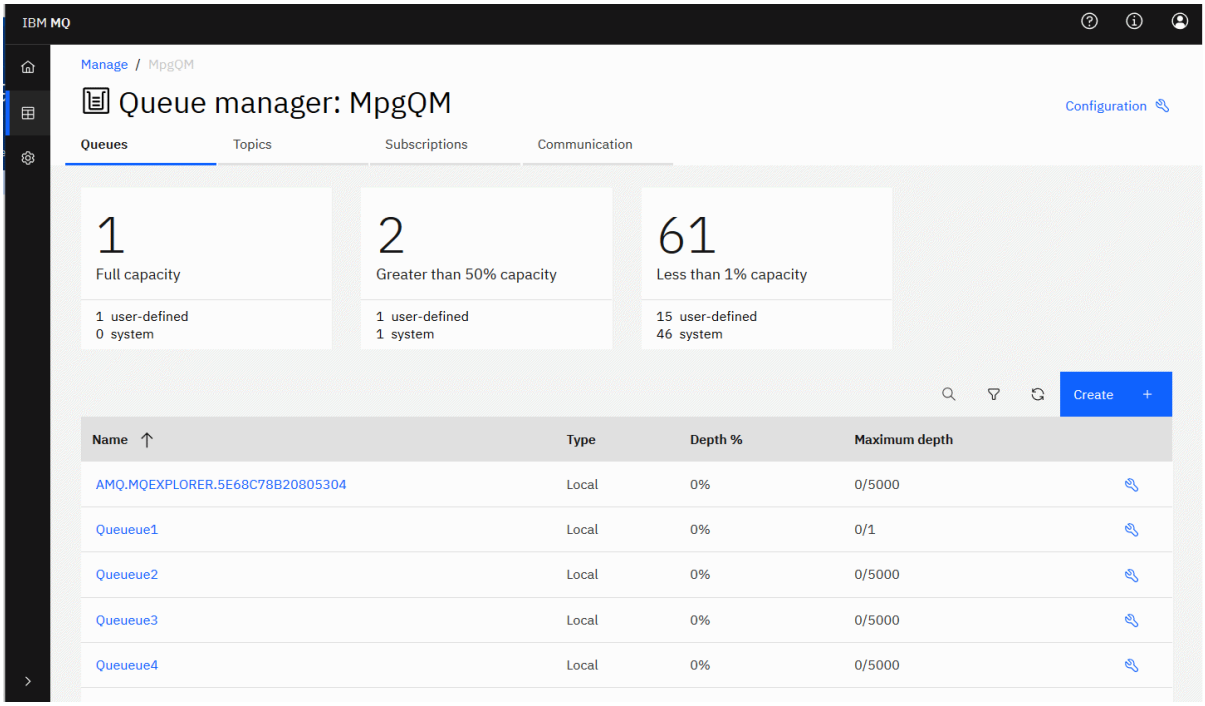


キュー・マネージャーごとにメニューがあり、実行中のキュー・マネージャーの停止または構成、停止したキュー・マネージャーの開始または削除を行えます。



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。

「**キュー**」タブ:

- 新規キューを作成する
- 既存キューを構成する
- キュー名をクリックして既存のメッセージを表示し、新規メッセージを作成する

「**トピック**」タブ:

- 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**」タブ:

- 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

「**通信 (Communications)**」タブ:

- リスナー:
  - リスナーを開始、停止、および構成する
  - 新規リスナーを作成する
- 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
  - チャネルをリセットする (「**拡張**」メニュー項目)
  - チャネルの未確定メッセージを解決する (「**拡張**」メニュー項目)

- 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
  - チャネルをリセットする (「**拡張**」メニュー項目)
  - チャネルの未確定メッセージを解決する (「**拡張**」メニュー項目)

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

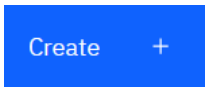
Multi 「管理」ビューには、以下に追加されたローカル・キュー・マネージャーがリストされます。IBM MQ Console が実行されている IBM MQ インストール済み環境。 同じシステム上にあるが、異なる IBM MQ インストール済み環境に関連付けられているキュー・マネージャーは、リストされません。

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、MQ Console が実行されているシステム上で定義されます。 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、複製されたデータ・キュー・マネージャー (RDQM) をサポートしていません。

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**」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**」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**」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。
  - 95 ページの『認証情報オブジェクトの操作』
  - 96 ページの『キュー・マネージャーの権限レコードの操作』
  - 97 ページの『チャネル認証レコードの操作』

V 9.2.0 認証情報オブジェクトの操作

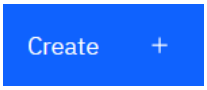
コンソールを使用して、キュー・マネージャーの認証情報オブジェクトを追加および削除することができます。 プロパティーを表示および設定したり、オブジェクトの権限レコードを管理したりすることもできます。

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**」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。

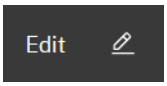
  a) 削除する認証情報オブジェクトのスパナのアイコン をリストから選択します。

  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内容を保存します。

- 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。

キュー・マネージャーの権限レコードの操作

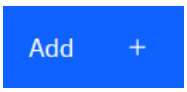
ユーザーおよびグループに対して権限レコードを指定すると、そのユーザーおよびグループがキュー・マネージャーに対して持つアクセス権限を制御できます。

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、一般的な権限を制御する「権限レコード」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とグループを制御する「作成権限レコード」という 2 つのタイプがあります。

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**」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) 権限レコードのリスト・ビューで「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 (その名前が権限レコードの名前として使用されます)。
  d) 付与する権限を選択します (権限について詳しくは、「 -- 」を参照してください)
  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) 権限レコードの作成リスト・ビューで、「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 (その名前が権限レコードの名前として使用されます)。
  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のメニュー ⋮ を開き、「**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) 削除する権限レコードのメニュー ⋮ を開き、「**編集**」を選択します。
  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 9.2.0 チャネル認証レコードの操作

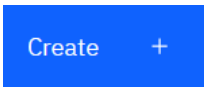
IBM MQ Console を使用して、キュー・マネージャー上のチャネル認証レコードを追加および削除することができます。 チャネル認証レコードのプロパティーを表示および設定することもできます。

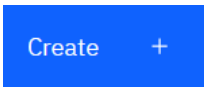
このタスクについて

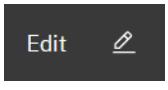
チャネル認証レコードを使用すれば、接続システムに与えるアクセス権限をチャネル・レベルでさらに細かく制御できるようになります。

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**」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。 「**使用可能なすべてのオプションを表示 (Show all available options)**」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」ボタン 

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内容を保存します。

V 9.2.3 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・システムにキュー・マネージャーを準備して、リモートで管理できるようにする必要があります。100 ページの『コマンド・ラインを使用してリモート・キュー・マネージャーを MQ Console に接続する』のステップ 100 ページの『1』 を参照してください。

また、IBM MQ Console からのリモート接続を使用する方法を制御する構成ファイルをセットアップする必要もあります。 構成ファイルは、**setmqweb** コマンドで **remote** パラメーターを使用することによって作成します (リモート・キュー・マネージャーの接続動作の構成と setmqweb を参照)。 構成ファイルを直接編集することはできません。

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 (100 ページの『コマンド・ラインを使用してリモート・キュー・マネージャーを MQ Console に接続する』のステップ 101 ページの『2』 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。100 ページの『コマンド・ラインを使用してリモート・キュー・マネージャーを MQ Console に接続する』を参照してください。

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 9.2.3 コマンド・ラインを使用してリモート・キュー・マネージャーを *MQ Console* に接続する

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 MQ Console に接続できます。 リモート・キュー・マネージャーは、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

- mqweb サーバーが 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. ローカル・キュー・マネージャー QML を構成して、リモート接続を受け入れます。

   a) キュー・マネージャーをリモート管理するためのサーバー接続チャネルを作成します。

   MQ Console を使用してサーバー接続チャネルを作成することも、コマンド・ラインで **DEFINE CHANNEL** MQSC コマンドを使用することもできます。

   例えば、リモート・キュー・マネージャー QM1 用のサーバー接続チャネル QM1.SVRCONN を作成するには、以下のコマンドを入力します。

```
runmqsc QM1
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。

b) 適切なユーザー ID によるこのチャネルへのアクセスが、サーバー接続チャネルで許可されていることを確認してください。 このユーザー ID は、MQ Console がリモート接続を行うシステム上で mqweb サーバーを開始するために使用されるユーザー ID でなければなりません。

MQ Console を使用して適切な権限レコードを作成することも、コマンド・ラインで **SET CHLAUTH** MQSC コマンドを使用することもできます。

例えば、リモート・キュー・マネージャー QM1 の QM1.SVRCONN にアクセスする権限をユーザー exampleUser に付与するには、次のコマンドを入力します。

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

この例では、exampleUser がどの IP アドレスからでも接続できるように **address** パラメーターが設定されています。 代わりに、**address** パラメーターを特定の IP アドレスに設定することもできます。 例えば、MQ Console がキュー・マネージャーにリモート接続するときの接続元となる IP アドレスのみにアクセスを制限することができます。 このコマンドで使用可能なオプションについて詳しくは、「**SET CHLAUTH**」を参照してください。

c) ALW

着信ネットワーク接続を受け入れるためのリスナーを作成します。

MQ Console を使用してリスナーを作成することも、コマンド行で **DEFINE LISTENER** MQSC コマンドを使用することもできます。

例えば、リモート・キュー・マネージャー QM1 のポート 1414 にリスナー REMOTE.LISTENER を作成するには、次のコマンドを入力します。

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

d) リスナーが実行されていることを確認します。

MQ Console を使用してリスナーを開始することも、コマンド行で **START LISTENER** MQSC コマンドを使用することもできます。

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、次のコマンドを入力します。

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

z/OS 例えば、z/OS でリスナーを開始するには、次のコマンドを入力します。

```
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。

2. 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 MQ Console を使用して、ローカル・キュー・マネージャー定義から 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。 つまり、 CCDT ファイルにはリモート・キュー・マネージャー (QM1) の接続データが含まれている必要があり、MQ Console を持つホストと同じホストのローカル・キュー・マネージャー (QML) のホストに保管する必要があります。 要約すると、これはローカル・キュー・マネージャーの QML がリモート・キュー・マネージャー QM1 に接続するためのものです。

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 100 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"
      },
      "transmissionSecurity": {
        "cipherSpecification": "",
        "certificateLabel": "",
        "certificatePeerName": ""
      },
      "type": "clientConnection"
    }
  ]
}
```

3. **setmqweb remote** コマンドを使用して、リモート・キュー・マネージャー情報を MQ Console 構成に追加します。 リモート・キュー・マネージャー情報を表示するインストール済み環境に関連付けられている **setmqweb** コマンドを使用する必要があります。

   少なくとも、リモート・キュー・マネージャーを 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。 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 を参照してください。

   例えば、サンプルの CCDT ファイルを使用し、接続に使用するユーザー名を exampleUser と指定して、サンプルのリモート・キュー・マネージャー QM1 を追加するには、次のコマンドを入力します。

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

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 9.2.0 IBM MQ 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

- トピック
- サブスクリプション
- 通信オブジェクト:
  - リスナー
  - キュー・マネージャー・チャネル
  - アプリケーション・チャネル

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」、「トピック」、「サブスクリプション」、または「通信」タブをクリックして、操作するオブジェクトのタイプを選択します。
3. そのタイプのオブジェクトの詳しい操作手順については、以下のトピックのいずれかを参照してください。

V 9.2.0 キューの操作

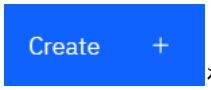
「**キュー**」タブで、特定のキュー・マネージャーに存在するキューを参照できます。 キューの追加と削除、キューでのメッセージの追加とクリア、メッセージの表示、キューのプロパティーの表示と設定、およびキューの権限レコードの管理を行えます。

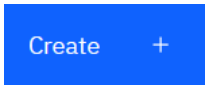
このタスクについて

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 

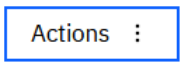
をクリックします。
  b) 作成するキューのタイプを選択します。
    - ローカル・キュー - このキューが属するキュー・マネージャーの内部に、メッセージを保管します。
    - 別名キュー - 同じキュー・マネージャーに属する別のキューを指すポインター。
    - リモート・キュー - 別のキュー・マネージャーに属する別のキューを指すポインター。
    - モデル・キュー - 動的キュー・マネージャーの作成時に使用されるキューのテンプレート。
  c) 作成するキューのタイプについて、必要な情報を入力します。 デフォルトでは、値を入力できるように表示されるのは、最小限の推奨プロパティーです。 「**使用可能なすべてのオプションを表示 (Show all available options)**」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) キューのリストで、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)「アクション」ボタンを 

して **メッセージのクリア**を選択します。

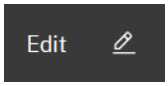
  c)「**メッセージの消去 (Clear messages)**」をクリックして、キューを消去することを確認します。
- キュー上のメッセージをブラウズするには、キューのリスト・ビューでそのキューをクリックします。そのキュー上のメッセージのリストが表示されます。
- キューを削除するには、次のようにします。

  a) 削除するキュー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、「**キューの削除**」をクリックします。

  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キューの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内容を保存します。
- 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) 権限レコードを編集するキュー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。 96 ページの『キュー・マネージャーの権限レコードの操作』を参照してください。

V 9.2.0 トピックの操作

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。

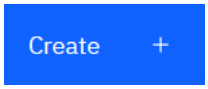
このタスクについて

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。 このトピック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、「**トピック**」タブを選択します。 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)「 **トピックス** 」タブで、「作成」ボタン 

をクリックします。

b) 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。「**使用可能なすべてのオプションを表示 (Show all available options)**」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。

c)「**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。

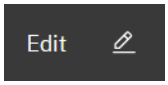
  a) 削除するトピックの横にあるスパナのアイコン  をクリックします。

  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。

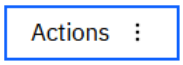
  d)「**保存**」をクリックして変更内容を保存します。

- 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。 必要に応じて、テスト・サブスクリプションを作成できます。

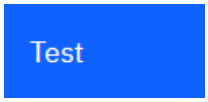
  a) トピックのリストで、パブリッシュするトピックをクリックします。

  b) 次の操作は、このトピックに一致するサブスクリプションがあるかどうかによって異なります。

  一致するサブスクリプションがない場合:

  a.「アクション」ボタンを 

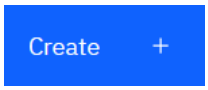
して **テスト・トピック**を選択します。

  b.「テスト」ボタン 

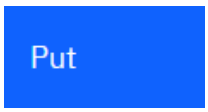
をクリックする。 テスト・メッセージがテスト・サブスクリプションに書き込まれます。

  トピックに一致するサブスクリプションがある場合:

  a. サブスクリプション名をクリックします。

  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 

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、106 ページの『サブスクリプションの操作』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 (96 ページの『キュー・マネージャーの権限レコードの操作』を参照)。

V 9.2.0 サブスクリプションの操作

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。 このサブスクリプション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、「**サブスクリプション**」タブを選択します。 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)「**サブスクリプション**」タブで、「作成」ボタン Create + 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。 「**使用可能なすべてのオプションを表示 (Show all available options)**」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 Edit
  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内容を保存します。
- 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 Create +
  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 Put 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。

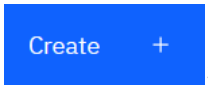
V 9.2.0 リスナーの操作

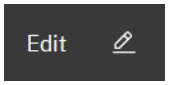
IBM MQ Console を使用して、リスナーの追加と削除、リスナーの開始と停止、リスナーのプロパティーの表示と設定、リスナーの権限レコードの管理を行うことができます。

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

- リスナーを作成するには、次のようにします。
  a)「**コミュニケーション**」タブで、「リスナー」ビューが表示されていることを確認し、「作成ボタン 

をクリックします
  b) 作成するリスナーについて、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。「**使用可能なすべてのオプションを表示 (Show all available options)**」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。
  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**」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内容を保存します。
- 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**」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 (96 ページの『キュー・マネージャーの権限レコードの操作』を参照)。
- 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**」を選択します。

c)「**リスナーの削除**」をクリックします。

V 9.2.0 キュー・マネージャーのチャネルの操作

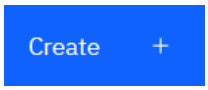
IBM MQ Console を使用して、キュー・マネージャーのチャネルを操作できます。キュー・マネージャーのチャネルの追加と削除、チャネルの開始と停止、チャネルのリセットと解決、チャネルの ping を行うことができます。 キュー・マネージャー・チャネルのプロパティーを表示および設定したり、チャネルの権限レコードを管理したりすることもできます。

このタスクについて

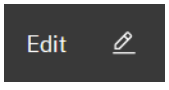
キュー・マネージャーのチャネルは、ネットワークを介してキュー・マネージャー間でメッセージを伝送するための論理的な通信リンクです。 キュー・マネージャーのチャネルのビューには、実行中のチャネルの数、再試行中のチャネルの数、停止したチャネルの数を示すクイック・ビューを表示するパネルが含まれています。

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)「**コミュニケーション**」タブで、キュー・マネージャー・チャネル・ビューが表示されていることを確認し、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。「**使用可能なすべてのオプションを表示 (Show all available options)**」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**」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内容を保存します。

- 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**」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 (96 ページの『キュー・マネージャーの権限レコードの操作』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 9.2.0 アプリケーションのチャネルの操作

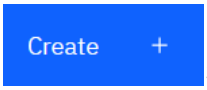
IBM MQ Console を使用して、アプリケーションのチャネルを操作できます。チャネルの追加と削除、チャネルの開始と停止、チャネルのリセットと解決、チャネルの ping を行うことができます。 アプリケーション・チャネルのプロパティーを表示および設定したり、チャネルの権限レコードを管理したりすることもできます。

このタスクについて

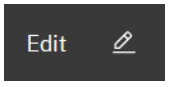
アプリケーションのチャネルは、アプリケーションがネットワークを介してキュー・マネージャーに接続するために使用する論理的な通信リンクです。 アプリケーションのチャネルのビューには、実行中のチャネルの数、再試行中のチャネルの数、停止したチャネルの数を示すクイック・ビューを表示するパネルが含まれています。

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)「**コミュニケーション**」タブで、アプリケーション・チャネル・ビューが表示されていることを確認し、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。「**使用可能なすべてのオプションを表示 (Show all available options)**」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**」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内容を保存します。
- 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 (96 ページの『キュー・マネージャーの権限レコードの操作』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 9.2.0 Web Console の設定

新規 Web Console の一般設定を指定できます。

設定アイコン Settings をクリックして、 Web コンソールの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。

V 9.2.0 z/OS Multi コンソール・タイプの切り替え

New Web Console (IBM MQ 9.2 のデフォルトの Web コンソール) と Dashboard Web Console (旧バージョンの IBM MQ の Web コンソール) を切り替えることができます。

このタスクについて

コンソールを切り替えるには、**setmqweb** コマンドを使用します。

Dashboard Web Console の使用に戻す場合は、 IBM MQ 9.1 資料の「 ダッシュボード Web コンソール 」を参照してください。 使用方法についての説明を記載しています。

このタスクを実行するユーザーは、**dspmqweb** コマンドと **setmqweb** コマンドを使用できる特定の特権を持つユーザーでなければなりません。

- z/OS z/OS では、**dspmqweb** コマンドと **setmqweb** コマンドの実行権限と、`mqwebuser.xml` ファイルに対する書き込み権限が必要です。
- 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。

重要 : z/OS

setmqweb または **dspmqweb** コマンドを z/OS で発行する前に、WLP_USER_DIR 環境変数を設定して、その変数が mqweb サーバー構成を指定するようにする必要があります。

そのためには、以下のコマンドを実行します。

```
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。 以下に例を示します。

```
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。

手順

- Dashboard Web Console に切り替えるには、以下のコマンドを使用します。
  - 次のコマンドを入力して、mqConsoleEnableUnsafeInline プロパティーを true に設定します。

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

    このプロパティーを設定すると、CSP ルール・セットで unsafe-inline が有効になるように制限が緩和されて現在の構成の安全性が低下する可能性がありますが、Dashboard Web Console ではこれが必須になっています。
  - 次のコマンドを入力して、mqweb サーバーが Dashboard Web Console を使用するように切り替えます。

    ```
    setmqweb properties -k mqConsoleEarName -v com.ibm.mq.console
    ```
- MQ Web サーバーのデフォルトの設定 (New Web Console を含む) に戻すには、以下のコマンドを使用します。

  ```
  setmqweb properties -r
  ```
- New Web Console に戻しつつ、mqweb サーバーに対して行ったその他のカスタマイズを保持するには、以下のコマンドを使用します。
  - 次のコマンドを入力して、セキュリティー構成を復元するように mqConsoleEnableUnsafeInline プロパティーを false に設定します。

    ```
    setmqweb properties -k mqConsoleEnableUnsafeInline -v false
    ```
  - 次のコマンドを入力して、mqweb サーバーが New Web Console を使用するように切り替えます。

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

関連資料

setmqweb

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、115 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、製品インストールの一部としてインストールすることも (IBM MQ のインストールおよびアンインストールを参照)、スタンドアロンの IBM MQ Explorer Fix Central からダウンロードできます (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)。 からインストールすることもできます。

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。
- **strmqikm** (ikeyman) GUI を呼び出して TLS 証明書の管理、証明書のキュー・マネージャーへの関連付け、証明書ストアの構成およびセットアップ (ローカル・マシンでのみ)。
- 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。

- 115 ページの『IBM MQ Explorer の前提ソフトウェアと定義』
- 115 ページの『IBM MQ Explorer のセキュリティー』
- 119 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』

- 120 ページの『クラスター・メンバーシップと IBM MQ Explorer』
- 120 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 の概要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、 117 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 117 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。
   - AIX AIX または 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。 **strmqikm** (iKeyman) GUI にあるオプションにチェック・マークを付けるか、**runmqckm** コマンドまたは **runmqakm** コマンドで `-stash` オプションを使用することによって、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 は、TLS セキュリティーに Java 鍵ストア・ファイル (JKS) を使用するので、IBM MQ Explorer の TLS を構成するために作成する鍵ストア・ファイルは、これと一致し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。 117 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 117 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**」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。

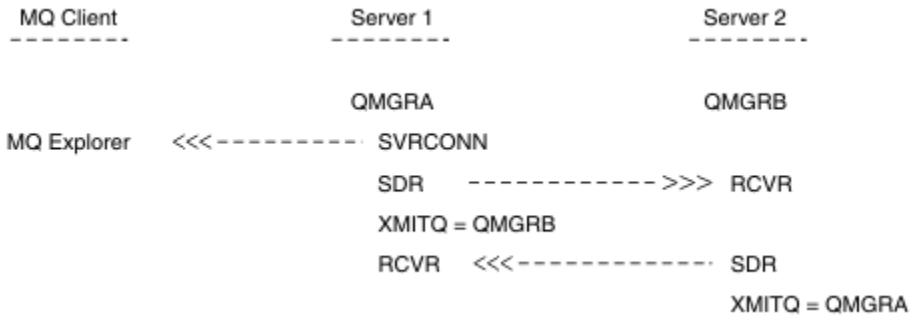
この接続が、中間キュー・マネージャーを経由してターゲット・キュー・マネージャーに接続するために使用される場合は、再びユーザー ID が、メッセージ記述子 (MQMD) の *UserIdentifier* パラメーターとして送られます。 ターゲット・キュー・マネージャーの MCA リスナーがこのメッセージを受け取るには、MCAUSER 属性を設定するか、書き込み権限を持つユーザー ID がすでに存在している必要があります。

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。186 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**」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、112 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
  - 11 ページの『MQSC コマンドによる管理』
  - 129 ページの『キュー・マネージャーの属性の表示および変更』
  - 132 ページの『ローカル・キューの処理』
  - 144 ページの『別名キューの処理』
  - 145 ページの『モデル・キューの処理』
  - 172 ページの『サービスの取り扱い』
  - 179 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説明については、 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、112 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。

V 9.2.0 IBM MQ 9.1.4 以降、指定した秒数のターゲット時間内にキュー・マネージャーを終了するオプションが用意されています。詳しくは、 125 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。

endmqm コマンドとそのオプションについて詳しくは、 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。 キュー・マネージャーがどのインストール済み環境に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 126 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

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・マネージャーは、重要なキュー・マネージャー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、必須のキュー・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、指定されたターゲット時間に準拠するために必要な場合に、キュー・マネージャーの重要な保守タスクを中断します。

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` シャットダウンでは、キュー圧縮や、時間がかかる可能性がある NPMCLASS (HIGH) メッセージの永続化などのハウスキーピング・アクションが引き続き実行されます。一方、時間制限のあるシャットダウンでは、これらの追加アクションがターゲット時間の達成に支障をきたすと終了します。

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 (122 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、126 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、127 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

表 7. 実行している場合に停止する *Windows* プロセス (続き)	
プロセス名	**説明**
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・マネージャーのプロセス ID を検出します。
   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
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

注 : kill -9 コマンドを使用すると、停止に失敗するプロセスを終了させることができます。

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説明については、 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

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。

このタスクについて

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には **DISPLAY QMGR** コマンドを使用し、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するには **ALTER QMGR** コマンドを使用します。

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
AMQ8408: Display Queue Manager details.
   QMNAME(QM1)                             ACCTCONO(DISABLED)
   ACCTINT(1800)                           ACCTMQI(OFF)
   ACCTQ(OFF)                              ACTIVREC(MSG)
   ACTVCONO (DISABLED)                     ACTVTRC (OFF)
   ALTDATE(2012-05-27)                     ALTTIME(16.14.01)
   AUTHOREV(DISABLED)                      CCSID(850)
   CHAD(DISABLED)                          CHADEV(DISABLED)
   CHADEXIT( )                             CHLEV(DISABLED)
   CLWLDATA( )                             CLWLEXIT( )
   CLWLLEN(100)                            CLWLMRUC(999999999)
   CLWLUSEQ(LOCAL)                         CMDEV(DISABLED)
   CMDLEVEL(800)                           COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)
   CONFIGEV(DISABLED)                      CRDATE(2011-05-27)
   CRTIME(16.14.01)                        DEADQ()
   DEFXMITQ( )                             DESCR( )
   DISTL(YES)                              INHIBTEV(DISABLED)
   IPADDRV(IPV4)                           LOCALEV(DISABLED)
   LOGGEREV(DISABLED)                      MARKINT(5000)
```

```
   MAXHANDS(256)                           MAXMSGL(4194304)
   MAXPROPL(NOLIMIT)                       MAXPRTY(9)
   MAXUMSGS(10000)                         MONACLS(QMGR)
   MONCHL(OFF)                             MONQ(OFF)
   PARENT( )                               PERFMEV(DISABLED)
   PLATFORM(WINDOWSNT)                        PSRTYCNT(5)
   PSNPMSG(DISCARD)                        PSNPRES(NORMAL)
   PSSYNCPT(IFPER)                         QMID(QM1_2011-05-27_16.14.01)
   PSMODE(ENABLED)                         REMOTEEV(DISABLED)
   REPOS( )                                REPOSNL( )
   ROUTEREC(MSG)                           SCHINIT(QMGR)
   SCMDSERV(QMGR)                          SSLCRLNL( )
   SSLCRYP( )                              SSLEV(DISABLED)
   SSLFIPS(NO)                             SSLKEYR(C:\Program Files\IBM\WebSphere
MQ\Data\qmgrs\QM1\ssl\key)
   SSLRKEYC(0)                             STATACLS(QMGR)
   STATCHL(OFF)                            STATINT(1800)
   STATMQI(OFF)                            STATQ(OFF)
   STRSTPEV(ENABLED)                       SYNCPT
   TREELIFE(1800)                          TRIGINT(999999999)
```

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 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** コマンドを使用する場合、確認のための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 (122 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 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説明については、 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。

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。 133 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、135 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 コマンドを使用して、IBM MQ オブジェクトが定義されたときに、デフォルト・オブジェクトから得られた属性を表示できます。

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 コマンドに **LIKE** 属性を使用してキュー定義をコピーできます。

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

キュー属性は 2 つの方法で変更できます。1 つは **ALTER QLOCAL** 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 QLOCAL** コマンドを使用する方法です。

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 コマンドを使用して、ローカル・キューをクリアします。

始める前に

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

MQSC コマンド **DELETE QLOCAL** を使用して、ローカル・キューを削除します。

このタスクについて

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。

ソース
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。

ソース
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。

V 9.2.0 IBM MQ 9.1.5 以降、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、138 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。

V 9.2.0 Multi IBM MQ キュー・ファイルの変更

IBM MQ 9.2.0 以降、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。

詳しくは、MAXFSIZE および 139 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。

この属性の PCF 属性は 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。

詳しくは、CURFSIZE を参照してください。

この属性の PCF 属性は MQIA_CUR_Q_FILE_SIZE です。

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。

詳しくは、CURMAXFS を参照してください。

この属性の PCF 属性は MQIA_CUR_MAX_FILE_SIZE です。

これらの 2 つの PCF 属性の詳細については、Inquire Queue および Inquire Queue (応答) を参照してください。

これらの属性は、MQSC コマンド、IBM MQ Explorer、administrative REST API を使用して設定および表示できます。

注 : IBM MQ Console では、MAXFSIZE の値の表示のみを行えます。つまり、値を構成することはできません。

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。

大きな MAXFSIZE 値を指定して新しく定義したキューを作成した場合は、適切なブロック・サイズを使用してキューが作成されます。 一方、ALTER QLOCAL コマンドを使用するなどして既存のキューの MAXFSIZE 値を増やした場合は、キュー・マネージャーがそのキューを再構成できるように、キューを空にしなければならないことがあります。

詳しくは、140 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

V 9.2.0 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

キュー・ファイルの新しいサイズを設定する前に、DISPLAY QLOCAL 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ます。 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。

デフォルト値よりもキュー・ファイルの最大サイズを大きくすることを検討している場合には、相関 ID や IBM MQ classes for JMS セレクター文字列などのメッセージ・セレクターを慎重に使用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) 以下のコマンドを発行して、SMALLQUEUE というローカル・ファイルを 500 ギガバイトのサイズで作成します。

```
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) 以下のコマンドを発行して、LARGEQUEUE というローカル・ファイルを 5 テラバイトのサイズで作成します。

```
DEFINE QLOCAL(LARGEQUEUE) MAXFSIZE(5242880)
     3 : DEFINE QLOCAL(LARGEQUEUE) MAXFSIZE(5242880)
AMQ8006I: IBM MQ queue created
```

V 9.2.0 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。

ブロック・サイズと細分度

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。 キューが空になったら、キュー・マネージャーが、構成されている 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 コマンドの新しい属性 CURMAXFS により、 キューが新しいブロック・サイズを使用するように変更されたことを確認できます。

以下の例では、CURMAXFS 値 4177920 から、キュー・ファイルのサイズが現在は約 4 テラバイトまで拡張可能であることを確認できます。 キューに構成されている MAXFSIZE 値が CURMAXFS 値より大きい場合、キュー・マネージャーは、キューが空になるまで待ってからキュー・ファイルのブロック・サイズを再構成します。

```
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・ファイルのサイズの確認

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・ファイルの現在の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削除されても、すぐには CURFSIZE 属性が減らないことがあります。

通常、キュー・ファイルの未使用スペースは、以下の状況でのみ解放されます。

- キューを開いているアプリケーションがないとき
- キュー・マネージャー・ログに 1000 件書き込まれた後
- キュー・マネージャーがシャットダウンするとき

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・キューのローカル定義です。 つまり、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、ローカルに定義する利点は、アプリケーションが、リモート・キューがあるキュー・マネージャーの ID によって修飾された名前を指定しなくても、ローカルに定義された名前によってリモート・キューを参照できることです。

リモート・キューのローカル定義の機能

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。

以下の例は、アプリケーションが、リモート・キュー・マネージャーが所有しているキューにメッセージを入れる方法を示しています。 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・キューに関してその他のコマンドを使用する

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・キュー定義を別名として使用する

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・マネージャー別名の詳細については、 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。

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。

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 (132 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 (141 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。

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。

```
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。

以下のコマンドは、アプリケーション ALPHA の PUT を使用可能にし、GET を使用不可にする別名を定義します。

```
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。

キュー別名でのその他のコマンドの使用

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。

関連資料
ALTER QALIAS
DEFINE QALIAS
DELETE QALIAS
配布リスト

モデル・キューの処理

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

DEFINE QMODEL 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・キューでのその他のコマンドの使用

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。

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には、送達不能キュー名を **crtmqm** コマンド (`crtmqm -u DEAD.LETTER.QUEUE` など) に指定するか、あるいは **ALTER QMGR** コマンド上の **DEADQ** 属性を使用した後でそれを指定します。 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。

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。 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 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。 147 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) にあるメッセージを処理するために、IBM MQ にはデフォルトの DLQ ハンドラーが用意されています。 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。

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 は、送達不能キュー・ハンドラー (DLQ ハンドラー) と呼ばれるデフォルト・ルーチンを提供しています。DLQ ハンドラーは、`runmqdlq` コマンドを使用して呼び出します。

DLQ 上のメッセージを処理する命令は、ユーザー作成ルール・テーブル を介して DLQ ハンドラーに提供されます。 つまり、DLQ ハンドラーは DLQ 上のメッセージとルール・テーブルの項目の突き合わせを行います。DLQ メッセージがルール・テーブルの項目と一致すると、DLQ ハンドラーはその項目に関連付けられたアクションを実行します。

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 送達不能キュー・ハンドラー *(IBM i)*

IBM i 送達不能キュー・ハンドラーの説明と呼び出し方。

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。

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、146 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。

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には、必ず MQDLH を提供するようにしてください。 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おり、STRMQMDLQ コマンドを使用して起動します。 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。

DLQ ハンドラーの起動

STRMQMDLQ コマンドを使用して、DLQ ハンドラーを起動します。 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 のパラメーターとして指定する。 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。

STRMQMDLQ コマンドは、規則テーブルから入力データを受け取ります。

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。

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 (*QueueManagerName*|' ' (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 (*QueueManagerName*| & 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 *DLQ 規則テーブルの処理 (IBM i)*

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、メッセージと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、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. 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。

IBM i すべての *DLQ* メッセージを確実に処理するための機能 *(IBM i)*

送達不能キュー・ハンドラーは、IBM i の DLQ で認識されていてもまだ削除されていないすべてのメッセージのレコードを保持しています。

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 が先入れ先出し法 (FIFO) で定義されている場合であっても、 DLQ ハンドラーが、 DLQ に入って来る新しいメッセージを必ず検出できるとは限らないので、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。

これらの理由から、必ず DLQ には可能な限り少ないメッセージを含めます。 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適切な処置をとることができます。 例えば、ACTION (IGNORE) は、DLQ 上のメッセージを放置するので、使用しないようにしてください (テーブルの中の他の規則によって明示的に処理されないメッセージには、ACTION (IGNORE) が適用されることに注意してください)。 その代わり、無視するメッセージに関して、別のキューにそのメッセージを移動するアクションを実行してください。 以下に例を示します。

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

また、テーブルの最後の規則は、テーブルの中のそれまでのルールから漏れたメッセージをまとめて扱えるものにします。 例えば、テーブルの中の最後のルールは、次のような形にすることができます。

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

これにより、表の最終規則に該当するメッセージがキュー REALLY.DEAD.QUEUE に転送され、手動で処理できるようになります。 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。

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)
```

DLQ ハンドラーの起動

runmqdlq コマンドを使用して、送達不能キュー・ハンドラーを起動します。 処理する DLQ の名前および使用するキュー・マネージャーの名前を指定するには、次の 2 つの方法があります。

2 つの方法は次のとおりです。

- runmqdlq へのパラメーターとしてコマンド・プロンプトから指定する。 以下に例を示します。

```
runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER <qrule.rul
```

- 規則テーブルで指定する。 以下に例を示します。

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

この例は、キュー・マネージャー ABC1.QUEUE.MANAGER が所有する ABC1.DEAD.LETTER.QUEUE という DLQ に適用されます。

DLQ またはキュー・マネージャーを例に示すように指定しなかった場合は、インストール先のデフォルト・キュー・マネージャーと共に、そのキュー・マネージャーの DLQ が使用されます。

runmqdlq コマンドは、 stdin から入力を受け取ります。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。

DLQ ハンドラーを実行するためには、DLQ 自体、および DLQ 上のメッセージの転送先となるあらゆるメッセージ・キューの両方へのアクセスが許可されていることが必要です。 DLQ ハンドラーがメッセージ・コンテキスト中のユーザー ID の権限を使用してキューにメッセージを書き込むことが可能になっている場合には、DLQ ハンドラーを実行するユーザーは他のユーザーの ID を借用する許可を持っている必要があります。

runmqdlq コマンドの詳細については、runmqdlq を参照してください。

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

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

V 9.2.3 **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)"
```

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 (*QueueManagerName*|' ' (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 (*QueueManagerName*| & 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。
- 次行の最初の非ブランク文字にルールが継続するよう指示するには、行の最後の非ブランク文字として正符号 (+) を使用します。 次行の先頭にルールが継続するよう指示するには、行の最後の非ブランク文字として負符号 (-) を使用します。 連結文字がキーワードおよびパラメーターの内部に現れても構いません。

  以下に例を示します。

```
APPLNAME('ABC+
D')
```

と指定すると 'ABCD' となり、

```
APPLNAME('ABC-
D')
```

これは 'ABC D' となります。

- 注釈行は、アスタリスク (*) で始まり、規則テーブルのどの位置にでも含めることができます。
- ブランク行は無視されます。
- DLQ ハンドラーのルール・テーブルの各項目は、1 つ以上のキーワードと、それらに関連付けられたパラメーターからなります。 パラメーターは、次の構文規則に従う必要があります。
  - 各パラメーター値は、有効な文字を 1 つ以上含んでいる必要があります。 引用符で囲んだ値の区切り用の引用符は、無効とみなされます。 例えば、次のパラメーターは有効です。

FORMAT('ABC')	有効文字数は 3
FORMAT(ABC)	有効文字数は 3
FORMAT('A')	有効文字数は 1
FORMAT(A)	有効文字数は 1
FORMAT(' ')	有効文字数は 1

    次のパラメーターは、有効文字を含んでいないので無効です。

    ```
    FORMAT('')
    FORMAT( )
    FORMAT()
    FORMAT
    ```

  - ワイルドカード文字はサポートされています。 後書きブランク以外の 1 文字の代わりに疑問符 (?) を使用し、また 0 個以上の隣接した文字の代わりにアスタリスク (*) を使用することができます。 アスタリスク (*) および疑問符 (?) は、パラメーター値の中では **常に** ワイルドカード文字と解釈されます。
  - 次のキーワードのパラメーターの中には、ワイルドカード文字を含めることはできません。ACTION、HEADER、RETRY、FWDQ、FWDQM、および PUTAUT。
  - パラメーター値の中の後書きブランク、および DLQ 上のメッセージ内のそれに対応するフィールドの中の後書きブランクは、ワイルドカード突き合わせの実行時には無効です。 ただし、引用符で囲まれたストリング内の先行および組み込みブランクは、ワイルドカード照合時に有効です。
  - 数値パラメーターには、疑問符 (?) のワイルドカード文字を含めることはできません。 1 個の数値パラメーター全体の代わりにアスタリスク (*) を使用できますが、数値パラメーターの一部分として含めることはできません。 例えば、次の数値パラメーターは有効です。

MSGTYPE(2)	応答メッセージのみが対象
MSGTYPE(*)	あらゆるメッセージ・タイプが対象
MSGTYPE('*')	あらゆるメッセージ・タイプが対象

    しかし、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。

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

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 からメッセージの小さいサブセットを 抽出するためのフィルターとして DLQ ハンドラーを使用する場合にも、DLQ ハンドラーは、DLQ 上にある未処理のメッセージの記録を保持し続けます。 また、DLQ が

先入れ先出し (FIFO) として定義されても、DLQ に到着する新規メッセージが参照されることを DLQ ハンドラーは保証できません。 キューが空ではないとき、DLQ は定期的に再スキャンされて、すべてのメッセージが検査されます。

以上の点から、DLQ にはできるだけ少数のメッセージを入れるようにしてください。廃棄したり他のキューに転送したりできない (その理由が何であろうと) メッセージをキュー上に累積させると、DLQ ハンドラーのワークロードが増大し、DLQ 自体が満杯になる可能性があります。

DLQ ハンドラーが DLQ を空にできるように適切な処置をとることができます。 例えば、ACTION (IGNORE) は、DLQ 上のメッセージを放置するので、使用しないようにしてください (テーブルの中の他の規則によって明示的に処理されないメッセージには、ACTION (IGNORE) が適用されることに注意してください)。 その代わり、無視するメッセージに関して、別のキューにそのメッセージを移動するアクションを実行してください。 例えば、

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

また、テーブルの最後の規則は、テーブルの中のそれまでのルールから漏れたメッセージをまとめて扱えるものにします。 例えば、テーブルの中の最後のルールは、次のような形にすることができます。

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

これにより、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。

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)

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
166 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。

166 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。

167 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。

167 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。

168 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。

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。 166 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、167 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。

すべてのトピックを表示するには、次を使用します。

```
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
```

このコマンドにより、次のような 3 つの指定の属性が表示されます。

```
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。

例えば、ORANGE.TOPIC という名前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 5 に変更する場合は、次のコマンドのいずれかを使用します。

- **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。しかし、すべての新規メッセージは、パブリッシュ側のアプリケーションで優先順位が指定されていない場合、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。

以下に例を示します。

```
DEFINE TOPIC (MAGENTA.TOPIC) +
LIKE (ORANGE.TOPIC)
```

このコマンドは、システム・デフォルト管理トピックの属性ではなく、オリジナルのトピック ORANGE.TOPIC と同じ属性を持つトピック MAGENTA.TOPIC を作成します。 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。

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。

以下に例を示します。

```
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。

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。

関連概念

169 ページの『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。

170 ページの『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。

171 ページの『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。

171 ページの『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。

172 ページの『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。

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。

例えば、以下の **DEFINE SUB** コマンドは、次のような特性を持つ ORANGE と呼ばれる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。 170 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、171 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。

例えば、次を使用します。

```
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。

例えば、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、次のコマンドのいずれかを使用します。

- **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。

以下に例を示します。

```
DEFINE SUB(BLUE) +
       LIKE(ORANGE)
```

サブスクリプション REAL の属性をサブスクリプション THIRD.SUB にコピーし、配信されたパブリケーションの correlID が、パブリッシャーの correlID ではなく、THIRD になるよう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。

```
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。

このタスクについて

以下の **runmqsc** コマンドでは、メッセージを受信したサブスクリプションのみが表示され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、 170 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、136 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。

新しい IBM MQ サービス定義を定義するには、MQSC コマンド DEFINE SERVICE を使用します。

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
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。

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。

関連概念
174 ページの『サービスの管理』
CONTROL パラメーターを使用すると、サービス・オブジェクトのインスタンスの開始および停止をキュー・マネージャーによって自動的に行う方法と、MQSC コマンド **START SERVICE** および **STOP SERVICE** で制御する方法のいずれかを使用することができます。

関連資料
DEFINE SERVICE
DISPLAY SVSTATUS
START SERVICE
STOP SERVICE

サービスの管理

CONTROL パラメーターを使用すると、サービス・オブジェクトのインスタンスの開始および停止をキュー・マネージャーによって自動的に行う方法と、MQSC コマンド **START SERVICE** および **STOP SERVICE** で制御する方法のいずれかを使用することができます。

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。 サーバー・サービス・オブジェクトの開始のログ項目の例を以下に示します。

```
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.
```

コマンド・サービス・オブジェクトの開始のログ項目の例を以下に示します。

```
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。 サーバー・サービス・オブジェクトの停止のログ項目の例を以下に示します。

```
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.
```

関連資料
175 ページの『追加の環境変数』

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。

STOP SERVICE
START SERVICE

追加の環境変数

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。

環境変数を追加できるファイル

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
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。

キュー・マネージャーの有効範囲 service.env ファイル
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

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 で指定できる環境変数。

service.env に環境変数を指定することができます。 例えば、 IBM MQ サービスがいくつかのコマンドを実行する場合は、 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。 変数に設定する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、予期しない結果が生じます。

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。 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 は、ファイル内で定義される変数のフォーマットを、名前と値の変数のペアのリストにします。 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。

service.env 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
```

関連資料

176 ページの『サービス定義での置き換え可能な挿入』
サービス・オブジェクトの定義では、トークンの置換が可能です。 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。 置換トークンは、以下に示す共通トークンのリストから、あるいは、ファイル service.env に定義された変数から取得することができます。

サービス定義での置き換え可能な挿入

サービス・オブジェクトの定義では、トークンの置換が可能です。 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。 置換トークンは、以下に示す共通トークンのリストから、あるいは、ファイル service.env に定義された変数から取得することができます。

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。

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。 この挿入の例については、177 ページの『サービス・オブジェクトの使用例』を参照してください。

サービス・オブジェクトの使用例

特に指定のない場合、このセクションのサービスは、UNIX スタイルのパス区切り文字を使用して記述されます。

サーバー・サービス・オブジェクトの使用

この例は、トリガー・モニターを開始するサーバー・サービス・オブジェクトの定義、使用、および変更の方法を示しています。

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、176 ページの『サービス定義での置き換え可能な挿入』を参照してください。

2. キュー・マネージャーが次に開始されるときに、このサーバー・サービス・オブジェクトのインスタンスが実行されます。 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します。

```
START SERVICE(S1)
```

3. **DISPLAY SVSTATUS** MQSC コマンドを使用すると、サーバー・サービス・プロセスの状況が表示されます。

```
DISPLAY SVSTATUS(S1)
```

4. 次に、この例で、サーバー・サービス・オブジェクトを変更し、サーバー・サービス・プロセスを手動で再開することによって、更新を取得させる方法を示します。 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。 以下の **ALTER SERVICE** MQSC コマンドを使用します。

```
ALTER SERVICE(S1) +
STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

   注: 実行中のサービスでは、サービスが再開されるまで、サービス定義に対する更新を取得しません。

5. 以下の **STOP SERVICE** および **START SERVICE** コマンドを使用して、サーバー・サービス・プロセスを再開し、変更を取得できるようにします。

```
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、177 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。

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

この例では、キュー・マネージャーの開始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始動するコマンド・サービス・オブジェクトの定義方法を示します。

1. **DEFINE SERVICE** MQSC コマンドを使用して、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')
```

説明

logger は、AIX システムまたは Linux システムによって提供される、システム・ログに書き込みを行うコマンドです。
+QMNAME+ は、キュー・マネージャーの名前を表すトークンです。

関連資料
DEFINE SERVICE

キュー・マネージャーの終了時のみのコマンド・サービス・オブジェクトの使用

この例では、キュー・マネージャーの停止時のみ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始動するコマンド・サービス・オブジェクトの定義方法を示します。

1. **DEFINE SERVICE** MQSC コマンドを使用して、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')
```

説明

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。

関連資料
DEFINE SERVICE

引数の引き渡しについて

この例では、キュー・マネージャーの開始時に runserv というプログラムを始動するサーバー・サービス・オブジェクトの定義方法を示します。

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。

始動する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。 このためには、以下のコマンドで示すように二重引用符を使用してコマンド・サービス・オブジェクトを定義します。

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) 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。

関連資料

DEFINE SERVICE

サービスの自動始動

次の例は、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・オブジェクトの定義方法を示しています。

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(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

DEFINE SERVICE

トリガー操作のためのオブジェクトの管理

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ための機能があります。 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。

トリガーについて詳しくは、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。

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、**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を定義する

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. IBM MQ 9.1 以降、 **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・セットとともに出荷されるため、 IBM MQ サーバーとクライアントの両方で使用できます。 IBM MQ 9.1 より前のバージョンでは、ユーティリティーはサーバー・パッケージにのみ付属していました。

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

183 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。 このユーティリティーは、IBM MQ 8.0 から製品に組み込まれています。

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。 このユーティリティーは、IBM MQ 8.0 から製品に組み込まれています。

qload ユーティリティーは、以前は SupportPac MO03 として提供されていました。

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

183 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

183 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

184 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

183 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、186 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。
  - 186 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』
  - 190 ページの『リモート管理でコマンド・サーバーを管理する』
  - 191 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』
  - 193 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

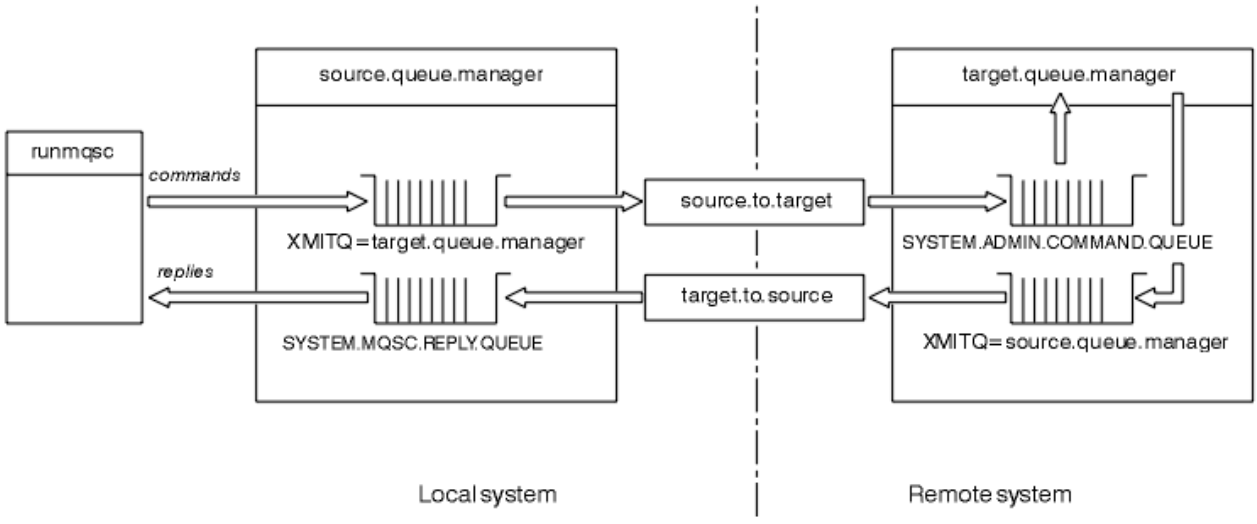


図 *15.* リモート管理のためのチャネルとキューのセットアップ

186 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) 次のコマンドを入力して、**runmqsc** 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) 次のコマンドを入力して、**runmqsc** 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・マネージャーに対して **runmqsc** 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、191 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、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、77 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) 次のコマンドを入力して、該当するキュー・マネージャーに対して **runmqsc** 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) 次のコマンドを入力して、**runmqsc** 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、186 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、190 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。
     - z/OS 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
```

     - Multi リモート・キュー・マネージャーが z/OS 上にない場合は、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。

- z/OS 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
```

- Multi リモート・キュー・マネージャーが z/OS 上にない場合は、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、すべてのキュー・マネージャー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。

V 9.2.0 IBM MQ 9.2.0 以降、CCSID 37 と 500 間の変換が IBM MQ Appliance、Windows、 Linux、および macOS でサポートされるようになりました。

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 196 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、 194 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

IBM MQ 9.0 より前のバージョンでは、製品の以前のバージョンは基本多言語面以外の Unicode コード・ポイント (U+FFFF を超えるコード・ポイント) が含まれたデータの変換はサポートしていませんでした。Unicode データ変換のサポートは、Unicode 3.0 標準に定義されているコード・ポイントに限られ、UTF-16 の 2 バイト固定幅サブセットである UTF-8 または UCS-2 のいずれかでエンコードされていました。

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。 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

IBM MQ 9.0 から、追加のファイル `ccsid_part2.tbl` が提供されます。

`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。

詳しくは、195 ページの『ccsid_part2.tbl ファイル』を参照してください。

ccsid.tbl ファイル

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

IBM MQ 9.0 以降、以下のプラットフォームでは、デフォルトのデータ変換方式が変更されています。

- Linux Linux - すべてのバージョン
- Windows Windows

詳しくは、ccsid_part2.tbl を使用して、196 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。

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

キュー・マネージャーの CCSID を変更するために ALTER QMGR コマンドの CCSID 属性を使用したとき、コマンド・サーバーおよびチャネル・プログラムを含む、実行しているすべてのアプリケーションが確実に停止され、再始動されるようにするためにキュー・マネージャーを停止し、再始動します。

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、193 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 Linux - すべてのバージョン
- Windows 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 Linux のすべてのバージョンの *MQDataRoot*/conv/table ディレクトリー。

- Windows 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。

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、198 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、200 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。
  詳しくは、201 ページの『z/OS での MFT エージェントの開始』を参照してください。

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 MFT エージェントまたはロガーを実行するためのガイダンスを参照してください。

   詳細は、 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)の作成と設定）をご覧ください。
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、MFT エージェントまたはロガーを Windows サービスとして実行するためのガイダンス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。

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 マシンのリブート時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
```

z/OS での MFT エージェントの開始

z/OS では、 z/OS UNIX System Services セッションから **fteStartAgent** コマンドを実行するだけでなく、対話式セッションを必要とせずに JCL から開始タスクとしてエージェントを開始することもできます。

開始タスクは、特定のユーザー ID の下で実行され、ユーザーのログオフに影響されないため、使用されます。

注 : 通常、開始済みタスクはログオン特権がない可能性がある管理ユーザーの下で実行されるため、エージェントを実行しているユーザーとして z/OS システムにログオンすることはできません。 そのエージェントに対して、**fteStartAgent**、**fteStopAgent**、**fteSetAgentTraceLevel** の各コマンド、および **-d** パラメーターが指定されている **fteShowAgentDetails** コマンドを実行することはできません。

IBM MQ 9.0.2 および IBM MQ 9.0.0 Fix Pack 1 以降、z/OS 上の Managed File Transfer エージェントでエージェント・プロパティー **adminGroup** を使用できるようになりました。 例えば、MFTADMIN というセキュリティー・マネージャー・グループを定義し、開始済みタスクのユーザー ID と管理者 TSO ID をこのグループに追加できます。 エージェント・プロパティー・ファイルを編集し、**adminGroup** プロパティーの値をこのセキュリティー・マネージャー・グループの名前に設定します。

```
adminGroup=MFTADMIN
```

これにより、このグループのメンバーは、開始済みタスクとして実行されているエージェントに対して、**fteStartAgent**、**fteStopAgent**、**fteSetAgentTraceLevel** の各コマンド、および **-d** パラメーターが指定されている **fteShowAgentDetails** コマンドを実行できます。

詳しくは、MFT agent.properties ファイルの **adminGroup** プロパティーを参照してください。

エージェントは、Java アプリケーションであり、JCL から実行できる z/OS UNIX System Services アプリケーションです。これを実行するには、エージェントに対して生成された Managed File Transfer コマンド PDSE ライブラリー・データ・セットから、 BFGAGSTP メンバーを使用します。 MFT コマンド PDSE ライブラリー・データ・セットを作成し、必要なエージェント用にカスタマイズする方法について詳しくは、MFT エージェントまたはロガー・コマンド・データ・セットの作成を参照してください。

エージェントがリモート z/OS キュー・マネージャーに接続できるようにする

LTS

z/OS 上の MFT エージェントは、以下のシナリオでクライアント接続を使用して z/OS 上のキュー・マネージャーに接続できます。

- MFT エージェントは IBM MQ 9.2.0 Long Term Support (APAR PH56722 適用済み) にあり、 IBM MQ Advanced for z/OS Value Unit Edition または IBM MQ Advanced for z/OS のいずれかの製品 ID (PID) に関連付けられています。
- MFT エージェントは IBM MQ 9.2.0 にあり、IBM MQ Advanced for z/OS Value Unit Edition の PID に関連付けられています。

IBM MQ 製品、関連する PID 値、およびエクスポート分類の詳細については、 IBM MQ 製品 ID およびエクスポート情報 を参照してください。

MFT インストールに関連した PID を設定する方法については、「**fteSetProductID**」を参照してください。

エージェントの実行に使用される PID はエージェント開始時にログに表示されます。

注 : 他の PID で実行されている z/OS 上の MFT エージェントは、バインディング・モード接続を使用する場合にのみ、ローカル・キュー・マネージャーに接続できます。

z/OS で実行されていないキュー・マネージャーにエージェントが接続しようとすると、メッセージ BFGQM1044E が発行され、エージェントの開始が終了します。

関連資料

203 ページの『z/OS 上の MFT エージェントの停止』
JCL からの開始タスクとして z/OS 上で Managed File Transfer Agent を実行している場合、エージェントは、 **fteStopAgent** コマンドに加えて、 z/OS オペレーター・コマンド **MODIFY** および **STOP** を受け入れます。

MFT agent.properties ファイル

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。

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。

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、198 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。

z/OS z/OS 上の MFT エージェントの停止

JCL からの開始タスクとして z/OS 上で Managed File Transfer Agent を実行している場合、エージェントは、 **fteStopAgent** コマンドに加えて、 z/OS オペレーター・コマンド **MODIFY** および **STOP** を受け入れます。

開始タスクは、特定のユーザー ID の下で実行され、ユーザーのログオフに影響されないため、使用されます。

注 : 通常、開始済みタスクはログオン特権がない可能性がある管理ユーザーの下で実行されるため、エージェントを実行しているユーザーとして z/OS システムにログオンすることはできません。 そのエージェントに対して、**fteStartAgent**、**fteStopAgent**、**fteSetAgentTraceLevel** の各コマンド、および **-d** パラメーターが指定されている **fteShowAgentDetails** コマンドを実行することはできません。

IBM MQ 9.0.2 および IBM MQ 9.0.0 Fix Pack 1 以降、z/OS 上の Managed File Transfer エージェントでエージェント・プロパティー **adminGroup** を使用できるようになりました。 例えば、MFTADMIN というセキュリティー・マネージャー・グループを定義し、開始済みタスクのユーザー ID と管理者 TSO ID をこのグループに追加できます。 エージェント・プロパティー・ファイルを編集し、**adminGroup** プロパティーの値をこのセキュリティー・マネージャー・グループの名前に設定します。

```
adminGroup=MFTADMIN
```

これにより、このグループのメンバーは、開始済みタスクとして実行されているエージェントに対して、**fteStartAgent**、**fteStopAgent**、**fteSetAgentTraceLevel** の各コマンド、および **-d** パラメーターが指定されている **fteShowAgentDetails** コマンドを実行できます。

詳しくは、 MFT agent.properties ファイル の **adminGroup** プロパティーを参照してください。

z/OS MODIFY コマンド (F) を使用したエージェント・シャットダウンの制御

MODIFY コマンドを使用すると、 **fteStopAgent** コマンドの代替手段として、制御された方法でエージェントを停止できます。 エージェントは現在進行中の転送は完了しますが、新規転送は開始しません。

以下に例を示します。

```
F job_name,APPL=STOP
```

ここで、*job_name* は エージェント・プロセスを実行しているジョブです。

z/OS STOP コマンド (P) を使用した即時エージェント・シャットダウン

STOP コマンドは、 **-i** パラメーターを指定して **fteStopAgent** コマンドを使用した場合の即時停止に相当します。 エージェントは、現在ファイルを転送中であっても、 即時停止されます。

以下に例を示します。

```
P job_name
```

ここで、*job_name* は エージェント・プロセスを実行しているジョブです。

関連資料

201 ページの『z/OS での MFT エージェントの開始』

z/OS では、 z/OS UNIX System Services セッションから **fteStartAgent** コマンドを実行するだけでなく、対話式セッションを必要とせずに JCL から開始タスクとしてエージェントを開始することもできます。

MFT agent.properties ファイル

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。

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。

- 1 分ごと
- 毎時
- 日次
- 週次
- 月次
- 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、 214 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、211 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

211 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 WebSphere MQ 7.5 以降では、IBM MQ Explorer を実行可能な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。

212 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 WebSphere MQ 7.5 以降では、IBM MQ Explorer を実行可能な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

210 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。

212 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

210 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。

213 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。

298 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

210 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。

212 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。 詳しくは、225 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。

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。

V 9.2.2 Long Term Support の場合、 IBM MQ 9.2.2 の前の Continuous Delivery の場合、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 9.2.2 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、218 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。

V 9.2.0 IBM MQ 9.1.5 以降、リソース・モニターは ScheduledExecutorService を使用することで、前のポーリングが完了してから、設定されたポーリング間隔が経過して初めて、次のポーリングを開始するようになりました。 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。

V 9.2.0 IBM MQ 9.1.3 以降、ファイルの転送が失敗した場合に、リソース・モニターのヒストリーをクリアできるようになりました。その結果、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、別の転送要求を送信できます。 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。 詳しくは、242 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、225 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、223 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、225 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、219 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、233 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

Long Term Support の場合、 IBM MQ 9.2.2 の前の Continuous Delivery の場合、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、198 ページの『MFT エージェントの開始』 および 203 ページの『MFT エージェントの停止』 を参照してください。

V 9.2.2 IBM MQ 9.2.2 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。

V 9.2.2 詳細については、MFT リソース・モニターの開始と MFT リソース・モニターの停止を参照してください。

V 9.2.2

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
V 9.2.2 **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。
V 9.2.2 **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、先に **fteStopMonitor** の呼び出しが行われたかどうかにかかわらず、エージェント始動の一部として開始します。
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、240 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

IBM MQ 9.1.0 以降、Managed File Transfer にはリソース・モニター・ロギングが組み込まれています。 詳しくは、236 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

225 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

219 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。

225 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。

231 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、220 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、222 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

225 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

生成された文書の<request> の後に <managedTransfer> エレメントが続きます。 このタスク定義を有効な <managedCall> 構造体に変換するには、以下のステップに従ってください。

手順

1. <managedTransfer>の開始タグと終了タグを <managedCall> タグで置き換えます。
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> と置き換えて、モニター・エージェント構成の詳細を一致させます。
4. <destinationAgent> および <trigger> エレメントを除去します。
5. <item> エレメントを削除する。
6. preSourceCall、 postSourceCall、 preDestinationCall、または postDestinationCall エレメントをすべて削除します。
7. <transferSet> エレメントに、新規 <call>...</call> 構造を挿入します。 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、 4 つまでのコマンド呼び出しを挿入することもできます。 この場合は、 <metaDataSet> エレメントと<item>エレメントの間に、以下の呼び出しエレメントの選択を挿入します。

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、前の例で説明されているように、 <command> エレメント構造を取ります。 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

225 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

219 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。

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 231 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、 236 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。
CurrentTimeStamp UTC	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
228 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`
${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`

表 14. 負のトークン索引の使用例 (続き)	
変数の指定	**変数置換後**
${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

225 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。

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。

エージェントが再始動された場合、モニターは、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、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

225 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、 235 ページの『詳細オプション』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。

詳細オプション

-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
```

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。

Number	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

Number	イベント	ログ・レベル	説明
			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。
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。

手順

- **fteSetAgentLoglevel** を使用してリソース・モニターのロギングのオン/オフを切り替える場合は、fteSetAgentLogLevel を参照して、**logMonitor** パラメーターの説明や各種オプションの使用例を確認してください。
- 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

V 9.2.2 MFT リソース・モニターの開始

IBM MQ 9.2.2 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

215 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

239 ページの『MFT リソース・モニターの停止』
IBM MQ 9.2.2 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)

V 9.2.2 MFT リソース・モニターの停止

IBM MQ 9.2.2 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

215 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

238 ページの『MFT リソース・モニターの開始』
IBM MQ 9.2.2 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。

IBM MQ 9.1 以降、**-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 の形式で名前が指定された別個の 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

V 9.2.0 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 (215 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。

しかし、IBM MQ 9.1.3 以降では、**fteClearMonitorHistory** コマンドか 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できるようになりました。 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 に付属のサンプル V 9.2.0 SCSQFCMD メンバーには、モニターの履歴を消去する 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** コマンドの使用方法 (例を含む) について詳しくは、fteClearMonitorHistory: リソース・モニター履歴の消去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

245 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

243 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。

ファイルからメッセージへの転送およびメッセージからファイルへの転送を実行するには、転送のソース・エージェントと宛先エージェントの両方のバージョンが、IBM WebSphere MQ 7.5 以降であるか、IBM WebSphere MQ File Transfer Edition 7.0.3 以降である必要があります。 メッセージからファイルへの転送に関しては、255 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。

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。

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 という名前で、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 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 AGENT_NEPTUNE はファイル /tmp/single_record.txt からデータを読み込み、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。

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。

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
```

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
```

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
```

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。

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、 252 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

251 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.
```

V 9.2.2 V 9.2.0.2 IBM MQ 9.2.0 Fix Pack 2 以降および IBM MQ 9.2.2 では、区切り文字サイズ・チェック・エラーが原因でファイルからの転送が失敗すると、空のメッセージが 1 件のみ送信されます。

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。

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、246 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。

IBM MQ 9.1.0 以降、メッセージからファイルへの転送では、IBM MQ の以前のバージョンの破壊的 GET とは異なり、ソース・エージェントがソース・キューのメッセージを参照するようになりました。 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。

IBM MQ 9.0.0 Fix Pack 2 および IBM MQ 9.0.4 以降、 Managed File Transfer が更新され、転送要求 XML ペイロード内の転送 ID と groupId 属性の値の比較検査 (以前に APAR IT18213 によって削除されたもの) が復元されるようになりました。 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。

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。

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。

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。

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。

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。

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。

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>
```

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。

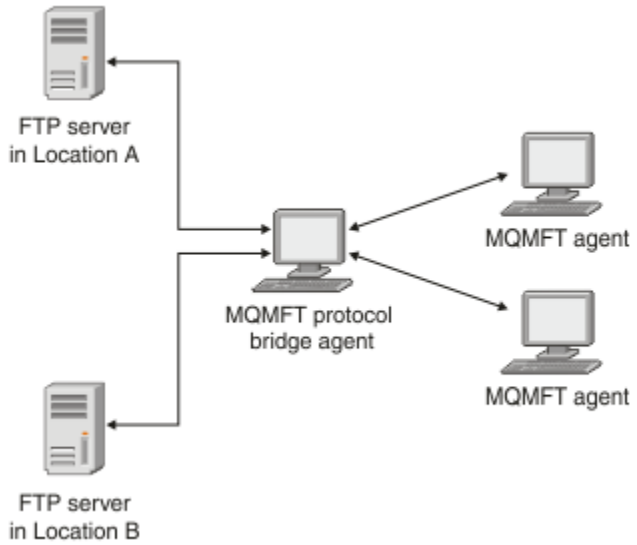
プロトコル・ブリッジは、Managed File Transfer のサービス・コンポーネントの一部として使用可能です。 MFT を実行する単一のシステムに、さまざまなファイル・サーバーに接続する複数の専用エージェントを作成することができます。

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、269 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。

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、266 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、『267 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

260 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 インターフェースの使用が推奨されますが、ProtocolBridgePropertiesExit.java インターフェースもサポートされています。 IBM WebSphere MQ File Transfer Edition から ProtocolBridgePropertiesExit.java インターフェースを既に実装していた場合は、IBM WebSphere MQ 7.5 以降でそれを使用できます。 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。

注 : IBM WebSphere MQ File Transfer Edition (FTE) は、サポートされなくなりました。 IBM MQ で FTE から Managed File Transfer コンポーネントにマイグレーションする場合は、Managed File Transfer のマイグレーション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.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。

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。

z/OS を使用している z/OS プラットフォーム、IBM WebSphere MQ 7.5 またはそれ以前のプラットフォームで ProtocolBridgeCredentials.xml ファイルを作成した場合、ファイルを編集する前に、ファイルタグを設定する必要があります。 次のコマンドを実行して、ファイルが ASCII コンテンツを持つものとしてマークを付けます。

```
chtag -t -c ISO8859-1 ProtocolBridgeCredentials.xml
```

注 : z/OS z/OS では、プロトコル・ブリッジ資格情報ファイルをデータ・セットに保管することができます。このファイルは、ユーザーが .xml ファイルの名前を指定することができます。

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。

   IBM WebSphere MQ File Transfer Edition 7.0.4 用に作成されたプロトコル・ブリッジ・エージェントには、ProtocolBridgeProperties.xml ファイル (または関連するユーザー出口) が含まれていないため、 IBM WebSphere MQ File Transfer Edition 7.0.4 Fix Pack 1 以降のサーバー名には自動的にサーバーのホスト名が割り当てられます。 したがって、更新された ProtocolBridgeCredentials.xml ファイルを <server> エントリーで使用すると、サーバーのホスト名に対応する名前が一致します。

   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、274 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。

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、273 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_name\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_name\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。

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 275 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。

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関する説明が必要な場合は、IBM Developer 記事の Configuring Secure Sockets Layer connectivity in IBM WebSphere MQ File Transfer Edition を参照するか、Oracle 社の keytool の資料 で keytool に関する情報を参照してください。

   注 : IBM WebSphere MQ File Transfer Edition (FTE) は、サポートされなくなりました。 IBM MQ で FTE から Managed File Transfer コンポーネントにマイグレーションする場合は、Managed File Transfer のマイグレーション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、262 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、269 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。

V 9.2.1 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)。

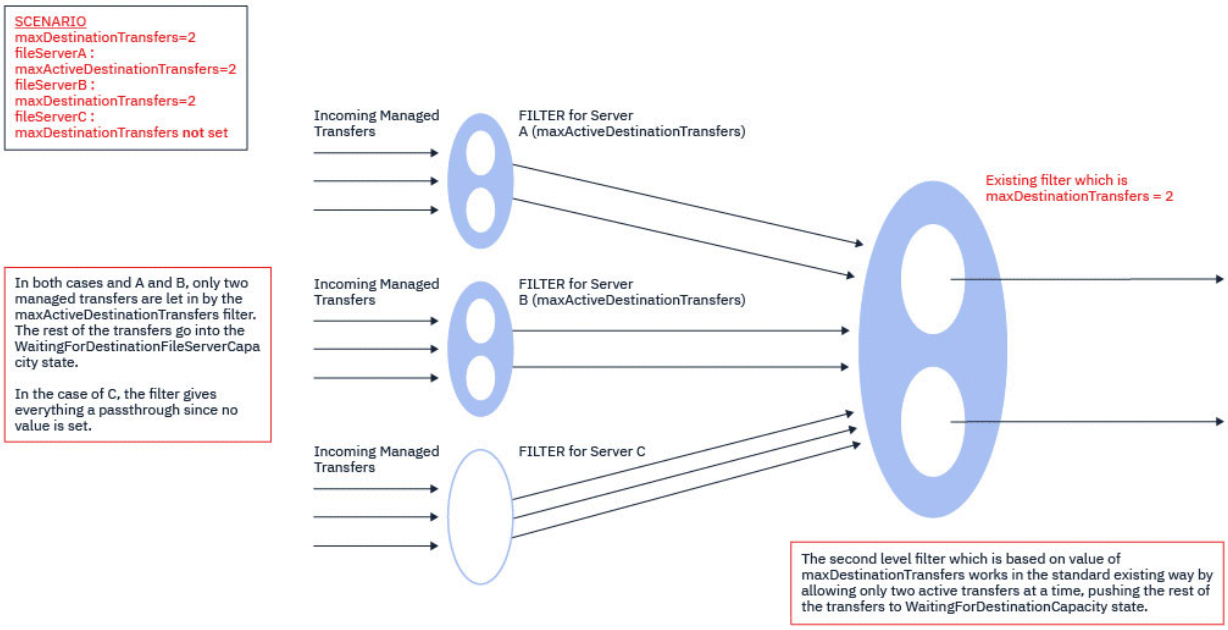
  これは、 fileServerA に障害が発生した場合に役立ちます。 fileServerA が再び稼働したら、 **maxActiveDestinationTransfers** の値を 5 に増やして、可能な宛先転送の全容量を許可することができます。
- プロトコル・ブリッジ・エージェントから fileServerB へのアクティブな宛先転送の最大数は 5 です。

  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・サーバーの **maxActiveDestinationTransfers** の値を **maxDestinationTransfers** の値よりも小さく設定して空きスロットを残すか、または、
- **maxActiveDestinationTransfers** 属性の値を、すべてのエンドポイント・サーバー間に等しく配分します。

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数が 10 に達すると、11 番目の管理対象転送はプロトコル・ブリッジ・エージェントにより失敗します。
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数が 10 に達すると、11 番目の管理対象転送はキューに入り、スロットが解放されるのを待機します。
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。

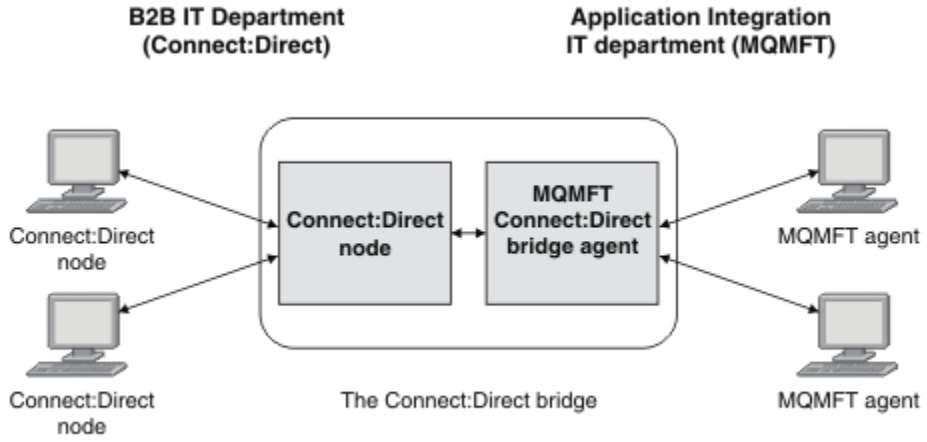
BFGSS0087E
このエラー・メッセージは、**maxActiveDestinationTransfers** 属性に指定された最大宛先転送数を超えたために、プロトコル・ブリッジ・エージェントが管理対象転送を拒否し、これを failed とマークした場合に、Explorer および audit.xml ファイルに記録されます。

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 9.2.1 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。

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オペレーティング・システム

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 z/OS 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。

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

282 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

285 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

282 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 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。 ソース・エージェントが IBM WebSphere MQ File Transfer Edition 7.0.4 以降の場合は、**-ds** パラメーターを使用することもできます。 ソース・エージェントは、FTE_ZOS1 という名前であり、IBM WebSphere MQ File Transfer Edition 7.0.3 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。 転送されるデータ・セットは、FTE_ZOS1 が配置されているシステム上の `//FTEUSER.SOURCE.LIB` に置かれています。 データ・セットは、CD_ZOS2 が配置されているシステム上のデータ・セット `//CDUSER.DEST.LIB` に転送されます。

注 : IBM WebSphere MQ File Transfer Edition (FTE) は、サポートされなくなりました。 IBM MQ で FTE から Managed File Transfer コンポーネントにマイグレーションする場合は、Managed File Transfer のマイグレーションを参照してください。

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

282 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

z/OS Connect:Direct ノードとの間のデータ・セット転送

関連資料

z/OS 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

282 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

284 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。

289 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。

285 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。

288 ページの『Connect:Direct ノードからの複数ファイルの転送』
Connect:Direct ブリッジを使用して、 Connect:Direct ノードから Managed File Transfer Agent に複数の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、 *connect_direct_node_name*:*file_path*.という形式で 1 つ以上のソース指定を指定することにより、複数ファイル転送のソースとして Connect:Direct ノードを指定できます。

関連資料

MFT `agent.properties` ファイル

z/OS Connect:Direct ノードからの複数ファイルの転送

Connect:Direct ブリッジを使用して、 Connect:Direct ノードから Managed File Transfer Agent に複数の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、 *connect_direct_node_name*:*file_path*.という形式で 1 つ以上のソース指定を指定することにより、複数ファイル転送のソースとして Connect:Direct ノードを指定できます。

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_Z という名前で、z/OS システム上で実行されています。 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 `/in/file1`、 `/in/file2`、および `/in/file3` にあります。 これらのファイルは、 FTE_Z が実行されているシステム上の区分データ・セット `//OBJECT.LIB` に転送されます。

手順

ソース指定の値として *connect_direct_node_name*:*file_path* の形式を使用し、**-sa** 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、ファイルの転送元とするノードです。

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_Z
                  -dp //'OBJECT.LIB' CD_NODE1:/in/file1
                  CD_NODE1:/in/file2 CD_NODE1:/in/file3
```

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 からの最初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_Z 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。 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

282 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

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。

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、297 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

282 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。

297 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、292 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。

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

291 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。 ソース指定のフォーマットについては、**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 ノードと同じ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。 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、さらに追加のパラメーターがあります。

-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

296 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 295 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

295 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

298 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

IBM MQ 9.1 から、転送リカバリー・タイムアウト・パラメーターをエージェントの `agent.properties` ファイルに追加することによって、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ことができます。 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。 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` ファイル内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。 設定しない場合、転送はデフォルトの動作に従います。 これは、 IBM MQ 9.1 より前の Managed File Transfer ソース・エージェントのデフォルト動作と同じです。この場合、エージェントは、正常に完了するまで、停止した転送のリカバリーを試行し続けます。

関連概念

297 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。

V 9.2.0 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

297 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、301 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 (301 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>
```

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を実行します。

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、テレメトリー (MQXR) サービスを開始する前に、 `SYSTEM.MQTT.TRANSMIT.QUEUE` を作成し、それに対するアクセス権限を構成します。

   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。 309 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. 311 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. 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 qMgr
   ```

   305 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. 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。
7. 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 305 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。
8. 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。

LTS

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+"') +
STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')
```

V 9.2.4
```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+" -sf "[DEFAULT]"') +
STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')
```

注 : V 9.2.4 IBM MQ 9.2.4 以降、**-sf** フラグは、TLS チャネル・パスフレーズを暗号化するために使用される鍵を入れた資格情報鍵ファイル名を指定します。 詳しくは、MQTT TLS チャネルのパスフレーズの暗号化を参照してください。

Windows 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を実行します。

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、テレ

メトリー (MQXR) サービスを開始する前に、 SYSTEM.MQTT.TRANSMIT.QUEUE を作成し、それに対するアクセス権限を構成します。

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。 309 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. 311 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. 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
   ```

   307 ページの『Creating SYSTEM.MQXR.SERVICE』のコード例も参照してください。
6. 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。
7. 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 307 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。
8. 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。

```
V 9.2.4
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
```

```
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+\." -g "+MQ_DATA_PATH+\." -sf "[DEFAULT]"') +
STOPCMD('+MQ_INSTALL_PATH+\mqxr\bin\endMQXRService.bat') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+\mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+\mqxr.stderr')
```

注 : V 9.2.4 IBM MQ 9.2.4 以降、**-sf** フラグは、TLS チャネル・パスフレーズを暗号化するために使用される鍵を入れた資格情報鍵ファイル名を指定します。 詳しくは、MQTT TLS チャネルのパスフレーズの暗号化を参照してください。

Windows Linux AIX 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。309 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
```

図 *16. Windows* でのクラスター・トピックの定義

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 ;
```

図 *17. MQTT v3* クライアント宛先にメッセージを送信するための *MQI* オブジェクト記述子

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。

```
javax.jms.Destination jmsDestination =
(javax.jms.Destination)jmsFactory.createQueue
("queue://ClientIdentifier/name");
```

図 *18. MQTT v3* クライアントにメッセージを送信するための *JMS* 宛先

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。310 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。

- 312 ページの『アクセス制御なし』
- 313 ページの『粗い細分度のアクセス制御』
- 313 ページの『中程度の細分度のアクセス制御』
- 313 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デジタル証明書を提供することにより、MQTT クライアントを WebSphere MQ に対して認証できます。 WebSphere 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

315 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』
IBM MQ 管理者は、サーバーのテレメトリー・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。

関連タスク

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*

IBM MQ 管理者は、サーバーのテレメトリー・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、317 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

317 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』
IBM MQ 管理者は、サーバーのテレメトリー・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)
CipherSpec および CipherSuite

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、トラストストアまたは JRE cacerts ストアに置かれている証明書に対して、サーバーから送信された証明書と一致する検査を実行します。

JRE 証明書ストアは JKS ファイル cacerts です。 これは JRE InstallPath\lib\security\にあります。 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パブリック証明書を、他の Java アプリケーションが使用する証明書と分けて保管する場合は、クライアントのトラストストアを使用してください。 クライアント側で実行されているすべての Java アプリケーションに、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てください。 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

IBM MQ 管理者は、サーバーのテレメトリー・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、 317 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

315 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』
IBM MQ 管理者は、サーバーのテレメトリー・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

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)
CipherSpec および CipherSuite

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

317 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』

IBM MQ 管理者は、サーバーのテレメトリー・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、316 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。314 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。316 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。318 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。 サーバーが、既に cacerts に保管されているルート証明書を持っている既知の CA が発行した証明書を使用している場合は、 トラストストアを構成しないことがあります。

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。

IBM 鍵管理ユーティリティを実行する IBM MQ コマンド strmqikm を使って cacerts 開くことができます。 パスワード changeit を使用して、cacerts を JKS ファイルとして開きます。 パスワードを変更して、この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

許可

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。 **strmqikm** コマンドを使用して **ikeyman** を始動してみるか、または、 IBM developer kits, Security information から、強度はあるものの権限が制限されている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。

323 ページの図 19 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 *19. 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 *20. AcceptAllProvider.java*

```
protected static class AcceptAllTrustManagerFactory extends
javax.net.ssl.TrustManagerFactorySpi {
public AcceptAllTrustManagerFactory() {}
protected void engineInit(java.security.KeyStore keystore) {}
protected void engineInit(
javax.net.ssl.ManagerFactoryParameters parameters) {}
protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
}
```

図 *21. AcceptAllTrustManagerFactory.java*

```
protected static class AcceptAllX509TrustManager implements
javax.net.ssl.X509TrustManager {
public void checkClientTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Client authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public void checkServerTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Server authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public java.security.cert.X509Certificate[] getAcceptedIssuers() {
return new java.security.cert.X509Certificate[0];
}
private static void report(String string) {
System.out.println(string);
}
}
```

図 *22. 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。 325 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、325 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。

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(414D5120514D31202020202020202020707E0A565C2D0020)
   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 の管理者から提供されたチャネル 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 ユーザーおよびパスワードに照らして個々のユーザーを認証し、成功した場合に共通 ID AMQPUser を採用するステップの例を示します。

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。

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。

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。 例えば、

```
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** コマンドを実行することにより、クライアントが使用している接続を検出します。 例えば、

```
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。 例えば、

```
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 332 ページの『2』 で使用した値と同じです。
4. コマンド行 1 に `Hello world` と入力します。 COMMINFO オブジェクトに指定されているポートと 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。

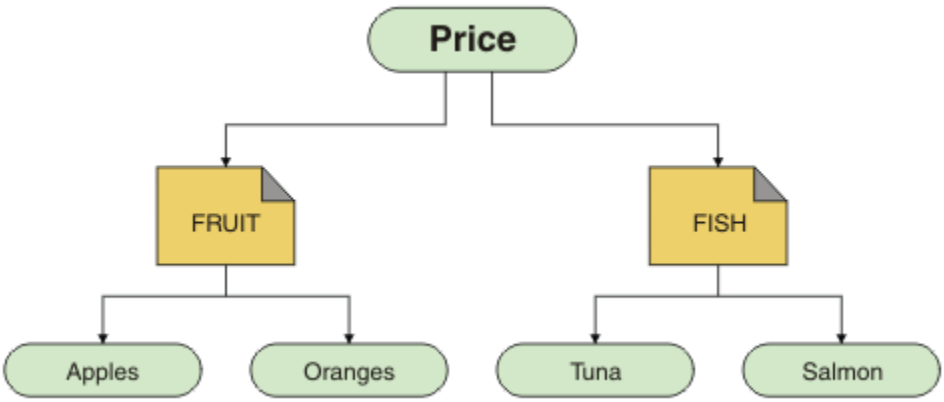
以下の図には、2 つの可能なマルチキャスト・データ・ストリームがあります。

```
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、 334 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。

V 9.2.0 V 9.2.0 XMS .NET マルチキャスト・メッセージング (RMM を使用) は、IBM MQ 9.2 から非推奨になり、 XMS .NET の将来のリリースで削除される予定です。

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

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**
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
Priority	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
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
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。

無効化
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
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

無効化
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。

動的
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

IBM i 上の IBM MQ を管理するために使用できる方法について説明します。

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 を管理する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。

- 341 ページの『CL コマンドを使用した IBM MQ for IBM i の管理』
- 355 ページの『IBM MQ for IBM i 管理の代替方法』

- 360 ページの『IBM i でのワーク・マネジメント』

関連概念

367 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』
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。

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

関連タスク

IBM i での構成情報の変更

IBM i でのセキュリティーのセットアップ

関連資料

411 ページの『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。

148 ページの『送達不能キュー・ハンドラー (IBM i)』
IBM i 送達不能キュー・ハンドラーの説明と呼び出し方。

IBM MQ for IBM i アプリケーションの問題判別

IBM i でのインストール可能サービスとコンポーネント

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、11 ページの『MQSC コマンドによる管理』および 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、411 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、343 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、345 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、148 ページの『送達不能キュー・ハンドラー (IBM i)』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。 344 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 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、リモート管理などの他のさまざまな管理方法を使用することもできます。

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、341 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。

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管理方法を使用できます。

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。

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。 次のとおりです。

1. PCF メッセージを使用してプログラムを作成する。 357 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。
2. MQAI を使用して、PCF メッセージを送信するプログラムを作成する。 36 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。
3. IBM MQ for Windows で使用可能な IBM MQ エクスプローラーを使用する。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、正しい PCF メッセージが生成されます。 357 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。
4. **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。 355 ページの『IBM i での MQSC コマンドによる管理』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 での MQSC コマンドによる管理

この情報を使用して、MQSC コマンドについて、およびそれらを使用して IBM MQ for IBM i を管理する方法について学習します。

IBM MQ スクリプト (MQSC) コマンドは、判読可能な形式、つまり EBCDIC テキストで書きます。 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。

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 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')
```

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でき、また既存のメンバーを保守できます。

MQSC コマンドは、RUNMQSC を発行するか、または、次のようにして対話式に入力することもできます。

1. キュー・マネージャー名を入力し、Enter キーを押して **WRKMQM** 結果パネルにアクセスする。
2. このパネルで F23=More options を選択します。
3. 356 ページの図 23 に示すパネルで、アクティブなキュー・マネージャーに対してオプション 26 を選択する。

そのような MQSC セッションを終了するには、 end と入力します。

356 ページの図 23 は、MQSC コマンド・ファイルからの抜粋で、属性が指定された MQSC コマンド (DEFINE QLOCAL) を示しています。

```
.
.
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 *23. MQSC* コマンド・ファイル *myprog.in* からの抽出

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。 正符号 (+) は、コマンドが次の行に続くことを示します。

MQSC に指定するオブジェクト属性は、このセクションでは大文字 (例えば、RQMNAME) で示されます。ただし、実際には大文字小文字の区別はありません。

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、z/OS を含めて、他のプラットフォームでも使用できます。

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。

IBM i 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、36 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 IBM MQ for IBM i での IBM MQ Explorer 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、359 ページの図 24 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 *24.* 「*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。 361 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	関数
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 362 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 362 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、361 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
```

この

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 (373 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。

システムの構成の詳細については、IBM i での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)
```

この

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、373 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、370 ページの図 25 に示します。

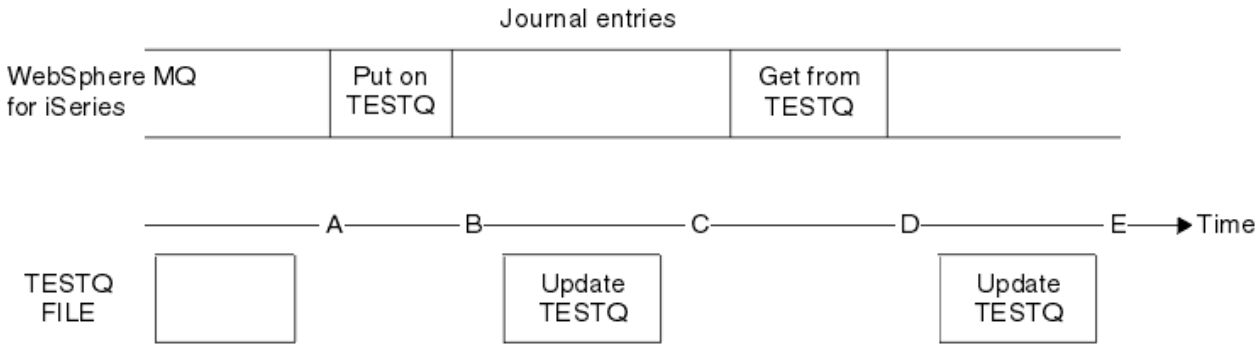


図 *25. MQM* オブジェクトの更新時のイベントのシーケンス

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、「367 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、373 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、QueueManager スタンザ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 373 ページの図 26 に示します。

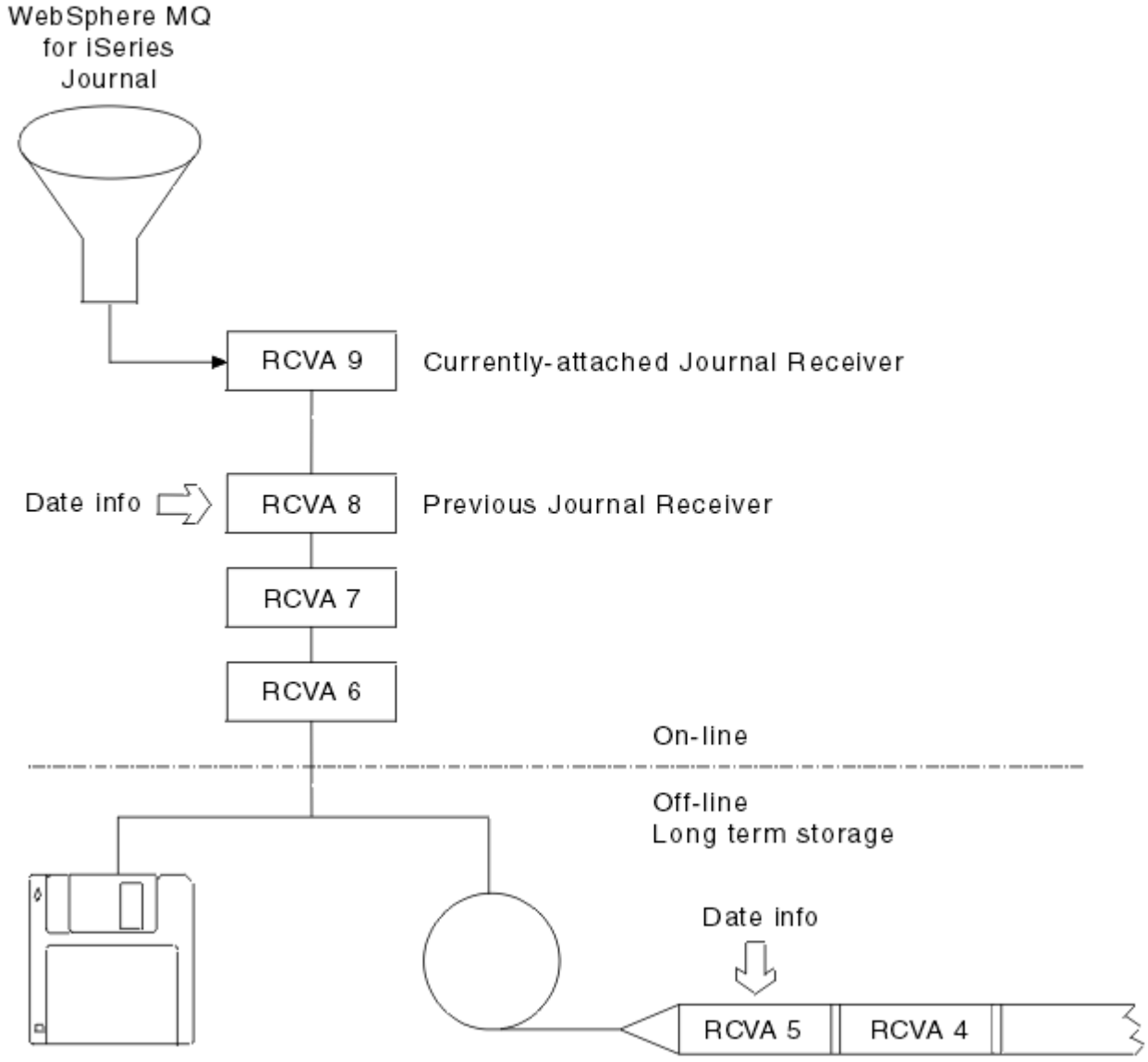


図 *26. ジャーナリング (IBM i)*

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 (367 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。

災害時回復の場合
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

392 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 396 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。383 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。383 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。383 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。383 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 の IBM MQ 9.1 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。

   385 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。

   384 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

405 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) 以下のコマンドを使用して、IASP をオンに変更します。

```
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。

388 ページの図 27 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。

388 ページの図 27 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 367 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。

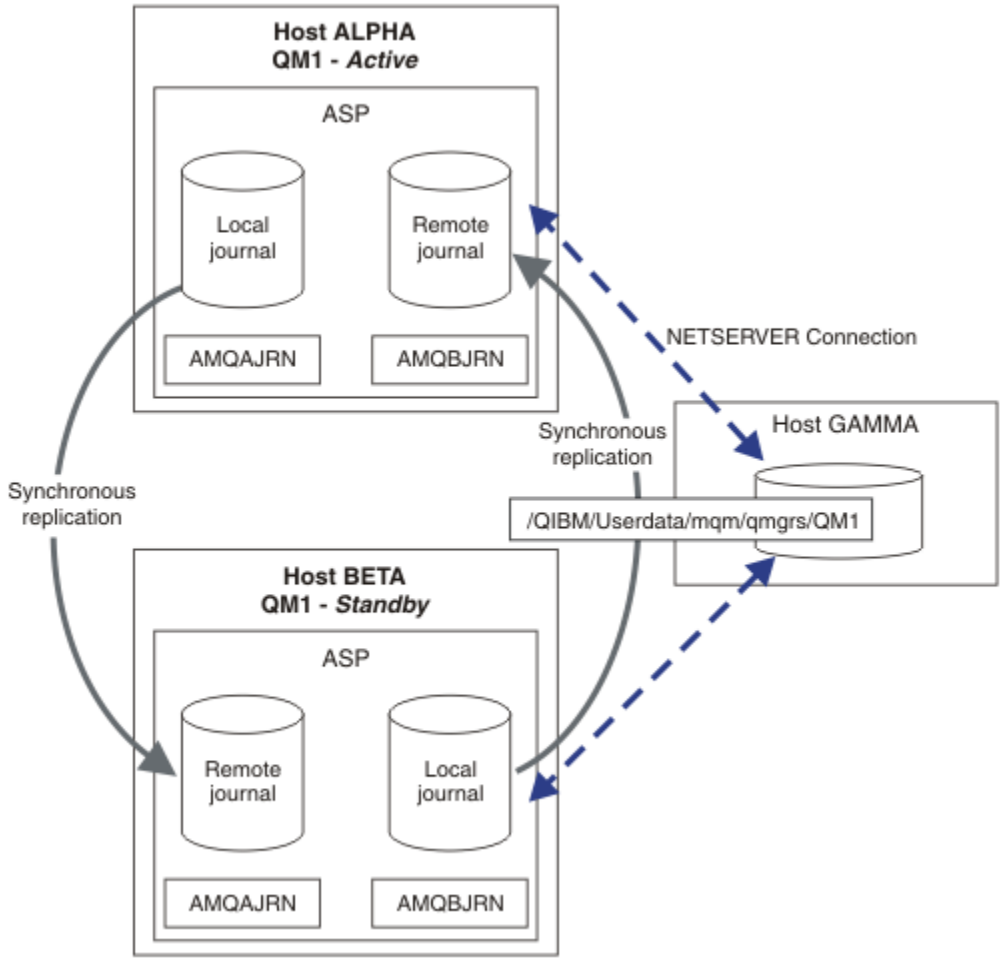


図 27. キュー・マネージャー・ジャーナルのミラーリング

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。

388 ページの図 27 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。

388 ページの図 27 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。 389 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、395 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。

- 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. 379 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、394 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、387 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

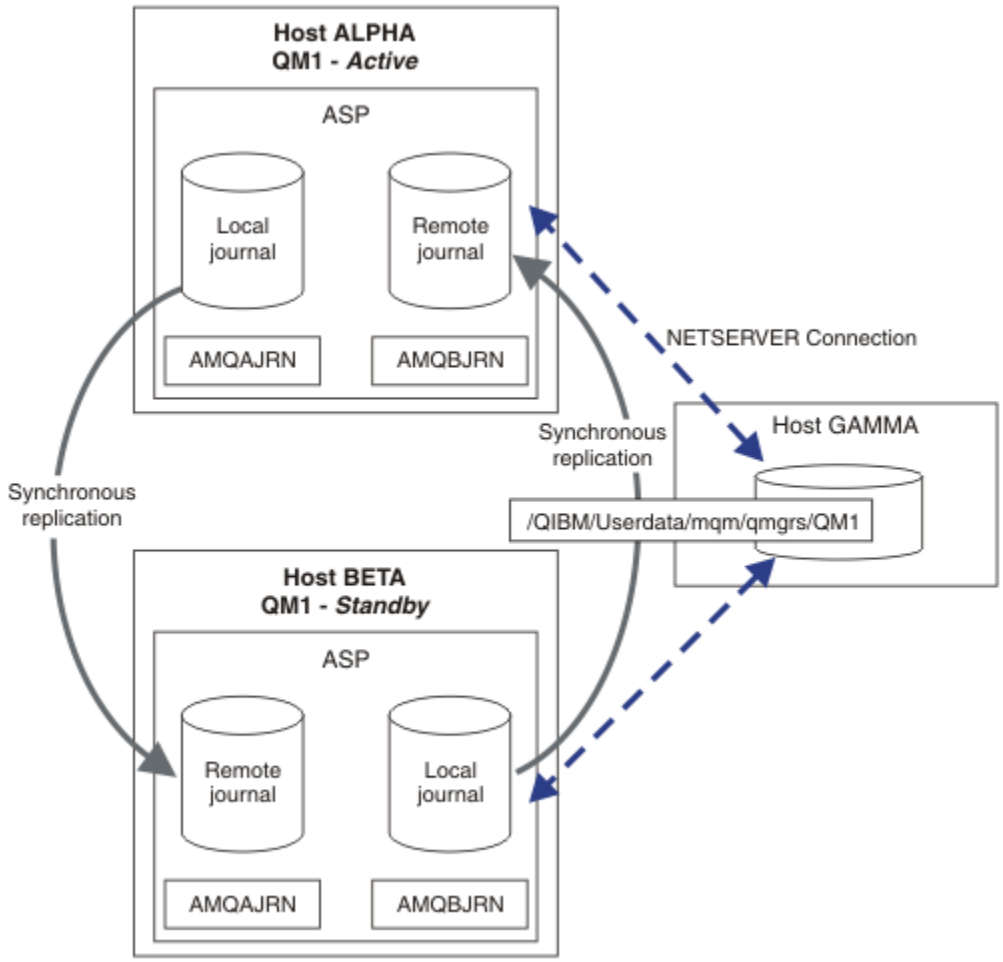


図 *28.* ミラーリングされたジャーナル構成

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 (400 ページの図 29 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 397 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 397 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 397 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 397 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. 379 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、394 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、387 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

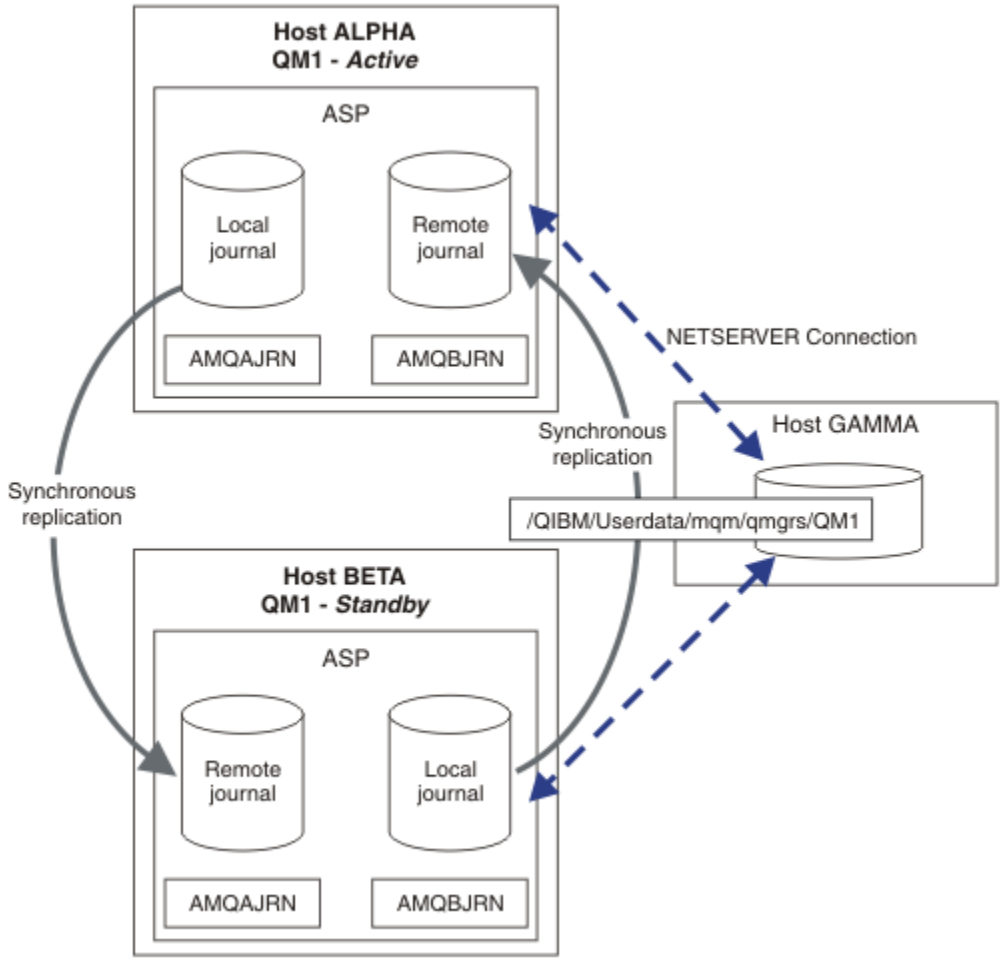


図 *29.* ミラーリングされたジャーナル構成

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。

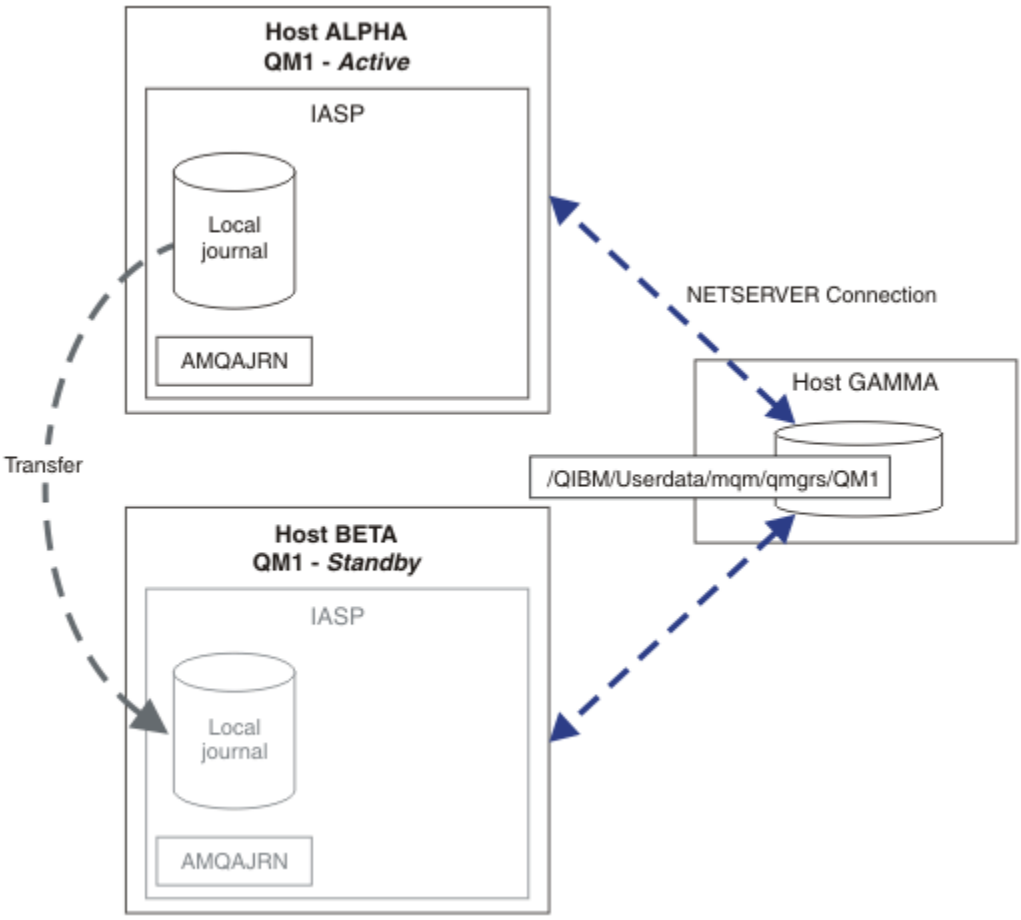


図 *30.* 独立 *ASP* を使ったキュー・マネージャー・ジャーナルの転送

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。 378 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、405 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、404 ページの図 31 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. 379 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

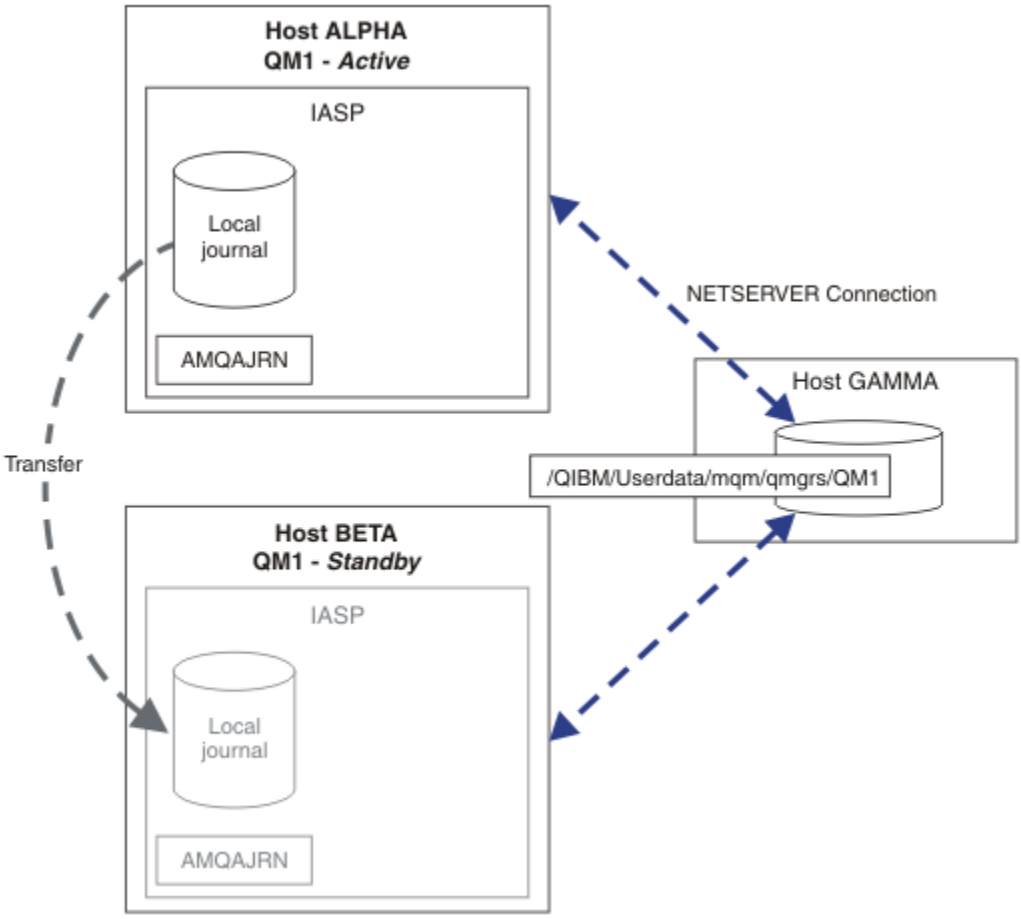


図 *31.* 独立 *ASP* を使ったキュー・マネージャー・ジャーナルの転送

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。 392 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 408 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。 392 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。

以下のコマンドを使用して、追加のキュー・マネージャー・インスタンスを追加および管理します。 これにより、`mqs.ini` ファイルが変更されます。

ADDMQMINF
このコマンドは、DSPMQMINF コマンドで `mqs.ini` ファイルから取り出された情報を使用して、異なる IBM i サーバー上に新規キュー・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 (412 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 (413 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 IBM MQ for z/OS の管理

キュー・マネージャーと関連リソースの管理には、それらのリソースをアクティブ化して管理するために頻繁に実行するタスクが含まれます。 キュー・マネージャーと関連リソースを管理するための最適な方法を選択してください。

IBM MQ for z/OS は、製品で提供される一連のユーティリティーおよびプログラムによって制御および管理できます。 IBM MQ スクリプト (MQSC) コマンドまたはプログラマブル・コマンド・フォーマット (PCF) を使用して、IBM MQ for z/OS を管理することができます。 IBM MQ for z/OS のコマンドの使用については、415 ページの『IBM MQ for z/OS へのコマンドの実行』を参照してください。

IBM MQ for z/OS は、システム管理に役立つ一連のユーティリティー・プログラムも提供しています。 各種のユーティリティー・プログラムとそれらの使用法については、424 ページの『IBM MQ for z/OS ユーティリティー』を参照してください。

IBM MQ for z/OS を管理する方法と、実行しなければならない場合がある様々な管理用タスクについて詳しくは、以下のリンクを参照してください。

関連概念

IBM MQ for z/OS の概念

関連タスク

121 ページの『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。

185 ページの『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。

7 ページの『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。

計画

z/OS での IBM MQ 環境の計画

構成

の構成 z/OS

IBM MQ for z/OS ユーティリティーの使用

関連資料

20 ページの『z/OS 上で MQSC コマンドを発行できるソース』
MQSC 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。

プログラマブル・コマンド・フォーマット・リファレンス

z/OS IBM MQ for z/OS へのコマンドの実行

キュー・マネージャーを制御するために IBM MQ スクリプト・コマンド (MQSC) をバッチ・モードまたは対話モードで使用することができます。

IBM MQ for z/OS では、MQSC コマンドをサポートしています。 このコマンドは次のソースから発行できます。

- z/OS コンソールまたは等価のもの (SDSF/TSO など)。

  z/OS コンソールを使用する場合は、コマンドの先頭に /cpf を追加する必要があります。ここで、cpf は、キュー・マネージャー・サブシステムのコマンド接頭部です。
- 初期設定入力データ・セット。
- 順次データ・セット内のコマンドのリストを処理する、 システムに提供されているバッチ・ユーティリティー、CSQUTIL。

  初期設定入力データ・セットまたは提供されたバッチ・ユーティリティーを使用する場合 /cpf をコマンドの先頭に追加する必要はありません。
- コマンド入力キューに対するメッセージとしてコマンドを送信することで、 適正に許可されたアプリケーション。 アプリケーションは、次のいずれかになります。
  - バッチ領域プログラム
  - CICS アプリケーション
  - IMS アプリケーション
  - TSO アプリケーション
  - 別の IBM MQ システム上のアプリケーション・プログラムまたはユーティリティー

419 ページの表 26 には、MQSC コマンドおよびそれらのコマンドを発行できる発行元について要約されています。

これらのコマンドの機能の大部分は、IBM MQ for z/OS 操作パネルと制御パネルから便利な方法で利用できます。

コマンドを (直接または間接的に) 使用してキュー・マネージャーのリソース定義に変更を加えた場合、その内容は、IBM MQ サブシステムの再始動があっても保存されます。

また、IBM MQ for z/OS はプログラム式コマンド・フォーマット (PCF) コマンドをサポートします。 このコマンドによって、IBM MQ を管理するためのアプリケーションを容易に作成できます。 MQSC コマンドは人間が読むことのできるテキスト形式であるのに対し、PCF はアプリケーションが テキスト・ストリングを解析する必要なく要求を作成して応答を読み取ることができるようにします。 MQSC コマンドと同様に、アプリケーションは PCF コマンドを、 コマンド入力キューにメッセージとして送信することにより発行します。 PCF コマンドの使用について、およびコマンドについての詳細は、「プログラマブル・コマンド・フォーマット・リファレンス」の資料を参照してください。

z/OS IBM MQ for z/OS での専用定義とグローバル定義

IBM MQ for z/OS でオブジェクトを定義するときに、その定義を他のキュー・マネージャーと共有するのか (グローバル 定義)、それともそのオブジェクト定義を 1 つのキュー・マネージャーだけで使用するのか (専用 定義) を選択できます。 このような選択のことをオブジェクトの属性指定 といいます。

グローバル定義

キュー・マネージャーがキュー共有グループに属している場合は、作成するオブジェクト定義をそのグループの他のメンバーと共有できます。 その場合は、オブジェクトを 1 回だけ定義すれば十分であり、結果的にシステム全体で必要な定義の総数を削減できます。

グローバル・オブジェクト定義は、共有リポジトリー (Db2® 共有データベース) に格納され、キュー共有グループに含まれているすべてのキュー・マネージャーからアクセスできる状態になります。 これらのオブジェクトは、GROUP という属性指定になります。

専用定義

1 つのキュー・マネージャーだけで必要なオブジェクト定義を作成する場合や、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ない場合は、キュー共有グループの他のメンバーと共有しないオブジェクト定義を作成できます。

専用オブジェクト定義は、定義元のキュー・マネージャーのページ・セット 0 に格納されます。 これらのオブジェクトは、QMGR という属性指定になります。

専用定義は、CF 構造以外のすべてのタイプの IBM MQ オブジェクト (つまり、チャネル、名前リスト、プロセス定義、キュー、キュー・マネージャー、ストレージ・クラス定義、認証情報オブジェクト) で作成できます。グローバル定義は、キュー・マネージャー以外のすべてのタイプのオブジェクトで作成できます。

グループ・オブジェクトの定義は、 その定義を使用するそれぞれのキュー・マネージャーのページ・セット 0 に、IBM MQ によって自動的にコピーされます。 必要があれば、定義のコピーを一時的に変更することも可能です。さらに、IBM MQ では、必要に応じて、リポジトリー・コピーに基づいてページ・セット・コピーをリフレッシュすることもできます。

IBM MQ は常に、始動時にリポジトリー・コピーに基づいてページ・セット・コピーをリフレッシュしようとします (チャネル・コマンドの場合は、チャネル・イニシエーターの再始動時になります)。さらに、グループ・オブジェクトが変更された場合にも、常にリフレッシュしようとします。

注 : 定義のコピーは、定義のコピーを作成した後にグループの定義が変更された場合にのみ、グループの定義からリフレッシュされます。

これにより、キュー・マネージャーが非活動状態だった時点でなされた変更を含め、 リポジトリーのバージョンがページ・セット・コピーに常に反映されるようになります。 コピーのリフレッシュは、DEFINE REPLACE コマンドの生成によって行われるので、リフレッシュが実行されない場合もあります。例えば、以下のような場合です。

- キューのコピーが開いていると、そのキューの使用方法を変更するリフレッシュ操作は失敗します。
- キューのコピーにメッセージが含まれていると、そのキューを削除するリフレッシュ操作は失敗します。
- キューのコピーの変更のために、FORCE 付きの ALTER が必要な場合。

上記の場合についてはコピーに対してリフレッシュは実行されませんが、 それ以外のすべてのキュー・マネージャーのコピーについては実行されます。

キュー・マネージャーのシャットダウン後に、キュー・マネージャーをスタンドアロン・モードで再始動すると、オブジェクトのローカル・コピーは削除されます (ただし、キューに関連メッセージがある場合などは例外です)。

ローカル・キューだけに該当する第 3 のオブジェクト属性指定があります。 この場合は、共有キューを作成できます。 共有キューの定義は、共有リポジトリーに格納され、キュー共有グループに含まれているすべてのキュー・マネージャーからアクセスできる状態になります。 さらに、共有キューに含まれているメッセージも、キュー共有グループに含まれているすべてのキュー・マネージャーからアクセスできる状態になります。 これについては、共用キューおよびキュー共用グループで説明し 共有キューは、SHARED というオブジェクト属性指定になります。

スタンドアロン・モードで始動するキュー・マネージャーと、キュー共有グループのメンバーとして始動するキュー・マネージャーについて、それぞれのオブジェクト属性指定オプションの効果を以下の表にまとめます。

後処理	**スタンドアロン・キュー・マネージャー**	**キュー共有グループのメンバー**
QMGR	オブジェクト定義がページ・セット 0 に格納されます。	オブジェクト定義がページ・セット 0 に格納されます。
GROUP	該当しません。	オブジェクト定義が共有リポジトリーに格納されます。 ローカル・コピーがグループ内の各キュー・マネージャーのページ・セット 0 に格納されます。
SHARED	該当しません。	キュー定義が共有リポジトリーに格納されます。 メッセージがグループ内のすべてのキュー・マネージャーからアクセスできる状態になります。

グローバル定義の操作

共有リポジトリーに格納されているオブジェクトの定義を変更する場合は、リポジトリーに格納されているバージョンを変更するのか、それともページ・セット 0 に格納されているローカル・コピーを変更するのかを指定する必要があります。 そのために、コマンドの一部としてオブジェクト属性指定のオプションを使用してください。

z/OS z/OS での異なるキュー・マネージャーに対するコマンドの送信

コマンドの有効範囲 を使用して、どのキュー・マネージャーに対してコマンドを実行するかを制御することができます。

コマンドを、そのコマンドが入力されているキュー・マネージャーに対して実行するのか、 それともキュー共有グループ中の他のキュー・マネージャーに対して実行するのかを選択できます。 また、特定のコマンドを、キュー共有グループ中のすべてのキュー・マネージャーに並行して発行することも選択できます。これは、MQSC コマンドと PCF コマンドの両方で可能です。

いずれを選択するかは、コマンドの有効範囲 によって決められます。 コマンドの有効範囲をオブジェクトの属性指定と一緒に使用して、オブジェクトのどのバージョンを処理するかを決めます。

例えば、オブジェクトの特定の属性を変更したいと思っており、 その定義は共有リポジトリーに保持されているとしましょう。

- 1 つのキュー・マネージャー上のバージョンだけに変更を加え、 リポジトリーや他のキュー・マネージャー上のバージョンには変更を加えないようにすることができる。
- 今後のユーザーのために共有リポジトリー中のバージョンには変更を加え、 既存のコピーは未変更のままにすることができる。
- 共有リポジトリー中のバージョンに変更を加え、 さらにキュー共有グループ中のキュー・マネージャーのうち、 ページ・セット 0 にオブジェクトのコピーがあるものすべてに変更内容を即時に反映できる。

コマンドの有効範囲を使用して、 コマンドをこのキュー・マネージャーに対して実行するのか、 別のキュー・マネージャーに対して実行するのか、 それともすべてのキュー・マネージャーに対して実行するのかを指定してください。 オブジェクト属性指定を使用して、 操作しようとしているオブジェクトが共有リポジトリー中にあるのか (グループ・オブジェクト)、 それともページ・セット 0 上のローカル・コピーなのか (キュー・マネージャー・オブジェクト) 指定してください。

共有キューを処理する場合はコマンドの有効範囲とオブジェクトの属性指定を指定する必要はありません。なぜなら、キュー共有グループ内の各キュー・マネージャーは共有キューを単一のキューとして扱うからです。

z/OS IBM MQ for z/OS のコマンドの要約

このトピックは、主な MQSC コマンドおよび PCF コマンドのリファレンスとして使用します。

418 ページの表 25 は、IBM MQ オブジェクトを変更、定義、削除、および表示するために IBM MQ for z/OS で使用できる MQSC コマンドおよび PCF コマンドを要約したものです。

表 *25.* 主な *MQSC* コマンドおよび *PCF* コマンドの要約 *(*オブジェクト・タイプ別*)*

MQSC コマンド	**ALTER**	**DEFINE**	**DISPLAY**	**DELETE**
PCF コマンド	**変更 (Change)**	**作成/コピー**	**照会**	**削除**
AUTHINFO	X	X	X	X
CFSTATUS			X	
CFSTRUCT	X	X	X	X
CHANNEL	X	X	X	X
CHSTATUS			X	

表 25. 主な *MQSC* コマンドおよび *PCF* コマンドの要約 (オブジェクト・タイプ別) (続き)				
MQSC コマンド	**ALTER**	**DEFINE**	**DISPLAY**	**DELETE**
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

表内の記号の意味:

- M = MQSC のみ
- P = PCF のみ
- X = 両方

他の IBM MQ リソースを管理したり、418 ページの表 25 に示されている操作に加えて他の操作を実行したりすることが可能な MQSC コマンドおよび PCF コマンドが、他にも多数存在します。

419 ページの表 26 に、すべての MQSC コマンドと、各コマンドの発行元を示します。

- CSQINP1 初期設定入力データ・セット
- CSQINP2 初期設定入力データ・セット
- z/OS コンソール (または同等のコンソール)
- SYSTEM.COMMAND.INPUT キューおよびコマンド・サーバー (アプリケーション、CSQUTIL、または CSQINPX 初期設定入力データ・セットから)

表 26. *MQSC* コマンドの実行元にできるソース				
コマンド	**CSQINP1**	**CSQINP2**	**z/OS コンソール**	**コマンド入力キューおよびサーバー**
ALTER AUTHINFO		X	X	X
ALTER BUFFPOOL		X	X	X
ALTER CFSTRUCT		X	X	X
ALTER CHANNEL		X	X	X
ALTER NAMELIST		X	X	X
ALTER PSID			X	X
ALTER PROCESS		X	X	X
ALTER QALIAS		X	X	X
ALTER QLOCAL		X	X	X

表 *26. MQSC* コマンドの実行元にできるソース *(*続き*)*

コマンド	CSQINP1	CSQINP2	z/OS コンソール	コマンド入力キューおよびサーバー
ALTER QMGR		X	X	X
ALTER QMODEL		X	X	X
ALTER QREMOTE		X	X	X
ALTER SECURITY	X	X	X	X
ALTER STGCLASS		X	X	X
ALTER SUB		X	X	X
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
DEFINE AUTHINFO		X	X	X
DEFINE BUFFPOOL	X	X		
DEFINE CFSTRUCT		X	X	X
DEFINE CHANNEL		X	X	X
DEFINE LOG			X	X
DEFINE NAMELIST		X	X	X
DEFINE PROCESS		X	X	X
DEFINE PSID	X		X	X
DEFINE QALIAS		X	X	X
DEFINE QLOCAL		X	X	X
DEFINE QMODEL		X	X	X
DEFINE QREMOTE		X	X	X
DEFINE STGCLASS		X	X	X
DEFINE SUB			X	X
DEFINE TOPIC		X	X	X
DELETE AUTHINFO		X	X	X
DELETE BUFFPOOL			X	X
DELETE CFSTRUCT		X	X	X
DELETE CHANNEL			X	X
DELETE NAMELIST		X	X	X
DELETE PROCESS		X	X	X
DELETE PSID			X	X

表 *26. MQSC* コマンドの実行元にできるソース *(*続き*)*

コマンド	CSQINP1	CSQINP2	z/OS コンソール	コマンド入力キューおよびサーバー
DELETE QALIAS		X	X	X
DELETE QLOCAL		X	X	X
DELETE QMODEL		X	X	X
DELETE QREMOTE		X	X	X
DELETE STGCLASS		X	X	X
DELETE SUB		X	X	X
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 CHSTATUS			X	X
DISPLAY CLUSQMGR			X	X
DISPLAY CMDSERV	X	X	X	X
DISPLAY CONN		X	X	X
DISPLAY CHINIT		X	X	X
DISPLAY GROUP		X	X	X
DISPLAY LOG	X	X	X	X
DISPLAY NAMELIST		X	X	X
DISPLAY PROCESS		X	X	X
DISPLAY QALIAS		X	X	X
DISPLAY QCLUSTER		X	X	X
DISPLAY QLOCAL		X	X	X
DISPLAY QMGR		X	X	X
DISPLAY QMODEL		X	X	X
DISPLAY QREMOTE		X	X	X
DISPLAY QSTATUS		X	X	X
DISPLAY QUEUE		X	X	X
DISPLAY SECURITY			X	X
DISPLAY STGCLASS		X	X	X
DISPLAY SUB		X	X	X
DISPLAY TOPIC		X	X	X

表 *26. MQSC* コマンドの実行元にできるソース *(*続き*)*

コマンド	**CSQINP1**	**CSQINP2**	**z/OS** コンソール	コマンド入力キューおよびサーバー
DISPLAY SYSTEM	X	X	X	X
DISPLAY THREAD		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	X	X
RECOVER CFSTRUCT			X	X
REFRESH CLUSTER		X	X	X
REFRESH QMGR		X	X	X
REFRESH SECURITY		X	X	X
RESET CHANNEL			X	X
RESET CLUSTER		X	X	X
RESET QSTATS		X	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 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 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

表 26. *MQSC* コマンドの実行元にできるソース (続き)				
コマンド	**CSQINP1**	**CSQINP2**	**z/OS コンソール**	**コマンド入力キューおよびサーバー**
STOP TRACE	X	X	X	X
SUSPEND QMGR			X	X

MQSC コマンドにある各コマンドの説明では、コマンドの実行元にできるソースが示されています。

z/OS IBM MQ for z/OS の初期化コマンド

初期化コマンドを使用して、キュー・マネージャーの始動を制御できます。

初期化入力データ・セットに含まれているコマンドは、キュー・マネージャーの始動時に IBM MQ が初期化される時点で処理されます。 初期化入力データ・セットから実行できるコマンドには、以下の 3 つのタイプがあります。

- 他の場所で定義できない IBM MQ エンティティーを定義するコマンド (DEFINE BUFFPOOL など)。

  これらのコマンドは、DD 名 CSQINP1 で参照するデータ・セットの中に組み込む必要があります。 これらのコマンドは、初期化の再始動フェーズの前に処理されます。 コンソール、操作パネル、制御パネル、アプリケーション・プログラムからこれらのコマンドを実行することはできません。 これらのコマンドに対する応答は、開始タスク・プロシージャーの CSQOUT1 ステートメントで参照する順次データ・セットに書き込まれます。
- 再始動後にリカバリーできる IBM MQ オブジェクトを定義するコマンド。 これらの定義は、DD 名 CSQINP2 で参照するデータ・セットの中で指定する必要があります。 これらの定義は、ページ・セット 0 に格納されます。 CSQINP2 は、初期化の再始動フェーズの後に処理されます。 これらのコマンドに対する応答は、開始タスク・プロシージャーの CSQOUT2 ステートメントで参照する順次データ・セットに書き込まれます。
- IBM MQ オブジェクトを操作するコマンド。 これらのコマンドも、DD 名 CSQINP2 で参照するデータ・セットの中で指定する必要があります。 例えば、IBM MQ に用意されているサンプルには、サブシステムの送達不能キューを指定するための ALTER QMGR コマンドが含まれています。 これらのコマンドに対する応答は、CSQOUT2 出力データ・セットに書き込まれます。

注 : IBM MQ オブジェクトが CSQINP2 で定義されていると、IBM MQ は、キュー・マネージャーの始動のたびにそれらのオブジェクトを再定義しようとします。 オブジェクトが既に存在すると、それらのオブジェクトを定義しようとする処理は失敗します。 オブジェクトを CSQINP2 で定義する必要がある場合は、DEFINE コマンドの REPLACE パラメーターを使用することによってこの問題を回避できますが、そのようにすると、キュー・マネージャーの前回の実行で追加された変更内容がオーバーライドされます。

IBM MQ for z/OS には、サンプル初期化データ・セット・メンバーが用意されています。 これらについては、 IBM MQ で提供されるサンプル定義に説明

分散キューイングのための初期化コマンド

START CHINIT コマンドで CSQINP2 初期化データ・セットを使用することもできます。 分散キューイング環境を定義するために他の一連のコマンドが必要な場合 (例えば、リスナーを始動する必要がある場合) は、IBM MQ に用意されている CSQINPX という第 3 の初期化入力データ・セットを使用できます。 このデータ・セットは、チャネル・イニシエーターの開始タスク・プロシージャーの一部として処理されます。

データ・セットに含まれている MQSC コマンドは、チャネル・イニシエーターの初期化の最後の時点で実行され、出力は、CSQOUTX DD ステートメントで指定されているデータ・セットに書き込まれます。 CSQINPX 初期化データ・セットを使用すれば、リスナーの始動などの操作を実行できます。

IBM MQ for z/OS には、サンプル・チャネル・イニシエーター初期化データ・セット・メンバーが用意されています。 IBM MQ で提供されるサンプル定義で説明されています。

パブリッシュ/サブスクライブのための初期化コマンド

パブリッシュ/サブスクライブ環境を定義するために一連のコマンドが必要な場合 (例えば、サブスクリプションを定義する場合)、IBM MQ に用意されている CSQINPT と呼ばれる第 4 の初期化入力データ・セットを使用できます。

このデータ・セットに含まれている MQSC コマンドは、パブリッシュ/サブスクライブの初期化の最後に実行され、CSQOUTT DD ステートメントによって指定されたデータ・セットに出力が書き込まれます。 例えば、サブスクリプションを定義する場合は、CSQINPT 初期化データ・セットを使用することもできます。

パブリッシュ/サブスクライブの初期化データ・セット・メンバーのサンプルは、IBM MQ for z/OS で提供されています。 IBM MQ で提供されるサンプル定義で説明されています。

z/OS IBM MQ for z/OS ユーティリティー

IBM MQ for z/OS には、システム管理のために使用できる一連のユーティリティー・プログラムが用意されています。

IBM MQ for z/OS には、さまざまな管理用タスクを実行するための一連のユーティリティー・プログラムが用意されています。例えば、以下のようなタスクを実行できます。

- メッセージ・セキュリティー・ポリシーを管理します。
- バックアップ、リストア、再編成のタスクを実行します。
- コマンドを実行し、オブジェクト定義を処理します。
- データ変換出口を生成します。
- ブートストラップ・データ・セットを変更します。
- ログに関する情報を表示します。
- ログを印刷します。
- Db2 の表や他の Db2 ユーティリティーをセットアップします。
- 送達不能キューにあるメッセージを処理します。

メッセージ・セキュリティー・ポリシー・ユーティリティー

メッセージ・セキュリティー・ポリシー・ユーティリティー (CSQ0UTIL) は、メッセージ・セキュリティー・ポリシーを管理するためのスタンドアロン・ユーティリティーとして実行されます。 詳しくは、メッセージ・セキュリティー・ポリシー・ユーティリティー (CSQ0UTIL) を参照してください。

CSQUTIL ユーティリティー

バックアップ、リストア、再編成のタスクを実行するために用意されているユーティリティー・プログラムです。 詳細については、『CSQUTIL ユーティリティー』を参照してください。

データ変換出口ユーティリティー

IBM MQ for z/OS のデータ変換出口ユーティリティー (**CSQUCVX**) は、データ変換出口ルーチンを作成するために実行するスタンドアロン・ユーティリティーです。

ログ目録変更ユーティリティー

IBM MQ for z/OS のログ目録変更ユーティリティー・プログラム (**CSQJU003**) は、ブートストラップ・データ・セット (BSDS) を変更するために実行するスタンドアロン・ユーティリティーです。 このユーティリティーを使用して、以下の機能を実行できます。

- アクティブ・ログ・データ・セットまたはアーカイブ・ログ・データ・セットを追加/削除します。
- アーカイブ・ログのパスワードを提供します。

ログ・マップ印刷ユーティリティー

IBM MQ for z/OS のログ・マップ印刷ユーティリティー・プログラム (**CSQJU004**) は、以下の情報を表示するために実行するスタンドアロン・ユーティリティーです。

- すべてのアクティブ・ログ・データ・セットとアーカイブ・ログ・データ・セットの両方のコピーのログ・データ・セット名とログ RBA の関連付け。 重複ロギングがアクティブになっていない場合は、データ・セットのコピーが 1 つだけ存在します。
- 新しいログ・データを書き込めるアクティブ・ログ・データ・セット。
- ブートストラップ・データ・セット (BSDS) 内のチェックポイント・レコードのキューの内容。
- アーカイブ・ログ・コマンド・ヒストリー・レコードの内容。
- システムとユーティリティーのタイム・スタンプ。

ログ印刷ユーティリティー

ログ印刷ユーティリティー・プログラム (**CSQ1LOGP**) は、スタンドアロン・ユーティリティーとして実行します。 このユーティリティーを実行するときに、以下の対象を指定できます。

- ブートストラップ・データ・セット (BSDS)
- アクティブ・ログ (BSDS なし)
- アーカイブ・ログ (BSDS なし)

キュー共有グループ・ユーティリティー

キュー共有グループ・ユーティリティー・プログラム (**CSQ5PQSG**) は、Db2 の表をセットアップし、キュー共有グループで必要な他の Db2 タスクを実行するためのスタンドアロン・ユーティリティーです。

アクティブ・ログ事前フォーマット・ユーティリティー

アクティブ・ログ事前フォーマット・ユーティリティー (**CSQJUFMT**) は、キュー・マネージャーでアクティブ・ログ・データ・セットを使用する前に、アクティブ・ログ・データ・セットをフォーマットするためのユーティリティーです。 このユーティリティーを使用してアクティブ・ログ・データ・セットを事前フォーマットすると、 アクティブ・ログを介したキュー・マネージャーの最初のパスで、 ログ書き込みのパフォーマンスが向上します。

送達不能キュー・ハンドラー・ユーティリティー

送達不能キュー・ハンドラー・ユーティリティー・プログラム (**CSQUDLQH**) は、スタンドアロン・ユーティリティーとして実行します。 送達不能キューにあるメッセージを検査し、このユーティリティーで指定する一連の規則に基づいてそれらのメッセージを処理します。

qload ユーティリティー

IBM MQ 8.0 以降、 IBM MQ Supportpac MO03 に同梱されている **qload** ユーティリティーが、 **dmpmqmsg** ユーティリティーとして IBM MQ に組み込まれました。

z/OS では、このユーティリティーは、SCSQLOAD ライブラリーの実行可能モジュール CSQUDMSG として使用可能であり、互換性のために別名は QLOAD です。 サンプル JCL も、SCSQPROC のメンバー CSQ4QLOD として用意されています。

z/OS IBM MQ for z/OS の CSQUTIL ユーティリティー

CSQUTIL ユーティリティー・プログラムは IBM MQ for z/OS と共に提供され、バックアップ、復元、および再編成のタスクを実行するのを支援したり、コマンドを発行してオブジェクト定義を処理したりします。

CSQUTIL ユーティリティー・プログラムについて詳しくは、IBM MQ ユーティリティー・プログラム (CSQUTIL) を参照してください。 このユーティリティー・プログラムを使用することにより、以下の関数を呼び出すことができます。

コマンド
MQSC コマンドを発行し、オブジェクト定義を記録し、クライアント・チャネル定義ファイルを作成します。

COPY
名前指定された IBM MQ for z/OS メッセージ・キューの内容か、名前指定されたページ・セットのすべてのキューの内容を読み取って、順次ファイルに書き込み、元のキューを保存します。

COPYPAGE
ページ・セット全体を、さらに大きなページ・セットにコピーします。

EMPTY
名前指定された IBM MQ for z/OS メッセージ・キューの内容か、名前指定されたページ・セットのすべてのキューの内容を削除し、キューの定義を保存します。

形式 (Format)
IBM MQ for z/OS ページ・セットを形式設定します。

LOAD
COPY 関数によって作成された順次ファイルから、名前指定された IBM MQ for z/OS メッセージ・キューの内容か、名前指定されたページ・セットのすべてのキューの内容を復元します。

PAGEINFO
1 つ以上のページ・セットから、ページ・セット情報を抽出します。

RESETPAGE
ページ・セット全体を他のページ・セットのデータ・セットにコピーし、コピー内のログ情報をリセットします。

SCOPY
キュー・マネージャーがオフラインの間に、キューの内容をデータ・セットにコピーします。

SDEFS
キュー・マネージャーがオフラインの間に、オブジェクト用に 1 組の定義コマンドを生成します。

SLOAD
前に行った COPY または SCOPY 操作の宛先データ・セットからメッセージを復元します。 SLOAD は、単一キューを処理します。

SWITCH
クラスター送信側チャネルに関連付けられた伝送キューの切り替えまたは照会を行います。

XPARM
チャネル開始プログラム・パラメーター・ロード・モジュールを、キュー・マネージャー属性に変換します (マイグレーション用)。

z/OS IBM MQ for z/OS の運用

IBM MQ for z/OS は、下記の基本手順によって操作します。

このセクションで説明する操作は、IBM MQ Explorer を使用して実行することもできます。このツールは IBM MQ for Windows および IBM MQ for Linux (x86 および x86-64 プラットフォーム) で配布されます。スタンドアロン IBM MQ Explorer は、 Fix Central からダウンロードすることができます。 詳細については、112 ページの『IBM MQ Explorer を使用した管理』を参照してください。

z/OS でのキュー・マネージャー・コマンドの発行

IBM MQ 制御コマンドは、z/OS コンソールから、またはユーティリティー・プログラム CSQUTIL を使用して実行できます。 コマンドは、コマンド接頭部ストリング (CPF) を使用して、どの IBM MQ サブシステムがコマンドを処理するのかを示します。

ほとんどの IBM MQ の操作環境は、IBM MQ コマンドを使用して制御できます。 IBM MQ for z/OS は、これらのコマンドの MQSC と PCF タイプ両方をサポートします。 このトピックでは、MQSC コマンドを使用して属性を指定する方法について説明しています。したがって、これらのコマンドおよび属性は PCF 名ではなく MQSC コマンド名を使用して表します。 MQSC コマンドの構文についての詳細は、MQSC コマンドを参照してください。 PCF コマンドの構文の詳細については、25 ページの『IBM MQ プログラマブル・コマンド・フォーマットの使用』を参照してください。 適切な許可を与えられたユーザーであれば、次のものから IBM MQ コマンドを実行できます。

- 初期化入力データ・セット (423 ページの『IBM MQ for z/OS の初期化コマンド』で説明されています)。
- z/OS コンソール、またはそれに相当するもの (SDSF など)
- z/OS マスター読み取りコマンド・ルーチン、MGCRE (SVC 34)
- IBM MQ ユーティリティー、CSQUTIL (IBM MQ ユーティリティー・プログラムで説明されています)。
- ユーザー・アプリケーション。次のプログラムが可能です。
  - CICS プログラム
  - TSO プログラム
  - z/OS バッチ・プログラム
  - IMS プログラム

  これについては、447 ページの『IBM MQ for z/OS 管理のためのプログラムの作成』を参照してください。

これらのコマンドの機能のほとんどには、TSO および ISPF からアクセス可能な便利な操作および制御パネルが 提供されています (432 ページの『IBM MQ for z/OS の操作および制御パネル』に説明されています)。

詳細については、以下を参照してください。

- 427 ページの『z/OS コンソールなどからのコマンドの実行』
  - コマンド接頭部ストリング
  - コマンド実行のための z/OS コンソールの使用
  - コマンド応答
- ユーティリティー・プログラム CSQUTIL からのコマンドの実行

z/OS コンソールなどからのコマンドの実行

すべての IBM MQ コマンドは、 z/OS コンソールまたはそれに相当するコンソールから出すことができます。 z/OS コマンドを発行できる任意の場所 (SDSF など) から、または MGCRE マクロを使用するプログラムから、 IBM MQ コマンドを発行することもできます。

コンソールで入力したコマンドの結果として表示できるデータの最大量は、32KB です。

注 :

1. IMS 端末から IMS/SSR コマンド形式を使用して IBM MQ コマンドを発行することはできません。 この機能は、IMS アダプターによってサポートされていません。
2. SDSF が提供する入力フィールドは、一部のコマンド、特にチャネル用のコマンドには短すぎる場合があります。

コマンド接頭部ストリング

各 IBM MQ コマンドには、428 ページの図 32 に示すように、接頭部としてコマンド接頭部ストリング (CPF) を付ける必要があります。

複数の IBM MQ サブシステムは z/OS 環境で実行できるので、CPF を使用して、どの IBM MQ サブシステムがコマンドを処理するのかを示します。 例えば、CPF が「+CSQ1」である CSQ1 というサブシステム用のキュー・マネージャーを開始するには、オペレーター・コンソールからコマンド +CSQ1 START QMGR を実行します。 この CPF は、(サブシステム CSQ1 の) サブシステム名表に定義されていなければなりません。 これについては、コマンド接頭部ストリング (CPF) の定義で説明されています。 例の中では、ストリング「+CSQ1」がコマンド接頭部として使用されています。

コマンド実行のための z/OS コンソールの使用

z/OS コンソールから単純コマンド (例えば、428 ページの図 32 の DISPLAY コマンド) を入力することができます。 しかし、複雑なコマンドを実行したり、コマンドをセットで頻繁に実行したりする場合は、別の方法を使う方が有効です。

```
+CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLOCAL)
```

図 *32. z/OS* コンソールからの *DISPLAY* コマンドの実行

コマンド応答

コマンドへの直接応答は、コマンドを実行したコンソールへ送られます。 IBM MQ は、 z/OS で使用可能な 拡張コンソール・サポート (EMCS) 機能をサポートしているため、4 バイトの ID を持つコンソールを使用できます。 さらに、START QMGR と STOP QMGR を除くすべてのコマンドは、MGCRE マクロを使用した プログラムから実行された場合、「コマンドおよび応答のトークン (CART)」の使用をサポートします。

ユーティリティー・プログラム CSQUTIL からのコマンドの実行

ユーティリティー・プログラム CSQUTIL の COMMAND 機能を使用することにより、順次データ・セットからコマンドを実行できます。 このユーティリティーは、コマンドをメッセージとしてシステム・コマンド入力キュー に転送し、 その応答を待ちます。 この応答は、SYSPRINT に元のコマンドと共に出力されます。 詳しくは、 IBM MQ ユーティリティー・プログラムを参照してください。

z/OS *z/OS* 上でのキュー・マネージャーの開始と停止

このトピックでは、キュー・マネージャーの停止および開始の概要を示します。

このセクションでは、キュー・マネージャーの開始と停止について説明します。 このセクションには、以下のトピックに関する情報が含まれます。

- 429 ページの『IBM MQ を開始する前に』
- 429 ページの『キュー・マネージャーの開始』
- 431 ページの『キュー・マネージャーの停止』

キュー・マネージャーの開始および停止は、比較的簡単です。 キュー・マネージャーが正常な状態で停止したとき、 その最後のアクションは終了チェックポイントを取ることです。 このチェックポイントおよびログは、キュー・マネージャーが再始動に必要とする情報を与えます。

このセクションでは、START コマンドおよび STOP コマンドについて説明し、異常終了が起きた場合の始動について簡単に説明します。

IBM MQ を開始する前に

IBM MQ がインストールされた後、それは正式の z/OS サブシステムとして定義されます。 次のメッセージが、z/OS の初期プログラム・ロード (IPL) の間に表示されます。

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm INITIALIZATION COMPLETE
```

ここで、*ssnm* は IBM MQ サブシステム名です。

これ以降、 システム制御コマンドの発行を許可されている任意の *z/OS* コンソールから、 そのサブシステムのキュー・マネージャーを開始できます。 つまり、 z/OS SYS コマンド・グループです。 START コマンドは、許可されたコンソールから出す必要があります。また、 JES または TSO を使用して実行依頼することはできません。

キュー共用グループを使用している場合は、まず RRS を開始し、次いで Db2 を開始してから、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

キュー・マネージャーを開始するには、START QMGR コマンドを実行します。 しかし、適切な許可を持っていなければ、START コマンドを正常に実行すること はできません。 IBM MQ セキュリティーについては、 z/OS 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。 429 ページの図 33 には、START コマンドの例が示されています。 (IBM MQ コマンドは、必ずコマンド接頭部ストリング (CPF) で始める必要があります。)

```
+CSQ1  START QMGR
+CSQ1  START QMGR PARM(NEWLOG)
```

図 *33. z/OS* コンソールからのキュー・マネージャーの開始

START QMGR コマンドの構文については、START QMGR を参照してください。

キュー・マネージャーは、バッチ・ジョブとして実行することも、z/OS コマンド START を使用して開始することもできません。 この方法を使用すると、後で異常終了するような、IBM MQ のアドレス・スペースを開始する可能性があります。 また、CSQUTIL ユーティリティー・プログラムまたはそれに類似したユーザー・アプリケーションからキュー・マネージャーを開始することもできません。

しかし、START QMGR コマンドを z/OS MGCRE (SVC 34) サービスへ渡すことによって、APF 許可プログラムからキュー・マネージャーを開始することができます。

キュー共有グループを使用する場合は、キュー・マネージャーの開始時に、関連する Db2 システムと RRS が活動状態になっている必要があります。

開始オプション

キュー・マネージャーを開始すると、システム・パラメーター・モジュールがロードされます。 システム・パラメーター・モジュールの名前は、 以下の 2 つのいずれかの方法で指定できます。

- /cpf START QMGR コマンドの PARM パラメーターを使用する。例えば、次のようにします。

  ```
  /cpf START QMGR PARM(CSQ1ZPRM)
  ```

- 開始プロシージャーのパラメーターを使用する。例えば、JCL EXEC ステートメントを次のようにコーディングします。

```
//MQM  EXEC PGM=CSQYASCP,PARM='ZPARM(CSQ1ZPRM)'
```

システム・パラメーター・モジュールは、キュー・マネージャーがカスタマイズされたときに指定された情報を提供します。

IBM MQ 9.1.0 から、**QMGRPROD** オプションを使用して、キュー・マネージャーの使用が記録される製品を指定できます。 また、**AMSPROD** オプションを使用して、AMS が使用されている場合にそれと同等になるものを指定することができます。 許可される値について詳しくは、MQSC START QMGR コマンドを参照してください。

JCL EXEC ステートメントの例を以下に示します。

```
//MQM EXEC  PGM=CSQYASCP,PARM='QMGRPROD(MQ)'
```

製品使用記録の詳細については、z/OS MVS プロダクト管理 を参照してください。

また、ENVPARM オプションを使用して、キュー・マネージャーのために JCL プロシージャー内の 1 つ以上のパラメーターを置き換えることができます。

例えば、DD 名 CSQINP2 が可変値になるよう、キュー・マネージャー開始プロシージャーを更新することができます。 これにより、開始プロシージャーを変更しなくても、DD 名 CSQINP2 を変更できます。 これは、変更を実施する場合やオペレーターのためにバックアウトを提供する場合、およびキュー・マネージャーを操作する場合に役立ちます。

キュー・マネージャー CSQ1 の開始プロシージャーが 430 ページの図 34 のようであったとします。

```
//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=*
```

図 *34.* 開始プロシージャーの例

次にキュー・マネージャーを、次のコマンドで開始したとします。

```
+CSQ1  START QMGR
```

この場合、使用される CSQINP2 は、CSQ1NORM と呼ばれるメンバーとなります。

ただし、新しい一連のプログラムを導入して、次回にキュー・マネージャー CSQ1 を開始するときには、 CSQINP2 定義をメンバー CSQ1NEW から取り込みます。 この場合には、 次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。

```
+CSQ1  START QMGR ENVPARM('INP2=NEW')
```

このようにすると、CSQ1NEW が CSQ1NORM の代わりに使用されることになります。 注: z/OS では、シンボリック・パラメーターの KEYWORD=value 指定は (INP2=NEW の場合と同じように) 255 文字までに制限されます。

異常終了後の開始

IBM MQ では、再始動が正常終了後のものか異常終了後のものかを自動的に検知します。

キュー・マネージャーを異常終了後に開始することは、STOP QMGR コマンドが 発行された後にキュー・マネージャーを開始することとは異なります。 STOP QMGR コマンドが実行されたら、システムは決められた順序に従って作業を完了し、 停止する前に終了チェックポイントを取ります。 キュー・マネージャーは再始動時に、システム・チェックポイントおよび回復ログからの情報を使用して、 終了時のシステム状態を決定します。

ただし、キュー・マネージャーは異常終了すると、作業を完了したり 終了チェックポイントを取ったりできずに終了します。 キュー・マネージャーを異常終了の後で再始動すると、ログ内の情報を使用してその終了時の状況をリフレッシュし、 種々のタスクの状況を知らせてきます。 通常、再始動処理では不一致の状態がすべて解決されます。 しかし場合によっては、 不一致の状態を解決するために特別なステップを実行しなければなりません。

開始時のユーザー・メッセージ

キュー・マネージャーを正常に始動すると、キュー・マネージャーによって始動メッセージのセットが生成されます。

キュー・マネージャーの停止

キュー・マネージャーを停止する前に、すべての IBM MQ 関連の要応答オペレーターへの書き込み (WTOR) メッセージは、応答 (例えば、ログ要求の書き込み) を受け取る必要があります。 431 ページの図 35 にあるそれぞれのコマンドは、実行中のキュー・マネージャーを終了します。

```
+CSQ1  STOP QMGR
+CSQ1  STOP QMGR MODE(QUIESCE)
+CSQ1  STOP QMGR MODE(FORCE)

+CSQ1  STOP QMGR MODE(RESTART)
```

図 *35.* キュー・マネージャーの停止

コマンド STOP QMGR は STOP QMGR MODE(QUIESCE) にデフォルト設定されます。

QUIESCvE モードでは、 IBM MQ は新しい接続スレッドを作成することを許可しませんが、既存のスレッドを続行できるようにします。これは、すべてのスレッドが終了したときにのみ終了します。 アプリケーションは、 キュー・マネージャーの休止イベントを通知するよう要求できます。 したがって、 通知を要求したアプリケーションに接続を切断する機会を与えるために、できるだけ QUIESCE モードを使用してください。 詳細は 終了中に起こること を参照してください

キュー・マネージャーが STOP QMGR MODE(QUIESCE) コマンドに応じて適時終了しない場合は、DISPLAY CONN コマンドを使用して接続スレッドが存在するかどうかを判別し、 関連アプリケーションを終了するために必要なステップを行います。 スレッドが存在しない場合、 STOP QMGR MODE(FORCE) コマンドを実行してください。

STOP QMGR MODE(QUIESCE) および STOP QMGR MODE(FORCE) コマンドは MVS 自動再始動マネージャー (ARM) から IBM MQ を登録解除し、ARM がキュー・マネージャーを自動的に再始動しないようにします。 STOP QMGR MODE(RESTART) コマンドは、ARM から IBM MQ を登録解除しないという点を除き、STOP QMGR MODE(FORCE) コマンドと同じ方法で動作します。 つまり、キュー・マネージャーは即時自動再始動に適しています。

IBM MQ サブシステムが ARM に登録されていない場合は、STOP QMGR MODE(RESTART) コマンドは拒否され、次のメッセージが z/OS コンソールに送信されます。

```
CSQY205I ARM element arm-element is not registered
```

このメッセージが出されない場合は、キュー・マネージャーは自動的に再始動されます。 ARM の詳細については、508 ページの『z/OS 自動再始動管理プログラム (ARM) の使用』を参照してください。

STOP QMGR MODE(FORCE) がキュー・マネージャーを終了しない場合にのみ、 キュー・マネージャー・アドレス・スペースを取り消してください。

キュー・マネージャーが、アドレス・スペースを取り消すことによって停止するか、コマンド STOP QMGR MODE(FORCE) を使用することによって停止すると、接続された CICS または IMS システムとの整合性が保たれます。 資源の再同期はキュー・マネージャーの再始動時に開始され、CICS または IMS システムとの接続が確立されたときに完了します。

注 : キュー・マネージャーを停止したとき、メッセージ IEF352I が出される場合があります。 z/OS は、アドレス・スペースを使用不可としてマークすることができないために整合性に問題が生じることを検出すると、このメッセージを出します。 このメッセージは無視して構いません。

メッセージの停止

STOP QMGR コマンドを実行したあと、 例えば次のように CSQY009I メッセージと CSQY002I メッセージが表示されます。

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM
USER(userid), STOP MODE(FORCE)
CSQY002I +CSQ1 QUEUE MANAGER STOPPING
```

ここで、userid は、STOP QMGR コマンドを実行したユーザー ID です。 MODE パラメーターはコマンド内で指定された値によって異なります。

STOP コマンドが正常に完了すると、次のメッセージが z/OS コンソールに表示されます。

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

ARM を使用している場合、MODE(RESTART) を指定しない限り、次のメッセージも表示されます。

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type
arm-element-type successful
```

次のメッセージが表示されるまで、キュー・マネージャーを再始動することはできません。

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```

z/OS IBM MQ for z/OS の操作および制御パネル

IBM MQ の操作および制御パネルを使用して、IBM MQ オブジェクトに対して管理タスクを行うことができます。 このトピックでは、コマンドおよび制御パネルについての概要を説明します。

これらのパネルを使用して、IBM MQ オブジェクトを定義、表示、変更、または削除します。 日常の管理およびオブジェクトへの小さな変更を行うには、 このパネルを使用します。 多数のオブジェクトをセットアップおよび変更する場合は、CSQUTIL ユーティリティー・プログラムの COMMAND 機能を使用します。

操作および制御パネルは、チャネル・イニシエーター用の制御をクラスター化およびセキュリティーのためにサポートします (例えば、 チャネルまたは TCP/IP リスナーを開始するため)。 また、これらのパネルによってスレッド、 およびページ・セットの使用状況に関する情報を表示できます。

これらのパネルは、MQSC タイプの IBM MQ コマンドを、システム・コマンド入力キューを介してキュー・マネージャーに送信することによって機能します。

注 :

1. IBM MQ for z/OS の操作および制御パネル (CSQOREXX) は、バージョン 7 以降で追加された新しい機能とパラメーターをすべてサポートしているとは限りません。 例えば、トピック・オブジェクトまたはサブスクリプションを直接操作するためのパネルはありません。

   他のパネルからは直接操作できないパブリッシュ/サブスクライブ定義、およびその他のシステム制御については、サポートされている以下のいずれかの手段を利用して管理できます。

   a. IBM MQ エクスプローラー

   b. z/OS コンソール

   c. プログラマブル・コマンド・フォーマット (PCF) メッセージ

   d. CSQUTIL の COMMAND 機能

   e. IBM MQ Web コンソール

   CSQOREXX パネルで汎用 **Command** アクションを使用すれば、SMDS 関連のコマンドを含め、あらゆる有効な MQSC コマンドを発行できます。 CSQUTIL の COMMAND 機能が発行する、すべてのコマンドを使用できます。
2. パネル内のコマンド行から直接 IBM MQ コマンドを実行することはできません。
3. 操作および制御パネルを使用するには、正しいセキュリティー許可が必要です。これについては、コマンド・セキュリティーとコマンド・リソース・セキュリティーのためのユーザー ID で説明しています。
4. CSQUTIL、または CSQOREXX パネルを使用してユーザー ID およびパスワードを指定することはできません。 その代わりに、ユーザー ID に、MQCONN 内の BATCH プロファイルに対する UPDATE 権限が付与されている場合は、**CHCKLOCL**(*REQUIRED* 設定を迂回できます。 詳しくは、 ローカルでバインドされたアプリケーションでの **CHCKLOCL** の使用 を参照してください。

z/OS 操作および制御パネルの起動および規則

IBM MQ を制御して、ISPF パネルを介してコマンドを発行することができます。

IBM MQ の操作パネルと制御パネルへのアクセス方法

ISPF/PDF 基本オプション・メニューが IBM MQ 用に更新されている場合は、そのメニューから IBM MQ 操作および制御パネルにアクセスできます。 メニューの更新については、タスク 20: 操作パネルおよび制御パネルをセットアップするを参照してください。

IBM MQ 操作および制御パネルへは、TSO コマンド処理プログラム・パネルからアクセスできます (通常は、ISPF/PDF 基本オプション・メニュー上のオプション 6)。 これを行うために実行する EXEC の名前は、CSQOREXX です。 これには 2 つのパラメーターがあります。 `thlqual` は使用する IBM MQ ライブラリーの高位修飾子、 `langletter` は使用する各国語ライブラリーを識別する文字です (例えば、 U.S の場合は E)。 英語) がアクティブになります。 使用している ISPF セットアップに IBM MQ ライブラリーが永続的にインストールされている場合は、これらのパラメーターを省略できます。 あるいは、TSO コマンド行から CSQOREXX を実行することもできます。

これらのパネルは、オペレーターおよび管理者が、 最低限の正式な訓練で使用できるよう設計されています。 パネルを実行しながらこれらの指示を読み、 示されているさまざまなタスクを試してください。

注 : パネルの使用中に、 SYSTEM.CSQOREXX.* 作成されます。

操作および制御パネルの規則

IBM MQ の文字ストリングおよび名前の一般規則については、 IBM MQ オブジェクトの命名規則 を参照してください。 しかし、操作および制御パネルだけに適用される規則は次のとおりです。

- ストリング (例えば、記述) を一重引用符や二重引用符で囲まないでください。
- アポストロフィまたは引用符をテキスト・フィールドに含める場合、それを繰り返したり、エスケープ文字を追加したりする必要はありません。 文字は、入力したとおりに保存されます。例えば、以下のようになります。

```
This is Maria's queue
```

  パネル処理プログラムが、その引用符を 2 つにして IBM MQ に渡します。 ただし、これを行うためにデータを切り捨てる必要がある場合には、 パネル処理プログラムが切り捨てを実行します。
- ほとんどのフィールドに大文字または小文字を使用することが可能で、Enter キーを押すと それらの文字は大文字に変換されます。 ただし、次の例外があります。
  - 大文字の A ～ Z で始めて、 その後に大文字の A ～ Z または数字を続けなければならないストレージ・クラス名およびカップリング・ファシリティー構造名。
  - 変換されない特定のフィールド。 これには以下が含まれます。
    - アプリケーション ID
    - 説明
    - 環境データ
    - オブジェクト名 (ただし、小文字のオブジェクト名を使用した場合は、それを z/OS コンソールから入力できない場合があります)
    - リモート・システム名
    - トリガー・データ
    - ユーザー・データ
- 名前の中では、先行ブランクと先行下線は無視されます。 このため、ブランクまたは下線で始まるオブジェクト名を付けることはできません。
- 下線は、ブランク・フィールドの範囲を示すために使用されます。 Enter キーを押すと、後に続く下線はブランクで置換されます。
- 多くの記述フィールドおよびテキスト・フィールドが、複数の部分にあり、それぞれの部分が IBM MQ によって別個に処理されます。 このため、後書きブランクは保持され、テキストは連続しません。

ブランク・フィールド

IBM MQ オブジェクトに対して 「**定義**」 アクションを指定すると、定義パネルの各フィールドに値が入ります。 IBM MQ が値を読み取る場所についての情報は、表示パネルに関する一般ヘルプ (拡張ヘルプ) を参照してください。 あるフィールドにブランクを入力し、そのフィールドでブランクの使用が許可されていない場合、IBM MQ はそのフィールドにインストールのデフォルト値を書き込むか、または必須の値を入力するようプロンプトを出します。

IBM MQ オブジェクトに対して **Alter** アクションを指定すると、変更パネルの各フィールドには、そのフィールドの現行値が含まれます。 あるフィールドにブランクを入力し、そのフィールドでブランクの使用が許可されていなければ、そのフィールドの値は変更されません。

z/OS z/OS でのオブジェクトおよびアクション

操作および制御パネルは、多数の異なるタイプのオブジェクトおよびそのオブジェクト上で実行できる多数のアクションを提供します。

アクションは初期パネルに一覧表示され、オブジェクトを操作したりオブジェクトに関する情報を表示することができます。 これらのオブジェクトには、一部のエクストラ・オブジェクトと共にすべての IBM MQ オブジェクトが組み込まれます。 オブジェクトは、以下のカテゴリーに分類されます。

- キュー、プロセス、認証情報オブジェクト、名前リスト、 ストレージ・クラス、CF 構造
- チャネル
- クラスター・オブジェクト
- キュー・マネージャーおよびセキュリティー
- 接続
- システム

IBM MQ オブジェクトで実行できるアクションの相互参照表については、アクション を参照してください。

キュー、プロセス、認証情報オブジェクト、名前リスト、 ストレージ・クラス、CF 構造

これらは基本の IBM MQ オブジェクトです。 オブジェクトは各タイプごとに多数存在する可能性があります。 これらは、リスト、フィルターを使用したリスト、定義、および削除が可能であり、 LIST または DISPLAY、FILTER を指定した LIST、DEFINE LIKE、MANAGE、および ALTER のアクションを使用して、表示および変更できる属性を持っています。 (オブジェクトは MANAGE アクションを使用して削除されます。)

このカテゴリーは次のオブジェクトで構成されています。

QLOCAL	ローカル・キュー
QREMOTE	リモート・キュー
QALIAS	キューを間接的に参照するための別名キュー
QMODEL	キューを動的に定義するためのモデル・キュー
QUEUE	任意のタイプのキュー
QSTATUS	ローカル・キューの状態
PROCESS	トリガー・イベントが起こるときに開始されるアプリケーションの情報
AUTHINFO	認証情報: LDAP サーバーを使用して証明書取り消しリスト (CRL) 検査を実行するために必要な定義
NAMELIST	キューまたはクラスターなどの名前のリスト
STGCLASS	ストレージ・クラス
CFSTRUCT	カップリング・ファシリティー (CF) 構造
CFSTATUS	CF 構造の状態

チャネル

チャネルは分散キューイングに使用されます。 タイプごとにチャネルが多数存在する 可能性があり、リスト、フィルターを使用したリスト、定義、削除、表示、 および変更を行うことができます。 START、STOP および PERFORM アクションを使用して利用可能な他の機能もあります。 PERFORM はリセット、PING、および解決の各チャネル機能を提供します。

このカテゴリーは次のオブジェクトで構成されています。

CHANNEL	任意のタイプのチャネル
SENDER	送信側チャネル
SERVER	サーバー・チャネル
RECEIVER	受信側チャネル
REQUESTER	要求側チャネル

CLUSRCVR	クラスター受信側チャネル
CLUSSDR	クラスター送信側チャネル
SVRCONN	サーバー接続チャネル
CLNTCONN	クライアント接続チャネル
CHSTATUS	チャネル接続の状態

クラスター・オブジェクト

クラスターに属するキューおよびチャネルのクラスター・オブジェクトは自動的に作成されます。 基本キューおよびチャネル定義を別のキュー・マネージャーに置くことができます。 各タイプごとにオブジェクトが多数存在する可能性があり、名前が重複する場合もあります。 これらはリスト、フィルターを使用したリスト、表示が可能です。 PERFORM、START、および STOP は LIST アクションを介しても使用できます。

このカテゴリーは次のオブジェクトで構成されています。

CLUSQ	クラスターに属するキュー用に作成されたクラスター・キュー
CLUSCHL	クラスターに属するチャネル用に作成されたクラスター・チャネル
CLUSQMGR	クラスター・チャネルと同じであるが、 キュー・マネージャー名によって識別されるクラスター・キュー・マネージャー

クラスター・チャネルおよびクラスター・キュー・マネージャーには、PERFORM、START、および STOP アクションがありますが、 DISPLAY アクションだけは間接的に実行されます。

キュー・マネージャーおよびセキュリティー

キュー・マネージャーおよびセキュリティー・オブジェクトには、単一のインスタンスがあります。 これらはリスト可能で、(LIST または DISPLAY、および ALTER アクションを 使用して) 表示および変更できる属性があり、PERFORM アクションで使用可能な他の機能もあります。

このカテゴリーは次のオブジェクトで構成されています。

MANAGER	キュー・マネージャー: PERFORM アクションは、中断および再開のクラスター機能を提供します。
セキュリティー	セキュリティー機能: PERFORM アクションは、リフレッシュおよび再検査機能を提供します。

接続

接続は、リスト、フィルターを使用したリスト、および表示が可能です。

このカテゴリーには、接続オブジェクトの CONNECT だけが含まれます。

システム

他の機能の集合。 このカテゴリーは次のオブジェクトで構成されています。

SYSTEM	システム機能
CONTROL	SYSTEM の同義語

使用可能な機能は次のとおりです。

LIST または DISPLAY	キュー共有グループ、分散キューイング、ページ・セット、またはデータ・セット使用状況情報を表示します。
PERFORM	クラスター化をリフレッシュまたはリセットします。
START	チャネル・イニシエーターまたはリスナーを開始します。
STOP	チャネル・イニシエーターまたはリスナーを停止します。

Actions

オブジェクトのタイプごとに実行できるアクションを次の表に示します。

表 *27. IBM MQ* オブジェクト用の有効な操作および制御パネルのアクション

オブジェクト	変更	類似定義	管理 (1)	リストまたは表示	フィルターを使用したリスト	実行	開始	停止
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
セキュリティー	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

注記 :

1. 削除および他の機能を提供します。
2. リストまたは表示アクションを使用します。

z/OS z/OS でのオブジェクト属性指定

作業に必要なオブジェクトの属性指定 を行えます。 属性指定は、オブジェクト**定義**の保存場所、およびオブジェクトの動作を指定できます。

この属性指定は、以下のオブジェクト・タイプのいずれかを処理する場合にのみ重要です。

- キュー
- チャネル
- プロセス
- 名前リスト
- ストレージ・クラス
- 認証情報オブジェクト

他のオブジェクト・タイプを処理する場合は、属性指定は無視されます。

指定できる値は、次のとおりです。

Q
QMGR。 オブジェクト定義はキュー・マネージャーのページ・セット上にあり、 キュー・マネージャーによってのみアクセス可能です。

C
COPY。 オブジェクト定義はキュー・マネージャーのページ・セット上にあり、 キュー・マネージャーによってのみアクセス可能です。 GROUP の属性指定を持つものとして定義されているオブジェクトのローカル・コピー。

P
PRIVATE。 オブジェクト定義はキュー・マネージャーのページ・セット上にあり、 キュー・マネージャーによってのみアクセス可能です。 オブジェクトは、QMGR または COPY の属性指定を持つものとして定義されています。

G
GROUP。 オブジェクト定義は共有リポジトリーにあり、キュー共有グループにあるすべてのキュー・マネージャーからアクセス可能です。

S
SHARED。 この属性指定はローカル・キューにのみ適用されます。 キュー定義は共有リポジトリーにあり、キュー共有グループにあるすべてのキュー・マネージャーからアクセス可能です。

A
ALL。 アクション・キュー・マネージャーがターゲット・キュー・マネージャーまたは * である場合は、**すべての**属性指定のオブジェクトが含まれます。それ以外の場合は、QMGR および COPY 属性指定のオブジェクトのみが含まれます。 これがデフォルトです。

z/OS z/OS での *ISPF* 制御パネルを使用したキュー・マネージャー、デフォルト、およびレベルの選択

ISPF 内で CSQOREXX exec を使用して、キュー・マネージャーを制御できます。

初期パネルを表示している間は、どのキュー・マネージャーにも接続されていません。 ただし、Enter キーを押すとただちにそのキュー・マネージャーまたは「**接続名**」フィールドで指定されたキュー共有グループのキュー・マネージャーに接続されます。 このフィールドはブランクにしておくことができます。 この場合、バッチ・アプリケーションにはデフォルト・キュー・マネージャーを使用することになります。 これは CSQBDEFV に定義されています (詳細については、タスク 19: バッチ、TSO、および RRS アダプターをセットアップするを参照)。

「**ターゲット・キュー・マネージャー**」フィールドを使用して、要求したアクションを実行するキュー・マネージャーを指定します。 このフィールドをブランクにしておくと、「**接続名**」フィールドに 指定された

キュー・マネージャーにデフォルト設定されます。 ターゲットとして、 接続先のキュー・マネージャーとは別のキュー・マネージャーを指定することもできます。 その場合は、 キュー・マネージャーの別名定義を提供するリモート・キュー・マネージャー・オブジェクトの名前を指定するのが普通です (その名前は、コマンド入力キューをオープンするときの *ObjectQMgrName* として使用されます)。 そのためには、 リモート・キュー・マネージャーにアクセスするための適切なキューとチャネルをセットアップする必要があります。

「**アクション・キュー・マネージャー**」フィールドを使用して、「**ターゲット・キュー・マネージャー**」フィールドに指定したキュー・マネージャーと同じキュー共有グループにあるキュー・マネージャーを、要求したアクションを実行するキュー・マネージャーとして指定することができます。 このフィールドに * を指定すると、 要求したアクションはキュー共有グループ内のすべてのキュー・マネージャーで実行されます。 このフィールドをブランクにしておくと、「**Target queue manager (ターゲット・キュー・マネージャー)**」フィールドに 指定された値にデフォルト設定されます。 「**アクション・キュー・マネージャー**」フィールドは、 MQSC コマンドで説明されている CMDSCOPE コマンド修飾子を使用した場合に対応します。

キュー・マネージャーのデフォルト

何らかのキュー・マネージャーのフィールドをブランクにした場合、またはキュー共有グループに接続することにした場合、 **Enter** を押すと 2 次ウィンドウが開きます。 このウィンドウは、後で使用するキュー・マネージャーの名前を確認します。 **Enter** キーを押して続行します。 いくつかの要求を行ったあと、初期パネルに戻ると、フィールドに実際の名前が入力されていることが分かります。

キュー・マネージャー・レベル

操作パネルと制御パネルは、z/OS の IBM WebSphere MQ 710 以降で実行されているキュー・マネージャーでのみ、正しく機能します。

この条件が満たされない場合、アクションは部分的に機能するか、 正しく機能しないか、まったく機能しないのいずれかになり、 キュー・マネージャーからの応答は認識されません。

アクション・キュー・マネージャーが IBM MQ 8.0.0 以上でない場合、一部のフィールドは表示されず、また、一部の値を入力できません。 使用できないオブジェクトおよびアクションもいくつかあります。 そのような場合は、 処理を続けるかどうかを確認するための 2 次ウィンドウが開きます。

z/OS *z/OS での ISPF 制御パネルによるファンクション・キーおよびコマンド行の使用*

パネルを使用するには、ファンクション・キーを使用するか、同等のコマンドを ISPF 制御パネルのコマンド域に入力する必要があります。

- ファンクション・キー
  - アクションの処理
  - 440 ページの『IBM MQ ユーザー・メッセージの表示』
  - アクションの取り消し
  - ヘルプの表示
- コマンド行の使用

ファンクション・キー

ファンクション・キーには、IBM MQ 用の特別の設定値があります。 (これはファンクション・キーには ISPF デフォルト値を使用できないことを意味します。以前にどこかで KEYLIST OFF ISPF コマンドを使用したことがある場合は、任意の操作および制御パネルのコマンド域に KEYLIST ON を入力して Enter を押し、IBM MQ 設定値を使用可能にしなければなりません。)

必要であれば、これらのファンクション・キー設定値をパネルに表示できます (441 ページの図 36 を参照)。 設定値が表示されない場合は、 いずれかの操作および制御パネルのコマンド域に PFSHOW と入力し、**Enter** キーを押してください。 設定値の表示を除去するには、コマンド PFSHOW OFF を使用します。

操作および制御パネル内のファンクション・キーの設定値は、CUA 標準に従っています。 通常の ISPF プロシージャー (**KEYLIST** ユーティリティーなど) でキーの設定値を変更することはできますが、お勧めできません。

注 : PFSHOW および **KEYLIST** コマンドを使用すると、 使用している他の論理 ISPF 画面に影響し、 操作および制御パネルを終了してもコマンドによる設定値が残ってしまいます。

アクションの処理
パネル上で要求されたアクションを実行するには、**Enter** キーを押します。 パネルからの情報は、キュー・マネージャーに送られて処理されます。

パネルで **Enter** キーを押すたびに、IBM MQ は 1 つ以上のオペレーター・メッセージを生成します。 操作が正常に実行されると、確認メッセージの CSQ9022I を受け取ります。正常でない場合は、エラー・メッセージを受け取ります。

IBM MQ ユーザー・メッセージの表示
IBM MQ のユーザー・メッセージを見るには、任意のパネルでファンクション・キー F10 を押します。

アクションの取り消し
初期パネルでは、F3 および F12 のどちらを使用しても、操作および制御パネルから出て、ISPF に戻ります。 キュー・マネージャーには、情報は何も送られません。

その他のパネルでファンクション・キー F3 または F12 を押すと、現在のパネルはそのまま残り、**Enter キーを最後に押してから入力したすべてのデータは無視されます**。 この場合も、キュー・マネージャーには、情報は何も送られません。

- F3 を押すと、初期パネルに直接戻ります。
- F12 を押すと、1 つ前のパネルに戻ります。

ヘルプの使用

各パネルは、それぞれ関連するヘルプ・パネルを持っています。 ヘルプ・パネルでは、次の ISPF プロトコルを使用します。

- タスクについての一般ヘルプ (全般ヘルプ) を見るには、ファンクション・キー F1 を押します。
- フィールドについての特別のヘルプを見るには、 そのフィールドにカーソルを置いてファンクション・キー F1 を押します。
- 一般ヘルプを表示するには、 フィールド・ヘルプ・パネルからファンクション・キー F5 を押します。
- 基本パネル (つまり、ファンクション・キー F1 を押したパネル) へ戻るには、 ファンクション・キー F3 を押します。
- ファンクション・キーのヘルプを表示するには、ヘルプ・パネルからファンクション・キー F6 を押します。

ヘルプ情報が 2 ページ目以降に続いている場合は、パネルの右上に「**続く**」と表示されます。 ヘルプ・ページを移動するには、次のファンクション・キーを押します。

- 次のヘルプ・ページに進む場合には、F11 (次ページがある場合)
- 前のヘルプ・ページに戻る場合には、F10 (前ページがある場合)

コマンド行の使用

操作および制御パネルが使用するコマンドをコマンド行を使用して実行する必要はまったくありません。 これらのコマンドはファンクション・キーから使用可能だからです。 コマンド行は、通常の ISPF コマンド (**PFSHOW** など) を入力できるようにするために提供されています。

ISPF コマンドの PANELID ON は、現行の CSQOREXX パネルの名前を表示します。

ISPF の設定内容に関係なく、初期設定ではコマンド行はパネルの下部のデフォルト位置に表示されます。どの操作および制御パネルからでも SETTINGS ISPF コマンドを使用して、コマンド行の位置を変更できます。 変更した後の設定値は操作および制御パネルに記憶され、 それ以後のセッションに適用されます。

z/OS での操作および制御パネルの使用

このトピックを使用して、CSQOREXX から表示される初期の制御パネルについて調べます。

441 ページの図 36 は、パネル・セッションを開始したときに表示されるパネルを示しています。

```
                  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
```

図 *36. IBM MQ* 操作および制御の初期パネル

このパネルから、以下のようなアクションを実行できます。

- 使用したいローカル・キュー・マネージャーを選択し、そのキュー・マネージャー、リモート・キュー・マネージャー、またはローカル・キュー・マネージャーと同じキュー共有グループの別のキュー・マネージャーのどれからコマンドを実行するかを選択します。 変更する必要があれば、キュー・マネージャー名を上書きします。
- 「**Action (アクション)**」フィールドに適切な番号を入力することによって、実行したいアクションを選択します。
- 作業を進めたいオブジェクト・タイプを指定します。 確信がないときにオブジェクトの種類を表示するには、 ファンクション・キー F1 を押します。
- 使用したいオブジェクトの属性指定を指定します。
- 指定されたタイプのオブジェクトのリストを表示します。 すでにアクション・キュー・マネージャー上に定義されているオブジェクト (指定したタイプのもの) の リストを表示するには、「**Name (名前)**」フィールドにアスタリスク (*) を 入力して **Enter** キーを押します。 次に、1 つまたは複数のオブジェクトを選択して、それらを順次に処理できます。 すべてのアクションがリストから使用できます。

注 : 表示されているオブジェクトのリストの結果から選択し、このリストで作業することをお勧めします。すべてのオブジェクト・タイプに対して使用できる 「**Display (表示)**」 アクションを使用してください。

z/OS でのコマンド機能の使用

エディターを使用して、キュー・マネージャーに受け渡す MQSC コマンドを入力または訂正します。

コマンド機能を開始するには、基本パネル CSQOPRIA からオプション「**8 コマンド**」を選択します。

CSQUTIL COMMAND 関数への入力として使用される順次ファイル *prefix*.CSQUTIL.COMMANDS (IBM MQ へのコマンドの実行を参照) の編集セッションが開始します。

コマンドの先頭にコマンド接頭部ストリング (CPF) を付ける必要はありません。

継続文字 **+** または **-**で現在行を終了することにより、後続の行で MQSC コマンドを続行することができます。 あるいは、行編集モードを使用して、長い MQSC コマンドまたはコマンド内の長い属性値の値を指定します。

行編集

行編集モードを使用するには、編集パネル内で適切な行にカーソルを移動し、**F4** を使用してスクロール可能なパネルに 1 行を表示します。 1 行のデータ量は、32 760 バイトまでです。

行編集を終了するには、次のようにします。

- **F3 (終了)** を使用すると、行に加えた変更が保存されて終了します。
- **F12 (取り消し)** を使用すると、編集パネルに戻り、行に対する変更は破棄されます。

編集セッションで加えられた変更を破棄する場合は、**F12 (取り消し)** を使用して、ファイルの内容を変更せずに編集セッションを終了してください。 コマンドは実行されません。

コマンドの実行

MQSC コマンドを入力し終えたら、 **F3 (終了)** を使用して編集セッションを終了してファイルの内容を保存し、 CSQUTIL を呼び出してキュー・マネージャーにコマンドを渡します。 コマンド処理の出力は、*prefix*.CSQUTIL.OUTPUT というファイルに保持されます。 このファイルの編集セッションが自動的に開始し、応答を参照できます。 このセッションを終了してメインメニューに戻るには、「**F3 終了**」を押します。

z/OS での IBM MQ オブジェクトの処理

本書で説明しているタスクの多くには、IBM MQ オブジェクトの操作が含まれています。 オブジェクト・タイプは、キュー・マネージャー、キュー、プロセス定義、名前リスト、チャネル、クライアント接続チャネル、リスナー、サービス、および認証情報オブジェクトです。

- 単純なキュー・オブジェクトの定義
- 他の種類のオブジェクトの定義
- オブジェクト定義の処理
- 名前リストの処理

単純なキュー・オブジェクトの定義

新規オブジェクトを定義するには、既存の定義を元にしてください。 このことは次の 3 つのうちいずれかの方法で行えます。

- 初期画面で選択したオプションの結果として表示されるリストのメンバーをオブジェクトに選択します。 その後、選択したオブジェクトに隣接するアクション・フィールドに、アクション・タイプ 2 (「**Define like (類似定義)**」) を入力します。 新しいオブジェクトには、選択したオブジェクトの属性 (属性指定を除く) が付与されます。 その後、必要に応じて新規オブジェクトの属性を任意に変更できます。
- 初期パネルで、「**Define like (類似定義)**」アクション・タイプを選択し、 定義するオブジェクトのタイプを「**Object type (オブジェクト・タイプ)**」フィールドに入力し、 特定の既存のオブジェクトの名前を「**Name (名前)**」フィールドに入力します。 新規に作成したオブジェクトの属性には、「**Name (名前)**」フィールドに 指定したオブジェクトと同じ属性 (属性指定を除く) が付与されます。 その後、必要に応じて新規オブジェクトの属性定義を任意に変更できます。
- 「**Define like (類似定義)**」アクション・タイプを選択し、 オブジェクト・タイプを選択してから、「**Name (名前)**」フィールドを ブランクにしておきます。 それから、新規オブジェクトを定義すると、 そのオブジェクトにはインストール用に定義したデフォルト属性が付与されます。 その後、必要に応じて新規オブジェクトの属性定義を任意に変更できます。

注 : 初期パネルでは定義するオブジェクトの名前を入力しませんが、 後に表示される「**Define (定義)**」パネルで入力することになります。

以下の例では、既存のキューをテンプレートとして使用するローカル・キューを定義する方法について説明します。

ローカル・キューの定義

操作および制御パネルからローカル・キュー・オブジェクトを定義するには、 既存のキュー定義を新規定義の元として使用します。 いくつかのパネルで設定作業を行います。 すべてのパネルで設定作業

が完了し、希望どおりに属性が定義されたら、Enter キーを押してそれらの定義をキュー・マネージャーに送信して、 そのキュー・マネージャーが実際のキューを作成できるようにします。

初期パネル上で、または初期パネルで選択されたオプションの結果として表示されるリストのオブジェクト項目の どちらかに対して、「**類似定義**」アクションを使用します。

例えば、初期パネルから始めて、以下のフィールドすべてを設定します。

アクション	2 (類似定義)
オブジェクト・タイプ	QLOCAL
名前	QUEUE.YOU.LIKE. 新しいキューに属性を提供するキューの名前です。

Enter キーを押して、「**Define a Local Queue (ローカル・キューの定義)**」パネルを 表示します。 キュー名フィールドはブランクになっているので、 新しいキューの名前を指定できます。 この説明は、この新規定義が基づいているキューに関する説明です。 このフィールドに、新しいキューに関する独自の説明を上書きします。

他のフィールドの値は、新規キューの元になっているキューの値 (属性指定を除く) です。 必要に応じて、これらのフィールドを上書きできます。 例えば、適切な許可を持ったアプリケーションが、 このキューにメッセージを書き込むことができる場合は、「**Put enabled (PUT 可能)**」フィールドに Y と入力します (まだ Y になっていない場合)。

ある 1 つのフィールドにカーソルを移動し、ファンクション・キー F1 を押すと、 フィールド・ヘルプが表示します。 フィールド・ヘルプは、 各属性に使用できる値についての情報を提供します。

最初のパネルの入力を完了したら、ファンクション・キー F8 を押して、2 番目のパネルを表示します。

ヒント :

1. この段階では Enter キーを押さないでください。押すと、残りのフィールドが作成される前にキューが作成されてしまいます。 (Enter キーを押すのが早すぎた場合でも心配ありません。 定義内容は後でいつでも変更できます。)
2. ファンクション・キー F3 、F12 はどちらも押さないでください。 押すと、入力したデータが失われます。

ファンクション・キー F8 を繰り返し押して残りのパネル (例えばトリガー定義、 イベント制御、バックアウト・レポートなどのパネル) を表示し、必要な情報を入力してください。

ローカル・キュー定義の入力が完了した場合

定義の入力が完了したら Enter キーを押し、情報を処理のためにキュー・マネージャーに送ります。 キュー・マネージャーは、指定された定義に従ってキューを作成します。 キューを作成したくない場合はファンクション・キー F3 を押し、 定義を終了して取り消します。

他の種類のオブジェクトの定義

他のタイプのオブジェクトを定義するには、ローカル・キューの定義に説明されているように、既存の定義に基づいて新しい定義を作成してください。

初期パネル上で、または初期パネルで選択されたオプションの結果として表示されるリストのオブジェクト項目の どちらかに対して、「**類似定義**」アクションを使用します。

例えば、初期パネルから始めて、以下のフィールドすべてを設定します。

アクション	2 (類似定義)
オブジェクト・タイプ	QALIAS、NAMELIST、PROCESS、CHANNEL、および他のリソース・オブジェクト。
名前	ブランクのままにするか、同じ種類の既存のオブジェクトの名前を入力します。

Enter キーを押して、対応する「DEFINE (定義)」パネルを表示します。 必要に応じてフィールドに入力したあと、その後その情報をキュー・マネージャーに送るために、再び Enter キーを押します。

ローカル・キューの定義と同様に、別のタイプのオブジェクトの定義には、 一般に入力するパネルがいくつかあります。 名前リストの定義には、さらに追加の作業があります。これについては、 444 ページの『名前リストの処理』 に説明があります。

オブジェクト定義の処理

いったんオブジェクトを定義すると、「**Action (アクション)**」フィールドにアクションを指定して、 オブジェクトの変更、表示、または管理ができます。

その場合、次のどちらかを行えます。

- 初期パネル上で選択したオプションによって表示されたリストから、 作業したいオブジェクトを選択します。 例えば、表示するオブジェクトの「**アクション**」フィールドに 1 を 入力し、「**オブジェクト・タイプ**」フィールドに Queue を 入力し、「**名前**」フィールドに * を入力すると、 システム内で定義されているすべてのキューのリストが表示されます。 こうしてこのリストから、作業する必要のあるキューを選択できます。
- 初期パネルから始めて、そのパネルで「**Object type (オブジェクト・タイプ)**」フィールド と「**Name (名前)**」フィールドに必要な情報を入力することにより、 処理するオブジェクトを指定します。

オブジェクト定義の変更
オブジェクト定義を変更するには、アクション 3 を指定して Enter キーを押し、 「ALTER (変更)」パネルを表示します。 これらのパネルは、「DEFINE (定義)」パネルと非常によく似ています。 必要な値を変更することができます。 変更が完了したら、Enter キーを押して、その情報をキュー・マネージャーに送ります。

オブジェクト定義の表示
オブジェクトの詳細を表示するだけで変更できないようにする場合は、アクション 1 を指定して Enter キーを押し、「DISPLAY (表示)」パネルを表示します。 この場合も、これらのパネルは「DEFINE (定義)」パネルによく似ていますが、 フィールドに変更を加えることはできません。 オブジェクト名を変更して、 別のオブジェクトの詳細を表示します。

オブジェクトの削除
オブジェクトを削除するには、アクション 4 (「Manage (管理)」) を指定します。 その結果表示されるメニューのアクションの 1 つに「**Delete (削除)**」アクションがあります。 「**Delete (削除)**」アクションを選択します。

要求を確認するよう求められます。 ファンクション・キー F3 または F12 を押すと、要求は取り消されます。 Enter キーを押すと、 要求は確認され、キュー・マネージャーへ渡されます。 その後、指定したオブジェクトは削除されます。

注 : チャネル・オブジェクトの大半は、チャネル・イニシエーターが開始されていなければ削除できません。

名前リストの処理

名前リストを処理する場合は、 他のオブジェクトにする場合と同じように処理します。

「DEFINE LIKE (類似定義)」または「ALTER (変更)」のアクションの場合、 名前をリストに追加したりリスト内の名前を変更したりするには、 ファンクション・キー F11 を押してください。 この作業には、ISPF エディターを使用します。 通常の ISPF 編集コマンドは、すべて使用できます。 別の行の名前リストにそれぞれの名前を入力してください。

このようにして ISPF 編集プログラムを使用する場合、ファンクション・キーの設定値は通常の ISPF の設定値であり、 他の操作および制御パネルで使用される**設定値ではありません**。

小文字で名前を指定する必要がある場合は、 エディター・パネルのコマンド行に、CAPS (OFF) を指定してください。 これを指定すると、それ以後に編集するすべての名前リストは、 CAPS (ON) を指定するまで小文字になります。

名前リストの編集を完了したら、ファンクション・キー F3 を押して、 ISPF 編集セッションを終了します。 次に Enter キーを押して、変更をキュー・マネージャーへ送ります。

注意 : この段階で、 Enter キーを押さないでファンクション・キー F3 を押すと、 入力した更新情報はすべて失われます。

z/OS 複数のクラスター伝送キューを使用したシステムの実装

チャネルが単一クラスターで使用されてもオーバーラップ・クラスターで使用されても、違いはありません。 チャネルが選択されて開始されると、チャネルは定義に応じて伝送キューを選択します。

手順

- DEFCLXQ オプションを使用する場合は、445 ページの『キューの自動定義の使用と切り替え』を参照してください。
- 段階的アプローチを使用する場合は、445 ページの『段階的アプローチを使用したクラスター送信側チャネルの変更』を参照してください。

z/OS キューの自動定義の使用と切り替え

DEFCLXQ オプションを使用する予定がある場合は、このオプションを使用します。 各チャネル、および各新規チャネルに対して、1 つのキューが作成されます。

手順

1. SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE の定義を確認し、必要に応じて属性を変更します。
   このキューは、メンバー SCSQPROC(csq4insx)で定義されます。
2. SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE モデル・キューを作成します。
3. このモデル・キューおよび SYSTEM.CLUSTER.TRANSMIT。* * キュー。
   z/OS では、チャネル・イニシエーター開始タスクのユーザー ID は、以下を必要とします。
   - 次の CLASS(MQADMIN) への制御アクセス

     ```
     ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channelname
     ```

   - 次の CLASS(MQQUEUE) への更新アクセス

     ```
     ssid.SYSTEM.CLUSTER.TRANSMIT.channelname
     ```

z/OS 段階的アプローチを使用したクラスター送信側チャネルの変更

段階的アプローチを使用する場合は、このオプションを使用します。 このプロセスによって、企業内のニーズに対応するように、場合に応じて新規クラスター送信側チャネルに移動することができます。

始める前に

- ビジネス・アプリケーションと、使用するチャネルを特定します。
- 使用するキューが存在するクラスターを表示します。
- チャネルを表示して、接続名、リモート・キュー・マネージャー名、およびチャネルがサポートするクラスターを表示します。

このタスクについて

- 伝送キューを作成します。 z/OS では、キューに使用するページ・セットを検討する必要があります。
- キューのセキュリティー・ポリシーをセットアップします。
- キュー・モニターを、このキュー名を含むように変更します。
- この伝送キューを使用するチャネルを決定します。 チャネルは名前が似ているため、CLCHNAME の総称文字「*」でチャネルを特定します。

- 新しい機能を使用する準備ができたら、この伝送キューを使用するチャネルの名前を指定するよう伝送キューを変更します。 例えば、 CLUSTER1.TOPARIS、または CLUSTER1.* または *.TOPARIS
- チャネルを開始します。

手順

1. DIS CLUSQMGR(xxxx) XMITQ コマンドを使用して、クラスターで定義されているクラスター送信側チャネルを表示します (*xxxx* はリモート・キュー・マネージャーの名前)。
2. 伝送キューのセキュリティー・プロファイルをセットアップして、キューにチャネル・イニシエーターへのアクセス権を付与します。
3. 使用される伝送キューを定義し、USAGE(XMITQ) INDXTYPE(CORRELID) SHARE および CLCHNAME(*value*) を指定します。

   チャネル・イニシエーター開始タスク・ユーザー ID は、以下のアクセスを必要とします。

   ```
   alter class(MQADMIN) ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channel
   update class(MQQUEUE ssid.SYSTEM.CLUSTER.TRANSMIT.channel
   ```

   また、SWITCH コマンドを使用するユーザー ID は、以下のアクセスを必要とします。

   ```
   alter cl(MQADMIN) ssid.QUEUE.queuename
   ```

4. チャネルを停止してから再始動します。

   チャネルが MQSC コマンドを使用して開始された場合、またはユーザーが CSQUTIL を使用した場合、チャネル変更が発生します。 CSQUTIL の SWITCH CHANNEL(*)STATUS を使用して、再始動する必要のあるチャネルを特定することができます。

   チャネルの開始時に問題が発生した場合は、チャネルを停止して問題を解決し、チャネルを再始動します。

   CLCHNAME 属性は、必要な頻度で変更することができます。

   使用される CLCHNAME の値は、チャネルの開始時の値であるため、チャネルがその開始時から定義を使い続けているときでも、CLCHNAME 定義を変更することができます。 チャネルが再始動されると、新しい定義が使用されます。

z/OS での変更の取り消し

変更の結果が期待どおりではなかった場合、変更をバックアウトする処理が必要です。

起こり得る問題

新規伝送キューが期待と異なる場合には、以下のようにします。

1. CLCHNAME が期待のものであるかどうかを確認します。
2. ジョブ・ログを見て、切り替えプロセスが終了しているかどうかを確認します。 終了していない場合は、待機して、チャネルの新規伝送キューを後で確認します。

複数のクラスター伝送キューを使用している場合は、伝送キュー定義を明示的に設計し、複雑な重複構成を避けることが重要です。 そうすることで、問題があった場合に、確実に元のキューや構成に戻ることができます。

別の伝送キューの使用に移行しているときに問題が発生した場合、変更を続行するには、その前にすべての問題を解決する必要があります。

新しい変更要求を行う前に、既存の変更要求が完了する必要があります。 例えば、次のようになります。

1. 最大サイズ 1 で新しい伝送キューを定義します。送信待ちのメッセージが 10 個あります。
2. CLCHNAME パラメーターでチャネル名を指定するように伝送キューを変更します。

3. チャネルを停止してから再始動します。 メッセージを移動しようとすると失敗し、問題が報告されます。
4. 伝送キューの CLCHNAME パラメーターを変更してブランクにします。
5. チャネルを停止してから再始動します。 チャネルは引き続き元の要求を完了しようとするため、チャネルは新規伝送キューを使用し続けます。
6. メッセージの移動が正常に完了するには、問題を解決してからチャネルを再始動する必要があります。

チャネルは、次に再始動されたときに変更点を検出するので、CLCHNAME をブランクに設定した場合、チャネルは指定の伝送キューを使用しなくなります。

この例で、伝送キューの CLCHNAME をブランクに変更した場合に、必ずしもチャネルが SYSTEM.CLUSTER.TRANSMIT キューを使用するわけではありません。これは、CLCHNAME パラメーターがチャネル名と一致する他の伝送キューがある可能性があるためです。 例えば、総称名、またはキュー・マネージャー属性 DEFCLXQ がチャネルに設定されていると、チャネルは SYSTEM.CLUSTER.TRANSMIT キューの代わりに動的キューを使用します。

z/OS IBM MQ for z/OS 管理のためのプログラムの作成

独自のアプリケーション・プログラムを作成して、キュー・マネージャーを管理することができます。 このトピックでは、独自の管理プログラムを作成するための要件について知ることができます。

汎用プログラミング・インターフェース情報の始まり

この一連のトピックでは、IBM MQ アプリケーション・プログラムから IBM MQ コマンドを実行するためのヒントおよび手順を記載しています。

注 : このトピックでは、MQI 呼び出しは C 言語表記法で書かれています。 COBOL、 PL/I、およびアセンブラーの各言語における呼び出し (コール) の典型的な呼び出しについては、関数呼び出しの資料を参照してください。

コマンド実行の手順について

概観すると、アプリケーション・プログラムからコマンドを実行するための手順は次のようになります。

1. IBM MQ コマンドを、要求メッセージと呼ばれるタイプの IBM MQ メッセージに組み込みます。 コマンドは MQSC または PCF 形式にできます。
2. このメッセージを、システム・コマンド入力キューと呼ばれる特別なキューに送ります (MQPUT を使用します)。 IBM MQ コマンド・プロセッサーはそのコマンドを実行します。
3. コマンドの実行結果を、応答メッセージ として応答先キューから読み取ります (MQGET を使用します)。 これらのメッセージには、コマンドが正常に実行されたかどうか、および正常に実行された場合は、 その結果がどうであったかを判別するために必要な、 ユーザー・メッセージが入っています。

このあとの処理は、アプリケーション・プログラムが行います。

この一連のトピックには、以下が含まれています。

z/OS 管理用プログラムのためのキューの準備

管理用プログラムでは、システム・コマンド入力用の、および応答を受信するための定義済みのキューがいくつか必要です。

この節では、MQSC 形式のコマンドについて説明します。 PCF の同等のコマンドについては、25 ページの『IBM MQ プログラマブル・コマンド・フォーマットの使用』を参照してください。

MQPUT 呼び出しまたは MQGET 呼び出しを実行するためには、 まず使用するキューを定義し、それをオープンしておく必要があります。

システム・コマンド入力キューの定義

システム・コマンド入力キューは、SYSTEM.COMMAND.INPUT と呼ばれるローカル・キューです。 提供された CSQINP2 の初期設定データ・セット thlqual.SCSQPROC(CSQ4INSG) には、 システム・コマンド入力キューのデフォルトの定義が入っています。 他のプラットフォーム上の IBM MQ との互換性のために、 SYSTEM.ADMIN.COMMAND.QUEUE も提供されています。 詳しくは、 IBM MQ で提供されるサンプル定義 を参照してください。

応答先キューの定義

IBM MQ コマンド・プロセッサーから応答メッセージを受け取るために、応答先キューを定義する必要があります。 この応答先キューは、応答メッセージの書き込みが許される属性を持つキューであれば、どんなキューでも構いません。 ただし、通常の操作では、次の属性を指定します。

- USAGE(NORMAL)
- NOTRIGGER (アプリケーションでトリガーを使用しない場合)

持続メッセージをコマンドに使用しないでください。それでも持続メッセージをコマンドに使用する場合は、応答先キューを一時動的キューにしないでください。

提供された CSQINP2 の初期設定データ・セット thlqual.SCSQPROC(CSQ4INSG) には、SYSTEM.COMMAND.REPLY.MODEL と呼ばれるモデル・キューの定義が入っています。 このモデルを使用して、動的応答先キューを作成することができます。

注 : コマンド・プロセッサーによって生成される応答は、最大 15 000 バイト長になる可能性があります。

応答先キューとして永続動的キューを使用する場合、キューの削除を試みる前に、アプリケーションですべての PUT および GET 操作が完了する時間を確保しておく必要があります。そうでない場合、MQRC2055 (MQRC_Q_NOT_EMPTY) が戻されることがあります。 このようになった場合は、数秒後にキューの削除を再試行します。

システム・コマンド入力キューのオープン

システム・コマンド入力キューをオープンするには、アプリケーションがキュー・マネージャーに接続されている必要があります。 このことを行うには、MQI 呼び出し MQCONN または MQCONNX を使用します。

次に MQI 呼び出し MQOPEN を使用して、システム・コマンド入力キュー をオープンします。 この呼び出しは次のように使用します。

1. **Options** パラメーターを MQOO_OUTPUT に設定します。
2. MQOD オブジェクト記述子フィールドを、次のように設定します。

   ObjectType
   MQOT_Q (オブジェクトはキューです。)

   ObjectName
   SYSTEM.COMMAND.INPUT

   ObjectQMgrName
   要求メッセージをローカル・キュー・マネージャーに送りたい場合は、 このフィールドをブランクのままにします。 これは、コマンドがローカルで処理されることを意味します。

   IBM MQ コマンドをリモート・キュー・マネージャー上で処理する場合は、その名前をここで書き込みます。 また、 分散キューイングおよびクラスターで説明されているように、正しいキューおよびリンクがセットアップされている必要があります。

応答先キューのオープン

IBM MQ コマンドからの応答を取り出すには、応答先キューをオープンする必要があります。 これを行う方法の 1 つは、モデル・キュー SYSTEM.COMMAND.REPLY.MODEL を MQOPEN 呼び出しに指定することによって、永続動的キューを応答先キューとして作成することです。 この呼び出しは次のように使用します。

1. **Options** パラメーターを MQOO_INPUT_SHARED に設定します。
2. MQOD オブジェクト記述子フィールドを、次のように設定します。

   ObjectType
   MQOT_Q (オブジェクトはキューです。)

   ObjectName
   応答先キューの名前。 指定したキュー名がモデル・キュー・オブジェクトの名前である場合、キュー・マネージャーは動的キューを作成します。

   ObjectQMgrName
   ローカル・キュー・マネージャーで応答を受け取るには、 このフィールドをブランクのままにします。

   DynamicQName
   作成する動的キューの名前を指定します。

z/OS コマンド・サーバーの使用

コマンド・サーバーは、コマンド・プロセッサー・コンポーネントと共に作動する、IBM MQ のコンポーネントです。 コマンド・サーバーにはフォーマット設定メッセージを送信することができ、コマンド・サーバーはそのメッセージを解釈し、管理要求を実行し、および応答を管理アプリケーションに返します。

コマンド・サーバーは、要求メッセージをシステム・コマンド入力キューから読み取り、 それらを検査し、有効なものをコマンド・プロセッサーにコマンドとして渡します。 コマンド・プロセッサーは、コマンドを処理し、 すべての応答を応答メッセージとして、指定された応答先キューに書き込みます。 最初の応答メッセージには、ユーザー・メッセージ CSQN205I が入っています。 詳しくは、 453 ページの『コマンド・サーバーからの応答メッセージの解釈』 を参照してください。 コマンド・サーバーは、チャネル・イニシエーターおよびキュー共有グループのコマンドがどこから実行された場合でも、それらの処理も行います。

コマンドを処理するキュー・マネージャーの識別

管理用プログラムから実行したコマンドを処理するキュー・マネージャーは、 メッセージの書き込み先となるシステム・コマンド入力キューを所有しているキュー・マネージャーです。

コマンド・サーバーを開始する

通常、コマンド・サーバーは、キュー・マネージャーが開始したときに、 自動的に開始します。 コマンド・サーバーは、メッセージ CSQ9022I 'START QMGR' NORMAL COMPLETION が、 START QMGR コマンドから戻されると、ただちに使用可能になります。 コマンド・サーバーは、システム終了フェーズで、接続されているすべてのタスクが切り離されたときに停止します。

コマンド・サーバーは、 START CMDSERV および STOP CMDSERV コマンドを使用して制御できます。 IBM MQ が再始動したときにコマンド・サーバーが自動的に開始しないように、STOP CMDSERV コマンドを CSQINP1 または CSQINP2 初期設定データ・セットに追加することができます。 ただし、これは、チャネル・イニシエーターまたはキュー共有グループのコマンドの処理の妨げになるので、お勧めできません。

STOP CMDSERV コマンドは、現在のメッセージの処理が完了した直後、 メッセージが処理中でなければただちに、コマンド・サーバーを停止します。

コマンド・サーバーが、プログラムの中の STOP CMDSERV コマンドで停止した場合、 そのプログラムの他のコマンドは処理することができなくなります。 コマンド・サーバーを再始動するためには、z/OS コンソールから START CMDSERV コマンドを出す必要があります。

キュー・マネージャーの実行中に、コマンド・サーバーを停止して再始動した場合、 コマンド・サーバーが停止したときにシステム・コマンド入力キューにあったすべてのメッセージは、 コマンド・サーバーが再始動したときに処理されます。 ただし、コマンド・サーバーが停止した後でキュー・マネージャーを停止して再始動した場合は、 コマンド・サーバーが再始動したときには、 システム・コマンド入力キュー上の持続メッセージのみが処理されます。 システム・コマンド入力キューに入っているすべての非持続メッセージは失われます。

コマンド・サーバーへのコマンドの送信

各コマンドについて、そのコマンドが含まれるメッセージを作成し、 そのメッセージをシステム・コマンド入力キューに書き込みます。

IBM MQ コマンドが含まれたメッセージの作成

必要な IBM MQ コマンドが設定された要求メッセージを作成することによって、そのコマンドをアプリケーション・プログラムに組み込むことができます。 このような各コマンドについて、次のようにします。

1. コマンドを表す文字ストリングが入ったバッファーを作成します。
2. 呼び出しの **buffer** パラメーターにそのバッファー名を指定した MQPUT 呼び出しを実行します。

C 言語でこれを行う最も簡単な方法は、「char」を使用してバッファーを定義することです。 以下に例を示します。

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

コマンドを作成する場合は、NULL 文字で終了する文字ストリングを使用してください。 このようにして定義されたコマンドの冒頭に、 コマンド接頭部ストリング (CPF) を指定しないでください。 このようにしておくと、コマンド・スクリプトを他のキュー・マネージャーで実行したい場合に、コマンド・スクリプトを変更する必要がありません。 ただし、応答先キューに書き込まれるすべての応答メッセージに CPF が組み込まれることを考慮する必要があります。

コマンド・サーバーは、引用符で囲まれていない小文字をすべて大文字に変換します。

コマンドの長さは、最大 32 762 文字です。

システム・コマンド入力キューへのメッセージの書き込み

コマンドが設定された要求メッセージをシステム・コマンド入力キューに書き込むには、 MQPUT 呼び出しを使用します。 この呼び出しでは、すでにオープンされている応答先キューの名前を指定します。

MQPUT 呼び出しを使用するには、次のようにします。

1. 次のような MQPUT のパラメーターを設定します。

   Hconn
   MQCONN または MQCONNX 呼び出しによって戻された接続ハンドル。

   Hobj
   システム・コマンド入力キューに対する MQOPEN 呼び出しによって戻されたオブジェクト・ハンドル。

   BufferLength
   フォーマット後のコマンドの長さ。

   Buffer
   コマンドが入っているバッファーの名前。
2. 次のような MQMD のフィールドを設定します。

   MsgType
   MQMT_REQUEST

   Format
   MQFMT_STRING または MQFMT_NONE

   キュー・マネージャーと同じコード・ページを使用していない 場合は、*CodedCharSetId* (適切な場合) および MQFMT_STRING を設定し、 コマンド・サーバーがメッセージを変換できるようにします。 コマンドが PCF として解釈されるため、MQFMT_ADMIN は設定しないでください。

ReplyToQ
応答先キューの名前。

ReplyToQMgr
応答をローカル・キュー・マネージャーに送りたい場合は、 このフィールドをブランクのままにします。 IBM MQ コマンドをリモート・キュー・マネージャーへ送る場合は、その名前をここで書き込みます。 また、 分散キューイングおよびクラスターで説明されているように、正しいキューおよびリンクがセットアップされている必要があります。

3. 必要に応じて、MQMD の他のフィールドを設定します。 通常、コマンドには非持続メッセージを使用してください。
4. 必要に応じて、*PutMsgOpts* オプションを設定します。

   MQPMO_SYNCPOINT (デフォルト) を指定した場合、 MQPUT 呼び出しの後に同期点呼び出しを行う必要があります。

MQPUT1 およびシステム・コマンド入力キューの使用
1 つのメッセージだけをシステム・コマンド入力キューに書き込みたい場合は、**MQPUT1** 呼び出しを使用することができます。 この呼び出しは、**MQOPEN** 機能、それに続く 1 メッセージの **MQPUT** 機能、さらにそれに続く **MQCLOSE** 機能を、1 つの呼び出しに結合したものです。 この呼び出しを使用する場合は、パラメーターを適宜修正してください。 詳しくは、MQPUT1 呼び出しを使用してキューに 1 つのメッセージを書き込むを参照してください。

z/OS コマンドに対する応答の取り出し

コマンド・サーバーは、受信する要求メッセージごとに、応答を応答キューに送信します。 どの管理アプリケーションも、応答メッセージを受信して処理する必要があります。

コマンド・プロセッサーがコマンドを処理すると、何らかの応答メッセージが、MQPUT 呼び出しで指定した応答先キューに書き込まれます。 コマンド・サーバーは、受信したコマンド・メッセージと同じ持続性で応答メッセージを送信します。

応答の待機

要求メッセージの応答を取り出すには、MQGET 呼び出しを使用します。 1 つの要求メッセージで、複数の応答メッセージが生成されることがあります。 詳しくは、『453 ページの『コマンド・サーバーからの応答メッセージの解釈』』を参照してください。

MQGET 呼び出しが、応答メッセージの生成を待つ時間を指定できます。 応答が届かない場合には、453 ページの『応答が受信されない場合』のトピックのチェックリストを使用してください。

MQGET 呼び出しを使用するには、次のようにします。

1. 次のパラメーターを設定します。

   Hconn
   MQCONN または MQCONNX 呼び出しによって戻された接続ハンドル。

   Hobj
   MQOPEN 呼び出しが戻した応答先キューのオブジェクト・ハンドル。

   Buffer
   応答を受け取るための区域の名前。

   BufferLength
   応答を受け取るためのバッファーの長さ。 これは、最低でも 80 バイトでなければなりません。
2. 発行したコマンドからの応答のみを受け取るようにするには、適切な *MsgId* および *CorrelId* フィールドを指定する必要があります。 これらのフィールドの指定は、MQPUT 呼び出しで指定したレポート・オプション MQMD_REPORT によって異なります。

   MQRO_NONE
   2 進ゼロ '00...00' (24 個の NULL)

MQRO_NEW_MSG_ID
2 進ゼロ '00...00' (24 個の NULL)

これは、これらのオプションを指定しなかった場合のデフォルトです。

MQRO_PASS_MSG_ID
MQPUT からの *MsgId* 。

MQRO_NONE
MQPUT 呼び出しからの *MsgId* 。

MQRO_COPY_MSG_ID_TO_CORREL_ID
MQPUT 呼び出しからの *MsgId* 。

これは、これらのオプションを指定しなかった場合のデフォルトです。

MQRO_PASS_CORREL_ID
MQPUT 呼び出しからの *CorrelId* 。

レポート・オプションの詳細については、レポート・オプションおよびメッセージ・フラグを参照してください。

3. 以下の *GetMsgOpts* フィールドを設定します。

Options
MQGMO_WAIT

キュー・マネージャーと同じコード・ページを使用しない場合は、MQGMO_CONVERT を設定し、MQMD に適切な *CodedCharSetId* を設定してください。

WaitInterval
ローカル・キュー・マネージャーからの応答の場合は、 5 秒にしてみてください。 ミリ秒単位でコード化すると、これは 5 000 になります。 リモート・キュー・マネージャーからの応答の場合、 およびチャネル制御コマンドおよび状況コマンドの場合は、 30 秒にしてみてください。 ミリ秒単位でコード化すると、これは 30 000 になります。

破棄されたメッセージ

コマンド・サーバーは、要求メッセージが無効であることを検出すると、このメッセージを破棄し、メッセージ CSQN205I を指定された応答先キューに書き込みます。 応答先キューがない場合は、CSQN205I メッセージは、 送達不能キューに書き込まれます。 このメッセージの戻りコードは、次のように、元の要求メッセージが無効である理由を示します。

00D5020F MQMT_REQUEST のタイプでない。

00D50210 メッセージの長さがゼロである。

00D50212 メッセージが 32 762 バイトより長い。

00D50211 メッセージがすべてブランクである。

00D5483E 変換が必要でしたが、 *Format* が MQFMT_STRING ではありませんでした。

その他 コマンド・サーバー・コードを参照。

コマンド・サーバー応答メッセージ記述子

すべての応答メッセージについて、次の MQMD メッセージ記述子フィールドが設定されます。

MsgType MQMT_REPLY

Feedback MQFB_NONE

Encoding MQENC_NATIVE

Priority 出されたメッセージの MQMD と同じ。

Persistence 出されたメッセージの MQMD と同じ。

CorrelId MQPUT のレポート・オプションによって異なる。

ReplyToQ なし。

コマンド・サーバーは、MQPMO 構造体の *Options* フィールドを MQPMO_NO_SYNCPOINT に設定します。 これは、応答を、次の同期点でまとめて取り出すのではなく、 作成された時点で取り出すことができることを意味します。

z/OS コマンド・サーバーからの応答メッセージの解釈

IBM MQ によって正しく処理された各要求メッセージについて、少なくとも 2 つの応答メッセージが生成されます。 各応答メッセージには、それぞれ 1 つの IBM MQ ユーザー・メッセージが入っています。

1 つの応答の長さは、実行されたコマンドによって異なります。 受け取る可能性のある最も長い応答は DISPLAY NAMELIST からのもので、最大 15 000 バイトの長さになることがあります。

最初のユーザー・メッセージ CSQN205I には、必ず次のものが含まれます。

- 応答のカウント (10 進数)。これは、応答の残りを受け取るループのカウンターとして使用できます。 カウントには、この最初のメッセージも数えられます。
- コマンド前処理プログラムからの戻りコード。
- 理由コード。これは、コマンド・プロセッサーからの理由コードです。

このメッセージには CPF は含まれません。

以下に例を示します。

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
```

COUNT フィールドの長さは 8 バイトで、値は右寄せです。 このフィールドは必ず位置 18、つまり COUNT= の直後から始まります。 RETURN フィールドは、長さが 8 バイトの文字フィールドで、位置 35 つまり RETURN= の直後にきます。 REASON フィールドは、長さが 8 バイトの文字フィールドで、位置 52 つまり REASON= の直後にきます。

RETURN= の値が 00000000 で、REASON= 値が 00000004 の場合は、一連の応答メッセージは未完了です。 CSQN205I メッセージで指示された応答を取り出したあとに、 もう一度 MQGET 呼び出しを実行して次の一連の応答を待ってください。 次の一連の応答の最初のメッセージも CSQN205I です。 このメッセージは、応答の件数を示しているので、他にもまだ応答があるかどうかが分かります。

個別のメッセージについて詳しくは、IBM MQ for z/OS のメッセージ、完了コード、および理由コードの資料を参照してください。

英語以外の言語を使用している場合、応答のテキストおよびレイアウトは、 ここで示したものと異なります。 ただし、メッセージ CSQN205I の中のカウントおよび戻りコードのサイズと位置は同じです。

z/OS 応答が受信されない場合

コマンド・サーバーへの要求に対する応答が受信されない場合、実行可能な一連のステップが存在します。

要求メッセージに対する応答が受信されない場合、次のチェックリストで調べてください。

- コマンド・サーバーが実行されているか。
- *WaitInterval* の時間の長さは十分か。
- システム・コマンド入力キューおよび応答先キューは、正しく定義されているか。
- これらのキューに対する MQOPEN 呼び出しは、正常に終了したか。
- システム・コマンド入力キューと応答先キューの両方が MQPUT 呼び出しと MQGET 呼び出しについて使用可能になっているか。

- キューの MAXDEPTH 属性および MAXMSGL 属性を増やすことを検討したか。
- *CorrelId* フィールドと *MsgId* フィールドを正しく使用しているか。
- キュー・マネージャーは実行を続けているか。
- コマンドを正しく作成したか。
- 使用しているすべてのリモート・リンクは、正しく定義され、 動作しているか。
- MQPUT 呼び出しは、正しく定義されたか。
- 応答先キューは、永続動的キューではなく、一時動的キューとして定義されているか。 (要求メッセージが持続メッセージである場合は、 応答に永続動的キューを使用しなければなりません。)

コマンド・サーバーが応答を生成したが、指定された応答先キューに書き込むことができない場合は、 コマンド・サーバーは、その応答を送達不能キューに書き込みます。

z/OS MGCRE を使用したコマンドの引き渡し

適切な許可を持っている場合、アプリケーション・プログラムは z/OS サービス・ルーチンを使用して、複数のキュー・マネージャーに要求を行うことができます。

適切な権限がある場合は、MGCRE (SVC 34) z/OS サービスを使用して、プログラムから複数のキュー・マネージャーに IBM MQ コマンドを渡すことができます。 CPF の値は、コマンドの宛先となる特定のキュー・マネージャーを 識別します。 CPF について詳しくは、コマンド・セキュリティーとコマンド・リソース・セキュリティーのためのユーザー ID および 427 ページの『z/OS でのキュー・マネージャー・コマンドの発行』を参照してください。

MGCRE を使用した場合は、「コマンドおよび応答のトークン (CART)」 を使用してコマンドへの直接応答を読み取ることができます。

z/OS コマンドおよびその応答の例

このトピックでは、コマンド・サーバーに対するコマンドと、コマンド・サーバーからの応答に関する一連の例を示します。

ここでは、IBM MQ メッセージとして作成できるコマンドおよび応答となるユーザー・メッセージについて、いくつかの例を示します。 特に断りがない限り、応答の各行は、個々のメッセージです。

- DEFINE コマンドからのメッセージ
- DELETE コマンドからのメッセージ
- DISPLAY コマンドからのメッセージ
- CMDSCOPE を伴うコマンドからのメッセージ
- CMDSCOPE を伴うコマンドを生成するコマンドからのメッセージ

DEFINE コマンドからのメッセージ

コマンド

```
DEFINE QLOCAL(Q1)
```

は、次のメッセージを生成します。

```
CSQN205I    COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

これらの応答メッセージは、通常の完了で生成されます。

DELETE コマンドからのメッセージ

コマンド

```
DELETE QLOCAL(Q2)
```

は、次のメッセージを生成します。

```
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
```

これらのメッセージは、Q2 と呼ばれるローカル・キューが存在していないことを示しています。

DISPLAY コマンドからのメッセージ

次の例は、DISPLAY コマンドからの応答を示しています。

送達不能キューの名前の検索

キュー・マネージャーについての送達不能キューの名前を知りたい場合は、次のコマンドをアプリケーション・プログラムから実行します。

```
DISPLAY QMGR DEADQ
```

次の 3 つのユーザー・メッセージが戻され、 これらのメッセージから必要な名前を抽出できます。

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE          )
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

DISPLAY QUEUE コマンドからのメッセージ

次の例は、コマンドからの結果が、そのコマンドに指定した属性値によって どのように異なるかを示しています。

例 1

次のコマンドを使用して、ローカル・キューを定義します。

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

アプリケーション・プログラムから次のコマンドを実行すると、

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

次の 3 つのユーザー・メッセージが戻されます。

```
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
```

注 : 2 番目のメッセージ CSQM401I は、ここでは 4 行で示されています。

例 2

2 つのキューには、次のように、文字 A で始まる名前 が付いています。

- A1 は、その PUT 属性が DISABLED に設定されたローカル・キューです。
- A2 は、その PUT 属性が ENABLED に設定されたリモート・キューです。

アプリケーション・プログラムから次のコマンドを実行すると、

```
DISPLAY QUEUE(A*) PUT
```

次の 4 つのユーザー・メッセージが戻されます。

```
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
```

注 : 2 番目と 3 番目のメッセージである CSQM401I と CSQM406I は、ここでは 3 行と 2 行で示されています。

DISPLAY NAMELIST コマンドからのメッセージ

次のコマンドを使用して、名前リストを定義します。

```
DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)
```

アプリケーション・プログラムから次のコマンドを実行すると、

```
DISPLAY NAMELIST(N1) NAMES NAMCOUNT
```

次の 3 つのユーザー・メッセージが戻されます。

```
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
```

注 : 2 番目のメッセージ CSQM407I は、ここでは 3 行で示されます。

CMDSCOPE を伴うコマンドからのメッセージ

次の例は、CMDSCOPE 属性を指定して入力したコマンドからの応答を示しています。

ALTER PROCESS コマンドからのメッセージ

コマンド

```
ALT PRO(V4) CMDSCOPE(*)
```

は、次のメッセージを生成します。

```
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
```

これらのメッセージは、コマンドがキュー・マネージャー MQ25 上で入力され、 2 つのキュー・マネージャー (MQ25 および MQ26) に送信されたことを通知しています。 コマンドは MQ25 上で正常に実行されましたが、 プロセス定義が MQ26 上には存在しなかったため、 コマンドはキュー・マネージャー MQ25 上で失敗しました。

DISPLAY PROCESS コマンドからのメッセージ

コマンド

```
DIS PRO(V*) CMDSCOPE(*)
```

は、次のメッセージを生成します。

```
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
```

これらのメッセージは、コマンドがキュー・マネージャー MQ25 上で入力され、 2 つのキュー・マネージャー (MQ25 および MQ26) に送信されたことを通知しています。 各キュー・マネージャー上で V の文字で始まる名前をもつ、 すべてのプロセスに関する情報が表示されます。

DISPLAY CHSTATUS コマンドからのメッセージ

コマンド

```
DIS CHS(VT) CMDSCOPE(*)
```

は、次のメッセージを生成します。

```
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
```

これらのメッセージは、コマンドがキュー・マネージャー MQ25 上で入力され、 2 つのキュー・マネージャー (MQ25 および MQ26) に送信されたことを通知しています。 各キュー・マネージャー上のチャネル状況に関する情報が表示されます。

STOP CHANNEL コマンドからのメッセージ

コマンド

```
STOP CHL(VT) CMDSCOPE(*)
```

は、次のメッセージを生成します。

```
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
```

これらのメッセージは、コマンドがキュー・マネージャー MQ25 上で入力され、 2 つのキュー・マネージャー (MQ25 および MQ26) に送信されたことを通知しています。 各キュー・マネージャー上で、チャネル VT が停止されました。

CMDSCOPE を伴うコマンドを生成するコマンドからのメッセージ

コマンド

```
DEF PRO(V2) QSGDISP(GROUP)
```

は、次のメッセージを生成します。

```
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
```

これらのメッセージは、コマンドがキュー・マネージャー MQ25 上で入力されたことを通知しています。 共有リポジトリー上でオブジェクトが作成されると、別のコマンドが生成されて、キュー共有グループ (MQ25 および MQ26) にあるすべてのアクティブ・キュー・マネージャーに送信されます。

z/OS での IBM MQ リソースの管理

このトピックにあるリンクを使用して、IBM MQ for z/OS で使用されるリソースを管理する方法 (例えば、ログ・ファイル、データ・セット、ページ・セット、バッファー・プール、およびカップリング・ファシリティー構造の管理方法) を見つけます。

IBM MQ for z/OS を使用するときに実行しなければならない場合がある、様々な管理用タスクの詳細については、以下のリンクをたどってください。

- 460 ページの『ログの管理』
- 468 ページの『ブートストラップ・データ・セット (BSDS) の管理』
- 476 ページの『ページ・セットの管理』
- 483 ページの『ページ・セットのバックアップおよび回復の方法』
- 487 ページの『CSQUTIL を使用したキューのバックアップおよび回復の方法』
- 487 ページの『バッファー・プールの管理』
- 489 ページの『z/OS でのキュー共有グループと共有キューの管理』

関連概念

IBM MQ for z/OS の概念

415 ページの『IBM MQ for z/OS の管理』
キュー・マネージャーと関連リソースの管理には、それらのリソースをアクティブ化して管理するために頻繁に実行するタスクが含まれます。 キュー・マネージャーと関連リソースを管理するための最適な方法を選択してください。

415 ページの『IBM MQ for z/OS へのコマンドの実行』
キュー・マネージャーを制御するために IBM MQ スクリプト・コマンド (MQSC) をバッチ・モードまたは対話モードで使用することができます。

498 ページの『z/OS での回復と再始動』
このトピックでは、IBM MQ によって使用されるリカバリーおよび再始動のメカニズムについて知ることができます。

関連タスク

z/OS での IBM MQ 環境の計画

z/OS でのキュー・マネージャーの構成

IBM MQ for z/OS ユーティリティーの使用

関連資料

20 ページの『z/OS 上で MQSC コマンドを発行できるソース』
MQSC 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。

424 ページの『IBM MQ for z/OS ユーティリティー』
IBM MQ for z/OS には、システム管理のために使用できる一連のユーティリティー・プログラムが用意されています。

プログラマブル・コマンド・フォーマット・リファレンス

z/OS ログの管理

このトピックでは、ログ・アーカイブ・プロセス、ログ・レコード圧縮の使用、ログ・レコードのリカバリー、およびログ・レコードの印刷を含む、IBM MQ ログ・ファイルの管理方法について知ることができます。

このトピックでは、IBM MQ ログの管理に関係するタスクについて説明します。 この章は、次の節で構成されています。

z/OS *ARCHIVE LOG* コマンドによるログの保存

許可されたオペレーターは、 **ARCHIVE LOG** コマンドを使用して、必要に応じていつでも現行の IBM MQ アクティブ・ログ・データ・セットをアーカイブすることができます。

ARCHIVE LOG コマンドを実行すると、IBM MQ は現在のアクティブ・ログ・データ・セットを切り捨て、非同期のオフロード・プロセスを実行し、さらに BSDS をそのオフロード・プロセスの記録に更新します。

ARCHIVE LOG コマンドには、**MODE(QUIESCE)** オプションがあります。 このオプションを使用すると、IBM MQ のジョブとユーザーはコミット点のあとで休止させられ、その結果としての整合点は、オフロードされる前に現在のアクティブ・ログに取り込まれます。

別の場所での回復を、バックアップの方針として計画している場合は、**MODE(QUIESCE)** オプションを使用することを検討してください。 このオプションは、システム全体に渡る整合点を作成し、それにより、回復時にアーカイブ・ログが最新のバックアップ・ページ・セット・コピーと共に使用されたときに、 不整合のデータの数を最小にすることができます。 以下に例を示します。

```
ARCHIVE LOG MODE(QUIESCE)
```

TIME パラメーターを指定せずに **ARCHIVE LOG** コマンドを実行した場合、デフォルトの休止期間は、CSQ6ARVP マクロの **QUIESCE** パラメーターの値になります。 ARCHIVE LOG MODE(QUIESCE) を完了するために必要な時間が、 指定された時間より少なければ、コマンドは正常に完了します。 そうでない場合には、時間切れになったときにコマンドは異常終了します。 期間は、**TIME** オプションを使用して明示的に指定できます。次に例を示します。

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

このコマンドは、**ARCHIVE LOG** の処理が行われるまでの休止期間が、 最大 60 秒であることを指定します。

注意 : 時間の条件が厳しい場合に **TIME** オプションを使用すると、IBM MQ 資源を使用するすべてのジョブおよびユーザーに対して、IBM MQ の可用性を大きく混乱させることがあります。

デフォルトでは、コマンドの処理は、コマンドを実行依頼した時刻とは 非同期に行われます (このコマンドを他の IBM MQ コマンドと同期的に処理するには、 **QUIESCE** を指定した WAIT (YES) オプションを使用しますが、 z/OS コンソールは **QUIESCE** 期間全体にわたって IBM MQ コマンド入力からロックされることに注意してください。)

休止期間の間は、次のようになります。

- キュー・マネージャー上のジョブおよびユーザーは、コミットの処理を実行することができますが、コミットのあとで何らかの IBM MQ 資源を更新しようとすると、延期させられます。

- データを読み取るだけのジョブおよびユーザーは影響を受けます。 それらのジョブおよびユーザーは、延期されたジョブまたはユーザーによって保持されたロックを待っている可能性があるからです。
- 新規タスクを開始することはできますが、それらのタスクでデータを更新することはできません。

DISPLAY LOG コマンドからの出力では、メッセージ CSQV400I を使用して、 休止が有効であることを示します。

V 9.2.0 例えば、IBM MQ 9.1.4 の場合は以下のようになります。

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     400
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
MAXCNOFF    0
WRTHRSH     20
DEALLCT     0
COMPLOG     NONE
ZHYWRITE    NO
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full zHyperWrite Encrypted DSName
 1     68  NO          NO        VICY.CSQ1.LOGCOPY1.DS01
 2     68  NO          NO        VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2019-08-15 09:49:30 using RBA=000000000891B000
Latest RBA=000000000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
```

例えば、 IBM MQ 9.1.2 より前の Long Term Support および Continuous Delivery の場合は、以下のようになります。

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     400
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
MAXCNOFF    0
WRTHRSH     20
DEALLCT     0
COMPLOG     NONE
ZHYWRITE    NO                     YES
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full PPRC DSName
 1     68  NO   VICY.CSQ1.LOGCOPY1.DS01
 2     68  NO   VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2014-04-15 09:49:30 using RBA=000000000891B000
Latest RBA=000000000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
```

すべての更新が休止させられると、BSDS の中の休止履歴レコードは、 アクティブ・ログ・データ・セットが切り捨てられた日付と時刻に更新され、 また現在のアクティブ・ログ・データ・セット内の最後に書き込まれた RBA に更新されます。 IBM MQ は現在のアクティブ・ログ・データ・セットを切り捨て、次に

利用可能なアクティブ・ログ・データ・セットへ切り替え、オフロードが開始したことを知らせるメッセージ CSQJ311I を出します。

静止期間が満了する前に更新を静止できない場合、IBM MQ はメッセージ CSQJ317I を発行し、**ARCHIVE LOG** 処理は終了します。 この場合、現在のアクティブ・ログ・データ・セットは切り捨てられず、次に使用可能なログ・データ・セットへは切り替えられません。またオフロードは開始しません。

次に、休止が正常に行われたかどうかにかかわらず、延期させられていたすべてのユーザーおよびジョブが再開され、IBM MQ は、休止が終了し更新活動が再開したことを示すメッセージ CSQJ312I を出します。

現在のアクティブ・ログが最後に使用可能なアクティブ・ログ・データ・セットである場合に **ARCHIVE LOG** が実行されると、このコマンドは処理されず、IBM MQ は、次のメッセージを出します。

```
CSQJ319I - csect-name CURRENT ACTIVE LOG DATA SET IS THE LAST
使用可能なアクティブ・ログ・データ・セット。 アーカイブ・ログ処理
WILL BE TERMINATED
```

別の **ARCHIVE LOG** コマンドが既に進行中であるときに **ARCHIVE LOG** が実行されると、新しいコマンドは処理されず、IBM MQ は、次のメッセージを出します。

```
CSQJ318I - ARCHIVE LOG COMMAND ALREADY IN PROGRESS
```

アーカイブ中に出されるメッセージについて詳しくは、IBM MQ for z/OS のメッセージを参照してください。

失敗した後のログ保存プロセスの再始動

ログ保存プロセス中に問題がある場合 (例えば、割り振りまたは磁気テープ装着の問題)、 アクティブ・ログの保存は延期されることがあります。 以下のコマンドを使用して、保存プロセスを取り消して再始動できます。

```
ARCHIVE LOG CANCEL OFFLOAD
```

このコマンドは現在進行中のすべてのオフロード・プロセスを取り消し、 保存プロセスを再始動します。 このプロセスは、保存されなかった最も古いログ・データ・セットで始まり、 オフロードを必要とするすべてのアクティブ・ログ・データ・セットを処理します。 延期されていたログ保存操作はすべて再始動されます。

現在使用しているログ保存タスクが機能しなくなっていることが確かである場合、 または直前の試行が失敗して再始動したい場合にのみ、このコマンドを使用します。 それは、このコマンドによってオフロード・タスクが異常終了し、 ダンプしてしまう可能性があるからです。

z/OS 保存とロギングの制御

CSQ6LOGP、CSQ6ARVP、および CSQ6SYSP マクロを使用して、圧縮、印刷、保存、リカバリー、およびロギングを制御することができます。 専用オブジェクトに対する変更のみが IBM MQ ログに記録されることに注意してください。 定義はグループ全体に伝搬され、ローカルに保持されるため、GROUP オブジェクトに対する変更 (共有インバウンド・チャネルなど) もログに記録されます。

保存およびロギングの多くの局面は、キュー・マネージャーをカスタマイズする際に、 システム・パラメーター・モジュールの CSQ6LOGP、CSQ6ARVP および CSQ6SYSP マクロを使用して設定されたパラメーターによって制御されます。 これらのマクロについて詳しくは、システム・パラメーター・モジュールの調整を参照してください。

これらのパラメーターの一部は、キュー・マネージャーの実行中に IBM MQ MQSC SET LOG、SET SYSTEM、および SET ARCHIVE コマンドを使用して変更できます。 これを 462 ページの表 28 で示します。

表 *28.* キュー・マネージャーの実行中に変更される可能性のある保存およびロギング・パラメーター

SET コマンド	**Parameters**
LOG	WRTHRSH、MAXARCH、DEALLCT、MAXRTU、COMPLOG
アーカイブ	すべて

表 28. キュー・マネージャーの実行中に変更される可能性のある保存およびロギング・パラメーター (続き)

SET コマンド	Parameters
SYSTEM	LOGLOAD

MQSC DISPLAY LOG、DISPLAY ARCHIVE および DISPLAY SYSTEM コマンドを使用して、すべてのパラメーターの設定値を表示することができます。 また、これらのコマンドは、保存およびロギングについての状況情報も表示します。

ログ圧縮の制御

ログ・レコードの圧縮は、以下を使用して使用可能または使用不可にすることができます。

- MQSC の SET コマンドおよび DISPLAY LOG コマンド。MQSC コマンドを参照してください。
- PCF インターフェースの起動。 24 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
- システム・パラメーター・モジュールの CSQ6LOGP マクロの使用。CSQ6LOGP の使用 を参照してください。

ログ・レコードの印刷

ログ・レコードの取り出しと印刷には、CSQ1LOGP ユーティリティーを使用します。 この説明については、ログ印刷ユーティリティーを参照してください。

ログの回復

通常、IBM MQ ログはバックアップおよび復元の必要はありません。特に、重複ロギングを使用している場合は、バックアップおよび復元は必要ありません。 しかし、ログ上の入出力エラーなどごくまれな状況で、ログの回復が必要な場合があります。 アクセス方式サービスを使用して、そのデータ・セットを削除して再定義し、 その後対応する重複ログをデータ・セットにコピーします。

z/OS アーカイブ・ログ・データ・セットの廃棄

アーカイブ・ログ・データ・セットを廃棄することができ、ログを自動または手動で廃棄することを選択できます。

作業単位の回復、ページ・セットが失われた場合はページ・セット・メディアの回復、 CF 構造体が失われた場合は CF 構造体メディアの回復を実行できるように、十分なログ・データを保持しておく必要があります。 回復に必要となる可能性のあるアーカイブ・ログ・データ・セットを廃棄しないでください。そのようなアーカイブ・ログ・データ・セットを廃棄してしまうと、必要な回復操作を実行することができなくなります。

アーカイブ・ログ・データ・セットを廃棄しても構わないことを確認したときは、 次のいずれかの方法で廃棄することができます。

- アーカイブ・ログ・データ・セットの自動削除
- 手操作によるアーカイブ・ログ・データ・セットの削除

アーカイブ・ログ・データ・セットの自動削除

アーカイブ・ログ・データ・セットを自動的に削除するために、 DASD またはテープの管理システムを使用することができます。 IBM MQ アーカイブ・ログ・データ・セットの保存期間は、CSQ6ARVP インストール・マクロの保存期間フィールド ARCRETN で指定します (詳細については、CSQ6ARVP の使用を参照)。

保存期間のデフォルトは、アーカイブ・ログを 9999 日間 (最大値) 保存することを指定するものです。

重要 : 保存期間を変更することはできますが、計画したバックアップ・サイクルの数に対応できることを確認する必要があります。

.

IBM MQ は、アーカイブ・ログ・データ・セットが作成されるとき、保存期間の値を JCL パラメーター RETPD の値として使用します。

MVS™/DFP ストレージ管理サブシステム (SMS) によって設定された保存期間は、この IBM MQ パラメーターによってオーバーライドすることができます。 通常、保存期間は、IBM MQ または SMS のいずれかが指定した値のうち、小さい方に設定されます。 ストレージ管理担当者と IBM MQ 管理担当者は、IBM MQ にとって適切な保存期間の値について、お互いに合意している必要があります。

注 : 外部からの手操作による、保存期間の指定変更の機能を提供しているテープ管理システムもあるため、IBM MQ では、アーカイブ・ログ・データ・セットについての情報を、自動的に BSDS から削除する方法はありません。 このため、データ・セットの保存期間が切れ、データ・セットがテープ管理システムによってスクラッチされたあとでも、長い間その保存ログ・データ・セットについての情報が BSDS に入っていることがあります。 また反対に、 データ・セットが期限切れに達する前に、アーカイブ・ログ・データ・セットの最大数を超えたため、データが BSDS から除去されることもあります。

保存ログ・データ・セットが自動的に削除されても、その操作では、 BSDS の中の保存ログのリストは更新されないことを忘れないでください。 BSDS は、470 ページの『BSDS の変更』で説明したログ目録変更ユーティリティーを使用して更新することができます。 この更新は、必須のものではありません。 古いアーカイブ・ログを記録しておくと、BSDS のスペースは無駄になりますが、その他の害はありません。

手操作による保存ログ・データ・セットの削除

メッセージ CSQI024I および CSQI025I に示されている最下位 RBA ID のログ・レコードまでのすべてのログ・レコードを保持する必要があります。 この RBA は、方法 1: フルバックアップを使用して回復点を作成するときに実行した DISPLAY USAGE コマンドを使用して得られます。

ログを廃棄する前に、非共有資源の回復点の作成を参照してください。

アーカイブ・ログ・データ・セットの検索および廃棄

回復するために必要な最小ログ RBA を設定したなら、次の手順を実行することにより、 以前のログ・レコードだけが入っている保存ログ・データ・セットを検索することができます。

1. ログ・マップ印刷ユーティリティーを使用して、BSDS の内容を印刷します。 出力の例については、ログ・マップ印刷ユーティリティーを参照してください。
2. `ARCHIVE LOG COPY n DATA SETS` という見出しの出力の部分を探してください。 重複ロギングを使用している場合は、2 つの部分があります。 STARTRBA および ENDRBA という見出しの欄は、各ボリュームに含まれている RBA の範囲を示しています。 メッセージ CSQI024I と CSQI025I に示されている 最小 RBA が範囲に含まれているボリュームを見つけてください。 これらが、保持が必要な最も古いボリュームです。 重複ロギングを使用している場合、このようなボリュームは 2 つあります。

   該当する範囲のボリュームがない場合は、次のケースのいずれかです。

   - 最小 RBA がまだ保存されていない場合。 すべてのアーカイブ・ログ・ボリュームを廃棄することができます。
   - ボリュームの数が、CSQ6LOGP マクロの MAXARCH パラメーターで許された数を超えたため、BSDS の中の保存ログ・ボリュームのリストが折り返してしまった場合。 BSDS に保存ログ・ボリュームが登録されていなければ、そのボリュームは回復には使用できません。 したがって、BSDS に、既存ボリュームの情報を追加することを考慮してください。 詳しくは、472 ページの『アーカイブ・ログに対する変更』を参照してください。

     また、MAXARCH の値を増やすことも考える必要があります。 詳細については、CSQ6LOGP の使用を参照してください。

3. 保存しておきたい最も古いボリュームの STARTRBA の値よりも小さい値の ENDRBA を持つすべての保存ログ・データ・セットまたはボリュームを削除します。 重複ロギングを使用している場合は、このようなコピーの両方を削除してください。

   BSDS のエントリーは循環するので、BSDS アーカイブ・ログ・セクションの先頭にあるいくつかのエントリーが、最後の部分にあるエントリーより新しい場合があります。 日付と時刻の両方を見て、それらの項目の経過時間を比較してください。 最小の LOGRBA が含まれるアーカイブ・ログの BSDS 項目より前にあるすべての項目を廃棄できるとは限りません。

   データ・セットを削除してください。 アーカイブ・ログがテープ上にある場合は、テープを消去します。 アーカイブ・ログが DASD 上にある場合は、z/OS ユーティリティーを実行して、各データ・セットを削除します。 次に、BSDS に既存の保存ボリュームだけを表示したい場合は、 ログ目録変更ユーティリティー (CSQJU003) を使用して、廃棄されたボリュームの項目を削除してください。 例については、472 ページの『アーカイブ・ログに対する変更』を参照してください。

z/OS ログ延期の影響

長期実行トランザクションによって、ログ・データ・セットにまたがって作業単位のログ・レコードが生成される可能性があります。 IBM MQ は、ログ延期、つまりログ・レコードを移動して保存されるログ・データの量を最適化する技法と、キュー・マネージャーの再始動時を使用して、このシナリオを処理します。

作業単位が長くなると考えられるとき、各ログ・レコードの表示は ログの後の方に書き込まれます。 これはログ延期 として知られています。 これについては、 ログ・ファイルで詳しく説明します。

キュー・マネージャーは、障害の後にオリジナルではなく延期されたログ・レコード を使用して、作業単位の整合性を確認します。 これには 2 つの利点があります。

- 作業単位の調整用に保存する必要のあるログ・データの量が 削減される。
- キュー・マネージャーの再始動時に、より少ないログ・データが検索されるため、 キュー・マネージャーの再始動が速くなる。

延期されたログ・レコードには、メディア回復操作についての 十分な情報は含まれていません。

ログに保持されるデータは、メディア回復と 作業単位の調整という 2 つの異なる目的のために使用されます。 CF 構造またはページ・セットのいずれかに影響するメディア障害が発生した場合に、キュー・マネージャーは、 前のコピーを復元し、ログに含まれるデータを使用してこれを更新することにより、 メディアを障害の時点まで回復することができます。 作業単位で実行される持続活動 はログに記録されるため、障害が発生した場合には、持続活動がバックアウトされるか、 または変更された資源でロックが回復されます。 キュー・マネージャー回復を使用可能にするために保持すべきログ・データの量は、これら 2 つの要素により影響を受けます。

メディア回復の場合、少なくても最新のメディア・コピーから メディア回復を実行することができ、バックアウトできるだけの 十分なログ・データを保持する必要があります。 (ご使用のサイトは、古いバックアップから回復できることを規定していることがあります。) 作業単位の整合性については、最も古い未完了または未確定の作業単位に対して ログ・データを保持する必要があります。

システムの管理を支援するため、キュー・マネージャーはログ保存ごとに 古い作業単位を検出し、それをメッセージ CSQJ160 と CSQJ161 に報告します。 内部タスクがこの古い作業単位についての作業単位ログ情報を読み取り、 ログ内の現在の位置に、より簡潔な形式で書き換えます。 これが行われた日時が、メッセージ CSQR026 に表示されます。 MQSC コマンド DISPLAY USAGE TYPE(DATASET) も、ログ・データの保持の管理に役立ちます。 このコマンドは、以下の 3 つのリカバリー情報を報告します。

1. 作業単位回復用に保持する必要のあるログの量。
2. ページ・セットのメディア回復用に保持する必要のあるログの量。
3. キュー共有グループ内のキュー・マネージャーの場合、CF 構造のメディア・リカバリー用に保持する必要のあるログの量。

上記のそれぞれの情報について、必要な最も古いログ・データをデータ・セットにマップする試みが行われます。 新しい作業単位が開始して停止すると、(1) がログの新しい位置に移動することが予想されます。 移動していない場合、長期実行中の UOW メッセージは、問題があることを警告します。 (2) は、 キュー・マネージャーが即時にシャットダウンおよび再始動された場合に、 ページ・セットのメディア回復に関連します。 キュー・マネージャーは、ページ・セットを最後にバックアップした 日時や、ページ・セットに

障害があった場合に使用しなければならないバックアップについては認識しません。 これは、通常、バッファー・プールに保持された変更内容がページ・セットに書き込まれるとき、チェックポイントの処理中にログ内のより新しい位置まで移動します。 (3) で、キュー・マネージャーは、このキュー・マネージャーまたはキュー共有グループの 他のキュー・マネージャーで行われる CF 構造バックアップについて認識しています。 ただし、CF 構造回復は、最後のバックアップ以降に CF 構造と相互作用している キュー共有グループのすべてのキュー・マネージャーから、ログ・データのマージ を必要とします。 つまり、ログ・データはログ・レコード・シーケンス番号 (または LRSN) によって 識別されます。 これはタイム・スタンプを基にしており、キュー共有グループ内の別のキュー・マネージャー上で異なる RBA よりむしろ、キュー共有グループ全体で適用できます。 これは、通常、BACKUP CFSTRUCT コマンドがこのキュー・マネージャーまたはキュー共有グループの他のキュー・マネージャーで実行されるとき、ログのより新しい位置まで移動します。

z/OS キュー・マネージャーのログのリセット

このトピックを使用して、キュー・マネージャーのログをリセットする方法を理解してください。

キュー・マネージャーのログ RBA に対して、ログ RBA 範囲の末尾から 0 に折り返すことを許可してはなりません。折り返しが起こると、キュー・マネージャーが停止し、すべての持続データがリカバリー不能になるからです。 ログ RBA の末尾は、値 FFFFFFFFFFFF (6 バイトの RBA が使用されている場合) か FFFFFFFFFFFFFFFF (8 バイトの RBA が使用されている場合) のどちらかです。

キュー・マネージャーは、使用されているログ範囲がきわめて大きくなっていること、および計画外停止を回避する処置の計画をする必要があることを示すために、メッセージ CSQI045I、CSQI046E、CSQI047E、 CSQJ031D、および CSQJ032E を発行します。

RBA 値が FFF800000000 (6 バイトのログ RBA が使用されている場合) または FFFFFFC000000000 (8 バイトのログ RBA が使用されている場合) に達すると、キュー・マネージャーは理由コード 00D10257 で終了します。

6 バイトのログ RBA が使用されている場合、キュー・マネージャーのログをリセットする代わりに、467 ページの『より大きなログ相対バイト・アドレスの実装』で説明されているプロセスに従って、キュー・マネージャーを変換して 8 バイトのログ RBA を使用することを検討してください。 8 バイトのログ RBA を使用するようキュー・マネージャーを変換するときには停止が必要ですが、その所要時間はログをリセットするより短く、ログのリセットが必要になるまでの期間が長くなります。

キュー・マネージャーの初期設定時に発行されるメッセージ CSQJ034I は、構成されたキュー・マネージャーのログ RBA 範囲の末尾を示し、使用されるログ RBA が 6 バイトか 8 バイトかを判別するために使用できます。

キュー・マネージャーのログのリセットを行うための手順は、以下のとおりです。

1. 未解決の作業単位を解決します。 未解決の作業単位の数は、キュー・マネージャーの始動時に、メッセージ CSQR005I で INDOUBT カウントとして表示されます。 各チェックポイントで、およびキュー・マネージャーのシャットダウン時に、キュー・マネージャーが自動的に次のコマンドを実行します。

   DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED)。このコマンドは、未解決の作業単位に関する情報を提供します。

   リカバリー単位の解決については、 未確定のリカバリー単位が解決される方法 を参照してください。 最終手段は、**RESOLVE INDOUBT MQSC** MQSC コマンドを使用して、未確定のリカバリー単位を手動で解決することです。

2. キュー・マネージャーをクリーン・シャットダウンします。

   STOP QMGR または **STOP QMGR MODE(FORCE)** のいずれかを使用できます。これらのコマンドはどちらも、変更されたページをバッファー・プールからページ・セットにフラッシュします。

3. キュー・マネージャーがキュー共有グループの一部である場合は、キュー共有グループのすべての構造に対して、他のキュー・マネージャーの CFSTRUCT バックアップを取ります。 これにより、最近のバックアップはこのキュー・マネージャーのログには含まれず、このキュー・マネージャーのログは CFSTRUCT リカバリーには不要になることが保証されます。

4. CSQJU003 を使用して新規のログと BSDS を定義します (ログ目録変更ユーティリティーの使用方法について詳しくは、ログ目録変更ユーティリティーを参照してください)。

5. このキュー・マネージャーのすべてのページ・セットに対して **CSQUTIL RESETPAGE** を実行します (この関数の使用方法について詳しくは、ページ・セットのコピーとログのリセットを参照してください)。ページ・セット RBA は個別にリセットできることに注目してください。この機能により、このステップでの経過時間を短くするために複数の並行ジョブ (例えばページ・セット当たり 1 つ) を実行依頼できます。
6. キュー・マネージャーを再始動します。

関連概念

467 ページの『より大きなログ相対バイト・アドレスの実装』
IBM MQ for z/OS 8.0 より前のバージョンでは、IBM MQ for z/OS は 6 バイトのログ RBA を使用してログ内のデータの場所を識別していました。 IBM MQ for z/OS 8.0 以降では、8 バイトのログ RBA を使用できます。これを使用すると、ログのリセットが必要になるまでの期間が長くなります。

z/OS より大きなログ相対バイト・アドレスの実装

IBM MQ for z/OS 8.0 より前のバージョンでは、IBM MQ for z/OS は 6 バイトのログ RBA を使用してログ内のデータの場所を識別していました。 IBM MQ for z/OS 8.0 以降では、8 バイトのログ RBA を使用できます。これを使用すると、ログのリセットが必要になるまでの期間が長くなります。

IBM MQ 9.2.0 Long Term サポートのキュー・マネージャー、および IBM MQ 9.2.4 の前に作成される Continuous Delivery キュー・マネージャーの場合、この機能を明示的に使用可能にする必要があります。

V 9.2.5 IBM MQ 9.2.5 以降で作成されたキュー・マネージャーの場合、この機能は既に使用可能になっています。

8 バイトのログ RBA を有効にする場合の考慮事項については、アドレス指定可能な最大ログ範囲を広げる計画を参照してください。

重要: V 9.2.0 キュー・マネージャーがキュー共有グループの一部ではなく、その後、8 バイトのログ RBA を有効にして IBM MQ for z/OS 9.0.0 にマイグレーションし直す場合は、それらのリリースで **OPMODE**=*NEWFUNC,900* を使用していることを確認してください。そうしないと、キュー・マネージャーの開始に失敗します。

単一の IBM MQ for z/OS キュー・マネージャーで 8 バイトのログ RBA を有効にするには、以下の指示をこの順序で実行してください。

1. V 9.2.0 キュー・マネージャーがキュー共有グループ内にある場合は、ステップ 467 ページの『2』に進む前に、IBM MQ for z/OS 9.0.0 にあるキュー共有グループ内のすべてのキュー・マネージャーが **OPMODE**= (*NEWFUNC,900*) で実行されていることを確認してください。

   これを行うためにキュー共有グループ全体を停止する必要はありません。 IBM MQ for z/OS 9.0.0 にある各キュー・マネージャーを順番に停止し、**OPMODE**= (*NEWFUNC,900*) に変更して、再始動することができます。

   キュー共有グループ内のすべての IBM MQ for z/OS 9.0.0 キュー・マネージャーを **OPMODE**= (*NEWFUNC,900*) で実行した後、すべてのキュー・マネージャーが新しい BSDS で実行されるまで、キュー共有グループ内の各キュー・マネージャーに対して以下のステップを実行します。
2. 現行 BSDS に類似した属性を新規 BSDS データ・セットに割り振ります。 CSQ4BSDS のサンプルを作成し、不要なステートメントを削除するか、既存の JCL を使用し、BSDS 名を.BSDS のように変更することが可能です。 `++HLQ++.NEW.BSDS01`.

   注:

   a. 新規 BSDS を割り振るジョブを実行依頼する前に、新規 BSDS の属性を確認します。 変更されている属性は、BSDS のサイズだけかもしれません。

   b. 新規 BSDS には現行 BSDS より多くのデータが格納されるため、新規データ・セットに十分な使用可能スペースが割り振られていることを確認する必要があります。 thlqual.SCSQPROC(CSQ4BSDS) 内のサンプル JCL には、新しい BSDS を定義する際の推奨値が含まれています。
3. キュー・マネージャーをクリーン・シャットダウンします。

4. BSDS 変換ユーティリティー (CSQJUCNV) を実行して、既存の BSDS を新規 BSDS データ・セットに変換します。 この実行には通常、数秒かかります。

   既存の BSDS はこのプロセス中に変更されないので、変換が失敗した場合には、キュー・マネージャーの初期設定にその BSDS を使用できます。

5. 現行 BSDS を名前変更して古い BSDS として扱い、新規 BSDS を名前変更して現行 BSDS とします。こうすると、次回キュー・マネージャーを再始動したときに新規データ・セットが使用されます。 DFSMS アクセス方式サービスの ALTER コマンドを使用できます。例えば、次のようにします。

   ```
   ALTER '++HLQ++.BSDS01' NEWNAME('++HLQ++.OLD.BSDS01')
   ALTER '++HLQ++.NEW.BSDS01' NEWNAME('++HLQ++.BSDS01')
   ```

   さらに、VSAM クラスターのデータと索引の両方の部分を名前変更するコマンドも必ず発行してください。

6. キュー・マネージャーを再始動する。 この始動には、6 バイトのログ RBA を使用して行う場合と同じ時間がかかります。

   変換した BSDS へのアクセスに失敗してキュー・マネージャーが正常に再始動しない場合、その失敗の原因を特定し、問題を解決して操作を再試行してください。 支援が必要な場合、IBM サポートに連絡してください。

   必要に応じて、以下を行うことによってこの時点で変更をバックアウトすることができます。

   a. 現行 BSDS を名前変更して新規 BSDS として扱う。
   b. 古い BSDS を名前変更して現行 BSDS として扱う。
   c. キュー・マネージャーを再始動する。

   変換した BSDS を使ってキュー・マネージャーを正常に再始動した後は、 古い BSDS を使用してキュー・マネージャーを始動しようとしないでください。

7. キュー・マネージャーの初期設定時に発行されるメッセージ CSQJ034I は、キュー・マネージャーのログ RBA の末尾が構成されたことを示します。 ログ RBA 範囲の末尾が FFFFFFFFFFFFFFFF と表示されることを確認します。 これは、8 バイトのログ RBA が使用されていることを示します。

注 : 新しいキュー・マネージャーで 8 バイトのログ RBA を有効にするには、最初に始動する前に、まずバージョン 1 形式の空の BSDS を作成し、BSDS 変換ユーティリティーの入力としてそれを使用し、バージョン 2 形式の BSDS を生成する必要があります。 このプロセスを実行する方法について詳しくは、ブートストラップとログ・データ・セットを作成するを参照してください。

関連概念

より大きなログ相対バイト・アドレス

関連タスク

アドレス指定可能な最大ログ範囲を広げる計画

関連資料

BSDS 変換ユーティリティー (CSQJUCNV)

z/OS ブートストラップ・データ・セット (BSDS) の管理

ブートストラップ・データ・セット (BSDS) を使用して、ログ・データ・セット、およびログ・レコードを参照することができます。 このトピックでは、BSDS の調査、変更、およびリカバリーを行う方法について知ることができます。

詳しくは、 ブートストラップ・データ・セットを参照してください。

このトピックでは、ブートストラップ・データ・セットの管理に関連したタスクについて説明します。 この章は、次の節で構成されています。

- 469 ページの『BSDS に含まれる内容の検出』
- 470 ページの『BSDS の変更』

- 474 ページの『BSDS の回復』

z/OS *BSDS* に含まれる内容の検出

ログ・マップ印刷ユーティリティー (CSQJU004) を使用して、BSDS の内容を調べます。

ログ・マップ印刷ユーティリティー (CSQJU004) は、 BSDS に保管されている情報を表示するバッチ・ユーティリティーです。 この実行方法については、ログ・マップ印刷ユーティリティーを参照してください。

BSDS には、次のものが含まれています。

- タイム・スタンプ
- アクティブ・ログ・データ・セットの状況

BSDS の中のタイム・スタンプ

ログ・マップ印刷ユーティリティーの出力には、タイム・スタンプが表示され ます。タイム・スタンプは、さまざまなシステム・イベントの日付と時刻を記録するために使用さ れるもので、BSDS の中に保管されています。

次のタイム・スタンプが、レポートの見出しの部分に含まれています。

SYSTEM TIMESTAMP
BSDS が最後に更新された日付と時刻を示しています。 BSDS タイム・スタンプは、次の場合に更新される可能性があります。

- キュー・マネージャーが開始します。
- ログの書き込みの活動中に、書き込み限界値に達した場合。 指定した出力バッファー数およびシステム活動の速度によっては、BSDS が 1 秒間に数回更新されることも、数秒間、数分間、あるいは数時間も更新されない場合もあります。 書き込み限界値の詳細については、CSQ6LOGP の使用の CSQ6LOGP マクロの WRTHRSH パラメーターを参照してください。
- エラーのために、IBM MQ が通常の重複 BSDS モードから単一 BSDS モードになってしまった場合。 これは、BSDS レコードの入手、挿入、指示、更新、削除など の要求が、正常に行われなかった場合に起こります。 このエラーが起こった場合、IBM MQ は、残りの BSDS のタイム・スタンプを更新して、使用不可になった BSDS に関して強制的にタイム・スタンプが一致しないようにします。

UTILITY TIMESTAMP
BSDS の内容が、ログ目録変更ユーティリティー (CSQJU003) によって変更された日付と時刻。

次のタイム・スタンプは、レポートのアクティブ・ログ・データ・セットおよび アーカイブ・ログ・データ・セットの部分に含まれます。

アクティブ・ログ日付
アクティブ・ログ項目が BSDS に作成された (つまり、CSQJU003 NEWLOG が行われた) 日付。

アクティブ・ログ時刻
アクティブ・ログ項目が BSDS に作成された (つまり、CSQJU003 NEWLOG が行われた) 時刻。

アーカイブ・ログ日付
アーカイブ・ログ項目が BSDS に作成された (つまり、CSQJU003 NEWLOG が 行われたか、保存自体が行われた) 日付。

アーカイブ・ログ時刻
アーカイブ・ログ項目が BSDS に作成された (つまり、CSQJU003 NEWLOG が 行われたか、保存自体が行われた) 時刻。

アクティブ・ログ・データ・セットの状況

BSDS は、アクティブ・ログ・データ・セットの状況を、次のうちの 1 つとして記録します。

NEW
データ・セットは定義されているが、IBM MQ によってまったく使用されていない場合、またはデータ・セットが最初に使用される前の個所まで、ログが切り捨てられた場合。 いずれの場合も、そのデータ・セットの開始および終了の RBA の値は、ゼロに リセットされます。

再使用可能
データ・セットは定義されているが、IBM MQ によってまったく使用されていないか、またはデータ・セットがオフロードされている場合。 ログ・マップ印刷出力では、最後の REUSABLE データ・セットの開始 RBA 値は、最後のアーカイブ・ログ・データ・セットの開始 RBA 値と等しくなります。

再使用不可能
データ・セットにオフロードされていないレコードが含まれている場合。

STOPPED
レコードの読み取り中にオフロード・プロセッサーでエラーとなり、そのレコードをアクティブ・ログの他のコピーから入手することができない場合。

TRUNCATED
次のいずれかの場合:

- 入出力エラーが発生し、このデータ・セットへの書き込みを IBM MQ が停止している場合。 アクティブ・ログ・データ・セットは、開始 RBA から、切り捨てられたアクティブ・ログ・データ・セットの最後の有効なレコード・セグメントまでが、オフロードされます。 最後の有効なレコード・セグメントの RBA は、アクティブ・ログ・データ・セットの終了の RBA より小さい値になります。 ロギングは、次に使用可能なアクティブ・ログ・データ・セットに切り替えられ、中断せずに続行されます。

または

- ARCHIVE LOG 機能が呼び出された場合で、アクティブ・ログが切り捨てられます。

状況は、ログ・マップ印刷ユーティリティーからの出力に表示されます。

z/OS *BSDS の変更*

BSDS をロギング・イベントのレコードで更新し続けるための特別な手順は、IBM MQ が自動的に行うため、必要ありません。

ただし、次のいずれかを行う場合には、BSDS を変更できます。

- アクティブ・ログ・データ・セットをさらに追加する場合。
- アクティブ・ログ・データ・セットを新しく割り振られたデータ・セットにコピーする場合。例えば、アクティブ・ログの割り振りを大きくするときなど。
- ログ・データ・セットを他の装置に移動する場合。
- 損傷した BSDS を回復する場合。
- 古くなったアーカイブ・ログ・データ・セットを廃棄する場合。

ログ目録変更ユーティリティー (CSQJU003) を実行することにより、 BSDS を変更することができます。 このユーティリティーは、キュー・マネージャーが非活動状態の場合、 または結果が不整合になる可能性のある場合のみに実行します。 このユーティリティーのアクションは、 SYSIN データ・セットの中のステートメントによって制御されます。 この節では、いくつかの例を示します。 詳細な説明については、ログ目録変更ユーティリティーを参照してください。

アクティブ・ログ・データ・セットは、キュー・マネージャーの始動時に IBM MQ が専用 (DISP=OLD) として割り振るため、キュー・マネージャーが非アクティブの場合にのみコピーすることができます。

z/OS アクティブ・ログに対する変更

このトピックでは、BSDS を使用してアクティブ・ログを変更する方法について知ることができます。

ログ変更ユーティリティーを使用して、 アクティブ・ログについて、BSDS の中の項目に追加、削除、および記録を行うことができます。 ここでは、例のみ示します。例の中のデータ・セット名を、 使用したいデータ・セット名に置き換えてください。 このユーティリティーの詳細については、ログ目録変更ユーティリティーを参照してください。

詳しくは、以下のセクションを参照してください。

- BSDS へのレコード項目の追加
- BSDS からのアクティブ・ログ・データ・セットについての情報の削除
- BSDS へのログ・データ・セットについての情報の記録
- アクティブ・ログのサイズの増加
- CSQJUFMT の使用

BSDS へのレコード項目の追加

アクティブ・ログに「stopped」としてフラグが付けられている場合、 そのログはロギングには再利用されません。ただし、読み取りには引き続き使用されます。 新しいアクティブ・ログ・データ・セットを定義するにはアクセス方式サービスを使用し、次に、新しいデータ・セットを BSDS に登録するためには ログ目録変更ユーティリティーを使用します。 例えば、次を使用します。

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

古いアクティブ・ログ・データ・セットの内容を新しいアクティブ・ログ・データ・セットにコピーする場合、NEWLOG 機能で、RBA の範囲、および開始と終了のタイム・スタンプを指定することもできます。

BSDS からのアクティブ・ログ・データ・セットについての情報の削除

アクティブ・ログ・データ・セットについての情報を BSDS から削除するには、 次を使用します。

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

BSDS へのログ・データ・セットについての情報の記録

既存の活動ログ・データ・セットについての情報を BSDS に記録する には、次を使用します。

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

この種の情報を含むレコードを BSDS に挿入する理由は、次のとおりです。

- データ・セットについての項目が削除されたが、再び必要になったため
- ある活動ログ・データ・セットの内容を他のデータ・セットにコピーするため
- BSDS をバックアップ・コピーから復元するため

アクティブ・ログのサイズの増加

この処理を実行するには、次の 2 つの方法があります。

1. キュー・マネージャーがアクティブである場合:
   a. JCL を使用して、新しいより大きなログ・データ・セットを定義します。
   b. MQSC DEFINE LOG コマンドを使用して、新しいログ・データ・セットをアクティブなキュー・マネージャーに追加します。
   c. MQSC ARCHIVE LOG コマンドを使用して、現在のアクティブ・ログを移動して、新しいより大きなログにします。
   d. より小さいアクティブ・ログ・データ・セットのアーカイブが完了するのを待ちます。

e. CSQJU003 ユーティリティーを使用して古い小さいアクティブ・ログを除去し、キュー・マネージャーをシャットダウンします。

f. キュー・マネージャーを再始動する。

2. キュー・マネージャーが非アクティブである場合:

a. キュー・マネージャーを停止させます。 これがアクティブであるときには、すべてのアクティブ・ログ・データ・セットがそれ専用として IBM MQ によって割り振られているため、このステップが必要です。

b. アクセス方式サービス ALTER を NEWNAME オプションと共に 使用して、アクティブ・ログ・データ・セットの名前を変更します。

c. アクセス方式サービス DEFINE を使用して、より大きな アクティブ・ログ・データ・セットを定義します。

古いデータ・セット名を再利用すると、ログ目録変更ユーティリティーを実行 して BSDS 内に新しい名前を確立する必要がなくなります。 古いデータ・セット名と正しい RBA の範囲は、すでに BSDS に入っています。

d. アクセス方式サービス REPRO を使用して、古い (名前変更された) データ・セットを、それぞれの適切な新しいデータ・セット内にコピーします。

注 : このステップを実行するには時間がかかるため、その間は全社的に活動ができなくなる可能性があります。

e. キュー・マネージャーを始動します。

すべてのログ・データ・セットが同じサイズだと、 システムの操作上、一貫性と効果性がより高くなります。 ログ・データ・セットが同じサイズでないと、 システムのログをトラックすることが難しくなり、スペースが無駄になることがあります。

CSQJUFMT の使用

アクティブ・ログのサイズを大きくする際には、CSQJUFMT フォーマットを実行しないでください。

(キュー・マネージャーによって新規アクティブ・ログに初めて書き込まれるときにパフォーマンス上のメリットを得るために) CSQJUFMT を実行すると、以下のメッセージが表示されます。

```
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
```

また、アクセス方式サービス・プログラムの REPRO を使用する場合は、必ず新しい空のログを定義してください。

REPRO を使用して、古い (名前変更された) データ・セットをそれぞれの新規データ・セット内にコピーする場合、デフォルトは NOREPLACE です。

つまり、指定されたデータ・セットに既に存在するレコードは、REPRO によって置換されません。 フォーマット設定がデータ・セットでなされるとき、RBA 値はリセットされます。 フォーマット設定の後、最終結果は、空ではないデータ・セットです。

z/OS アーカイブ・ログに対する変更

このトピックでは、アーカイブ・ログを変更する方法について知ることができます。

アーカイブ・ログに対する BSDS の中の項目のパスワードを追加、削除、および変更することができます。ここでは、例のみ示します。例の中のデータ・セット名を、 使用したいデータ・セット名に置き換えてください。 このユーティリティーの詳細については、ログ目録変更ユーティリティーを参照してください。

- アーカイブ・ログの追加
- アーカイブ・ログの削除
- アーカイブ・ログのパスワードの変更

アーカイブ・ログの追加

オブジェクトの回復が、既存のアーカイブ・ログ・データ・セットを読み取ることによって行われる場合、そのデータ・セットについての情報は、IBM MQ が見つけられるよう、BSDS に含まれていなければなりません。 既存のアーカイブ・ログ・データ・セットについての情報を BSDS に登録するには、次を使用します。

```
NEWLOG DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,
UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

アーカイブ・ログの削除

1 つまたは複数のボリューム上にある 保存ログ・データ・セット全体を削除するには、次を使用します。

```
DELETE DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

アーカイブ・ログのパスワードの変更

既存のアーカイブ・ログ・データ・セットのパスワードを変更した場合は、BSDS の中の 情報も変更する必要があります。

1. ログ・マップ印刷ユーティリティーを使用して、BSDS を表示します。
2. CSQJU003 ユーティリティーの DELETE 機能を使用して、変更されたパスワードのあるアーカイブ・ログ・データ・セットの項目を削除します (ログ目録変更ユーティリティーのトピックを参照してください)。
3. 新しいアーカイブ・ログ・データ・セットとして、データ・セットの名前を指定し ます。 CSQJU003 ユーティリティーの NEWLOG 機能 (トピック ログ目録変更ユーティリティー を参照) を使用し、新規パスワード、開始 RBA および終了 RBA 、およびボリューム通し番号 (これは印刷ログ・マップ・ユーティリティーの出力にあります。 ログ・マップ印刷ユーティリティー を参照) を指定してください。

新しいアーカイブ・ログ・データ・セットのパスワードを変更するには、 次を使用します。

```
ARCHIVE PASSWORD= password
```

新しいアーカイブ・ログ・データ・セットにパスワードを置くのを止めるには、 次を使用します。

```
ARCHIVE NOPASSWD
```

注 : 外部セキュリティー・マネージャーがない場合、ARCHIVE ユーティリティー機能のみを使用してください。

z/OS ログおよび *BSDS* に対する高位修飾子 *(HLQ)* の変更

このトピックでは、高位修飾子 (HLQ) を変更するのに必要な手順について知ることができます。

始める前に

通常、ログまたはデータ・セットを新しいデータ・セットにコピーする前に、キュー・マネージャーを終了する必要があります。 これを行うのは、確実にデータが一貫性のあるものとなり、再始動時にリカバリーを行う必要がないようにするためです。

このタスクについて

このタスクによって、ログおよび BSDS に対する HLQ の変更方法に関する情報が得られます。 そのためには、次の手順を実行してください。

手順

1. ログ印刷ユーティリティー CSQJU004 を実行し、ログ・データ・セット情報を記録します。 この情報は後で必要になります。
2. 以下のいずれかを実行できます。
   a) 名前変更するログおよび BSDS データ・セットに対して名前変更を行って、DSS バックアップおよびリストアを実行する。
   b) AMS DEFINE および REPRO を使用して HLQ データ・セットを作成し、古いデータ・セットからデータをコピーする。
3. MSTR プロシージャーおよび CHIN プロシージャーを修正し、新規データ・セットを指すようにする。
4. CSQJU003 を使用して、新しい BSDS コピー内の古いログ情報を削除します。
5. CSQJU003 の NEWLOG 関数を使用して、新しいログ・データ・セットを新しい BSDS に定義します。
   各ログに関する情報を、HLQ を除きすべて同じ状態に維持します。
6. 新しい BSDS は、古い BSDS の古いログに対して記録されたのと同じ情報を反映しているはずです。
   変更されているのは HLQ だけであるはずです。

次のタスク

キュー・マネージャーを始動する前に、新旧の BSDS に対する CSQJU004 出力を比較し、(HLQ を除いて) それらが完全に同一なものに見えることを確認します。

注 : これらの操作を行うには注意が必要です。 間違った操作を行うと、リカバリー不能な状態に陥る可能性があります。 PRINT LOG MAP UTILITY 出力を確認し、リカバリーや再始動に必要な情報すべてを含んでいることを確かめます。

z/OS *BSDS の回復*

IBM MQ が重複 BSDS モードで作動しているときに 1 つの BSDS が損傷を受けて IBM MQ が強制的に単一 BSDS モードになった場合でも、IBM MQ は次の再始動まで問題なく作動し続けます。

環境を重複 BSDS モードに戻すには、次のようにします。

1. アクセス方式サービスを使用して、損傷を受けた BSDS を名前変更 するか削除し、損傷を受けた BSDS と同じ名前で新しい BSDS を定義します。 制御ステートメントの例は、thlqual.SCSQPROC 内のジョブ CSQ4BREC にあります。
2. IBM MQ コマンド RECOVER BSDS を実行して、新しく割り振られたデータ・セットの中に有効な BSDS のコピーを作成し、再び重複 BSDS モードへ入ります。

IBM MQ が単一 BSDS モードで操作されていてその BSDS が損傷を受けたり、IBM MQ が重複 BSDS モードで操作されていて両方の BSDS が損傷を受けたりすると、キュー・マネージャーは停止し、BSDS データ・セットが修復されるまで再始動しません。 その場合は、次のようにします。

1. 最新のアーカイブ・ログ・データ・セットに関連する BSDS を突きとめます。 最新のアーカイブ・ログのデータ・セット名は、オフロード・プロセスが正常に完了したことを示すメッセージ CSQJ003I の最後にジョブ・ログの中で表示されています。 この手順の残りを実行する前に、上記の メッセージに示された、正常に行われたすべてのアーカイブ・ログを保管しておくことをお勧めします。

- アーカイブ・ログが DASD にある場合、BSDS は使用可能なすべての DASD に割り振られます。 BSDS 名は、対応するアーカイブ・ログ・データ・セットの名前と類似しています。 次の例にあるように、最後の修飾子の最初の文字を、A から B に変更するだけです。

  保存ログ名
  CSQ.ARCHLOG1. ***A*** 0000001

  BSDS コピー名
  CSQ.ARCHLOG1. ***B*** 0000001

- 保存ログがテープにある場合、BSDS は、 最初の保存ログ・ボリュームの最初のデータ・セットになります。 BSDS は、あとのボリュームに再度出てくることはありません。

2. 最新のアーカイブ・ログ・データ・セットに BSDS のコピーがない場合 (例えば、オフロードするときにエラーが発生したため)、より古いオフロード・プロセスから、BSDS の古いコピーを探します。
3. NEWNAME オプションを指定したアクセス方式サービス ALTER コマンドを使用して、 損傷した BSDS の名前を変更します。 損傷した BSDS を削除する場合は、アクセス方式サービスの DELETE コマンドを使用します。 アクセス方式サービスを使用して、損傷した個々の BSDS ごとに 、新しい BSDS を代替のデータ・セットとして定義します。 thlqual.SCSQPROC の中のジョブ CSQ4BREC には、新しい BSDS を定義するための アクセス方式サービスの制御ステートメントが入っています。
4. アクセス方式サービスの REPRO コマンドを使用して、 アーカイブ・ログからステップ 475 ページの『3』 で定義した代替の BSDS の 1 つに、 BSDS をコピーします。 2 番目の代替 BSDS に、 データをコピーしないでください。これは、ステップ 476 ページの『5』 で行います。

   a. 代替 BSDS の内容を印刷します。

      ログ・マップ印刷ユーティリティー (CSQJU004) を使用して、 代替 BSDS の内容を印刷します。 これにより、回復作業を続行する前に、代替 BSDS の内容を調べることができます。

   b. 代替 BSDS 内のアーカイブ・ログ・データ・セットの目録を更新します。

      ログ・マップ印刷ユーティリティーからの出力を調べ、代替 BSDS の中に、 BSDS のコピー元のアーカイブ・ログのレコードが含まれていないかどうかを確認します。 代替 BSDS が古いコピーである場合、その目録には、最近作成された アーカイブ・ログ・データ・セットがすべて含まれているとは限りません。 アーカイブ・ログ・データ・セットの BSDS 目録は、現在のサブシステムの目録を反映するよう更新する必要があります。

      ログ目録変更ユーティリティー (CSQJU003) の NEWLOG ステートメントを使用して、代替 BSDS を更新し、BSDS のコピー元のアーカイブ・ログのレコードを追加します。 アーカイブ・ログ・データ・セットがパスワード保護されている場合、 NEWLOG 機能の PASSWORD オプションを使用してください。 また、 アーカイブ・ログ・データ・セットがカタログ化されている場合、 必ず NEWLOG 機能の PASSWORD オプションを適切に CATALOG=YES に設定してください。 BSDS コピーより後に作成された追加のアーカイブ・ログ・データ・セットがあれば、NEWLOG ステートメントを使用して、それを追加します。

   c. 代替 BSDS のパスワードを更新します。

      BSDS には、アーカイブ・ログ・データ・セット用とアクティブ・ログ・データ・セット用の パスワードが入っています。 代替 BSDS 内のパスワードが、 インストール先で使用されている現在のパスワードを反映するように するには、PASSWORD オプションを指定して ログ目録変更 ARCHIVE ユーティリティー機能を使用します。

   d. 代替 BSDS の中のアクティブ・ログ・データ・セット目録を更新します。

      特殊な環境では、インストール先で、BSDS のコピー後にアクティブ・ログ・データ・セットの追加、 削除、または名前変更が行われている可能性があります。 この場合、代替 BSDS はインストール先で現在使用されているアクティブ・ログ・データ・セットの実際の数または名前を反映していません。

      代替 BSDS のログ目録からアクティブ・ログ・データ・セットを削除する必要がある場合は、ログ目録変更ユーティリティーの DELETE 機能を使用します。

      代替 BSDS のログ目録にアクティブ・ログ・データ・セットを追加する必要が ある場合は、ログ目録変更ユーティリティーの NEWLOG 機能を使用します。 NEWLOG 機能で、RBA の範囲を正しく指

定するようにしてください。 アクティブ・ログ・データ・セットがパスワード保護されている場合は、PASSWORD オプションを使用してください。

代替 BSDS ログ目録の中のアクティブ・ログ・データ・セットの名前を変更する 必要がある場合は、ログ目録変更ユーティリティーの DELETE 機能を使用し、 それに続いて NEWLOG 機能を使用します。 NEWLOG 機能で、RBA の範囲を正しく指定するようにしてください。 アクティブ・ログ・データ・セットがパスワード保護されている場合は、PASSWORD オプションを使用してください。

e. 代替 BSDS 内のアクティブ・ログ RBA 範囲を更新します。

その後、キュー・マネージャーが再始動すると、BSDS 内に表示された アクティブ・ログ・データ・セットの RBA と、実際のアクティブ・ログ・データ・セットに 存在する RBA が比較されます。 RBA が一致しないと、キュー・マネージャーは再始動しません。 古い BSDS のコピーを使用すると、問題は大きくなります。 この問題を解決するためには、ログ目録変更ユーティリティー (CSQJU003) を使用して、実際のアクティブ・ログ・データ・セットの RBA を使用して BSDS の中にある RBA を調整してください。 これは、次のように実行します。

- ログ・レコード印刷ユーティリティー (CSQ1LOGP) を使用して、 アクティブ・ログ・データ・セットの要約レポートを印刷します。 これには、開始および終了の RBA が表示されます。
- すべてのアクティブ・ログ・データ・セットの RBA が分かると、 実際の RBA の範囲と、ここで印刷した RBA の範囲を比較します。

  RBA の範囲が、すべてのアクティブ・ログ・データ・セットで等しい場合、追加の作業を行わずに、次の回復ステップに進むことができます。

  RBA の範囲が等しくない場合は、BSDS の中の値が、実際の値になるよう調整します。 RBA の範囲を調整する必要があるアクティブ・ログ・データ・セットごとに、 ログ目録変更ユーティリティーの DELETE 機能を使用して、代替 BSDS の 目録から、そのアクティブ・ログ・データ・セットを削除します。 次に NEWLOG 機能を使用して、その活動ログ・データ・セットを BSDS に 再定義します。 アクティブ・ログ・データ・セットがパスワード保護されている場合は、NEWLOG 機能の PASSWORD オプションを使用してください。

f. アクティブ・ログの各コピーに、2 つのアクティブ・ログ・データ・セットだけしか指定されていない場合は、キュー・マネージャーの再始動中に IBM MQ に問題が起こることがあります。 問題が起こる可能性があるのは、アクティブ・ログ・データ・セットの 1 つが満杯で、まだオフロードされておらず、もう 1 つのアクティブ・ログ・データ・セットも満杯になりかけているときです。 この場合には、アクティブ・ログの各コピーについて新しいアクティブ・ログ・データ・ セットを追加し、その新しい各アクティブ・ログ・データ・セットを、代替 BSDS のログ目録に定義します。

アクティブ・ログのそれぞれのコピー用に新しいアクティブ・ログ・データ・セットを定義するにはアクセス方式サービス・プログラムの DEFINE コマンドを使用し、置換 BSDS 内に新しいアクティブ・ログ・データ・セットを定義するにはログ目録変更ユーティリティーの NEWLOG 機能を使用します。 NEWLOG ステートメントで RBA の範囲を指定する必要はありません。 ただし、アクティブ・ログ・データ・セットがパスワード保護されている場合は、NEWLOG 機能の PASSWORD オプションを使用してください。 このタスクを行うための制御ステートメントの例は、thlqual.SCSQPROC 内のジョブ CSQ4LREC にあります。

5. 更新された BSDS を 2 番目の新しい BSDS データ・セットにコピーします。 このとき、これらの BSDS は同じになります。

   この時点で、2 番目の代替 BSDS の内容を印刷するために、 ログ・マップ印刷ユーティリティー (CSQJU004) を使用してください。
6. 現在のアクティブ・ログ・データ・セットの内容が失われた場合のアクションについては、 アクティブ・ログの問題を参照してください。
7. 新しく構成された BSDS を使用して、キュー・マネージャーを再始動します。 IBM MQ は現在の RBA と、アーカイブが必要なアクティブ・ログを判別します。

z/OS ページ・セットの管理

このトピックでは、キュー・マネージャーと関連付けられたページ・セットを管理する方法について知ることができます。

このトピックでは、キュー・マネージャーに関連するページ・セットの追加、コピー、および一般的な管理の方法について説明します。 この章は、次の節で構成されています。

ページ・セット、ストレージ・クラス、バッファー、およびバッファー・プールの説明、および適用されるパフォーマンスの考慮事項の一部については、 ページ・セット を参照してください。

ページ・セットに対する高位修飾子 (HLQ) の変更方法

このタスクによって、ページ・セットに対する HLQ の変更方法に関する情報が得られます。 このタスクを実行するには、以下を行います。

1. 新しい HLQ ページ・セットを定義します。
2. サイズの割り振りが古いページ・セットと同じである場合、REPRO を使用して、既存のページ・セットを空の新しい HLQ ページ・セットにコピーします。
3. ページ・セットのサイズを大きくする場合は、CSQUTIL の FORMAT 機能を使用して宛先ページをフォーマットしてから、CSQUTIL の COPYPAGE 機能を使用してソース・ページ・セットから宛先ページ・セットにすべてのメッセージをコピーします。

   詳しくは、 ページ・セットのフォーマット (FORMAT)、および ページ・セットの拡張 (COPYPAGE)を参照してください。
4. キュー・マネージャー・プロシージャー CSQP00xx DD ステートメントを変更し、新しい HLQ ページ・セットを指すようにします。

キュー・マネージャーを再始動し、ページ・セットが変更されていることを検査します。

キュー・マネージャーにページ・セットを追加する方法

ここでの説明は、キュー・マネージャーがすでに実行されていることを前提としています。 キュー・マネージャーが、例えば新しいキューを使用する新しいアプリケーションを処理しなければならないような場合に、 ページ・セットを追加する必要が生じることがあります。

新しいページ・セットを追加するには、次の手順を使用します。

1. 新しいページ・セットを定義し、フォーマットします。 このためのベースとして、thlqual.SCSQPROC(CSQ4PAGE) の中のサンプル JCL を使用することができます。 詳細については、ページ・セットのフォーマット (FORMAT) を参照してください。

   意図的に行う場合以外は、使用中のページ・セットをフォーマットしないように注意してください。 使用中のページ・セットを意図的にフォーマットするためには、FORMAT ユーティリティー機能の FORCE オプションを使用します。
2. DSN オプションを指定して DEFINE PSID コマンドを使用し、 ページ・セットをバッファー・プールに関連付けます。
3. DEFINE STGCLASS コマンドを発行して、ページ・セットの適切なストレージ・クラス定義を追加します。

4. オプションで、キュー・マネージャーの構成方法を文書化するには、次のようにします。
   a. 新しいページ・セットをキュー・マネージャーの開始済みタスク・プロシージャーに追加します。
   b. 新しいページ・セットの定義を、CSQINP1 初期設定データ・セットに追加します。
   c. 新しいストレージ・クラスの定義を、CSQ4INYR 初期設定データ・セット・メンバーに追加します。

DEFINE PSID コマンドおよび DEFINE STGCLASS コマンドの詳細については、DEFINE PSID および DEFINE STGCLASS を参照してください。

ページ・セットが満杯になった場合にどうするか

IBM MQ コマンド DISPLAY USAGE を使用することにより、ページ・セットの使用率を調べることができます。 例えば、次のコマンドを入力するとします。

```
DISPLAY USAGE PSID(03)
```

この場合、ページ・セット 03 の現在の状態が表示されます。 これによって、このページ・セットに空ページがどの程度あるかが分かります。

ページ・セットに対して 2 次エクステントを定義しておくと、 ページ・セットは、満杯になるたびに動的に拡張されます。 最後に、すべての 2 次エクステントを使い切るか、またはディスク・スペースがなくなります。 この状態になると、 アプリケーションは MQRC_STORAGE_MEDIUM_FULL の戻りコードを受け取ります。

アプリケーションが MQI 呼び出しから MQRC_STORAGE_MEDIUM_FULL の戻りコードを受け取った場合は、ページ・セットに十分なスペースが残っていません。 問題が持続するか、再発する可能性がある場合は、それを解決するための対策を立てなければなりません。

この問題には、以下のいくつかの方法で対処します。

- キューをあるページ・セットから別のページ・セットに移動することにより、ページ・セット間の負荷のバランスをとる。
- ページ・セットを拡張する。 指示については、480 ページの『ページ・セットのサイズの増加方法』を参照してください。
- ページ・セットを再定義して、4 GB を超えて最大 64 GB のサイズまで拡張できるようにする。 詳しくは、4 GB より大きくするためのページ・セットの定義を参照してください。

ページ・セット間の負荷のバランスをとる方法

ページ・セット上の負荷のバランスをとるとは、1 つまたは複数のキューに関連するメッセージを、 ある 1 つのページ・セットから、別の使用率の低いページ・セットに移動することです。 ページ・セットの拡張が現実的でない場合に、この手法を使用してください。

ある 1 つのページ・セットを使用しているキューを識別するために、該当の IBM MQ コマンドを使用します。 例えば、ページ・セット 02 にマップされているキューを調べるには、 まずコマンドを使用して、ページ・セット 02 にマップされているストレージ・クラスを調べます。

```
DISPLAY STGCLASS(*) PSID(02)
```

次のコマンドを使って、そのストレージ・クラスを使用しているキューを調べます。

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

非共有キューの移動

キューおよびその中のメッセージを、あるページ・セットから別のページ・セットに移動するには、MQSC MOVE QLOCAL コマンドを使用します (MOVE QLOCAL で説明されています)。 新しいページ・セットに移動したい 1 つまたは複数のキューを識別した あと、それらのキューの 1 つ 1 つについて、次の手順を実行してください。

1. 移動するキューがどのアプリケーションでも使用されていないこと (つまり、 DISPLAY QSTATUS コマンドからの IPPROCS および OPPROCS 値がゼロであること)、およびコミットされていないメッセージがないこと (DISPLAY QSTATUS コマンドからの UNCOM 値が NO であること) を確認します。

   注 : この状態が続いていることを確認する唯一の方法は、キューのセキュリティー許可を一時的に変更することです。 詳しくは、キュー・セキュリティーのためのプロファイルを参照してください。

   変更できない場合は、設定 PUT(DISABLED) などの予防措置にもかかわらず、アプリケーションがキューの使用を開始した場合に、 この手順の後の段階で失敗する可能性があります。 しかし、この手順でメッセージが失われることはありません。
2. キュー定義を変更して ムクプット を使用不可にすることにより、アプリケーションが移動中のキューにメッセージを書き込むことを防止します。 キュー定義を PUT(DISABLED)に変更します。
3. 次のコマンドを使って、 移動するキューと同じ属性をもった一時キューを定義します。

   ```
   DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
   ```

   注 : この一時キューが以前の実行のときからすでに存在している場合は、その一時キューを削除してから定義を行います。
4. 次のコマンドを使って、メッセージを一時キューに移動します。

   ```
   MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
   ```
5. 次のコマンドを使用して、移動するキューを削除します。

   ```
   DELETE QLOCAL(QUEUE_TO_MOVE)
   ```
6. 必要なページ・セットにマップされる新しいストレージ・クラスを定義します。 例えば、次のようにします。

   ```
   DEFINE STGCLASS(NEW) PSID(nn)
   ```

   次回のキュー・マネージャーの再始動に備えて、 CSQINP2 データ・セットに新しいストレージ・クラス定義を追加します。
7. ストレージ・クラス属性を変更することによって、 移動中のキューを再定義します。

   ```
   DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
   ```

   キューを再定義する場合、 その定義はステップ 479 ページの『3』 で作成した一時キューに基づくものです。
8. 次のコマンドを使用して、新しいキューにメッセージを戻します。

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

9. これで、ステップ 479 ページの『3』 で作成したキューは不要になります。 次のコマンドを使用して、削除します。

```
DELETE QL(TEMP_QUEUE)
```

10. 移動するキューが CSQINP2 データ・セットに定義されていた場合、 CSQINP2 データ・セットの中の該当の DEFINE QLOCAL コマンドの STGCLASS 属性を変更します。 既存のキュー定義が置き換えられるようにするため、REPLACE キーワードを追加してください。

480 ページの図 37 は、負荷バランスのジョブから抜き出した一部を示したものです。

```
//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)
/*
```

図 *37.* ページ・セットの負荷平準化ジョブからの抜き出し

ページ・セットのサイズの増加方法

最初に、4 GB より大きいページ・セットを割り振ることができます。4 GB より大きくするためのページ・セットの定義を参照してください。

ページ・セットは、EXPAND(SYSTEM) または EXPAND(USER) を指定することにより、 フルになりかかると自動的に拡張されるように定義することができます。 ご使用のページ・セットが EXPAND(NONE) で定義された場合、次の 2 つの方法のいずれかで拡張することができます。

- この定義を変更して、自動拡張を許可します。 自動拡張を許可するページ・セットの変更を参照してください。
- より大きな新規のページ・セットを作成し、古いページ・セットから 新しいページ・セットにメッセージをコピーします。 より大きな新規のページ・セットへのメッセージの移動を参照してください。

4 GB より大きくするためのページ・セットの定義

データ・セットが「拡張アドレス可能度」を指定して VSAM に定義されていれば、IBM MQ は、最大 64 GB までのサイズのページ・セットを使用できます。 拡張アドレス可能度は、SMS データ・クラスによって与えられる属性です。

注 : ページ・セットおよびアクティブ・ログ・データ・セットは、拡張アドレス・ボリューム (EAV) の拡張アドレッシング・スペース (EAS) 部分に配置することが可能で、z/OS V1.12 以降では、アーカイブ・ログ・データ・セットも EAS に配置することができます。

下記のサンプル JCL で示される例では、管理クラス「EXTENDED」が「拡張アドレス可能度」を指定して SMS に定義されています。 既存のページ・セットが、拡張アドレス可能度を持つとして現在定義 されていない場合は、以下の方法を使用して、拡張アドレス可能度のフォーマットのデータ・セットにマイグレーションします。

1. キュー・マネージャーを停止させます。
2. アクセス方式サービス・プログラムを使用して、既存のページ・セットを名前変更します。
3. 宛先ページ・セットを、既存のページ・セットと同じサイズで、DATACLAS(EXTENDED) を指定して定義します。

   注 : 拡張フォーマット・データ・セットは、SMS により管理される必要があります。 下記の内容は、VSAM データ・セットに対して拡張フォーマットを要求するためのメカニズムです。

   - DSNTYPE の値が EXT およびそのサブパラメーター R または P (各々、R は必須、P は優先を表す) のデータ・クラスを使用。
   - DD ステートメント上で DSNTYPE=EXTREQ (拡張フォーマットが必須) または DSNTYPE=EXTPREF (拡張フォーマットを優先) を指定。
   - DD ステートメント上で LIKE= パラメーターを指定して、既存の拡張フォーマット・データ・セットを参照。

   詳しくは、 拡張フォーマット・データ・セットの定義に関する制約事項を参照してください。
4. CSQUTIL の COPYPAGE 機能を使用して、すべてのメッセージをソース・ページ・セットから宛先ページ・セットにコピーします。 詳細については、ページ・セットの拡張 (COPYPAGE) を参照してください。
5. キュー・マネージャーを再始動する。
6. ページ・セットをシステム拡張機構を使用するように変更して、 その現在の割り振りを超えて、継続して増大できるようにします。

以下の JCL は、アクセス方式サービス・プログラムのコマンドの例を示しています。

```
//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))
/*
```

自動拡張を許可するページ・セットの変更

EXPAND(USER) または EXPAND(SYSTEM) オプションを指定して ALTER PSID コマンドを使用します。 ページ・セットの拡張についての一般情報は、ALTER PSID およびページ・セットの拡張 (COPYPAGE) を参照してください。

より大きな新規のページ・セットへのメッセージの移動

この手法では、キュー・マネージャーの停止および再始動が行われます。 その結果、共有キュー上にない非持続メッセージは再始動時に削除されます。 削除しない非持続メッセージがある場合は、代わりにロード・バランシングを使用してください。 詳細は、 478 ページの『ページ・セット間の負荷のバランスをとる方法』を参照してください。 ここでの説明では、拡張するページ・セットはソース・ページ・セットと呼び、新しい大きなページ・セットは宛先 ページ・セットと呼びます。

次のステップを行います。

1. キュー・マネージャーを停止させます。
2. 宛先ページ・セットを定義します。このとき、ソース・ページ・セット より大きくするため、より大きな 2 次エクステント値を指定します。

3. CSQUTIL の FORMAT 機能を使用して、宛先ページ・セットをフォーマットします。 詳細については、ページ・セットのフォーマット (FORMAT) を参照してください。
4. CSQUTIL の COPYPAGE 機能を使用して、すべてのメッセージをソース・ページ・セットから宛先ページ・セットにコピーします。 詳細については、ページ・セットの拡張 (COPYPAGE) を参照してください。
5. 次のいずれかを実行することにより、 宛先ページ・セットを使用してキュー・マネージャーを再始動します。
   - キュー・マネージャーの開始済みタスク・プロシージャーを変更して、宛先ページ・セットを 参照するようにします。
   - アクセス方式サービスを使用してソース・ページ・セットを削除してから、 宛先ページ・セットの名前を変更し、ソース・ページ・セットと同じ名前にします。

注意 :

IBM MQ ページ・セットを削除する前に、必要なバックアップ・コピーを既に作成してあることを必ず確認してください。

ページ・セットを縮小する方法

IBM MQ 管理者以外のすべてのユーザーが、キュー・マネージャーを使用できないようにします。 例えばアクセス・セキュリティーの設定を変更するなどで実行します。

ほとんど空に近い (DISPLAY USAGE コマンドで分かります) 大きなページ・セットがある場合は、 そのサイズを縮小できます。 この手順では、CSQUTIL の COPY、FORMAT、および LOAD の各機能を使用します (IBM MQ ユーティリティー・プログラムを参照)。 この手順は、このページ・セットのサイズを縮小するには現実的ではないので、 ページをゼロ (0) には設定しません。縮小するための唯一の方法は、 キュー・マネージャーを再度初期化することです (506 ページの『キュー・マネージャーの再初期設定』を参照)。 この手順の前提条件は、すべての UOW が完了し、ページ・セットが一貫するように、システムからすべてのユーザーを除去することです。

1. STOP QMGR コマンドを QUIESCE 属性または FORCE 属性と共に使用して、キュー・マネージャーを停止します。
2. CSQUTIL の SCOPY 機能を PSID オプションを使用して実行し、大きいページ・セットに入っているすべてのメッセージ・データを 1 つの順次データ・セットの中にコピーして保存します。
3. 新規の小さなページ・セットのデータ・セットを定義し、大きな ページ・セットの代わりにします。
4. CSQUTIL の FORMAT TYPE(NEW) 機能を、ステップ 482 ページの『3』 で作成したページ・セットに対して実行します。
5. ステップ 482 ページの『3』 で作成したページ・セットを使用して、キュー・マネージャーを再始動します。
6. CSQUTIL の LOAD 機能を使用して、ステップ 482 ページの『2』 で保管したすべてのメッセージをロードし直します。
7. すべてのユーザーに、キュー・マネージャーへのアクセスを許可します。
8. 不要になった大きなページ・セットを削除します。

ページ・セットの再導入の方法

特定の状況においては、古いページ・セットを再びキュー・マネージャーに対してオンライン にできると便利です。 特定のアクションを行わないかぎり、古いページ・セットがオンラインになると、 キュー・マネージャーは、ページ・セット自体の中に保管されているか、 またはチェックポイント・レコードに保管されているページ・セット・リカバリー RBA を古いものとして認識します。 そのため、ページ・セットのメディア・リカバリーを自動的に開始して、 ページ・セットを最新のものにします。

そのようなメディア・リカバリーは、キュー・マネージャーの再始動時にのみ実行することができ、特に、テープ上に保存されたアーカイブ・ログを読み取らなければならない場合、 非常に長い時間かかる可能性があります。 ただし、通常、この状況においては、ページ・セットは介入期間の間オフラインになっており、 そのため、ログにはページ・セット・リカバリーに関係する情報は入っていません。

以下の 3 つの選択が可能です。

完全メディア・リカバリーを実行するようにする。

1. キュー・マネージャーを停止させます。
2. キュー・マネージャーの開始済みタスク・プロシージャーと CSQINP1 初期設定データ・セットの両方で、定義がページ・セットに対して使用可能であることを確認します。
3. キュー・マネージャーを再始動する。

ページ・セット上のすべてのメッセージが破棄されるようにする。

この選択は、ページ・セットが長時間 (例えば、数カ月間) オフラインになっており、ここで他の目的のために再利用することに決定した場合に便利です。

1. TYPE(NEW) オプションを指定した CSQUTIL の FORMAT 機能を使用して、 ページ・セットをフォーマット設定します。
2. ページ・セットの定義を、キュー・マネージャーの開始済みタスク・プロシージャーと CSQINP1 初期設定データ・セットの両方に追加します。
3. キュー・マネージャーを再始動する。

フォーマット設定に TYPE(NEW) オプションを使用すると、ページ・セットの現在の内容を消去して、ページ・セットに関するチェックポイントの中のすべてのヒストリカル情報を無視するよう、キュー・マネージャーに指示します。

メディア・リカバリー処理を回避して、ページ・セットをオンラインにする。

この手法は、キュー・マネージャーのクリーン・シャットダウン以降に、 ページ・セットがオフラインであったことが確実である場合にのみ使用してください。 この選択は、通常、キュー・マネージャーが開始している間にバックアップが実行されるなど、 操作上の問題のため、ページ・セットが短期間オフラインになっていたような場合に 最も適しています。

1. TYPE(REPLACE) オプションを指定した CSQUTIL の FORMAT 機能を使用して、 ページ・セットをフォーマット設定します。
2. DSN オプションを指定した DEFINE PSID コマンドを使用してページ・セット を動的にキュー・マネージャーに戻して追加するか、またはページ・セットをキュー・マネージャー の再始動時に追加できるようにするかのいずれかを行います。

フォーマット設定に TYPE(REPLACE) オプションを使用すれば、ページ・セットがキュー・マネージャー によってクリーンにクローズされたことが確認され、 メディア・リカバリーが実行されないようにマークが付けられます。 ページ・セットの内容に対して、その他の変更は行われません。

z/OS ページ・セットのバックアップおよび回復の方法

バックアップと回復には、さまざまな手段を使用できます。 このトピックでは、それらの手段について説明します。

このセクションには、以下のトピックが記載されています。

- 483 ページの『非共用資源の回復点の作成』
- 485 ページの『ページ・セットのバックアップ』
- 485 ページの『ページ・セットの回復』
- ページ・セットの削除方法

共有資源の回復点を作成する方法については、492 ページの『共有キューの回復』 を参照してください。

非共用資源の回復点の作成

IBM MQ がオブジェクトと非共有持続メッセージを現在の状態に回復できるのは、次の条件が両方とも当てはまる場合です。

1. 以前の点からのページ・セットのコピーが存在する。

2. すべての IBM MQ ログが、その点からの回復を実行するために使用可能である。

これらは非共有資源の回復点を表します。

オブジェクトとメッセージは、どちらもページ・セットに保持されます。 異なるキューからの複数のオブジェクトとメッセージが、 同じページ・セットに入ります。 回復する目的で、オブジェクトとメッセージを切り離してバックアップすることはできません。 したがって、正しくデータを回復するためには、1 つのページ・セット全体をバックアップする必要があります。

IBM MQ 回復ログには、すべての持続メッセージとオブジェクトへの変更がレコードとして含まれています。 IBM MQ に障害 (例えば、ページ・セット上の入出力エラーなどで) が起きた場合、バックアップ・コピーを復元してキュー・マネージャーを再始動することによって、ページ・セットを回復できます。 IBM MQ は、バックアップ・コピーの時点からのログの変更をページ・セットに適用します。

回復点の作成方法には 2 通りあります。

全バックアップ
キュー・マネージャーを停止します。これにより、すべての更新が強制的にページ・セットに適用されます。

これによって、 回復点以降にバックアップをとったページ・セットのデータ・セットとログだけを使用して、 回復点から再始動することができます。

ファジー・バックアップ
キュー・マネージャーを停止しないで、 ページ・セットのファジー・バックアップ・コピーを取ります。

この方法を使用した場合、 関連するログがその後損傷したか失われたときは、 ページ・セットのファジー・バックアップ・コピーを使用して回復することはできません。 その理由は、ページ・セットのファジー・バックアップ・コピーには、 キュー・マネージャーの状態について不整合のビューが入っており、その バックアップ・コピーは使用可能なログによって異なるためです。 ログが使用可能でない場合は、サブシステムが活動状態にない間に取られた最後の ページ・セットのバックアップ・コピー (方法 1) に戻る必要があります。その時点以降のデータの消失は受け入れます。

方法 1: 全バックアップ

この方法では、キュー・マネージャーの終了を行います。 これによって、すべての更新は強制的にページ・セットに適用されるため、 ページ・セットは整合状態になります。

1. キュー・マネージャーを使用しているすべての IBM MQ アプリケーションを停止します (最初にそれらのアプリケーションを完了させることができます)。 これは、例えばアクセス・セキュリティーおよびキューの設定を変更するなどで実行します。
2. 活動がすべて完了すると、未確定のリカバリー単位を表示し、解決します。 (DISPLAY CONN および RESOLVE INDOUBT の説明に従って、コマンド DISPLAY CONN および RESOLVE INDOUBT を使用します。)

   これにより、ページ・セットは整合性のある状態になります。 この作業を行わない場合は、ページ・セットの不整合が生じる可能性があり、 実際には「ファジー」バックアップを行っていることになります。
3. ARCHIVE LOG コマンドを実行して、 最新のログ・データがログ・データ・セットに書き込まれるようにします。
4. STOP QMGR MODE(QUIESCE) コマンドを実行します。 CSQI024I または CSQI025I メッセージで最も小さい RBA 値を記録します (詳細は、CSQI024I および CSQI025I を参照)。 RBA 値によって示されたものから現在のものまでログ・データ・セットを保管する必要があります。
5. キュー・マネージャーのページ・セットすべてのバックアップ・コピーを取ります (485 ページの『ページ・セットのバックアップ』を参照)。

方法 2: ファジー・バックアップ

この方法では、キュー・マネージャーの終了を行いません。 したがって、更新はバックアップ処理時に仮想記憶バッファーに入っている可能性があります。 これは、ページ・セット間に整合性がない状態であることを意味しており、 ログによる回復の場合にだけ、それらのページ・セットを使用できます。

1. DISPLAY USAGE TYPE(ALL) コマンドを実行し、 CSQI024I または CSQI025I メッセージの RBA 値を記録します (詳細は、CSQI024I および CSQI025I を参照)。
2. ページ・セットのバックアップ・コピーを取ります (485 ページの『ページ・セットのバックアップ』を参照)。
3. ARCHIVE LOG コマンドを実行して、 最新のログ・データがログ・データ・セットに書き込まれるようにします。 回復点から再始動するには、RBA 値によって示されたものから現在のものまでログ・データ・セットを保管する必要があります。

ページ・セットのバックアップ

ページ・セットを回復するためには、IBM MQ は、ログをどこまでさかのぼるか認識している必要があります。 IBM MQ は、回復ログ順序番号 (LSN) と呼ばれるログ RBA 番号を、各ページ・セットのページ 0 に保持しています。 この番号は、そこから IBM MQ がページ・セットを回復できる、ログ内の開始 RBA です。 ページ・セットをバックアップすると、この番号もコピーされます。

あとで、このコピーがページ・セットの回復に使用される場合、IBM MQ は、この RBA 値から現在の RBA までの、すべてのログ・レコードにアクセスできなければなりません。 したがって、保持したい最も古いページ・セットのバックアップ・コピーから IBM MQ が回復することができるよう、十分なログ・レコードを保持しなければなりません。

ADRDSSU COPY 関数を使用して、ページ・セットをコピーします。

詳しくは、 論理データ・セットの COPY DATASET コマンド構文 の資料を参照してください。

以下に例を示します。

```
//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)
/*
//
```

キュー・マネージャーが実行されている間にページ・セットをコピーする場合、 まずページ・セットのページ 0 をコピーするコピー・ユーティリティーを使用する必要があります。 このユーティリティーを使用しないと、ページ・セットのデータが壊される可能性があります。

ページ・セットの動的な拡張処理において割り込みが起きた場合でも (例えば、システムへの電源が落ちるなど)、ADRDSSU を使用してページ・セットのバックアップを取れます。

アクセス方式サービス・プログラム IDCAMS `LISTCAT ENT('page set data set name') ALLOC` を実行すると、 `HI-ALLOC-RBA` が `HI-USED-RBA` より高いことが分かります。

このページ・セットが次に満杯になると、可能であれば再び拡張され、 他の新しいエクステントと共に、HI-USED-RBA と HI-ALLOC-RBA 間のページが使用されます。

オブジェクト定義のバックアップ

オブジェクト定義のコピーもバックアップしてください。 そのためには、CSQUTIL COMMAND 機能の MAKEDEF 機能 (IBM MQ へのコマンドの実行 (COMMAND) を参照) を使用します。

キュー・マネージャーのバックアップ・コピーを取るときには必ずオブジェクト定義をバックアップし、常に最新の状態にしておいてください。

ページ・セットの回復

キュー・マネージャーが障害のために終了した場合には、通常、 キュー・マネージャーを再始動することができます。その際、 再始動の間にすべての回復処理が行われます。 ただし、いずれのページ・セットまたはログ・データ・セットも 入手できない場合には、こうした回復は行えません。 どの程度まで回復できるかは、ページ・セットおよびログ・データ・セットのバックアップ・コピーをどれだけ入手できるかによって異なります。

回復点から再始動するには、次のものが必要です。

- 回復したいページ・セットのバックアップ・コピー
- 484 ページの『方法 2: ファジー・バックアップ』で記述されている "ファジー" バックアップ・プロセスを使用した場合は、記録された RBA 値を含むログ・データ・セット、 ARCHIVE LOG コマンドによって作成されたログ・データ・セット、およびこれらの間のすべてのログ・データ・セットが含まれています。
- 全バックアップを使用した場合でも、ARCHIVE LOG コマンドによって作成したログ・データ・セットの後にログ・データ・セットがない場合には、すべてのページ・セットに対して CSQUTIL ユーティリティーの FORMAT TYPE(REPLACE) 機能を実行する必要は**ありません**。

ページ・セットを現在の状態まで回復するには、ARCHIVE LOG コマンド 以降のすべてのログ・データ・セットおよびレコードも必要です。

ページ・セットの回復方法は 2 つあります。 いずれの方法を使用する場合にも、キュー・マネージャーは停止していなければなりません。

簡単な回復

次の方法はより簡単な方法であり、ほとんどの回復状況に適応します。

1. バックアップから復元したいページ・セットを削除します。
2. ADRDSSU COPY 関数を使用して、バックアップ・コピーからページ・セットを復元します。

   あるいは、バックアップ・コピーを元の名前に変更するか、 またはバックアップ・ページ・セットを指すよう、キュー・マネージャーの 手順で CSQP00xx DD ステートメントを変更することができます。 しかし、ページ・セットが失われるか、または壊れた場合は、 復元するためのバックアップ・コピーを取ることができません。
3. キュー・マネージャーを再始動する。
4. キュー・マネージャーが正常に再始動した場合、アプリケーションを再始動することができます。
5. 復元したページについて、通常のバックアップ手順を復帰させます。

高度な回復

この方法は、回復するページ・セットが大きい場合、または最後にバックアップ・コピーを取った後にページ・セットに対して多くの活動があった場合に、パフォーマンス上の利点があります。 しかし、簡単な方法と比べるとより多くの手操作を行う必要があります。したがって、 エラーが発生する危険性が高くなり、回復を実行するためにかかる時間が長くなります。

1. バックアップから復元したいページ・セットを削除し再定義します。
2. ADRDSSU を使用して、ページ・セットのバックアップ・コピーを、新しいページ・セットにコピーします。 ページ・セットが動的に拡張されるように、 2 次エクステント値を使用して新しいページ・セットを定義します。

   あるいは、バックアップ・コピーを元の名前に変更するか、 またはバックアップ・ページ・セットを指すよう、キュー・マネージャーの 手順で CSQP00xx DD ステートメントを変更することができます。 しかし、ページ・セットが失われるか、または壊れた場合は、 復元するためのバックアップ・コピーを取ることができません。
3. キュー・マネージャーの CSQINP1 定義を変更して、 回復されるページ・セットと関連するバッファー・プールをできるだけ大きく取ります。 バッファー・プールのサイズを大きくできると、変更したすべてのページをバッファー・プール内に保持でき、ページ・セットへの入出力量を減らすことができます。
4. キュー・マネージャーを再始動する。

5. キュー・マネージャーが正常に再始動した場合、(静止機能を使用して) キュー・マネージャーを停止し、次にそのページ・セットの通常のバッファー・プール定義を使用して再始動します。 キュー・マネージャーが正常に再始動した後、アプリケーションを再始動することができます。
6. 復元したページについて、通常のバックアップ手順を復帰させます。

キュー・マネージャーを再始動したときの処理

キュー・マネージャーは、再始動時に、ページ・セットに対して行われた、 ログに登録されているすべての変更をページ・セットの再始動点から適用します。 このようにして、IBM MQ は複数のページ・セットを復旧させることができます。 メディア回復の間も、必要であれば、ページ・セットは動的に拡張されます。

IBM MQ は、再始動時に次のうちの最も小さい値を取ることにより、開始するログ RBA を決定します。

- 各ページ・セットのチェックポイント・ログ・レコードからの回復 LSN
- 各ページ・セットのページ 0 からの回復 LSN
- バックアップが取られたときにシステムの中にあった、最も古い未完了リカバリー単位の RBA

すべてのオブジェクト定義は、ページ・セット 0 に保管されます。 メッセージは、任意の使用可能なページ・セットに保管できます。

注 : ページ・セット 0 が使用できない場合、キュー・マネージャーを再始動することはできません。

ページ・セットの削除方法

ページ・セットは、DELETE PSID コマンドを使用して削除します (このコマンドの詳細については、DELETE PSID を参照)。

ストレージ・クラスによってまだ参照されているページ・セットは削除できません。 DISPLAY STGCLASS を使用して、ページ・セットを参照するストレージ・クラスを検索してください。

データ・セットは IBM MQ から割り振り解除されますが削除はされません。 これは将来の使用のために残されるか、または z/OS 機能を使用して削除することができます。

ページ・セットをキュー・マネージャーの開始済みタスク・プロシージャーから除去します。

ページ・セットの定義を CSQINP1 初期設定データ・セットから除去します。

z/OS CSQUTIL を使用したキューのバックアップおよび回復の方法

このトピックを使用して、CSQUTIL を使用したバックアップおよび復元についての詳細情報を参照します。

キューのバックアップおよび復元のために、CSQUTIL ユーティリティーの機能を使用することができます。 キューをバックアップするには、COPY または SCOPY 機能を使用して、 メッセージをキューからデータ・セットにコピーします。 キューを復元するには、補足機能である LOAD または SLOAD を使用します。 詳しくは、 IBM MQ ユーティリティー・プログラムを参照してください。

z/OS バッファー・プールの管理

このトピックは、バッファー・プールを変更または削除する場合に使用します。

このトピックでは、バッファー・プールの変更と削除を行う方法について説明します。 この章は、次の節で構成されています。

- 488 ページの『バッファー・プール内のバッファー数を変更する方法』
- 488 ページの『バッファー・プールを削除する方法』

バッファー・プールは、初期設定入力データ・セット CSQINP1 から発行される DEFINE BUFFPOOL コマンドを使用して、キュー・マネージャーの初期設定中に定義されます。 その属性は、このトピックで詳述されているプロセスを使用して、キュー・マネージャーの実行時に、ビジネス要件に応じて変更することができます。 キュー・マネージャーは、現在のバッファー・プール属性をチェックポイント・ログ・レコー

ドの中に記録します。 それらのサイズは、CSQINP1 内のバッファー・プール定義に REPLACE 属性が含まれていなければ、以降のキュー・マネージャーの再始動で自動的に復元されます。

DISPLAY USAGE コマンドを使用して、現在のバッファー属性を表示します。

DSN オプションを指定した DEFINE PSID コマンドを使用して、バッファー・プールを動的に定義することもできます。

バッファー・プールを動的に変更した場合、初期設定データ・セット CSQINP1 内のバッファー・プールの定義も更新する必要があります。

ページ・セット、ストレージ・クラス、バッファー、およびバッファー・プールの説明と、パフォーマンスに関して適用される考慮事項については、z/OS での計画を参照してください。

注 : バッファー・プールは、非常に多くのストレージを使用します。 バッファー・プールのサイズを増やすか、新規バッファー・プールを定義する際は、十分な量のストレージを確保してください。 詳しくは、アドレス・スペース・ストレージを参照してください。

バッファー・プール内のバッファー数を変更する方法

バッファー・プールが小さすぎる場合、コンソール上にメッセージ CSQP020E が表示される可能性があります。この場合、以下のように ALTER BUFFPOOL コマンドを使用して、バッファー・プールに追加のバッファーを割り振ることができます。

1. ログ内の CSQY220I メッセージを参照して、新規バッファー用に使用可能なスペース量を決定します。使用可能なスペースは MB で報告されます。 バッファーのサイズが 4 KB の場合、使用可能なスペースの MB ごとに、256 のバッファーを 割り振ることができます。 フリー・スペースの一部は他のタスクに必要なため、 フリー・スペースをすべてバッファーに割り振ることはしないでください。

   バッファー・プールで固定の 4 KB ページを使用する場合、つまり、PAGECLAS 属性が FIXED4KB である場合は、LPAR 上に使用可能な十分な実ストレージを確保してください。
2. 報告されたフリー・スペースが不十分である場合、 次のコマンドを使用して、別のバッファー・プールからいくつかのバッファーを解放してください。

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   buf-pool-id は、スペースの再利用元となるバッファー・プールです。 *integer* は、このバッファー・プールに割り振られるバッファーの新しい数です。 この数は、バッファー・プールに割り振られている元のバッファー数より小さくなければなりません。
3. 次のコマンドを使用して拡張するバッファー・プールにバッファーを追加します。

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   buf-pool-id は、拡張されるバッファー・プールです。 *integer* は、このバッファー・プールに割り振られるバッファーの新しい数です。 この数は、バッファー・プールに割り振られている元のバッファー数より大きくなければなりません。

バッファー・プールを削除する方法

バッファー・プールがページ・セットによって使用されなくなった場合、 このバッファー・プールを削除して、割り振られている仮想ストレージを開放してください。

バッファー・プールは、DELETE BUFFPOOL コマンドを使用して削除します。 このバッファー・プールを使用しているページ・セットがあると、このコマンドは失敗します。

ページ・セットを削除する方法については、487 ページの『ページ・セットの削除方法』を参照してください。

z/OS z/OS でのキュー共有グループと共有キューの管理

IBM MQ は、キュー共有グループ、共有キュー、およびカップリング・ファシリティーなどの、さまざまなタイプの共有リソースを使用できます。 このトピックでは、これらの共有リソースを管理するために必要な手順について検討します。

このセクションでは、以下のトピックに関する情報を取り上げます。

- 489 ページの『キュー共有グループの管理』
- 492 ページの『共有キューの管理』
- 497 ページの『グループ・オブジェクトの管理』
- 497 ページの『カップリング・ファシリティーの管理』

z/OS キュー共有グループの管理

キュー共有グループ (QSG) でキュー・マネージャーを追加または除去し、関連した Db2 表を管理することができます。

このトピックには、以下のタスクに関するセクションがあります。

- 489 ページの『キュー共有グループのセットアップ』
- 490 ページの『キュー共用グループにキュー・マネージャーを追加する』
- 491 ページの『キュー共用グループからキュー・マネージャーを除去する』
- 492 ページの『Db2 表からキュー共用グループを除去する』
- 492 ページの『Db2 定義の整合性の検証』

キュー共有グループのセットアップ

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

以下の手順に従って、キュー共有グループをセットアップします。

1. Db2 データ共有グループを使用する最初のキュー共有グループである場合は、Db2 環境をセットアップします。
2. カップリング・ファシリティーをセットアップします。
3. キュー共有グループを Db2 表に追加します。 キュー共有グループ・ユーティリティー (CSQ5PQSG) の ADD QSG 機能を使います。 このプログラムについては、キュー共有グループ・ユーティリティーで説明されています。 thlqual.SCSQPROC(CSQ45AQS) にサンプルが用意されています。
4. 490 ページの『キュー共用グループにキュー・マネージャーを追加する』のステップに従って、キュー共有グループにキュー・マネージャーを追加します。
5. 497 ページの『カップリング・ファシリティー構造の追加』のステップに従って、 IBM MQ にアプリケーション構造を定義します。
6. 必要に応じて、非共有キューから共有キューにマイグレーションします。
7. 可用性のために、キュー共有グループ間の入出力で使用する共有チャネルを作成します。
   - キュー共有グループへの接続の場合:
     - VIPA ソケットまたはハードウェア・ルーターをセットアップして、QSG で使用可能なキュー・マネージャー間のワークロードを分散します。
     - QSGDISP(GROUP) を使用して受信側チャネルを定義することで、そのチャネル定義を QSG のすべてのキュー・マネージャーで使用できるようにします。
     - QSG への MCA チャネル接続のために、INDISP(GROUP) を使用して、キュー・マネージャーごとにリスナーを開始します。 QSG へのクライアント接続は、INDISP(QMGR) で開始されたリスナーに接続されたままにする必要があります。

- 特定のキュー・マネージャー名ではなく QSG 名を使用して接続するように、アプリケーションを変更します。
- アプリケーションによる QSG の任意のキュー・マネージャーへの接続が許可されるように、QSG のすべてのキュー・マネージャーのチャネル認証規則が同じであることを確認します。

- キュー共有グループからの接続の場合:
  - 共用伝送キューを定義します。
  - QSGDISP(GROUP) および DEFCDISP(SHARED) を使用してアウトバウンド・チャネルを定義します。

既存のチャネルを共有チャネルに変換する場合、チャネルが使用する同期キューが変更されることによって、チャネルを開始する前に、RESET CHANNEL コマンドを発行することが必要になる場合があります。

キュー共用グループにキュー・マネージャーを追加する

キュー・マネージャーは、既存のキュー共有グループに追加することができます。

次の点に注意してください。

- キュー共有グループにキュー・マネージャーを追加するためには、そのキュー共有グループが既に存在していなければなりません。
- 1 つのキュー・マネージャーに対して、それをメンバーにできるのは 1 つのキュー共有グループだけです。

以下の手順に従って、キュー共有グループにキュー・マネージャーを追加します。

1. キュー共有グループの ESM セキュリティー管理を実装するタスクを実行して、キュー・マネージャーおよびチャネル・イニシエーターのユーザー ID に適切なアクセス権限を付与します。
2. キュー共有グループが、SMDS にデータをオフロードするように構成された CF 構造を持つ場合、SMDS 環境をセットアップするタスクを実行します。
3. キュー・マネージャーを停止させます。
4. キュー共有グループ・ユーティリティー (CSQ5PQSG) の ADD QMGR 機能を使います。 このプログラムについては、キュー共有グループ・ユーティリティーで説明されています。 thlqual.SCSQPROC(CSQ45AQM) にサンプルが用意されています。
5. システム・パラメーター・モジュールを変更してキュー共有グループ・データを追加します。
   a. CSQ6SYSP を変更して QSGDATA パラメーターを指定します。 詳細については、CSQ6SYSP の使用を参照してください。
   b. システム・パラメーター・モジュールをアセンブルしてリンクします。 ロード・モジュールのために別の名前を使用することも可能です。
   c. 新しいモジュールを使用するために始動プロセスを変更します。
6. サンプル・メンバー thlqual.SCSQPROC(CSQ4INSS) をコピーして調整し、必要な CF 構造と SYSTEM キューを定義します。 カスタマイズしたメンバーをキュー・マネージャー始動 JCL の CSQINP2 DD に追加します。
7. キュー共有グループ・システム・パラメーター・モジュールを使用して、キュー・マネージャーを再始動します。
8. オプションとして、キュー・マネージャー名の代わりにキュー共有グループ名の接頭部が付いているセキュリティー・プロファイルにマイグレーションします。
9. QSG への接続に共有チャネルが使用されている場合、アプリケーションによる QSG の任意のキュー・マネージャーへの接続が許可されるように、QSG の他のキュー・マネージャーでそれらをミラーリングするチャネル認証規則を作成します。
10. 10. オプションで、 QSG 内のキュー・マネージャーに接続されているアプリケーションが、 QSG 内の他のキュー・マネージャーがホストするキューにメッセージを書き込むことができるようにするために、以下のいずれかを実行します。

- コマンド ALTER QMGR IGQ(ENABLED) を発行して、 グループ内キューイング をオンにします。
- QSG の他のキュー・マネージャーへの伝送キューおよびチャネルを定義します。 伝送キューをターゲット・キュー・マネージャーと同じ名前で定義すると、リモート・キューおよびキュー・マネージャー別名を定義する必要がなくなります。

注 : 以前のバージョンの IBM MQ を実行するキュー・マネージャーが含まれる既存のキュー共有グループにキュー・マネージャーを追加するには、最初にグループ内にある IBM MQ の最新バージョンの共存 PTF を、グループ内のすべての以前のキュー・マネージャーに適用する必要があります。

キュー共用グループからキュー・マネージャーを除去する

キュー共有グループからキュー・マネージャーを除去できるのは、キュー・マネージャーのログが別のプロセスで必要とされず、キュー・マネージャーにより所有されるすべての SMDS が空である場合のみです。

詳しくは、 共用メッセージ・データ・セットの削除 および DELETE CFSTRUCT を参照してください。

ログに次のものが含まれる場合、ログが必要になります。

- キュー共有グループが使用するカップリング・ファシリティー (CF) アプリケーション構造体の 1 つの最新のバックアップ
- 将来の復元プロセスに必要なデータ。つまり、キュー・マネージャーが、 最新のバックアップ排他インターバル値が記述する時間から、 回復可能構造体を使用している場合

これらの事項の一方または両方に該当する場合、またはキュー・マネージャーにより所有される SMDS にメッセージが含まれている場合、キュー・マネージャーを除去することはできません。 将来の復元プロセスに必要なキュー・マネージャーのログを判別するには、TYPE(BACKUP) オプションを指定して MQSC DISPLAY CFSTATUS コマンドを使用します (このコマンドの詳細については、DISPLAY CFSTATUS を参照してください)。

以下のステップを実行して、キュー共有グループからキュー・マネージャーを除去します。

1. 共有キューにメッセージを書き込むキュー・マネージャーに接続されている、すべてのアプリケーションを停止します。
2. このキュー・マネージャーが関係する未確定の作業単位を解決します。
3. コマンド DISPLAY USAGE TYPE(SMDS) を発行して、キュー・マネージャーが所有する SMDS にメッセージがあるかどうかを判別します。
4. アプリケーション構造に対するオフロード・メッセージがある場合は、これらのメッセージがキューから取り出されるまで待機します。 DISPLAY USAGE TYPE(SMDS) によって報告されるオフロード・メッセージの数がゼロになってから作業を進める必要があります。
5. STOP QMGR MODE(QUIESCE) を使用してキュー・マネージャーを完全にシャットダウンします。
6. 暫時、待機します。このインターバルは、次のステップの BACKUP CFSTRUCT コマンドの EXCLINT パラメーターの値と同じかそれ以上です。
7. MQSC BACKUP CFSTRUCT コマンドを使用し、前のステップで必要になった EXCLINT 値を指定して、別のキュー・マネージャーで、回復可能な CF 構造体ごとに CF 構造体バックアップを実行します。
8. コマンド DISPLAY CFSTATUS (*) TYPE (BACKUP) からの出力を調べて、CF 構造体を復元するためにキュー・マネージャーのログが必要ないことを確認します。
9. CSQ5PQSG ユーティリティーの REMOVE QMGR 機能を使って、 キュー共有グループからキュー・マネージャーを除去します。 このプログラムについては、キュー共有グループ・ユーティリティーで説明されています。 thlqual.SCSQPROC(CSQ45RQM) にサンプルが用意されています。
10. キュー・マネージャーを再始動する前に、QSGDATA システム・パラメーターをデフォルト値にリセットし、システム・パラメーター・モジュールを再作成します。 システム・パラメーターを調整する方法については、CSQ6SYSP の使用を参照してください。

キュー共有グループの最後のキュー・マネージャーを除去する際は、REMOVE ではなく FORCE オプションを使用する必要があります。 これにより、リカバリーに必要なキュー・マネージャー・ログの整合性検査は実行されずに、キュー共有グループからキュー・マネージャーが除去されます。 この操作は、キュー共有グループを削除する場合にのみ実行するべきです。

Db2 表からキュー共用グループを除去する

Db2 表からキュー共有グループを除去するには、キュー共有グループ・ユーティリティー (CSQ5PQSG) の REMOVE QSG 機能を使用します。 このプログラムについては、キュー共有グループ・ユーティリティーで説明されています。 thlqual.SCSQPROC(CSQ45RQS) にサンプルが用意されています。

Db2 の共通データ共有グループ表からキュー共有グループを除去するためには、491 ページの『キュー共用グループからキュー・マネージャーを除去する』 に説明されている方法で、その前にそのキュー共有グループからすべてのキュー・マネージャーを除去しておく必要があります。

キュー共有グループ管理表からキュー共有グループのレコードを削除すると、そのキュー共有グループに関するすべてのオブジェクトと管理情報が他の IBM MQ Db2 表から削除されます。 それには、共有キューとグループ・オブジェクトの情報も含まれます。

Db2 定義の整合性の検証

Db2 オブジェクト定義が何らかの理由で不整合となった場合、キュー共有グループ内で共有キューの問題が発生することがあります。

キュー・マネージャー、CF 構造、および共有キューに対する Db2 オブジェクト定義の整合性を検証するには、キュー共有グループ・ユーティリティー (CSQ5PQSG) の VERIFY QSG 機能を使用します。 このプログラムについては、キュー共有グループ・ユーティリティーで説明されています。

z/OS 共有キューの管理

このトピックでは、共有キューの回復、移動、およびマイグレーションの方法について説明します。

ここでは、下記のタスクについて説明します。

- 492 ページの『共有キューの回復』
- 493 ページの『共有キューの移動』
- 495 ページの『非共有キューから共有キューへのマイグレーション』
- Db2 接続の中断

共有キューの回復

IBM MQ は、以下のすべてについて当てはまる場合、共有キューの持続メッセージを回復できます。

- メッセージを含む CF 構造のバックアップが実行されている。
- キュー共有グループのすべてのキュー・マネージャーのすべてのログが、 バックアップが行われた時点からの回復を実行するために使用可能である。
- Db2 が使用可能であり、構造バックアップ表が最新の CF 構造のバックアップよりも新しい。

1 つの共有キュー上のメッセージは、カップリング・ファシリティー (CF) 構造の中に保管されています。 持続メッセージは共有キューに書き込むことができ、非共有キュー上の持続メッセージのように、キュー・マネージャー・ログにコピーされます。 MQSC BACKUP CFSTRUCT および RECOVER CFSTRUCT コマンドが提供され、回復の見込みがないカップリング・ファシリティー障害のイベントで、CF 構造の回復を行うことができます。 このような状況では、影響を受けた構造に保管されていた非持続メッセージは失われますが、持続メッセージは回復可能です。 構造の回復が行われるまで、構造を使用したこれ以上のアプリケーション活動は妨げられます。

回復を可能にするには、MQSC BACKUP CFSTRUCT コマンドを使って、 カップリング・ファシリティー・リスト構造を頻繁にバックアップすることが必要です。 CF 構造のメッセージは、バックアップを作成するキュー・マネージャーの保存ログ・ データ・セットに書き込まれます。 バックアップのレコード、つまりバックアップされている CF 構造の名前、バックアップを実行しているキュー・マネージャーの名前、そのキュー・マネージャーのログ上でのこのバックアップの RBA 範囲、およびバックアップ時間が、Db2 に書き込まれます。 共有キューを現在使用していなくても (例えば、キュー共有グループを将来使用する目的でセットアップした場合)、CF リスト構造をバックアップしてください。

CF 構造は、回復を実行できるキュー・マネージャーに対して MQSC RECOVER CFSTRUCT コマンドを発行することによって回復できます。 この場合、キュー共有グループ内の任意のキュー・マネージャーを使用できます。 回復する CF 構造体を 1 つだけ指定するか、または複数の CF 構造体を同時に回復することが可能です。

前述のとおり、CF リスト構造を頻繁にバックアップすることが重要です。 そうしないと、CF 構造のリカバリーに長時間を要する可能性があります。 さらに、リカバリー処理は取り消すことができません。

共有キューの定義は 1 つの Db2 データベースの中に保管されているため、必要に応じて標準的な Db2 データベース・プロシージャーを使って回復することができます。 詳しくは、 共用キューおよびキュー共用グループ を参照してください。

共有キューの移動

ここでは、 あるカップリング・ファシリティー構造から別のカップリング・ファシリティー構造へ共有キューを移動することによってロード・バランシングを実行する方法について説明します。 また、非共有キューを共有キューに移動したり、共有キューを非共有キューに移動したりする方法についても説明します。

キューを移動する場合、プロシージャーの一部として一時キューを定義する必要があります。 なぜなら、キューの名前は固有でなければならず、 キューの性質が異なっている場合でも 2 つのキューの名前を同じにすることはできないためです。 IBM MQ では、2 つのキューの名前を同じにすることが許容されてはいますが (493 ページの『2』 のステップを参照)、それらのキューを使用することはできません。

- あるカップリング・ファシリティー構造から別のカップリング・ファシリティー構造にキューを移動する
- 非共有キューを共有キューに移動する
- 共有キューを非共有キューに移動する

あるカップリング・ファシリティー構造から別のカップリング・ファシリティー構造にキューを移動する

キューおよびその中のメッセージを、ある CF 構造から別の CF 構造に移動するには、MQSC MOVE QLOCAL コマンドを使用します。 新しい CF 構造に移動したいキューを特定した後、各キューごとに下記の手順を使ってキューを移動します。

1. 移動するキューがどのアプリケーションでも使用されていないこと、 つまりキュー共有グループ内のすべてのキュー・マネージャーについてキュー属性 IPPROCS および OPPROCS が 0 であることを確認します。
2. キュー定義を変更して ムクプット を使用不可にすることにより、アプリケーションが移動中のキューにメッセージを書き込むことを防止します。 キュー定義を PUT(DISABLED)に変更します。
3. 次のコマンドを使って、 移動するキューと同じ属性をもった一時キューを定義します。

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
```

   注 : この一時キューが以前の実行のときから存在している場合は、 その一時キューを削除してから定義を行います。

4. 次のコマンドを使って、メッセージを一時キューに移動します。

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. 次のコマンドを使用して、移動するキューを削除します。

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. 次のコマンドを使って移動するキューを再定義し、その際に CFSTRUCT 属性を変更します。

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
```

キューを再定義する場合、 その定義はステップ 493 ページの『3』 で作成した一時キューに基づくものです。

7. 次のコマンドを使用して、新しいキューにメッセージを戻します。

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

8. これで、ステップ 493 ページの『3』 で作成したキューは不要になります。 次のコマンドを使用して、削除します。

```
DELETE QL(TEMP_QUEUE)
```

9. 移動するキューが CSQINP2 データ・セットに定義されていた場合、 CSQINP2 データ・セットの中の該当の DEFINE QLOCAL コマンドの CFSTRUCT 属性を変更します。 既存のキュー定義が置き換えられるようにするため、REPLACE キーワードを追加してください。

494 ページの図 38 は、キューを 1 つの CF 構造から別の CF 構造に移動するジョブのサンプルを示します。

```
//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)
/*
```

図 *38.* キューを *1* つの *CF* 構造から別の *CF* 構造に移動するサンプル・ジョブ

非共有キューを共有キューに移動する

非共有キューを共有キューに移動する手順は、 ある CF 構造から別の CF 構造にキューを移動する手順に似ています (493 ページの『あるカップリング・ファシリティー構造から別のカップリング・ファシリティー構造にキューを移動する』 を参照)。 495 ページの図 39 にサンプル・ジョブを示します。

注 : 共有キュー上のメッセージ数は、メッセージの最大サイズ、メッセージ永続性、およびキュー・インデックス・タイプに関するいくつかの制限によって影響を受けるため、非共有キューを共有キューに移動できないことがあります。

```
//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)
/*
```

図 *39.* サンプル・ジョブ*:* 非共有キューを共有キューに移動する

共有キューを非共有キューに移動する

共有キューを非共有キューに移動する手順は、 ある CF 構造から別の CF 構造にキューを移動する手順に似ています (493 ページの『あるカップリング・ファシリティー構造から別のカップリング・ファシリティー構造にキューを移動する』を参照)。

495 ページの図 40 にサンプル・ジョブを示します。

```
//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)
/*
```

図 *40.* サンプル・ジョブ*:* 共有キューを非共有キューに移動する

非共有キューから共有キューへのマイグレーション

非共有キューから共有キューへのマイグレーションは、次の 2 段階に分けられます。

- キュー共有グループ内の最初の (または唯一の) キュー・マネージャーのマイグレーション
- キュー共有グループ内のその他のキュー・マネージャーのマイグレーション

キュー共有グループ内の最初の (または唯一の) キュー・マネージャーのマイグレーション

495 ページの図 39 に、非共有キューを共有キューに移動するジョブの例が示されています。 マイグレーションの必要な各キューごとにそれを実行してください。

注 :

1. 共有キュー上のメッセージ数は、メッセージの最大サイズ、メッセージ永続性、およびキュー・インデックス・タイプに関するいくつかの制限によって影響を受けるため、非共有キューを共有キューに移動できないことがあります。
2. 共有キューの正しいインデックス・タイプとして正しいものを使用してください。 伝送キューを共有キューにマイグレーションする場合、インデックス・タイプは MSGID でなければなりません。

キューが空の場合、またはその上のメッセージを保つ必要がない場合、 キューのマイグレーション作業はもっと簡単です。 そのような場合のジョブの例を、496 ページの図 41 に示します。

```
//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)
/*
```

図 *41.* サンプル・ジョブ*:* メッセージを含まない非共有キューを共有キューに移動する

キュー共有グループ内のその他のキュー・マネージャーのマイグレーション

1. 既存の共有キューのいずれかと同じ名前ではないキューについては、 495 ページの図 39 または 496 ページの図 41 に示されている方法で各キューを移動します。
2. 既存の共有キューのいずれかと同じ名前のキューについては、 496 ページの図 42 に示されているコマンドを使ってメッセージを共有キューに移動します。

```
MOVE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR)
```

図 *42.* 非共用キューから共用キューにメッセージを移動する

Db2 への接続の中断

キュー・マネージャーを停止せずに、共有キューに関連した Db2 の表またはパッケージに保守またはサービスを適用する場合、データ共有グループ (DSG) 内のキュー・マネージャーを Db2 から一時的に切断する必要があります。

そのためには、次のようにします。

1. MQSC コマンド SUSPEND QMGR FACILITY (Db2) を使用します。
2. バインドを行います。
3. Db2 に再接続するには、MQSC コマンド RESUME QMGR FACILITY(Db2) を使用します。

これらのコマンドの使用には制限があることに注意してください。

重要 : Db2 接続が中断されている間、以下の操作は使用できなくなります。 そのため、この作業は企業が多忙ではない時間に行う必要があります。

- 管理のための共有キュー・オブジェクトへのアクセス (定義、削除、変更)
- 共有チャネルの開始
- Db2 のメッセージの保管

- CFSTRUCT のバックアップまたはリカバリー

z/OS グループ・オブジェクトの管理

このトピックでは、グループ・オブジェクトを処理する方法について知ることができます。

グループ・オブジェクトの定義は、 その定義を使用するそれぞれのキュー・マネージャーのページ・セット 0 に、IBM MQ によって自動的にコピーされます。 定義のコピーは一時的に変更することができます。 また、IBM MQ では、リポジトリー・コピーに基づいてそのページ・セット・コピーをリフレッシュすることができます。 IBM MQ の始動時ごとに、ページ・セット・コピーがリポジトリー・コピーに基づいて自動的にリフレッシュされます (チャネル・オブジェクトの場合、これはチャネル・イニシエーターの再始動時に実行されます)。 これにより、キュー・マネージャーが非活動状態だった時点でなされた変更を含め、リポジトリーのバージョンがページ・セット・コピーに常に反映されるようになります。

リフレッシュが実行されない状況もあります。例えば、

- キューのコピーがオープンされている場合、 そのキューの使用方法を変更するリフレッシュ操作は失敗します。
- キューのコピーにメッセージが含まれている場合、 そのキューを削除するリフレッシュ操作は失敗します。

上記の場合についてはコピーに対してリフレッシュは実行されませんが、 それ以外のすべてのキュー・マネージャーのコピーについては実行されます。 グループ・オブジェクトの追加、変更、あるいは削除を実行した後、 およびキュー・マネージャーまたはチャネル・イニシエーターの再始動時には、 コピー・オブジェクトに何か問題がないかどうかを調べ、 問題があればそれを修正するようにしてください。

z/OS カップリング・ファシリティーの管理

このトピックを使用して、カップリング・ファシリティー (CF) の構造を追加または除去する方法を理解してください。

ここでは、下記のタスクについて説明します。

- 497 ページの『カップリング・ファシリティー構造の追加』
- 497 ページの『カップリング・ファシリティー構造の除去』

カップリング・ファシリティー構造の追加

カップリング・ファシリティー構造を追加するには、以下の手順を実行します。

1. CFRM ポリシー・データ・セットに CF ストラクチャーを定義します。 カップリング・ファシリティーのセットアップにはカップリング・ファシリティーのセットアップに関する情報が載せられており、そこではカップリング・ファシリティー構造の命名規則や、CFRM ポリシー・データ・セットの中で構造を定義する方法について説明されています。
2. SMDS にメッセージ・データをオフロードする構造を構成する場合、データ・セットを割り振って、事前フォーマットします。 詳しくは、共有メッセージ・データ・セットの作成を参照してください。
3. DEFINE CFSTRUCT コマンドを使用して、 IBM MQ に対して構造を定義します。

カップリング・ファシリティー構造の除去

カップリング・ファシリティー構造を除去するには、以下の手順を実行します。

1. 下記のコマンドを使うことによって、削除するカップリング・ファシリティー構造を使用しているすべてのキューのリストを入手します。

```
DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)
```

2. その構造を使用しているすべてのキューを削除します。
3. DELETE CFSTRUCT コマンドを使用して、 IBM MQ から CF 構造を削除します。
4. 構造が、SMDS にメッセージ・データをオフロードするように構成されていた場合、SMDS を削除します。
5. CFRM ポリシー・データ・セットから構造の定義を除去した後、 IXCMIAPU ユーティリティーを実行します。 (これは、カップリング・ファシリティーのセットアップに記載されているカスタマイズ作業 (カップリング・ファシリティーのセットアップ) の逆です。)

z/OS カップリング・ファシリティー・リスト・モニターのチューニング

このトピックでは、カップリング・ファシリティー・リスト・モニターについて説明します。

カップリング・ファシリティー (CF) リスト・モニターは、IBM MQ 共有キューが属するリスト構造の状態をモニターするために使用されます。 ある共有キューにメッセージが追加され、そのキュー項目数がゼロからゼロ以外に遷移すると、CF はそのキュー共有グループ内のすべてのキュー・マネージャーに通知します。 通知を受けたキュー・マネージャーは、いくつかのアクションを実行する可能性があります。その中には、TRIGGER(FIRST) を使用しているトリガー・モニターや、get-wait を行っているアプリケーションに通知するアクションも含まれます。

デフォルトでは、CF は、キュー共有グループ内のすべてのキュー・マネージャーに同時に通知します。 構成によっては、このために以下のような問題が発生することがあります。

- ワークロード分散がスキューする。つまり、キュー共有グループ内の特定のキュー・マネージャー (多くの場合、最速の LPAR で実行されているキュー・マネージャーか、CF に最も近いキュー・マネージャー) にメッセージの多くが送られてしまう。
- 多数の GET が失敗する。結果として CPU 時間が浪費される。

z/OS V2R3 では、**KEYRNOTIFYDELAY** という新しいカップリング・ファシリティー・リソース・マネージャー (CFRM) 属性が導入されました。この属性は、共有キューが属するリスト構造 (つまり、管理構造ではなくアプリケーション構造) で使用することが可能で、特定のワークロードにおいてワークロードのスキューや空の MQGET 呼び出しによる影響を最小限に抑えることができます。

KEYRNOTIFYDELAY は、CFLEVEL 22 以上で実行される CF での構造に対してのみ設定できます。

この値は、0 から 1,000,000 マイクロ秒の範囲の 1 桁から 7 桁までの 10 進数でなければなりません。 ゼロ以外の値に設定した場合、キューの項目数がゼロからゼロ以外に遷移すると、CF はそのキュー共有グループからキュー・マネージャーを 1 つ選択し、そのキュー・マネージャーに通知した後、グループ内の他のすべてのキュー・マネージャーに通知します。

キュー・マネージャーはラウンドロビン方式で選択されます。 選択されたキュー・マネージャーが **KEYRNOTIFYDELAY** で記述された時間間隔内にメッセージを処理しなかった場合に、キュー共有グループ内の他のすべてのキュー・マネージャーにも通知されます。

KEYRNOTIFYDELAY について詳しくは、 Understanding Keyrange Monitoring Notification Delay を参照してください。

LISTNOTIFYDELAY と **SUBNOTIFYDELAY** という類似した 2 つの CFRM 属性があることに注意してください。 これらのどちらも、IBM MQ ワークロードに対して測定可能な影響は与えません。

z/OS z/OS での回復と再始動

このトピックでは、IBM MQ によって使用されるリカバリーおよび再始動のメカニズムについて知ることができます。

z/OS 再始動 IBM MQ

キュー・マネージャーが終了した場合、キュー・マネージャーがどのように終了したかに応じて、異なる再始動の手順が必要になります。 このトピックでは、使用可能なさまざまな再始動の手順について知ることができます。

このトピックでは、下記の状況でキュー・マネージャーを再始動する方法について説明します。

- 499 ページの『通常シャットダウン後の再始動』
- 499 ページの『異常終了後の再始動』
- 499 ページの『ページ・セットを失った場合の再始動』
- 499 ページの『ログ・データ・セットを失った場合の再始動』
- CF 構造を失った場合の再始動

通常シャットダウン後の再始動

STOP QMGR コマンドによってキュー・マネージャーが停止すると、 システムは決められた順序に従って作業を完了し、停止する前に終了チェックポイントを取ります。 キュー・マネージャーは再始動時に、システム・チェックポイントおよび回復ログからの情報を使用して、 終了時のシステム状態を決定します。

キュー・マネージャーを再始動するには、 START QMGR コマンドを使います (428 ページの『z/OS 上でのキュー・マネージャーの開始と停止』を参照)。

異常終了後の再始動

IBM MQ では、再始動が正常終了後のものか異常終了後のものかを自動的に検知します。

キュー・マネージャーの異常終了後の開始は、 STOP QMGR コマンドが実行された後の開始とは異なります。 キュー・マネージャーが異常終了する場合、作業を完了しないまま終了するか、 または終了チェックポイントを取ることなく終了します。

キュー・マネージャーを再始動するには、 START QMGR コマンドを使います (428 ページの『z/OS 上でのキュー・マネージャーの開始と停止』を参照)。 キュー・マネージャーの異常終了後にそれを再始動すると、 ログ内の情報を使用してその終了時の状況がリフレッシュされ、 種々のタスクの状況が通知されます。

通常、再始動処理では不一致の状態がすべて解決されます。 しかし場合によっては、 不一致の状態を解決するために特別なステップを実行しなければなりません。 これについては、513 ページの『作業単位の手動回復』で説明されています。

ページ・セットを失った場合の再始動

ページ・セットを失った場合にキュー・マネージャーを再始動するには、 再始動の前にバックアップ・コピーからページ・セットを復元する必要があります。 これについては、483 ページの『ページ・セットのバックアップおよび回復の方法』で説明されています。

そのような状況ではメディア回復に時間がかかるため、 キュー・マネージャーの再始動に長い時間がかかる場合があります。

ログ・データ・セットを失った場合の再始動

キュー・マネージャーを (STOP QMGR コマンドを使用して) 停止した後、ログの 2 つのコピーが両方とも失われるか、あるいは壊れていた場合、整合性のある 1 組のページ・セット (方法 1: フルバックアップを使用して作成されたもの) があれば、キュー・マネージャーを再始動することができます。

次の手順に従ってください。

1. キュー・マネージャーの中の既存の各ページ・セットに対応する新しいページ・セットを定義します。 ページ・セット定義については、タスク 15: ページ・セットを定義するを参照してください。

   新しい各ページ・セットが、それに対応する元のページ・セットより大きくなるようにします。
2. CSQUTIL の FORMAT 機能を使用して、宛先ページ・セットをフォーマットします。 詳細については、ページ・セットのフォーマットを参照してください。

3. CSQUTIL の RESETPAGE 機能を使用して、既存のページ・セットをコピーするか、またはそのままリセットすることにより、 各ページのログ RBA をリセットします。 この機能の詳細については、ページ・セットのコピーとログのリセットを参照してください。
4. CSQJU003 を使用してキュー・マネージャーのログ・データ・セットと BSDS を再定義します (ログ目録変更ユーティリティーを参照)。
5. 新しいページ・セットを使用して、キュー・マネージャーを再始動します。 そのためには、次のうちのいずれかを実行してください。
   - キュー・マネージャー開始済みタスク・プロシージャーを変更して、新しいページ・セットを参照するようにします。 詳しくは、タスク 6: IBM MQ キュー・マネージャー用のプロシージャーを作成するを参照してください。
   - アクセス方式サービスを使用することによって、古いページ・セットを削除してから、 新しいページ・セットの名前を変更して古いページ・セットと同じ名前にします。

注意 : IBM MQ ページ・セットを削除する前に、必要なバックアップ・コピーを既に作成してあることを必ず確認してください。

キュー・マネージャーがいずれかのキュー共用グループのメンバーである場合、 通常はログの逸失や破損によって GROUP および SHARED のオブジェクト定義が影響を受けることはありません。 しかし、共有キューのメッセージのいずれかが、失われたログまたは破損したログの対象となっていた作業単位に関係している場合、そのような未コミットのメッセージに対してどんな影響があるかは予測不能です。

注 : ログが破損していてキュー・マネージャーがキュー共有グループのメンバーである場合は、共有持続メッセージを回復できない可能性があります。 ただちに、キュー共有グループ内の別のアクティブ・キュー・マネージャーで、RECOVER(YES) 属性を持つすべての CF 構造に対して BACKUP CFSTRUCT コマンドを実行してください。

CF 構造を失った場合の再始動

CF 構造を失ってもキュー・マネージャーは終了しないため、 キュー・マネージャーを再始動する必要はありません。

z/OS z/OS での代替サイト回復

1 つのキュー・マネージャーまたはキュー共有グループを回復するか、あるいはディスクのミラーリングについて検討することができます。

詳しくは、以下のセクションを参照してください。

- 代替サイトでの単一キュー・マネージャーの回復
- キュー共有グループの回復
  - CF 構造のメディア回復
  - 基本サイトでのキュー共有グループのバックアップ
  - 代替サイトでのキュー共有グループの回復
- ディスク・ミラーリングの使用

代替サイトでの単一キュー・マネージャーの回復

IBM MQ のコンピューティング・センターが完全に機能しなくなった場合には、回復サイトにある別のキュー・マネージャーまたはキュー共有グループで回復することができます。 (代替サイトでのキュー共有グループについての回復手順は、 504 ページの『代替サイトでのキュー共有グループの回復』を参照してください。)

回復サイトにある別のキュー・マネージャーで回復するには、 ページ・セットとログを定期的にバックアップしておく必要があります。 災害時回復の目標は、すべてのデータ回復操作がそうであるように、データ、(更新) 処理作業量、および処理時間の損失を最小限にすることです。

回復サイトにおいて、

- 回復キュー・マネージャーの名前は、 失われたキュー・マネージャーと同じで**なければなりません**。

- 回復キュー・マネージャーで使用されるシステム・パラメーター・モジュール (例えば CSQZPARM) には、それに対応する失われたキュー・マネージャーと同じパラメーターを指定する必要があります。

この作業が終了したなら、 下記の手順に従ってキュー・マネージャーをすべて確立し直します。 この手順は、 単一のキュー・マネージャーのために回復サイトで災害時回復を実行するのに使用できます。 ここでは、次のものがすべて使用可能であることを想定しています。

- 1 次サイトの正常実行によって作成されたアーカイブ・ログと BSDS のコピー (1 次サイトではキュー・マネージャーと共にアクティブ・ログも失われます)。
- 1 次サイトにあるキュー・マネージャーからのページ・セットのコピー。 使用可能な最新のアーカイブ・ログ・コピーと同じ時点のものかそれよりも古いもの。

アクティブ・ログおよびアーカイブ・ログについて重複ロギングを使用できますが、 その場合は両方のコピーに対して BSDS 更新を適用する必要があります。

1. 新しいページ・セットのデータ・セットを定義し、 1 次サイトからのページ・セットのコピーに入っているデータをそれらにロードします。
2. 新しいアクティブ・ログ・データ・セットを定義します。
3. 新しい BSDS データ・セットを定義し、 アクセス方式サービス REPRO を使用して最新の保存 BSDS をその中にコピーします。
4. ログ・マップ印刷ユーティリティー CSQJU004 を使用して、 その最新の BSDS の情報を印刷します。 この BSDS が保存された時点での最新のアーカイブ・ログは、 アクティブ・ログとして切り捨てられており、アーカイブ・ログとしては現れません。 そのログの STARTRBA と ENDRBA を記録しておいてください。
5. この最新のアーカイブ・ログ・データ・セットを、 ログ目録変更ユーティリティー CSQJU003 を使って、 復元した BSDS に登録します。 このとき、ステップ 501 ページの『4』 で記録した STARTRBA と ENDRBA を使用します。
6. CSQJU003 の DELETE オプションを使用して、BSDS からすべてのアクティブ・ログ情報を除去します。
7. CSQJU003 の NEWLOG オプションを使用して、BSDS にアクティブ・ログを追加します。 このときは STARTRBA または ENDRBA を指定しないでください。
8. CSQJU003 を使用して、再始動制御レコードを BSDS に追加します。 CRESTART CREATE,ENDRBA=highrba を指定します (highrba は使用可能な最新のアーカイブ・ログの RBA の上限 (ステップ 501 ページの『4』 で記録したもの) に 1 を加えた値)。

   この時点で BSDS にはすべてのアクティブ・ログ (空の状態) および使用可能なすべてのアーカイブ・ログが記述され、 それらのログの末尾を超えたチェックポイントは記述されません。
9. START QMGR コマンドを使用してキュー・マネージャーを再始動します。 初期設定時に、次のようなオペレーター応答メッセージが出されます。

   ```
   CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
   REPLY Y TO CONTINUE, N TO CANCEL
   ```

   Y を入力して、キュー・マネージャーを開始します。 キュー・マネージャーが開始し、CRESTART ステートメントで指定された ENDRBA までのデータが回復されます。

CSQJU003 と CSQJU004 の使用については、IBM MQ ユーティリティーの使用を参照してください。

以下の例は、ステップ 6、7、8 での CSQJU003 の入力ステートメントのサンプルを示しています。

```
* 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
```

回復サイトでチャネル・イニシエーターを再始動する場合には、ARM を使用して異なる z/OS イメージ上でチャネル・イニシエーターを再始動する場合と同じような注意点を考慮する必要があります。 詳しくは、510 ページの『IBM MQ ネットワークでの ARM の使用』を参照してください。 復旧戦略では、IBM MQ 製品ライブラリや、IBM MQ (CICS,)を使用するアプリケーション・プログラミング環境の復旧もカバーする必要があります。

災害時回復シナリオにおいて、 ログ目録変更ユーティリティー (CSQJU003) のその他の機能を使うこともできます。 HIGHRBA 機能を使用すると、ブートストラップ・データ・セット内で書き込まれた RBA の最高値およびオフロードされた RBA の最高値を更新することができます。 CHECKPT 機能を使用すると、BSDS 内で新しいチェックポイント・キュー・レコードを追加したり、既存のチェックポイント・キュー・レコードを削除したりできます。

注意 : これらの機能を使用すると、IBM MQ データの整合性に影響を与えることがあります。 これらの機能は、災害時回復シナリオの場合のみ、IBM サービス技術員の指導のもとで使用してください。

高速コピー手法

キュー・マネージャーをフリーズしてすべてのページ・セットとログのコピーを作成すると、 これらのコピーは整合したものになり、 代替サイトでのキュー・マネージャーの再始動に使用することができます。 これらのコピーを使用するとメディア回復の作業が少ないため、通常はキュー・マネージャーを迅速に再始動できます。

SUSPEND QMGR LOG コマンドを使ってキュー・マネージャーをフリーズします。 このコマンドによって、 バッファー・プールがページ・セットにフラッシュされ、 チェックポイントが取られ、 以後のログ書き込み活動が停止します。 ログ書き込み活動が中断すると、 キュー・マネージャーは RESUME QMGR LOG コマンドが実行されるまで事実上フリーズされます。 キュー・マネージャーがフリーズ状態の間に、ページ・セットとログをコピーします。

FLASHCOPY または SNAPSHOT などのコピー・ツールを使ってページ・セットとログを高速にコピーすると、 キュー・マネージャーのフリーズ状態の時間を最小限にすることができます。

ただしキュー共有グループでは、SUSPEND QMGR LOG コマンドは適切なソリューションではない場合があります。 これを有効に行うには、 すべてのログのコピーに回復の同一の時点が含まれていなければなりません。 つまり、SUSPEND QMGR LOG コマンドが、キュー共有グループ内のすべてのキュー・マネージャーで同時に実行されることが必要になります。 そのためキュー共有グループ全体がしばらくの間フリーズされることになります。

キュー共有グループの回復

基本サイトに災害が発生した場合、 基本サイトのバックアップ・データ・セットを使って、 リモート・サイトでキュー共有グループを再始動することができます。 キュー共有グループを回復するには、そのキュー共有グループ内のすべてのキュー・マネージャー間で回復を調整し、主として Db2 などの他の資源に合わせる必要があります。 このセクションでは、これらのタスクについて詳しく説明します。

- CF 構造のメディア回復
- 基本サイトでのキュー共有グループのバックアップ
- 代替サイトでのキュー共有グループの回復

CF 構造のメディア回復

共有キューでの持続メッセージの保持に使用される CF 構造のメディア回復には、ログに記録されている更新内容を適用して順方向回復が行えるメディアのバックアップがあることが必要です。 MQSC BACKUP CFSTRUCT コマンドを使って定期的に CF 構造のバックアップをとってください。 共有キュー (MQGET および MQPUT) に対する更新はすべて、 更新を実行するキュー・マネージャーのログに書き込まれます。 CF 構造のメディア回復を行うには、ログに記録された更新内容を、CF 構造を使用していたすべてのキュー・マネージャーのログのバックアップに対して適用する必要があります。 MQSC RECOVER CFSTRUCT コマンドを使用すると、IBM MQ は関係のあるキュー・マネージャーのログを自動的にマージし、そこでの更新内容を最新のバックアップに対して適用します。

CF 構造のバックアップが BACKUP CFSTRUCT コマンドを処理したキュー・マネージャーのログに書き込まれるため、 さらにデータ・セットを収集して代替サイトに移す必要はありません。

基本サイトでのキュー共有グループのバックアップ

基本サイトでは日常的に一連のバックアップの整合性を確立しておく必要があります。 災害発生時にはこれらのバックアップを使用して、 代替サイトでキュー共有グループを再作成することができます。 単一のキュー・マネージャーの場合は任意の時点まで回復することができ、通常はリモート・サイトで使用可能なログの末尾まで回復できます。 ただし持続メッセージが共有キューに保管されていた場合は、 キュー共有グループ内のいずれかのキュー・マネージャーがキューに対して更新 (MQPUT または MQGET) を行った可能性があるため、 キュー共有グループ内のすべてのキュー・マネージャーのログをマージして共有キューを回復する必要があります。

キュー共有グループの回復の場合は、 すべてのキュー・マネージャーのログ・データのログ範囲に含まれる時点を設定する必要があります。 ただしログのメディアの回復が行えるのは**順方向**だけであるため、 設定する時点は BACKUP CFSTRUCT コマンドの実行以降で、 ページ・セットのバックアップの実行以降の時点でなければなりません。 (回復の時点は、通常は労働日の最終日や週末になることが考えられます。)

次の図は、キュー共有グループ内の 2 つのキュー・マネージャーを時系列で示したものです。 それぞれのキュー・マネージャーで、ページ・セットのファジー・バックアップが行われます (方法 2: ファジー・バックアップを参照してください)。 キュー・マネージャー A で BACKUP CFSTRUCT コマンドが実行されます。 続いて各キュー・マネージャーで ARCHIVE LOG コマンドが実行され、 アクティブ・ログが切り捨てられてキュー・マネージャーからオフラインのメディアにコピーされます。 これが代替サイトに移すことができるメディアです。 ログの終わりは ARCHIVE LOG コマンドが実行された時点を示しており、 通常は代替サイトで使用可能なログ・データの範囲になります。 回復する時点は、 ページ・セットまたは CF 構造のバックアップの終了時点と、代替サイトで使用可能なログの最も早い終了時点の間でなければなりません。

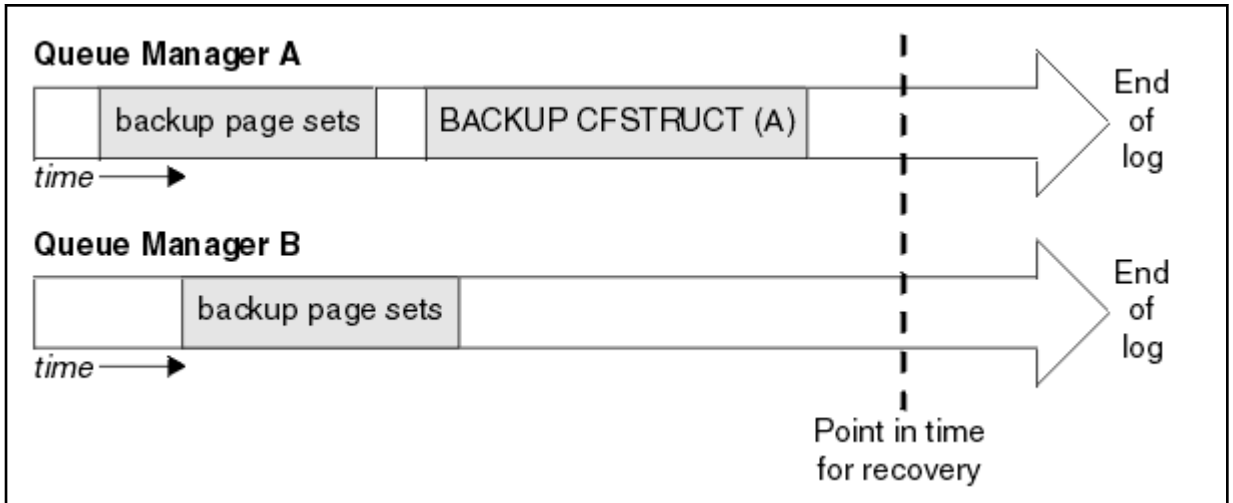


図 *43.* キュー共有グループ内の *2* つのキュー・マネージャーでの回復の時点

IBM MQ は、CF 構造のバックアップに関連した情報を Db2 内の表に記録します。 要件に応じて、IBM MQ Db2 のリカバリーのポイント・イン・タイムを調整したい場合があります。そうしないと、BACKUP

CFSTRUCT コマンドが終了した後で、IBM MQ CSQ.ADMIN_B_STRBACKUP テーブルのコピーを取るのに十分である可能性があります。

回復の準備を次のように行います。

1. キュー共有グループ内の各キュー・マネージャーのページ・セットのバックアップを作成します。
2. RECOVER(YES) 属性を持つ各 CF 構造に対して BACKUP CFSTRUCT コマンドを実行します。 このコマンドは単一のキュー・マネージャーから実行できますが、 負荷のバランスをとるためにキュー共有グループ内の別の複数のキュー・マネージャーから実行することもできます。
3. すべてのバックアップが完了したら、ARCHIVE LOG コマンドを実行してアクティブ・ログを切り替えて、 キュー共有グループ内の各キュー・マネージャーのログと BSDS のコピーを作成します。
4. キュー共有グループ内のすべてのキュー・マネージャーのページ・セットのバックアップ、アーカイブ・ログ、アーカイブ BSDS、および選択した Db2 バックアップ情報をオフサイトに移します。

代替サイトでのキュー共有グループの回復

キュー共有グループを回復する前に、次のように環境を準備する必要があります。

1. カップリング・ファシリティーの情報がキュー共有グループをインストールしたときに行った始動時のままで古い場合は、 まずこれを次のように除去します。

   注 : カップリング・ファシリティーに古い情報がない場合は、このステップを省略してもかまいません。

   a. 次の z/OS コマンドを入力して、そのキュー共有グループの CF 構造を表示します。

   ```
   D XCF,STRUCTURE,STRNAME= qsgname
   ```

   b. キュー共有グループ名で始まるすべての構造に対して、次のように z/OS コマンド SETXCF FORCE CONNECTION を使って、それらの構造との接続を強制的に切断します。

   ```
   SETXCF FORCE,CONNECTION,STRNAME= strname,CONNAME=ALL
   ```

   c. それぞれの構造に対して次のコマンドを使って、すべての CF 構造を削除します。

   ```
   SETXCF FORCE,STRUCTURE,STRNAME= strname
   ```

2. Db2 システムとデータ共有グループを復元します。
3. CSQ.ADMIN_B_STRBACKUP 表を回復して、基本サイトで行った最新の構造のバックアップに関する情報を含めます。

   注 : STRBACKUP 表には最新の構造のバックアップ情報を含めることが重要です。 古い構造のバックアップ情報を使用すると、最近の DISPLAY USAGE TYPE(DATASET) コマンドで示された情報をもとに廃棄したデータ・セットが必要になることがあります。 この場合は回復した CF 構造に正確な情報が含まれないことになります。
4. キュー共有グループ内のそれぞれのキュー・マネージャーに対して、CSQ5PQSG ユーティリティーの ADD QMGR コマンドを実行します。 これにより、各キュー・マネージャーの XCF グループ項目が復元されます。

   このシナリオでユーティリティーを実行する場合、通常は以下のメッセージが表示されます。

```
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
```

キュー共有グループ内のキュー・マネージャーを回復するには、次のようにします。

1. 新しいページ・セットのデータ・セットを定義し、 1 次サイトからのページ・セットのコピーに入っているデータをそれらにロードします。

2. 新しいアクティブ・ログ・データ・セットを定義します。
3. 新しい BSDS データ・セットを定義し、アクセス方式サービス REPRO を使用して最新の保存 BSDS をその中にコピーします。
4. ログ・マップ印刷ユーティリティー CSQJU004 を使用して、 その最新の BSDS の情報を印刷します。 この BSDS が保存された時点での最新のアーカイブ・ログは、 アクティブ・ログとして切り捨てられており、アーカイブ・ログとしては現れません。 このログの STARTRBA、STARTLRSN、ENDRBA、 および ENDLRSN の値を記録します。
5. この最新のアーカイブ・ログ・データ・セットを、 ログ目録変更ユーティリティー CSQJU003 を使って、 復元した BSDS に登録します。 このとき、ステップ 505 ページの『4』 で記録した値を使用します。
6. CSQJU003 の DELETE オプションを使用して、BSDS からすべてのアクティブ・ログ情報を除去します。
7. CSQJU003 の NEWLOG オプションを使用して、BSDS にアクティブ・ログを追加します。 このときは STARTRBA または ENDRBA を指定しないでください。
8. キュー共有グループの *recoverylrsn* を計算します。 *recoverylrsn* は、キュー共有グループ内のすべてのキュー・マネージャーにおける最も低い ENDLRSN (ステップ 505 ページの『4』 で記録したもの) から 1 を引いた値です。 例えば、キュー共有グループ内に 2 つのキュー・マネージャーがあり、そのうちの 1 つの ENDLRSN が B713 3C72 22C5 であり、もう 1 つのキュー・マネージャーが B713 3D45 2123 である場合、 *recoverylrsn* は B713 3C72 22C4 です。
9. CSQJU003 を使用して、再始動制御レコードを BSDS に追加します。 次を指定します。

   ```
   CRESTART CREATE,ENDLRSN= recoverylrsn
   ```

   ここで、 *recoverylrsn* はステップ 505 ページの『8』 で記録した値です。

   この時点で BSDS にはすべてのアクティブ・ログ (空の状態) および使用可能なすべてのアーカイブ・ログが記述され、 それらのログの末尾を超えたチェックポイントは記述されません。

   キュー共有グループ内の各キュー・マネージャーの BSDS に、CRESTART レコードを追加する必要があります。
10. START QMGR コマンドを使用して、キュー共有グループ内の各キュー・マネージャーを再始動します。 初期設定時に、次のようなオペレーター応答メッセージが出されます。

    ```
    CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
    REPLY Y TO CONTINUE, N TO CANCEL
    ```

    Y と応答してキュー・マネージャーを開始してください。 キュー・マネージャーが開始し、CRESTART ステートメントで指定された ENDRBA までのデータが回復されます。

    IBM WebSphere MQ 7.0.1 以降では、開始される最初のキュー・マネージャーは、それ自身およびキュー共有グループの他のメンバーのために管理構造の区画を再作成できます。 この段階でキュー共有グループ内のそれぞれのキュー・マネージャーを再始動する必要はなくなりました。
11. すべてのキュー・マネージャーに関する管理構造データが再作成されたら、それぞれの CF アプリケーション構造に対して RECOVER CFSTRUCT コマンドを発行してください。

    単一のキュー・マネージャーですべての構造に対して RECOVER CFSTRUCT コマンドを実行する場合、ログのマージ処理が行われるのは 1 回だけです。 別々のキュー・マネージャーで各 CF 構造に対してコマンドを実行すると、 各キュー・マネージャーでログのマージのステップを実行する必要があるため、 単一のキュー・マネージャーで実行する場合のほうが高速になります。

キュー共有グループで条件付き再始動処理が使われる場合、ピアの管理の再作成を実行する IBM WebSphere MQ 7.0.1 以降のキュー・マネージャーは、自身の CRESTART LRSN と同じものがピアの BSDS に含まれるかどうかを検査します。 これは、再作成された管理構造の整合性を確認するために行われます。 したがって、グループのいずれかのメンバーが次回に無条件で再始動される前に QSG 内の他のピアが自身の CRESTART 情報を処理できるよう、他のピアを再始動することは重要です。

ディスク・ミラーリングの使用

代替サイトでデータ・セットを同期的にコピーするために、IBM Metro Mirror (旧名 PPRC) などのディスク・ミラーリング・テクノロジーを使用するインストール済み環境が多くなりました。 このような場合、代替サイトの IBM MQ ページ・セットおよびログは基本サイトと実質的に同じであるため、詳しく説明した手順のほとんどは不要になります。 このようなテクノロジーを使用する場合、代替サイトでキュー共有グループを再始動するための手順を要約すると、次のようになります。

- 代替サイトの IBM MQ CF 構造を消去します。 (多くの場合、これらには、以前の災害復旧演習の残余情報が含まれます。)
- IBM MQ キュー共有グループによって使用されるデータベース内の Db2 システムおよびすべての表をリストアします。
- キュー・マネージャーを再始動します。 IBM WebSphere MQ 7.0.1 より前では、キュー共有グループに定義されたそれぞれのキュー・マネージャーを再始動する必要があります。各キュー・マネージャーは、キュー・マネージャー再始動時に自身の管理構造の区画を回復するためです。 各キュー・マネージャーが再始動された後、ホーム LPAR にないキュー・マネージャーを再びシャットダウンできます。 IBM WebSphere MQ 7.0.1 以降では、開始される最初のキュー・マネージャーは、それ自身およびキュー共有グループの他のメンバーのために管理構造の区画を再作成します。キュー共有グループ内のそれぞれのキュー・マネージャーを再始動する必要はなくなりました。
- 管理構造が再作成された後、アプリケーション構造を回復します。

V 9.2.0 IBM MQ 9.1.2 以降では、Metro Mirror を使用してミラーリングされるアクティブ・ログへの書き込み時に zHyperWrite の使用がサポートされます。 zHyperWrite は、Metro Mirror の使用によるパフォーマンスへの影響の軽減に役立ちます。詳しくは、IBM MQ での Metro Mirror の使用を参照してください。

z/OS キュー・マネージャーの再初期設定

キュー・マネージャーが異常終了すると、再始動できない場合があります。 その原因としては、 使用しているページ・セットまたはログが失われたり切り捨てられたり破壊されたりしたことが考えられます。 そのような場合には、 キュー・マネージャーの再初期設定 (コールド・スタートの実行) が必要になることがあります。

注意

コールド・スタートを実行するのは、 それ以外の方法ではキュー・マネージャーを再始動できない場合だけにしてください。 コールド・スタートの実行によりキュー・マネージャーとオブジェクト定義は回復できますが、 メッセージ・データは**回復できません**。 このトピックで説明されているその他の再始動シナリオによって解決しないかどうか、まず確認するようにしてください。

再始動に成功すると、すべての IBM MQ オブジェクトが定義されて使用可能になりますが、メッセージ・データは存在しません。

注 : クラスターの一部になっているキュー・マネージャーは、 再初期設定しないでください。 まずクラスターからキュー・マネージャーを除去し (クラスター内の他のキュー・マネージャーで RESET CLUSTER コマンドを使用)、 次にそのキュー・マネージャーを再初期設定し、 最後にそのキュー・マネージャーを新規のキュー・マネージャーとしてクラスターに再導入する必要があります。

これは、再初期設定時にキュー・マネージャー ID (QMID) が変更され、 古いキュー・マネージャー ID を持つクラスター・オブジェクトがクラスターから除去されてしまうためです。

詳細は、以下のセクションを参照してください。

- キュー共有グループに属していないキュー・マネージャーの再初期設定
- キュー共有グループに属するキュー・マネージャーの再初期設定

キュー共用グループに属していないキュー・マネージャーの再初期設定

キュー・マネージャーを再初期設定するには、以下の手順に従ってください。

1. キュー・マネージャーの再始動時に使用するオブジェクト定義ステートメントを準備します。 そのためには、次のどちらかを実行します。
   - ページ・セット 0 が使用可能な場合は、CSQUTIL SDEFS 機能を使用します (IBM MQ 定義コマンドのリストの生成を参照)。 すべてのオブジェクト・タイプ (認証情報オブジェクト、CF 構造、チャネル、名前リスト、プロセス、キュー、およびストレージ・クラス) の定義を入手する必要があります。
   - ページ・セット 0 が使用可能でない場合は、 最後にオブジェクト定義をバックアップした時点での定義を使用します。
2. キュー・マネージャー・データ・セットを再定義します (ステップ 507 ページの『1』 が完了するまでこの作業は実行しないでください)。

   詳細については、ブートストラップ・データ・セットとログ・データ・セットの作成とページ・セットの定義を参照してください。
3. 新しく定義して初期設定したログ・データ・セット、BSDS、およびページ・セットを使用してキュー・マネージャーを再始動します。 ステップ 507 ページの『1』 で作成したオブジェクト定義入力ステートメントを、 CSQINP2 初期設定入力データ・セットの入力として使用してください。

キュー共有グループに属するキュー・マネージャーの再初期設定

キュー共有グループの場合のキュー・マネージャーの再初期設定は複雑になります。 その場合、ページ・セットまたはログの問題のために、1 つまたは複数のキュー・マネージャーを再初期設定することが必要になる可能性があり、さらに Db2 またはカップリング・ファシリティーに関する問題を処理しなければならない可能性もあります。 そのため、いくつかの方法があります。

コールド・スタート
キュー共有グループ全体の再初期設定には、カップリング・ファシリティーの全構造をリセットする (FORCE オプションを使用) こと、キュー共有グループの全オブジェクト定義を Db2 から消去すること、ログと BSDS を削除または再定義すること、そしてキュー共有グループ内の全キュー・マネージャーのページ・セットを書式設定することが関係しています。

共有定義を保存する方法
ログと BSDS を削除または再定義し、 キュー共有グループ内の全キュー・マネージャーのページ・セットを書式設定し、 カップリング・ファシリティーの全構造をリセットします (FORCE オプションを使用)。 再始動時には、すべてのメッセージが削除されています。 キュー・マネージャーは、Db2 データベース内にまだ存在している GROUP オブジェクトに対応する COPY オブジェクトを再作成します。 共有キューがまだ存在していれば使用することができます。

単一のキュー・マネージャーを再初期設定する方法
ログと BSDS を削除または再定義し、単一のキュー・マネージャーのページ・セットを書式設定します (それにより、それ専用のオブジェクトとメッセージはすべて削除されます)。 キュー・マネージャーは再始動時に、Db2 データベース内にまだ存在している GROUP オブジェクトに対応する COPY オブジェクトを再作成します。 共有キューがまだ存在していてそこにメッセージが存在していれば、それらを使用することができます。

キュー共有グループの特定時点回復
これは、代替サイト災害時回復シナリオです。

共有オブジェクトは、Db2 回復で可能な時点まで回復されます (Db2 システムの障害を参照)。 各キュー・マネージャーは、 代替サイトで利用可能なバックアップ・コピーに基づいて可能な時点まで回復されます。

持続メッセージはキュー共有グループ内で使用することが可能で、MQSC RECOVER CFSTRUCT コマンドを 使って回復することができます。 このコマンドは、障害の起きた時刻まで 復旧します。 ただし共有キューの非持続メッセージは回復できません。 それらは、CSQUTIL ユーティリティー・プログラムの COPY 機能を使って別個にバックアップ・コピーを作成しておかない限り、すべて失われてしまいます。

各キュー・マネージャーを同じ時点に復元する必要はありません。異なるキュー・マネージャー上のローカル・オブジェクト (実際に回復されるもの) 間に依存関係がなく、再始動時のキュー・マネージャーの Db2 との再同期化で、各キュー・マネージャーごとに必要に応じて COPY オブジェクトが作成または削除されるためです。

z/OS z/OS 自動再始動管理プログラム (ARM) の使用

このトピックでは、ARM を使用してキュー・マネージャーを自動的に再始動する方法について知ることができます。

このセクションでは、以下のトピックに関する情報を取り上げます。

- 508 ページの『ARM とは?』
- 508 ページの『ARM ポリシー』
- 510 ページの『IBM MQ ネットワークでの ARM の使用』

ARM とは?

z/OS 自動再始動マネージャー (ARM) は、キュー・マネージャーの可用性を改良できる z/OS 回復機能です。ジョブまたはタスクが失敗するか、または稼働しているシステムに障害が発生した場合、 ARM によりオペレーターが介入することなくジョブまたはタスクを再始動することができます。

キュー・マネージャーまたはチャネル・イニシエーターに障害が起きた場合、ARM は同じ z/OS イメージ上でそれを再始動します。 z/OS および関連するサブシステムやアプリケーションのグループ全体が失敗した場合、ARM は、事前定義順にシスプレックス内の別の z/OS イメージ上で、失敗したすべてのシステムを自動的に再始動することができます。 これはシステム間再始動 と呼ばれます。

ARM を使用してチャネル・イニシエーターを再始動するのは、例外的な状況の場合のみです。 キュー・マネージャーが ARM によって再始動される場合、チャネル・イニシエーターを CSQINP2 初期設定データ・セットから再始動してください (510 ページの『IBM MQ ネットワークでの ARM の使用』を参照)。

z/OS の障害時には、ARM を使用して、シスプレックス内の別の z/OS イメージでキュー・マネージャーを再始動することができます。 別の z/OS イメージでの IBM MQ ARM 再始動のネットワークへの影響については、 510 ページの『IBM MQ ネットワークでの ARM の使用』で説明しています。

自動再始動を使用可能にするには、次のようにします。

- ARM カップル・データ・セットをセットアップします。
- z/OS で実行する自動再始動アクションを *ARM* ポリシー に定義します。
- ARM ポリシーを開始する。

また、始動時に IBM MQ は ARM を登録する必要があります (これは自動的に実行されます)。

注 : 別の z/OS イメージでキュー・マネージャーを自動的に再始動する場合、キュー・マネージャーが再始動する可能性がある各 z/OS イメージごとにすべてのキュー・マネージャーを、シスプレックス内で固有の 4 文字のサブシステム名を使ってサブシステムとして定義する必要があります。

ARM カップル・データ・セット

ARM をサポートするキュー・マネージャーを開始するには、 その前に、ARM に必要なカップル・データ・セットを定義し、 それらをオンラインにして活動状態にしておく必要があります。 キュー・マネージャーの始動時にカップル・データ・セットが利用不能である場合、IBM MQ 自動 ARM 登録は失敗します。 その場合、カップル・データ・セットがないため、IBM MQ は ARM がサポートされていないと解釈し、初期設定は続行されます。

ARM 結合データ・セットについては、「*z/OS MVS* シスプレックスのセットアップ 」を参照してください。

z/OS *ARM* ポリシー

自動リスタート・マネージャー・ポリシーは、キュー・マネージャーのどんな再始動も制御できる ARM 機能を制御するためのユーザー定義のルールです。

ARM の機能は、ユーザー定義の *ARM* ポリシー により制御されます。 ARM により再始動されるキュー・マネージャー・インスタンスを実行している各 z/OS イメージは、活動状態の ARM ポリシーのある ARM カップル・データ・セットに接続されていなければなりません。

IBM では、デフォルトの ARM ポリシーを提供しています。 新しいポリシーを定義することも、 管理データ・ユーティリティー (IXCMIAPU) (z/OS に付属) を使用してポリシーのデフォルトをオーバーライドすることもできます。 「*z/OS MVS* シスプレックスのセットアップ 」には、このユーティリティーについての説明と、ARM ポリシーの定義方法の詳細が記載されています。

509 ページの図 44 に、ARM ポリシーの例を示します。 このサンプル・ポリシーでは、キュー・マネージャーが失敗した場合または全体システムの障害が発生した場合に、 シスプレックス内でキュー・マネージャーを再始動します。

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

図 *44. ARM* ポリシーのサンプル

詳しくは、以下を参照してください。

- ARM ポリシーの定義
- ARM ポリシーのアクティブ化
- ARM への登録

ARM ポリシーの定義

次のように ARM ポリシーをセットアップします。

- キュー・マネージャー・インスタンスごとに RESTART_GROUP を定義します。それには、キュー・マネージャー・インスタンスに接続している CICS または IMS サブシステムも含まれます。 サブシステムの命名規則を使用している場合は、「?」 およびエレメント名の「*」ワイルドカード文字を使用して、RESTART_GROUP を最小限の定義作業で定義します。
- チャネル・イニシエーターに TERMTYPE (ELEMTERM) を指定して、チャネル・イニシエーターに障害が発生し、 z/OS イメージに障害が発生していない場合にのみ再始動するように指示します。
- キュー・マネージャーが失敗した場合、または z/OS イメージに障害が発生した場合に再始動するよう指示するために、キュー・マネージャーに TERMTYPE(ALLTERM) を指定します。
- キュー・マネージャーとチャネル・イニシエーターの両方に RESTART_METHOD(BOTH, PERSIST) を指定します。 これは、最後の始動時に保存された JCL を (システム記号を解決してから) 使用して再始動するよう ARM に指示するものです。 個々のエレメントに障害が発生したのか、それとも z/OS イメージに障害が発生したのかには関係なく、ARM に対して上記の動作を指示します。
- 他のすべての ARM ポリシー・オプションについては、デフォルトの値を受け入れます。

ARM ポリシーのアクティブ化

自動再始動管理ポリシーを開始するには、次の z/OS コマンドを実行します。

```
SETXCF START,POLICY,TYPE=ARM,POLNAME= mypol
```

ポリシーを開始すると、ARM カップル・データ・セットに接続されているすべてのシステムは 同じ活動状態のポリシーを使用します。

自動再始動を使用不可にするには、SETXCF STOP コマンドを使います。

ARM への登録

IBM MQ は、(ARM が使用可能であれば) キュー・マネージャーの始動時に *ARM* エレメント として自動的に登録されます。 終了時には、登録解除しないよう指定されているのでない限り登録解除されます。

始動時に、キュー・マネージャーは ARM を使用できるかどうかを判別します。 使用できる場合、IBM MQ は SYSMQMGR*ssid* という名前を使用して登録されます (*ssid* は 4 文字のキュー・マネージャー名、SYSMQMGR はエレメント・タイプ)。

始動時に ARM に登録された場合、 STOP QMGR MODE(QUIESCE) コマンドと STOP QMGR MODE(FORCE) コマンドを実行することによって、 ARM からキュー・マネージャーを登録解除できます。 これにより、ARM はこのキュー・マネージャーを再始動しなくなります。 STOP QMGR MODE(RESTART) コマンドを使用してもキュー・マネージャーは ARM から登録解除されないため、このコマンドは即時自動再始動に適しています。

チャネル・イニシエーターの各アドレス・スペースは ARM が使用可能であるかどうかを判別し、 使用可能であれば、 SYSMQCH*ssid* というエレメント名を登録します (*ssid* はキュー・マネージャー名、 SYSMQCH はエレメント・タイプ)。

チャネル・イニシエーターが正常に停止した場合、それは必ず ARM から登録解除されます。 登録されたままになるのは、それが異常終了する場合だけです。 キュー・マネージャーが失敗した場合、チャネル・イニシエーターは必ず登録解除されます。

z/OS *IBM MQ ネットワークでの ARM の使用*

キュー・マネージャーをセットアップして、その再始動時にチャネル・イニシエーターとそれに関連するリスナーが自動的に開始されるようにすることができます。

LU 6.2 と TCP/IP の 2 つの通信プロトコルの両方について、同じ z/OS イメージ上でキュー・マネージャーを完全に自動再始動するには、次のようにします。

- 適切な START LISTENER コマンドを CSQINPX データ・セットに追加することにより、 リスナーを自動的に開始します。
- 適切な START CHINIT コマンドを CSQINP2 データ・セットに追加することにより、チャネル・イニシエーターを自動的に開始します。

TCP/IP または LU6.2 でのキュー・マネージャーの再始動については、以下を参照してください。

- 510 ページの『別の z/OS イメージでの再始動 - TCP/IP の場合』
- 512 ページの『別の z/OS イメージでの再始動 - LU 6.2 の場合』

CSQINP2 データ・セットおよび CSQINPX データ・セットについては、タスク 13: 初期設定入力データ・セットをカスタマイズするを参照してください。

別の z/OS イメージでの再始動 - TCP/IP の場合

通信プロトコルとして TCP/IP を使用して仮想 IP アドレスを使っている場合は、それらが他の z/OS イメージ上で回復するように構成することによって、そのキュー・マネージャーに接続しているチャネルを変更せずに再接続することができます。 別の方法として、キュー・マネージャーを別の z/OS イメージに移動した後で TCP/IP アドレスを割り振りし直すこともできますが、これができるのは、クラスターを使っている場合か、または WLM 動的ドメイン・ネーム・システム (DNS) の論理グループ名を使ってキュー共有グループに接続している場合だけです。

- クラスターを使っている場合
- キュー共有グループに接続する場合

クラスターを使っている場合

z/OS ARM は、同じシスプレックス内にある別の z/OS イメージ上でキュー・マネージャーを再始動することによって、システム障害に対応します。そのシステムの TCP/IP アドレスは、元の z/OS イメージに対して異なるものになります。 以下では、キュー・マネージャーが ARM 再始動によって別の z/OS イメージに移動された後に、 IBM MQ クラスターを使用してキュー・マネージャーの TCP/IP アドレスを再割り当てする方法について説明します。

クライアント・キュー・マネージャーはキュー・マネージャーの障害を (チャネル障害として) 検出すると、 そのクラスター伝送キュー上の該当するメッセージを、 宛先クラスター・キューの別のインスタンスを制御する別のサーバー・キュー・マネージャーに割り振りし直して対応します。 しかし、類似性の制約によって元のサーバーにバインドされているメッセージ、 またはバッチ終了処理中にサーバー・キュー・マネージャーに障害が発生したために未確定状態になっているメッセージについては、クライアント・キュー・マネージャーが割り振りし直すことはできません。 これらのメッセージを処理するには、次のことを実行します。

1. z/OS キュー・マネージャーごとに、異なるクラスター - 受信側チャネル名と、異なる TCP/IP ポートを割り振ります。 z/OS イメージ上の 1 つの TCP/IP スタックを 2 つのシステムが共有できるようにするため、各キュー・マネージャーごとに異なるポートが必要です。 それら 2 つのキュー・マネージャーのうちの 1 つはその z/OS イメージ上で最初から稼働しているキュー・マネージャーであり、もう 1 つはシステム障害後に ARM がその z/OS イメージ上で再始動するキュー・マネージャーです。 z/OS イメージごとに各ポートを構成してください。それにより ARM は、どの z/OS イメージ上のどのキュー・マネージャーでも再始動できるようになります。
2. キュー・マネージャーと z/OS イメージの組み合わせごとに、チャネル・イニシエーターの始動時に参照されるチャネル・イニシエーター・コマンド入力ファイル (CSQINPX) としてそれぞれ異なるものを作成します。

   各 CSQINPX ファイルには、そのキュー・マネージャーに固有の START LISTENER PORT(port) コマンド、およびそのキュー・マネージャーと z/OS イメージの組み合わせに固有のクラスター - 受信側チャネルのための ALTER CHANNEL コマンドが含まれている必要があります。 ALTER CHANNEL コマンドの接続名は、それを再始動する z/OS イメージの TCP/IP 名に設定する必要があります。 このコマンドには、再始動されるキュー・マネージャーに固有のポート番号が、 接続名の一部として含まれていなければなりません。

   各キュー・マネージャーの開始 JCL には、この CSQINPX ファイルに対して固定データ・セット名を指定できます。また z/OS イメージごとに、各 CSQINPX ファイルのそれぞれの各種バージョンが非共有 DASD ボリューム上になければなりません。

ARM の再始動が行われると、IBM MQ は、変更されたチャネル定義をクラスター・リポジトリーに通知します。次にこのリポジトリーは、サーバー・キュー・マネージャーに関心を示しているすべてのクライアント・キュー・マネージャーに対して、このチャネル定義を公開します。

クライアント・キュー・マネージャーは、 サーバー・キュー・マネージャーの障害をチャネル障害として扱い、 障害の発生したチャネルを再始動しようとします。 クライアント・キュー・マネージャーが新しいサーバー接続名を知ると、 チャネル再始動により、 再始動されるサーバー・キュー・マネージャーにクライアント・キュー・マネージャーが再接続されます。 その後、クライアント・キュー・マネージャーは、 そのメッセージの再同期処理を実行し、 そのクライアント・キュー・マネージャーの伝送キュー上にある未確定メッセージをすべて解決できます。 その結果、正常な処理を続行できます。

キュー共有グループに接続する場合

TCP/IP 動的ドメイン・ネーム・システム (DNS) の論理グループ名を使ってキュー共有グループに接続する場合、チャネル定義内の接続名には、物理マシンのホスト名や IP アドレスではなく、 キュー共有グループの論理グループ名を指定します。 そのチャネルが開始すると、それは動的 DNS に接続した後、キュー共有グループの中のキュー・マネージャーの 1 つに接続します。 このプロセスについては、キュー共有グループを使用する IBM MQ for z/OS の通信のセットアップで説明しています。

例外的なイメージ障害においては、下記のいずれかが発生します。

- 障害の発生したイメージ上のキュー・マネージャーは、シスプレックス上で実行されている動的 DNS から登録解除されます。 この接続障害に対してチャネルは、RETRYING 状態になることによって応答

します。 その後、チャネルはシスプレックス上で実行されている動的 DNS に接続します。 動的 DNS は、残りのイメージ上でまだ実行されているキュー共有グループの残りのメンバーのうちの 1 つに対して、 インバウンド要求を割り振ります。

- キュー共有グループ内のその他のキュー・マネージャーが活動状態でなく、ARM が キュー・マネージャーとチャネル・イニシエーターを別のイメージ上で再始動する場合、 グループ・リスナーはその新しいイメージから動的 DNS に登録されます。 つまり、論理グループ名 (チャネルの接続名フィールドに指定されたもの) が動的 DNS に接続され、そしてその時点で別のイメージで実行されている同じキュー・マネージャーにそれが接続されます。 この場合、チャネル定義を変更する必要はありません。

このようなタイプの回復処理が実行されるためには、 下記の点に注意する必要があります。

- z/OS において動的 DNS は、シスプレックス内の z/OS イメージのいずれかで実行されます。 そのイメージに障害が発生するという場合には、シスプレックス内で 2 次ネーム・サーバーが 1 次ネーム・サーバーの 代替機能として活動状態になっているように、動的 DNS を構成しておく必要があります。 1 次および 2 次動的 DNS サーバーについては、「*OS/390® SecureWay CS IP* 構成」資料を参照してください。
- TCP/IP グループ・リスナーが、この z/OS イメージ上で使用できない特定の IP アドレスで開始されたものである可能性があります。 その場合、そのリスナーは新しいイメージ上の別の IP アドレスで開始しなければならないことがあります。 仮想 IP アドレスを使用している場合、他の z/OS イメージ上で回復するようにそれらを構成することによって、START LISTENER コマンドを変更する必要がないようにすることができます。

別の z/OS イメージでの再始動 - LU 6.2 の場合

LU 6.2 通信プロトコルだけを使用する場合は、シスプレックス内の別の z/OS イメージ上でキュー・マネージャーが自動再始動した後でネットワークの再接続を可能にするために、以下の手順を実行する必要があります。

- 固有のサブシステム名を使用して、 シスプレックス内に各キュー・マネージャーを定義します。
- 固有の LUNAME を使用して、シスプレックス内に各チャネル・イニシエーターを定義します。 これは、キュー・マネージャー属性および START LISTENER コマンドで指定されます。

  注 : LUNAME は APPC サイド表の中の項目の名前となり、 その項目が実際の LUNAME にマッピングされることになります。
- シスプレックス内の各 z/OS イメージにより参照される共有 APPC サイド表をセットアップします。 これには、チャネル・イニシエーターの LUNAME ごとに項目が 1 つずつ含まれている必要があります。 これについては、「*z/OS MVS* 計画*: APPC/MVS* 管理」を参照してください。
- シスプレックス内のチャネル・イニシエーターごとに SYS1.PARMLIB の APPCPM *xx* メンバーをセットアップして LUADD を含めるようにすることによって、 そのチャネル・イニシエーターの APPC サイド表項目をアクティブ化します。 それらのメンバーは、各 z/OS イメージによって共有されていなければなりません。 該当する SYS1.PARMLIB メンバーは、z/OS コマンド SET APPC= *xx* によって活動化されます。これは、以下のテキストで説明されているように、別の z/OS イメージ上のキュー・マネージャー (およびそのチャネル・イニシエーター) の ARM 再始動時に自動的に発行されます。
- LU62ARM キュー・マネージャー属性を使用することによって、 チャネル・イニシエーターごとに、 この SYS1.PARMLIB メンバーの *xx* 接尾部を指定します。 これにより、LUNAME をアクティブ化するのに必要な z/OS コマンド SET APPC=*xx* がチャネル・イニシエーターによって実行されるようになります。

チャネル・イニシエーターがその z/OS イメージが正常なときに失敗した場合だけチャネル・イニシエーターを再始動するように ARM ポリシーを定義します。XCFAS アドレス・スペースに関連したユーザー ID には、IBM MQ コマンド START CHINIT を実行できる許可が付与されなければなりません。 z/OS イメージにも障害が発生した場合は、チャネル・イニシエーターを自動的に再始動しないでください。その場合は、CSQINP2 データ・セットとおよび CSQINPX データ・セットでコマンドを使用して、チャネル・イニシエーターとリスナーを開始します。

z/OS 作業単位の手動回復

作業単位 CICS、IMS、RRS、またはキュー共有グループ内の他のキュー・マネージャーを手動で回復することができます。 キュー・マネージャー・コマンドを使用して、キュー・マネージャーへの各接続に関連した作業単位の状況を表示することができます。

このトピックでは、以下の点に関する情報を取り上げます。

- 513 ページの『接続とスレッドの表示』
- 513 ページの『CICS リカバリー単位の手動回復』
- 517 ページの『IMS リカバリー単位の手動回復』
- 519 ページの『RRS リカバリー単位の手動回復』
- 519 ページの『キュー共有グループ中の別のキュー・マネージャーにおけるリカバリー単位のリカバリー』

接続とスレッドの表示

DISPLAY CONN コマンドを使用すると、 キュー・マネージャーとそれに関連する作業単位への接続に関する情報を得ることができます。 活動状態の作業単位を表示すれば、現在発生していることを調べたり、 キュー・マネージャーのシャットダウンの前に何を終了する必要があるかを調べたりできます。 また、回復処理のために未解決の作業単位を表示させることもできます。

活動状態の作業単位

活動状態の作業単位だけを表示するには、次のコマンドを使用します。

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ ACTIVE)
```

未解決の作業単位

未解決の作業単位 (「未確定スレッド」とも呼ばれる) は、2 フェーズ・コミット操作の 2 番目の パスにある作業単位です。 資源は IBM MQ に保持されます。 未解決の作業単位を表示するには、次のコマンドを使用します。

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

未解決の作業単位の状況を解決するには、外部からの 介入が必要です。 これは、以降の節に説明があるように、回復調整システム (CICS、IMS、または RRS) を開始するだけの場合と、それ以上のことが関連する場合があります。

z/OS *CICS* リカバリー単位の手動回復

このトピックでは、CICS アダプターの再始動時に何が実行されるか、そして、未解決リカバリー単位があった場合にそれをどう処理したらよいかについて知ることができます。

CICS アダプター再始動時の処理

接続が切れた場合、 アダプターは再接続処理 において再始動フェーズ にならなければなりません。 再始動フェーズは、資源を再同期化します。 CICS と IBM MQ との再同期化によって、未確定作業単位が識別され解決されます。

再同期は、次の要求によって発生します。

- 分散キューイング・コンポーネントからの明示的な要求
- IBM MQ への接続時の暗黙の要求

IBM MQ への接続によって再同期が発生した場合、下記の順序でイベントが発生します。

1. 接続処理は、IBM MQ から、未確定の作業単位 (UOW) ID のリストを取得します。
2. UOW ID がコンソール上で CSQC313I メッセージの中に表示されます。
3. UOW ID が CICS へ渡されます。
4. CICS が、各未確定 UOW ID ごとに再同期タスク (CRSY) を開始します。
5. 各未確定 UOW ごとのタスクの結果がコンソールに表示されます。

接続処理中に表示される下記のメッセージを見る必要があります。

CSQC313I
UOW が未確定であることを示しています。

CSQC400I
UOW を識別し、下記のメッセージのうちの 1 つがその後に表示されます。

- CSQC402I または CSQC403I は、UOW が正常に解決 (コミットまたはバックアウト) されたことを示しています。
- CSQC404E、CSQC405E、CSQC406E、または CSQC407E は、UOW が解決されなかったことを示しています。

CSQC409I
すべての UOW が正常に解決されたことを示しています。

CSQC408I
正常に解決されなかった UOW があることを示しています。

CSQC314I
* で強調表示されている UOW ID は自動的に解決されないことを警告します。 これらの UOW は、分散キューイング・コンポーネントの再始動時に、 それによって明示的に解決される必要があります。

514 ページの図 45 は、 z/OS コンソールに表示される一連の再始動メッセージの例を示しています。

```
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
```

図 *45.* 再始動メッセージの例

CSQC313I メッセージの合計数は、CSQC402I メッセージの数と CSQC403I メッセージの数の合計です。 これらの合計が等しくない場合、接続処理で解決できなかった UOW があることになります。 解決できない UOW は、 CICS の問題 (例えば、コールド・スタート) または IBM MQ を使用するか、キューイングを分散することによって引き起こされます。 これらの問題が解決された場合は、接続をいったん切断して再接続することにより、 あらためて再同期化を開始できます。

あるいは、RESOLVE INDOUBT コマンドとメッセージ CSQC400I に示された UOW ID を使用することによって、 未解決の各 UOW を手動で解決する方法もあります。 次に、 CICS 内の リカバリー単位記述子 をクリーンアップするために、切断および接続を開始する必要があります。 UOW を手動で解決するには、UOW の正しい出力を知っておく必要があります。

未解決の UOW に関連付けられているすべてのメッセージは、 IBM MQ によってロックされ、バッチ、 TSO 、または CICS タスクはそれらにアクセスできない。

CICS に障害が発生し、緊急時再始動が必要な場合は、 CICS システムの GENERIC APPLID を変更 しないでください 。 もしそれを変更してから IBM MQ に再接続すると、IBM MQ とのデータの整合性は保証されません。 これは、 IBM MQ が CICS の新規インスタンスを別の CICS として扱うためです (APPLID が異なるためです)。 そのため、正しくない CICS ログに基づいて未確定の解決が実行されることになります。

CICS リカバリー単位の手動解決の方法

アダプターが異常終了すると CICS と IBM MQ は、異常終了がどのサブシステムによって引き起こされたかに応じて、動的に、または再始動時に未確定リストを作成します。

注 : 作業単位を表示するために DFH$INDB サンプル・プログラムを使用した場合、IBM MQ の UOW が常に正しく表示されるとは限りません。

CICS が IBM MQ に接続する時点で、解決されていないリカバリー単位が 1 つ以上存在する場合があります。

次のメッセージのうちの 1 つがコンソールに送られます。

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E
- CSQC408I

これらのメッセージの意味について詳しくは、CICS アダプターとブリッジのメッセージのメッセージを参照してください。

CICS は、接続開始時に解決されなかったリカバリー単位の詳細を保存しています。 その項目が IBM MQ が提供するリストに現れなくなると、その項目は除去されます。

CICS が解決できないリカバリー単位は、すべて IBM MQ コマンドを使用して手動で解決する必要があります。 この手動による手順がインストール・システム内で使用されることはほとんどありません。 手動操作が必要になるのは、操作上のエラーまたはソフトウェアの問題によって自動解決ができない場合だけです。 未確定の解決時に不整合が検出された場合は、 その不整合について必ず調査してください。

リカバリー単位を解決するには、次の手順に従ってください。

1. 次のコマンドを使用することにより、リカバリー単位のリストを IBM MQ から入手します。

```
+CSQ1  DISPLAY CONN( * ) WHERE(UOWSTATE EQ UNRESOLVED)
```

   以下のメッセージを受け取ります。

```
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
```

CICS 接続の場合の NID は、CICS のアプリケーション ID と、同期点ログ項目が書き込まれた時点で CICS によって提供される固有の番号とで構成されます。 この固有の番号は、同期点処理の時点で CICS システム・ログと IBM MQ ログの両方に書き込まれるレコードの中に保管されます。 その値は、CICS において回復トークン と呼ばれます。

2. CICS ログを走査して、特定のリカバリー単位に関連する項目を検索します。

   タスク関連のインストールの場合は、 回復トークン・フィールド (JCSRMTKN) がネットワーク ID から入手した値と等しい PREPARE レコードを検索します。 ネットワーク ID は、IBM MQ により DISPLAY CONN コマンド出力で提供されます。

   リカバリー単位に関する CICS ログの中の PREPARE レコードによって、CICS タスク番号が提供されます。 この CICS タスクに関するログ上の他の項目は、すべてこの番号を使用して見つけることができます。

   ログ走査においては、CICS ジャーナル印刷ユーティリティー DFHJUP を使用することができます。 このプログラムの使用の詳細については、「*CICS* 操作およびユーティリティーの手引き」を参照してください。

3. IBM MQ ログを走査して、特定のリカバリー単位に関連した NID を持つレコードを検索します。 次にそのレコードの URID を使用して、このリカバリー単位の残りのログ・レコードを入手します。

   IBM MQ ログの走査においては、このセッションの開始 RBA が IBM MQ の開始メッセージ CSQJ001I で提供されることに注意してください。

   この操作には、ログ・レコード印刷プログラム (CSQ1LOGP) を使用できます。

4. 必要なら、IBM MQ で未確定の解決を実行します。

   IBM MQ は、 IBM MQ RESOLVE INDOUBT コマンドを使用して、リカバリー単位のリカバリー処置を取るよう指示することができます。

   特定の *connection-name* に関連するすべてのスレッドを回復するには、 NID(*) オプションを使用します。

   このコマンドは、次のいずれかのメッセージを生成することにより、スレッドがコミットされたかバックアウトされたかを示します。

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

未確定の解決を実行する場合、CICS およびアダプターは、リカバリー単位をコミットまたはバックアウトするための IBM MQ へのコマンドを認識しません。その影響を受けるのは IBM MQ の資源だけだからです。 しかし CICS は、IBM MQ によって解決できなかった未確定スレッドについての詳細は保持しています。 提供されたリストが空の場合、または CICS に詳細情報があるリカバリー単位がそのリストに含まれていない場合、この情報は除去されます。

z/OS *IMS* リカバリー単位の手動回復

このトピックでは、IMS アダプターの再始動時に何が実行されるか、そして、未解決リカバリー単位があった場合にそれをどう処理したらよいかについて知ることができます。

IMS アダプター再始動時の処理

キュー・マネージャーの再始動または IMS /START SUBSYS コマンドの後に IBM MQ への接続が再始動されるたびに、 IMS は以下の再同期プロセスを開始します。

1. IMS は、未確定であると見なす作業単位 (UOW) ID のリストを、IBM MQ IMS アダプターに一度に 1 つずつ、コミットまたはバックアウトの解決パラメーターを付けて提供します。
2. IMS アダプターは解決要求を IBM MQ に渡し、結果を IMS に報告します。
3. すべての IMS 解決要求を処理した後、IMS アダプターは、IMS システムによって開始された、IBM MQ がまだ未確定のまま保持しているすべての UOW のリストを IBM MQ から取得します。 それらは、メッセージ CSQQ008I の中で IMS マスター端末に報告されます。

注 : UOW が未確定である間、関連する IBM MQ メッセージは IBM MQ によってロックされているため、どのアプリケーションもそのメッセージを入手することはできません。

IMS リカバリー単位の手動解決の方法

IMS が IBM MQ に接続すると、 IBM MQ には、解決されていない未確定のリカバリー単位が 1 つ以上ある場合があります。

IMS が解決しなかった未確定のリカバリー単位が IBM MQ にある場合は、 IMS マスター端末で次のメッセージが出されます。

```
CSQQ008I nn units of recovery are still in doubt in queue manager qmgr-name
```

このメッセージが出された場合、 IMS はコールド・スタートされたか、または未完了のログ・テープで開始されたかのいずれかです。 このメッセージは、ソフトウェア・エラーまたは他のサブシステムの障害のために、IBM MQ または IMS が異常終了した場合にも出されることがあります。

CSQQ008I メッセージを受け取った後は、次の状態になります。

- 接続は活動状態のままです。
- IMS アプリケーションは、引き続き IBM MQ の資源にアクセスできます。
- 一部の IBM MQ リソースは、閉鎖されたままの状態です。

未確定スレッドが解決されない場合、IMS メッセージ・キューの構築を開始することができます。 IMS キューがその容量いっぱいになると、IMS は終了します。 このようになった場合の難しさを認識していなければならず、未確定リカバリー単位が完全に解決されるまで IMS を監視する必要があります。

回復手順

IMS 作業単位を回復するには、次の手順を使用します。

1. /SWI OLDS を使用することによって IMS ログを強制的にクローズしてから、IMS ログを保存します。 ユーティリティー DFSERA10 を使って、以前の IMS ログ・テープからレコードを印刷します。タイプ X'3730' のログ・レコードはフェーズ 2 のコミット要求を示し、タイプ X'38' のログ・レ

コードは打ち切り要求を示します。 それぞれの従属領域内の最後のトランザクションについて、要求されたアクションを記録してください。

2. DL/I バッチ・ジョブを実行することによって、関係する PSB のうちコミット点に達していないもののそれぞれをバックアウトします。 トランザクションがまだ処理中であるため、この処理にはかなり時間がかかることがあります。 また、この処理はいくつかのレコードをロックするため、残りの処理および残りのメッセージ・キューに影響を与えることがあります。
3. 次のコマンドを使用することにより、IBM MQ から未確定リカバリー単位のリストを生成します。

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

以下のメッセージを受け取ります。

```
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)
```

IMS の場合の NID は、IMS 接続名と IMS が提供する固有な番号とで構成されます。 IMS においてこの値は、回復トークン と呼ばれます。 詳しくは、「*IMS* カストマイズの手引き」を参照してください。

4. DISPLAY THREAD メッセージ中に表示された NID (IMSID プラス OASN の 16 進数) と、 DFSERA10 出力中に示された OASN (4 バイトの 10 進数) とを比較します。 コミットするかバックアウトするかを決定します。
5. 以下のように、 RESOLVE INDOUBT コマンドを使用して、 IBM MQ で未確定の解決を実行します。

```
RESOLVE INDOUBT( connection-name )
ACTION(COMMIT|BACKOUT)
NID( network-id )
```

connection-name に関連するすべてのスレッドを回復するには、 NID(*) オプションを使用します。 このコマンドにより下記のうちのいずれかのメッセージが出され、 スレッドがコミットされたかバックアウトされたかが示されます。

```
CSQV414I  THREAD network-id COMMIT SCHEDULED
CSQV415I  THREAD network-id BACKOUT SCHEDULED
```

未確定の解決を実行する場合、IMS およびアダプターは、未確定のリカバリー単位をコミットまたはバックアウトするための IBM MQ へのコマンドを認識しません。その影響を受けるのは IBM MQ の資源だけだからです。

z/OS *RRS* リカバリー単位の手動回復

このトピックでは、未確定 RRS リカバリー単位が存在するかどうかを判別する方法、およびそれらのリカバリー単位を手動で解決する方法について知ることができます。

RRS が IBM MQ に接続する時点で、解決されていない未確定リカバリー単位が 1 つ以上 IBM MQ に存在する場合があります。 RRS が解決しなかった未確定リカバリー単位が IBM MQ に存在すると、次のメッセージの 1 つが z/OS コンソールに出されます。

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

未確定リカバリー単位に関する情報を表示するツール、およびそれを手動で解決するための手法が、IBM MQ と RRS の両方に提供されています。

IBM MQ では、DISPLAY CONN コマンドを使用して、未確定 IBM MQ スレッドに関する情報を表示します。このコマンドからの出力には、RRS を調整システムとする IBM MQ スレッドのための RRS リカバリー単位 ID が含まれます。 これは、リカバリー単位の結果を判別するために使用できます。

手動で IBM MQ 未確定スレッドを解決するには、RESOLVE INDOUBT コマンドを使います。 正しい決定を確認した後、このコマンドを使用してリカバリー単位のコミットまたはバックアウトのいずれかを実行することができます。

z/OS キュー共有グループ中の別のキュー・マネージャーにおけるリカバリー単位のリカバリー

このトピックを使用して、キュー共有グループ中の別のキュー・マネージャーのリカバリー単位を特定し、手動でリカバリーします。

キュー共有グループのメンバーであるキュー・マネージャーに障害が発生し、 それを再始動できない場合には、 そのグループ中の他のキュー・マネージャーにおいて対等なリカバリー処理を実行し、 それを引き継ぐようにすることができます。 しかし、 そのリカバリー単位の最後の属性指定は障害の発生したキュー・マネージャーでしか認識されていないため、 そのキュー・マネージャーには対等なリカバリーでは解決できないリカバリー単位が含まれている可能性があります。 それらのリカバリー単位は、そのキュー・マネージャーの再始動時には解決されることになりますが、それまでの間は未確定の状態です。

したがって、一部の資源 (例えばメッセージ) がロックされている可能性があり、 その場合には、 そのグループ内の他のキュー・マネージャーからそれの資源を使用することができなくなります。 このような状況では、DISPLAY THREAD コマンドを使うことによって、 非活動キュー・マネージャー上でそれらの作業単位を表示することができます。 グループ内の他のキュー・マネージャーからメッセージを利用できるようにするため、それらのリカバリー単位を手動で解決する場合には、RESOLVE INDOUBT コマンドを使用できます。

DISPLAY THREAD コマンドを実行することによって未確定のリカバリー単位を表示する 場合には、QMNAME キーワードを使うことによって、非活動キュー・マネージャーの名前を指定できます。 例えば、下記のコマンドを実行した場合、

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

次のメッセージを受け取ります。

```
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
```

指定したキュー・マネージャーが活動状態の場合は、IBM MQ は未確定スレッドに関する情報を戻さず、下記のメッセージを発行します。

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

手動で未確定スレッドを解決するには、IBM MQ コマンド RESOLVE INDOUBT を使います。 そのコマンドでは、QMNAME キーワードを使うことによって非活動キュー・マネージャーの名前を指定できます。

このコマンドを使うことによってリカバリー単位をコミットまたはバックアウトすることができます。 このコマンドで解決されるのは回復単位の共用部分だけです。ローカル・メッセージは影響を受けず、キュー・マネージャーが再始動するか CICS、IMS、または RRS バッチに再接続する時点までロックされています。

z/OS IBM MQ と IMS

IBM MQ-IMS アダプターおよび IBM MQ-IMS ブリッジは、IBM MQ が IMS と相互作用できるようにする 2 つのコンポーネントです。 これらのコンポーネントは、通常、IMS アダプターおよび IMS ブリッジと呼ばれます。

z/OS IMS アダプターの操作

このトピックでは、 IBM MQ を IMS システムに接続する IMS アダプターの操作方法について説明します。

注 : IMS アダプターでは、操作および制御パネルは使用していません。

このトピックには、次のセクションがあります。

- 520 ページの『IMS 接続の制御』
- 521 ページの『IMS 制御領域からの接続』
- 523 ページの『未確定リカバリー単位の表示』
- 524 ページの『IMS 従属領域の接続の制御』
- 527 ページの『IMS からの切断』
- 527 ページの『IMS トリガー・モニターの制御』

z/OS *IMS* 接続の制御

このトピックでは、 IBM MQ への接続を制御およびモニターする IMS オペレーター・コマンドについて説明します。

IMS は、IBM MQ との接続を制御およびモニターするために、次のオペレーター・コマンドを提供します。

/CHANGE SUBSYS
IMS から未確定リカバリー単位を削除します。

/DISPLAY OASN SUBSYS
未解決のリカバリー・エレメントを表示します。

/DISPLAY SUBSYS
接続状況およびスレッドの活動を表示します。

/START SUBSYS
IMS 制御領域をキュー・マネージャーに接続します。

/STOP SUBSYS
キュー・マネージャーから IMS を切断します。

/TRACE
IMS トレースを制御します。

これらのコマンドについて詳しくは、ご使用の IMS のレベルについて「*IMS/ESA®* オペレーター・リファレンス」を参照してください。

IMS コマンドの応答は、コマンドを実行した端末に送られます。 IMS コマンドを実行するための許可は、IMS セキュリティーに基づいて与えられます。

z/OS *IMS* 制御領域からの接続

このトピックでは、IMS から IBM MQ に接続するために使用可能なメカニズムについて知ることができます。

IMS は、IMS を使用するそれぞれのキュー・マネージャーに対して、制御領域から 1 つずつの接続を行います。 IMS は、以下のいずれかの方法で接続できるようになっている必要があります。

- 次の場合に、自動的に接続を行います。
  - コールド・スタートの初期設定時。
  - IMS がシャットダウンされたときに IBM MQ 接続がアクティブであった場合は、 IMS のウォーム・スタート。
- IMS コマンドに応答して接続を行います。

```
/START SUBSYS sysid
```

  ここで、*sysid* はキュー・マネージャー名です。

  このコマンドは、キュー・マネージャーが活動状態であるかどうかにかかわらず発行することができます。

この接続は、MQ API が最初にキュー・マネージャーを呼び出した後、初めて接続されます。 それまでは、IMS コマンド /DIS SUBSYS では 'NOT CONN' の状況が示されます。

IMS とキュー・マネージャーのどちらを先に開始するかは、重要ではありません。

キュー・マネージャーが STOP QMGR コマンドまたは IMS コマンド /STOP SUBSYS で停止されるか、または異常終了したときは、IMS は、キュー・マネージャーへの接続を自動的に再び可能にすることはできません。 したがって、IMS コマンド /START SUBSYS を使用して接続する必要があります。

次のような IMS コマンドがキュー・マネージャー・コンソール・ログに表示される場合:

```
MODIFY IMS*,SS*
```

IMS マスター・ログを調べて、 IBM MQ に IMS アダプター MODIFY コマンドを発行するための RACF 権限があることを確認してください。

アダプターの初期設定およびキュー・マネージャーへの接続

アダプターは、IMS 外部サブシステム接続機能を使用して、IMS の制御領域および従属領域にロードされる 1 組のモジュールです。

次の手順で、アダプターの初期設定およびキュー・マネージャーへの接続を行います。

1. IMS.PROCLIB からサブシステム・メンバー (SSM) を読み取ります。 選択する SSM は、IMS EXEC パラメーターです。 IMS を接続できる各キュー・マネージャーごとに、1 つの項目がメンバー内にあります。 各項目には、IBM MQ アダプターについての制御情報が含まれています。
2. IMS アダプターをロードします。

   注 : IMS は、SSM メンバー内で定義された個々の IBM MQ インスタンスごとに、アダプター・モジュールのコピーを 1 つずつロードします。
3. IBM MQ 用の外部サブシステム・タスクを接続します。
4. 接続名として CTL EXEC パラメーター (IMSID) を指定して、アダプターを実行します。

接続が、初期設定の一部である場合も、IMS コマンド /START SUBSYS の結果行われる場合も、処理は同じです。

IMS が接続を試みたときにキュー・マネージャーが活動状態であれば、次のメッセージが送られます。

- z/OS コンソールへの場合:

```
DFS3613I ESS TCB INITIALIZATION COMPLETE
```

- IMS マスター端末に送る場合:

```
CSQQ000I IMS/TM imsid connected to queue manager ssnm
```

IMS が接続を試みたときに キュー・マネージャーが活動状態でない場合は、アプリケーションが MQI 呼び出しを行うたびに、次のメッセージが IMS マスター端末に送られます。

```
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
```

IMS への接続の開始時、またはシステムの開始時に、DFS3607I メッセージを受け取った場合、これはキュー・マネージャーが使用不可能であることを示します。 これによって大量のメッセージが生成されないようにするには、 次のどちらかの操作を実行する必要があります。

1. 関連するキュー・マネージャーを開始する
2. 次の IMS コマンドを発行します。

   ```
   /STOP SUBSYS
   ```

   このようにすると、IMS はキュー・マネージャーへの接続を想定しません。

上記のいずれも実行しなかった場合は、ジョブが領域にスケジュールされるたび、 およびアプリケーションによってキュー・マネージャーへの接続要求が出されるたびに、 DFS3607I メッセージおよび関連する CSQQ001I メッセージが出されます。

スレッド接続

MPP または IFP 領域では、アプリケーション・プログラムが IBM MQ 呼び出しを行わない場合でも、最初のアプリケーション・プログラムがその領域にスケジュールされると、 IMS はスレッド接続を行います。 BMP 領域では、アプリケーションが最初に IBM MQ 呼び出し (MQCONN または MQCONNX) を行ったときに、スレッド接続が行われます。 このスレッドは、領域の持続期間中または接続が停止するまで保存されます。

メッセージ・ドリブンの領域およびメッセージ・ドリブンでない領域のどちらについても、 スレッドに関連する回復スレッド相互参照 ID (*Thread-xref*) は、次のとおりです。

```
PSTid + PSBname
```

ここで、

PSTid
区画仕様表の領域 ID

PSBname
プログラム仕様ブロック名

接続 ID を、固有の ID として IBM MQ コマンドの中で使用することができます。このようにすると、IBM MQ は、生成するすべてのオペレーター・メッセージに、これらの ID を自動的に挿入します。

z/OS 未確定リカバリー単位の表示

未確定のリカバリー単位を表示し、それらの復旧を試行できます。

このトピックでの未確定リカバリー単位のリストおよびリカバリーに使用される操作ステップは、 比較的簡単なケースのみです。 キュー・マネージャーが IMS に接続されているときに異常終了すると、IMS が作業をコミットしたりバックアウトしたりし、このコミットやバックアウトを IBM MQ が認識していないという事態もありえます。 キュー・マネージャーが再始動したとき、その作業は未確定 と呼ばれる状態です。 この場合、作業の状況についての判断を行う必要があります。

未確定リカバリー単位のリストを表示するには、次のコマンドを実行します。

```
+CSQ1  DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

IBM MQ は、以下のようなメッセージで応答します。

```
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(000000000000001F0000000007472616E5F6964547565204E6F762020...)
QMURID(000000026232)
URTYPE(XA)
USERID( )
APPLTAG(CSQ1CHIN)
ASID(0000)
APPLTYPE(CHINIT)
CHANNEL( )
CONNAME( )
END CONN DETAILS
```

このメッセージ内の属性の説明については、DISPLAY CONN コマンドの説明を参照してください。

未確定リカバリー単位のリカバリー

未確定回復単位を回復するためには、次のコマンドを実行します。

```
+CSQ1  RESOLVE INDOUBT( connection-name ) ACTION(COMMIT|BACKOUT)
NID( net-node.number )
```

ここで、

connection-name
IMS システム ID。

ACTION
このリカバリー単位をコミットするか (COMMIT) バックアウトするか (BACKOUT) を示します。

net-node.number
関連する net-node.number。

RESOLVE INDOUBT コマンドを実行したとき、次のどちらかのメッセージが表示されます。

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

未回復項目の解決

指定された回数だけ、IMS は未回復項目 (RRE) のリストを作成します。 RRE は、IBM MQ にとって未確定になっている可能性のある回復単位です。 これらは、次のいくつかの状況で発生します。

- キュー・マネージャーが活動状態でない場合、IMS には、キュー・マネージャーが活動状態になるまで解決できない RRE があります。 これらの RRE は問題とはなりません。
- キュー・マネージャーがアクティブであり、 IMS に接続しており、 IMS が IBM MQ がコミットした作業をバックアウトする場合、IMS アダプターはメッセージ CSQQ010E を発行します。 2 つのシステムのデータが一致していなければならない場合、これは問題となります。 この問題の解決方法については、517 ページの『IMS リカバリー単位の手動回復』を参照してください。
- キュー・マネージャーが活動状態であり、IMS に接続している場合、依然として RRE が存在するにもかかわらず、その問題がメッセージで通知されない場合があります。 IMS への IBM MQ 接続が確立された後、以下の IMS コマンドを発行して、問題があるかどうかを調べることができます。

```
/DISPLAY OASN SUBSYS sysid
```

RRE を除去するには、次の IMS コマンドの 1 つを実行します。

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

ここで、*nnnn* は、 +CSQ1 DISPLAY コマンドに応答して表示される、元のアプリケーションの順序番号です。 これはプログラム・インスタンスのスケジュール番号です。すなわち、最新の IMS コールド・スタート以降にこのプログラムが何番目に呼び出されたかを示します。 IMS は、同じスケジュール番号の未確定リカバリー単位を 2 つ持つことはできません。

上記のコマンドは、IMS の状況をリセットします。しかし、このコマンドによって IBM MQ と通信が行われることはありません。

z/OS *IMS* 従属領域の接続の制御

IMS と IBM MQ との接続は、制御、モニター、および必要に応じて終了することができます。

IMS 従属領域の接続の制御には、 次の活動があります。

- 従属領域からの接続
- 領域エラー・オプション

- 接続の活動のモニター
- 従属領域からの切り離し

従属領域からの接続

制御領域で使用される IMS アダプターは従属領域にもロードされます。 接続は、各従属領域から IBM MQ に対して行われます。 この接続は、IBM MQ と IMS の作業のコミットメントを調整するために使用されます。 初期設定および接続を行うために、IMS は次のことを行います。

1. IMS.PROCLIB からサブシステム・メンバー (SSM) を読み取ります。

   サブシステム・メンバーは、従属領域の EXEC パラメーターに指定することができます。 この指定がない場合は、制御領域 SSM が使用されます。 その領域を IBM MQ に接続する可能性がない場合は、 アダプターがロードされないようにするために、項目のないメンバーを指定します。

2. IBM MQ アダプターをロードします。

   バッチ・メッセージ・プログラムの場合、 アプリケーションが最初のメッセージ処理コマンドを実行するまで、ロードは行われません。 その時点で、IMS は接続を試みます。

   メッセージ処理プログラム領域または IMS 高速パス領域の場合、この試みは、 領域が初期設定されたときに行われます。

領域エラー・オプション

キュー・マネージャーが活動状態でない場合、または最初のメッセージ処理コマンドがアプリケーション・プログラムから送られたときに資源が利用できない場合、 取られる処置は、 SSM 項目で指定されたエラー・オプションで決まります。 このオプションは、次のとおりです。

R
該当の戻りコードがアプリケーションに送られます。

Q
アプリケーションは、異常終了コード U3051 で異常終了します。 入力メッセージは、再びキューに入れられます。

A
アプリケーションは、異常終了コード U3047 で異常終了します。 入力メッセージは破棄されます。

接続の活動のモニター

スレッドは、アプリケーションが最初の成功した IBM MQ 要求を行うときに、従属領域から確立されます。 接続および現在接続を使用するアプリケーションについての情報は、 IBM MQ から次のコマンドを実行することによって表示できます。

```
+CSQ1 DISPLAY CONN(*) ALL
```

このコマンドは、以下のようなメッセージを生成します。

```
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)
```

制御領域の場合は、*thread-xref* は、特別な値 CONTROL になります。 従属領域の場合、この値は、PSBname と連結した PSTid となります。 *auth-id* は、ジョブ・カードからのユーザー・フィールドか z/OS 開始プロシージャー表からの ID になります。

表示されたリストの説明は、「IBM MQ for z/OS のメッセージ、完了コード、および理由コード」にあるメッセージ CSQV402I の説明を参照してください。

IMS では、IBM MQ への接続をモニターするための表示コマンドを提供しています。 このコマンドは、各従属領域接続で活動状態のプログラム、LTERM ユーザー名、および制御領域の接続状況を表示します。 このコマンドは、次のとおりです。

```
/DISPLAY SUBSYS name
```

IMS と IBM MQ の間の接続の状況は、 次のいずれかとして表示されます。

```
CONNECTED
NOT CONNECTED
CONNECT IN PROGRESS
STOPPED
STOP IN PROGRESS
INVALID SUBSYSTEM NAME= name
SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING
```

各従属領域からのスレッド状況は、次のどちらかです。

```
CONN
CONN, ACTIVE (includes LTERM of user)
```

従属領域からの切り離し

IMS.PROCLIB の SSM メンバーの中の値を変更するには、 従属領域を切断します。 そのためには、次のことを行う必要があります。

1. 次の IMS コマンドを発行します。

```
/STOP REGION
```

2. SSM メンバーを更新する
3. 次の IMS コマンドを発行します。

```
/START REGION
```

z/OS *IMS からの切断*

IMS またはキュー・マネージャーのいずれかが終了すると、接続は終了します。 あるいは、IMS マスター端末のオペレーターが明示的に接続を切断することもできます。

IMS と IBM MQ との接続を終了するために、次の IMS コマンドを実行します。

```
/STOP SUBSYS sysid
```

このコマンドは、次のメッセージを、コマンドを実行した端末 (通常、マスター端末オペレーター (MTO)) に送ります。

```
DFS058I STOP COMMAND IN PROGRESS
```

IMS コマンド

```
/START SUBSYS sysid
```

は、接続を再確立するために必要です。

注 : IMS コマンド /STOP SUBSYS は、IMS トリガー・モニターが実行中の場合は完了しません。

z/OS *IMS トリガー・モニターの制御*

CSQQTRMN トランザクションを使用して、IMS トリガー・モニターを停止、および開始することができます。

IMS トリガー・モニター (CSQQTRMN トランザクション) については、IMS トリガー・モニターのセットアップで説明されています。

IMS トリガー・モニターを制御する場合は、以下を参照してください。

- CSQQTRMN の開始
- CSQQTRMN の停止

CSQQTRMN の開始

1. モニターしたい各開始キューごとに、プログラム CSQQTRMN を実行するバッチ型 BMP を開始します。

2. 次の情報を含んでいるデータ・セットを指す DD 名 CSQQUT1 を追加するために、 バッチ JCL を修正します。

```
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
```

ここで、

q_manager_name	キュー・マネージャーの名前 (ブランクの場合は、CSQQDEFV に指定されたデフォルトが想定されます)
init_q_name	モニターする開始キューの名前
lterm	エラー・メッセージの宛先の IMS LTERM 名 (ブランクの場合、デフォルト値は MASTER)
CONSOLEMESSAGES=YES	指定された IMS LTERM へ送られたメッセージが、z/OS コンソールへも送られることを要求します。 このパラメーターが省略されるか、スペルに誤りがあると、コンソールへのメッセージ送信はありません (デフォルト)。

3. CSQQUT1 入力の処理の印刷レポートが必要な場合は、CSQQUT2 の DD 名を追加します。

注 :

1. データ・セット CSQQUT1 は LRECL=80 で定義されます。 他の DCB 情報はそのデータ・セットから取られます。 データ・セット CSQQUT2 の DCB は RECFM=VBA および LRECL=125 です。
2. 各レコードには、キーワードを 1 つだけ入れることができます。 キーワードの値は、そのキーワードに続く最初のブランクによって区切られます。 これは、注釈を入れることが可能であることを意味します。 1 桁目にアスタリスクがあると、入力レコード全体が注釈になります。
3. QMGRNAME キーワードまたは LTERM キーワードのスペルに誤りがあると、CSQQTRMN はそのキーワードにデフォルトを使用します。
4. トリガー・モニター BMP ジョブを実行依頼する前に、サブシステムが IMS で開始されたこと (/START SUBSYS コマンドによって) を確認してください。 開始されていない場合、トリガー・モニター・ジョブは異常終了コード U3042 で終了します。

CSQQTRMN の停止

CSQQTRMN は、いったん開始されると、次のいずれかのイベントによって IBM MQ と IMS の間の接続が切断されるまで実行されます。

- キュー・マネージャーの終了
- IMS の終了

あるいは、z/OS STOP **jobname** コマンドが入力されるまで実行されます。

z/OS IMS ブリッジの制御

このトピックでは、IMS ブリッジの制御に使用できる IMS コマンドについて説明します。

IBM MQ-IMS ブリッジを制御する IBM MQ コマンドはありません。 ただし、以下の方法で、IMS にメッセージが送達されないようにすることができます。

- 非共有キューの場合、すべてのブリッジ・キューに対して ALTER QLOCAL(xxx) GET(DISABLED) コマンドを使用します。
- クラスター・キューの場合、SUSPEND QMGR CLUSTER(xxx) コマンドを使用します。 これは、別のキュー・マネージャーがクラスター・ブリッジ・キューもホスティング している場合に限り有効です。

- クラスター・キューの場合、SUSPEND QMGR FACILITY(IMSBRIDGE) コマンドを使用します。 これ以上のメッセージは IMS に送信されませんが、未解決のトランザクションに対する応答は IMS から受信されます。

  IMS への メッセージの送信を再開するには、RESUME QMGR FACILITY(IMSBRIDGE) コマンドを発行します。

MQSC コマンド DISPLAY SYSTEM を使用して、ブリッジが中断されているかどうかを表示することもできます。

これらのコマンドについて詳しくは、MQSC コマンドを参照してください。

詳細については、以下を参照してください。

- 529 ページの『IMS ブリッジの開始と停止』
- 529 ページの『IMS 接続の制御』
- ブリッジ・キューの制御
- 530 ページの『IMS ブリッジの再同期』
- tpipe 名の扱い
- IMS からのメッセージの削除
- tpipe の削除
- 532 ページの『IMS トランザクションの有効期限』

IMS ブリッジの開始と停止

OTMA を開始することによって、IBM MQ ブリッジを開始します。 IMS コマンド

```
/START OTMA
```

を使用して開始するか、IMS システム・パラメーターに OTMA=YES を指定することによって、自動的に開始します。 OTMA がすでに開始されている場合は、 キュー・マネージャーの開始が完了したときに、ブリッジが自動的に開始します。 IBM MQ イベント・メッセージは、OTMA が開始されたときに生成されます。

IMS コマンド

```
/STOP OTMA
```

を使用して、OTMA 通信を停止します。 このコマンドを実行すると、IBM MQ イベント・メッセージが生成されます。

IMS 接続の制御

IMS は、IBM MQ との接続を制御およびモニターするために、次のオペレーター・コマンドを提供します。

/DEQUEUE TMEMBER *tmember* TPIPE *tpipe*
メッセージを Tpipe から除去します。 すべてのメッセージを除去するためには PURGE を指定し、最初のメッセージのみを除去するためには PURGE1 を指定します。

/DISPLAY OTMA
OTMA サーバーとクライアントについて要約情報を表示し、 またクライアント状況を表示します。

/DISPLAY TMEMBER *name*
OTMA クライアントについての情報を表示します。

/DISPLAY TRACE TMEMBER *name*
トレースされている対象に関する情報を表示します。

/SECURE OTMA
セキュリティー・オプションを設定します。

/START OTMA
OTMA を介する通信を使用可能にします。

/START TMEMBER *tmember* TPIPE *tpipe*
指定された Tpipe を開始します。

/STOP OTMA
OTMA を介する通信を停止します。

/STOP TMEMBER *tmember* TPIPE *tpipe*
指定された Tpipe を停止します。

/TRACE
IMS トレースを制御します。

これらのコマンドについて詳しくは、使用している IMS のレベルについて「*IMS/ESA* オペレーター・リファレンス」を参照してください。

IMS コマンドの応答は、コマンドを実行した端末に送られます。 IMS コマンドを実行するための許可は、IMS セキュリティーに基づいて与えられます。

ブリッジ・キューの制御

ブリッジを介して行われる、XCF メンバー名 *tmember* を持つキュー・マネージャーとの通信を停止するには、次の IMS コマンドを実行します。

```
/STOP TMEMBER tmember TPIPE ALL
```

通信を再開するために、次の IMS コマンドを実行します。

```
/START TMEMBER tmember TPIPE ALL
```

キューの Tpipe は、MQ DISPLAY QUEUE コマンドを使用して 表示することができます。

単一 Tpipe 上でキュー・マネージャーとの通信を停止するには、次の IMS コマンドを実行します。

```
/STOP TMEMBER tmember TPIPE tpipe
```

活動状態のブリッジ・キューごとに 1 つか 2 つの Tpipe が作成されます。したがって、このコマンドを実行すると、IBM MQ キューとの通信が停止します。 通信を再開するために、次の IMS コマンドを実行します。

```
/START TMEMBER tmember TPIPE tpipe
```

また、IBM MQ キューの属性を変更することにより、通信を抑制することもできます。

IMS ブリッジの再同期

キュー・マネージャー、IMS、または OTMA の再始動時には、IMS ブリッジも、必ず自動的に再始動されます。

IMS ブリッジが最初に行うタスクは、IMS との同期を取り直すことです。 このタスクには、同期化されたすべての Tpipe についての IBM MQ と IMS の検査順序番号が関与します。 同期化 Tpipe は、コミット・モード・ゼロ (コミット後送信) を使用して IBM MQ - IMS ブリッジ・キューから IMS に持続メッセージが送信されるときに使用されます。

ブリッジが IMS と再同期できないと、IMS センス・コードがメッセージ CSQ2023E に戻され、OTMA への接続が停止されます。 また、ブリッジが個別の IMS Tpipe と再同期できないと、IMS センス・コードがメッセージ CSQ2025E に戻され、Tpipe が停止されます。 Tpipe がコールド・スタートされていると、回復可能順序番号は自動的に 1 にリセットされます。

ブリッジが、Tpipe との再同期時に、 一致しない順序番号を発見すると、メッセージ CSQ2020E が出されます。 IBM MQ コマンド RESET TPIPE を使用して、IMS Tpipe との再同期を開始します。 XCF グループ名、メンバー名、および Tpipe の名前を入力する必要があります。 この情報はメッセージによって提供されます。

次のものを指定することもできます。

- IBM MQ によって送信されるメッセージの Tpipe に設定する新規の回復可能順序番号、およびパートナーの受信順序番号として設定される新規の回復可能順序番号。 この番号を指定しないと、パートナーの受信順序番号は、現在の IBM MQ 送信順序番号に設定されます。
- IBM MQ によって受信されるメッセージの Tpipe に設定する新規の回復可能順序番号、およびパートナーの送信順序番号として設定される新規の回復可能順序番号。 この番号を指定しないと、パートナーの送信順序番号は、現在の IBM MQ 受信順序番号に設定されます。

Tpipe に関連した未解決のリカバリー単位がある場合、これもメッセージで通知されます。 IBM MQ コマンド RESET TPIPE を使用して、リカバリー単位をコミットするのかバックアウトするのかを指定してください。 リカバリー単位をコミットする場合は、メッセージのバッチは既に IMS に送られていて、ブリッジ・キューから削除されます。 リカバリー単位をバックアウトする場合、メッセージはブリッジ・キューに戻され、その後 IMS に送られます。

コミット・モード 1 (送信してからコミット) の Tpipe は同期化されません。

コミット・モード 1 のトランザクションに関する考慮事項

IMS では、コミット・モード 1 (CM1) のトランザクションは、出力応答を同期点より前に送信します。

CM1 トランザクションでは、 例えば次に示すような原因で応答を送信できない場合があります。

- 応答を送信する Tpipe が停止している
- OTMA が停止している
- OTMA クライアント (つまりキュー・マネージャー) が失われた
- 応答先キューと送達不能キューが使用不可である

これらの原因のため、メッセージを送信した IMS アプリケーションはコード U0119 で疑似異常終了します。 この場合、IMS トランザクションとプログラムは停止しません。

これらの理由によって、IMS からの応答が送達されないだけでなく、IMS へもメッセージが送信されない場合が少なくありません。 U0119 の異常終了が発生するのは、次の場合です。

- IMS 内にメッセージがあるときに、Tpipe、OTMA、またはキュー・マネージャーのいずれかが停止した
- IMS が別の Tpipe 上で着信メッセージに応答し、その Tpipe が停止した
- IMS が別の OTMA クライアントに応答し、そのクライアントが使用不可である

U0119 異常終了が発生するたびに、 IMS への着信メッセージと IBM MQ への応答メッセージの両方が失われます。 CM0 トランザクションの出力がこのいずれかの理由で送達されない場合、その出力は IMS 内の Tpipe 上のキューに入ります。

tpipe 名の扱い

IBM MQ - IMS ブリッジの制御に使用されるコマンドの多くは、*tpipe* 名を必要とします。 このトピックでは、tpipe 名の詳細の調べ方について説明します。

IBM MQ - IMS ブリッジを制御する多くのコマンドには、*tpipe* 名が必要です。 tpipe 名は DISPLAY QUEUE コマンドから取得できますが、以下の点に注意してください。

- tpipe 名は、ローカル・キューを定義するときに割り当てられます。
- ローカル・キューには tpipe 名が 2 つ与えられます。1 つは同期用、もう 1 つは非同期用です。
- 特定のローカル・キューに固有の IMS と IBM MQ の間で何らかの通信が行われるまで、TPIPE 名は IMS に認識されません。
- IBM MQ - IMS ブリッジで tpipe を使用するには、XCF グループ・フィールドおよびメンバー名フィールドが正しく入力されたストレージ・クラスに、関連するキューが割り当てられていなければなりません。

IMS からのメッセージの削除

Tmember/Tpipe が停止している場合、IMS ブリッジを介して IBM MQ を宛先とするメッセージは削除することができます。 XCF メンバー名 *tmember* を持っているキュー・マネージャーへの 1 つのメッセージを削除するためには、次の IMS コマンドを実行します。

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

Tpipe 上ですべてのメッセージを削除するためには、次の IMS コマンドを実行します。

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

tpipe の削除

IMS tpipe をユーザー自身が削除することはできません。 IMS は次の場合に削除されます。

- IMS がコールド・スタートすると、同期 tpipe が削除されます。
- IMS が再始動されると、非同期 tpipe が削除されます。

IMS トランザクションの有効期限

トランザクションには有効期限が関連付けられています。どの IBM MQ メッセージにも有効期限を関連付けることができます。 有効期限の間隔は、MQMD.Expiry フィールドを使用して、アプリケーションから IBM MQ に渡されます。 この時間は、メッセージの有効期限が切れるまでの時間で、0.1 秒単位の値で表されます。 メッセージの有効期限が切れた後に、そのメッセージの MQGET を実行しようとすると、キューからそのメッセージが除去され、期限切れ処理が実行されます。 有効期限は、IBM MQ ネットワークでキュー・マネージャー間をメッセージが流れるにつれて短くなっていきます。 IMS メッセージが IMS ブリッジを越えて OTMA に渡されると、メッセージの残りの有効期限が、トランザクションの有効期限として OTMA に渡されます。

トランザクションに有効期限が指定されている場合、OTMA は IMS 内の次の 3 つの異なる場所の入力トランザクションを期限切れにします。

- XCF から受信する入力メッセージ
- 入力メッセージのエンキュー時間
- アプリケーションの GU 時間

GU 時間が過ぎると期限切れ処理は実行されません。

トランザクション EXPRTIME は、以下のものによって提供できます。

- IMS トランザクション定義
- IMS OTMA メッセージ・ヘッダー

- IMS DFSINSX0 ユーザー出口
- IMS CREATE または UPDATE TRAN コマンド

IMS は、0243 でトランザクションを打ち切ることにより、トランザクションを期限切れにしたことを示し、メッセージを発行します。 発行されるメッセージは、非共有キュー環境では DFS555I、共有キュー環境では DFS2224I です。

z/OS で操作 Advanced Message Security

Advanced Message Security アドレス・スペースは、z/OS MODIFY コマンドを使用してコマンドを受け入れます。

Advanced Message Security (AMS) アドレス・スペースに対してコマンドを入力するには、z/OS MODIFY コマンドを使用します。

例:

```
F qmgrAMSM, cmd
```

ここで、*qmgr* は、開始タスク名の接頭部です。

533 ページの表 29 に、受け入れられる MODIFY コマンドを示します。

表 *29. Advanced Message Security* アドレス・スペース *MODIFY* コマンド

コマンド	オプション	説明
DISPLAY		バージョン情報の表示
REFRESH	KEYRING POLICY ALL	鍵リング証明書、セキュリティー・ポリシー、またはその両方をリフレッシュします。
SMFAUDIT	SUCCESS FAILURE ALL	AMS が正常にメッセージを保護または保護解除した場合、AMS がメッセージの保護または保護解除に失敗した場合、あるいはその両方の場合に、SMF 監査が必要かどうかを設定します。
SMFTYPE	0 - 255	AMS がメッセージを保護または保護解除するときに生成される SMF レコード・タイプを設定します。 SMF 監査を無効にする場合は、レコード・タイプ 0 を指定します。

注 : オプションを指定する場合は、コンマで区切る必要があります。 以下に例を示します。

```
F qmgrAMSM,REFRESH KEYRING
F qmgrAMSM,SMFAUDIT ALL
F qmgrAMSM,SMFTYPE 180
```

REFRESH コマンド

REFRESH コマンドを発行することで有効になる変更は、**REFRESH** コマンドが完了した後に MQOPEN を発行するアプリケーションに適用されます。 キューをオープンしている既存のアプリケーションは、アプリケーションがキューをオープンした時点からのオプションを使用し続けます。 新規の値を使用するには、アプリケーションでキューをクローズして再オープンする必要があります。

開始と停止 AMS

Advanced Message Security アドレス・スペースを開始または停止するためにコマンドを入力する必要はありません。 AMS アドレス・スペースは、AMS が CSQ6SYSP の **SPLCAP** パラメーターで有効にされている

場合にキュー・マネージャーが開始されると自動的に開始され、キュー・マネージャーが停止すると停止します。

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、536 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。

V 9.2.0 V 9.2.0 **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。

V 9.2.0 V 9.2.0 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。

- V 9.2.0 V 9.2.0 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、 536 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。

V 9.2.0 パスワード暗号鍵の指定

IBM MQ 9.1.5 以降、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 MQIPT の構成に含まれている場合は、MQIPT が開始時に読み取り可能なファイルに、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。
- V 9.2.0 V 9.2.0 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

V 9.2.0 V 9.2.0

IBM MQ 9.2.0 以降、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することができます。 ローカル管理を使用すると、管理対象の MQIPT インスタンスと同じシステムで実行する場合にのみ、 **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

V 9.2.0 V 9.2.0

IBM MQ 9.2.0 以降、MQIPT には、保護されていないコマンド・ポート 1 つと、TLS で保護されたコマンド・ポート 1 つを構成することができます。 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、541 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、538 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。

V 9.2.0 V 9.2.0 TLS コマンド・ポートを使用した MQIPT の管理

IBM MQ 9.2.0 以降、 **mqiptAdmin** コマンドによって発行される管理コマンドを listen するために TLS コマンド・ポートを使用するように MQIPT を構成できます。 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・ポートには、PKCS #12 の鍵リング、または PKCS #11 の Cryptographic Token Interface に対応した暗号ハードウェアに保管されたサーバー証明書を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** コマンドからアクセスできる PKCS #12 の鍵リング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、541 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、**mqiptKeycmd** (iKeyman) コマンド・ライン・インターフェース (CLI) 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リングとデジタル証明書を管理する方法について説明します。 CLI は、**mqiptKeycmd** コマンドを使用することで使用できます。 鍵リングとデジタル証明書の管理に使用できるその他のコマンドについて詳しくは、**mqiptKeyman** および **mqiptKeycmd** 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) TLS コマンド・ポートで使用する PKCS #12 の鍵リング・ファイルを作成します。 この鍵リングを使用して、TLS コマンド・ポートのサーバー証明書を保管します。

CLI を使用して鍵リング・ファイルを作成するには、以下のコマンドを入力します。

```
mqiptKeycmd -keydb -create -db filename -pw password -type pkcs12
```

filename は作成する鍵リング・ファイルの名前、 *password* は鍵リングのパスワードです。

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。

iKeyman CLI を使用して証明書要求を作成するには、以下のコマンドを入力します。

```
mqiptKeycmd -certreq -create -db filename -pw password
            -label label -size key_size -sig_alg algorithm
            -dn distinguished_name -file certreq_filename -type pkcs12
```

ここで、

-db *filename*
鍵リング・ファイル名を指定します。

-pw *password*
鍵リング・パスワードを指定します。

-label *label*
証明書ラベルを指定します。

-size *key_size*
鍵のサイズを指定します。

-sig_alg *algorithm*
項目の鍵ペアの作成に使用される非対称署名アルゴリズムを指定します。

-dn *distinguished_name*
二重引用符で囲んだ X.500 識別名を指定します。

-file *certreq_filename*
認証要求のファイル名を指定します。

c) 手順 539 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 から署名付きの証明書が送られたら、その署名付きの証明書を鍵リング・ファイルに受け取ります。

CLI を使用して、署名付きの証明書を鍵リングに受け取るには、以下のコマンドを入力します。

```
mqiptKeycmd -cert -receive -file cert_filename -db filename
            -pw password -type pkcs12
```

cert_filename は証明書が含まれているファイルの名前、*filename* は鍵リング・ファイルの名前、*password* は鍵リングのパスワードです。

e) **mqiptPW** コマンドを使用して鍵リング・パスワードを暗号化します。

次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、暗号化する鍵リング・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 539 ページの『1.a』 で作成した鍵リングのファイル名に設定します。

iii) **SSLCommandPortKeyRingPW** の値として、手順 539 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 539 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) **mqiptAdmin** コマンドでトラストストアとして使用する PKCS #12 の鍵リングを作成します。

CLI を使用して鍵リング・ファイルを作成するには、以下のコマンドを入力します。

```
mqiptKeycmd -keydb -create -db filename -pw password -type pkcs12
```

filename は作成する鍵リング・ファイルの名前、 *password* は鍵リングのパスワードです。

b) TLS コマンド・ポート証明書に署名した CA の CA 証明書を、手順 540 ページの『2.a』 で作成した鍵リングにインポートします。

iKeyman CLI を使用して CA 証明書をインポートするには、以下のコマンドを入力します。

```
mqiptKeycmd -cert -add -db filename -pw password -type pkcs12
            -label certlabel -file cert_filename
```

ここで、

filename
鍵リングのファイル名を指定します

パスワード
鍵リングの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 証明書が入っているファイルの名前を指定します

c) **mqiptPW** コマンドを使用して鍵リング・パスワードを暗号化します。

次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。 プロンプトが表示されたら、暗号化する鍵リング・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。

d) **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
手順 540 ページの『2.a』 で作成した鍵リングのファイル名です。

key_ring_password
ステップ 540 ページの『2.c』 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。 ステップ 540 ページの『2.c』 で鍵リング・パスワードを暗号化するために暗号鍵ファイルを使用した場合にのみ、**PasswordProtectionKeyFile** プロパティーを指定する必要があります。

e) **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 541 ページの『2.d』 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、541 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。

V 9.2.0 V 9.2.0 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。
   次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
ステップ 541 ページの『1』 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。

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**
  - V 9.2.0 V 9.2.0 **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**
- V 9.2.0 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。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。 本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510
東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
日本アイ・ビー・エム株式会社
法務・知的財産
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
〒 103-8510
103-8510
東京 103-8510、日本

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 INTERNATIONAL BUSINESS MACHINES CORPORATION は、法律上の瑕疵担保責任、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 "" 国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものでは ありません。 それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他の ライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で 決定されたものです。 そのため、他の操作環境で得られた結果は、異なる可能性があります。 一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。 さらに、一部の測定値が、推定値である可能性があります。 実際の結果は、異なる可能性があります。 お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、 もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。 したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、 予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。これらの名前はすべて架空のものであり、名前や住所が類似する個人や企業が実在して いるとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報 (提供されている場合) は、このプログラムで使用するアプリケーション・ソフトウェアの作成を支援することを目的としています。

本書には、プログラムを作成するユーザーが WebSphere MQ のサービスを使用するためのプログラミング・インターフェースに関する情報が記載されています。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。 診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

重要 : この診断、修正、およびチューニング情報は、変更される可能性があるため、プログラミング・インターフェースとして使用しないでください。

商標

IBM、 IBM ロゴ、 ibm.com®は、世界の多くの国で登録された IBM Corporation の商標です。 現時点での IBM の商標リストについては、 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml をご覧ください。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

この製品には、 Eclipse Project (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: