7.5

IBM WebSphere 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、163 ページの『特記事項』に記載されている情報をお読みください。

本書は、 IBM® WebSphere® MQ バージョン 7 リリース 5、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

目次

IBM WebSphere MQ 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 WebSphere MQ オブジェクトの管理では、ローカル管理とリモート管理が可能です。7 ページの『ローカル管理およびリモート管理』を参照してください。

IBM WebSphere 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。 各インターフェースの詳細については、このトピックのそれぞれのセクションとリンクを参照してください。

IBM WebSphere 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。

- 5 ページの『IBM WebSphere MQ の制御コマンド』
- 5 ページの『IBM WebSphere MQ スクリプト (MQSC) コマンド』
- 6 ページの『プログラマブル・コマンド・フォーマット (PCF)』

そのほかに、IBM WebSphere 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。

- 6 ページの『IBM WebSphere MQ Explorer』
- 7 ページの『Windows のデフォルト構成アプリケーション』
- 7 ページの『Microsoft Cluster Service (MSCS)』

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 WebSphere 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、8 ページの『管理タスクの自動化』を参照してください。

IBM WebSphere MQ の制御コマンド

制御コマンドを使用すると、キュー・マネージャーそのものに対して管理タスクを実行することができます。

IBM WebSphere MQ for Windows、 UNIX and Linux® システムには、システム・コマンド行で発行する 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、キュー・マネージャーの作成および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 WebSphere MQ 制御コマンドを参照してください。

IBM WebSphere 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。

このセクションでは、MQSC コマンドで指定するオブジェクト属性を (RQMNAME のように) 大文字で表記していますが、実際には大/小文字の区別はありません。 MQSC コマンド属性名は 8 文字までに制限されています。

MQSC コマンドは、すべてのオペレーティング・システムで使用できます。 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。

Windows、 UNIX 、または Linux では、システム・コマンド行で発行される単一コマンドとして MQSC を使用できます。 複雑なコマンドや複数のコマンドを実行する場合は、Windows、UNIX、Linux のシステム・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 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、MQSC 参照を参照してください。

73 ページの『スクリプト (MQSC) コマンド』には、各 MQSC コマンドおよびその構文の説明が含まれています。

ローカル管理での MQSC コマンドの使用の詳細については、72 ページの『MQSC コマンドによるローカル管理タスクの実行』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 WebSphere 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、9 ページの『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。

IBM WebSphere MQ Explorer

IBM WebSphere 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 WebSphere MQ Explorer を開始できます。

Linux では、IBM WebSphere 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 WebSphere 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。詳細およびサポートパック MS0T のダウンロードについては、https://www.ibm.com/support/docview.wss?uid=swg24021041 を参照してください。

詳しくは、56 ページの『IBM WebSphere MQ Explorer による管理』を参照してください。

Windows のデフォルト構成アプリケーション

Windows デフォルト構成プログラムを使用して、IBM WebSphere MQ オブジェクトの スターター (またはデフォルト) セットを作成できます。 作成されるデフォルト・オブジェクトの要約については、表 1. Windows のデフォルト構成アプリケーションによって作成されるオブジェクトを参照してください。

Microsoft Cluster Service (MSCS)

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 WebSphere MQ クラスターと混同しないことが重要です。 相違点は、次のとおりです。

IBM WebSphere MQ クラスター
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。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
相互接続されたコンピューターのグループ。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クラスター内の別のコンピューターへアプリケーションの状態データが転送され、そこで操作が再開されるように構成されています。

Microsoft Cluster Service (MSCS) のサポート には、MSCS を使用するように IBM WebSphere MQ for Windows システムを構成する方法に関する詳細情報が記載されています。

関連概念
WebSphere MQ 技術概要
68 ページの『ローカル IBM WebSphere MQ オブジェクトの管理』
このセクションでは、メッセージ・キュー・インターフェース (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 WebSphere MQ オブジェクトの管理方法を説明します。 ここでは、ローカル管理とは、IBM WebSphere 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。

105 ページの『リモート IBM WebSphere MQ オブジェクトの管理』

XA リソース・マネージャーとの連絡が失われる際の考慮事項
関連タスク
計画
構成
関連資料
トランザクション・サポートのシナリオ

ローカル管理およびリモート管理

WebSphere 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 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。 WebSphere 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。

WebSphere MQ は、リモート管理 という管理方法による、1 つの地点からの管理をサポートします。 これにより、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、WebSphere MQ エクスプローラーにも適用されます。 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、WebSphere MQ エクスプローラーにも適用されます。

リモート管理の詳細は、105 ページの『リモート IBM WebSphere MQ オブジェクトの管理』に記載しています。

IBM WebSphere MQ 制御コマンドの使用方法

このセクションでは、IBM WebSphere MQ 制御コマンドの使用方法を説明します。

制御コマンドを発行する場合は、ユーザー ID が mqm グループのメンバーである必要があります。 詳しくは、 WebSphere MQ on UNIX、 Linux および Windows システムを管理する権限 を参照してください。 さらに、次の環境特有の情報に注意してください。

WebSphere MQ for Windows
すべての制御コマンドはコマンド行から発行できます。 コマンド名およびフラグは大文字小文字の区別をしません。コマンド名およびフラグを大文字、小文字または大文字と小文字の組み合わせで入力することができます。 しかし、制御コマンドの引数 (キュー名など) は大文字小文字の区別をします。

構文記述では、ハイフン (-) はフラグ標識として使用されます。 ハイフンの代わりに斜線 (/) を使用できます。

WebSphere MQ for UNIX and Linux システム
すべての WebSphere MQ 制御コマンドはシェルから発行できます。 すべてのコマンドは、大/小文字が区別されます。

制御コマンドのサブセットは、IBM WebSphere MQ エクスプローラーを使用して発行できます。

詳しくは、WebSphere MQ 制御コマンドを参照してください。

管理タスクの自動化

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、WebSphere 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

WebSphere 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、WebSphere 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 WebSphere 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 は、コマンドを指定します (例: *Change Queue* (MQCMD_CHANGE_Q))。

PCF データ構造とその実装方法の詳細説明については、9 ページの『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。

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。

この製品資料で説明するプログラマブル・コマンド・フォーマットは、以下でサポートされます。

- IBM WebSphere MQ for AIX®
- IBM WebSphere MQ for HP-UX
- IBM WebSphere MQ for Linux
- IBM WebSphere MQ for Solaris
- IBM WebSphere MQ for Windows
- IBM WebSphere MQ for HP Integrity NonStop Server

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。

WebSphere 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。 PCF コマンドは、 WebSphere 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、および ストレージ・クラス) を管理するためのシステム管理アプリケーション・プログラムで使用できます。 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。

プログラマブル・コマンド・フォーマットの使用

WebSphere 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。

- 10 ページの『PCF コマンド・メッセージ』
- 13 ページの『応答』
- IBM WebSphere MQ オブジェクトの命名規則
- 15 ページの『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。

z/OS 以外の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。

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、WebSphere 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

WebSphere 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。

Report
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

Encoding
Windows、UNIX、または Linux 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・データに使用しているエンコード方式を設定します。必要に応じて変換が実行されます。

CodedCharSetId
以下に送信する場合:Windows、 UNIX または Linux システムでは、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。必要に応じて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

Offset
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。

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。WebSphere 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、バージョン 6.0 以降の IBM WebSphere 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 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。

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 WebSphere 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。

IBM WebSphere MQ for Windows、 UNIX and Linux システム

Windows UNIX Linux

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、WebSphere MQ オブジェクト権限の検査は、16 ページの表 1 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

WebSphere MQ (HP Integrity NonStop Server 用)

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 WebSphere MQ オブジェクト権限の検査は、16 ページの表 1 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。

- 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

WebSphere MQ オブジェクト権限

表 *1. Windows*、*HP Integrity NonStop Server*、 *UNIX and Linux* システム-オブジェクト権限

コマンド	WebSphere 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

表 *1. Windows*、*HP Integrity NonStop Server*、 *UNIX and Linux* システム-オブジェクト権限 *(*続き*)*		
コマンド	**WebSphere MQ オブジェクト権限**	**クラス権限 (オブジェクト・タイプに対するもの)**
Copy Channel (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
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

表 *1. Windows*、*HP Integrity NonStop Server*、 *UNIX and Linux* システム-オブジェクト権限 *(*続き*)*		
コマンド	**WebSphere 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 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
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

表 *1. Windows*、*HP Integrity NonStop Server*、 *UNIX and Linux* システム-オブジェクト権限 *(*続き*)*		
コマンド	**WebSphere 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)**
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
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、システム管理ガイド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。

WebSphere 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、 チャネルの表示に記載しています。

MQAI を使用して PCF の使い方を単純化する

MQAI は、 AIX、 HP-UX、 IBM i、 Linux、Solaris、および Windows の各プラットフォームで使用可能な WebSphere MQ への管理インターフェースです。

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。

以下の方法で MQAI を使用します。

PCF メッセージの使用法の単純化
MQAI は、WebSphere MQ を管理する簡単な方法です。独自の PCF メッセージを書き込む必要がないので、複雑なデータ構造に関連する問題を回避することができます。

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。 これを行うには、すべての構造についてプログラムに複数のステートメントを指定する必要があり、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。

エラー条件の処理の簡略化
PCF コマンドから戻りコードを戻すのは難しいですが、MQAI によりエラー条件の処理が簡単になります。

データ・バッグを作成し入力した後、`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。mqExecute 呼び出しはいずれの応答メッセージも待ちます。 `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 の詳細については、20 ページの『IBM WebSphere MQ 管理インターフェース (MQAI) の紹介』を参照してください。

IBM WebSphere MQ 管理インターフェース (MQAI) の紹介

IBM WebSphere MQ 管理インターフェース (MQAI) は IBM WebSphere MQ へのプログラミング・インターフェースです。 このインターフェースは、IBM WebSphere 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 概念および関連用語

MQAI は WebSphere MQ に対するプログラミング・インターフェースで、C 言語だけでなく Visual Basic for Windows も使用します。 z/OS 以外のプラットフォームで使用可能です。

MQAI は、データ・バッグを使用して WebSphere MQ キュー・マネージャーで管理タスクを実行します。データ・バッグを使用すると、もう 1 つの管理インターフェース、つまりプログラマブル・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) を処理できます。 MQAI は、MQGET 呼び出しおよび MQPUT 呼び出しを使用するよりも簡単に PCF を操作できます。

データ・バッグの詳細については、46 ページの『データ・バッグ』を参照してください。 PCF の詳細については、9 ページの『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。

MQAI の使用

MQAI を使用して、次のことを実行できます。

- PCF メッセージの使用方法を単純化します。 MQAI によって、WebSphere MQ の管理が容易になります。独自の PCF メッセージを作成する必要がないため、複雑なデータ構造に関連する問題を避けることができます。
- エラー条件をより簡単に処理します。 WebSphere MQ スクリプト (MQSC) コマンドから 戻りコードを返すのは困難ですが、MQAI を使用すれば、プログラムは簡単にエラー状態を処理できるようになります。
- アプリケーション間でデータを交換します。 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 WebSphere 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。 MQAI を使用した WebSphere MQ

の管理とアプリケーション間のデータの交換の詳細については、19 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。

MQAI の使用例

MQAI の使用方法をデモンストレーションするプログラム例を、次のリストにいくつか示します。 サンプルでは以下のタスクを実行します。

1. ローカル・キューを作成します。 21 ページの『ローカル・キューを作成するサンプル C プログラム (amqsaicq.c)』
2. 簡単なイベント・モニターによる画面へのイベントの表示 25 ページの『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)』
3. すべてのローカル・キューとその現在の項目数を示すリストの印刷 37 ページの『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)』
4. すべてのチャネルとそのタイプのリストの印刷 32 ページの『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)』

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、WebSphere MQ の場合と同じライブラリーにリンクします。 WebSphere MQ アプリケーションの構築方法については、WebSphere MQ 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 WebSphere 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 WebSphere MQ 構成のヒントは、41 ページの『IBM WebSphere MQ 構成のヒント』で説明されています。

IBM WebSphere MQ 管理インターフェース (MQAI)

IBM WebSphere MQ for Windows、 AIX、 Linux、 HP-UX、および Solaris は、 IBM WebSphere MQ 管理インターフェース (MQAI) をサポートします。 MQAI は、MQI の代わりに PCF を送信および受信するための IBM WebSphere MQ に対するプログラミング・インターフェースです。

MQAI はデータ・バッグ を使用します。MQAI を使用して直接 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 では、パラメーターをデータ・バッグに渡すことで PCF メッセージへのプログラミング・アクセスを容易にして、各構造体に必要なステートメントが 1 つのみになるようにしています。 このアクセスによって、プログラマーは配列を処理し、ストレージを割り振る必要がなくなり、また PCF の詳細を入れる必要がある程度なくなります。

MQAI は、PCF メッセージをコマンド・サーバーに送信し、応答を待つことにより、WebSphere MQ を管理します。

MQAI については、本書の第 2 部で説明します。 MQAI に対するコンポーネント・オブジェクト・モデル・インターフェースの説明については、Java の使用の資料を参照してください。

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   */
/*               WebSphere 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 WebSphere 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,                  /* WebSphere 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);

}
```

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 WebSphere 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 needs 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;
}
```

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 WebSphere 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);
}
```

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 WebSphere 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 WebSphere 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,                   /* WebSphere 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);
}
```

IBM WebSphere MQ 構成のヒント

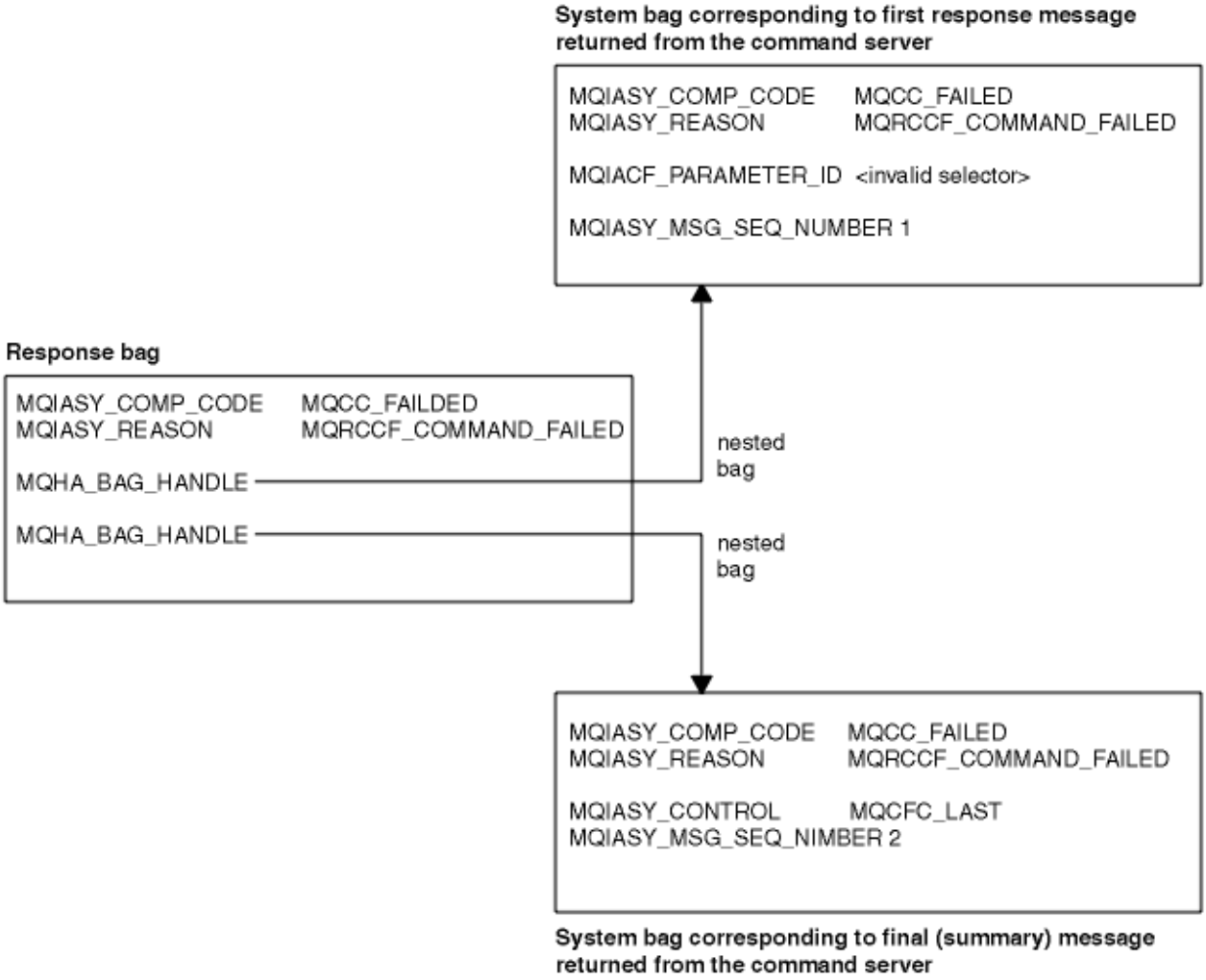
MQAI 使用時のプログラミングのヒント

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。 ここでは、MQAI を使用して WebSphere MQ を構成するためのヒントをいくつか紹介します。

- WebSphere 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 WebSphere 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、55 ページの『mqExecute 呼び出しを使用したコマンド・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。



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。 索引付けについて詳しくは、42 ページの『MQAI での索引付け』を参照してください。

- 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。 データ変換について詳しくは、43 ページの『MQAI でのデータ変換』を参照してください。

- 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。 メッセージ記述子の私用について詳しくは、45 ページの『MQAI でのメッセージ記述子の使用』を参照してください。

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、43 ページの図 1 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。

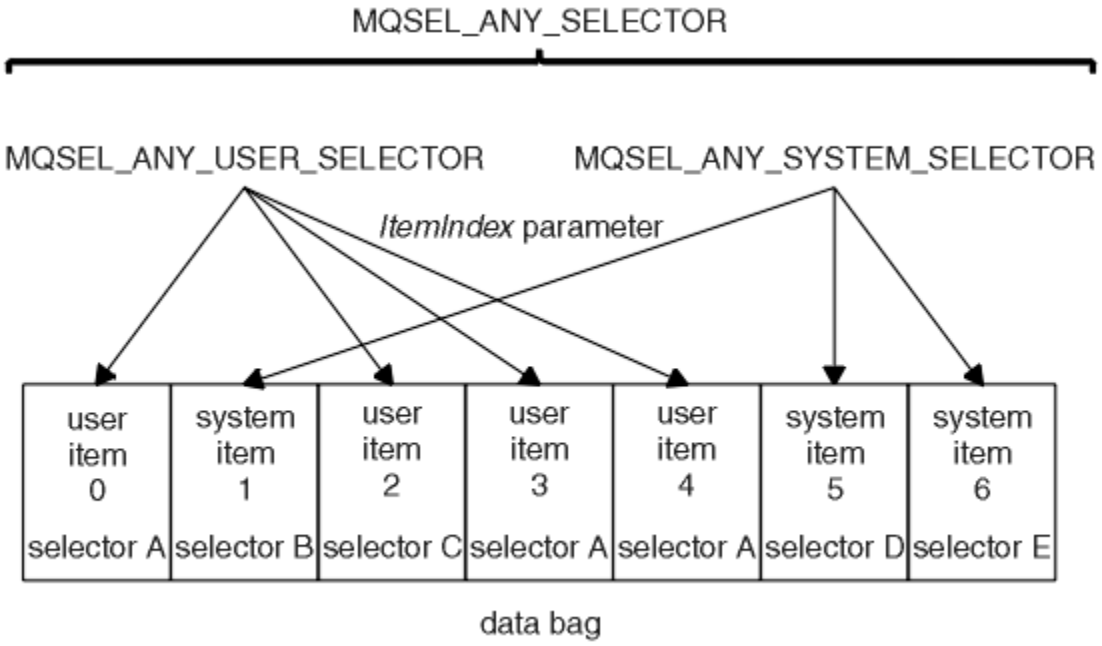


図 *1.* 索引付け

43 ページの図 1 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。

Selector	***ItemIndex***
セレクター A	1
MQSEL_ANY_USER_SELECTOR	2
MQSEL_ANY_SELECTOR	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、バッグ内の情報の変更および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。

MQAI でのデータ変換

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 呼び出しまたは 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。

44 ページの表 2 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 *2. 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
mqBufferToBag	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

表 2. *CCSID* 処理 (続き)

MQAI 呼び出し	**CCSID**	**呼び出しへの入力**	**呼び出しへの出力**

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。

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 3. *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、45 ページの表 3 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 4. *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、45 ページの表 4 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、45 ページの表 5 に示す値を取ります。

表 5. 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

表 5. メッセージ記述子値 (続き)	
パラメーター	**値**
MsgType	45 ページの表 4 を参照
Expiry	30 秒 (注 46 ページの『1』)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	バッグ CCSID により異なる (注 46 ページの『2』)
Format	45 ページの表 4 を参照
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	注 46 ページの『3』 を参照
ReplyToQMgr	ブランク

注:

1. この値は、 **OptionsBag** パラメーターを使用して **mqExecute** 呼び出しでオーバーライドできます。 これについては、 **mqExecute** を参照してください。
2. 43 ページの『MQAI でのデータ変換』 を参照。
3. タイプ **MQMT_REQUEST** のメッセージ用の、ユーザー指定の応答キューまたは MQAI 生成の一時動的キューの名前。 あるいはブランク。

データ・バッグ

データ・バッグは、MQAI を使用して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方法です。

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、49 ページの『データ 項目』 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、42 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。

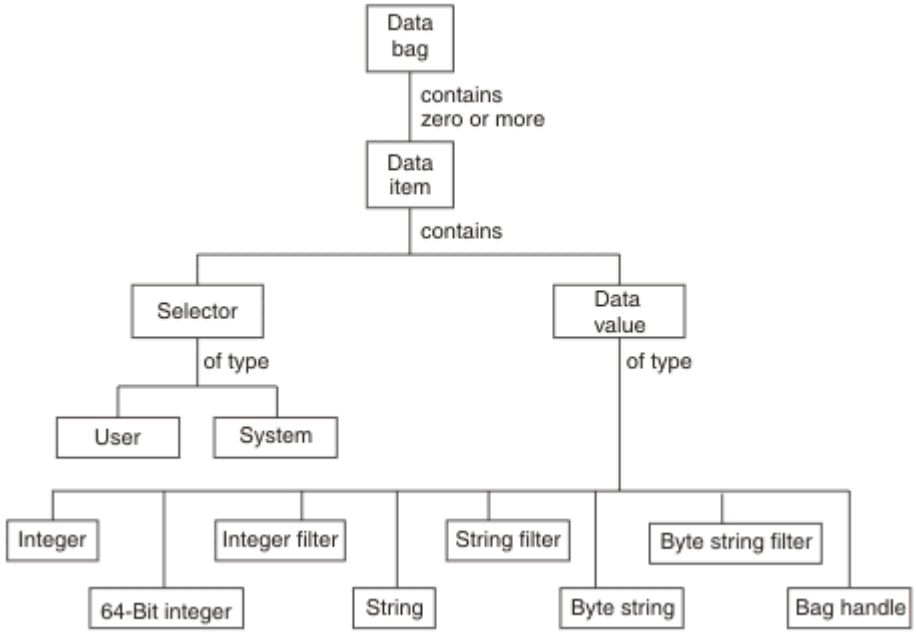


図 2. *MQAI* 概念の階層

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 WebSphere 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。 48 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、WebSphere 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、48 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、WebSphere 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: 48 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: 48 ページの『データ・バッグの書き込みおよび受信』

- データ・バッグにデータ項目を追加する: 49 ページの『バッグへのデータ項目の追加』
- データ・バッグ内に照会コマンドを追加する: 50 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: 51 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: 53 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: 52 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: 53 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: 53 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: 53 ページの『バッグおよびバッファーの変換』

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、49 ページの『データ 項目』 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、55 ページの『mqExecute 呼び出しを使用したコマンド・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。

データ・バッグの書き込みおよび受信

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。 これにより、MQAI はアプリケーションではなくバッファーを処理します。 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、12 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。

データ 項目

データ項目は、作成時にデータ・バッグにデータを設定するために使用されます。 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、49 ページの『システム項目』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。

- 53 ページの『データ項目のカウント』.
- 54 ページの『データ項目の削除』.
- 49 ページの『バッグへのデータ項目の追加』.
- 50 ページの『データ項目のフィルター処理および照会』.

システム項目

システム項目は、次を実行するために使用できます。

- 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 (52 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。 (54 ページの『データ項目の削除』を参照してください。)

バッグへのデータ項目の追加

データ・バッグを作成すると、それにデータ項目を入れることができます。 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、49 ページの『データ 項目』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 50 ページの図 3 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。

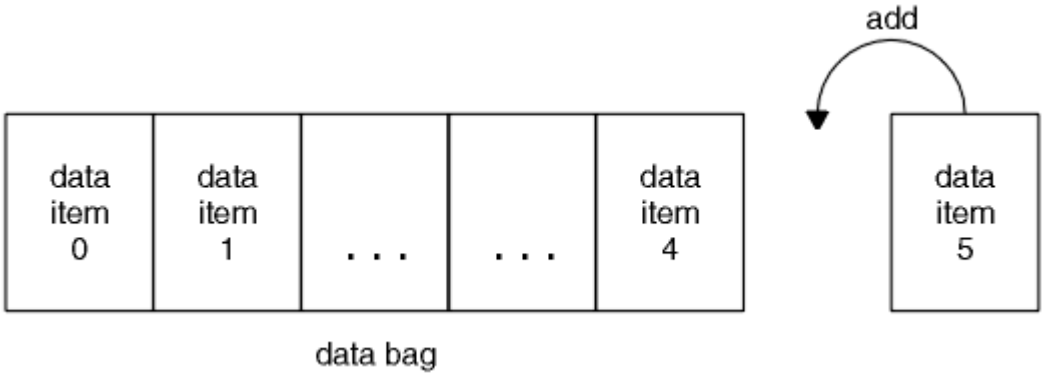


図 *3.* データ項目の追加

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 49 ページの『システム項目』を参照してください。

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、WebSphere 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。

データ項目のフィルター処理および照会

MQAI を使用して WebSphere 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、9 ページの『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, ...)
```

データ項目のフィルター処理および照会の他の例については、21 ページの『MQAI の使用例』を参照してください。

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。

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。 以下のように行えます。

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 (52 ページの図 4 を参照)。

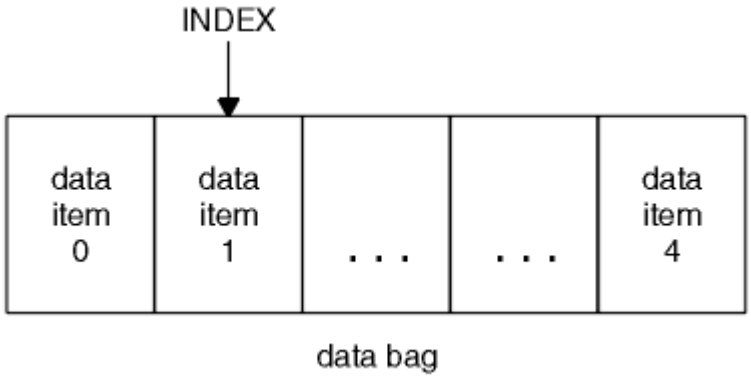


図 *4.* 単一データ項目の変更

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。 (52 ページの図 5 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。

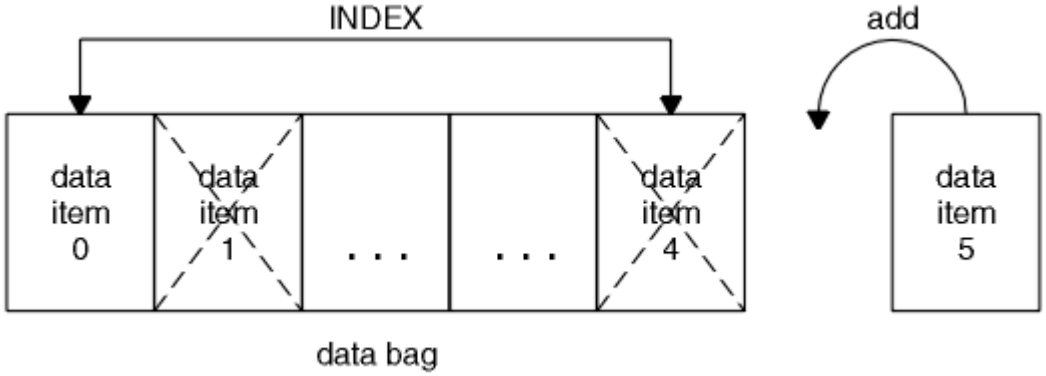


図 *5.* すべてのデータ項目の変更

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

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。

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。

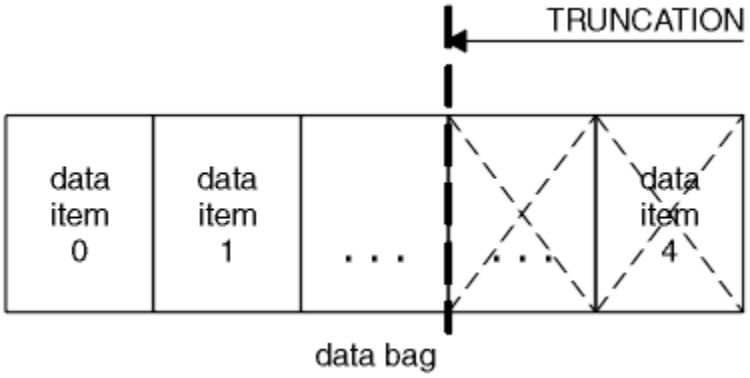


図 *6.* バッグの切り捨て

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。

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 53 ページの図 7 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。

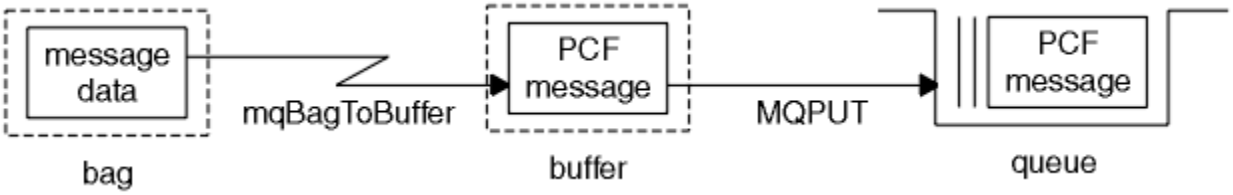


図 *7.* バッグの *PCF* メッセージへの変換

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 53 ページの図 8 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。

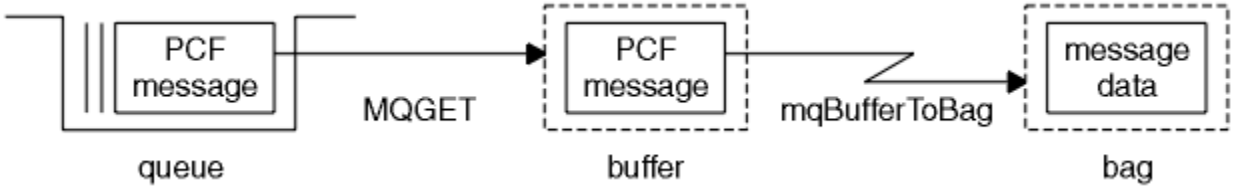


図 *8. PCF* メッセージのバッグ形式への変換

データ項目のカウント

mqCountItems 呼び出しは、データ・バッグに保管されているユーザー項目またはシステム項目、あるいはその両方の数をカウントし、その数を返します。 例えば、 `mqCountItems(Bag, 7, ...)`は、セレク

ターが 7 のバッグ内の項目数を返します。 これは、個々のセレクター別、ユーザー・セレクター別、システム・セレクター別、またはすべてのセレクター別に項目をカウントできます。

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

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

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。 以下のように行えます。

- 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、54 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、53 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、53 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。

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。 (54 ページの図 9 を参照してください。)

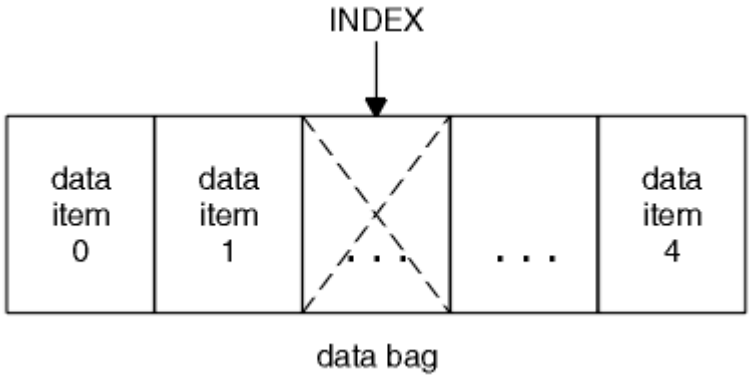


図 *9.* 単一データ項目の削除

または

2. 指定されたセレクターのすべてのオカレンス。 (54 ページの図 10 を参照してください。)

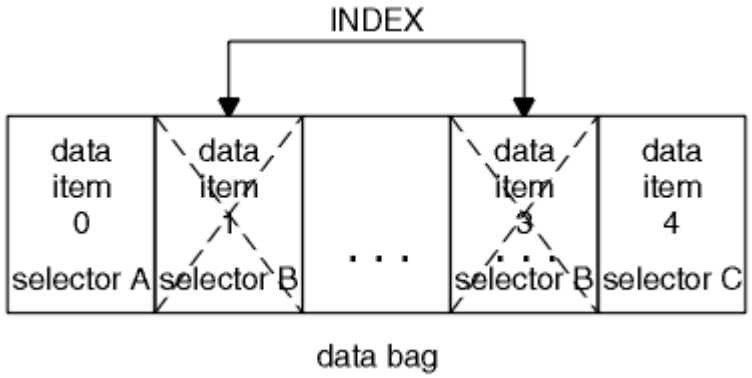


図 *10.* すべてのデータ項目の削除

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。

mqExecute 呼び出しを使用したコマンド・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 WebSphere 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。

55 ページの図 11 にネスト の概念を示します。

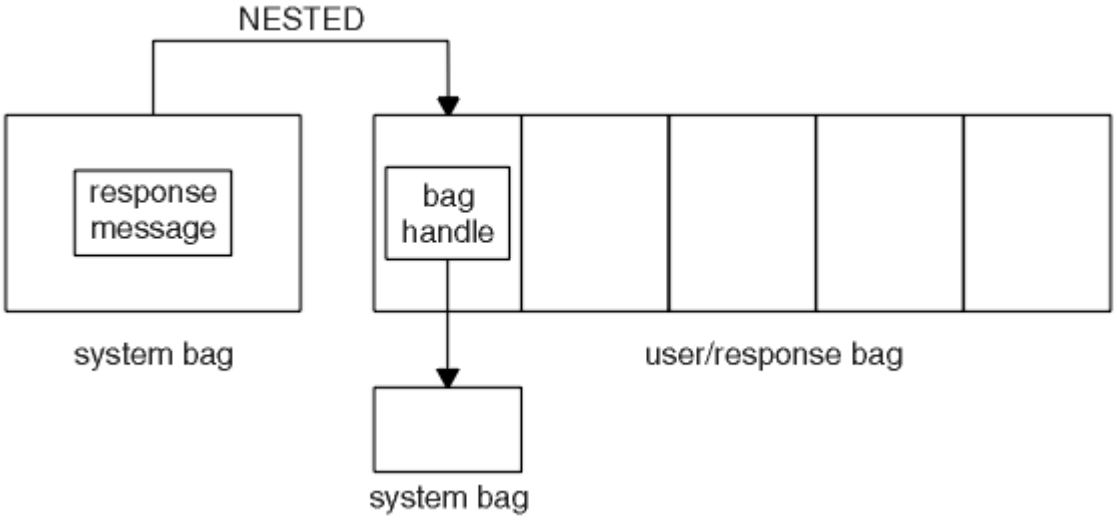


図 *11.* ネスト

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。

IBM WebSphere MQ Explorer による管理

IBM WebSphere MQ Explorer を使用すると、Windows、または Linux (x86 および x86-64 プラットフォーム) 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 WebSphere MQ for Windows、および IBM WebSphere MQ for Linux (x86 および x86-64 プラットフォーム) には、制御コマンドまたは MQSC コマンドの使用に代えて、管理タスクを実行するための IBM WebSphere 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 WebSphere 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 WebSphere MQ Explorer を使用すると、目的のキュー・マネージャーとクラスターを IBM WebSphere MQ Explorer でポイントすることによって、Windows、または Linux (x86-64 プラットフォーム) 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またはリモートで管理することができます。 IBM WebSphere MQ Explorer を使用して管理できる IBM WebSphere MQ のプラットフォームおよびレベルについては、 57 ページの『リモート・キュー・マネージャー』で説明しています。

IBM WebSphere MQ Explorer がリモート IBM WebSphere 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、 58 ページの『前提ソフトウェアおよび定義』を参照してください。

これにより、Windows または Linux (x86 および x86-64 プラットフォーム) システム・ドメイン内において、ローカルまたはリモートで、通常は IBM WebSphere MQ の作業環境の設定と微調整に関連付けられた作業を実行できます。

Linux では、複数の Eclipse がインストールされている場合、IBM WebSphere 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 WebSphere MQ Explorer を開始します。

Linux では、IBM WebSphere 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 WebSphere MQ Explorer を使用してできること

これは、IBM WebSphere MQ Explorer を使用して実行できるタスクのリストです。

IBM WebSphere MQ エクスプローラーでは、以下を実行できます。

- 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 WebSphere 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
- 「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
- Secure Sockets Layer (SSL) 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。
- ikeyman GUI を呼び出して SSL (Secure Sockets Layer) 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)。
- WebSphere MQ オブジェクトからの JMS オブジェクトの作成、および JMS オブジェクトからの WebSphere MQ オブジェクトの作成。
- 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。

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 WebSphere 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。

WebSphere MQ エクスプローラー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。

リモート・キュー・マネージャー

接続することのできるサポートされたキュー・マネージャーに 2 つの例外があります。

Windows または Linux (x86 および x86-64 プラットフォーム) システムから、 WebSphere MQ エクスプローラーは、以下の例外を除いて、サポートされるすべてのキュー・マネージャーに接続できます。

- バージョン 6.0 より前の WebSphere MQ for z/OS キュー・マネージャー。
- 現在サポートされている MQSeries® V2 キュー・マネージャー。

IBM WebSphere MQ エクスプローラーは、異なるコマンド・レベルとプラットフォーム間の機能の相違を処理します。 ただし、認識しない属性があると、その属性は不可視になります。

WebSphere MQ V5.3 コンピューター上の IBM WebSphere MQ エクスプローラーを使用して、Windows 上の V6.0 以降のキュー・マネージャーをリモート管理する場合は、 WebSphere MQ for Windows V5.3 コンピューターにフィックスパック 9 (CSD9) 以降をインストールする必要があります。

WebSphere MQ V6.0 以降のコンピューター上の WebSphere MQ エクスプローラーを使用して、 iSeries 上の V5.3 キュー・マネージャーをリモート管理する場合は、以下のようにします。 フィックスパック 11 (CSD11) 以降を WebSphere MQ for iSeries V5.3 コンピューターにインストールする必要があります。 このフィックスパックで、WebSphere MQ エクスプローラーと iSeries キュー・マネージャー間の接続の問題が修正されます。

IBM WebSphere MQ Explorer を使用するかどうかの決定

ご使用のシステムで IBM WebSphere 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 WebSphere MQ エクスプローラーを使用してキュー・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、WebSphere 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 WebSphere MQ エクスプローラーは、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 WebSphere MQ エクスプローラー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
WebSphere 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。 WebSphere MQ エクスプローラーは、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 WebSphere MQ エクスプローラー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 WebSphere MQ エクスプローラーがそれらに接続することはないからです。

IBM WebSphere MQ Explorer の設定

この項では、IBM WebSphere 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。

- 58 ページの『前提ソフトウェアおよび定義』
- 58 ページの『機密保護』
- 62 ページの『キュー・マネージャーとクラスターの表示と非表示』
- 63 ページの『クラスター・メンバーシップ』
- 64 ページの『データ変換』

前提ソフトウェアおよび定義

IBM WebSphere 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 WebSphere 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 WebSphere MQ リスナーでかまいません。または、UNIX 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。

機密保護

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 WebSphere MQ を使用する場合、IBM WebSphere MQ エクスプローラー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 WebSphere MQ Explorer を使用するための許可

任意のユーザーが IBM WebSphere 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。

WebSphere MQ エクスプローラー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。

WebSphere MQ エクスプローラー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、WebSphere MQ エクスプローラー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.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。

WebSphere MQ for z/OS 上のリモート・キュー・マネージャーに接続し、 IBM WebSphere MQ エクスプローラーを使用してリモート管理用タスクを実行するには、以下を提供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、WebSphere MQ エクスプローラー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

WebSphere MQ オブジェクトに権限を付与する方法については、 UNIX または Linux システムおよび Windows での WebSphere 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。

WebSphere MQ エクスプローラーのデフォルト・フィルターは、すべての WebSphere 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 >WebSphere 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。

リモート・キュー・マネージャーへの接続におけるセキュリティー

IBM WebSphere MQ エクスプローラーと各リモート・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 WebSphere MQ エクスプローラーは、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。

WebSphere MQ エクスプローラーでのセキュリティー出口の使用

WebSphere MQ エクスプローラーを使用して、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。

WebSphere MQ エクスプローラーから、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、WebSphere MQ エクスプローラーを使用して指定します。 詳細は、WebSphere MQ のヘルプ・センターを参照してください。

IBM WebSphere MQ Explorer による *SSL* 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 WebSphere 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。 SSL 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た MQI チャネルの確立方法については、IBM WebSphere MQ MQI クライアントの概要を参照してください。

60 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 61 ページの『IBM WebSphere MQ Explorer をホストするシステム上でのタスク』に説明されているように、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した場合は、IBM WebSphere MQ Explorer により、SSL 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. サーバー接続とクライアント接続のチャネルの対を定義し、両方のチャネルでのサーバー接続の *SSLCIPH* 変数に適切な値を指定します。 *SSLCIPH* 変数の詳細については、SSL 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 WebSphere MQ Explorer をホスティングす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 および SSL 個人証明書を、キュー・マネージャーの以下の SSL ディレクトリーに置きます。
   - UNIX and Linux システムの場合は、`/var/mqm/qmgrs/+QMNAME+/SSL`
   - Windows システムの場合は、`C:\Program Files\WebSphere 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 タイプのキー・データベース・ファイルを作成します。 iKeyman GUI でオプションにチェック・マークを付けるか、 **runmqckm** コマンドで `-stash` オプションを使用して、ファイル内のパスワードをスタッシュします。
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 Secure Sockets Layer を使用する方法について詳しくは、 UNIX、 Linux および Windows システムでの SSL または TLS の取り扱い を参照してください。

IBM WebSphere MQ Explorer をホストするシステム上でのタスク

IBM WebSphere 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 WebSphere MQ Explorer は、SSL 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 WebSphere MQ Explorer の SSL 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 システムおよび 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、MQ エクスプローラーを開始します。
5. IBM WebSphere MQ Explorer ツールバーから、「**ウィンドウ**」->「**設定**」をクリックし、「**WebSphere MQ エクスプローラー**」 を展開して、 「**SSL クライアント証明書ストア**」をクリックします。 61 ページの『IBM WebSphere 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 60 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 WebSphere 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。

別のキュー・マネージャーを経由した接続

IBM WebSphere MQ エクスプローラーを使うと、IBM WebSphere MQ エクスプローラー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 WebSphere MQ エクスプローラーが 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、 名前解決 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。

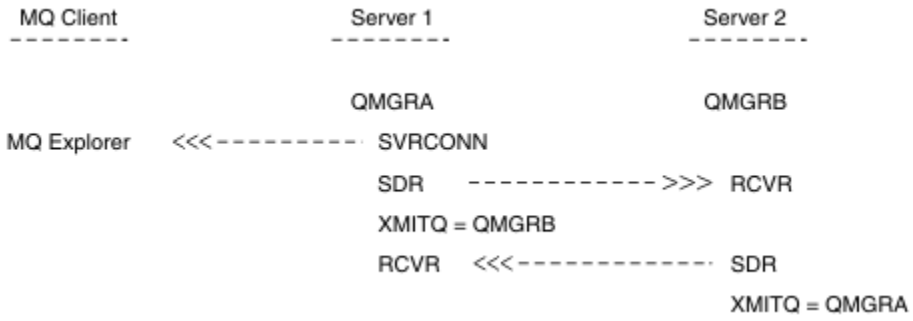
ターゲット・キュー・マネージャーのコマンド・サーバーは、メッセージ記述子 (MQMD) の *UserIdentifier* パラメーターにあるユーザー ID を指定して、メッセージを伝送キューに書き込みます。 この書き込みが正常に行われるには、そのユーザー ID が書き込み権限付きで、ターゲット・キュー・マネージャーにすでに存在している必要があります。

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 WebSphere MQ エクスプローラー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 WebSphere MQ エクスプローラーは、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 WebSphere MQ エクスプローラーに接続できるようになりました。
- QMGRB を WebSphere MQ エクスプローラーに接続するときは、中間キュー・マネージャーとして QMGRA を選択します。

この場合、 IBM WebSphere MQ エクスプローラーから 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。108 ページの『リモート管理のためにチャネルおよび伝送キューを作成する』を参照してください。

キュー・マネージャーとクラスターの表示と非表示

IBM WebSphere MQ エクスプローラーでは、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. Queue Managers ツリー・ノードを右クリックして、「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 WebSphere MQ エクスプローラー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. WebSphere MQ エクスプローラーのナビゲーション・ペインで、上部の **IBM WebSphere MQ** ツリー・ノードを右クリックし、「**エクスポート**」 **MQ エクスプローラー設定**」 を選択します。
3. 「**MQ エクスプローラー**」**>**「**MQ エクスプローラーの設定**」 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. WebSphere MQ エクスプローラーのナビゲーション・ペインの上部 **IBM WebSphere MQ** ツリー・ノードを右クリックして、「**インポート**」 **MQ エクスプローラー設定**」 を選択します。
2. 「**MQ エクスプローラー**」**>**「**MQ エクスプローラーの設定**」 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 WebSphere MQ エクスプローラーにインポートします。

クラスター・メンバーシップ

IBM WebSphere MQ エクスプローラーは、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 WebSphere MQ エクスプローラーの稼働中にキュー・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 WebSphere MQ エクスプローラーを保守して、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 WebSphere MQ エクスプローラー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、WebSphere MQ エクスプローラー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。 WebSphere MQ エクスプローラーは、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、WebSphere MQ エクスプローラー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 WebSphere MQ エクスプローラーはクライアント接続によってではなく、直接接続します。

データ変換

IBM WebSphere MQ エクスプローラーは、CCSID 1208 (UTF-8) で動作します。 これにより IBM WebSphere MQ エクスプローラーでは、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 WebSphere MQ エクスプローラーでは、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 WebSphere MQ エクスプローラーとキュー・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 WebSphere MQ エクスプローラー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 のセキュリティー

WebSphere MQ 準備ウィザードは、Windows サービスが、それを使用する必要のあるプロセス間で共有されるように、特別なユーザー・アカウントを作成します。

Windows サービスは、IBM WebSphere MQ インストール済み環境のクライアント・プロセス間で共有されます。 インストールごとに 1 つのサービスが作成されます。 各サービスは MQ_*InstallationName* という名前で、表示名は IBM WebSphere MQ(*InstallationName*) となります。 Version 7.1 より前では、1 つのサーバーに 1 つのインストール済み環境のみがある場合、 Windows サービスには、表示名 IBM MQSeries を持つ MQSeriesServices という名前が付けられていました。

各サービスは対話式および非対話式のログオン・セッション間で共有する必要があるため、それらを特別なユーザー・アカウントの下で起動する必要があります。 すべてのサービスに対して 1 つの特別なユーザー・アカウントを使用することも、個別の特別なユーザー・アカウントを作成することもできます。 それぞれの特別なユーザー・アカウントには、"「サービスとしてログオン」"するユーザー権限必要です。詳しくは、65 ページの『IBM WebSphere MQ Windows サービスに必要なユーザー権限』を参照してください。 ユーザー ID にサービスを実行する権限がない場合、サービスは開始されず、Windows システム・イベント・ログにエラーが返されます。 多くの場合、IBM WebSphere MQ 準備ウィザードを実行し、それによりユーザー ID が正しくセットアップされます。 ただし、ユーザー ID を手動で構成した場合は、問題が発生し、解決が必要になる可能性があります。

IBM WebSphere MQ をインストールして IBM WebSphere MQ 準備ウィザードを初めて実行すると、MUSR_MQADMIN というサービスのローカル・ユーザー・アカウントが作成され、必要な設定と権限 ("サービスとしてログオン"を含む) が設定されます。

以降のインストールでは、IBM WebSphere MQ の準備ウィザードによって、MUSR_MQADMINx という名前のユーザー・アカウントが作成されます。ここで x は、まだ存在しないユーザー ID を示す、次に使用可能な番号です。 MUSR_MQADMINx 用のパスワードは、アカウントの作成時にランダムに生成され、サービス用のログオン環境を構成するために使用されます。 生成されたパスワードは有効期限切れになりません。

一定期間が過ぎるとアカウントのパスワードの変更を必要とするアカウント・ポリシーがシステム上でセットアップされていても、この IBM WebSphere MQ アカウントはその影響を受けません。

このパスワードは、この一回限りの処理以外では使用されず、Windows オペレーティング・システムによって、レジストリーの安全な部分に保管されます。

Active directory の使用 (Windows のみ)

Active Directory を使用しているドメイン・コントローラー上にユーザー・アカウントが定義されている一部のネットワーク構成では、IBM WebSphere MQ を実行しているローカル・ユーザー・アカウントが、その他のドメイン・ユーザー・アカウントのグループ・メンバーシップを照会するために必要な権限を持っ

ていないことがあります。 IBM WebSphere MQ の準備ウィザードは、テストを実行し、ネットワーク構成についてユーザーに尋ねることにより、これを確認します。

IBM WebSphere MQ を実行しているローカル・ユーザー・アカウントが必要な権限を持っていない場合、IBM WebSphere MQ の準備ウィザードは、特定のユーザー権限を持つドメイン・ユーザー・アカウントのアカウント詳細について入力するようにユーザーに求めてきます。 ドメイン・ユーザー・アカウントが必要なユーザー権限については、65 ページの『IBM WebSphere MQ Windows サービスに必要なユーザー権限』を参照してください。 ユーザーがドメイン・ユーザー・アカウントの有効なアカウント詳細を IBM WebSphere MQ の準備ウィザードに入力すると、ウィザードは IBM WebSphere MQ Windows サービスを、新規アカウントの下で実行するように構成します。 アカウント詳細は、レジストリーのセキュア部分に保持され、ユーザーが読み取ることはできません。

サービスが稼働すると IBM WebSphere MQ Windows サービスも起動し、サービスが終了するまで稼働し続けます。 この Windows サービスの起動後にサーバーにログオンする IBM WebSphere MQ 管理者は、IBM WebSphere MQ Explorer を使用してサーバー上のキュー・マネージャーを管理することができます。 これによって、IBM WebSphere MQ Explorer と既存の Windows サービス・プロセスを接続できます。 ただし、これら 2 つの処理の実行には、次に示すように、それぞれ異なるレベルの許可が必要です。

- 起動プロセス: 起動許可
- IBM WebSphere MQ 管理者: アクセス権

IBM WebSphere MQ Windows サービスに必要なユーザー権限

このトピックの表では、IBM WebSphere MQ インストール済み環境向けの Windows サービスを実行するために、ローカルおよびドメイン・ユーザー・アカウントに必要なユーザー権限をリストしています。

バッチ・ジョブとしてログオン	IBM WebSphere MQ Windows サービスをこのユーザー・アカウントで実行できるようにします。
サービスとしてログオン	ユーザーは、IBM WebSphere MQ Windows サービスを設定し、構成済みアカウントを使用してログオンできる。
システムをシャットダウン	サービスのリカバリーが失敗したときにシステムをシャットダウンするように構成されている場合、IBM WebSphere MQ Windows サービスはサーバーを再始動できる。
割り当て量の増加	オペレーティング・システムの `CreateProcessAsUser` 呼び出しに必要。
オペレーティング・システムの一部として動作する	オペレーティング・システムの `LogonUser` 呼び出しに必要。
全探索検査の迂回	オペレーティング・システムの `LogonUser` 呼び出しに必要。
処理レベル・トークンの置換	オペレーティング・システムの `LogonUser` 呼び出しに必要。

注 : ASP および IIS アプリケーションを実行する環境では、「プログラムのデバッグ」権限が必要になる可能性があります。

ご使用のドメイン・ユーザー・アカウントには、これらの Windows ユーザー権限が、ローカル・セキュリティー・ポリシー・アプリケーションにリストされるのと同様に有効なユーザー権限として設定されている必要があります。 そうでない場合は、ローカル・セキュリティー・ポリシー・アプリケーションをサーバー上でローカルに使用するか、またはドメイン・セキュリティー・アプリケーション・ドメイン全体を使用して、ユーザー権限を設定します。

IBM WebSphere MQ サービスと関連したユーザー名の変更

IBM WebSphere MQ サービスと関連したユーザー名を MUSR_MQADMIN からその他の名前に変更する必要がある場合があります。 (例えば、キュー・マネージャーが DB2® と関連がある場合、8 文字を超えるユーザー名は受諾されないので、この変更を行う必要があります。)

手順

1. 新規ユーザー・アカウントを作成します (例えば **NEW_NAME**)。
2. IBM WebSphere MQ の準備ウィザードを使用して、新規ユーザー・アカウントの詳細を入力します。

IBM WebSphere MQ Windows サービス・ユーザー・アカウントのパスワードの変更

このタスクについて

IBM WebSphere MQ Windows サービスのローカル・ユーザー・アカウントのパスワードを変更するには、以下のステップを実行します。

手順

1. サービスを実行しているユーザーを識別します。
2. 「コンピュータの管理」パネルから、IBMIBM WebSphere MQ のサービスを停止します。
3. 個人のパスワードを変更する場合と同じようにして、必要なパスワードを変更します。
4. 「コンピュータの管理」パネルから、IBM WebSphere MQ サービスのプロパティーに移動します。
5. 「**ログオン**」ページを選択します。
6. 指定したアカウント名が、パスワードが変更されたユーザーと一致していることを確認します。
7. 「**パスワード**」フィールドおよび「**確認パスワード**」フィールドにパスワードを入力し、「**OK**」をクリックします。

ドメイン・ユーザー・アカウントの下で実行されているインストール済み環境の *IBM WebSphere MQ Windows* サービス

このタスクについて

インストール済み環境の IBM WebSphere MQ Windows サービスがドメイン・ユーザー・アカウントの下で実行されている場合、以下のようにアカウントのパスワードを変更することもできます。

手順

1. ドメイン・コントローラー上でドメイン・アカウントのパスワードを変更します。 パスワードを変更するには、ドメイン管理者に問い合わせる必要があります。
2. ステップに従って、IBMIBM WebSphere MQ サービスの「**ログオン**」ページを変更します。

   ユーザー・インターフェース・アプリケーションから発行される MQSC コマンドや、システムの始動、シャットダウン、またはサービスのリカバリー時に自動的に実行される MQSC コマンドはすべて、IBM WebSphere MQ Windows サービスを実行しているユーザー・アカウントで実行されます。 したがって、このユーザー・アカウントは、IBM WebSphere MQ 管理権限を持っていなければなりません。 デフォルトでは、このユーザー・アカウントは、サーバー上のローカル **mqm** グループに追加されます。 このメンバーシップが削除されると、IBM WebSphere MQ Windows サービスは機能しなくなります。 ユーザー権限について詳しくは、65 ページの『IBM WebSphere MQ Windows サービスに必要なユーザー権限』を参照してください。

   IBM WebSphere MQ Windows サービスを実行するユーザー・アカウントでセキュリティー上の問題が発生した場合、システムのイベント・ログにエラー・メッセージと説明が書き込まれます。

関連概念

64 ページの『Active directory の使用 (Windows のみ)』
Active Directory を使用しているドメイン・コントローラー上にユーザー・アカウントが定義されている一部のネットワーク構成では、IBM WebSphere MQ を実行しているローカル・ユーザー・アカウントが、その他のドメイン・ユーザー・アカウントのグループ・メンバーシップを照会するために必要な権限を持っていないことがあります。 IBM WebSphere MQ の準備ウィザードは、テストを実行し、ネットワーク構成についてユーザーに尋ねることにより、これを確認します。

リソース・マネージャーとしての Db2 との IBM WebSphere MQ の調整

IBM WebSphere MQ Explorer からキュー・マネージャーを開始するとき、または IBM WebSphere MQ V7 を使用しているときに、Db2 の調整で問題が生じた場合は、キュー・マネージャーのエラー・ログを調べます。

以下のようなエラーが発生していないか、キュー・マネージャーのエラー・ログを調べてください。

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
Host(HOST_1) Installation(Installation1)
VMRF(7.1.0.0) QMgr(A.B.C)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
    for xa_open. The queue manager is continuing without this resource manager.
```

説明: IBM WebSphere MQ サービス・プロセス `amqsvc.exe` を実行するユーザー ID (デフォルトの名前は MUSR_MQADMIN) が、グループ DB2USERS に関するグループ・メンバーシップ情報を含んでいないアクセス・トークンで引き続き実行されています。

解決方法: IBM WebSphere MQ サービスのユーザー ID が DB2USERS のメンバーであることを確認してから、以下の一連のコマンドを使用します。

- サービスを停止します。
- 同じユーザー ID で実行されている他のプロセスを停止します。
- これらのプロセスを再開します。

マシンをリブートすれば、ここまでのステップは確実に行われますが必要ありません。

IBM WebSphere MQ Explorer の拡張

IBM WebSphere MQ for Windows、および IBM WebSphere MQ for Linux (x86 および x86-64 プラットフォーム) には、制御コマンドまたは MQSC コマンドを使用する代わりに管理タスクを実行するための IBM WebSphere MQ Explorer という管理インターフェースが用意されています。

この情報は、WebSphere MQ for Windows、および WebSphere MQ for Linux (x86 および x86-64 プラットフォーム) にのみ適用されます。

IBM WebSphere MQ エクスプローラーでは、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 WebSphere MQ エクスプローラーを拡張すると、システム管理者は、WebSphere MQ エクスプローラーをカスタマイズして WebSphere MQ を管理する方法を改善できます。

詳しくは、 IBM WebSphere MQ Explorer 製品資料の「*IBM WebSphere MQ Explorer* の拡張」を参照してください。

IBM WebSphere MQ Taskbar アプリケーションの使用 (Windows のみ)

IBM WebSphere 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 WebSphere 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、WebSphere 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。WebSphere 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、WebSphere 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、 WebSphere MQ アラート・モニターを開きます。
- 「**終了**」 をクリックして、 WebSphere MQ Taskbar アプリケーションを終了します。
- **WebSphere MQ エクスプローラー**」 をクリックして、 IBM WebSphere MQ エクスプローラーを開始します。
- WebSphere MQ を停止するには、 「**停止**」 **WebSphere MQ** をクリックします。
- 「**バージョン情報**」 **WebSphere MQ** をクリックして、 WebSphere MQ アラート・モニターに関する情報を表示します。

IBM WebSphere MQ アラート・モニター・アプリケーション (Windows のみ)

IBM WebSphere MQ アラート・モニターは、ローカル・マシン上の IBM WebSphere 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、WebSphere 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。

WebSphere MQ アラート・モニターでは、以下のことが可能です。

- IBM WebSphere MQ エクスプローラー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 WebSphere 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 WebSphere MQ オブジェクトの管理

このセクションでは、メッセージ・キュー・インターフェース (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 WebSphere MQ オブジェクトの管理方法を説明します。 ここでは、ローカル管理とは、IBM WebSphere 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。

このセクションで詳しく説明するアプローチのほかに、IBM WebSphere MQ エクスプローラーを使用して、ローカル WebSphere MQ オブジェクトを管理することもできます。56 ページの『IBM WebSphere MQ Explorer による管理』を参照してください。

このセクションでは、次の内容を取り上げます。

- MQI を使用するアプリケーション・プログラム
- 72 ページの『MQSC コマンドによるローカル管理タスクの実行』
- 81 ページの『キュー・マネージャーの処理』
- 83 ページの『ローカル・キューの処理』
- 88 ページの『別名キューの処理』
- 90 ページの『モデル・キューの処理』
- 97 ページの『サービスの取り扱い』
- 103 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・マネージャーの開始および停止

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。

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、以下のように strmqm コマンドを使用します。

```
strmqm saturn.queue.manager
```

WebSphere MQ for Windows および WebSphere MQ for Linux (x86 および x86-64 プラットフォーム) システムでは、以下のようにしてキュー・マネージャーを開始できます。

1. IBM WebSphere MQ エクスプローラーを開きます。
2. 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。
3. Start をクリックします。 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 WebSphere MQ から断続的に発行されます。

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。

キュー・マネージャーを自動的に開始する

WebSphere MQ for Windows では、WebSphere MQ エクスプローラー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、56 ページの『IBM WebSphere MQ Explorer による管理』を参照してください。

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。 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。

例えば、QMB という名前のキュー・マネージャーを停止するには、次のコマンドを入力します。

```
endmqm QMB
```

WebSphere MQ for Windows および WebSphere MQ for Linux (x86 および x86-64 プラットフォーム) システムでは、以下のようにしてキュー・マネージャーを停止できます。

1. IBM WebSphere MQ エクスプローラーを開きます。
2. 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。
3. Stop... 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。
4. 「制御」または「即時」を選択します。
5. OK をクリックします。 キュー・マネージャーが停止します。

静的シャットダウン

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。 これは完了するまでに多少の時間がかかります。 静的シャットダウンは、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です。

このタイプのシャットダウンを使用して、アプリケーションに対して停止するよう通知します。 次のコマンドを発行する場合:

```
endmqm -c QMB
```

すべてのアプリケーションが停止されたのがどの時点かは通知されません。 (endmqm -c QMB コマンドは、 endmqm QMB コマンドと同等です。)

ただし、次のコマンドを発行する場合:

```
endmqm -w QMB
```

すべてのアプリケーションが停止し、キュー・マネージャーが終了するまで、コマンドは待機します。

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。

即時シャットダウンの場合、次のように入力します。

```
endmqm -i QMB
```

プリエンプティブ・シャットダウン

注 : この方法は、**endmqm** コマンドを使用してもキュー・マネージャーを停止できなかったときにのみ使用してください。 この方法は、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。

即時シャットダウンが機能しない場合は、-p フラグを指定したプリエンプティブ・シャットダウンを用いる必要があります。 以下に例を示します。

```
endmqm -p QMB
```

これにより、キュー・マネージャーが即時に停止します。 この方法でも解決しない場合には、別の方法として 70 ページの『手動によるキュー・マネージャーの停止』 を参照してください。

endmqm コマンドとそのオプションの詳細については、endmqm を参照してください。

キュー・マネージャーのシャットダウンに問題がある場合

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フラグを付加します。

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を行えなかった場合には、ここで説明する手段で再度行ってみてください。

endmqm コマンドを使用してキュー・マネージャーを停止する通常の方法です。 キュー・マネージャーを手動で停止するには、このセクションで説明されているいずれかの手順を使用してください。 制御コマンドを使用したキュー・マネージャーに対する操作の実行方法について詳しくは、キュー・マネージャーの作成および管理を参照してください。

Windows でのキュー・マネージャーの停止

IBM WebSphere MQ for Windows でキュー・マネージャーを停止するために、プロセスおよび IBM WebSphere MQ サービスを終了する方法について説明します。

WebSphere MQ for Windows で実行しているキュー・マネージャーを停止するには、次の手順に従います。

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、プロセスを以下の順序で終了します (実行中の場合)。

AMQZMUC0	重要なプロセス・マネージャー
AMQZXMA0	実行コントローラー
AMQZFUMA	OAM プロセス
AMQZLAA0	LQM エージェント
AMQZLSA0	LQM エージェント
AMQZMUF0	ユーティリティー・マネージャー
AMQZMGR0	プロセス・コントローラー
AMQZMUR0	再開始可能なプロセス・マネージャー
AMQFQPUB	パブリッシュ・サブスクライブ・プロセス
AMQFCXBA	ブローカー・ワーカー・プロセス
AMQRMPPA	プロセス・プール・プロセス
AMQCRSTA	非スレッド化応答側ジョブ・プロセス
AMQCRS6B	LU62 受信側チャネルおよびクライアント接続
AMQRRMFA	リポジトリー・プロセス (クラスターの)
AMQZDMAA	据え置きメッセージ・プロセッサー
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
AMQZTRCN	トレース

3. Windows の「コントロールパネル」で、 **管理ツール** > 「**サービス**」 から WebSphere 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。

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。 特定のキュー・マネージャーに関連するプロセスを判別するのに役立つ詳細情報については、Microsoft の Web サイト (https://www.microsoft.com) からダウンロードできる *Process Explorer* (procexp.exe) などのツールを使用することを検討してください。

UNIX and Linux システムでのキュー・マネージャーの停止

IBM WebSphere MQ for UNIX and Linux でキュー・マネージャーを停止するために、プロセスおよび IBM WebSphere MQ サービスを終了する方法。 標準的な方法でキュー・マネージャーの停止および除去を行えなかった場合には、ここで説明する手段で試行できます。

WebSphere MQ for UNIX and Linux システムで実行しているキュー・マネージャーを停止するには、次の手順に従います。

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. 実行中のキュー・マネージャー・プロセスをすべて終了します。 ps コマンドを使用してディスカバーされたプロセス ID を指定して、 kill コマンドを使用します。

   次の順序でプロセスを終了します。

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)
amqzdmaa	据え置きメッセージ・プロセッサー
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。

キュー・マネージャーを手動で停止すると、FFST が取得され、 /var/mqm/errors. に FDC ファイルが配置される可能性があります。これは、キュー・マネージャーの障害とは見なされません。

この方法で終了した後でも、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。

MQSC コマンドによるローカル管理タスクの実行

このセクションでは、MQSC コマンドについて紹介し、いくつかの共通タスクのために MQSC コマンドを使用する方法について説明します。

IBM WebSphere MQ for Windows または IBM WebSphere MQ for Linux (x86 および x86-64 プラットフォーム) を使用する場合は、 IBM WebSphere MQ エクスプローラーを使用して、このセクションで説明されている操作を実行することもできます。 詳細については、56 ページの『IBM WebSphere MQ Explorer による管理』を参照してください。

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。 このセクションでは、キュー・

マネージャー、キュー、およびプロセス定義について説明します。チャネル、クライアント接続チャネル、およびリスナー・オブジェクトの管理については、オブジェクト を参照してください。 キュー・マネージャー・オブジェクトを管理するためのすべての MQSC コマンドについては、73 ページの『スクリプト (MQSC) コマンド』を参照してください。

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。 (このコマンドの詳細については、runmqsc を参照してください。) 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。 (テキスト・ファイルからのコマンドの実行については、76 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。)

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 とおりの方法で実行できます。

- コマンドを実行せずにでコマンドを検証する。この方法では、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、実行されません。
- ローカル・キュー・マネージャー上でコマンドを実行する。この方法では、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。
- リモート・キュー・マネージャー上でコマンドを実行する。この方法では、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。

構文を表示するには、コマンドの後に疑問符を付けて実行することもできます。

このセクションでは、MQSC コマンドで指定するオブジェクト属性を (RQMNAME のように) 大文字で表記していますが、実際には大/小文字の区別はありません。 MQSC コマンド属性名は 8 文字までに制限されています。 MQSC コマンドは、 IBM i および z/OS などの他のプラットフォームで使用できます。

MQSC コマンドについては、MQSC 参照セクションに含まれている複数のトピックで要約しています。

スクリプト (MQSC) コマンド

MQSC コマンドには、WebSphere MQ プラットフォームで人間が理解できるコマンドを発行する統一された方式があります。 プログラム式コマンド形式 (PCF) コマンドについて詳しくは、9 ページの『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。

このコマンドの一般的な形式は、MQSC コマンドに示されています。

MQSC コマンドを使用する場合は、以下の規則に従う必要があります。

- 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。
- 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。 (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。
- ブランク、小文字、または次の特殊文字以外の特殊文字を含むストリングは、単一引用符で囲む必要があります。

- ピリオド (.)
- 順方向斜線 (/)
- 下線 (_)
- パーセント記号 (%)

ただし、次の場合を除きます。

- 末尾がアスタリスクの総称値
- アスタリスク 1 つ (例、TRACE(*))
- コロンを含む範囲指定 (例、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。

- z/OS 以外の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、間にスペースのない 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。 ただし、使用されている属性が以下のいずれかである場合は例外です。
  - TOPICSTR
  - SUB
  - USERDATA
  - SELECTOR

  この場合、間にスペースのない 2 つの単一引用符はゼロ長ストリングとして解釈されます。
- v7.0 では、MQCHARV タイプに基づくストリング属性 (SELECTOR または SUBUSERDATA など) 内の末尾ブランクは、すべて有効なものとして扱われます。つまり、'abc ' は 'abc' と同じではありません。
- 左括弧の直後に右括弧が続いて、間に有意な情報がない場合、例えば、

```
NAME ( )
```

  などは、特に注記がある場合を除き無効です。
- 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。

IBM WebSphere MQ 管理のための MQSC コマンド使用の概要については、72 ページの『MQSC コマンドによるローカル管理タスクの実行』を参照してください。

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。 それらの特殊文字およびその使用法について詳しくは、特別な意味を持つ文字を参照してください。

MQSC コマンドを使用してスクリプトを作成する方法について調べるには、コマンド・スクリプトの作成を参照してください。

MQSC コマンドの完全なリストは、MQSC コマンドを参照してください。

関連タスク
コマンド・スクリプトの作成

WebSphere MQ オブジェクト名

MQSC コマンドでオブジェクト名を使用する方法。

例では、いくつかの長い名前をオブジェクトに使用しています。 これは、取り扱うオブジェクトのタイプの識別に役立ちます。

MQSC コマンドを出す場合に必要なのは、キューのローカル名の指定のみです。 例では、次のようなキュー名を使用しています。

```
ORANGE.LOCAL.QUEUE
```

この名前の LOCAL.QUEUE という部分は、このキューがローカル・キューであることを示すためのものです。 一般に、ローカル・キューの名前に、この部分は**必要ではありません**。

キュー・マネージャー名として saturn.queue.manager という名前も使用しています。 この名前の queue.manager という部分は、このオブジェクトがキュー・マネージャーであることを示すためのものです。 一般に、キュー・マネージャーの名前に、この部分は必要ではありません。

MQSC コマンドでの大/小文字の区別

MQSC コマンドは、属性も含めて、大文字または小文字のどちらで書いても構いません。 MQSC コマンドの中のオブジェクト名は、単一引用符で囲まれていない限り、大文字変換されます (つまり、QUEUE と queue の区別は付けられません)。 引用符を使用しないと、オブジェクトは大文字の名前で処理されます。 詳しくは、MQSC リファレンス を参照してください。

runmqsc コマンドの呼び出しは、すべての WebSphere MQ 制御コマンドと同様に、一部の WebSphere MQ 環境では大/小文字が区別されます。 詳細については、制御コマンドの使用を参照してください。

標準入出力

標準入力装置 (stdin とも呼ばれます) は、システムへの入力が行われる装置のことです。 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。 標準出力装置 (stdout とも呼ばれます) は、システムからの出力が送られる装置のことです。 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。

オペレーティング・システム・コマンドおよび WebSphere MQ 制御コマンドでは、< 演算子は入力をリダイレクトします。 この演算子の後にファイル名がある場合は、入力はそのファイルから行われます。 同様に、> 演算子は出力をリダイレクトします。この演算子の後にファイル名がある場合には、出力はそのファイルに送られます。

対話式の MQSC コマンドの使用

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。

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、次のように入力します。

```
runmqsc
```

このコマンドでは、キュー・マネージャー名が指定されていません。したがって、MQSC コマンドは、デフォルト・キュー・マネージャーによって処理されます。 異なるキュー・マネージャーを使用する場合には、**runmqsc** コマンド上にキュー・マネージャー名を指定してください。 例えば、キュー・マネージャー jupiter.queue.manager に対して MQSC コマンドを実行するには、次のコマンドを使用します。

```
runmqsc jupiter.queue.manager
```

この後、入力するすべての MQSC コマンドは、このキュー・マネージャーによって処理されます。ただし、このキュー・マネージャーが同じノード上にあって、既に実行されている場合に限ります。

この時点で、必要に応じて任意の MQSC コマンドを入力できます。 例えば、次のコマンドを試してください。

```
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。 (これは、コマンド入力の最終行の末尾で誤って連結文字を入力してしまった場合などに特に役立ちます。)

MQSC コマンドからのフィードバック

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。 以下に例を示します。

```
AMQ8006: WebSphere MQ queue created.
```

このメッセージは、キューが作成されたことを確認するものです。

```
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。

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、MQSC リファレンスを参照して正しい構文を確認してください。

対話式での MQSC コマンドへの入力を終了する

MQSC コマンドの処理を停止するには、END コマンドを入力します。

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。

テキスト・ファイルからの MQSC コマンドの実行

MQSC コマンドを対話式で実行する方法は、迅速にテストを行うのに適していますが、非常に長いコマンドや、繰り返し実行するコマンドの場合には、stdin をテキスト・ファイルからリダイレクトする方法を検討してください。

75 ページの『標準入出力』には、stdin および stdout についての情報が含まれています。 stdin をテキスト・ファイルからリダイレクトするには、最初に MQSC コマンドの入ったテキスト・ファイルを、通常のテキスト・エディターを使用して作成します。 runmqsc コマンドを使用するとき、シェル・リダイレ

クト演算子を使用してください。 例えば、次のコマンドは、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 < myprog.in
```

同様にして、出力をファイルにリダイレクトすることもできます。 入力用の MQSC コマンドが格納されているファイルを *MQSC* コマンド・ファイル と呼びます。 キュー・マネージャーからの応答が格納されている出力ファイルを出力ファイル と呼びます。

runmqsc コマンドで stdin と stdout の両方をリダイレクトするためには、次の形式のコマンドを使用します。

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

このコマンドは、MQSC コマンド・ファイル myprog.in. に含まれる MQSC コマンドを呼び出します。キュー・マネージャー名が指定されていないため、MQSC コマンドはデフォルトのキュー・マネージャーに対して実行されます。 出力はテキスト・ファイル myprog.out に送られます。77 ページの図 12 は、MQSC コマンド・ファイル myprog.in および 78 ページの図 13 からの抽出を示しています。これに対応する myprog.out の出力の抽出が示されています。

デフォルトのものではないキュー・マネージャー (saturn.queue.manager) について、runmqsc コマンドで stdin および stdout をリダイレクトするために、次の形式のコマンドを使用します。

```
runmqsc saturn.queue.manager < myprog.in > myprog.out
```

MQSC コマンド・ファイル

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。77 ページの図 12 属性を持つ MQSC コマンド (DEFINE QLOCAL) を示す MQSC コマンド・ファイルからの抜粋です。 MQSC リファレンスには、各 MQSC コマンドおよびその構文の説明が含まれています。

```
 .
 .
 .
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 *12. MQSC* コマンド・ファイルからの抽出

WebSphere 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 コマンド・レポート

runmqsc コマンドはレポート を戻し、これは stdout に送られます。 レポートには、次のものが含まれています。

- MQSC コマンドをレポートのソースとして識別するヘッダー。

```
Starting MQSC for queue manager jupiter.queue.manager.
```

  ここで、 jupiter.queue.manager はキュー・マネージャーの名前です。
- 発行された MQSC コマンドの番号付きリスト (任意)。 デフォルトでは、入力のテキストが出力にエコーされます。 この出力の中では、78 ページの図 13 に示されているように、各コマンドには順序番号が先頭に付けられています。 ただし、runmqsc コマンドに -e フラグを指定すると、この出力を抑制することができます。
- エラーのあったコマンドについての構文エラー・メッセージ。
- 各コマンドの実行結果を示すオペレーター・メッセージ。 例えば、DEFINE QLOCAL コマンドの正常終了を示すオペレーター・メッセージは、次のとおりです。

```
AMQ8006: WebSphere MQ queue created.
```

- スクリプト・ファイルの実行時の一般エラーのために出されるその他のメッセージ。
- 読み取られたコマンドの数、構文エラーのあったコマンドの数、処理できなかったコマンドの数を示す、レポートの簡単な統計の要約。

  注: キュー・マネージャーが処理を試みるのは、構文エラーのなかったコマンドのみです。

```
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: WebSphere MQ queue created.
      :
 .
 .
```

図 *13. MQSC* コマンド・レポート・ファイルからの抽出

システムに提供された MQSC コマンド・ファイルを実行する

以下の MQSC コマンド・ファイルが WebSphere MQ に付属しています。

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

WebSphere MQ for Windows では、これらのファイルは *MQ_INSTALLATION_PATH*\tools\mqsc\samples のディレクトリーにあります。 *MQ_INSTALLATION_PATH* WebSphere MQ がインストールされている上位ディレクトリーを表します。

UNIX and Linux システムでは、これらのファイルは *MQ_INSTALLATION_PATH*/samp のディレクトリーにあります。 *MQ_INSTALLATION_PATH* WebSphere MQ がインストールされている上位ディレクトリーを表します。

次のコマンドが実行されました。

```
runmqsc < amqscos0.tst >test.out
```

runmqsc を使用してコマンドを確認する

runmqsc コマンドを使用すると、MQSC コマンドを実際に実行しなくても、ローカル・キュー・マネージャーでそれらのコマンドを確認することができます。 これを確認するには、例えば次のように、-v フラグを runmqsc コマンドに設定します。

```
runmqsc -v < myprog.in > myprog.out
```

MQSC コマンド・ファイルに対して runmqsc を呼び出すと、キュー・マネージャーは、実際に MQSC コマンドを実行することなく、各コマンドを確認してレポートを戻します。 これによって、コマンド・ファイル内のコマンドの構文を検査できます。 これは、次のような場合に特に重要です。

- コマンド・ファイルから多数のコマンドを実行する場合
- MQSC コマンド・ファイルを複数回繰り返して使用する場合

戻されるレポートは、78 ページの図 13 に示されているものと同様です。

リモートから MQSC コマンドを確認するには、この方法を用いることはできません。 例えば、次のコマンドを試行するとします。

```
runmqsc -w 30 -v jupiter.queue.manager < myprog.in > myprog.out
```

キュー・マネージャーがリモートであることを示すために使用する -w フラグは無視され、コマンドは検証モードでローカルで実行されます。 30 は、WebSphere MQ がリモート・キュー・マネージャーからの応答を待つ秒数です。

バッチ・ファイルからの MQSC コマンドの実行

非常に長いコマンドや、繰り返し実行するコマンドの場合には、stdin をバッチ・ファイルからリダイレクトする方法を検討してください。

stdin をバッチ・ファイルからリダイレクトするには、最初に MQSC コマンドの入ったバッチ・ファイルを、通常のテキスト・エディターを使用して作成します。 runmqsc コマンドを使用するとき、シェル・リダイレクト演算子を使用してください。 以下に例を示します。

1. テスト・キュー・マネージャー TESTQM を作成します。
2. TCP/IP ポート 1600 を使用するために、一致する CLNTCONN およびリスナーのセットを作成します。
3. テスト・キュー TESTQ を作成します。
4. amqsputc サンプル・プログラムを使用して、メッセージをキューに入れます。

```
export MYTEMPQM=TESTQM
export MYPORT=1600
export MQCHLLIB=/var/mqm/qmgrs/$MQTEMPQM/@ipcc

crtmqm $MYTEMPQM
strmqm $MYTEMPQM
runmqlsr -m $MYTEMPQM -t TCP -p $MYPORT &

runmqsc $MYTEMPQM << EOF
  DEFINE CHANNEL(NTLM) CHLTYPE(SVRCONN) TRPTYPE(TCP)
  DEFINE CHANNEL(NTLM) CHLTYPE(CLNTCONN) QMNAME('$MYTEMPQM') CONNAME('hostname($MYPORT)')
  ALTER  CHANNEL(NTLM) CHLTYPE(CLNTCONN)
  DEFINE QLOCAL(TESTQ)
EOF

amqsputc TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM
```

図 *14.* バッチ・ファイルから *MQSC* コマンドを実行するためのスクリプトの例

MQSC コマンドで起こった問題の解決

MQSC コマンドを実行できない場合は、このトピックにある情報を使用して、共通問題に該当するものがないか確認してください。 コマンドが生成するエラーを読んでも、問題は必ずしも明らかにはなりません。

MQSC コマンドを実行できない場合は、次の情報に記載された共通問題に該当するものがないか確認してください。 生成されたエラーを読んでも、問題は必ずしも明らかにはなりません。

runmqsc コマンドを使用するとき、次の点に注意してください。

- **<** オペレーターを使用して、ファイルからの入力をリダイレクトします。 この演算子を省略すると、キュー・マネージャーはそのファイル名をキュー・マネージャー名として解釈し、次のエラー・メッセージを発行します。

  ```
  AMQ8118: WebSphere MQ queue manager does not exist.
  ```

- 出力をファイルにリダイレクトする場合には、**>** リダイレクト演算子を使用します。 デフォルトでは、ファイルは runmqsc が呼び出される時点で現行の作業ディレクトリーに入れられます。 出力を特定のファイルおよびディレクトリーに送るには、完全修飾ファイル名を指定してください。
- 以下のコマンドを使用して、すべてのキュー・マネージャーを表示し、コマンドを実行するキュー・マネージャーが作成されているかどうか確認します。

  ```
  dspmq
  ```

- キュー・マネージャーが実行中でなければなりません。 実行されていない場合は、開始してください (キュー・マネージャーの開始を参照)。 既に実行されているキュー・マネージャーを開始しようとする場合、エラー・メッセージを受け取ります。
- デフォルト・キュー・マネージャーが定義されていない場合、runmqsc コマンドにキュー・マネージャー名を指定します。 これを定義しなければ、次のようなエラー・メッセージが出されます。

  ```
  AMQ8146: WebSphere MQ queue manager not available.
  ```

- MQSC コマンドを runmqsc コマンドのパラメーターとして指定することはできません。 例えば、次のものは無効です。

  ```
  runmqsc DEFINE QLOCAL(FRED)
  ```

- runmqsc コマンドを発行する前に、MQSC コマンドを入力することはできません。
- runmqsc から制御コマンドを実行することはできません。 例えば、MQSC コマンドを対話式で実行している間は、strmqm コマンドを発行してキュー・マネージャーを開始することはできません。 開始する場合、以下のものと類似したエラー・メッセージを受け取ります。

```
runmqsc
  .
  .
Starting MQSC for queue manager jupiter.queue.manager.

    1 : strmqm saturn.queue.manager
AMQ8405: Syntax error detected at or near end of cmd segment below:-s

AMQ8426: Valid MQSC commands are:
    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND
     2 : end
```

キュー・マネージャーの処理

キュー・マネージャー属性を表示または変更するために使用できる MQSC コマンドの例。

キュー・マネージャーの属性を表示する

runmqsc コマンドに指定されたキュー・マネージャーの属性を表示するには、次の MQSC コマンドを使用します。

```
DISPLAY QMGR
```

このコマンドから得られる一般的な出力を 82 ページの図 15 に示します。

```
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(750)                           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)
```

図 *15. DISPLAY QMGR* コマンドからの一般的出力

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、ステム SYSTEM.*に一致する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。

キュー・マネージャーの属性の変更

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。

関連資料

キュー・マネージャーの属性

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 リファレンス」を参照してください。

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した。

例えば、下記の DEFINE QLOCAL コマンドは、以下の特性を持つ ORANGE.LOCAL.QUEUE と呼ばれる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. 説明のための値を除いて、上に示した属性値はすべてデフォルト値です。 ここでは、これらは説明の目的で示してあります。 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。 84 ページの『デフォルト・オブジェクト属性の表示』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、85 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・マネージャーに通知する必要があります。

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。

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

DISPLAY QUEUE コマンドを使用して、WebSphere 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。

WebSphere 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。

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

このコマンドの構文は、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。

ローカル・キュー定義のコピー

DEFINE コマンドに LIKE 属性を指定すると、キュー定義をコピーできます。

以下に例を示します。

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE)
```

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。 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。 以下に例を示します。

```
DEFINE QLOCAL (THIRD.QUEUE) +
```

```
        LIKE (ORANGE.LOCAL.QUEUE) +
        MAXMSGL(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、4194304 バイトではなく、1024 バイトになるように指定されます。

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・キューを定義する場合、それは DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE) と同じになります。

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、ALTER QLOCAL コマンドを使用するか、あるいは REPLACE 属性を指定して DEFINE QLOCAL コマンドを使用します。

83 ページの『ローカル・キューの定義』では、キュー ORANGE.LOCAL.QUEUE が定義されました。 ここで、例えば、このキューの最大メッセージ長を 10,000 バイトに減らすとします。

- 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。例えば、次のように指定します。

  ```
  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。 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、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

CLEAR コマンドを使用して、ローカル・キューをクリアします。

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。

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

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。

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

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。

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、そのキューは削除できません。 ただし、キューがコミットされたメッセージを 1 つ以上持っているが、コミットされていないメッセージがない場合は、PURGE オプションを指定した場合にのみ削除することができます。 以下に例を示します。

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

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。

キューのブラウズ

キュー上のメッセージの内容を調べる必要がある場合、WebSphere 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。 このブラウザーは、ソースおよび実行可能形式の両方で提供されています。

MQ_INSTALLATION_PATH は、WebSphere MQ がインストールされている上位ディレクトリーを表します。

WebSphere MQ for Windows では、サンプル・キュー・ブラウザーのファイル名およびパス名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

WebSphere MQ for UNIX and Linux では、ファイル名とパス名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

サンプルには、2 つの入力パラメーター、つまり、キュー・マネージャー名とキュー名が必要です。 以下に例を示します。

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

このコマンドの一般的な結果を 87 ページの図 16 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
```

図 *16.* キュー・ブラウザーからの一般的な結果

大規模キューの使用可能化

IBM WebSphere MQ では、2 GB を超えるキューがサポートされます。

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。 AIX、HP-UX、Linux、および Solaris システムでは、2 GB より大きいキュー・ファイルを作成する場合、事前に大きいファイルの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。

tar などの一部のユーティリティーは、2 GB より大きいファイルを処理することはできません。 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、IBM WebSphere MQ Web サイトにある、プラットフォーム固有のパフォーマンス報告を参照してください。

```
https://www.ibm.com/software/integration/ts/mqseries/
```

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。

V 7.5.0.8

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。 Version 7.5.0, Fix Pack 8 以降では、 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ている場合、IBM WebSphere MQ は MQRC_ALIAS_BASE_Q_TYPE_ERROR を返します。

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 (83 ページの『ローカル・キューの定義』を参照)。
- リモート・キューのローカル定義 (113 ページの『リモート・キューのローカル定義の作成』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。

以下の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)
```

以下の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
```

以下の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。これおよびその他の別名キュー・コマンドについて詳しくは、MQSC リファレンスを参照してください。

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

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。

以下の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)
```

以下の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)
```

以下の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)
```

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 リファレンス」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

91 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。

91 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。

92 ページの『管理トピックの属性の変更』
トピックの属性は、**ALTER TOPIC** コマンドを使用する、または **REPLACE** 属性を指定して **DEFINE TOPIC** コマンドを使用する、という 2 つの方法で変更できます。

92 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。

93 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。 トピック・ツリーについて詳しくは、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。 91 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、92 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。

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)
```

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 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 の値を表示するには、DISPLAY TPSTATUS 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)
```

管理トピックの属性の変更

トピックの属性は、**ALTER TOPIC** コマンドを使用する、または **REPLACE** 属性を指定して **DEFINE TOPIC** コマンドを使用する、という 2 つの方法で変更できます。

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。

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)
```

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。

```
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。 トピック・ツリーについて詳しくは、トピック・ツリーを参照してください。

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 リファレンス」を参照してください。

関連概念

94 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。

94 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。

95 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の属性は、**ALTER SUB** コマンドを使用する、または **REPLACE** 属性を指定して **DEFINE SUB** コマンドを使用する、という 2 つの方法で変更できます。

96 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。

96 ページの『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。
- 説明およびトピック・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。 94 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 TEST 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。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、95 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。

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: WebSphere 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: WebSphere 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: WebSphere 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)
```

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の属性は、**ALTER SUB** コマンドを使用する、または **REPLACE** 属性を指定して **DEFINE SUB** コマンドを使用する、という 2 つの方法で変更できます。

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。

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)
```

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)
```

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

このタスクについて

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。

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ているメッセージをチェックするには、94 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、86 ページの『キューのブラウズ』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。

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 WebSphere 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。

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 を実行した場合、MQSC コマンド START SERVICE によって始動されたプログラムが、停止プログラムを実行する前にまだアクティブであるかどうか判別するチェックが行われません。

サービス・オブジェクトの定義

さまざまな属性を使用して、サービス・オブジェクトを定義します。

属性は次のとおりです。

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することができます。

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。 これはデフォルト値です。

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
98 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。

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

99 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。

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。

注 :

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。UNIX and Linux システムでは /var/mqm にあります。Windows システムでは、インストール時に選択されたデータ・ディレクトリーにあります。
- キュー・マネージャー・スコープの service.env ファイル。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位置は、以下のとおりです。
  - UNIX and Linux システムでは、/var/mqm/qmgrs/QMNAME/service.env です。
  - Windows システムでは、C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env です。

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 に環境変数を指定することができます。 例えば、IBM WebSphere 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 で PATH=*$PATH*:/opt/mqm/bin と指定すると、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。

ファイル service.env には、変数を名前と値の変数の対をリストする形式で定義します。 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 ファイルの例を以下に示します。

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                                *#
#* Licensed Materials - Property of IBM                             *#
#*                                                                  *#
#* 63H9336                                                          *#
#* (C) Copyright IBM Corporation 2005, 2024.                        *#
#*                                                                  *#
#* <NOC_COPYRIGHT>                                                  *#
#*                                                                  *#
#********************************************************************#
```

```
#*******************************************************************#
#* Module Name: service.env                                        *#
#* Type       : WebSphere 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

100 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
WebSphere MQ がインストールされている位置。

MQ_DATA_PATH
WebSphere MQ データ・ディレクトリーの位置。

- UNIX and Linux システムでは、WebSphere 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 システムでは、WebSphere MQ データ・ディレクトリーの位置は WebSphere 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、WebSphere MQ のファイル名について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、101 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. 以下の 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、WebSphere 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、100 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、サーバー・サービス・オブジェクトのインスタンスは、以下の MQSC コマンドで即時に開始できます。

```
START SERVICE(S1)
```

3. 以下の MQSC コマンドを使用して、サーバー・サービス・プロセスの状況を表示します。

```
DISPLAY SVSTATUS(S1)
```

4. 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 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 MQSC 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、101 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。

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. 以下の 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、システム・ログに書き込みを行う UNIX and Linux システム提供のコマンドです。
+QMNAME+ は、キュー・マネージャーの名前を表すトークンです。

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. 以下の 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 WebSphere 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。

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. 以下の 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。

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. 以下の 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。

トリガー操作のためのオブジェクトの管理

WebSphere 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 WebSphere 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。

リモート IBM WebSphere MQ オブジェクトの管理

このセクションでは、MQSC コマンドを使用して、リモート・キュー・マネージャー上の IBM WebSphere MQ オブジェクトを管理する方法だけでなく、リモート・キュー・オブジェクトを使用して、メッセージおよび応答メッセージの宛先を制御する方法についても説明します。

このセクションでは、次の項目について説明します。

- 105 ページの『チャネル、クラスター、およびリモート・キューイング』
- 107 ページの『ローカル・キュー・マネージャーからのリモート管理』
- 113 ページの『リモート・キューのローカル定義の作成』
- 115 ページの『リモート・キュー定義を別名として使用する』
- 116 ページの『コード化文字セット間のデータ変換』

チャネル、クラスター、およびリモート・キューイング

キュー・マネージャーはメッセージを送信し、必要な場合には、戻される応答を受信して他のキュー・マネージャーと通信します。 受信側キュー・マネージャーは、次のようになります。

- 同じマシン上
- 同じ場所または別の場所にある別のマシン上
- ローカル・キュー・マネージャーと同じプラットフォームで実行
- WebSphere MQ がサポートする別のプラットフォームで実行

メッセージの発信元としては次のようなものがあります。

- あるノードから別のノードにデータを転送するユーザー作成アプリケーション・プログラム
- PCF コマンドまたは MQAI を使用するユーザー作成管理アプリケーション
- IBM WebSphere MQ エクスプローラー。
- キュー・マネージャーは、次のものを送信します。
  - 別のキュー・マネージャーへの観測イベント・メッセージ
  - `runmqsc` コマンドから発行される MQSC コマンドを間接モードで送信する (この場合、コマンドは別のキュー・マネージャーで実行されます)。

メッセージをリモート・キュー・マネージャーに送信する前に、ローカル・キュー・マネージャーには、メッセージの到着を検出したり、メッセージを転送したりするメカニズムが必要です。そのメカニズムは次のもので構成されます。

- 最低 1 つのチャネル
- 伝送キュー
- チャネル・イニシエーター

リモート・キュー・マネージャーがメッセージを受信するためには、リスナーが必要です。

チャネルは、2 つのキュー・マネージャー間の単方向通信リンクであり、リモート・キュー・マネージャーの多数のキューにメッセージを伝送することができます。

チャネルの各端は、個別に定義されます。 例えば、一方の端が送信側、つまりサーバーであれば、他方の端は受信側、つまり要求側にする必要があります。 簡単なチャネルは、ローカル・キュー・マネージャーの端の 送信側チャネル定義 およびリモート・キュー・マネージャーの端の受信側チャネル定義 で構成されます。 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。

リモート・キュー・マネージャーを、ローカル・キュー・マネージャーが送信したメッセージに対して応答するよう設定する場合、2 番目のチャネルは、ローカル・キュー・マネージャーに応答を戻すようにセットアップします。

チャネルを定義するには、MQSC コマンド DEFINE CHANNEL を使用します。 このセクションでは、特に断りがない限り、チャネルに関連した例はデフォルト・チャネル属性を使用しています。

チャネルの各端にはメッセージ・チャネル・エージェント (MCA) があり、メッセージの送受信を制御します。 MCA は、伝送キューからメッセージを受け取り、キュー・マネージャー間の通信リンクにメッセージを書き込みます。

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。 リモート・キュー定義 上の伝送キューの名前を指定します。

MCA は、複数のスレッドを使用してメッセージを転送できます。 このプロセスを、パイプライン と呼びます。 パイプラインを使用すると、MCA のメッセージ転送効率が向上し、その結果、チャネルのパフォーマンスが向上します。 パイプラインを使用するチャネルの構成方法の詳細については、チャネルの属性を参照してください。

108 ページの『リモート管理のためにチャネルおよび伝送キューを作成する』は、リモート管理をセットアップするためにこれらの定義をどのように使用するかを示しています。

一般的な分散キューイングのセットアップに関する詳細については、分散キューイング・コンポーネントを参照してください。

クラスターを使用するリモート管理

分散キューイングを使用する WebSphere MQ のネットワークでは、キュー・マネージャーはすべて独立しています。 この場合、あるキュー・マネージャーから別のキュー・マネージャーへメッセージを送信するには、伝送キュー、リモートのキュー・マネージャーへのチャネル、およびメッセージの宛先になるすべてのリモート・キューが定義されていなければなりません。

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、その属性、およびセットアップ方法の詳細については、キュー・マネージャーのクラスターを参照してください。

ローカル・キュー・マネージャーからのリモート管理

このセクションでは、MQSC および 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 管理する方法について説明します。

キューおよびチャネルの作成方法は、基本的には MQSC および PCF コマンド両方の方法と同じです。 このセクションでは、理解しやすいように各例で MQSC コマンドを示しています。 PCF コマンドを使った管理プログラムの作成の詳細は、10 ページの『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・キュー・マネージャーへの MQSC コマンドの送信は、対話式で行うか、あるいはコマンドを収めたテキスト・ファイルから行います。 リモート・キュー・マネージャーは、同じマシン上に存在する場合もありますが、通常は、異なったマシン上に存在します。 UNIX and Linux システム、Windows システム、IBM i、および z/OS を含め、他の WebSphere MQ 環境内のキュー・マネージャーをリモート側で管理することができます。

リモート管理を実施するには、特定のオブジェクトを作成する必要があります。 特殊な要件がない限り、デフォルト値 (最大メッセージ長など) で十分です。

リモート管理のためのキュー・マネージャーを作成する

MQSC コマンドを使用して、リモート管理のためのキュー・マネージャーを作成する方法。

108 ページの図 17 は、**runmqsc** コマンドを使用するリモート管理に必要なキュー・マネージャ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、MQSC コマンドを発行できるソース・キュー・マネージャーであり、それらのコマンド (オペレーター・メッセージ) の結果も、ここに戻されます。 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。

注 : -w オプションを指定して **runmqsc** を使用する場合は、`source.queue.manager` がデフォルトのキュー・マネージャー **である必要があります**。 キュー・マネージャーの作成についての詳細は、crtmqm を参照してください。

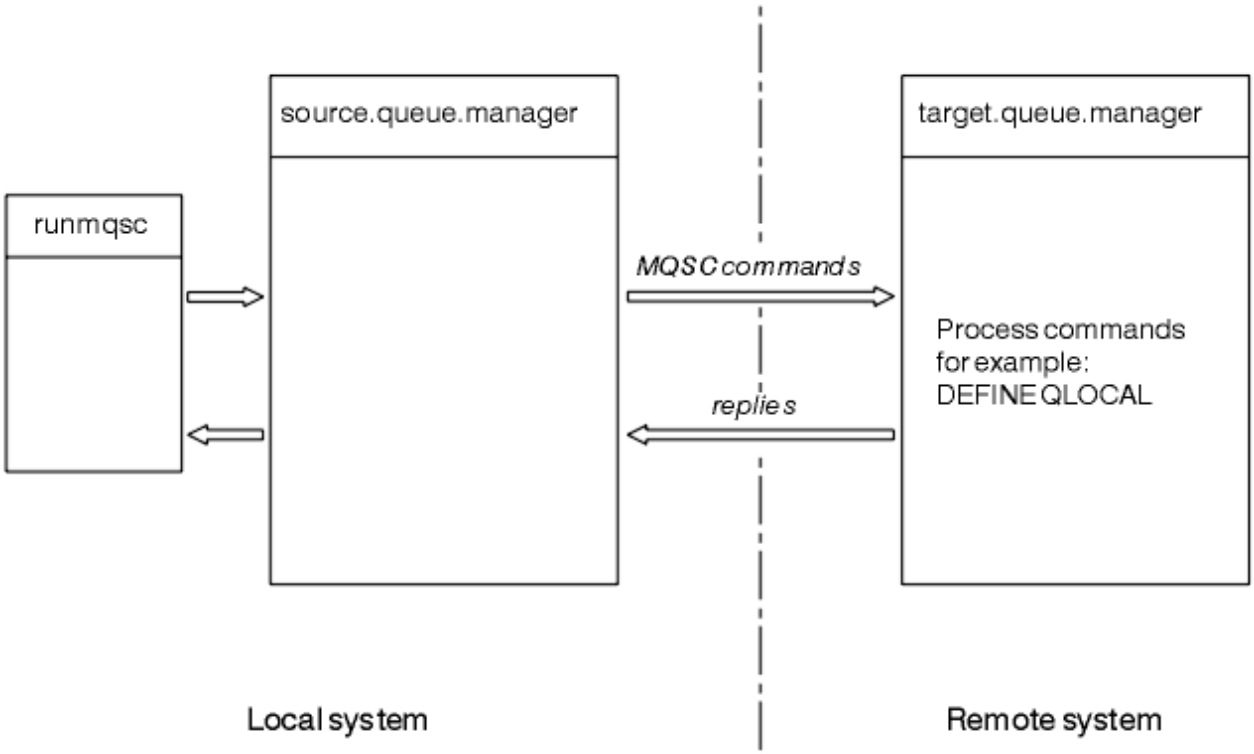


図 *17. MQSC* コマンドを使用するリモート管理

両方のシステムで、次を操作を実行していない場合は実行してください。

- `crtmqm` コマンドを使用してキュー・マネージャーおよびデフォルト・オブジェクトを作成します。
- `strmqm` コマンドを使用して、キュー・マネージャーを開始します。

ターゲット・キュー・マネージャーに対して、次のようにします。

- コマンド・キュー SYSTEM.ADMIN.COMMAND.QUEUE が存在している。 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。

これらのコマンドは、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。

リモート管理のためにチャネルおよび伝送キューを作成する

MQSC コマンドを使用して、リモート管理のためにチャネルおよび伝送キューを作成する方法。

リモートから MQSC コマンドを実行するには、2 つのチャネル (各方向ごとに 1 つ) およびそれらに関連した伝送キューをセットアップします。 この例では、TCP/IP がトランスポート・タイプとして使用されていること、および関係している TCP/IP アドレスをユーザーが把握していることが前提条件です。

チャネル `source.to.target` は、MQSC コマンドをソース・キュー・マネージャーからターゲット・キュー・マネージャーに送るためのものです。 送信側は `source.queue.manager` であり、受信側は `target.queue.manager` です。 チャネル `target.to.source` は、コマンドの出力および生成されたオペレーター・メッセージをソース・キュー・マネージャーに戻すためのものです。 各チャネルごとに伝送キューを定義する必要もあります。 このキューは、受信側のキュー・マネージャーの名前が付けられたローカル・キューです。 キュー・マネージャー別名を使用していない限り、リモート管理を行うために、XMITQ 名がリモート・キュー・マネージャー名と一致している必要があります。109 ページの図 18 この構成を要約します。

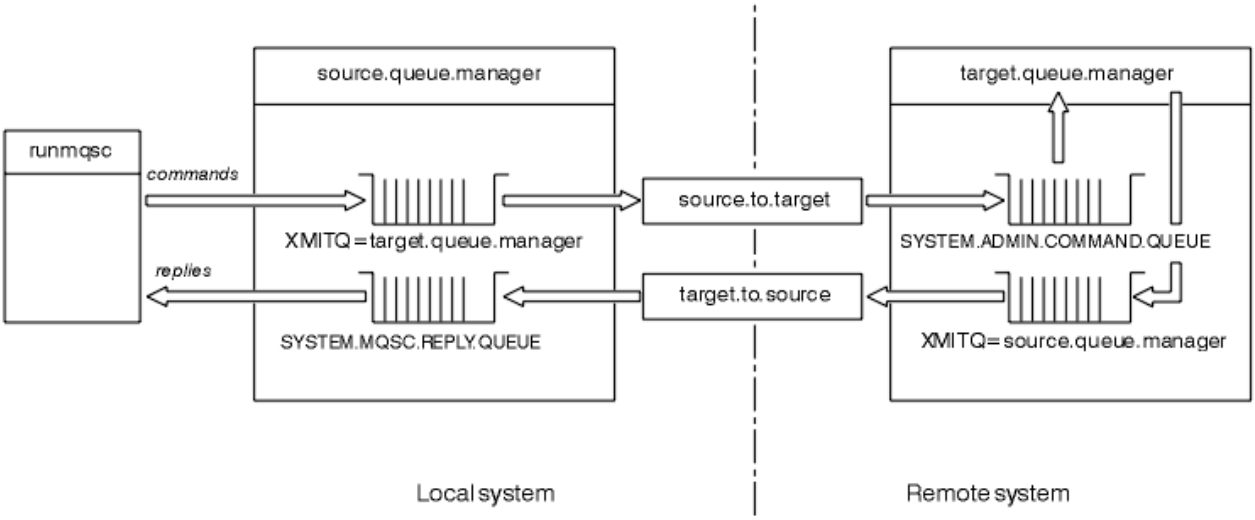


図 *18.* リモート管理のためのチャネルとキューのセットアップ

チャネルのセットアップについての詳細は、分散キューイングを使用したアプリケーションの接続を参照してください。

チャネル、リスナーおよび伝送キューを定義する

送信元キュー・マネージャー (source.queue.manager) に対して、以下の MQSC コマンドを発行して、チャネル、リスナー、および伝送キューを定義します。

1. 次のように、送信元キュー・マネージャーに対して送信元チャネルを定義します。

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)
```

2. 次のように、送信元キュー・マネージャーに対して受信側チャネルを定義します。

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. 送信元キュー・マネージャーに対してリスナーを定義します。

```
DEFINE LISTENER ('source.queue.manager') +
       TRPTYPE (TCP)
```

4. 次のように、送信元キュー・マネージャーに対して伝送キューを定義します。

```
DEFINE QLOCAL ('target.queue.manager') +
       USAGE (XMITQ)
```

ターゲット・キュー・マネージャー (target.queue.manager) に対して以下のコマンドを発行して、チャネル、リスナー、および伝送キューを作成します。

1. 次のように、ターゲット・キュー・マネージャーに対して送信側チャネルを定義します。

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)
```

2. 次のように、ターゲット・キュー・マネージャーに対して受信側チャネルを定義します。

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. 次のように、ターゲット・キュー・マネージャーに対してリスナーを定義します。

```
DEFINE LISTENER ('target.queue.manager') +
       TRPTYPE (TCP)
```

4. 次のように、ターゲット・キュー・マネージャーに対して伝送キューを定義します。

```
DEFINE QLOCAL ('source.queue.manager') +
       USAGE (XMITQ)
```

注 : 送信側チャネル定義の CONNAME 属性に 指定された TCP/IP 接続名は、説明のためだけに示したものです。 これは、接続の他方 側のマシンのネットワーク名です。 各自ネットワークに合った値を使用してください。

リスナーおよびチャネルを開始する

MQSC コマンドを使用して、リスナーおよびチャネルを開始する方法。

以下の MQSC コマンドを使用して、両方のリスナーを開始します。

1. 次の MQSC コマンドを発行して、ソース・キュー・マネージャー source.queue.manager でリスナーを開始します。

```
START LISTENER ('source.queue.manager')
```

2. 次の MQSC コマンドを発行して、ターゲット・キュー・マネージャー target.queue.manager でリスナーを開始します。

```
START LISTENER ('target.queue.manager')
```

以下の MQSC コマンドを使用して、両方の送信側チャネルを開始します。

1. 次の MQSC コマンドを発行して、ソース・キュー・マネージャー source.queue.manager で送信側チャネルを開始します。

```
START CHANNEL ('source.to.target')
```

2. 次の MQSC コマンドを発行して、ターゲット・キュー・マネージャー target.queue.manager で送信側チャネルを開始します。

```
START CHANNEL ('target.to.source')
```

チャネルの自動定義

MQSC コマンド、ALTER QMGR (または PCF コマンド Change Queue Manager) を使用してキュー・マネージャー・オブジェクトを更新することによって、受信側の自動定義およびサーバー接続定義を使用可能にします。

WebSphere MQ がインバウンド接続要求を受信したが、適切な受信側またはサーバー接続チャネルが見つからない場合、WebSphere MQ は自動的にチャネルを作成します。 自動定義は、WebSphere MQ に用意されている 2 つのデフォルト定義 SYSTEM.AUTO.RECEIVER および SYSTEM.AUTO.SVRCONN。

チャネル定義の自動作成について詳しくは、チャネルの準備を参照してください。 クラスターのチャネルの自動定義については、クラスター・チャネルの自動定義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

コマンド・サーバーを開始、停止、およびその状況を表示する方法。 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。

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。

注 : リモート管理では、ターゲット・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。 この状況を回避してください。

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。 コマンド・サーバーが稼働している場合、WebSphere MQ for Windows または WebSphere MQ for Linux (x86 および x86-64 プラットフォーム) のユーザーは、 IBM WebSphere MQ エクスプローラーを使用して以下のセクションで説明されている操作を実行できます。 詳しくは、56 ページの『IBM WebSphere MQ Explorer による管理』を参照してください。

コマンド・サーバーを開始する

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。 キュー・マネージャー属性の値は、パラメーター SCMDSERV を指定した MQSC コマンド ALTER QMGR を使用して変更できます。 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。

SCMDSERV が MANUAL に設定されている場合は、コマンド・サーバーを以下のコマンドを使用して始動します。

```
strmqcsv 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・キューに入れられます。

キュー・マネージャーのコマンド・サーバーの状況を表示するには、次の MQSC コマンドを発行します。

```
DISPLAY QMSTATUS CMDSERV
```

コマンド・サーバーを停止する

直前の例で開始されたコマンド・サーバーを終了するには、次のコマンドを使用します。

```
endmqcsv saturn.queue.manager
```

コマンド・サーバーを停止するには、次の 2 つの方法があります。

- 制御された停止の場合、-c フラグを指定した endmqcsv コマンドを使用します。これはデフォルトです。
- 即時停止の場合、-i フラグを指定した endmqcsv コマンドを使用します。

注 : 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。

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

runmqsc コマンドの特定の形式を使用して、リモート・キュー・マネージャー上で MQSC コマンドを実行できます。

MQSC コマンドをリモートから処理する場合には、コマンド・サーバーがターゲット・キュー・マネージャー上で実行されている**必要があります**。(ソース・キュー・マネージャーで実行されている必要はありません)。 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、111 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。

次に、送信元キュー・マネージャーに対して次のように入力することにより、間接モードで対話式に MQSC コマンドを実行できます。

```
runmqsc -w 30 target.queue.manager
```

この形式の runmqsc コマンド (-w フラグ付き) は、間接モードで MQSC コマンドを実行します。このモードでは、コマンドがコマンド・サーバーの入力キューに修正された形式で書き込まれ、順次に実行されます。

MQSC コマンドを入力すると、このコマンドはリモート・キュー・マネージャー (ここでは、target.queue.manager) にリダイレクトされます。 タイムアウトは 30 秒に設定されます。したがって、30 秒以内に応答がなければ、ローカル (送信元) 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。

ソース・キュー・マネージャーは、デフォルトのローカル・キュー・マネージャーに設定されます。 **runmqsc** コマンドに -m *LocalQmgrName* オプションを指定すれば、ローカル・キュー・マネージャーを介してコマンドを発行するように指示できます。

間接モードでは、MQSC コマンドもリモート・キュー・マネージャー上で実行することができます。 以下に例を示します。

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

ここで、mycomds.in は MQSC コマンドを含んでいるファイルであり、report.out はレポート・ファイルです。

コマンドのリモート発行に関する提案されているメソッド

リモート・キュー・マネージャーに対してコマンドを発行するときには、以下の方法を使用することを考慮してください。

1. リモート・システムで実行する MQSC コマンドは、コマンド・ファイルに入れます。
2. runmqsc コマンドに -v フラグを指定することにより、MQSC コマンドをローカルで確認します。

   runmqsc を使用して別のキュー・マネージャー上の MQSC コマンドを確認することはできません。
3. コマンド・ファイルがローカルでエラーなしで実行されることを確認します。
4. リモート・システムでコマンド・ファイルを実行します。

リモートからの MQSC コマンドの使用に問題がある場合

リモートから MQSC コマンドを実行することが困難な場合には、次のことを確認してください。

- ターゲット・キュー・マネージャーのコマンド・サーバーを開始しましたか。
- 有効な伝送キューを定義しましたか。

- 次の両方について、メッセージ・チャネルの両端を定義しましたか。
  - コマンドが送信されるチャネル
  - 応答が戻されるチャネル
- チャネル定義に正しい結合名 (CONNAME) を指定しましたか。
- メッセージ・チャネルを開始する前に、リスナーを開始しましたか。
- 切断時間の間隔が期限切れでないかどうか (チャネルが開始したが、しばらくしてシャットダウンされた場合など) について検査しましたか。 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。
- 意味をなさない送信元キュー・マネージャーからターゲット・キュー・マネージャー (例えば、リモート・キュー・マネージャーではサポートされていないパラメーターを含む要求) に要求を送信します。

80 ページの『MQSC コマンドで起こった問題の解決』も参照してください。

リモート・キューのローカル定義の作成

リモート・キューのローカル定義は、リモート・キュー・マネージャー上のキューを参照するローカル・キュー・マネージャー上の定義です。

ローカル位置からリモート・キューを定義する必要はありませんが、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働きについて理解する

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、アプリケーションは、 MQOPEN 呼び出しから戻されるハンドルを指定して、 MQPUT 呼び出しを発行します。 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。

伝送キューの定義

伝送キューとは、キュー・マネージャーがメッセージ・チャネルを介してメッセージをリモート・キュー・マネージャーに転送する際に使用するローカル・キューのことです。

チャネルは、リモート・キュー・マネージャーへの片方向リンクを提供します。 メッセージは、チャネルがメッセージを受け入れることができるまで、伝送キューにキューイングされます。 チャネルを定義する際には、メッセージ・チャネルの送信側に伝送キュー名を指定してください。

MQSC コマンド属性 USAGE は、キューが伝送キューであるか、通常のキューであるかを定義します。

デフォルト伝送キュー

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

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

例えば、次の MQSC コマンドでは、`target.queue.manager` に送られるメッセージ用として、デフォルト伝送キューが `source.queue.manager` に作成されます。

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

アプリケーションは、メッセージを伝送キューに直接書き込むことも、リモート・キュー定義を介して間接的に書き込むこともできます。 113 ページの『リモート・キューのローカル定義の作成』も参照してください。

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。

応答先キューの別名について詳しくは、 応答先キューの別名とクラスター を参照してください。

コード化文字セット間のデータ変換

WebSphere 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、データ変換で定義されています。

キュー・マネージャーがメッセージを組み込み形式に変換できない場合

CCSID が別の各国語グループを表している場合には、キュー・マネージャーはメッセージを組み込み形式に 自動的に変換することはできません。 例えば、CCSID 850 と CCSID 1025 (キリル文字スクリプトを使用する言語用の EBCDIC コード化文字セット) 間の変換はサポートされていません。これは、一方のコード化文字セットの文字の多くが、もう一方のコード化文字セットで表現できないためです。 さまざまな各国語で稼働しているキュー・マネージャーのネットワークがあり、一部のコード化文字セット間でのデータ変換がサポートされていない場合に、デフォルト変換を使用することができます。 デフォルトのデータ変換については、117 ページの『デフォルトのデータ変換』で説明しています。

ファイル ccsid.tbl

ファイル ccsid.tbl は、次の目的に使用されます。

- WebSphere MQ for Windows では、このファイルによってサポートされている すべてのコード・セットを記録しています。
- AIX および HP-UX プラットフォームでは、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。
- 他のすべての UNIX and Linux プラットフォームの場合、サポートされるコード・セットは WebSphere MQ が提供する変換テーブルに保持されます。
- 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 (これを行う方法はファイルで指示されています)。
- 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 に記録されている情報を更新することができます。例えば、使用しているオペレーティング・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る場合に、更新が必要となる場合があります。

WebSphere MQ for Windows では、 ccsid.tbl はデフォルトでディレクトリー `C:\Program Files\IBM\WebSphere MQ\conv\table` にあります。

WebSphere MQ for UNIX and Linux システムでは、ccsid.tbl は `/var/mqm/conv/table` のディレクトリーにあります。

デフォルトのデータ変換

データ変換が通常はサポートされていない 2 つのマシン間でチャネルをセットアップする場合、チャネルが作動するようにデフォルトのデータ変換を使用可能にしなければなりません。

デフォルトのデータ変換を実施するには、ccsid.tbl ファイルを編集して、デフォルト EBCDIC CCSID とデフォルト ASCII CCSID を指定します。 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、WebSphere 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。 WebSphere MQ オブジェクト名 (IBM WebSphere MQ オブジェクトの命名で定義されている) にとって有効な文字のセットのみを使用する場合には、通常、この要件を満たします。 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。

IBM WebSphere MQ Telemetry の管理

IBM WebSphere MQ Telemetry の管理は、IBM WebSphere 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 WebSphere 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。 JAAS、SSL、および IBM WebSphere MQ オブジェクト権限マネージャーを使用して、 IBM WebSphere MQ Telemetry のセキュリティーを構成します。

IBM WebSphere 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 WebSphere 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。 JAAS、SSL、および IBM WebSphere MQ オブジェクト権限マネージャーを使用して、IBM WebSphere MQ Telemetry のセキュリティーを構成します。

コマンド行を使用した管理

IBM WebSphere MQ Telemetry は、コマンド行で IBM WebSphere MQ MQSC コマンドを使用して完全に管理できます。

IBM WebSphere MQ Telemetry 資料には、MQ Telemetry Transport v3 クライアント・アプリケーションの基本的な使用法を示すサンプル・スクリプトも含まれています。

使用する前に、「IBM WebSphere MQ Telemetry 用アプリケーションの開発」セクションの IBM WebSphere MQ Telemetry サンプル・プログラム にあるサンプルを読んで理解してください。

関連概念

WebSphere MQ Telemetry

122 ページの『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 WebSphere MQ アプリケーションは、クライアントが作成したサブスクリプションにパブリッシュして、またはメッセージを直接送信して、MQTT v3 クライアントにメッセージを送信できます。 どちらの方法を使用する場合でも、メッセージは SYSTEM.MQTT.TRANSMIT.QUEUE に置かれ、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には、いくつかの方法があります。

124 ページの『MQTT クライアントの識別、許可、および認証』

131 ページの『SSL を使用したテレメトリー・チャネルの認証』

133 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』

134 ページの『MQTT クライアントおよびテレメトリー・チャネルの SSL 構成』

139 ページの『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。

141 ページの『IBM WebSphere MQ Telemetry デーモン (デバイス用) の概念』
IBM WebSphere MQ Telemetry デーモン (デバイス用) は、拡張 MQTT V3 クライアント・アプリケーションです。 これは、他の MQTT クライアントからのメッセージをストア・アンド・フォワードするために使用します。 MQTT クライアントのように IBM WebSphere MQ に接続しますが、他の MQTT クライアントを接続することもできます。

関連タスク

118 ページの『テレメトリー対応キュー・マネージャーの構成 (Linux および AIX)』
IBM WebSphere MQ Telemetry が動作するようにキュー・マネージャーを構成するには、以下の手動ステップを実行します。 IBM WebSphere MQ Telemetry の Support for IBM WebSphere MQ Explorer を使用して自動化プロシージャーを実行することにより、より単純な構成をセットアップできます。

120 ページの『Windows 上のテレメトリー用キュー・マネージャーの構成』
IBM WebSphere MQ Telemetry が動作するようにキュー・マネージャーを構成するには、以下の手動ステップを実行します。 IBM WebSphere MQ Telemetry の Support for IBM WebSphere MQ Explorer を使用して自動化プロシージャーを実行することにより、より単純な構成をセットアップできます。

関連資料

MQXR プロパティー

テレメトリー対応キュー・マネージャーの構成 (Linux および AIX)

IBM WebSphere MQ Telemetry が動作するようにキュー・マネージャーを構成するには、以下の手動ステップを実行します。 IBM WebSphere MQ Telemetry の Support for IBM WebSphere MQ Explorer を使用して自動化プロシージャーを実行することにより、より単純な構成をセットアップできます。

始める前に

1. IBM WebSphere MQ および IBM WebSphere MQ Telemetry フィーチャーのインストール方法については、「IBM WebSphere MQ Telemetry のインストール」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_unix.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 MQ エクスプローラー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

IBM WebSphere MQ Explorer の IBM WebSphere MQ Telemetry 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。 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 WebSphere MQ Explorer を使用した IBM WebSphere MQ Telemetry のインストールの検査 および IBM WebSphere MQ Telemetry サンプル・プログラムを参照してください。

さまざまな許可スキームを使用して IBM WebSphere 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。

   テレメトリーのサンプル・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。

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc qMgr
   ```

   テレメトリー (MQXR) サービスは、最初に開始されたときに SYSTEM.MQTT.TRANSMIT.QUEUE を作成します。

   このタスクでは、手動でそれを作成します。 SYSTEM.MQTT.TRANSMIT.QUEUE へのアクセスを許可するには、テレメトリー (MQXR) サービスを開始する前にそれが存在していなければならないからです。
3. デフォルト伝送キューの設定

   テレメトリー (MQXR) サービスは、最初に開始されたときに SYSTEM.MQTT.TRANSMIT.QUEUE をデフォルト伝送キューにするようキュー・マネージャーを変更するわけではありません。

   SYSTEM.MQTT.TRANSMIT.QUEUE をデフォルト伝送キューにするには、デフォルト伝送キュー・プロパティーを変更します。 IBM WebSphere MQ Explorer を使用するか、以下の例のコマンドを使用して、プロパティーを変更します。

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

   デフォルト伝送キューを変更すると、既存の構成に支障が生じる可能性があります。 デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に変更する理由は、MQTT クライアントにメッセージを直接送信しやすくするためです。 デフォルト伝送キューを変更しない場合は、IBM WebSphere MQ メッセージを受信するクライアントごとにリモート・キュー定義を追加する必要があります (123 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)。
4. 125 ページの『MQTT クライアントによる WebSphere MQ オブジェクトへのアクセスを許可』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
   ```

   120 ページの図 19 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 WebSphere 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。

   ステップ 4 で定義したユーザー ID のいずれかの ID になるようにテレメトリー・チャネルを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 WebSphere MQ Telemetry サンプル・プログラムも参照してください。

例

120 ページの図 19 は、 Linux で SYSTEM.MQXR.SERVICE を手動で作成するための **runmqsc** コマンドを示しています。

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

図 *19. installMQXRService_unix.mqsc*

Windows 上のテレメトリー用キュー・マネージャーの構成

IBM WebSphere MQ Telemetry が動作するようにキュー・マネージャーを構成するには、以下の手動ステップを実行します。 IBM WebSphere MQ Telemetry の Support for IBM WebSphere MQ Explorer を使用して自動化プロシージャーを実行することにより、より単純な構成をセットアップできます。

始める前に

1. IBM WebSphere MQ および IBM WebSphere MQ Telemetry フィーチャーのインストール方法については、「IBM WebSphere MQ Telemetry のインストール」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 MQ エクスプローラー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

IBM WebSphere MQ Explorer の IBM WebSphere MQ Telemetry 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。 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 WebSphere MQ Explorer を使用した IBM WebSphere MQ Telemetry のインストールの検査 および IBM WebSphere MQ Telemetry サンプル・プログラムを参照してください。

さまざまな許可スキームを使用して IBM WebSphere 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。

```
echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
```

   テレメトリー (MQXR) サービスは、最初に開始されたときに SYSTEM.MQTT.TRANSMIT.QUEUE を作成します。

   このタスクでは、手動でそれを作成します。 SYSTEM.MQTT.TRANSMIT.QUEUE へのアクセスを許可するには、テレメトリー (MQXR) サービスを開始する前にそれが存在していなければならないからです。
3. デフォルト伝送キューの設定

```
echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
```

   図 *20.* デフォルト伝送キューの設定

   テレメトリー (MQXR) サービスは、最初に開始されたときに SYSTEM.MQTT.TRANSMIT.QUEUE をデフォルト伝送キューにするようキュー・マネージャーを変更するわけではありません。

   SYSTEM.MQTT.TRANSMIT.QUEUE をデフォルト伝送キューにするには、デフォルト伝送キュー・プロパティーを変更します。 プロパティーは、IBM WebSphere MQ Explorer を使用するか、121 ページの図 20 のコマンドを使用して変更します。

   デフォルト伝送キューを変更すると、既存の構成に支障が生じる可能性があります。 デフォルト伝送キューを SYSTEM.MQTT.TRANSMIT.QUEUE に変更する理由は、MQTT クライアントにメッセージを直接送信しやすくするためです。 デフォルト伝送キューを変更しない場合は、IBM WebSphere MQ メッセージを受信するクライアントごとにリモート・キュー定義を追加する必要があります (123 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)。
4. 125 ページの『MQTT クライアントによる WebSphere MQ オブジェクトへのアクセスを許可』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
```

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 WebSphere 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。

   ステップ 4 で定義したユーザー ID のいずれかの ID になるようにテレメトリー・チャネルを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 WebSphere MQ Telemetry サンプル・プログラムも参照してください。

SYSTEM.MQXR.SERVICE の手動作成

122 ページの図 21 は、Windows で SYSTEM.MQXR.SERVICE を手動で作成するための **runmqsc** コマンドを示しています。

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
    CONTROL(QMGR) +
    DESCR('Manages clients using MQXR protocols such as MQTT') +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
    STARTARG('-m +QMNAME+  -d "+MQ_Q_MGR_DATA_PATH+\." -g "+MQ_DATA_PATH+\."') +
    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')
```

図 *21. installMQXRService_win.mqsc*

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 WebSphere MQ アプリケーションは、クライアントが作成したサブスクリプションにパブリッシュして、またはメッセージを直接送信して、MQTT v3 クライアントにメッセージを送信できます。 どちらの方法を使用する場合でも、メッセージは SYSTEM.MQTT.TRANSMIT.QUEUE に置かれ、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には、いくつかの方法があります。

MQTT クライアント・サブスクリプションへの応答としてのメッセージのパブリッシュ

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 WebSphere MQ に接続され、そのキュー・マネージャー名が ClientIdentifier として設定されます。 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、特定のクライアントへのキーとしてターゲット・キュー・マネージャー名を使用して、 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 WebSphere MQ オブジェクト名解決を使用すると、 *ClientIdentifier* は以下のように解決されます。 123 ページの表 6 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 に設定して、パブリケーションがクライアントに送信される SYSTEM.MQTT.TRANSMIT.QUEUE に書き込ま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 6. *MQTT* キュー・マネージャー別名のネーム解決					
	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 WebSphere 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 *22. 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 クライアントの複数のサブスクライバーとパブリッシャーは、異なるキュー・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、WebSphere 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

クライアントによるサブスクリプションの作成および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の受信の代替として、メッセージを MQTT v3 クライアントに直接送信します。 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できませんが、WebSphere MQ アプリケーションなどのその他のアプリケーションは直接送信できます。

WebSphere 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ため、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 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 *23. 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 *24. 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。 124 ページの表 7 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 *7. MQTT* クライアント・リモート・キュー定義のネーム解決

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、MQTT 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。 クライアントに持続メカニズムがない場合、そのクライアントは、受け入れる直接送信され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作成します。 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。

MQTT クライアントの識別、許可、および認証

遠隔測定 (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 WebSphere MQ トピックをパブリッシュしたり、それにサブスクライブしたりします。 WebSphere MQ 管理者は、WebSphere MQ の許可に使用される 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 Username および Password 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 SSL 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。

WebSphere MQ オブジェクトにアクセスする MQTT クライアントを許可するには、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 WebSphere 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 SSL 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 SSL 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。

MQTT クライアントの ID および許可

ClientIdentifier、Username、または共通クライアント ID を使用して、WebSphere MQ オブジェクトにアクセスする許可を得ます。

WebSphere 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、WebSphere 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。 選択肢は以下のとおりです。

1. クライアント ID。
2. 管理者がチャネルに提供する ID。
3. 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。

WebSphere 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、キュー・マネージャー QM1 の管理者によって提供されるチャネル ID である MQTTClient を許可するには、次のようにします。

```
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
```

MQTT クライアントによる *WebSphere MQ* オブジェクトへのアクセスを許可

WebSphere MQ オブジェクトへのパブリッシュおよびサブスクライブ行う権限を MQTT クライアントに与えるには、次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 WebSphere 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。 WebSphere MQ の管理者は WebSphere 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 WebSphere 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、WebSphere 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を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。

MQTT クライアントはそれぞれアクセス権を必要とするため、何万もの MQTT クライアントが存在するシステムを管理することは困難です。 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 WebSphere 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。

- 126 ページの『アクセス制御なし』
- 127 ページの『粗い細分度のアクセス制御』
- 127 ページの『中程度の細分度のアクセス制御』
- 127 ページの『細い細分度のアクセス制御』

これらのモデルにより、WebSphere MQ にパブリッシュおよびサブスクライブし、WebSphere MQ からパブリケーションを受け取るためのアクセス権の *mqttUsers* セットを割り当てます。

アクセス制御なし
MQTT クライアントは、WebSphere 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 である *mqttUser* を作成します。
2. *mqttUser* を mqm グループに追加します。「Windows でのグループへのユーザーの追加」または「Linux でのグループへのユーザーの追加 」を参照してください。

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 である *mqttUser* を作成します。
2. すべてのトピックへパブリッシュおよびサブスクライブし、パブリケーションを MQTT クライアントに送信する権限を *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 クライアントへのトピックの送信を行う権限を *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、アクセス制御の既存のシステムに組み込まれます。

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。 WebSphere 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* にパブリッシュできます。

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/サブスクライブのトピック・ツリーの複数の管理トピックに割り当てられている複数のグループ、Publish*X* と 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. さまざまなトピックへの権限を別々の Publish*X* グループおよび Subscribe*X* グループに与え、MQTT クライアントへのメッセージ送信を行う権限を *mqtt* グループに与えます。

```
setmqaut -m qMgr -t topic -n topic1 -p PublishX -all +pub
setmqaut -m qMgr -t topic -n topic1 -p SubscribeX -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

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 Password を使用します。

WebSphere MQ 管理者は、クライアントが接続する MQTT チャネルを構成して、Username を認証するかまたはまったく認証しないかを決めます。 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. WebSphere MQ 管理者は WebSphere MQ エクスプローラーを使用して、MQTT チャネルの ID を MQTTClientUser などの任意の名前に設定できます。
2. WebSphere 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. WebSphere MQ 管理者は、JAAS を使用するクライアントの UserName を認証するように MQTT チャネルを構成します。

SSL 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。

クライアントに秘密署名付きデジタル証明書を提供することで、MQTT クライアントを IBM WebSphere MQ に対して認証できます。 IBM WebSphere MQ 管理者は、MQTT クライアントが SSL 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。

SSL 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。

-
-

SSL を使用した *MQTT* クライアント認証のためのテレメトリー・チャネル構成

IBM WebSphere MQ 管理者は、サーバー側のテレメトリー・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。 SSL 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。 SSL 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 SSL 接続を受け入れません。

SSL 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す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を持ってい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。

WebSphere 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、WebSphere 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、 WebSphere MQ 管理者により作成された ID です。

また、SSL 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。現在サポートされている SSL 暗号スイートのアルファベット順のリストを以下に示します。

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5

- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 SHA-2 暗号スイートを使用する予定の場合は、MQTT チャネルで SHA-2 暗号スイートを使用するためのシステム要件を参照してください。

関連概念

132 ページの『SSL 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 WebSphere MQ 管理者は、サーバー側のテレメトリー・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。 SSL 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。 SSL 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 SSL 接続を受け入れません。

CipherSpec および CipherSuite

関連資料

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

SSL を使用したテレメトリー・チャネルの認証

MQTT クライアントとキュー・マネージャーとの間の接続は、常に MQTT クライアントによって開始されます。 MQTT クライアントは常に SSL クライアントです。 サーバーのクライアント認証および MQTT クライアントのサーバー認証は、どちらもオプションです。

クライアントが匿名接続をサポートする CipherSpec を使用するように構成されていない限り、クライアントは常にサーバーの認証を試行します。 認証が失敗すると、接続は確立されません。

SSL を使用する代わりに、例えば IPsec など、いくつかの種類の仮想プライベート・ネットワーク (VPN) は TCP/IP 接続のエンドポイントを認証します。 VPN は、ネットワーク上を流れる各 IP パケットを暗号化します。 そのような VPN 接続が確立されると、トラステッド・ネットワークが確立されます。 VPN ネットワーク上で TCP/IP を使用して、MQTT クライアントをテレメトリー・チャネルに接続できます。

SSL を使用するサーバー認証は、機密情報の送信先となるサーバーを認証します。 クライアントは、サーバーから送信された証明書、トラストストアに格納されている証明書、または JRE `cacerts` ストアに格納されている証明書と突き合わせ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た一部の OGSi 環境では、 この環境は別のトラスト・プロバイダーを提供します。

SSL 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。

-
-

SSL 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 WebSphere MQ 管理者は、サーバー側のテレメトリー・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。 SSL 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。 SSL 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 SSL 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。 WebSphere 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、SSL 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、SSL 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、SSL 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。現在サポートされている SSL 暗号スイートのアルファベット順のリストを以下に示します。

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA

- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 SHA-2 暗号スイートを使用する予定の場合は、MQTT チャネルで SHA-2 暗号スイートを使用するためのシステム要件を参照してください。

関連概念

129 ページの『SSL を使用した MQTT クライアント認証のためのテレメトリー・チャネル構成』
IBM WebSphere MQ 管理者は、サーバー側のテレメトリー・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。 SSL 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。 SSL 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 SSL 接続を受け入れません。

CipherSpec および CipherSuite

関連資料

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

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、SSL 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、SSL 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、131 ページの『SSL 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 SSL 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 SSL 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 SSL 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、SSL 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 SSL 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 SSL 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、SSL 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。

MQTT クライアントおよびテレメトリー・チャネルの SSL 構成

MQTT クライアントおよび WebSphere MQ Telemetry (MQXR) サービスは、Java Secure Socket Extension (JSSE) を使用して、SSL を使用するテレメトリー・チャネルに接続します。 IBM WebSphere MQ Telemetry デーモン (デバイス用) は、SSL をサポートしません。

テレメトリー・チャネルと MQTT クライアントを認証するよう SSL を構成し、クライアントとそのテレメトリー・チャネルの間のメッセージの転送を暗号化します。

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。

TCP/IP を介した SSL プロトコルを使用するように、Java MQTT クライアントとテレメトリー・チャネルの間の接続を構成できます。 保護対象は、JSSE を使用するために SSL 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。128 ページの『SSL 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。131 ページの『SSL 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。133 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、SSL 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. IBM WebSphere MQ テレメトリー・チャネル
2. MQTT Java クライアント
3. JRE

IBM WebSphere 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。 `MqttConnectionOptions.SSLProperties` 属性に MQTT Java クライアント・パラメーターを設定します。 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。

IBM WebSphere MQ テレメトリー・チャネル

WebSphere 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 SSL 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、SSL 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 SSL プロパティーを持っている場合、クライアントは SSL 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、SSL 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、提供する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、SSL 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 SSL 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 SSL 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、チャネル鍵ストアおよび JRE の cacerts ストアに格納されている証明書と突き合わせて認証されます。

CipherSuite

CipherSuite は、オプションの SSL 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 SSL 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 から共通の CipherSpec について折衝します。 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。

SSL 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 SSL プロパティーを MqttConnectionOptions.SSLProperties に設定します。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については、MqttConnectOptions クラスで説明しています。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。

プロトコル

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。 信頼するすべての証明書を JRE の cacerts ストアに配置できます。

クライアント用に別のトラストストアが必要な場合には、このトラストストアを構成します。 既にルート証明書を cacerts に保管している既知の CA が発行した証明書をサーバーが使用している場合は、トラストストアを構成しないこともあります。

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 SSL 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 の構成ファイルは、 *Java Installation Directory*\jre\lib\security にあります。 IBM WebSphere 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 *8. JRE SSL* 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
その他の UNIX 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ています。 トラストストアを指定していない限り、デフォルトでは cacerts が使用されます。 cacerts ストアを使用する場合、またはトラストストアを指定していない場合は、cacerts 内の署名者のリストを検討して、セキュリティー要件を満たすようそれを編集する必要があります。

cacerts を開くには、WebSphere MQ コマンド strmqikm を使用します。これは、IBM 鍵管理ユーティリティーを実行します。 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・プロバイダーとその他のデフォルトのセキュリティー・プロパティーを登録します。

権限

java.policy ファイルを使用して、リソースに付与された許可を変更します。 javaws.policy は許可を javaws.jar に付与します。

暗号化の強度

一部の JRE は、暗号化の強度を下げて出荷されています。 鍵を鍵ストアにインポートできない場合は、暗号化の強度が下げられているのが原因である場合があります。 **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、SSL 接続は暗号化されますが、認証されません。

138 ページの図 25 のコード・スニペットは、MQTT クライアントに対して 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 *25. 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 *26. 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 *27. 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 *28. AcceptAllX509TrustManager.java*

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。

WebSphere MQ 管理者は、どの MQTT チャネルで JAAS を使用するクライアント認証が必要となるかを構成します。 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 に定義されています。

jaas.config ファイルは、JAAS 構成名に基づいて編成されます。 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。 140 ページの図 29 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 を使用することの問題点は、テレメトリー (MQXR) サービスが MQTT チャネルの ID ではなく mqm の ID を使用して稼働することです。 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 WebSphere 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実装です。 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 WebSphere 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。

クラスパスを service.env ファイル内に設定して、テレメトリー (MQXR) サービスで使用されるクラスパスを増強することができます。 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 では変数を使用できません。例えば、CLASSPATH=%CLASSPATH% は間違いです。 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 で定義された CLASSPATH がそれに追加されます。

テレメトリー (MQXR) サービスは、MQTT チャネルに接続したクライアントの Username および Password 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、141 ページの図 30 を参照してください。

例

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";
};
```

図 *29. jaas.config* ファイルのサンプル

MQTT クライアントによって提供される Username および Password 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;
}
```

図 *30. JAASLoginModule.Login()* メソッドのサンプル

IBM WebSphere MQ Telemetry デーモン (デバイス用) の概念

IBM WebSphere MQ Telemetry デーモン (デバイス用) は、拡張 MQTT V3 クライアント・アプリケーションです。 これは、他の MQTT クライアントからのメッセージをストア・アンド・フォワードするために使用します。 MQTT クライアントのように IBM WebSphere MQ に接続しますが、他の MQTT クライアントを接続することもできます。

デーモンはパブリッシュ/サブスクライブ・ブローカーです。 MQTT V3 クライアントはこれに接続して、パブリッシュの場合はトピック・ストリングを使用し、サブスクライブの場合はトピック・フィルターを使用して、トピックへのパブリッシュおよびサブスクライブを行います。 トピック・ストリングは階層型をしており、そのトピック・レベルは / で分割されます。 トピック・フィルターは、単一レベルの + ワイルドカードとトピック・ストリングの最後の部分としてマルチレベルの # ワイルドカードを含めることができるトピック・ストリングです。

注 : デーモンのワイルドカードは、WebSphere Message Broker v6 のより厳しい規則に従います。 IBM WebSphere MQ の場合とは異なります。 それは、複数のマルチレベル・ワイルドカードをサポートします。ワイルドカードは、トピック・ストリングの任意の場所の、任意の数の階層レベルで使用可能です。

複数の MQTT v3 クライアントが、リスナー・ポートを使用してデーモンに接続されます。 デフォルトのリスナー・ポートは変更可能です。 複数のリスナー・ポートを定義して、それらに異なるネーム・スペースを割り振ることができます (149 ページの『WebSphere MQ Telemetry デーモン (デバイス用) のリスナー・ポート』を参照)。 デーモン自体が MQTT v3 クライアントとなります。 デーモンのブリッジ接続を構成して、あるデーモンを別のデーモンのリスナー・ポートまたは WebSphere MQ のテレメトリー (MQXR) サービスに接続することができます。

WebSphere MQ Telemetry デーモン (デバイス用) に複数のブリッジを構成することができます。 それらのブリッジを使用して、パブリケーションを交換できるデーモンのネットワークも一緒に接続します。

各ブリッジで、そのローカル・デーモンのトピックにパブリッシュおよびサブスクライブできます。 また、別のデーモン、WebSphere MQ パブリッシュ/サブスクライブ・ブローカー、または接続されている他の MQTT v3 ブローカー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もできます。 トピック・フィルターを使用することで、ブローカー間で伝搬するパブリケーションを選択できます。 パブリケーションはいずれの方向にも伝搬可能です。 つまり、パブリケーションは、あるローカル・デーモンからそれに接続されている各リモート・ブローカーに伝搬することも、接続されているいずれかのブローカーからロ

ーカル・デーモンに伝搬することもできます (142 ページの『IBM WebSphere MQ Telemetry デーモン (デバイス用) のブリッジ』を参照)。

IBM WebSphere MQ Telemetry デーモン (デバイス用) のブリッジ

IBM WebSphere MQ Telemetry デーモン (デバイス用) のブリッジでは、MQTT v3 プロトコルを使用して、2 つのパブリッシュ/サブスクライブ・ブローカーを接続します。 このブリッジによって、ブローカー間でのパブリケーションの両方向の伝搬が行われます。 一方の終端は WebSphere MQ Telemetry デーモン (デバイス用) のブリッジ接続で、もう一方の終端はキュー・マネージャー、または別のデーモンのブリッジ接続の場合があります。 キュー・マネージャーは、テレメトリー・チャネルを使用して、ブリッジ接続に接続されます。 デーモンは、デーモン・リスナーを使用してブリッジ接続に接続されます。

IBM WebSphere MQ Telemetry デーモン (デバイス用) では、他のブローカーへの 1 つ以上の同時接続がサポートされます。 デーモンからの接続はブリッジ接続と呼ばれ、デーモン構成ファイル内の接続項目によって定義されます。 IBM WebSphere MQ への接続は、次の図に示されているように、IBM WebSphere MQ のテレメトリー・チャネルを使用して行われます。

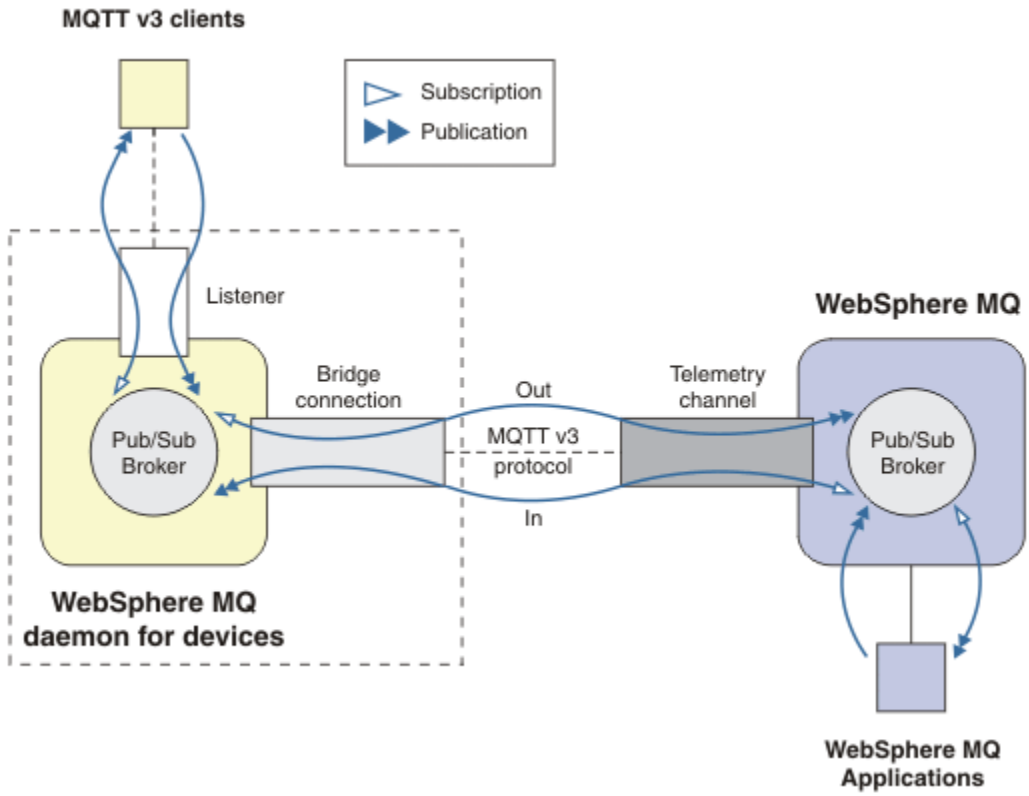


図 *31. IBM WebSphere MQ への IBM WebSphere MQ Telemetry daemon for devices の接続*

ブリッジは、別のブローカーを MQTT v3 クライアントとしてデーモンに接続します。 ブリッジ・パラメーターは、MQTT v3 クライアントの属性をミラーリングします。

ブリッジは接続を行うだけではありません。 2 つのパブリッシュ/サブスクライブ・ブローカー間でパブリッシュおよびサブスクライブ・エージェントとして機能します。 ローカル・ブローカーは IBM WebSphere MQ Telemetry デーモン (デバイス用) であり、リモート・ブローカーは MQTT v3 プロトコルをサポートするパブリッシュ/サブスクライブ・ブローカーです。 通常、リモート・ブローカーは別のデーモンまたは IBM WebSphere MQ となります。

ブリッジの仕事は、2 つのブローカー間でパブリケーションを伝搬することです。 ブリッジは双方向です。 パブリケーションをいずれの方向にも伝搬します。 142 ページの図 31 に、ブリッジが IBM WebSphere MQ Telemetry デーモン (デバイス用) を IBM WebSphere MQ に接続する方法が示されています。 143 ペ

ージの『ブリッジのトピック設定例』では、例を挙げて、トピック・パラメーターを使用してブリッジを構成する方法を示しています。

142 ページの図 31 の In 矢印と Out 矢印は、ブリッジの双方向性を示しています。 矢印の一方の終端で、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と一致するパブリケーションは、矢印の反対側の終端でブローカーにパブリッシュされます。 矢印には、パブリケーションのフローに応じてラベルが付けられています。 パブリケーションのフローは、In の場合はデーモンに入り、Out の場合はデーモンから出て行きます。 ここで重要なのは、ラベルがコマンド構文で使用されることです。 In と Out はパブリケーションのフロー方向を示すものであり、サブスクリプションの送信方向を示すものではないということに注意してください。

他のクライアント、アプリケーション、またはブローカーを、IBM WebSphere MQ または WebSphere MQ Telemetry デーモン (デバイス用) に接続できます。 これらは接続先のブローカーのトピックにパブリッシュおよびサブスクライブします。 ブローカーが IBM WebSphere MQ である場合、トピックがクラスター化または分散されている可能性があり、ローカル・キュー・マネージャーで明示的に定義されていない可能性があります。

ブリッジの用途

複数のブリッジ接続と複数のリスナーを使用して、複数のデーモンを一緒に接続します。 複数のブリッジ接続と複数のテレメトリー・チャネルを使用して、複数のデーモンと複数のキュー・マネージャーを一緒に接続します。 複数のブローカーを一緒に接続する場合は、ループを作成することができます。 その際に注意すべき点は、パブリケーションがブローカーのループを無限に循環し、それが検出されない可能性があることです。

IBM WebSphere MQ にブリッジ接続されるデーモンを使用する理由をいくつか以下に示します。

WebSphere MQ への MQTT クライアントの接続数の削減
デーモンの階層を使用することで、単一のキュー・マネージャーが一度に接続できるクライアントの数よりも多くの数のクライアントを WebSphere MQ に接続できます。

MQTT クライアントと WebSphere MQ 間でのメッセージのストア・アンド・フォワード
クライアントに独自のストレージがない場合は、ストア・アンド・フォワードを使用することで、クライアントと IBM WebSphere MQ 間で接続が継続されないようにすることができます。 その場合、MQTT クライアントと WebSphere MQ の間で複数のタイプの接続を使用することができます (モニターと制御に関するテレメトリーの概念とシナリオを参照)。

MQTT クライアントと WebSphere MQ 間で交換されるパブリケーションのフィルター
通常、パブリケーションでは、メッセージがローカルで処理されるものと、他のアプリケーションに関係するものに分けられます。 ローカル・パブリケーションにはセンサーとアクチュエーター間の制御フローが含まれる場合があり、リモート・パブリケーションには測定値、状況、および構成コマンドの要求が含まれる場合があります。

パブリケーションのトピック・スペースの変更
異なるリスナー・ポートに接続されているクライアントからのトピック・ストリングが、互いに競合しないようにします。 示されている例では、異なる建物から得られたメーター測定値のラベル付けにデーモンを使用しています (異なるクライアント・グループのトピック・スペースの分離を参照)。

ブリッジのトピック設定例

リモート・ブローカーにすべてパブリッシュする (デフォルトを使用)

デフォルトの方向は out と呼ばれ、ブリッジはトピックをリモート・ブローカーにパブリッシュします。 topic パラメーターは、トピック・フィルターを使用して、伝搬されるトピックを制御します。

ブリッジは、144 ページの図 32 の topic パラメーターを使用して、MQTT クライアントまたは他のブローカーによってローカル・デーモンにパブリッシュされたすべてのものをサブスクライブします。 ブリッジは、ブリッジによって接続されているリモート・ブローカーにトピックをパブリッシュします。

```
connection Daemon1
topic #
```

図 *32.* リモート・ブローカーにすべてパブリッシュする

リモート・ブローカーにすべてパブリッシュする (明示的)

以下のコード・フラグメントにある topic の設定値を使用すると、デフォルトを使用した場合と同じ結果になります。 唯一の違いは、**direction** パラメーターが明示的である点です。 out 方向を使用して、ローカル・ブローカーであるデーモンにサブスクライブし、リモート・ブローカーにパブリッシュします。 ブリッジがサブスクライブしたローカル・デーモンで作成されたパブリケーションは、リモート・ブローカーでパブリッシュされます。

```
connection Daemon1
topic # out
```

図 *33.* リモート・ブローカーにすべてパブリッシュする *(*明示的*)*

ローカル・ブローカーにすべてパブリッシュする

out 方向を使用する代わりに、逆方向の in を設定することができます。 以下のコード・フラグメントでは、ブリッジによって接続されたリモート・ブローカーでパブリッシュされたすべてのものにサブスクライブするように、ブリッジが構成されています。 ブリッジは、ローカル・ブローカーであるデーモンにトピックをパブリッシュします。

```
connection Daemon1
topic # in
```

図 *34.* ローカル・ブローカーにすべてパブリッシュする

ローカル・ブローカーのエクスポート・トピックからリモート・ブローカーのインポート・トピックにすべてのものをパブリッシュする

2 つの追加トピック・パラメーター (**local_prefix** と **remote_prefix**) を使用して、前の例のトピック・フィルター # を変更します。 1 つのパラメーターはサブスクリプションで使用されるトピック・フィルターの変更に使用され、もう 1 つのパラメーターはパブリケーションがパブリッシュされるトピックの変更に使用されます。 その結果、一方のブローカーで使用されるトピック・ストリングの先頭が、もう一方のブローカーの別のトピック・ストリングに置き換えられます。

トピック・コマンドの方向に応じて、**local_prefix** と **remote_prefix** の意味は逆になります。 方向が out の場合 (デフォルト) は、**local_prefix** がトピック・サブスクリプションの一部として使用され、リモート・パブリケーションのトピック・ストリングの **local_prefix** 部分が **remote_prefix** に置き換えられます。 方向が in の場合は、**remote_prefix** がリモート・サブスクリプションの一部になり、トピック・ストリングの **remote_prefix** 部分が **local_prefix** に置き換えられます。

トピック・ストリングの最初の部分は、多くの場合、トピック・スペースを定義するものと見なされます。 追加パラメーターを使用して、トピックがパブリッシュされるトピック・スペースを変更します。この操作を行うことで、伝搬されるトピックがターゲット・ブローカーの別のトピックと競合しないようにしたり、マウント・ポイント・トピック・ストリングを削除したりすることができます。

例えば、以下のコード・フラグメントでは、デーモン上のトピック・ストリング export/# へのすべてのパブリケーションが、リモート・ブローカー上の import/# にリパブリッシュされます。

```
topic # out export/ import/
```

図 *35.* ローカル・ブローカーのエクスポート・トピックからリモート・ブローカーのインポート・トピックにすべてのものをパブリッシュする

リモート・ブローカーのエクスポート・トピックからローカル・ブローカーのインポート・トピックにすべてのものをパブリッシュする

以下のコード・フラグメントは、逆の構成を示しています。つまり、ブリッジは、リモート・ブローカーで export/# トピック・ストリングを使用してパブリッシュされたものすべてにサブスクライブし、それをローカル・ブローカーの import/# にパブリッシュします。

```
connection Daemon1
topic # in import/ export/
```

図 *36.* リモート・ブローカーのエクスポート・トピックからローカル・ブローカーのインポート・トピックにすべてのものをパブリッシュする

元のトピック・ストリングを使用して、1884/ マウント・ポイントからリモート・ブローカーにすべてのものをパブリッシュする

以下のコード・フラグメントでは、ブリッジは、ローカル・デーモンでマウント・ポイント 1884/ に接続されたクライアントによってパブリッシュされたものすべてにサブスクライブします。 ブリッジは、マウント・ポイントにパブリッシュされたすべてのものをリモート・ブローカーにパブリッシュします。 マウント・ポイントのストリング 1884/ は、リモート・ブローカーにパブリッシュされたトピックから削除されます。 *local_prefix* はマウント・ポイントのストリング 1884/ と同じであり、*remote_prefix* はブランク・ストリングです。

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

図 *37.* 元のトピック・ストリングを使用して、*1884/* マウント・ポイントからリモート・ブローカーにすべてのものをパブリッシュする。

異なるデーモンに接続されている異なるクライアントのトピック・スペースを分離する

あるアプリケーションが、電力メーターが建物のメーター測定値をパブリッシュするために作成されたとします。 測定値は、MQTT クライアントを使用して、同じ建物にホストされているデーモンにパブリッシュされます。 パブリケーション対象として選択されたトピックは power です。 同じアプリケーションが、複合施設内のいくつかの建物にデプロイされます。 複合施設全体をモニターしデータを保管するために、すべての建物から得られた測定値は、ブリッジ接続を使用して集約されます。 この接続により、建物のデーモンが中央の WebSphere MQ にリンクされます。

それぞれの建物のクライアント・アプリケーションは同じですが、データは建物ごとに区別される必要があります。 各測定値には power トピックがあり、区別するために建物の番号を接頭部として付ける必要があります。 複合施設内の最初の建物からのブリッジでは、接頭部として meters/building01/ を使用し、2 つ目の建物からのブリッジの場合、接頭部は meters/building02/ となります。 他の建物からの測定値も同じパターンに従います。 WebSphere MQ は、 meters/building01/power のようなトピックの読み取り値を受け取ります。

アプリケーションがパブリッシュするトピック・スペースを実際に構成できる可能性のある例を以下に示します。

各デーモンの構成ファイルには、以下のコード・フラグメントのパターンに従ったトピック・ステートメントがあります。

```
connection Daemon1
topic power out "" meters/building01/
```

図 *38.* 異なるデーモンに接続されているクライアントのトピック・スペースを分離する

local_prefix パラメーターを使用しない部分のプレースホルダーとして、空ストリングを指定します。

同じデーモンに接続されているクライアントのトピック・スペースを分離する

すべての電力メーターを接続する際に単一のデーモンを使用すると仮定します。 複数の異なるポートに接続するようにアプリケーションを構成できると想定すると、以下のコード・フラグメントにあるように、建物ごとに異なるメーターを異なるリスナー・ポートに接続することにより、建物を区別できます。 ここでは、マウント・ポイントを使用する可能性がある例を示します。

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

図 *39.* 同じデーモンに接続されているクライアントのトピック・スペースを分離する

両方向に流れるパブリケーションの異なるトピックをリマップする

以下のコード・フラグメントの構成では、ブリッジはリモート・ブローカーの単一トピック b にサブスクライブし、b に関するパブリケーションをローカル・デーモンに転送して、トピックを a に変更します。 また、ブリッジは、ローカル・ブローカーの単一トピック x にサブスクライブし、 x に関するパブリケーションをリモート・ブローカーに転送して、トピックを y に変更します。

```
connection Daemon1
topic "" in a b
topic "" out x y
```

図 *40.* 両方向に流れるパブリケーションの異なるトピックをリマップする

この例での重要な点は、異なるトピックが両方のブローカーでサブスクライブおよびパブリッシュされることです。 両方のブローカーのトピック・スペースはそれぞれ独立したものです。

両方向に流れるパブリケーションの同じトピックをリマップする (ループ)

前の例とは異なり、147 ページの図 41 の構成では、通常、ループになります。 トピック・ステートメント `topic "" in a b` の場合、ブリッジはリモートで b にサブスクライブし、ローカルで a にパブリッシュします。 もう一方のトピック・ステートメントの場合、ブリッジはローカルで a にサブスクライブし、リモートで b にパブリッシュします。 同じ構成を、147 ページの図 42 にあるように作成することができます。

一般的な結果として、クライアントがリモート側で b にパブリッシュした場合、パブリケーションはトピック a のパブリケーションとしてローカル・デーモンに転送されます。 ただし、ブリッジによってトピック a のローカル・デーモンにパブリッシュされると、パブリケーションは、ブリッジによってローカル・トピック a に対して作成されたサブスクリプションと一致します。 サブスクリプションは `topic "" out a b` です。 結果として、パブリケーションはトピック b のパブリケーションとしてリモート・ブローカーに戻されます。 これで、ブリッジはリモート・トピック b にサブスクライブされ、サイクルが再び開始されます。

一部のブローカーでは、ループが発生しないようにループ検出が実装されています。 しかし、ループ検出メカニズムは、異なるタイプのブローカーが一緒にブリッジ接続される場合にも機能する必要があります。 WebSphere MQ が WebSphere MQ Telemetry デーモン (デバイス用) にブリッジ接続される場合は、ループ検出は機能しません。 2 つの IBM WebSphere MQ Telemetry デーモン (デバイス用) が一緒にブリッジ接続される場合には機能します。 デフォルトでは、ループ検出はオンに設定されています (try_private を参照)。

```
connection Daemon1
topic "" in a b
topic "" out a b
```

図 *41. !*両方向に流れるパブリケーションについて同じトピックを再マップする

```
connection Daemon1
topic "" both a b
```

図 *42. !both* を使用して、両方向に流れるパブリケーションについて同じトピックを再マップします。

146 ページの図 40 の構成は 147 ページの図 41 と同じです。

IBM WebSphere MQ Telemetry daemon for devices のブリッジ接続の可用性

最初に使用可能なリモート・ブローカーに接続するように、複数の IBM WebSphere MQ Telemetry daemon for devices のブリッジ接続アドレスを構成します。 ブローカーが複数インスタンスのキュー・マネージャーの場合は、その両方の TCP/IP アドレスを指定します。 使用可能な場合は、1 次サーバーに接続または再接続するように 1 次接続を構成します。

接続ブリッジ・パラメーター addresses は、TCP/IP ソケット・アドレスのリストです。 ブリッジは、正常に接続されるまで、各アドレスへの接続を順に試行します。 round_robin および start_type 接続パラメーターは、正常に接続された後のアドレスの使用方法を制御します。

start_type が auto、manual、または lazy の場合、接続に失敗すると、ブリッジは再接続を試行します。 ブリッジは、アドレスを 1 つずつ順に使用します。この場合の接続試行の間隔は約 20 秒となります。 `start_type` が once の場合、接続に失敗すると、ブリッジは自動的に再接続を試行しません。

round_robin が true の場合、ブリッジによる接続試行はリストの最初のアドレスから開始され、リストのアドレスを 1 つずつ順に試します。 リストの最後のアドレスでの試行が終わると、再び、最初のアドレスから試行が開始されます。 リストにアドレスが 1 つしかない場合は、20 秒ごとにそのアドレスで再試行されます。

round_robin が false の場合、リストの最初のアドレス (1 次サーバーと呼ぶ) が優先されます。 1 次サーバーへの最初の接続試行に失敗した場合、ブリッジは引き続き、バックグラウンドで 1 次サーバーへの再接続を試行します。 同時に、ブリッジは、リスト内の他のアドレスを使用して、接続を試行します。 バックグラウンドでの 1 次サーバーへの接続試行に成功した場合、ブリッジは現行の接続を切断し、1 次サーバー接続に切り替えます。

接続が、例えば **connection_stop** コマンドを実行して、任意に切断された場合、接続が再開されると、再度、同じアドレスを使用して接続が試行されます。 接続の失敗、またはリモート・ブローカーでの接続の中断が原因で接続が切断された場合、ブリッジは 20 秒間待機します。 その後、リスト内の次のアドレスへの接続を試行するか、リストにアドレスが 1 つしかない場合は同じアドレスへの接続を試行します。

複数インスタンス・キュー・マネージャーへの接続

複数インスタンス・キュー・マネージャーの構成では、キュー・マネージャーは異なる IP アドレスを持つ 2 つの異なるサーバーで実行されます。 通常、テレメトリー・チャネルは特定の IP アドレスを指定せずに構成されます。 これらのチャネルはポート番号のみを指定して構成されます。 テレメトリー・チャネルが開始されると、デフォルトで、ローカル・サーバー上の最初の使用可能なネットワーク・アドレスが選択されます。

ブリッジ接続の addresses パラメーターは、キュー・マネージャーで使用される 2 つの IP アドレスで構成します。 round_robin は true に設定します。

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、キュー・マネージャーはスタンバイ・インスタンスに切り替わります。 デーモンは、アクティブ・インスタンスへの接続が切断したことを検出し、スタンバイ・インスタンスへの再接続を試行します。 その場合、ブリッジ接続用に構成されたアドレスのリストにある他の IP アドレスが使用されます。

ブリッジの接続先のキュー・マネージャーは、引き続き同じキュー・マネージャーです。 キュー・マネージャーの状態はキュー・マネージャー自体が回復させます。 cleansession が false に設定されている場合、ブリッジ接続セッションは、フェイルオーバーの前と同じ状態に復元されます。 接続は遅延後に再開されます。 "少なくとも 1 回" または "最大 1 回" のサービス品質を持つメッセージは失われず、サブスクリプションは引き続き機能します。

再接続時間は、スタンバイ・インスタンスの開始時に再開するクライアントとチャネルの数、および未完了のメッセージ数によって異なります。 ブリッジ接続の場合、両方の IP アドレスへの再接続は、接続が再確立されるまで何度でも試行できます。

複数インスタンス・キュー・マネージャーのテレメトリー・チャネルは、特定の IP アドレスを使用して構成しないでください。 IP アドレスは 1 つのサーバーでのみ有効です。

IP アドレスを管理する代替の高可用性ソリューションを使用するのであれば、特定の IP アドレスを使用してテレメトリー・チャネルを構成するのが適切な場合があります。

cleansession

ブリッジ接続は MQTT v3 クライアント・セッションです。 接続で新規セッションを開始するか、既存のセッションを復元するかを制御できます。 既存のセッションを復元する場合、ブリッジ接続では、前のセッションからのサブスクリプションと保存パブリケーションが保持されます。

addresses に複数の IP アドレスがリストされており、IP アドレスが異なるキュー・マネージャーにホストされているテレメトリー・チャネルに接続されているか、異なるテレメトリー・デーモンに接続されている場合は、cleansession を false に設定しないでください。 セッション状態は、キュー・マネージャー間やデーモン間では転送されません。 異なるキュー・マネージャーまたはデーモンで既存のセッションを再開しようとすると、新規セッションが開始されます。 未確定メッセージは失われ、サブスクリプションが予期したとおりに動作しない可能性があります。

notifications

アプリケーションは、notifications を使用して、ブリッジ接続が実行されているかどうかを把握できます。 通知は、値が 1 (接続) または 0 (切断) のパブリケーションです。 これは、notification_topic パラメーターで定義された *topicString* にパブリッシュされます。 *topicString* のデフォルト値は $SYS/broker/connection/ *clientIdentifier*/state です。 デフォルトの *topicString* には、接頭部 $SYS が含まれます。 $SYS で始まるトピックには、$SYS で始まるトピック・フィルターを定義してサブスクライブします。 トピック・フィルター # (すべてにサブスクライブ) の場合、デーモンの $SYS で始まるトピックにはサブスクライブしません。 $SYS は、アプリケーション・トピック・スペースとは異なる特殊なシステム・トピック・スペースを定義するものと考えてください。

notifications により、IBM WebSphere MQ Telemetry daemon for devices はブリッジの接続時または切断時に MQTT クライアントへ通知できます。

keepalive_interval

keepalive_interval ブリッジ接続パラメーターは、ブリッジが TCP/IP ping をリモート・サーバーに送信する間隔を設定します。 デフォルトの間隔は 60 秒です。 ping は、リモート・サーバーまたはファイアウォールで接続の非アクティブ期間が検出されて、TCP/IP セッションが終了されないようにします。

CLIENTID

ブリッジ接続は MQTT v3 クライアント・セッションであり、ブリッジ接続パラメーター clientid で設定される clientIdentifier を持ちます。 cleansession パラメーターを false に設定して、前のセッション

を再開するために再接続する場合は、各セッションで使用する `clientIdentifier` が同じでなければなりません。 clientid のデフォルト値は *hostname.connectionName* で、この値は変わりません。

WebSphere MQ Telemetry デーモン (デバイス用) のインストール、検査、構成、および制御

デーモンのインストール、構成、および制御はファイル・ベースで行います。

デーモン を実行する予定のデバイスに Software Development Kit をコピーして、デーモンをインストールします。

例として、MQTT クライアント・ユーティリティーを実行し、パブリッシュ/サブスクライブ・ブローカーとして WebSphere MQ Telemetry デーモン (デバイス用) に接続します。 特定の MQTT v3 クライアントへのメッセージのパブリッシュ を参照してください。

デーモンの構成は、構成ファイルを作成して行います。WebSphere MQ Telemetry デーモン (デバイス用) の構成ファイルを参照してください。

実行中のデーモンの制御は、`amqtdd.upd` ファイルにコマンドを作成して行います。 デーモンは、5 秒ごとにこのファイルを読み取り、コマンドを実行して、このファイルを削除します。WebSphere MQ Telemetry デーモン (デバイス用) のコマンド・ファイルを参照してください。

WebSphere MQ Telemetry デーモン (デバイス用) のリスナー・ポート

リスナー・ポートを使用して、MQTT V3 クライアントを WebSphere MQ Telemetry デーモン (デバイス用) に接続します。 リスナー・ポートには、マウント・ポイントと最大接続数を指定できます。

リスナー・ポートは、このポートに接続するクライアントの MQTT クライアント `connect(serverURI)` メソッドで指定されたポート番号に対応している必要があります。 デフォルトでは、クライアントとデーモンの両方で 1883 に設定されています。

デーモンのデフォルト・ポートは、デーモン構成ファイルでグローバル定義 `port` を設定することにより変更可能です。 `listener` 定義をデーモン構成ファイルに追加することで、特定のポートを設定できます。

デフォルト・ポート以外の各リスナー・ポートでは、マウント・ポイントを指定して、クライアントを分離することができます。 マウント・ポイントを持つポートに接続されるクライアントは、他のクライアントから分離されます (149 ページの『WebSphere MQ Telemetry デーモン (デバイス用) のマウント・ポイント』を参照)。

任意のポートに接続可能なクライアントの数を制限することができます。 グローバル定義 `max_connections` を設定し、デフォルト・ポートへの接続を制限します。または、各リスナー・ポートに `max_connections` を指定します。

例

構成ファイルの例を以下に示します。デフォルト・ポートを 1883 から 1880 に変更し、ポート 1880 への接続数を 10000 に制限しています。 ポート 1884 への接続数を 1000 に制限しています。 ポート 1884 に接続されているクライアントは、他のポートに接続されているクライアントから分離されます。

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

WebSphere MQ Telemetry デーモン (デバイス用) のマウント・ポイント

MQTT クライアントで使用されているリスナー・ポートにマウント・ポイントを関連付けることで、WebSphere MQ Telemetry デーモン (デバイス用) に接続することができます。 マウント・ポイントにより、あるリスナー・ポートを使用する MQTT クライアントで交換されるパブリケーションおよびサブスクリプションを、別のリスナー・ポートに接続されている MQTT クライアントから分離します。

マウント・ポイントを指定したリスナー・ポートに接続されたクライアントは、その他のリスナー・ポートに接続されているクライアントと直接トピックを交換することはできません。 マウント・ポイントを指定していないリスナー・ポートに接続されたクライアントは、どのクライアントのトピックにもパブリッシュおよびサブスクライブを行うことができます。 クライアントは、マウント・ポイントを介して接続されているかどうかを認識しないため、クライアントで作成されたトピック・ストリングに影響はありません。

マウント・ポイントは、パブリケーションおよびサブスクリプションのトピック・ストリングの接頭部として付けられるテキストのストリングです。 マウント・ポイントを指定したリスナー・ポートに接続されたクライアントで作成されたすべてのトピック・ストリングに、接頭部として付けられます。 このテキストのストリングは、リスナー・ポートに接続されているクライアントに送信されたすべてのトピック・ストリングから削除されます。

リスナー・ポートにマウント・ポイントが指定されていない場合、ポートに接続されているクライアントで作成および受信されたパブリケーションとサブスクリプションのトピック・ストリングは変更されません。

末尾に / が付いたマウント・ポイント・ストリングを作成します。 これにより、マウント・ポイントが、そのマウント・ポイントのトピック・ツリーの親トピックとなります。

例

構成ファイルには以下のリスナー・ポートが含まれています。

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

ポート 1883 に接続されているあるクライアントが、MyTopic へのサブスクリプションを作成します。 デーモンはそのサブスクリプションを 1883/MyTopic として登録します。 ポート 1883 に接続されている別のクライアントは、トピック MyTopic でメッセージをパブリッシュします。 デーモンは、トピック・ストリングを 1883/MyTopic に変更し、一致するサブスクリプションを検索します。 ポート 1883 のこのサブスクライバーは、元のトピック・ストリングが MyTopic のパブリケーションを受信します。 デーモンはトピック・ストリングからマウント・ポイント接頭部を削除します。

ポート 1884 に接続されている別のクライアントも、トピック MyTopic でパブリッシュを行います。 この場合、デーモンはトピックを 1884/MyTopic として登録します。 ポート 1883 のサブスクライバーはパブリケーションを受信しません。これは、異なるマウント・ポイントでは異なるトピック・ストリングのサブスクリプションが作成されるためです。

ポート 1885 に接続されているクライアントが、トピック 1883/MyTopic でパブリッシュを行います。 この場合、デーモンはトピック・ストリングを変更しません。 ポート 1883 のサブスクライバーは MyTopic へのパブリケーションを受信します。

デバイス用 WebSphere MQ Telemetry デーモンのサービスの品質、永続サブスクリプション、および保存パブリケーション

サービスの品質の設定は、実行中のデーモンに対してのみ適用されます。 デーモンが停止すると、その停止が制御下で行われたものであっても、障害が原因であっても、送信中のメッセージの状態は失われます。 デーモンが停止した場合、メッセージが少なくとも 1 回、あるいは多くても 1 回送達されるかどうかは保証されません。 デバイス用 WebSphere MQ Telemetry デーモンがサポートするのは、制限付きの持続性です。 デーモンがシャットダウンしたときに保存パブリケーションとサブスクリプションを保存するには、**retained_persistence** 構成パラメーターを設定します。

WebSphere MQ とは異なり、デバイス用 WebSphere MQ Telemetry デーモンは、持続データをジャーナル処理しません。 セッション状態、メッセージ状態、および保存パブリケーションは、トランザクションとして保存されません。 デフォルトでは、このデーモンは、停止するとすべてのデータを廃棄します。 オプションを設定することにより、定期的にサブスクリプションと保存パブリケーションのチェックポイント

を指定できます。 このデーモンが停止すると、メッセージの状況は常に失われます。 非保存パブリケーションはすべて失われます。

保存パブリケーションをファイルに定期的に保存するには、デーモン構成オプション Retained_persistence を true に設定します。 このデーモンが再始動すると、最後に自動保存された保存パブリケーションが復元されます。 デフォルトでは、クライアントによって作成された保存メッセージは、デーモンの再始動時には復元されません。

持続セッションで作成されたサブスクリプションを定期的にファイルに保存するには、デーモン構成オプション Retained_persistence を true に設定します。 Retained_persistence が true に設定されている場合、CleanSession が false("持続セッション") に設定されたセッションでクライアントが作成したサブスクリプションが復元されます。 このデーモンは、再始動時にそれらのサブスクリプションを復元し、復元されたサブスクリプションがパブリケーションの受信を開始します。 クライアントがパブリケーションを受信するのは、CleanSession を false に設定してクライアントが再始動したときです。 デフォルトでは、デーモンが停止した場合、クライアント・セッション状態は保存されないため、クライアントが CleanSession を false に設定していても、サブスクリプションは復元されません。

Retained_persistence は自動保存のための仕組みです。 直近の保存パブリケーションあるいはサブスクリプションは保存されない場合があります。 保存パブリケーションおよびサブスクリプションの保存頻度は変更できます。 次の保存までの間隔、または次の保存までに行われた変更の数を、構成オプション autosave_on_changes および autosave_interval を使用して設定します。

持続性を設定するための構成例

```
# Sample configuration
# Daemon listens on port 1882 with persistence in /tmp
# Autosave every minute
port 1882
persistence_location /tmp/
retained_persistence true
autosave_on_changes false
autosave_interval 60
```

WebSphere MQ Telemetry デーモン (デバイス用) のセキュリティー

WebSphere MQ Telemetry デーモン (デバイス用) では、接続するクライアントの認証、他のブローカーに接続するための資格情報の使用、およびトピックへのアクセスの制御を行えます。 デーモンで提供されるセキュリティーは、WebSphere MQ Telemetry C クライアントを使用してビルドされることにより限定的なものになっています。これは SSL サポートを提供していません。 したがって、デーモンとの接続は暗号化されず、証明書を使用して認証することはできません。

デフォルトでは、セキュリティーはオンになっていません。

クライアントの認証

MQTT クライアントでは、MqttConnectOptions.setUserName メソッドと MqttConnectOptions.setPassword メソッドを使用して、ユーザー名およびパスワードを設定することができます。

デーモンに接続されているクライアントの認証は、クライアントが提供するユーザー名およびパスワードを、パスワード・ファイル内の項目と照合することにより行います。 認証を使用可能にするには、パスワード・ファイルを作成し、デーモン構成ファイルで password_file パラメーターを設定します (password_file を参照)。

デーモン構成ファイルに allow_anonymous パラメーターを設定すると、ユーザー名やパスワードを設定せずに接続したクライアントを、認証を確認するデーモンに接続することができます (allow_anonymous を参照)。 クライアントでユーザー名またはパスワードが提供され、password_file パラメーターが設定されている場合は、常にパスワード・ファイルとの照合が行われます。

デーモン構成ファイルに clientid_prefixes パラメーターを設定すると、接続を特定のクライアントに制限します。 接続するクライアントは、clientid_prefixes パラメーターにリストされているいずれかの接頭部で始まる clientIdentifiers を持たなければなりません (clientid_prefixes を参照)。

ブリッジ接続セキュリティー

ブリッジ接続される各 WebSphere MQ Telemetry デーモン (デバイス用) が MQTT V3 クライアントとなります。 各ブリッジ接続に対して、デーモン構成ファイルでブリッジ接続パラメーターとしてユーザー名とパスワードを設定することができます (username および password を参照)。 その結果、ブリッジ自体をブローカーに対して認証できるようになります。

トピックへのアクセス制御

クライアントが認証されている場合、デーモンで、トピックへのアクセス制御を各ユーザーに対して行うこともできます。 デーモンは、クライアントがパブリッシュまたはサブスクライブするトピックと、アクセス制御ファイル内のアクセス・トピック・ストリングとのマッチングに基づいて、アクセス制御を許可します (acl_file を参照)。

アクセス制御リストは 2 つの部分に分かれています。 1 つ目の部分では、匿名クライアントを含むすべてのクライアントからのアクセスを制御します。 2 つ目の部分には、パスワード・ファイル内の任意のユーザーに関するセクションがあります。 ここには、各ユーザーに対する特定のアクセス制御がリストされます。

例

以下に、セキュリティー・パラメーターの例を示します。

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

図 *43.* デーモン構成ファイル

```
Fred:Fredpassword
Barney:Barneypassword
```

図 *44.* パスワード・ファイル *passwords.txt*

```
topic home/public/#
topic read meters/#
user Fred
topic write meters/fred
topic home/fred/#
user Barney
topic write meters/barney
topic home/barney/#
```

図 *45.* アクセス制御ファイル *acl.txt*

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 WebSphere 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、WebSphere 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

WebSphere 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 153 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 WebSphere MQ Multicast のトピック・トポロジー

この例を利用して、IBM WebSphere 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 WebSphere 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の全範囲は 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詳しくは、システム管理者に連絡を取るか、IPv4 Multicast Address Space Registry 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。

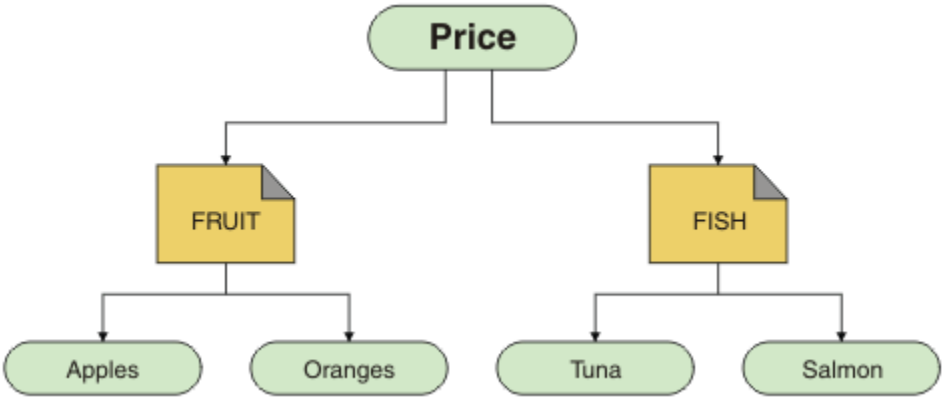
以下の図には、2 つの可能なマルチキャスト・データ・ストリームがあります。

```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)

DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

ここで、 *239.XXX.XXX.XXX* と *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。

WebSphere 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、WebSphere MQ メッセージ形式について学習したり、WebSphere 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。

WebSphere 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。 WebSphere 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、 155 ページの表 9 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、WebSphere 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 WebSphere Message Broker 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。

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、メッセージが MCPROP の値に従って送信された場合の動作と、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 9. 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

表 9. 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**
MQMD Version	伝送されない	適用外
レポート	デフォルト以外の場合に含まれる	0
MsgType	デフォルト以外の場合に含まれる	MQMT_DATAGRAM
Expiry	デフォルト以外の場合に含まれる	0
Feedback	デフォルト以外の場合に含まれる	0
Encoding	デフォルト以外の場合に含まれる	MQENC_NORMAL(equiv)
CodedCharSetId	デフォルト以外の場合に含まれる	1208
Format	デフォルト以外の場合に含まれる	MQRFH2
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、WebSphere MQ Multicast メッセージングでデータ変換が機能する方法を理解するために使用します。

WebSphere MQ Multicast は、共有のコネクションレス・プロトコルです。このため各クライアントは、データ変換に対して固有の要求をすることができません。 同じマルチキャスト・ストリームをサブスクライブするすべてのクライアントは、同じバイナリー・データを受け取ります。したがって、WebSphere MQ データ変換が必要な場合、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。

WebSphere 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 WebSphere 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。

キュー・マネージャーに接続されているアプリケーションに関する接続情報を表示するには、MQSC コマンド **DISPLAY CONN** を使用します。 **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、WebSphere 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 WebSphere 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
イベント・メッセージは、メッセージ信頼性が信頼性しきい値未満の場合に書き込まれます。 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、WebSphere 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、WebSphere MQ の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、WebSphere 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。

WebSphere 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。

WebSphere 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

この情報は、.ini ファイルの構成および WebSphere MQ LLM での相互運用性などの、拡張 WebSphere 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・インストール済み環境でのセキュリティーに関する考慮事項については、 マルチキャスト・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 WebSphere 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

この情報を利用して、.ini ファイル内の WebSphere MQ Multicast フィールドについて理解を深めてください。

ini ファイル内で追加の WebSphere MQ Multicast 構成を行うことができます。 以下のように、アプリケーションのタイプによって、使用する必要がある特定の ini ファイルが違ってきます。

- 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。

MQ_DATA_PATH は WebSphere MQ データ・ディレクトリーの場所 (/var/mqm/mqclient.ini) で、*QMNAME* は .ini ファイルの適用先のキュー・マネージャーの名前です。

.ini ファイルには、WebSphere MQ Multicast: の動作を微調整するために使用するフィールドが含まれています。

```
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。 これはデフォルト値です。

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。 これはデフォルト値です。

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、IPv6、またはその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。 これはデフォルト値です。

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。 これはデフォルト値です。

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。 偽のハートビート・タイムアウト・イベントが発生しないように、ハートビート間隔は **HeartbeatTimeout** 値よりかなり小さくなければなりません。 デフォルト値は 2000 です。

WebSphere 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、WebSphere MQ Multicast と WebSphere 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 WebSphere 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、WebSphere 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 *10. WebSphere MQ* メッセージ・プロパティーから *WebSphere MQ LLM* プロパティーへのマッピング

WebSphere MQ プロパティー	**WebSphere 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: WebSphere MQ Low Latency Messaging を参照してください。

HP Integrity NonStop Server の管理

この情報を使用して、IBM WebSphere MQ Client for HP Integrity NonStop Server の管理タスクについて学習します。

以下の 2 つの管理タスクを実行できます。

1. Pathway からの TMF/Gateway の手動による開始。
2. Pathway からの TMF/Gateway の停止。

TMF/Gateway を Pathway から手動で開始する

最初の参加要求で Pathway に TMF/Gateway を自動的に開始させることも、Pathway から TMF/Gateway を手動で開始することもできます。

手順

Pathway から TMF/Gateway を手動で開始するには、次の PATHCOM コマンドを入力します。

```
START SERVER <server_class_name>
```

TMF/Gateway が未確定トランザクションの回復を完了する前に、クライアント・アプリケーションが登録要求を行うと、その要求は 1 秒間保留されます。 その時間内にリカバリーが完了しなければ、参加は拒否されます。 その後、クライアントはトランザクション MQI を使用すると MQRC_UOW_ENLISTMENT_ERROR エラーを受け取ります。

TMF/Gateway を Pathway から停止する

このタスクでは、TMF/Gateway を Pathway から停止する方法、および停止した TMF/Gateway を再始動する方法について説明します。

手順

1. TMF/Gateway に対して新規参加要求が行われないようにするには、次のコマンドを入力します。

   ```
   FREEZE SERVER <server_class_name>
   ```

2. TMF/Gateway にすべての未完了操作を完了させて終了させるには、次のコマンドを入力します。

   ```
   STOP SERVER <server_class_name>
   ```

3. TMF/Gateway を最初の参加で自動的に再始動させるか、手動で再始動させるには、ステップ 1 および 2 の後に、次のコマンドを入力します。

   ```
   THAW SERVER <server_class_name>
   ```

   アプリケーションは新規登録要求を行うことができず、**THAW** コマンドを発行するまで **START** コマンドを発行することはできません。

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。

IBM 〒 103-8510 東京都中央区日本橋箱崎町 19 番 21 号 IBM Corporation 日本アイ・ビー・エム株式会社 法務・知的財産知的財産権ライセンス渉外 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:

〒 103-8510 東京都中央区日本橋箱崎町 19-21 IBM 日本アイ・ビー・エム株式会社法務・知的財産権ライセンス渉外

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。

IBM Corporation ソフトウェア・インターオペラビリティー・コーディネーター、部門 49XA 3605 道路 52 N ロチェスター、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 では、それら製品のテストを行っていません。したがって、IBM 以外の製品の性能や互換性などに関する主張の正確性については確約できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。これらの名前はすべて架空のものであり、名前や住所が類似する個人や企業が実在して いるとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報 (提供されている場合) は、このプログラムで使用するアプリケーション・ソフトウェアの作成を支援することを目的としています。

本書には、プログラムを作成するユーザーが IBM WebSphere MQ のサービスを使用するためのプログラミング・インターフェースに関する情報が記載されています。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。 診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

重要 : この診断、修正、およびチューニング情報は、変更される可能性があるため、プログラミング・インターフェースとして使用しないでください。

商標

IBM、 IBM ロゴ、 ibm.com®は、世界の多くの国で登録された IBM Corporation の商標です。 現時点での IBM の商標リストについては、 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml をご覧ください。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

この製品には、 Eclipse Project (http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

部品番号:

(1P) P/N: