BEA Tuxedo®

BEA Tuxedo /Q コンポーネント

BEA Tuxedo 8.1
2003 年 1 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

目次

3. BEA Tuxedo /Q の C 言語プログラミング

4. BEA Tuxedo/Q COBOL 言語プログラミング

A. サンプル・アプリケーション

このマニュアルについて

このマニュアルでは、BEA Tuxedo 環境で /Q コンポーネントを設定、プログラミング、および使用する方法について説明します。BEA Tuxedo /Q コンポーネントを使用すると、メッセージを永続的な記憶域 (ディスク) や一時的な記憶域 (メモリ) のキューに登録でき、後で取り出して処理できます。

このマニュアルでは、以下の内容について説明します。

- 第 1 章「BEA Tuxedo /Q の概要」では、BEA Tuxedo /Q コンポーネントのアーキテクチャの概要と、/Q コンポーネントを使用するために必要な管理タスクおよびプログラミング・タスクを説明します。
- 第 2 章「BEA Tuxedo /Q の管理」では、BEA Tuxedo /Q コンポーネントの設定方法と管理方法を説明します。
- 第 3 章「BEA Tuxedo /Q の C 言語プログラミング」では、C 言語関数を使用して、メッセージをキューに登録したりキューから取り出す方法を説明します。
- 第 4 章「BEA Tuxedo/Q COBOL 言語プログラミング」では、COBOL 言語関数を使用して、メッセージをキューに登録したりキューから取り出す方法を説明します。
- 付録 A 「サンプル・アプリケーション」では、BEA Tuxedo の /Q コンポーネントを使用したクライアント・サーバ・アプリケーションのサンプルを示します。

対象読者

このマニュアルは、次のような読者を対象としています。

- BEA Tuxedo 環境でメッセージ・キュー・アプリケーションの設定と管理を担当する管理者
- BEA Tuxedo 環境でメッセージ・キュー・アプリケーションのプログラミングを行うアプリケーション開発者

このマニュアルは、BEA Tuxedo プラットフォーム、および C 言語または COBOL 言語のプログラミングの知識があることを前提としています。

e-docs Web サイト

BEA 製品のマニュアルは BEA 社の Web サイト上で参照することができます。BEA ホーム・ページの [製品のドキュメント] をクリックするか、または http://edocs.beasys.co.jp/e-docs/index.html に直接アクセスしてください。

マニュアルの印刷方法

このマニュアルは、ご使用の Web ブラウザで一度に 1 ファイルずつ印刷できます。Web ブラウザの [ファイル] メニューにある [印刷] オプションを使用してください。

このマニュアルの PDF 版は、e-docs Web サイトの BEA Tuxedo マニュアル・ページから入手できます。また、マニュアルの CD-ROM にも収められています。この PDF を Adobe Acrobat Reader で開くと、マニュアル全体または

一部をブック形式で印刷できます。PDF 形式を利用するには、BEA Tuxedo Documents ページの [PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader をお持ちではない場合は、Adobe Web サイト (http://www.adobe.co.jp/) から無償でダウンロードできます。

関連情報

以下の BEA Tuxedo マニュアルには、BEA Tuxedo 環境で BEA Tuxedo /Q コンポーネントを使用する方法、およびメッセージ・キュー・アプリケーションを実装する方法についての関連情報が掲載されています。

- 『BEA Tuxedo のファイル形式とデータ記述方法』の `TMQUEUE`(5) および `TMQFORWARD`(5)

サポート情報

皆様の BEA Tuxedo マニュアルに対するフィードバックをお待ちしています。ご意見やご質問がありましたら、電子メールで docsupport-jp@bea.com までお送りください。お寄せいただきましたご意見は、BEA Tuxedo マニュアルの作成および改訂を担当する BEA 社のスタッフが直接検討いたします。

電子メール メッセージには、BEA Tuxedo 8.0 リリースのマニュアルを使用していることを明記してください。

BEA Tuxedo に関するご質問、または BEA Tuxedo のインストールや使用に際して問題が発生した場合は、www.bea.com の BEA WebSUPPORT を通して BEA カスタマ・サポートにお問い合わせください。カスタマ・サポートへの問い合わせ方法は、製品パッケージに同梱されている カスタマ・サポート・カードにも記載されています。

カスタマ・サポートへお問い合わせの際には、以下の情報をご用意ください。

- お客様のお名前、電子メール・アドレス、電話番号、Fax 番号
- お客様の会社名と会社の住所
- ご使用のマシンの機種と認証コード
- ご使用の製品名とバージョン
- 問題の説明と関連するエラー・メッセージの内容

表記上の規則

このマニュアルでは、以下の表記規則が使用されています。

規則	項目
太字	用語集に定義されている用語を示します。
Ctrl + Tab	2 つ以上のキーを同時に押す操作を示します。
イタリック体	強調またはマニュアルのタイトルを示します。

等幅テキスト	コード・サンプル、コマンドとオプション、データ構造とメンバ、データ型、ディレクトリ、およびファイル名と拡張子を示します。また、キーボードから入力する文字も示します。
	例 :
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

規則	項目
等幅太字	コード内の重要な単語を示します。 例 : `void **commit** ( )`
等幅イタリック体	コード内の変数を示します。 例 : `String *expr*`
大文字	デバイス名、環境変数、および論理演算子を示します。 例 : LPT1 SIGNON OR
{ }	構文の行で選択肢を示します。かっこは入力しません。

[ ]	構文の行で省略可能な項目を示します。かっこは入力しません。
	例 :
	`buildobjclient [-v] [-o name ] [-f *file-list*]...`
	`[-l *file-list*]...`

|	構文の行で、相互に排他的な選択肢を分離します。記号は入力しません。

...	コマンド行で次のいずれかを意味します。
	■ コマンド行で同じ引数を繰り返し指定できること
	■ 省略可能な引数が文で省略されていること
	■ 追加のパラメータ、値、その他の情報を入力できること
	省略符号は入力しません。
	例 :
	`buildobjclient [-v] [-o name ] [-f *file-list*]...`
	`[-l *file-list*]...`

. . .	コード例または構文の行で、項目が省略されていることを示します。省略符号は入力しません。

章

1 BEA Tuxedo /Q の概要

ここでは、次の内容について説明します。

- はじめに
- キュー・システムのコンポーネントとタスク
- 管理者のタスク
- プログラマのタスク

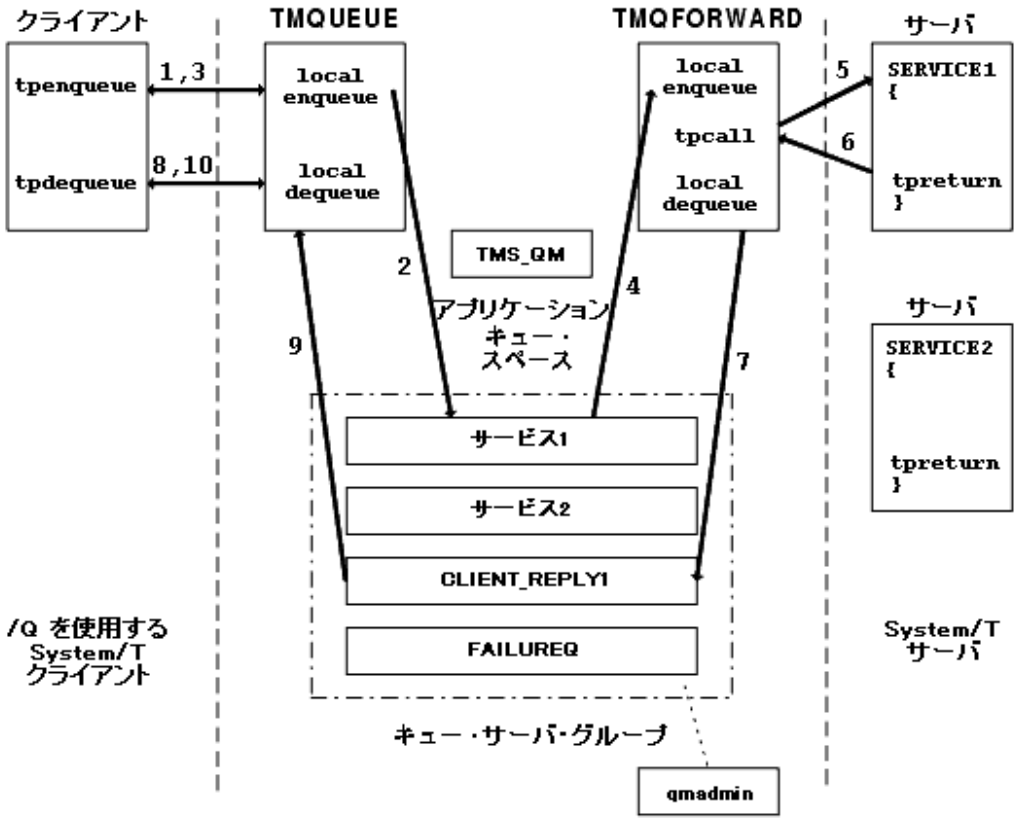
はじめに

BEA Tuxedo/Q コンポーネントを使用すると、メッセージを永続的な記憶域 (ディスク) や一時的な記憶域 (メモリ) のキューに登録でき、後で取り出して処理できます。BEA Tuxedo アプリケーション・トランザクション・モニタ・インターフェイス (ATMI) を使用して、キューへのメッセージの登録、またはキューからのメッセージの取り出しを行う関数を実行できます。応答メッセージおよびエラー・メッセージは、キューに登録されて、後でクライアント・プログラムに返されます。管理コマンドのインタプリタを使用すると、キューの作成、表示、および変更を行うことができます。また、サーバを使用して、メッセージのキューへの登録、キューからの取り出し、キューからの転送、およびキューに関係するトランザクション管理の要求を受け取ることができます。

キュー・システムのコンポーネントとタスク

次の図は、キュー・システムのコンポーネントを示しています。

図 1-1 キュー・サービスの呼び出し



この図は、キュー・サービスの呼び出しで、キュー・システムの各コンポーネントがどのように動作するかを示しています。ここでは、この図を使用して、管理者とプログラマが BEA Tuxedo /Q コンポーネントを定義したり、メッセージをキューに入れて処理したり、応答を受け取る方法について説明します。図に示されているコンポーネントのサブセットを使用すると、単純なピア・ツー・ピア通信でキュー・サービスを使用できます。

キュー・スペースは、1 つのリソースです。このリソースには、X/Open の XA 準拠のリソース・マネージャ・インターフェイスからアクセスできます。このインターフェイスは、別の XA 準拠のリソース・マネージャと連携して、キューへの登録およびキューからの取り出しが 2 フェーズ・コミット・トランザクションの一部として行われるために必要です。

管理者のタスク

BEA Tuxedo の管理者は、サーバを定義し、1-2 ページの「キュー・サービスの呼び出し」で縦の 2 本の破線の間に示されているキュー・スペースおよびキューを作成する必要があります。

キュー・サーバ・グループは、最低 1 つ定義します。その場合、`TMS_QM` をトランザクション・マネージャ・サーバにします。

このほかに、システムで提供される 2 つのサーバをコンフィギュレーション・ファイルに定義する必要があります。この 2 つのサーバは、次の処理を行います。

- メッセージ・キュー・サーバ `TMQUEUE(5)` は、メッセージをキューに登録したり、キューから取り出したりします。このサーバは、クライアントとサーバがキューに対してローカルであるかどうかにかかわらず、クライアントとサーバの代わりにメッセージに関する操作を行います。
- メッセージ転送サーバ `TMQFORWARD(5)` は、メッセージをキューから取り出して、アプリケーション・サーバに転送します。BEA Tuxedo システムでは、サーバ用の `main()` が用意されています。この関数は、サーバの初期化と終了、サービス・ルーチンへの着信要求の受信とディスパッチを行うためのバッファの割り当て、応答の正しい宛先へのルーティングを行います。このすべての処理は、アプリケーションに透過的です。既

存のサーバは、自分のメッセージをキューから取り出したり、応答をキューに登録することは行いません。BEA Tuxedo /Q の目的の 1 つは、そのような既存のサーバを変更せずに、キューに登録されたメッセージに対して既存のサーバを利用できるようにすることです。TMQFORWARD サーバは、キュー・スペース内の 1 つ以上のキューからメッセージを取り出し、そのメッセージをキューと同じ名前のサービスを持つサーバに送信し、応答を待ちます。そして、発信元によって応答キューまたは異常終了キューが指定されている場合はメッセージの発信元の指定に従って 、対応する応答キューまたは異常終了キューに、正常終了応答または異常終了応答を登録します。

また、管理者は、キュー管理プログラム qmadmin(1) または APPQ_MIB(5) 管理情報ベース (MIB) を使用して、キュー・スペースを作成する必要があります。キュー・スペースには、キューが格納されます。たとえば、1-2 ページの「キュー・サービスの呼び出し」では、APP キュー・スペース内に 4 つのキューが存在しています。各キュー・スペースはリソース・マネージャ・インスタンスであり、1 つのグループに単一のリソース・マネージャ (RM) だけが存在できるため、キュー・スペースとキュー・サーバ・グループは 1 対 1 のマッピングです。

キュー・スペースの概念では、次に示す方法で、キュー間でオーバヘッドを共有することにより、キューに対応する管理オーバヘッドを減少させることができます。

- キュー・スペース内のキューは、メッセージ用の永続的な記憶域および非永続的な記憶域を共有します。
- 1-2 ページの「キュー・サービスの呼び出し」の単一のメッセージ・キュー・サーバ TMQUEUE は、単一のキュー・スペース内で、複数のキューに対してメッセージの登録と取り出しを行います。
- 1-2 ページの「キュー・サービスの呼び出し」の単一のメッセージ転送サーバ TMQFORWARD は、単一のキュー・スペース内で、複数のキューに対してメッセージの取り出しと転送を行います。
- 1-2 ページの「キュー・サービスの呼び出し」のトランザクション・マネージャ・サーバ TMS_QM の 2 つのインスタンスは、単一のキュー・スペース内で、複数のキューに対してトランザクションを実行します。トランザクション・マネージャ・サーバの 1 つのインスタンスは、非ブロッキング・トランザクション用に予約されています。そのため、高速

で処理され、ブロッキング・トランザクションによって遅延されません。ブロッキング・トランザクションはトランザクション・マネージャ・サーバの 2 番目のインスタンスで処理されます。

管理者は、キュー・スペースに対して、アプリケーション・コンフィギュレーション内に単一のサーバ・グループを定義できます。その場合、UBBCONFIG にグループを指定するか、または tmconfig(1) を使用して (tmconfig、wtmconfig(1) を参照)、グループを動的に追加します。

- 最後に、管理者が 1 つのキュー・スペース内にあるキュー間でメッセージを移動させると、メッセージが異なる安定記憶領域にある場合よりもオーバヘッドが少なくなります。これは、1 フェーズ・コミットを行うことができるからです。

キューを定義する作業の 1 つに、キューのメッセージの順序を指定することがあります。キューの順序付けは、メッセージの可用期間、有効期限、優先順位、FIFO、LIFO のいずれか、またはこれらの組み合わせで決定します。

管理者は、この中から 1 つ以上の基準を指定します。その場合、最上位の基準から先に指定します。FIFO および LIFO の値は、最後に指定します。メッセージは、指定された順序付け基準に従ってキューに登録され、キューの先頭から取り出されます。管理者は、必要な数だけメッセージ・キュー・サーバを設定して、クライアントが安定キューに生成する要求に対応できるようにします。

データ依存型ルーティングは、同じサービスを提供するサーバが存在する複数のサーバ・グループ間でルーティングを行うために使用します。

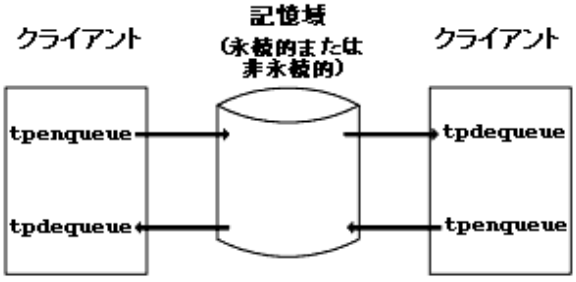
管理上の目的から、定期的に取り出されないキューがしきい値に達したときに実行されるコマンドを設定できます。これは、キューに使用されるキュー・スペースのバイト数、ブロック数、または割合、あるいはキューのメッセージの数に基づいて設定します。このコマンドでは、TMQFORWARD サーバを起動してキューを取り出すか、または管理者にメールを送信して手動で処理します。

BEA Tuxedo システムでは、BEA Tuxedo インフラストラクチャのキューイング・サービス・コンポーネントを使用して、いくつかの操作が行われます。BEA Tuxedo インフラストラクチャは、セキュリティ、スケーラビリティ、キュー処理、トランザクションなどのサービスを提供します。たとえ

ば、共用メモリの管理操作は、Queuing Services コンポーネントで提供されます。BEA Tuxedo アプリケーションで現在使用できない関数もあります。それらについては、その関数の説明で注記してあります。

キュー機能をクライアント間のピア・ツー・ピア通信で使用すると、クライアントは転送サーバを使用せずに、ほかのクライアントと通信できます。次の図は、ピア・ツー・ピア通信モデルを示しています。

図 1-2 ピア・ツー・ピア通信



プログラマのタスク

1-2 ページの「キュー・サービスの呼び出し」のステップ 1 ～ 3 で、tpenqueue(3c) を使用して、クライアントはアプリケーション・キュー・スペース内の SERVICE1 キューにメッセージを登録します。オプションとして、tpenqueue() の呼び出しに、応答キューおよび異常終了キューの名前を含めることができます。この例では、これらはキュー CLIENT_REPLY1 および FAILURE_Q になっています。クライアントは、メッセージに付ける相関識別子の値を指定することもできます。この値は、キュー間で同じです。そのため、キューのメッセージに対応付けられたどの応答メッセージまたは異常終了メッセージも、応答キューまたは異常終了キューから読み取られるときに識別できます。

クライアントは、デフォルトのキューの順序付け (たとえば、メッセージがキューから取り出せるようになるまでの時間) を使用するか、デフォルトのキューの順序付けを無効にすること (たとえば、メッセージをキューの先頭

に置くこと、またはキュー上の別のメッセージの前に置くことを要求すること) ができます。tpenqueue() は、TMQUEUE サーバにメッセージを送信し、そのメッセージはキューに登録され、承認 (ステップ 3) がクライアントに送信されます。肯定応答は、クライアントが直接確認することはできません。ただし、クライアントが正常な戻り値を取得すると、肯定応答が送られたと見なすことができます。異常終了の戻り値には、異常終了の内容についての情報が含まれています。

キュー・マネージャによって割り当てられたメッセージ識別子は、アプリケーションに返されます。この識別子は、特定のメッセージをキューから取り出す場合に使用します。また、この識別子は、別の tpenqueue() 内で使用して、キューに次に登録されるメッセージの前にあるメッセージを識別することができます。

キューに登録されたメッセージをキューから取り出すには、そのメッセージをキューに登録したトランザクションが正常にコミットされている必要があります。

BEA Tuxedo /Q をキュー・サービスの呼び出しに使用している場合に、メッセージがキューの先頭に到達すると、TMQFORWARD サーバはそのメッセージをキューから取り出し、tpcall(3c) を介してキューと同じ名前を持つサービスに転送します。1-2 ページの「キュー・サービスの呼び出し」では、このキューとサービスは SERVICE1 という名前で、ステップ 4、5、および 6 がそれを示しています。クライアントの識別子、およびアプリケーションの認証キーは、メッセージをキューに登録したクライアントに設定されます。この 2 つは、キューから取り出されたメッセージがサービスに送信されるときに、そのメッセージに付加されます。

サービスが応答を返すと、TMQFORWARD はその応答 (オプションとしてユーザ戻りコードを伴う) を応答キューに登録します (1-2 ページの「キュー・サービスの呼び出し」のステップ 7 を参照)。

その後 (1-2 ページの「キュー・サービスの呼び出し」のステップ 8、9、および 10)、クライアントは tpdequeue(3c) を使用して、応答キュー CLIENT_REPLY1 から読み込んで応答メッセージを取得します。

tpdequeue() に TPQPEEK フラグを設定すると、メッセージをキューから削除せずに取り出すことができます。有効期限が過ぎたメッセージ、または管理者によって削除されたメッセージは、直ちにキューから削除されます。

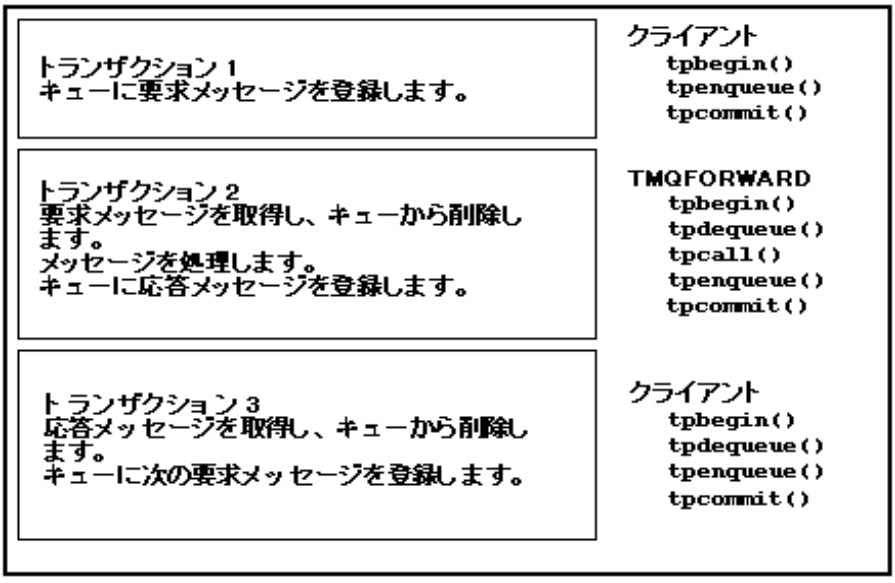
トランザクション管理

トランザクション管理の目的の 1 つは、グローバル・トランザクション内でメッセージをキューに登録したりキューから取り出して、信頼性を確実にすることです。ただし、これとは相反する目的として、関与するトランザクションの数を最小限にして実行のオーバヘッドを減少させることがあります。

呼び出し元は、キュー処理を呼び出し元のトランザクションから分離して、自身が関与しているあらゆるトランザクションとは別のトランザクションで、メッセージをキューに登録することもできます。ただし、この場合のタイムアウトは、メッセージがキューに登録されているかどうかにかかわらず、認識されません。

より良い方法としては、次の図に示すように、呼び出し元のトランザクション内でメッセージをキューに登録します。

図 1-3 トランザクション境界の判定



この図では、クライアントがトランザクションを開始し、メッセージをキューに登録し、トランザクションをコミットしています。メッセージは、TMQFORWARD で開始される 2 番目のトランザクション内でキューから取り出されます。サービスは tpcall(3c) で呼び出されて実行され、応答が同じトランザクション内でキューに登録されます。クライアントは 3 番目のトランザクションを開始し、応答をキューから取り出します。また、別の要求メッセージをキューに登録する場合もあります。次の要求のキューへの登録は、前の要求からの応答をキューから取り出す処理と同じトランザクション内で行うことができます。そのため、実行中の処理では、3 番目と最初のトランザクションを 1 つにまとめることができます。

注記　システムでは、同じトランザクション内で、あるメッセージからの応答をキューから取り出し、次の要求をキューに入れることが可能です。ただし、同じトランザクション内で、要求をキューに入れ、応答をキューから取り出すことはできません。メッセージをキューから取り出すには、要求をキューに登録するトランザクションが正常にコミットされることが必要です。

応答メッセージの処理

応答メッセージは、tpenqueue() の呼び出し時に、アプリケーションによって指定することもしないことも可能です。その場合、それぞれ次のような結果になります。

- キュー・メッセージに応答キューが指定されていない場合、メッセージの処理以外の作業は要求されません。
- 応答キューを指定するメッセージがキューから取り出された場合、メッセージの発信者は、要求の実行が正常に終了したときに応答がキューに登録されることを要求しています。
  - アプリケーションが tpdequeue() を使用してメッセージを明示的にキューから取り出す場合、そのアプリケーションは tpenqueue() を呼び出して応答をキューに登録する必要があります。通常、これは要求メッセージがキューから取り出されて実行されるのと同じトランザクション内で行われます。そのため、この操作全体はアトミックに行われます。つまり、トランザクションが成功した場合にのみ、応答がキューに登録されます。

- TMQFORWARD によってメッセージが自動的に処理される場合 (キューから取り出され、tpcall() を介してアプリケーションに渡される場合)、アプリケーション・サービスが正常に終了したとき、つまり TPSUCCESS が設定されて tpreturn(3c) が呼び出され、tpcall() が 1 を返さなかったときは、TMQFORWARD は応答をキューに登録します。tpcall() がデータを受け取ると、使用された型付きバッファは応答キューに登録されます。tpcall() でデータが受信されなかった場合、データなしのメッセージ (つまり、NULL メッセージ) が、キューに登録されます。メッセージがキューに登録されることは、それが NULL であっても、操作の完了を示すには十分です。

エラー処理

エラーを処理するには、アプリケーションに起こるエラーの特徴を理解することと、BEA Tuxedo 管理者とアプリケーション・プログラム開発者の注意深い計画と協力が必要です。BEA Tuxedo /Q では、メッセージがトランザクション内でキューから取り出され、そのトランザクションがロールバックされたときに、再試行パラメータが 0 より大きい場合は、メッセージがキューに戻されて、再度取り出しと実行が可能になります。

一時的な障害の場合は、少し時間をおいてから、メッセージの取り出しと実行を再試行します。たとえば、アプリケーション・データベースに対するアクティビティが多数ある場合は、別のトランザクションによってデータベースのロックが解除されるには、少しの時間さえあればよい場合があります。通常、再試行の回数に上限を設定することも、なんらかのアプリケーション不備によってリソースを著しく浪費することを防ぐには有用です。管理者がキューを設定するときに、再試行の回数と遅延時間 (秒単位) の両方を指定できます。再試行の回数 0 は、再試行が行われないことを示します。再試行の回数が上限に達すると、管理者がキュー・スペースに設定するエラー・キューにメッセージが移動されます。

障害が一時的なものではない場合もあります。たとえば、キュー・メッセージが存在しないアカウントに対して操作を要求する場合などです。その場合、再試行してリソースを浪費しないようにします。アプリケーション・プログラマまたは管理者が、特定の操作の失敗が一時的ではないと判断した場合、単に再試行の回数を 0 に設定します。同じサービスに対する障害が、一

時的なものであったり、永続的なものであったりする場合がよくあります。管理者およびアプリケーション開発者は、いろいろな方法でエラーに対処できるようにしておきます。

アプリケーションは、直接メッセージをキューから取り出すことも、TMQFORWARD サーバを使用することもできます。また、トランザクションをコミットするようにロジックで指示されている場合に、トランザクションがロールバックされたり、メッセージがキューに再度登録されたりするエラーが発生することがあります。そのため、多様な問題が発生します。このような問題とその対処法については、2-1 ページの「BEA Tuxedo /Q の管理」、3-1 ページの「BEA Tuxedo /Q の C 言語プログラミング」、および 4-1 ページの「BEA Tuxedo/Q COBOL 言語プログラミング」を参照してください。

まとめ

BEA Tuxedo /Q によって BEA Tuxedo アプリケーション・プログラマおよび管理者に提供される機能は、次のようにまとめることができます。

- アプリケーション・プログラミング・インターフェイスを使用して、要求をキューに登録して後で処理することができます。システムでは、要求が 1 回だけ正常に実行されることが保証されます。デフォルトでは、失敗した場合はメッセージがキューに戻されます。また、アプリケーション・プログラミング・インターフェイスを使用して、キューの先頭から、またはメッセージ識別子や相関識別子によって、メッセージを取り出すことができます。
- アプリケーション・プログラムや管理者は、キューのメッセージの順序付けを制御できます。その場合、基準を設定して順序付けします。基準としては、メッセージの可用期間、有効期限、優先順位、LIFO、FIFO のいずれか、またはこれらの組み合わせを使用できます。アプリケーションは、順序付けを無効化して、キューの先頭や、既にキューに登録されている特定のメッセージの前にメッセージを入れることができます。
- BEA Tuxedo サーバを使用して、クライアントおよびサーバ (リモート・クライアントやサーバの場合もあります) に代わって、メッセージをキューに登録したりキューから取り出すことができます。サーバのコピーをいくつ設定するかは、管理者が決定します。

- BEA Tuxedo サーバを使用して、キューからメッセージを取り出して、サービスに転送して実行できます。このサーバを使用すると、既存のサーバを変更せずにキューに登録された要求を処理できます。各転送サーバは、1 つ以上のキューを処理するように設定できます。トランザクションでは、1 回だけ処理が行われることが保証されます。管理者は、設定する転送サーバの数を決定します。
- 管理者は、処理のためにキューに格納されたメッセージを管理できます。たとえば、失敗時に要求を再試行する回数、再試行を行う間隔、キュー上のメッセージの再順序付け、キューの容量の管理などを行います。

キューに登録されたメッセージを使用するアプリケーション・パラダイムは多数あります。このようなパラダイムでは、マシン、サーバ、またはリソースが使用できない場合、または信頼性が低い場合 (広域ネットワークやワイヤレス・ネットワークの場合など) に、要求をキューに登録できます。また、ワーク・フローで、プロセスの次のステップを行うために各ステップでキュー要求を生成できます。さらに、処理に時間がかかるトランザクションのバッチ処理で、発信元が完了を待たなくても、最終的にはメッセージが処理されたことを保証できます。ピア・ツー・ピア関係で、関係がない 2 つのアプリケーション間で、データ・パイプを提供することもできます。

2 BEA Tuxedo /Q の管理

ここでは、次の内容について説明します。

- はじめに
- コンフィギュレーション
- キュー・スペースとキューの作成
- 暗号化メッセージ・バッファの処理
- BEA Tuxedo /Q 機能の保守
- Windows 標準入出力

はじめに

BEA Tuxedo /Q の管理者は、次の 3 つのタスクを行います。

- リソースの設定
- キュー・スペースおよびキューの作成
- 機能の監視および管理

コンフィギュレーションやキューの属性には、アプリケーションの要件が反映されていなければなりません。そのため、アプリケーション開発者とプログラマがよく協力することが大切です。

サンプル・プログラム qsample

キュー機能の使い方を簡単に示すサンプルがソフトウェアに提供されています。詳細については、A-1 ページの「サンプル・アプリケーション」を参照してください。

コンフィギュレーション

BEA Tuxedo /Q コンポーネントには、3 つのサーバが提供されています。1 つは、`TMS_QM` という BEA Tuxedo /Q リソース・マネージャ用のトランザクション・マネージャ・サーバ (TMS) です。このサーバは、キュー機能でグローバル・トランザクションを管理します。このサーバは、コンフィギュレーション・ファイルの `GROUPS` セクションで定義されていることが必要です。

ほかの 2 つは、`TMQUEUE(5)` および `TMQFORWARD(5)` で、ユーザにサービスを提供します。この 2 つのサーバは、コンフィギュレーション・ファイルの `SERVERS` セクションで定義されている必要があります。

`TMQFORWARD` の機能がアプリケーションのニーズを完全には満たさない場合、アプリケーション独自のキュー・サーバを作成することもできます。たとえば、エラー・キューに移動したメッセージをキューから取り出す特別なサーバが必要な場合があります。

アプリケーションでは、ピア・ツー・ピア通信を選択することもできます。この通信では、転送サーバを使用せずに、アプリケーションはほかのアプリケーションと、クライアントはほかのクライアントと通信します。

QM サーバ・グループの指定

標準要件であるグループ名のタグおよび GRPNO の値 (詳細については UBBCONFIG(5) を参照) のほかに、アプリケーションが使用するキュー・スペースごとに 1 つのサーバ・グループを定義する必要があります。TMSNAME および OPENINFO パラメータを設定する必要があります。次は、この 2 つのパラメータの例です。

```
TMSNAME=TMS_QM
```

および

```
OPENINFO="TUXEDO/QM:<device_name:<queue_space_name>"
```

TMS_QM は、BEA Tuxedo /Q のトランザクション・マネージャ・サーバの名前です。OPENINFO パラメータの TUXEDO/QM は、$TUXDIR/udataobj/RM にあるリソース・マネージャのリテラル名です。<*device_name*> および <*queue_space_name*> の値は、状況に応じて決定されます。汎用デバイス・リストのパス名、およびキュー・スペースに対応付けられた名前をそれぞれ設定します。これらの値は、qmadmin(1) を使用して、BEA Tuxedo の管理者が指定します。

注記　これらの値は、時系列順に指定されている必要はありません。コンフィギュレーション・ファイルは、キュー・スペースを定義する前でも定義した後でも作成できます。コンフィギュレーションが定義され、以前に作成されたキュー・スペースとキューが使用できることが重要です。

GROUPS セクションのエントリごとに、キュー・スペースを 1 つだけ使用できます。CLOSEINFO パラメータは使用されません。

次は、TMQUEUE(5) のリファレンス・ページから引用した例です。

```
*GROUPS
TMQUEUEGRP1 GRPNO=1 TMSNAME=TMS_QM
        OPENINFO="TUXEDO/QM:/dev/device1:myqueuespace"
TMQUEUEGRP2 GRPNO=2 TMSNAME=TMS_QM
        OPENINFO="TUXEDO/QM:/dev/device2:myqueuespace"
```

メッセージ・キュー・サーバの指定

TMQUEUE(5) リファレンス・ページでは、コンフィギュレーション・ファイルの SERVERS セクションについて詳しく説明しています。ここでは、特に大切な内容について説明します。

操作のタイムアウト

TMQUEUE でタイムアウトが認識されるのは、CLOPT パラメータで 2 つのダッシュ (--) の後に -t timeout オプションが指定されている場合です。このタイムアウト値は、サーバでトランザクションが有効ではないことが検出された場合に、サーバ内で開始された操作に適用されます。つまり、クライアントが tpbegin(3c) を呼び出さずに tpenqueue(3c) または tpdequeue(3c) を呼び出した場合、トランザクションは有効になっていません。または、クライアントがトランザクションを開始し、TPNOTRAN フラグを設定してクライアントのトランザクションでキューに要求を登録しない場合に、tpenqueue() または tpdequeue() を呼び出したときも、トランザクションは有効になっていません。*timeout* のデフォルト値は 30 秒です。tpdequeue 要求の応答で *flags* に TPQWAIT が設定されている場合、待ち時間が -t *timeout* に指定された時間 (秒単位) を超えると、TPETIME エラーが返されます。

注記　ctl は、TPQCTL 型の構造体で、tpenqueue(3c) および tpdequeue(3c) で使用されて、呼び出し元プロセスとシステム間でパラメータの受け渡しが行われます。TPQWAIT は、tpdequeue のフラグ設定の 1 つで、プロセスが応答メッセージを待機することを示します。この構造体の詳細については、3-6 ページの「TPQCTL 構造体」、および 4-7 ページの「TPQUEDEF-REC 構造体」を参照してください。COBOL でこれに相当するものは、TPQUEDEF-REC レコードです。

キュー・スペース名、キュー名、およびサービス名

キュー・スペース名、キュー名、およびサービス名は、混同しやすいので注意してください。まず、メッセージ・キュー・サーバ TMQUEUE の指定では注意が必要です。このサーバをコンフィギュレーション・ファイルで指定する場合は、CLOPT パラメータの -s フラグを使用して、サーバのあるインスタン

スからサービスを受け取るキュー・スペースの名前を指定できます。これは、関数 TMQUEUE で宣言されたサービスと同じです。キュー・スペースを 1 つだけ使用するアプリケーションでは、CLOPT -s オプションを指定する必要はありません。デフォルトの -s TMQUEUE:TMQUEUE が使用されます。アプリケーションに複数のキュー・スペースが必要な場合、キュー・サーバの SERVERS セクションの -s オプションの引数として、キュー・スペースの名前を指定します。

この指定を行う別の方法としては、buildserver(1) を使用してメッセージ・キュー・サーバを再度ビルドし、同じように -s オプションを使用してキュー・スペースの名前を指定することができます。この方法では、実行可能サーバにサービス名が固定、つまり「ハード・コーディング」されます。

```
# 次の 2 つの指定は同じです。

*SERVERS
TMQUEUE SRVGRP="TMQUEUEGRP1" SRVID=1000 RESTART=Y GRACE=0 \
        CLOPT="-s myqueuespace:TMQUEUE"
and

buildserver -o TMQUEUE -s myqueuespace:TMQUEUE -r TUXEDO/QM \
        -f ${TUXDIR}/lib/TMQUEUE.o
followed by
 ..
 ..
 ..
TMQUEUE SRVGRP="TMQUEUEGRP1" SRVID=1000 RESTART=Y GRACE=0 \
        CLOPT="-A"
```

データ依存型ルーティング

前の節で、メッセージ・キュー・サーバ内のサービス (キュー・スペース名) の指定について説明しました。この機能を使用すると、キューのデータ依存型ルーティングを行うことができます。データ依存型ルーティングでは、メッセージがキューに登録されて、メッセージ・バッファのフィールド値に応じて特定のグループ内のサービスによって処理されます。その場合、同じキュー・スペース名を 2 つの異なるグループに指定します。そして、ルーティングをコンフィギュレーション・ファイルで設定して、キューにあるメッセージの処理を行うグループを指定します。次は、TMQUEUE(5) のリファレンス・ページから引用した例です。キュー・スペース名は変えてあります。

```
*GROUPS
TMQUEUEGRP1 GRPNO=1 TMSNAME=TMS_QM
        OPENINFO="TUXEDO/QM:/dev/device1:myqueuespace"
TMQUEUEGRP2 GRPNO=2 TMSNAME=TMS_QM
        OPENINFO="TUXEDO/QM:/dev/device2:myqueuespace"
*SERVERS
TMQUEUE SRVGRP="TMQUEUEGRP1" SRVID=1000 RESTART=Y GRACE=0 \
        CLOPT="-s ACCOUNTING:TMQUEUE"
TMQUEUE SRVGRP="TMQUEUEGRP2" SRVID=1000 RESTART=Y GRACE=0 \
        CLOPT="-s ACCOUNTING:TMQUEUE"
*SERVICES
ACCOUNTING  ROUTING="MYROUTING"
*ROUTING
MYROUTING  FIELD=ACCOUNT BUFTYPE="FML" \
        RANGES="MIN-60000:TMQUEUEGRP1,60001-MAX:TMQUEUEGRP2"
```

バッファ・タイプのカスタマイズ

TMQUEUE では、ATMI のすべての標準バッファ・タイプがサポートされています。アプリケーションでほかのバッファ・タイプを追加する必要がある場合は、$TUXDIR/tuxedo/tuxlib/types/tmsypesw.c をコピーし、独自のバッファ・タイプのエントリを追加し、最終行の NULL が残っていることを確認し、変更したファイルを buildserver(1) コマンドへの入力として使用します。buildserver コマンドの使用例については、TMQUEUE(5) のリファレンス・ページを参照してください。

追加したサーバ名は、CLOPT パラメータ (前述を参照) に指定せずに、buildserver コマンドの -s オプションを使用して TMQUEUE に対応付けることができます。

バッファ・サブタイプ

バッファのサブタイプは tpalloc(3c) サブタイプ・パラメータを使用して割り当てることができ、tptypes(3c) 関数を使用して後でそれを取得できます。この機能により、ユーザ定義の ATMI バッファ・タイプを新しく作成せずに、データに型を割り当てることができます。これは、文字配列 (CARRAY) または文字列 (STRING) を含むバッファで特に有用です。

メッセージ転送サーバの指定

TMQFORWARD(5) は、BEA Tuxedo /Q で提供される 3 番目のサーバです。このサーバは、指定されたキューからメッセージを取り出し、tpcall(3c) を介してそのメッセージを BEA Tuxedo サーバに渡し、対応する応答メッセージを処理します。コンフィギュレーション・ファイルでサーバを定義する方法については、TMQFORWARD(5) のリファレンス・ページを参照してください。ここでは、特に大切な内容について説明します。

TMQFORWARD は 1 つのサーバであり、アプリケーションで使用される各インスタンスは、コンフィギュレーション・ファイルの SERVERS セクションで定義されていなければなりません。ただし、TMQFORWARD には、通常のサーバとは異なる次のような特徴があります。次に例を示します。

- TMQFORWARD にはサービスを指定しません。
- 通常の要求 / 応答関係で行われるように、TMQFORWARD にクライアント・プロセスがメッセージをポストすることはできません。
- TMQFORWARD は、MSSQ セットのメンバとして定義できません。
- TMQFORWARD は、応答キューを持ちません。

TMQFORWARD のインスタンスは、キューに対応するサーバ・グループを使用して、そのキュー・スペースに結び付けられます。特に、そのグループの OPENINFO 文の第 3 フィールドを使用して、キュー・スペースと結び付けられます。以下に、これ以外の主なパラメータ、特に 2 つのダッシュに続く CLOPT パラメータについて説明します。

キュー名およびサービス名 (-q オプション)

-q *queuename*,*queuename*... は必須パラメータです。このパラメータは、サーバのこのインスタンスが確認する 1 つ以上のキューを指定します。*queuename* は、最大 15 文字までの NULL で終了する文字列です。これは TMQFORWARD によって一度キューから取り出された待機メッセージを処理するアプリケーション・サービス名と同じです。また、この名前は、メッセー

ジ・キュー・サーバ TMQUEUE を呼び出す準備をするときに、プログラマが tpenqueue(3c) または tpdequeue(3c) の 2 番目の引数として指定する名前でもあります。

トランザクション・タイムアウトの指定 (-t オプション)

TMQFORWARD は、TMQFORWARD が開始して終了するトランザクション内で機能します。-t *trantime* オプションは、トランザクションがタイムアウトになるまでの時間 (秒単位) を指定します。TMQFORWARD がキュー上にメッセージがあることを確認すると、トランザクションが開始されます。応答が応答キューまたは異常終了キューのいずれかに登録されると、そのトランザクションがコミットされます。そのため、トランザクションには、メッセージを処理するサービスの呼び出しと応答の受信が含まれています。デフォルト値は 60 秒です。

アイドル時間の指定 (-i オプション)

TMQFORWARD は一度呼び出されると、割り当てられているキューを定期的に確認します。キューが空であった場合は、-i *idletime* 秒間停止してから再度確認を行います。値が指定されていない場合 デフォルト値の 30 秒が使用されます。この値を 0 に設定すると、キューが継続的に確認されます。これは、キューが空の場合が多いときは、CPU リソースを浪費することになります。

サーバ終了の指定 (-e オプション)

-e オプションが指定されていると、キューが空の場合にサーバが正常にシャットダウンし、ユーザ・ログ・メッセージが生成されます。この機能は、キューに指定するしきい値コマンドに対して使用します。-e オプションとしきい値コマンドの詳細については、2-10 ページの「キュー・スペースとキューの作成」を参照してください。

サービスの異常終了後のメッセージの削除 (-d オプション)

サービス要求が TMQFORWARD から呼び出された後で異常終了した場合、トランザクションはロールバックされ、後で再試行できるようにメッセージは (キューに指定された再試行の制限回数まで) キューに戻されます。-d オプションを使用すると、次の処理も行われます。つまり、異常終了したサービスで NULL 以外の応答が返された場合、異常終了キューがメッセージに対応付けられ、キューが存在するときは、応答 (および、それに対応付けられた *tpurcode*) が異常終了キューに登録されて元の要求メッセージは削除されます。また、-d オプションを使用すると、キューに設定されてた再試行回数の制限値に達すると同時に元の要求メッセージが削除され、元の要求メッセージはエラー・キューに登録されます。

このオプションで重要なことは、ただむやみに再試行するのではなく、発信元のライアントが異常終了メッセージを調べ、再試行が必要であるかどうかを判断するようにコーディングできることです。これにより、本質的な問題、たとえば、アカウントが存在していないめに、レコードが「見つからない」場合などが原因で失敗した場合に、それを処理する方法が提供されます。

バッファ・タイプのカスタマイズ

カスタマイズしたアプリケーション・バッファ・タイプをタイプ・スイッチに追加し、buildserver(1) コマンドを使用して TMQFORWARD に組み込むことができます。ただし、TMQFORWARD をカスタマイズした場合は、-s オプションでサービス名を指定することはできません。

動的なコンフィギュレーション

コンフィギュレーション・パラメータについて、UBBCONFIG パラメータの観点から説明してきました。ただし、GROUPS セクションおよび SERVERS セクション内の指定は、tmconfig(1) を使用して、実行中のアプリケーションの TUXCONFIG ファイルに追加することもできます。tmconfig、wtmconfig(1) を参照してください。グループおよびサーバは、一度定義したら再起動する必要があります。

キュー・スペースとキューの作成

ここでは、qmadmin(1) の 3 つのコマンドについて説明します。これらのコマンドは、BEA Tuxedo /Q コンポーネントのリソースを設定するために使用します。APPQ_MIB 管理情報ベース (MIB) を使用すると、BEA Tuxedo /Q をプログラムで管理できるようになります。MIB の詳細については、APPQ_MIB(5) のリファレンス・ページを参照してください。

qmadmin のコマンドの使用方法

qmadmin のほとんどの主なコマンドでは、位置パラメータが使用されています。コマンドの呼び出し時に、コマンド行に位置パラメータ (オプションの前にダッシュ (-) が付かないパラメータ) が指定されていない場合、必要な情報を入力するように qmadmin で要求されます。

汎用デバイス・リストのエントリの作成 (crdl)

汎用デバイス・リスト (UDL) は、BEA Tuxedo で制御される VTOC ファイルです。このファイルは、BEA Tuxedo を実行するマシンに物理的な記憶空間をマッピングします。UDL のエントリは、キューとキュー・スペースのメッセージが格納されるディスク領域を指定します。BEA Tuxedo はその領域への入出力を管理します。BEA Tuxedo の新規インストールの一部としてキュー機能がインストールされる場合、コンフィギュレーション・ファイルの初回のロード時に、tmloadcf(1) によって UDL が生成されます。

キュー・スペースを作成する前に、UDL にそのキュー・スペースのエントリを作成する必要があります。次は、コマンドの例です。

```
# まず、/Q 管理インターフェイス qmadmin を呼び出します。
# QMCONFIG 変数は、UDL が置かれる既存のデバイスを
# 指定します。
QMCONFIG=/dev/rawfs qmadmin
# 次に、デバイス・リストのエントリを作成します。
```

```
crdl /dev/rawfs 50 500
# 上記のコマンドは、ブロック 50 で開始される物理ページを 500 個確保します。
# UDL に以前にリストされたエントリがない場合、オフセット ( ブロック番号 ) として 0 を使用します。
```

既存の BEA Tuxedo UDL にエントリを追加する場合は、QMCONFIG 変数の値には TUXCONFIG で指定されたパス名と同じ値を指定する必要があります。qmadmin を一度呼び出したら、新しいエントリを作成する前に、lidl コマンドを実行してどの領域を使用できるのかを確認します。

キュー・スペースの作成 (qspacecreate)

キュー・スペースでは、IPC 資源が使用されます。そのため、キュー・スペースを定義する場合は、共用メモリ・セグメントとセマフォを割り当てることになります。前述のように、このコマンドを使用する簡単な方法はプロンプトを表示することです。また、APPQ_MIB(5) の T_APPQSPACE クラスを使用して、キュー・スペースを作成することもできます。プロンプトは以下の順で表示されます。

```
> qspacecreate
Queue space name: myqueuespace
IPC Key for queue space:230458
Size of queue space in disk pages:200
Number of queues in queue space:3
Number of concurrent transactions in queue space:3
Number of concurrent processes in queue space:3
Number of messages in queue space:12
Error queue name: errq
Initialize extents (y, n [default=n]):
Blocking factor [default=16]: 16
```

このプログラムでは、最後の 3 行を除くすべてのプロンプトに値を入力する必要があります。この例からわかるように、最後の 2 つのプロンプトではデフォルト値が使用されています。error queue name の名前はほとんどの場合に必要ですが、必須ではありません。ここで名前を指定した場合は、qcreate コマンドを使用してエラー・キューを作成する必要があります。error queue name の名前が指定されていないと、通常は (たとえば、再試行回数の上限値に達した場合などに) エラー・キューに送られるメッセージは、永続的に失われることに注意してください。

共用メモリの予約領域は、キュー・スペース内のすべてのキューの永続的ではないメッセージを格納するために使用されます。プログラムでは、この予約領域のサイズを指定するように要求されません。永続的ではない (メモリ・ベースの) メッセージが必要な場合は、qspacecreate コマンド行に -n オプションを入力して、メモリ領域のサイズを指定する必要があります。

IPC キーには、ほかの IPC 資源の要件と競合しない値を指定します。この値は 32,768 より大きく 262,143 未満にします。

キュー・スペースのサイズ、キューの数、および一度に登録できるメッセージの数は、アプリケーションのニーズによって決定されます。UDL エントリで指定されたページ数より大きいサイズを指定することはできません。また、これらのパラメータに関連して、キュー・スペース内の個々の queue capacity パラメータを検討する必要があります。これらのパラメータを使用すると、(a) キューに登録できるメッセージの上限値を設定し、(b) キューに登録されたメッセージの数がしきい値に達したときに実行するコマンドの名前を指定できます。キュー・スペースに同時に登録できるメッセージの数を小さくすると、キューのしきい値に達しなくなります。

並列トランザクションの数を計算するには、以下の各項目を 1 つのトランザクションとしてカウントします。

- このキュー・スペースを使用するグループの TMS_QM サーバ
- このキュー・スペースを使用するグループの TMQUEUE サーバまたは TMQFORWARD サーバ
- qmadmin

クライアント・プログラムが tpenqueue を呼び出す前にトランザクションを開始する場合、キュー・スペースに同時にアクセスするクライアント数だけカウント数を増やします。最悪のケースは、すべてのクライアントがキュー・スペースに同時にアクセスすることです。

同時実行プロセスの数としては、このキュー・スペースを使用するグループの TMQUEUE サーバ、TMQFORWARD サーバ、または TMS_QM サーバごとに 1、余裕として 1 をカウントします。

qspacecreate コマンドの使用時に、キュー・スペースを初期化するように設定できます。また、キュー・スペースを初めてオープンするときに、qopen コマンドで初期化するように設定することもできます。

キューの作成 (qcreate)

使用するキューは、qmadmin の qcreate コマンドで作成する必要があります。まず、qopen コマンドでキュー・スペースをオープンします。キュー・スペース名が指定されていない場合は、qopen で名前を入力するように求められます。APPQ_MIB(5) の T_APPQ クラスを使用して、キューを作成することもできます。

qcreate のプロンプトは、次の順で表示されます。

```
> qcreate
Queue name: service1
Queue order (priority, time, fifo, lifo): fifo
Out-of-ordering enqueuing (top, msgid, [default=none]): none
Retries [default=0]:2
Retry delay in seconds [default=0]: 30
High limit for queue capacity warning (b for bytes used, B for
blocks used,
% for percent used, m for messages [default=100%]):80%
Reset (low) limit for queue capacity warning [default=0%]:0%
Queue capacity command:
No default queue capacity command
Queue 'service1' created
```

これらすべてのプロンプト (queue name のプロンプト以外) では、入力を省略できます。queue name が指定されていない場合、プログラムで警告メッセージが表示され、再度プロンプトが表示されます。これ以外のパラメータについてはプログラムでデフォルト値が用意されており、デフォルト値の使用を通知するメッセージが表示されます。

デフォルトの配信オプションやメモリのしきい値のオプションを入力するようには求められません。デフォルトの配信オプションを使用すると、配信モードが指定されていないメッセージを永続 (ディスク・ベースの) ストレージに配信するか、または非永続 (メモリ・ベースの) ストレージに配信するかを指定できます。メモリのしきい値オプションでは、非永続メモリのしきい値に達したときに、コマンドを実行するための値を指定できます。これらのオプションを使用するには、qcreate コマンド行で -d と -n をそれぞれ指定する必要があります。

キューの順序の指定

メッセージは、このパラメータで指定された順序でキューに登録され、キューからの取り出しに順序付け条件が設定されている場合を除き、キューの先頭から取り出されます。priority、expiration、time がキューの順序付け条件として選択されている場合、メッセージは TPQCTL 構造体の値に従ってキューに挿入されます。順序付け条件を組み合わせて指定する場合は、最も重要な条件を最初に指定します。複数指定する場合は、カンマ (,) で区切ります。fifo または lifo (この 2 つは、相互に排他的) が指定されている場合は、この値を最後に指定します。パラメータが指定された順序によって、キューの順序付け条件が決定されます。たとえば、priority, fifo と指定されている場合、キューはメッセージの優先順位に従って取り出され、同じ優先順位のメッセージの場合は先入れ先出しで取り出されます。

順序を無視したキュー登録

管理者が順序を無視したキュー登録を有効にした場合、つまりプロンプトで top または msgid が選択されている場合、プログラマはキューの先頭、または msgid で識別されるメッセージの前にメッセージが入るように指定できます。その場合、tpenqueue 呼び出しの TPQCTL 構造体の値を使用してください。このオプションは、よく検討してから使用します。一度このオプションを設定すると、それを変更するにはキューを破棄して、再度作成する必要があります。

再試行パラメータの指定

待機メッセージ機能の通常の動作では、メッセージをキューから取り出すトランザクションがロールバックされると、メッセージがキューに戻されます。そのメッセージがキューの先頭に到達すると、再度キューから取り出されます。必要な再試行の回数、および次の再試行までの遅延時間を指定できます。キューから取り出されたメッセージが再試行のためにキューに戻されると、キューの順序付けは遅延時間 (秒単位) だけ中断されます。この間、メッセージをキューから取り出すことはできません。

再試行回数のデフォルト値は 0 です。これは、再試行が一度も行われないことを示します。再試行回数の上限値に達すると、メッセージがキュー・スペースのエラー・キューに移されます。その場合、エラー・キューが指定され、作成されていることが前提です。エラー・キューが存在しない場合は、メッセージは破棄されます。

遅延時間には、秒単位が使用されます。メッセージ・キューがすぐにいっぱいになり、キューに戻されたメッセージがすぐに先頭に達する場合、遅延要因を作成して CPU サイクルを節約することができます。再試行に関しては、使用するアプリケーションでのこれまでの経験に基づいて決定します。特定のキューに対応付けられたサービスに大量の競合がある場合、一時的な問題が多数発生します。このような問題に対処するには、再試行回数に大きな値を指定します。この値は、主観的なものです。次の再試行を行うまでの時間も同じです。ロールバックされたトランザクションで通知される問題が解消されないものである場合、再試行回数を 0 にしてメッセージを直ちにエラー・キューに移動します。これは、TMQFORWARD で -d オプションを指定する場合の動作とよく似ています。ただし、長さが 0 以外の異常終了メッセージは、キューから削除するために TMQFORWARD で自動的に受信される点が異なります。

キューの容量の上限

qcreate コマンドには、キューの管理を部分的に自動化する 3 つのパラメータがあります。これらのパラメータは、上下限のしきい値を設定します。しきい値は、バイト数、ブロック数、メッセージ数、またはキューの容量に対する割合 (パーセント) で表すことができます。これにより、しきい値の上限に達したときに実行されるコマンドを指定できます。実際には、コマンドはしきい値の上限に達したときに 1 度だけ実行され、しきい値の上限より先に下限に達しない限り、再度実行されることはありません。

次は、これらのパラメータの使い方の 2 つの例です。

```
High limit for queue capacity warning (b for bytes used, B for
blocks used, % for percent used, m for messages [default=100%]):80%
Reset (low) limit for queue capacity warning [default=0%]:10%
Queue capacity command:/usr/app/bin/mailme myqueuespace service1
```

この例は、しきい値の上限としてディスク・ベースのキュー容量の 80% を設定し、キューの 80% が使用されたときにコマンドが実行されるように指定しています。このコマンドは、しきい値に達したときにメール・メッセージを送信するように作成されたスクリプトです。(myqueuespace および service1 は、このコマンドに対する仮定的な引数です。キューがいっぱいになったことが一度通知されると、その状況に対応するための操作を行うことができます。警告メッセージを再度受け取るのは、キューのロードが容量の 10% 以下になり、もう一度 80% に上がった場合だけです。永続的ではな

い (メモリ・ベースの) キューの容量を管理するために、qcreate コマンドの -n オプションを使用して、しきい値を設定したり、コマンドが実行されるように設定することもできます。

注記　Windows マシンを使用している場合、qmadmin() セッションでコマンドを設定する方法の詳細については、2-22 ページの「Windows 標準入出力」を参照してください。

2 番目の例は、もう少し自動化されていて、TMQFORWARD サーバの -e オプションを利用しています。

```
High limit for queue capacity warning (b for bytes used, B for
blocks used, % for percent used, m for messages [default=100%]):90%
Reset (low) limit for queue capacity warning [default=0%]:0%
Queue capacity command:tmboot -i 1002
```

この例では、キューの予備の TMQFORWARD サーバとして SRVID=1002 が設定されていること、その CLOPT パラメータに -e オプションが指定されていることが前提となっています。また、サーバが起動していないこと、または起動している場合は、キューが空であることが検出されたためにサーバが自身をシャットダウンしたことも前提となっています。キューが容量の 90% に達したとき、tmboot コマンドが実行されて予備のサーバが起動します。-e オプションを使用すると、キューが空の場合にサーバが自身をシャットダウンします。この例では、既に起動しているサーバに対して不要な tmboot コマンドを起動しないために、しきい値の下限として 0% が設定されています。

この 3 つのオプションのデフォルト値は、100%、0%、およびコマンドなしです。

応答キューおよび異常終了キュー

キューの作成方法、およびキューの操作を行うためのパラメータの指定方法について説明してきました。これまで、メッセージが登録されるキューを作成する場合に、同じ名前のサービスでキューが処理されることが前提となっていました。キューは、ピア・ツー・ピア通信などほかの目的で使用される場合もあります。キュー作成時のパラメータは、キューの用途が異なっても同じものが使用されます。メッセージがサービス・キューに登録される際に使用される TPQCTL 構造体には、応答キューおよび異常終了キューの名前を指定するためのフィールドがあります。TMQFORWARD は、要求されたサービスに対して TMQFORWARD が呼び出した tpacall(3c) が正常終了したか異

常終了したかを検出します。そして、これらのキューが管理者によって作成されてものである場合は、その応答を設定に応じてキューに登録します。応答キューまたは異常終了キューが存在しない場合は、サービスからの正常終了または異常終了の応答メッセージは削除されます。その場合、要求の発信元であるクライアントには、キューに入れられた要求の結果に関しては、何も通知されません。サービスから応答メッセージがない場合でも、応答キューが存在するときは、TMQFORWARD によって長さ 0 のメッセージがそのキューに登録されて、要求の発信元であるクライアントに結果が通知されます。

応答キューまたは異常終了キューを作成する場合は、これらのキューからのメッセージの取り出しは、事前に登録した要求に関する情報を求めているクライアント・プロセスによって行われることに注意してください。このようなメッセージをキューから取り出す一般的な方法は、そのメッセージに対応付けられている msgid (メッセージ識別子) または corrid (相関識別子) を使用することです。これは、キューの先頭からメッセージを取り出す方法と対照的です。そのため、キューの順序付け条件は、あまり意味を持ちません。その場合、fifo を設定しておけば十分です。retries パラメータおよび retry delay パラメータは、応答キューの場合は意味がないので、デフォルト値を使用してください。queue capacity のしきい値とコマンドは、応答キューでも利用できます。これらを使用して管理者に警告を通知し、管理者が対応できるようにすると便利です。

エラー・キュー

エラー・キューは、システムのキューです。qspacecreate プロンプトの 1 つで、キュー・スペースのエラー・キューの名前を入力するように求められます。指定された名前のエラー・キューが実際に作成されている場合、そのエラー・キューに再試行回数の上限に達したサービス・キューのメッセージが移されます。エラー・キューのメッセージは、qmadmin のコマンドを使用して手動で処理することも、APPQ_MIB MIB を使用して自動化された方法で処理することもできます。このようなエラー・キューの管理方法は、管理者が決定します。エラー・キューには、queue capacity パラメータを使用することができます。ただし、qname を除くほかのすべての qcreate のパラメータは使用できません。

注記　エラー・キューとサービス異常終了キューには、同じキューを使用しないことをお勧めします。同じキューを使用すると、アプリケーションの管理が難しくなり、クライアントが管理者の領域にまでアクセスすることも起こり得ます。

暗号化メッセージ・バッファの処理

通常、TMQUEUE および TMQFORWARD で暗号化メッセージ・バッファが処理される場合、暗号解読は行われません。ただし、『BEA Tuxedo のセキュリティ機能』の 1-72 ページの「/Q との互換性および相互運用性」で説明したように、/Q コンポーネントでメッセージが登録されたバッファを解読することが必要な場合があります。

「/Q との互換性および相互運用性」で説明したように、トランザクションに関与しない tpdequeue() 操作では、呼び出しプロセスが有効な解読キーを持っていない場合、キューの暗号化されたメッセージが破壊されます。そのため、アプリケーションのプログラマは、暗号化されたメッセージを取り出すためにプロセスで tpdequeue() を呼び出す場合は、事前にそのプロセスの解読キーをオープンしておく必要があります。オープンしていない場合、メッセージは失われます。

解読キーをオープンする方法については、『BEA Tuxedo のセキュリティ機能』の 2-61 ページの「プラグインによる復号化キーの初期化」、および 3-51 ページの「暗号化されたメッセージを受信するためのコードの記述」を参照してください。

BEA Tuxedo /Q 機能の保守

ここでは、効率的にキュー・スペースを操作できるように、キューの管理者が行うべき作業について説明します。

キュー・スペースのエクステントの追加

キュー・スペースのディスク領域を増やす必要がある場合は、qmadmin(1) の qaddext コマンドを使用します。APPQ_MIB(5) の T_APPQSPACE クラスの TA_MAXPAGES 属性を使用して、エクステントを追加することもできます。qmadmin コマンドは、キュー・スペース名およびページ数を引数として取ります。ページは、QMCONFIG 変数に指定されたデバイスに対して、UDL で定義されているエクステントから割り当てられます。キュー・スペースは、アクティブでないことが条件です。感嘆符を使用すると、qmadmin 外のコマンドを実行できるので、対応付けられたサーバ・グループをシャットダウンできます。次に例を示します。

```
> !tmshutdown -g TMQUEUEGRP1
```

次のコードが続きます。

```
> qclose
> qaddext myqueue 100
```

キュー・スペースは、クローズしていなければなりません。オープンしているキュー・スペースにエクステントの追加を試みると、qmadmin によってそのキューがクローズします。現在キュー・スペースにある永続的ではないメッセージは、qaddext コマンドが発行されて正常終了すると、すべて失われます。

キュー・スペースのバックアップの作成および移動

キュー・スペースのバックアップを作成する場合は、UNIX コマンドの dd(1) を使用すると便利です。まず、対応付けられたサーバ・グループをシャットダウンします。次のようにコマンドを入力します。

```
tmshutdown -g TMQUEUEGRP1
dd if=<qspace_device_file> of=<output_device_filename>
```

そのほかのオプションについては、UNIX システムの dd リファレンス・ページを参照してください。

キュー・スペースをアーキテクチャが同じである別のマシンに移動する場合も、同じコマンドを使用できます。ただし、現在のデバイス名が移動先のマシンに存在しない場合は、qmadmin の chdl コマンドで開始して、新しいデバイス名を指定する必要があります。

dd コマンドを持たないサーバ・プラットフォームでも、同じようなアーカイブ方法を使用できます。まず、バックアップを作成するキュー・スペース、または移動するキュー・スペースを含むグループをシャットダウンします。次に、アーカイブ・ツールを使用して、バックアップとして使用できるファイルなどの媒体、またはキュー・スペースを別のサーバに移動する際に使用されるファイルなどの媒体に、キュー・スペースのデバイスを保存します。

キュー・スペースの異種マシンへの移動

異なるアーキテクチャ (主にバイトの並び順) を持つマシンにキュー・スペースを移動する場合、その手順は複雑になります。まず、キュー・スペースにあるすべてのキューからすべてのメッセージを取り出すアプリケーション・プログラムを作成して実行します。次に、それらのメッセージをマシンに依存しない形式で書き出します。最後に、メッセージを新しいキュー・スペースに登録します。

TMQFORWARD および非グローバルのトランザクション

TMQFORWARD を使用してキューから取り出されて転送されるメッセージは、グループの境界を越えて処理されるので、グローバル・トランザクション内で実行されます。リソース・マネージに対応付けられていないサーバ、またはグローバル・トランザクション内で実行していないサーバによってメッセージが実行される場合は、TMSNAME=TMS (NULL XA インターフェイスの場合) であるサーバ・グループが必要です。

TMQFORWARD およびコミット制御

メッセージが実行のためにキューから取り出された際に TMQFORWARD で開始されたグローバル・トランザクションは、tpcommit() によって終了します。管理者は、コンフィギュレーション・ファイルに CMTRET パラメータを設定して、トランザクションのログへの記録時または完了時にトランザクションのコミットを設定できます。UBBCONFIG(5) の RESOURCES セクションの CMTRET の説明を参照してください。

トランザクション・タイムアウトの処理

トランザクション・タイムアウトを処理するには、キューの管理者と、メッセージをキューから取り出すクライアント・プログラムを開発するプログラマが協力することが必要です。tpdequeue(3c) が呼び出され、その *flags* 引数に TPQWAIT が設定されている場合、TMQUEUE サーバはメッセージがキューに到着するまで待機します。タイムアウトになるまでの時間 (秒単位) は、次の条件によって決定されます。

- tpbegin 呼び出しで指定されている timeout (トランザクションがクライアントで開始される場合)
- TMQUEUE サーバの -t *timeout* フラグ (クライアントがトランザクションを開始していない場合)

TPQWAIT が設定されている場合にメッセージがすぐに利用できないときは、TMQUEUE にはほかのサービス要求を処理するためのアクション・リソースが必要になります。キュー・スペースが同時に処理できるアクション数を増やすこともできます。その場合、qspacecreate または qspacechange コマンドに、-A *actions* オプションを使用します。このオプションは、同時に処理できる追加のアクション数を指定します。処理の待機中にほかのアクションを実行できる場合、条件が満たされるまでブロッキング操作は中断されます。実行できるアクションがない場合は、tpdequeue の呼び出しは失敗します。

TMQFORWARD および利用できないサービスに対する再試行

TMQFORWARD サーバがサービスにメッセージを転送したときにそのサービスが利用できなかった場合、キューに対する再試行回数の上限に達することがあります。メッセージは、エラー・キューが存在する場合はそこに移されます。このような状態を避けるには、管理者が TMQFORWARD サーバをシャットダウンするか、または再試行回数の上限値を増やします。

メッセージはエラー・キューに移動されると、元のキューとは関連がなくなります。サービスが利用可能になったときに、管理者がメッセージをサービス・キューに戻してエラーを修復すると、キュー名が TPQCTL 構造体の corrid に格納されて、キュー名がメッセージと対応付けられます。

Windows 標準入出力

2-15 ページの「キューの容量の上限」で説明した qchange ...Queue capacity command など、qmadmin() セッション内で設定したコマンドを実行するために、Windows の CreateProcess() 関数では DETACHED PROCESS として子プロセスが生成されます。このようなプロセスには、標準入出力用のコンソールがありません。そのため、たとえば、標準 DOS 構文で qchange ...Queue capacity command を設定して、組み込みの DOS コマンド (dir または date など) を実行し、次に標準出力をファイルにパイプまたは転送すると、コマンドが正常終了してもファイルは空です。

この問題を解決する方法として、たとえば、qchange ...Queue capacity command を実行するために、date /t > x.out コマンドでファイルの date 情報を取得します。その場合、次のように実行します。

```
qmadmin
> qopen yourQspace
> qchange yourQname
> go through all the setups... the threshold queue capacity warning,
   and so on
> "Queue capacity command:" cmd /c date /t > x.out
```

この処理を *yourFile*.cmd などのコマンド・ファイルから行う場合は、date /t > x.out コマンドを *yourFile*.cmd に追加し、次のように実行します。

```
qmadmin
> qopen yourQspace
> qchange yourQname
> go through all the setups... the threshold queue capacity warning,
   and so on
> "Queue capacity command: " yourFile.cmd
```

3 BEA Tuxedo /Q の C 言語プログラミング

ここでは、次の内容について説明します。

- はじめに
- 必要とされる知識
- 要求の発信元
- デフォルトの場合の注意事項
- メッセージのキューへの登録
- メッセージのキューからの取り出し
- メッセージの順次処理

はじめに

ここでは、キューへのメッセージの登録とキューからのメッセージの取り出しを行う ATMI C 言語関数 `tpenqueue(3c)`、`tpdequeue(3c)`、およびいくつかの補助関数の使用方法について説明します。

必要とされる知識

キュー機能を使用するクライアント・プログラムまたはサーバ・プログラムをコーディングする BEA TUXEDO プログラマには、BEA Tuxedo ATMI にバインドされた C 言語についての知識が必要です。BEA Tuxedo プログラミングに関する全般的な説明については、『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』を参照してください。ATMI 関数の詳細については、『BEA Tuxedo C リファレンス』を参照してください。

要求の発信元

BEA Tuxedo /Q のキューにメッセージを登録する呼び出しは、アプリケーションに対応付けられているあらゆるクライアント・プロセスまたはサーバ・プロセスから行うことができます。たとえば、次の発信元があります。

- キュー・スペースと同じマシン、またはネットワーク上の別のマシンにあるクライアントまたはサーバ。
- 会話型プログラム。ただし、キュー (または、`TMQUEUE(5)` サーバ) との会話接続は確立できません。
- サーバ側の代理プロセスを介するワークステーション・クライアント。管理インターフェイスも完全にサーバ側にあります。

デフォルトの場合の注意事項

ここでは、BEA Tuxedo /Q プログラミングについて、1-2 ページの「キュー・サービスの呼び出し」の主に左側部分について説明します。この図では、クライアント (または、クライアントとして機能するプロセス) は `tpenqueue(3c)` を呼び出し、`TMQUEUE(5)` サーバを通して利用できるキュー・

スペースを指定して、メッセージをキューに登録しています。クライアントは、その後、TMQUEUE への tpdequeue(3c) 呼び出しを介して、応答を取得します。

1-2 ページの「キュー・サービスの呼び出し」では、キューに入れられたメッセージが、サーバ TMQFORWARD(5) によってキューから取り出され、処理のために tpcall(3c) を介してアプリケーション・サーバに送信されています。tpcall() に対する応答が受信されると、TMQFORWARD は応答メッセージをキューに登録します。TMQFORWARD の主な目的は、キュー・スペースと既存のアプリケーション・サービスとの間にインターフェイスを提供することです。そのため、アプリケーションにコードを追加する必要はありません。そのため、ここでは、クライアントとキュー・スペースとの間の処理を中心に説明します。

キュー機能の使い方を簡単に示すサンプルがソフトウェアに提供されています。詳細については、A-1 ページの「サンプル・アプリケーション」を参照してください。

メッセージのキューへの登録

次は、tpenqueue() の構文です。

```
#include <atmi.h>
int tpenqueue(char *qspace, char *qname, TPQCTL *ctl,
     char *data, long len, long flags)
```

tpenqueue() が呼び出されると、*qspace* で識別されるキュー・スペース内の *qname* キューにメッセージを格納するようにシステムが指示されます。メッセージは *data* が指すバッファ内にあり、その長さは *len* で示されます。*flags* にビット設定を行うと、システムに tpenqueue() 呼び出しの処理方法が通知されます。登録されたメッセージおよび応答の処理方法のさらに詳しい情報は、*ctl* が指す TMQCTL 構造体で定義されます。

tpenqueue(3c) の引数

tpenqueue(3c) の処理を制御するいくつかの重要な引数があります。その一部について、以下に説明します。

tpenqueue():qspace 引数

qspace は、管理者によって既に作成されたキュー・スペースを識別します。サーバがコンフィギュレーション・ファイルの SERVERS セクションで定義されている場合、そのサーバが提供するサービス名は、実際のキュー・スペース名 (GROUPS セクションの OPENINFO パラメータの一部として指定されます) のエイリアスになります。たとえば、アプリケーションがサーバ TMQUEUE を使用する場合、*qspace* 引数が指す値は、TMQUEUE が宣言するサービス名になります。サービス名のエイリアスが何も定義されていない場合、デフォルトのサービス名はサーバ名 TMQUEUE と同じになります。その場合、コンフィギュレーション・ファイルには次の内容が記述されています。

```
TMQUEUE
        SRVGRP = QUE1  SRVID = 1
        GRACE = 0  RESTART = Y CONV = N
        CLOPT = "-A"
```

または

```
       CLOPT = "-s TMQUEUE"
```

サーバ・グループ QUE1 のエントリには、OPENINFO パラメータを使用して、リソース・マネージャ、デバイスのパス名、およびキュー・スペース名を指定します。クライアント・プログラムにおける qspace 引数は、次のように記述されます。

```
if (tpenqueue("TMQUEUE", "STRING", (TPQCTL *)&qctl,
        (char *)reqstr, 0,0) == -1) {
        Error checking
}
```

TMQUEUE(5) リファレンス・ページの例では、サーバを作成してコンフィギュレーション・ファイルで指定する際に、サービスのエイリアスを指定する方法が示されています。A-1 ページの「サンプル・アプリケーション」のサンプル・プログラムでも、サービスのエイリアスが指定されています。

tpenqueue():qname 引数

キュー・スペース内で、キューを使用してサービスを呼び出している場合、メッセージ・キューは要求を処理できるアプリケーション・サービスに従って命名されます。*qname* は、そのようなアプリケーション・サービスを指すポインタです。それ以外の場合、qname は、アプリケーション (メッセージをキューに登録したアプリケーション、または別のアプリケーション) によってキューから取り出されるまで、メッセージを格納しておく場所を単に示す名前です。

tpenqueue():data および len 引数

data は、処理対象のメッセージが格納されたバッファを指すポインタです。そのバッファは、tpalloc(3c) を呼び出して割り当てられたものであることが必要です。*len* は、メッセージの長さを指定します。BEA Tuxedo のバッファ・タイプには、メッセージの長さを指定する必要がないのもの (FML など) もあります。その場合、*len* は無視されます。*data* は NULL にすることもできます。その場合、*len* は無視され、メッセージはデータ部分なしでキューに登録されます。

tpenqueue():flags 引数

flags の値は、tpdequeue() 呼び出しの処理方法を BEA Tuxedo システムに通知するために使用されます。次は、有効なフラグです。

TPNOTRAN
: 呼び出し元がトランザクション・モードにあり、このフラグが設定されていると、メッセージは呼び出し元と同じトランザクション内ではキューに登録されません。このフラグを設定する、トランザクション・モードの呼び出し元は、メッセージをキューに登録するときに、やはりトランザクション・タイムアウトの影響を受けます (それ以外はなし)。キューへのメッセージの登録が失敗した場合、呼び出し元のトランザクションは影響されません。

TPNOBLOCK
: ブロッキング状態が存在する場合、メッセージはキューに登録されません。このフラグが設定されている場合に、メッセージの転送先である内部バッファがいっぱいであるなどのブロッキング状態が存

在すると、呼び出しは失敗し、tperrno(5) に TPEBLOCK が設定されます。このフラグが設定されている場合に、ターゲットのキューが別のアプリケーションによって 排他的にオープンされているというブロッキング状態が存在すると、呼び出しは失敗し、tperrno() に TPEDIAGNOSTIC が設定され、TPQCTL 構造体の診断フィールドに QMESHARE が設定されます。後者の場合、BEA Tuxedo システム以外の BEA 製品に基づくほかのアプリケーションが、キューイング・サービス API (QSAPI) を使用して読み取りと書き込み、またはそのいずれかを排他的に行うためにキューをオープンしています。

TPNOBLOCK が設定されていない場合に、ブロッキング状態が存在すると、その状態が解消されるかタイムアウト (トランザクション・タイムアウトまたはブロッキング・タイムアウト) が発生するまで、呼び出し元はブロックされます。タイムアウトが発生した場合、呼び出しは失敗し、tperrno() に TPETIME が設定されます。

TPNOTIME

このフラグを設定すると、呼び出し元は無制限にブロックでき、ブロッキング・タイムアウトが適用されなくなります。ただし、トランザクション・タイムアウトは発生する可能性があります。

TPSIGRSTRT

このフラグが設定されていると、基となるシステム・コールがシグナルによって中断された場合、中断されたシステム・コールが再度呼び出されます。このフラグが設定されていない場合に、シグナルによってシステム・コールが中断されると、呼び出しは失敗し、tperrno(5) に TPGOTSIG が設定されます。

TPQCTL 構造体

tpdequeue() の 3 番目の引数は、TPQCTL 型の構造体へのポインタです。TPQCTL 構造体には、アプリケーションで使用されるメンバと BEA Tuxedo システムで使用されるメンバがあり、アプリケーション・プログラムとキュー機能間の両方向でパラメータがやり取りされます。tpenqueue() を呼び出すクライアントは、フラグを設定して、システム側で入力する必要のあ

るフィールドをマークします。この構造体は、tpdequeue() でも使用されます。一部のフィールドは、アプリケーションが tpdequeue() を呼び出すまで使用されません。次のコード例は、この構造体全体を示しています。

コード リスト 3-1　tpqctl_t **構造体**

```
#define TMQNAMELEN 15
#define TMMSGIDLEN 32
#define TMCORRIDLEN 32

struct tpqctl_t {            /* キュー・プリミティブの制御パラメータ */
     long flags;             /* どの値が設定されるかのかを示します。*/
     long deq_time;          /* キューから取り出すときの絶対時間 / 相対時間 */
     long priority;          /* 登録優先順位 */
long diagnostic;       /*  失敗の原因を示します。*/
  char msgid[TMMSGIDLEN];          /* 既存メッセージの ID ( そのメッセージの前に登録されます ) */
     char corrid[TMCORRIDLEN];    /* メッセージを識別するときに使用される相関識別子 */
     char replyqueue[TMQNAMELEN+1];   /* 応答メッセージ用のキューの名前 */
     char failurequeue[TMQNAMELEN+1]; /* 異常終了メッセージを登録するキューの名前 */
     CLIENTID cltid;         /* 発信元クライアントのクライアント識別子 */
     long urcode;            /* アプリケーション・ユーザ戻りコード */
     long appkey;            /* アプリケーション認証クライアント・キー */
     long delivery_qos;      /* サービスの配信品質 */
     long reply_qos;         /* サービスの応答メッセージの品質 */
     long exp_time;          /* 有効期限の時間 */
};
typedef struct tpqctl_t TPQCTL;
```

以下は、tpdequeue() の入力情報を制御する *flags* パラメータの有効なフラグです。

TPNOFLAGS
: フラグおよび値は設定されません。制御構造体から情報は取得されません。構造体のフィールドに値を設定しないことは、TPNOFLAGS を設定することと同じです。

TPQTOP
: このフラグを設定すると、キューの順序付けは無効になり、メッセージはキューの先頭に登録されます。この要求は、順序付けを無

効にしてメッセージをキューの先頭に登録するようにキューが設定されているかどうかによって、使用できない場合があります。TPQTOP と TPQBEFOREMSGID は、相互に排他的なフラグです。

TPQBEFOREMSGID
: このフラグを設定すると、キューの順序付けが無効になり、メッセージは *ctl->msgid* によって識別されるメッセージの前に登録されます。この要求は、順序付けを無効にするようにキューが設定されているかどうかによって、使用できない場合があります。TPQTOP と TPQBEFOREMSGID は、相互に排他的なフラグです。メッセージ識別子の値は 32 バイト全体が意味を持つので、*ctl->msgid* で識別される値は、たとえば NULL 文字を埋め込むなどして、完全に初期化する必要があります。

TPQTIME_ABS
: このフラグが設定されていると、*ctl->deq_time* で指定された時間の経過後、メッセージが処理されます。*deq_time* は、使用しているオペレーティング・システムで利用できる場合は、time(2) または mktime(3C)、あるいは BEA Tuxedo システムで提供される gp_mktime(3c) によって生成される絶対時間です。*ctl->deq_time* に設定される値は、世界協定時 (UTC) 1970 年 1 月 1 日 00:00:00 から経過した秒数です。絶対時間は、キュー・マネージャ・プロセスが存在するマシン・クロックに基づいて設定されます。TPQTIME_ABS と TPQTIME_REL は、相互に排他的なフラグです。

TPQTIME_REL
: このフラグが設定されていると、キューへの登録が完了してからの相対時間の経過後、メッセージが処理されます。*ctl->deq_time* は、キューへの登録が完了してから、送信されたメッセージが処理されるまでの遅延秒数を指定します。TPQTIME_ABS と TPQTIME_REL は、相互に排他的なフラグです。

TPQPRIORITY
: このフラグが設定されていると、要求がキューに登録されるときの優先順位が *ctl->priority* に格納されます。優先順位は、1 以上 100 以下の範囲でなければなりません。優先順位によって順序付けられたキューでは、数値が高いほど優先順位も高くなり、高い数値のメッセージが低い数値のメッセージより先にキューから取り出されます。優先順位によって順序付けられていないキューの場合、値は情報にすぎません。

このフラグを設定しなかった場合、メッセージの優先順位はデフォルトの 50 になります。

TPQCORRID

このフラグが設定されていると、要求が tpdequeue(3c) によってキューから取り出されるときに、*ctl->corrid* に指定された相関識別子の値が使用されます。この識別子は、キューに登録されたすべての応答メッセージまたは異常終了メッセージに付加されるので、アプリケーションは応答を特定の要求に結び付けることができます。相関識別子の値は 32 バイト全体が意味を持つので、*ctl->corrid* に指定される値は、たとえば NULL 文字を埋め込むなどして、完全に初期化する必要があります。

TPQREPLYQ

このフラグが設定されていると、*ctl->replyqueue* に指定された応答キューが、キューに入れられたメッセージに対応付けられます。メッセージへの応答はすべて、要求メッセージと同じキュー・スペース内の、指定されたキューに登録されます。この文字列は NULL で終了し、15 文字以下であることが必要です。サービスに対する応答が生成されても、応答キューが指定されていないか存在しない場合は、その応答は削除されます。

TPQFAILUREQ

このフラグが設定されていると、*ctl->failurequeue* に指定されている異常終了キューが、キューに入れられたメッセージに対応付けられます。(1) キューに登録されたメッセージが TMQFORWARD() によって処理され、(2) TMQFORWARD が -d オプションで開始され、さらに (3) サービスが異常終了して NULL 以外の応答を返す場合は、その応答と 関連する tpurcode によって構成される異常終了メッセージが、元の要求メッセージと同じキュー・スペース内で指定されたキューに登録されます。この文字列は NULL で終了し、15 文字以下であることが必要です。

TPQDELIVERYQOS、TPQREPLYQOS

TPQDELIVERYQOS フラグが設定されていると、*ctl->delivery_qos* で指定されたフラグにより、メッセージの配信サービスの品質が制御されます。その場合、相互に排他的な 3 つのフラグ TPQQOSDEFAULTPERSIST、TPQQOSPERSISTENT、TPQQOSNONPERSISTENT のいずれかを *ctl->delivery_qos* に設定しなければなりません。

TPQDELIVERYQOS フラグが設定されていない場合、ターゲットのキューのデフォルトの配信方針がメッセージに対するサービスの配信品質を指定します。

TPQREPLYQOS フラグが設定されていると、*ctl->reply_qos* で指定されるフラグが、メッセージの応答に対するサービスの品質を制御します。その場合、相互に排他的な 3 つのフラグ TPQQOSDEFAULTPERSIST、TPQQOSPERSISTENT、TPQQOSNONPERSISTENT のいずれかを *ctl->reply_qos* に設定しなければなりません。TPQREPLYQOS フラグは、TMQFORWARD によって処理されるメッセージから応答が返されるときに使用されます。サービスを呼び出すときに TMQFORWARD を使用しないアプリケーションでは、自身の応答メカニズムのヒントとして TPQREPLYQOS フラグを使用できます。

TPQREPLYQOS が設定されていない場合、*ctl->replyqueue* キューのデフォルトの配信方針が応答に対するサービスの配信品質を指定します。デフォルトの配信方針は、メッセージに対する応答がキューに登録されるときに決定される点に注意してください。つまり、元のメッセージがキューに登録されてから応答が登録されるまでの間に、応答キューのデフォルトの配信方針が変更された場合、応答が最後に登録される時点で有効な方針が使用されます。

次は、*ctl->delivery_qos* と *ctl->reply_qos* の有効なフラグです。

TPQQOSDEFAULTPERSIST
: このフラグは、ターゲットのキューで指定されているデフォルトの配信方針を使用して、メッセージが配信されるように指定します。

TPQQOSPERSISTENT
: このフラグは、メッセージがディスク・ベースの配信方法を使用して、永続的な記憶域に配信されることを指定します。このフラグが設定されていると、ターゲットのキューに指定されたデフォルトの配信方針がこのフラグで上書きされます。

TPQQOSNONPERSISTENT
: このフラグは、メッセージがメモリ・ベースの配信方法を使用して、非永続的な記憶域に配信されることを指定します。特に、メッセージはキューから取り出されるまでメモリ内に登録されています。このフラグが設定されていると、ターゲットのキューに指定されたデフォルトの配信方針がこのフ

ラグで上書きされます。呼び出し元がトランザクション・モードの場合、非永続メッセージは呼び出し元のトランザクション内でキューに登録されます。ただし、システムがシャットダウンやクラッシュした場合、またはキュー・スペースとしての IPC 共用メモリが削除された場合、非永続メッセージは失われます。

TPQEXPTIME_ABS

このフラグが設定されていると、メッセージに有効期限の絶対時間が適用されます。これは、キューからメッセージが削除される絶対時間です。
有効期限の絶対時間は、キュー・マネージャ・プロセスが存在するマシン・クロックによって決定されます。

有効期限の絶対時間は、*ctl->exp_time* に格納された値で示されます。*ctl->exp_time* の値は、time(2)、mktime(3C)、または gp_mktime(3c) によって生成された絶対時間、つまり世界協定時 (UTC) 1970 年 1 月 1 日 00:00:00 から経過した秒数に設定されなければなりません。

キューへの登録操作の時間より早い絶対時間が指定されると、操作は成功しますが、メッセージはしきい値の計算の対象になりません。有効期限の時間がメッセージの使用可能時間より前の場合、使用可能時間が有効期限の切れる時間より前になるようにいずれかの時間を変更しない限り、メッセージをキューから取り出すことはできません。また、これらのメッセージがキューからの取り出しの対象になったことがなくても、有効期限が切れるとキューから削除されます。トランザクション内でメッセージの期限が切れた場合、それによってトランザクションが異常終了することはありません。トランザクション内でキューへの登録、またはキューからの取り出し中に有効期限が切れたメッセージは、トランザクションが終了した時点でキューから削除されます。メッセージの有効期限が切れたことの通知は行われません。

TPQEXPTIME_ABS、TPQEXPTIME_REL、および TPQEXPTIME_NONE は、相互に排他的なフラグです。この 3 つのどのフラグも設定されていない場合、ターゲットのキューに対応しているデフォルトの有効期限の時間がメッセージに適用されます。

TPQEXPTIME_REL

このフラグが設定されていると、メッセージに有効期限の相対時間が適用されます。これは、メッセージがキューに到達してから、

キューから削除されるまでの秒数です。有効期限の時間は、*ctl->exp_time* に格納された値で示されます。

有効期限の時間がメッセージの使用可能時間より前の場合、使用可能時間が有効期限の切れる時間より前になるようにいずれかの時間を変更しない限り、メッセージをキューから取り出すことはできません。また、これらのメッセージがキューからの取り出しの対象になったことがなくても、有効期限が切れるとキューから削除されます。トランザクション中にメッセージの期限が切れても、トランザクションが異常終了することはありません。トランザクション内でキューへの登録、またはキューからの取り出し中に有効期限が切れたメッセージは、トランザクションが終了した時点でキューから削除されます。メッセージの有効期限が切れたことの通知は行われません。

TPQEXPTIME_ABS、TPQEXPTIME_REL、および TPQEXPTIME_NONE は、相互に排他的なフラグです。この 3 つのどのフラグも設定されていない場合、ターゲットのキューに対応しているデフォルトの有効期限の時間がメッセージに適用されます。

TPQEXPTIME_NONE

このフラグが設定されていると、キューのデフォルトの方針に有効期限の時間が含まれている場合でも、メッセージの有効期限が切れることはありません。

TPQEXPTIME_ABS、TPQEXPTIME_REL、および TPQEXPTIME_NONE は、相互に排他的なフラグです。この 3 つのどのフラグも設定されていない場合、ターゲットのキューに対応しているデフォルトの有効期限の時間がメッセージに適用されます。

また、TPQCTL の *urcode* フィールドにユーザ戻りコードを設定することができます。この値は、メッセージをキューから取り出すために tpdequeue(3c) を呼び出すアプリケーションに返されます。

tpenqueue() からの出力では、次のフィールドが TPQCTL 構造体に設定されます。

```
long flags;             /* どの値が設定されるかを示します。*/
char msgid[32];         / キューに登録されたメッセージの ID */
long diagnostic;        /*  失敗の原因を示します。*/
```

次は、tpenqueue() からの出力情報を制御する *flags* パラメータの有効なビットです。tpenqueue() の呼び出し時にこのフラグがオンになっていると、/Q サーバ TMQUEUE(5) は、構造体の対応する要素にメッセージ識別子を挿入します。tpenqueue() の呼び出し時にこのフラグがオフになっていると、TMQUEUE() は、構造体の対応する要素にメッセージ識別子を挿入しません。

TPQMSGID
: このフラグが設定されている場合に、tpdequeue() の呼び出しが成功すると、メッセージ識別子が *ctl->msgid* に格納されます。メッセージ識別子の値は 32 バイト全体が意味を持つので、*ctl->msgid* に格納される値は、たとえば NULL 文字を埋め込むなどして、完全に初期化する必要があります。初期化に使用される実際の埋め込み文字は、BEA Tuxedo /Q コンポーネントのリリースによって異なります。

制御構造体の残りのメンバは、tpenqueue() への入力では使用されません。

tpdequeue() の呼び出しが異常終了して tperrno(5) に TPEDIAGNOSTIC が設定された場合、失敗の原因を示す値が *ctl->diagnostic* に返されます。次は、返される値です。

[QMEINVAL]
: 無効なフラグ値が指定されています。

[QMEBADRMID]
: 無効なリソース・マネージャ識別子が指定されています。

[QMENOTOPEN]
: リソース・マネージャが現在オープンされていません。

[QMETRAN]
: トランザクション・モードで、または TPNOTRAN フラグを設定して呼び出しが行われ、キューにメッセージを登録するトランザクションの開始を試みたときに、エラーが発生しました。この診断は、BEA Tuxedo リリース 7.1 以降のキュー・マネージャからは返されません。

[QMEBADMSGID]
: 無効なメッセージ識別子が指定されています。

[QMESYSTEM]
システム・エラーが発生しました。エラーの正確な内容がログ・ファイルに書き込まれます。

[QMEOS]
オペレーティング・システムのエラーが発生しました。

[QMEABORTED]
操作がアボートされました。アボートされた操作がグローバル・トランザクション内で実行されていた場合、グローバル・トランザクションは「ロールバックのみ」とマークされます。それ以外の場合、キュー・マネージャは操作をアボートします。

[QMEPROTO]
トランザクションの状態がアクティブではないときに、キューへの登録が行われました。

[QMEBADQUEUE]
無効または削除されたキューの名前が指定されています。

[QMENOSPACE]
キュー上に領域がないなどリソース不足が原因で、サービスの品質 (永続的な記憶域または非永続的な記憶域) が指定されたメッセージがキューに登録されませんでした。次のいずれかのリソース設定を超えると、QMENOSPACE が返されます。(1) キュー・スペースに割り当てられたディスク容量 (永続的)、(2) キュー・スペースに割り当てられたメモリ容量 (非永続的)、(3) 同時にアクティブ状態になるトランザクションの最大数 (キュー・スペースで許容される数であることが必要です)、(4) キュー・スペースに一度に入れることができる最大メッセージ数、(5) キューイング・サービス・コンポーネントが処理できる並列アクションの最大数、または (6) キューイング・サービス・コンポーネントを同時に使用できる認証されたユーザの最大数。

[QMERELEASE]
新機能がサポートされていないバージョンの BEA Tuxedo システムのキュー・マネージャに対して、メッセージのキューへの登録が試みられました。

[QMESHARE]
: 指定されたキューのメッセージの登録時に、そのキューが別のアプリケーションによって 排他的にオープンされています。別のアプリケーションとは、BEA Tuxedo システム以外の BEA 製品に基づくもので、キューをオープンして、キューイング・サービス API (QSAPI) を使用して読み取りおよび書き込み、またはそのいずれかを排他的に行っています。

キューの順序の無効化

キューの作成時に、管理者が tpenqueue() 呼び出しでキュー上のメッセージの順序を無効にできるようにした場合、次の 2 つの方法でこの機能を利用できます。この 2 つの方法は、相互に排他的です。*flags* に TPQTOP を設定すると、メッセージをキューの先頭に置くことができます。または、*flags* に TPQBEFOREMSGID、*ctl->msgid* に既存のメッセージの ID を設定して、メッセージを特定の既存のメッセージの前に置くこともできます。この方法では、メッセージ ID を使用できるように、以前の呼び出しで取得されたメッセージ ID が保存されていることが必要です。管理者は、キューでサポートされている方法を通知する必要があります。キューは、この 2 つのいずれかまたは両方を使用できるように、あるいはどちらも使用できないように作成できます。

キューの優先順位の無効化

ctl->priority に値を設定して、メッセージの優先順位を指定することができます。この値は、1 以上 100 以下の範囲でなければなりません。数値が高いほど優先順位も高くなります。キューの順序付けパラメータの中に priority が含まれていない場合、ここで優先順位を設定しても取り出しの順序には影響しません。ただし、優先順位の値は保持されるので、メッセージがキューから取り出されるときに検査されます。

メッセージの使用可能時間の設定

deq_time に、絶対時間またはキューへの登録が完了してからの相対時間として、メッセージが処理されるまで時間を指定できます。*flags* に、TPQTIME_ABS または TPQTIME_REL のいずれかを設定して、値の処理方法を指定できます。キューは、time を順序付けの基準として作成することができます。その場合、メッセージは使用可能時間によって順序付けされます。

BEA Tuxedo /Q には、gp_mktime(3c) 関数が提供されています。この関数は、tm 構造体の日付と時刻を 1970 年 1 月 1 日から経過した秒数に変換します。time(2) および mktime(3C) 関数を gp_mktime(3c) の代わりに使用することもできます。値は、time_t 型 (typedef'd で指定された long 型) で返されます。キューからメッセージが取り出される絶対時間を設定するには、次の手順に従います。ここでは、2001 年 12 月 9 日正午 12:00 を使用します。

1. 使用する日時の値を tm 構造体に入力します。

```
#include <stdio.h>
#include <time.h>
static char *const wday[] = {
         "Sunday", "Monday", "Tuesday", "Wednesday",
         "Thursday", "Friday", "Saturday", "-unknown-"
};
struct tm time_str;
/*...*/
time_str.tm_year = 2001 - 1900;
time_str.tm_mon = 12 - 1;
time_str.tm_mday = 9;
time_str.tm_hour = 12;
time_str.tm_min = 0;
time_str.tm_sec = 1;
time_str.tm_isdst = -1;
```

2. gp_mktime を呼び出して deq_time に値を代入し、*flags* を設定して絶対時間が提供されることを示します。

```
#include <atmi.h>
TPQCTL qctl;
if ((qctl->deq_time = (long)gp_mktime(&time_str)) == -1) {
       /* エラーのチェック */
}
qctl->flags = TPQTIME_ABS
```

3. tpenqueue() を呼び出します。

```
if (tpenqueue(qspace, qname, qctl, *data,*len,*flags) == -1) {
       /* エラーのチェック */
}
```

キューからメッセージを取り出す相対時間、たとえば、キューへの登録操作が完了してから *nnn* 秒などを指定する場合、deq_time に秒数を指定し、flags に TPQTIME_REL を設定します。

tpenqueue() とトランザクション

tpenqueue() の呼び出し元がトランザクション・モードにある場合に、TPNOTRAN が設定されていないと、キューへの登録は呼び出し元のトランザクション内で行われます。呼び出し元は、TPENQUEUE() が成功したか失敗したかによって、メッセージがキューに登録されたかどうかを判断できます。呼び出しが正常に行われると、メッセージがキューに登録されたことが保証されます。呼び出しが失敗すると、メッセージがキューに登録された部分も含めて、トランザクションがロールバックされます。

tpenqueue() の呼び出し元がトランザクション・モードにない場合、または TPNOTRAN が設定されている場合、メッセージは呼び出し元のトランザクションとは別のトランザクションでキューに登録されます。tpenqueue() への呼び出しが正常な戻り値を返した場合、メッセージがキューに登録されたことが保証されます。tpenqueue() の呼び出しが通信エラーまたはタイムアウトによって失敗した場合は、その障害がメッセージ登録の前に発生したのか後に発生したのか、呼び出し元には判断できません。

呼び出し元がトランザクション・モードにないときに TPNOTRAN を指定しても意味がありません。

メッセージのキューからの取り出し

次は、tpdequeue() の構文です。

```
#include <atmi.h>
int tpdequeue(char *qspace, char *qname, TPQCTL *ctl, \
        char **data, long *len, long flags)
```

この呼び出しが行われると、*qspace* で識別されるキュー・スペース内の *qname* キューからメッセージを取り出すようにシステムが指示されます。メッセージは、**data* が指すアドレスにあるバッファ (tpalloc(3c) によって割り当てられたもの) に挿入されます。*len* はデータの長さを示します。tpdequeue() から返された *len* が 0 の場合、そのメッセージにはデータ部分がないことを示します。*flags* にビット設定を行うと、システムに tpenqueue() 呼び出しの処理方法が通知されます。*ctl* が指す TPQCTL 構造体には、この呼び出しの処理方法についての詳細な情報が記述されます。

tpdequeue(3c) の引数

tpdequeue(3c) の処理を制御するいくつかの重要な引数があります。その一部について、以下に説明します。

tpdequeue():qspace 引数

qspace は、管理者によって既に作成されたキュー・スペースを識別します。TMQUEUE サーバがコンフィギュレーション・ファイルの SERVERS セクションで定義されている場合、そのサーバが提供するサービス名は、実際のキュー・スペース名 (GROUPS セクションの OPENINFO パラメータの一部として指定されます) のエイリアスになります。たとえば、アプリケーションがサーバ TMQUEUE を使用する場合、*qspace* 引数が指す値は、TMQUEUE が宣言するサービス名になります。サービス名のエイリアスが何も定義されていない場合、デフォルトのサービス名はサーバ名 TMQUEUE と同じになります。その場合、コンフィギュレーション・ファイルには次が記述されています。

```
TMQUEUE
        SRVGRP = QUE1  SRVID = 1
        GRACE = 0  RESTART = Y CONV = N
        CLOPT = "-A"
または
        CLOPT = "-s TMQUEUE"
```

サーバ・グループ QUE1 のエントリには、OPENINFO パラメータを使用して、リソース・マネージャ、デバイスのパス名、およびキュー・スペース名を指定します。クライアント・プログラムにおける *qspace* 引数は、次のように記述されます。

```
if (tpdequeue("TMQUEUE", "REPLYQ", (TPQCTL *)&qctl,
        (char **)&reqstr, &len,TPNOTIME) == -1) {
        Error checking
}
```

TMQUEUE(5) リファレンス・ページの例では、サーバを作成してコンフィギュレーション・ファイルで指定する際に、サービスのエイリアスを指定する方法が示されています。A-1 ページの「サンプル・アプリケーション」のサンプル・プログラムでも、サービスまたはキュー・スペース名のエイリアスが指定されています。

tpdequeue():qname 引数

キュー・スペース内のキュー名は、そのキュー・スペースにアクセスするアプリケーション間で一貫していなければなりません。これは、応答キューでは特に重要です。qname が応答キューを参照する場合、管理者はほかのキューと同じ方法で応答キュー、そして多くの場合、エラー・キューも作成します。*qname* は、メッセージまたは応答を取り出すキューを指すポインタです。

tpdequeue():data および len 引数

この 2 つの引数は、tpenqueue() で使用される場合と若干意味が異なります。**data* は、キューから取り出されたメッセージをシステムが格納するバッファのアドレスを指します。tpdequeue() が呼び出された場合に、この値が NULL であることはエラーです。

tpdequeue() が戻ると、*len* は取り出されたデータのデータ長情報を持つ long 型を指します。0 は応答にデータがなかったことを示します。アプリケーションによっては、これは正当で正常な応答です。長さ 0 の応答を受信した場合でも、それをキューに登録された要求の正常処理を示すために使用できます。バッファが tpdequeue() 呼び出しの前と比べて変更されているかどうかを確認する場合は、tpdequeue() 呼び出しの前にデータ長を保存し、それを呼び出しが終了した後で *len* と比較します。

tpdequeue():flags 引数

flags の値は、tpdequeue() 呼び出しの処理方法を BEA Tuxedo システムに通知するために使用されます。次は、有効なフラグです。

TPNOTRAN

呼び出し元がトランザクション・モードにある場合に、このフラグが設定されていると、メッセージは呼び出し元のトランザクション内ではキューから取り出されません。このフラグを設定するトランザクション・モードの呼び出し元には、メッセージのキューから取り出し時にトランザクション・タイムアウトが適用されます。それ以外は適用されません。メッセージのキューからの取り出しが失敗した場合でも、呼び出し元のトランザクションは影響されません。

TPNOBLOCK

ブロッキング状態が存在する場合、メッセージはキューから取り出されません。このフラグが設定されている場合に、メッセージの転送先である内部バッファがいっぱいであるなどのブロッキング状態が存在すると、呼び出しは失敗し、tperrno(5) に TPEBLOCK が設定されます。このフラグが設定されている場合に、ターゲットのキューが別のアプリケーションによって 排他的にオープンされているというブロッキング状態が存在すると、呼び出しは失敗し、tperrno() に TPEDIAGNOSTIC が設定され、TPQCTL 構造体の診断フィールドに QMESHARE が設定されます。後者の場合、BEA Tuxedo システム以外の BEA 製品に基づくほかのアプリケーションが、キューイング・サービス API (QSAPI) を使用して読み取りと書き込み、またはそのいずれかを排他的に行うためにキューをオープンしています。

TPNOBLOCK が設定されていない場合に、ブロッキング状態が存在すると、その状態が解消されるかタイムアウト (トランザクション・

タイムアウトまたはブロッキング・タイムアウト) が発生するまで、呼び出し元はブロックされます。(TPQCTL 構造体の) *flags* に TPQWAIT オプションが指定されている場合は、このブロッキング条件には、キュー自体のブロッキングは含まれません。

TPNOTIME
: このフラグを設定すると、呼び出し元は無制限にブロックでき、ブロッキング・タイムアウトが適用されなくなります。ただし、トランザクション・タイムアウトは発生する可能性があります。

TPNOCHANGE
: このフラグが設定されていると、**data* が指すバッファのタイプは変更されません。デフォルトでは、**data* が指すバッファとは異なるタイプのバッファが受信されると、受信側が着信バッファのタイプを識別する限り、**data* のバッファ・タイプは、受信されたバッファのタイプに変更されます。つまり、受信されたバッファのタイプとサブタイプは、**data* が指すバッファのタイプとサブタイプと一致しなければなりません。

TPSIGRSTRT
: このフラグが設定されていると、基となるシステム・コールがシグナルによって中断された場合、中断されたシステム・コールが再度呼び出されます。このフラグが設定されていない場合に、シグナルによってシステム・コールが中断されると、呼び出しは失敗し、tperrno(5) に TPGOTSIG が設定されます。

TPQCTL 構造体

tpdequeue() の 3 番目の引数は、TPQCTL 型の構造体へのポインタです。TPQCTL 構造体には、アプリケーションで使用されるメンバと BEA Tuxedo システムで使用されるメンバがあり、アプリケーション・プログラムとキュー機能間の両方向でパラメータがやり取りされます。tpdequeue() を呼び出すクライアントはフラグを設定して、システムで値が提供されるべきフィールドをマークします。前述のように、この構造体は tpenqueue() でも使用されます。一部のメンバは、tpenqueue() だけに適用されます。この構造体全体のコード例は、3-7 ページの「tpqctl_t 構造体」に示されています。

tpdequeue() への入力では、次のフィールドを TPQCTL 構造体に設定します。

```
long flags;             /* どの値が設定されるかを示します。*/
char msgid[32];   /* キューから取り出すメッセージの ID */
char corrid[32];  /* キューから取り出すメッセージの相関識別子 */
```

次は、tpdequeue() の入力として有効なフラグです。

TPNOFLAGS
: フラグは設定されません。制御構造体から情報は取得されません。

TPQGETBYMSGID
: このフラグが設定されていると、*ctl->msgid* で識別されるメッセージ識別子を持つメッセージが取り出されることが要求されます。メッセージ識別子は、以前に呼び出された tpenqueue() で決定されています。メッセージがあるキューから別のキューに移動された場合、メッセージ識別子は変更されます。メッセージ識別子の値は 32 バイト全体が意味を持つので、*ctl->msgid* で識別される値は、たとえば NULL 文字を埋め込むなどして、完全に初期化される必要があります。

TPQGETBYCORRID
: このフラグが設定されていると、*ctl->corrid* で指定された相関識別子を持つメッセージを取り出すことが要求されます。相関識別子は、アプリケーションが tpenqueue() でキューにメッセージを登録したときに指定されます。相関識別子の値は 32 バイト全体が意味を持つので、*ctl->corrid* に指定される値は、たとえば NULL 文字を埋め込むなどして、完全に初期化する必要があります。

TPQWAIT
: このフラグが設定されていると、キューが空の場合はエラーが返されません。代わりに、メッセージを取り出すことができるようになるまで、プロセスが待機します。TPQWAIT が TPQGETBYMSGID または TPQGETBYCORRID と併せて設定されている場合、指定されたメッセージ識別子または相関識別子を持つメッセージがキューに存在しないときは、エラーが返されません。代わりに、基準を満たすメッセージを取り出すことができるようになるまで、プロセスは待機します。プロセスには呼び出し元のトランザクション・タイムアウトが適用されます。また、トランザクション・モードではない場合、TMQUEUE プロセスで -t オプションで指定されたタイムアウトが適用されます。

要求する基準を満たすメッセージがすぐには利用できず、設定されたアクションのリソースを使い尽くしてしまった場合には、tpdequeue() は -1 を返し tperrno() を TPEDIAGNOSTIC に設定し、TPQCTL 構造体の診断フィールドを QMESYSTEM に設定します。

TPQWAIT 制御パラメータを指定する tpdequeue() の各要求では、条件を満たすメッセージがすぐに利用できない場合、キュー・マネージャ (TMQUEUE) のアクション・オブジェクトを使用できる必要があります。アクション・オブジェクトが使用できない場合、tpdequeue() 要求は失敗します。利用できるキュー・マネージャのアクション数は、キュー・スペースの作成時または変更時に指定されます。待機中のキューからの取り出し要求が完了すると、対応するアクション・オブジェクトは別の要求に使用できるようになります。

TPQPEEK

このフラグが設定されていると、指定されたメッセージが読み取られてもキューから削除されません。TPNOTRAN フラグも設定する必要があります。

あるスレッドが TPQPEEK を使用してメッセージを破棄せずにキューから取り出す場合、その取り出し要求をシステムが処理している短時間の間、非ブロッキング状態のほかのメッセージ取り出し操作にメッセージが認識されないことがあります。たとえば、特定の選択基準 (メッセージ識別子や相関識別子など) を使用してメッセージをキューから取り出す操作が、破棄せずに取り出しが現在行われているメッセージを探している場合などがあります。

次は、tpdequeue() からの出力情報を制御する *flags* パラメータの有効なビットです。どのフラグ・ビットでも、tpdequeue() の呼び出し時にオンになっていると、メッセージがキューに登録されたときに提供された値が、構造体の対応するフィールド (3-7 ページの「tpqctl_t 構造体」を参照) に格納され、そのビットは設定されたままになります。値が使用できない (つまり、メッセージがキューに登録されたときに値が指定されなかった) 場合、または tpdequeue() を呼び出したときにビットが設定されなかった場合、tpdequeue() はフラグがオフの状態で完了します。

TPQPRIORITY

このフラグが設定され、tpdequeue() の呼び出しが成功し、メッセージが明示的な優先順位でキューに登録された場合、その優先順位は *ctl->priority* に格納されます。優先順位は 1 以上 100 以内の範囲

内で、数値が高いほど優先順位も高くなります。つまり、高い数値のメッセージが低い数値のメッセージよりも先にキューから取り出されます。優先順位によって順序付けられていないキューの場合、値は情報にすぎません。

メッセージがキューに登録されるときに優先順位が明示的に指定されずに tpdequeue() 呼び出しが正常に終了した場合、このメッセージの優先順位は 50 になります。

TPQMSGID
: このフラグが設定されている場合に、tpdequeue() の呼び出しが成功すると、メッセージ識別子が *ctl->msgid* に格納されます。メッセージ識別子の値は、32 バイト全体が意味を持ちます。

TPQCORRID
: このフラグが設定され、tpdequeue() の呼び出しが成功し、メッセージが相関識別子によってキューに登録されていた場合、相関識別子が *ctl->corrid* に格納されます。相関識別子の値は、32 バイト全体が意味を持ちます。BEA Tuxedo /Q が提供するメッセージの応答には、すべて元の要求メッセージの相関識別子が付いています。

TPQDELIVERYQOS
: このフラグが設定され、tpdequeue() の呼び出しが成功し、メッセージがサービスの配信品質と共にキューに登録されていた場合、TPQQOSDEFAULTPERSIST、TPQQOSPERSISTENT、または TPQQOSNONPERSISTENT フラグが *ctl->delivery_qos* に格納されます。メッセージがキューに登録されたときに配信サービスの品質が明示的に指定されていない場合は、対象となるキューのデフォルトの配信方針によってメッセージ配信の品質が決まります。

TPQREPLYQOS
: このフラグが設定され、tpdequeue() の呼び出しが成功し、メッセージがサービスの応答品質と共にキューに登録されていた場合、TPQQOSDEFAULTPERSIST、TPQQOSPERSISTENT、または TPQQOSNONPERSISTENT フラグが *ctl->reply_qos* に格納されます。メッセージがキューに登録されていたときに応答のサービス品質が明示的に指定されていない場合、*ctl->replyqueue* キューのデフォルトの配信方針によって、すべての応答に対する配信サービスの品質が決まります。

デフォルトの配信方針は、メッセージに対する応答がキューに登録されるときに決定される点に注意してください。つまり、元のメッセージがキューに登録されてから応答が登録されるまでの間に、応答キューのデフォルトの配信方針が変更された場合、応答が最後に登録される時点で有効な方針が使用されます。

TPQREPLYQ
: このフラグが設定され、tpdequeue() の呼び出しが成功し、メッセージが応答キューと共にキューに登録されていた場合、その応答キューの名前が *ctl->replyqueue* に格納されます。メッセージへの応答は、要求メッセージと同じキュー・スペース内の指定されたキューに登録されます。

TPQFAILUREQ
: このフラグが設定され、tpdequeue() の呼び出しが成功し、メッセージが異常終了キューと共にキューに登録されていた場合、その異常終了キューの名前が *ctl->failurequeue* に格納されます。異常終了メッセージは、要求メッセージと同じキュー・スペース内の指定された異常終了キューに登録されます。

flags パラメータの残りのビットは、tpdequeue() が呼び出されるとクリア (0 に設定) されます。これには、TPQTOP、TPQBEFOREMSGID、TPQTIME_ABS、TPQTIME_REL、TPQEXPTIME_ABS、TPQEXPTIME_REL、および TPQEXPTIME_NONE が含まれます。これらのビットは、tpenqueue() の入力情報を制御する *flags* パラメータの有効なビットです。

tpenqueue() の呼び出しが失敗し、tperrno(5) に TPEDIAGNOSTIC が設定された場合は、失敗の原因を示す値が *ctl->diagnostic* に返されます。*ctl->diagnostic* の有効なコードには、3-6 ページの「TPQCTL 構造体」に示されている tpenqueue() の値 (QMENOSPACE と QMERELEASE を除く) のほかに、次のものがあります。

[QMENOMSG]
: キューから取り出せるメッセージはありません。メッセージがキュー上に存在し、別のアプリケーション・プロセスがこのメッセージをキューから読み取った可能性があることに注意してください。その場合、その別のプロセスがトランザクションをロールバックしたときに、メッセージがキューに戻されます。

[QMEINUSE]

メッセージ識別子または相関識別子を使用してメッセージをキューから取り出す際に、指定されたメッセージが別のトランザクションによって使用されています。それ以外の場合、現在キューにあるすべてのメッセージは、ほかのトランザクションによって使用されています。この診断は、BEA Tuxedo リリース 7.1 以降のキュー・マネージャからは返されません。

TPQWAIT の使用

TPQCTL 構造体の *flags* に TPQWAIT が設定されて tpdequeue() が呼び出された場合に、メッセージをすぐに取り出すことができないと、tpdequeue() が呼び出し元に制御を返す前に、TMQUEUE サーバ は tpdequeue() 要求に一致するメッセージがキューに到着するのを待ちます。TMQUEUE プロセスは待機中の要求を無視し、ほかのプロセスの要求を処理しながら、最初の要求の条件が満たされるのを待ちます。TPQGETBYMSGID や TPQGETBYCORRID も指定されている場合、サーバは指定されたメッセージ識別子や相関識別子を持つメッセージをキューから取り出せるようになるまで待ちます。このどちらのフラグも設定されていない場合、サーバは任意のメッセージがキューに登録されるまで待ちます。呼び出しがトランザクション・モードの場合、サーバの待機時間は、呼び出し元のトランザクション・タイムアウトによって制御されます。呼び出しがトランザクション・モードでない場合、TMQUEUE サーバの CLOPT パラメータに指定される -t オプションで制御されます。

TMQUEUE サーバは、要求を処理するアクション・リソースを利用できる限り、複数の待機中 TPDEQUEUE() 要求を同時に処理できます。十分なアクション・リソースがキュー・スペースに対して設定されていない場合、TPDEQUEUE() は失敗します。使用しているシステムがこれに該当する場合は、キュー・スペースに対するアクション・リソースの数を増やしてください。

TMQFORWARD サービス使用時のエラー処理

キューからのメッセージの取り出しで、エラー処理の最良の方法を考慮する場合、次の 2 種類のエラーを区別しておきます。

- 要求されたサービスに転送するために、TMQFORWARD(5) がメッセージをキューから取り出すときに発生するエラー
- 要求を処理するサービスに発生するエラー

デフォルトでは、メッセージがトランザクション内でキューから取り出され、そのトランザクションがロールバックされると、retry パラメータが 0 より大きい場合は、そのメッセージはキューに戻されて、再度キューからの取り出しと実行が可能になります。一時的な障害が解消されるように (たとえば、データベースのロックを別のトランザクションが解除するように) 少し時間をおいてから、メッセージの取り出しと実行を再度試みます。通常、再試行の回数に上限を設定することも、アプリケーション不備によってリソースを著しく浪費することを防ぐために有用です。管理者がキューを設定するときに、再試行の回数と遅延時間 (秒単位) の両方を指定できます。再試行の回数 0 は、再試行が行われないことを示します。再試行の回数が上限に達すると、管理者がキュー・スペースに設定するエラー・キューにメッセージが移動されます。エラー・キューが設定されていない場合、再試行回数の上限に達したメッセージは単に削除されます。エラー・キューにあるメッセージは、管理者によって処理されなければなりません。管理者は、アプリケーションの要件を満たす方法で発信元に通知する必要があります。選択されたメッセージ処理方法は、メッセージをキューに登録した発信元プログラムにほとんど透過的です。メッセージは一度正常にキューに登録されると、tpenqueue() のパラメータおよびキューの属性に従って処理されることが実質的には保証されます。メッセージがエラー・キューに移動されたことの通知は、キューのパラメータを適切に調整されたシステムではほとんどありません。

異常終了キュー (通常、キュー・スペースのエラー・キューとは異なります) は、キューに登録された各メッセージと対応付けられます。このキューは、異常終了メッセージを置く場所として、キューにメッセージを登録する呼び出しで指定されます。特定の要求に対する異常終了メッセージは、アプリケーション生成の相関識別子で識別できます。相関識別子は、キューにメッセージを登録するときに、そのメッセージと対応付けられます。

成功するまで (または事前に定義された回数まで) 再試行を繰り返すというデフォルトの動作は、時間が経過すれば解決されるような一時的な障害が失敗の原因であり、メッセージが後から適切に処理される場合は適しています。

障害が一時的なものではない場合もあります。たとえば、存在しないアカウントでの操作を要求するメッセージがキューに入れられている場合 (アプリケーションも適当な時間に存在していない場合) です。その場合、再試行してリソースを浪費しないようにします。アプリケーション・プログラマまたは管理者が、特定の操作の失敗が一時的ではないと判断した場合、単に再試行回数を 0 に設定します。ただし、この場合、このようなメッセージが入るキュー・スペースのエラー・キューを継続的にクリアするメカニズム (キューを定期的に読み取るバックグラウンドのクライアントなど) が必要です。また、同じサービスに対する障害でも、一時的なもの (データベース・ロックの競合など) であったり、永続的なもの (アカウントが存在しないなど) であったりする場合が多くあります。

メッセージが TMQFORWARD によって処理される (キューから取り出され、tpcall() を介してアプリケーションに渡される) 場合、tpcall() が返す情報には、TPESVCFAIL エラーが一時的な障害と永続的な障害のどちらによって生じたものであるかを示すメカニズムはありません。

キューからの取り出しをアプリケーションが処理する場合、簡単な解決方法はたとえ操作が失敗しても、そのサービスに対して成功を返すことです。つまり、TPSUCCESS を設定して tpreturn を実行します。これにより、トランザクションはコミット可能になり、メッセージはキューから削除されます。応答メッセージが使用されている場合、サービスから返されるバッファの情報によって操作の失敗を示すことができ、メッセージが応答キューに登録されます。tpreturn の *rcode* 引数も、アプリケーション固有の情報を返すために使用できます。

サービスが失敗し、トランザクションをロールバックする必要がある場合、TMQFORWARD が処理をそれ以上行わずに、2 番目のトランザクションを実行してキューからメッセージを削除するかどうかは明確ではありません。デフォルトでは、TMQFORWARD は失敗したサービスに対するメッセージを削除しません。TMQFORWARD のトランザクションはロールバックされ、メッセージはキューに格納されます。TMQFORWARD にコマンド行オプションを指定すると、サービスが失敗して 0 より長い応答メッセージが返された場合に、メッセージがキューから削除されます。メッセージは、2 番目のトランザクションで削除されます。この処理を行うには、キューに遅延時間および再試行回数を設定する必要があります。メッセージが失敗したキューに対応付けられている場合、メッセージがキューから削除されたトランザクションと同じトランザクションで、応答データが異常終了キューに登録されます。

TMQFORWARD を通して呼び出されたサービスからの応答をキューから取り出す手順

アプリケーションがキューに登録されたメッセージに対する応答を要求している場合、次の手順に従います。

1. 前提条件として、キュー・スペースに応答キューおよび異常終了キューがあることが必要です。アプリケーションは、相関識別子の内容について一貫していなければなりません。サービスは、論理的な失敗の場合に TPSUCCESS を返し、tpreturn の *rcode* 引数に情報を示すコードを返すようにコーディングします。

2. tpenqueue() を呼び出してキューにメッセージを登録する際に、次のフラグ・ビットがオンになるように flags を設定します。

   ```
   TPQCORRID     TPQREPLYQ
   TPQFAILUREQ   TPQMSGID
   ```

   この呼び出しを行う前に、corrid、replyqueue、および failurequeue の値を設定します。呼び出しから制御が戻ったら、corrid を保存します。

3. tpdequeue() を呼び出して応答を確認する際に、*qname* 引数に応答キューを指定し、次のフラグ・ビットがオンになるように flags を設定します。

   ```
   TPQCORRID       TPQREPLYQ
   TPQFAILUREQ     TPQMSGID
   TPQGETCORRID
   ```

   この呼び出しを行う前に、保存した相関識別子を使用して corrid に入力します。tpdequeue() の呼び出しが失敗し、tperrno(5) に TPEDIAGNOSTIC が設定された場合、詳しい情報を diagnostic から取得できます。エラー・コード QMENOMSG を受信した場合、キューから取り出すことのできるメッセージがなかったことを示します。

4. もう 1 つの tpdequeue() 呼び出しを設定します。この呼び出しでは、*qname* が異常終了キューの名前を指し、次のフラグ・ビットがオンになるように flags を設定します。

   ```
   TPQCORRID       TPQREPLYQ
   TPQFAILUREQ     TPQMSGID
   TPQGETBYCORRID
   ```

corrid に相関識別子を挿入します。呼び出しから制御が戻ったら、*len* を調べてデータが受信されたかどうかを確認し、また、urcode を調べてサービスがユーザ戻りコードを返したかどうかを確認します。

メッセージの順次処理

メッセージの順次処理は、あるサービスがそのトランザクションがコミットされる前に、連鎖的に次のサービス用にメッセージをキューに登録することによって行われます。最初の発信元のプロセスは、reply_queue に対する一連の TPDEQUEUE() 呼び出しによって順次処理の進行状況をトラッキングできます。ただし、各メンバが同じ相関識別子を使用し、長さ 0 の応答を返すことが必要です。

また、任意通知型通知を使用して、順次処理全体が成功したという通知を最初の発信元に返すこともできます。順次処理の最後のトランザクションが tpcommit で終了したことを確認するには、tpdequeue() から戻された TPQCTL 構造体、またはサービスに渡された TPSVCINFO 構造体で渡されるクライアント識別子を使用して、tpnotify を呼び出す操作を追加します。最初の発信元であるクライアントは、tpsetunsol を呼び出して、使用されている任意通知型メッセージ・ハンドラを指定しておかなければなりません。

ピア・ツー・ピア通信でのキューの使用

キューへのメッセージの登録およびキューからのメッセージの取り出しに関するこれまでの説明では、キューが要求 / 応答の一形態として使用されていることが暗黙の前提になっていました。メッセージ自体はサービス要求である必要はありません。キュー機能は、あるプロセスから別のプロセスに、サービス要求と同じように効果的にデータを転送できます。アプリケーション間またはクライアント間のこの通信方式は、ピア・ツー・ピア通信と呼ばれます。

使用するアプリケーションが、このような目的で BEA Tuxedo /Q を使用することに適している場合は、管理者に別のキューを作成してもらい、そのキューからメッセージを取り出す独自の受信用プログラムをコーディングします。

4 BEA Tuxedo/Q COBOL 言語プログラミング

ここでは、次の内容について説明します。

- はじめに
- 必要とされる知識
- 要求の発信元
- デフォルトの場合の注意事項
- メッセージのキューへの登録
- メッセージのキューからの取り出し
- メッセージの順次処理

はじめに

ここでは、キューへのメッセージの登録とキューからのメッセージの取り出しを行う ATMI COBOL 言語関数 `TPENQUEUE(3cbl)`、`TPDEQUEUE(3cbl)`、およびいくつかの補助関数の使用方法について説明します。

必要とされる知識

キュー機能を使用するクライアント・プログラムまたはサーバ・プログラムをコーディングする BEA TUXEDO プログラマには、BEA Tuxedo ATMI にバインドされた COBOL 言語についての知識が必要です。BEA Tuxedo プログラミングに関する全般的な説明については、『COBOL を使用した BEA Tuxedo アプリケーションのプログラミング』を参照してください。ATMI 関数の詳細については、『BEA Tuxedo COBOL リファレンス』を参照してください。

要求の発信元

BEA Tuxedo /Q のキューにメッセージを登録する呼び出しは、アプリケーションに対応付けられているあらゆるクライアント・プロセスまたはサーバ・プロセスから行うことができます。たとえば、次の発信元があります。

- キュー・スペースと同じマシン、またはネットワーク上の別のマシンにあるクライアントまたはサーバ。
- 会話型プログラム。ただし、キュー (または、`TMQUEUE(5)` サーバ) との会話接続は確立できません。
- ネイティブ側の代理プロセスを介するワークステーション・クライアント。管理インターフェイスも完全にサーバ側にあります。

デフォルトの場合の注意事項

ここでは、BEA Tuxedo /Q プログラミングについて、1-2 ページの「キュー・サービスの呼び出し」の主にクライアント部分について説明します。この図では、クライアント (または、クライアントとして機能するプロ

セス) は TPENQUEUE(3cbl) を呼び出し、TMQUEUE(5) サーバを通して利用できるキュー・スペースを指定して、メッセージをキューに登録しています。クライアントは、その後、TMQUEUE への TPDEQUEUE(3cbl) 呼び出しを介して、応答を取得します。

図では、キューに入れられたメッセージが、サーバ TMQFORWARD(5) によってキューから取り出され、処理のために TPCALL(3cbl) を介してアプリケーション・サーバに送信されています。TPCALL に対する応答が受信されると、TMQFORWARD は応答メッセージをキューに登録します。TMQFORWARD は、キュー・スペースと既存のアプリケーション・サービスとの間にインターフェイスを提供するので、アプリケーションにコードを追加する必要はありません。そのため、ここでは、クライアントとキュー・スペースとの間の処理を中心に説明します。

基本的なモデルについて説明し、その後でカスタマイズの例をいくつか示します。

メッセージのキューへの登録

次は、TPENQUEUE() の構文です。

```
01 TPQUEDEF-REC.
   COPY TPQUEDEF.
01 TPTYPE-REC.
   COPY TPTYPE.
01 DATA-REC.
   COPY User Data.
01 TPSTATUS-REC.
   COPY TPSTATUS.
CALL "TPENQUEUE" USING TPQUEDEF-REC TPTYPE-REC DATA-REC TPSTATUS-REC.
```

TPENQUEUE() が呼び出されると、*TPQUEDEF-REC* 内の QSPACE-NAME で識別されるキュー・スペース内の *TPQUEDEF-REC* の QNAME キューにメッセージを格納するようにシステムに指示します。メッセージは *DATA-REC* にあり、*TPTYPE-REC* 内の LEN にはメッセージの長さが入ります。*TPQUEDEF-REC* の設定によって、システムに TPENQUEUE() 呼び出しの処理方法が通知されます。キューに登録されるメッセージ、および応答の処理方法についてのさらに詳しい情報は、*TPQUEDEF-REC* 構造体で定義されます。

TPENQUEUE() の引数

TPENQUEUE(3cbl) の処理を制御するいくつかの重要な引数があります。その一部について、以下に説明します。

TPENQUEUE():TPQUEDEF-REC 引数内の QSPACE-NAME

QSPACE-NAME は、管理者によって既に作成されたキュー・スペースを識別します。サーバがコンフィギュレーション・ファイルの SERVERS セクションで定義されている場合、そのサーバが提供するサービス名は、実際のキュー・スペース名 (GROUPS セクションの OPENINFO パラメータの一部として指定されます) のエイリアスになります。たとえば、アプリケーションがサーバ TMQUEUE を使用する場合、QSPACE-NAME 引数が指す値は、TMQUEUE が宣言するサービス名になります。サービスのエイリアスが何も定義されていない場合、デフォルトのサービス名はサーバ名 TMQUEUE と同じになります。その場合、コンフィギュレーション・ファイルには次の記述されています。

```
TMQUEUE
        SRVGRP = QUE1  SRVID = 1
        GRACE = 0  RESTART = Y CONV = N
        CLOPT = "-A"
または
        CLOPT = "-s TMQUEUE"
```

サーバ・グループ QUE1 のエントリには、OPENINFO パラメータを使用して、リソース・マネージャ、デバイスのパス名、およびキュー・スペース名を指定します。クライアント・プログラムにおける QSPACE-NAME 引数は、次のように記述されます。

```
 01  TPQUEDEF-REC.
     COPY TPQUEDEF.
 01  TPTYPE-REC.
     COPY TPTYPE.
 01  TPSTATUS-REC.
     COPY TPSTATUS.
 01  USER-DATA-REC  PIC X(100).
*
*
*
 MOVE LOW-VALUES TO TPQUEDEF-REC.
 MOVE "TMQUEUE" TO QSPACE-NAME IN TPQUEDEF-REC.
 MOVE "STRING" TO QNAME IN TPQUEDEF-REC.
```

```
SET TPTRAN IN TPQUEDEF-REC TO TRUE.
SET TPBLOCK IN TPQUEDEF-REC TO TRUE.
SET TPTIME IN TPQUEDEF-REC TO TRUE.
SET TPSIGRSTRT IN TPQUEDEF-REC TO TRUE.
MOVE LOW-VALUES TO TPTYPE-REC.
MOVE "STRING" TO REC-TYPE IN TPTYPE-REC.
MOVE LENGTH OF USER-DATA-REC TO LEN IN TPTYPE-REC.
CALL "TPENQUEUE" USING
        TPQUEDEF-REC
        TPTYPE-REC
        USER-DATA-REC
        TPSTATUS-REC.
```

TMQUEUE(5) リファレンス・ページの例では、サーバを作成してコンフィギュレーション・ファイルで指定する際に、サービスのエイリアスを指定する方法が示されています。A-1 ページの「サンプル・アプリケーション」のサンプル・プログラムでも、サービスのエイリアスが指定されています。

TPENQUEUE():TPQUEDEF-REC 引数内の QNAME

キュー・スペースで、メッセージ・キューを使用してサービスが呼び出されている場合、メッセージ・キューは、要求の処理に利用できるアプリケーション・サービスに従って名前が付けられます。QNAME には、そのような値が含まれます。QNAME がアプリケーション・サービスではない場合の例外については、4-33 ページの「TMQFORWARD を通して呼び出されたサービスからの応答をキューから取り出す手順」で説明します。

TPENQUEUE():DATA-REC および TPTYPE-REC 引数内の LEN

DATA-REC には、処理対象のメッセージが入ります。*TPTYPE-REC* 内の LEN は、メッセージの長さを指定します。BEA Tuxedo のレコード・タイプには、LEN を指定する必要がないもの (VIEW など) もあります。その場合、引数は無視されます。*TPTYPE-REC* 内の RECTYPE が SPACES の場合、*DATA-REC* および LEN は無視され、メッセージはデータ部なしでキューに登録されます。

TPENQUEUE():TPQUEDEF-REC の設定値

TPQUEDEF-REC の設定値は、TPENQUEUE() 呼び出しの処理方法を BEA Tuxedo システムに通知するために使用されます。次は、有効な設定です。

TPNOTRAN

呼び出し元がトランザクション・モードにあり、この設定が使用されていると、メッセージは呼び出し元と同じトランザクション内ではキューに登録されません。この値が設定されたトランザクション・モードの呼び出し元には、トランザクション・タイムアウトが適用されます。それ以外は適用されません。この設定を使用した状態で呼び出されたキューへのメッセージの登録が失敗した場合、呼び出し元のトランザクションは影響されません。TPNOTRAN または TPTRAN のいずれかを設定しなければなりません。

TPTRAN

呼び出し元がトランザクション・モードにある場合、この設定はキューへのメッセージの登録が同じトランザクション内で行われることを指定します。TPNOTRAN または TPTRAN のいずれかを設定しなければなりません。

TPNOBLOCK

ブロッキング状態が存在する場合、メッセージはキューに登録されません。TPNOBLOCK が設定されている場合に、メッセージの転送先である内部バッファがいっぱいであるなどのブロッキング状態が存在すると、呼び出しは失敗し、tperrno(5) に TPEBLOCK が設定されます。TPNOBLOCK が設定されている場合に、ターゲットのキューが別のアプリケーションによって排他的にオープンされているというブロッキング状態が存在すると、呼び出しは失敗し、tperrno() に TPEDIAGNOSTIC が設定され、TPQCTL 構造体の診断フィールドに QMESHARE が設定されます。後者の場合、BEA Tuxedo システム以外の BEA 製品に基づくほかのアプリケーションが、キューイング・サービス API (QSAPI) を使用して読み取りと書き込み、またはそのいずれかを排他的に行うためにキューをオープンしています。TPNOBLOCK または TPBLOCK のいずれかを設定しなければなりません。

TPBLOCK

TPBLOCK が設定されている場合に、ブロッキング状態が存在すると、その状態が解消されるかタイムアウト (トランザクション・タイムアウトまたはブロッキング・タイムアウト) が発生するまで、呼び出し元はブロックされます。TPNOBLOCK または TPBLOCK のいずれかを設定しなければなりません。

TPNOTIME
: この設定は、呼び出しにブロッキング・タイムアウトが適用されないことを示します。ただし、トランザクション・タイムアウトは発生する可能性があります。TPNOTIME または TPTIME のいずれかを設定しなければなりません。

TPTIME
: この設定は、呼び出しにブロッキング・タイムアウトが適用されることを示します。TPNOTIME または TPTIME のいずれかを設定しなければなりません。

TPSIGRSTRT
: この設定は、基となるシステム・コールがシグナルによって中断された場合、中断されたシステム・コールが再度呼び出されることを指定します。TPSIGRSTRT または TPNOSIGRSTRT のいずれかを設定しなければなりません。

TPNOSIGRSTRT
: この設定は、基となるシステム・コールがシグナルによって中断された場合、中断されたシステム・コールが再度呼び出されないことを指定します。呼び出しは失敗し、TP-STATUS に TPEGOTSIG が設定されます。TPSIGRSTRT または TPNOSIGRSTRT のいずれかを設定しなければなりません。

TPQUEDEF-REC 構造体

TPQUEDEF-REC 構造体には、アプリケーションで使用されるメンバと BEA Tuxedo システムで使用されるメンバがあり、アプリケーション・プログラムとキュー機能間の両方向でパラメータがやり取りされます。この構造体は、COBOL COPY ファイル内で定義されています。*TPQUEDEF-REC* を呼び出すクライアントは、設定値を使用して、システム側で入力する必要のあるフィールドをマークします。この構造体は、TPDEQUEUE() でも使用されます。一部のメンバは、アプリケーションが TPDEQUEUE() を呼び出すまで使用されません。次のコード例は、この構造体全体を示しています。

コード リスト 4-1 TPQUEDEF-REC **構造体**

```
05 TPBLOCK-FLAG         PIC S9(9) COMP-5.
        88 TPBLOCK              VALUE 0.
        88 TPNOBLOCK            VALUE 1.
05 TPTRAN-FLAG  PIC S9(9) COMP-5.
        88 TPTRAN               VALUE 0.
        88 TPNOTRAN             VALUE 1.
05 TPTIME-FLAG  PIC S9(9) COMP-5.
        88 TPTIME               VALUE 0.
        88 TPNOTIME             VALUE 1.
05 TPSIGRSTRT-FLAG      PIC S9(9) COMP-5.
        88 TPNOSIGRSTRT         VALUE 0.
        88 TPSIGRSTRT           VALUE 1.
05 TPNOCHANGE-FLAG      PIC S9(9) COMP-5.
        88 TPCHANGE             VALUE 0.
        88 TPNOCHANGE           VALUE 1.
05 TPQUE-ORDER-FLAG     PIC S9(9) COMP-5.
        88 TPQDEFAULT           VALUE 0.
        88 TPQTOP               VALUE 1.
        88 TPQBEFOREMSGID       VALUE 2.
05 TPQUE-TIME-FLAG      PIC S9(9) COMP-5.
        88 TPQNOTIME            VALUE 0.
        88 TPQTIME-ABS          VALUE 1.
        88 TPQTIME-REL          VALUE 2.
05 TPQUE-PRIORITY-FLAG  PIC S9(9) COMP-5.
        88 TPQNOPRIORITY        VALUE 0.
        88 TPQPRIORITY          VALUE 1.
05 TPQUE-CORRID-FLAG    PIC S9(9) COMP-5.
        88 TPQNOCORRID          VALUE 0.
        88 TPQCORRID            VALUE 1.
05 TPQUE-REPLYQ-FLAG    PIC S9(9) COMP-5.
        88 TPQNOREPLYQ          VALUE 0.
        88 TPQREPLYQ            VALUE 1.
05 TPQUE-FAILQ-FLAG     PIC S9(9) COMP-5.
        88 TPQNOFAILUREQ        VALUE 0.
        88 TPQFAILUREQ          VALUE 1.
05 TPQUE-MSGID-FLAG     PIC S9(9) COMP-5.
        88 TPQNOMSGID           VALUE 0.
        88 TPQMSGID             VALUE 1.
05 TPQUE-GETBY-FLAG     PIC S9(9) COMP-5.
        88 TPQGETNEXT           VALUE 0.
        88 TPQGETBYMSGIDOLD     VALUE 1.
        88 TPQGETBYCORRIDOLD    VALUE 2.
        88 TPQGETBYMSGID        VALUE 3.
        88 TPQGETBYCORRID       VALUE 4.
05 TPQUE-WAIT-FLAG      PIC S9(9) COMP-5.
        88 TPQNOWAIT            VALUE 0.
```

```
        88 TPQWAIT              VALUE 1.
05 TPQUE-DELIVERY-FLAG  PIC S9(9) COMP-5.
        88 TPQNODELIVERYQOS     VALUE 0.
        88 TPQDELIVERYQOS       VALUE 1.
05 TPQUEQOS-DELIVERY-FLAG       PIC S9(9) COMP-5.
        88 TPQQOSDELIVERYDEFAULTPERSIST VALUE 0.
        88 TPQQOSDELIVERYPERSISTENT     VALUE 1.
        88 TPQQOSDELIVERYNONPERSISTENT  VALUE 2.
05 TPQUE-REPLY-FLAG     PIC S9(9) COMP-5.
        88 TPQNOREPLYQOS                VALUE 0.
        88 TPQREPLYQOS                  VALUE 1.
05 TPQUEQOS-REPLY-FLAG  PIC S9(9) COMP-5.
        88 TPQQOSREPLYDEFAULTPERSIST    VALUE 0.
        88 TPQQOSREPLYPERSISTENT        VALUE 1.
        88 TPQQOSREPLYNONPERSISTENT     VALUE 2.
05 TPQUE-EXPTIME-FLAG   PIC S9(9) COMP-5.
        88 TPQNOEXPTIME         VALUE 0.
        88 TPQEXPTIME-ABS       VALUE 1.
        88 TPQEXPTIME-REL       VALUE 2.
        88 TPQEXPTIME-NONE      VALUE 3.
05 TPQUE-PEEK-FLAG      PIC S9(9) COMP-5.
        88 TPQNOPEEK            VALUE 0.
        88 TPQPEEK              VALUE 1.
05 DIAGNOSTIC           PIC S9(9) COMP-5.
        88 QMEINVAL             VALUE -1.
        88 QMEBADRMID           VALUE -2.
        88 QMENOTOPEN           VALUE -3.
        88 QMETRAN              VALUE -4.


        88 QMEBADMSGID          VALUE -5.
        88 QMESYSTEM            VALUE -6.
        88 QMEOS                VALUE -7.
        88 QMEABORTED           VALUE -8.
        88 QMEPROTO             VALUE -9.
        88 QMEBADQUEUE          VALUE -10.
        88 QMENOMSG             VALUE -11.
        88 QMEINUSE             VALUE -12.
        88 QMENOSPACE           VALUE -13.
        88 QMERELEASE           VALUE -14.
        88 QMEINVHANDLE         VALUE -15.
        88 QMESHARE             VALUE -16.
05 DEQ-TIME             PIC 9(9) COMP-5.
05 EXP-TIME             PIC 9(9) COMP-5.
05 PRIORITY             PIC S9(9) COMP-5.
05 MSGID                PIC X(32).
05 CORRID               PIC X(32).
05 QNAME                PIC X(15).
05 QSPACE-NAME          PIC X(15).
```

```
05 REPLYQUEUE           PIC X(15).
05 FAILUREQUEUE         PIC X(15).
05 CLIENTID OCCURS 4 TIMES PIC S9(9) COMP-5.
05 APPL-RETURN-CODE     PIC S9(9) COMP-5.
05 APPKEY               PIC S9(9) COMP-5.
```

次は、TPENQUEUE の入力情報を制御するパラメータの有効な設定です。

TPQTOP

この値を設定すると、キューの順序付けは無効になり、メッセージはキューの先頭に登録されます。この要求は、順序付けを無効にするようにキューが設定されているかどうかによって、使用できない場合があります。デフォルトのキューの順序を使用する場合は、TPQDEFAULT を設定します。TPQTOP、TPQBEFOREMSGID、TPQDEFAULT のいずれかを設定しなければなりません。

TPQBEFOREMSGID

この値を設定すると、キューの順序付けが無効になり、メッセージは MSGID によって識別されるメッセージの前に登録されます。この要求は、順序付けを無効にするようにキューが設定されているかどうかによって、使用できない場合があります。デフォルトのキューの順序を使用する場合は、TPQDEFAULT を設定します。TPQTOP、TPQBEFOREMSGID、TPQDEFAULT のいずれかを設定しなければなりません。

メッセージ識別子の値は 32 バイト全体が意味を持つので、MSGID で識別される値は、たとえば空白を埋め込むなどして、完全に初期化する必要があります。

TPQTIME-ABS

この値が設定されていると、DEQ-TIME で指定された時間の経過後、メッセージが処理されます。DEQ-TIME は、time(2) または mktime(3C) によって生成された絶対時間値、つまり世界協定時 (UTC) 1970 年 1 月 1 日 00:00:00 から経過した秒数を示します。絶対時間値も相対時間値も設定されていない場合は、TPQNOTIME を設定します。TPQTIME-ABS、TPQTIME-REL、TPQNOTIME のいずれかを設定しなければなりません。絶対時間は、キュー・マネージャ・プロセスが存在するマシン・クロックによって決定されます。

TPQTIME-REL

この値が設定されていると、キューへの登録が完了してからの相対時間の経過後にメッセージが処理されます。DEQ-TIME は、キューへの登録が完了した後、送信されたメッセージが処理されるまでの遅延秒数を指定します。絶対時間値も相対時間値も設定されていない場合は、TPQNOTIME を設定します。TPQTIME-ABS、TPQTIME-REL、TPQNOTIME のいずれかを設定しなければなりません。

TPQPRIORITY

この値が設定されていると、メッセージがキューに登録されるときの優先順位が PRIORITY に格納されます。優先順位は、1 以上 100 以下の範囲でなければなりません。数値が高いほど優先順位も高くなり、高い数値のメッセージが低い数値のメッセージより先にキューから取り出されます。優先順位によって順序付けられていないキューでは、この値は参考として使用されます。TPQNOPRIORITY が設定されている場合、デフォルトでメッセージの優先順位が 50 になります。

TPQCORRID

この値が設定されていると、メッセージが TPDEQUEUE() によってキューから取り出されるときに、CORRID に指定された相関識別子の値が使用されます。この識別子は、キューに登録されたすべての応答メッセージまたは異常終了メッセージに付加されるので、アプリケーションは応答を特定の要求に結び付けることができます。相関識別子を使用できない場合は、TPQNOCORRID を設定します。

相関識別子の値は 32 バイト全体が意味を持つので、CORRID に指定される値は、たとえば空白を埋め込むなどして、完全に初期化する必要があります。

TPQREPLYQ

この値が設定されていると、REPLYQUEUE に指定された応答キューが、キューに入れられたメッセージに対応付けられます。メッセージへの応答はすべて、要求メッセージと同じキュー・スペース内の、指定されたキューに登録されます。応答キューの名前を使用できない場合は、TPQNOREPLYQ を設定します。

TPQFAILUREQ

この値が設定されていると、FAILUREQUEUE に指定された異常終了キューが、キューに入れられたメッセージに対応付けられます。(1)

キューに登録されたメッセージが TMQFORWARD() によって処理され、(2) TMQFORWARD が -d オプションで開始され、さらに (3) サービスが異常終了して NULL 以外の応答を返す場合は、その応答と関連する tpurcode によって構成される異常終了メッセージが、元の要求メッセージと同じキュー・スペース内で指定されたキューに登録されます。異常終了キューの名前を使用できない場合は、TPQNOFAILUREQ を設定します。

TPQDELIVERYQOS
TPQREPLYQOS

TPQDELIVERYQOS が設定されていると、TPQUEQOS-DELIVERY-FLAG で指定されたフラグが、メッセージの配信に対するサービスの品質を制御します。相互に排他的な次のフラグ、TPQQOSDELIVERYDEFAULTPERSIST、TPQQOSDELIVERYPERSISTENT、または TPQQOSDELIVERYNONPERSISTENT のいずれかを設定する必要があります。TPQDELIVERYQOS が設定されていない場合は、TPQNODELIVERYQOS を設定する必要があります。TPQNODELIVERYQOS が設定されている場合、ターゲットのキューのデフォルトの配信方針がメッセージに対するサービスの配信品質を指定します。

TPQREPLYQOS が設定されていると、TPQUEQOS-REPLY-FLAG で指定されるフラグが、応答メッセージの配信に対するサービスの品質を制御します。相互に排他的な次のフラグ、TPQQOSREPLYDEFAULTPERSIST、TPQQOSREPLYPERSISTENT、または TPQQOSREPLYNONPERSISTENT のいずれかを設定する必要があります。TPQREPLYQOS フラグは、TMQFORWARD によって処理されるメッセージから応答が返されるときに使用されます。サービスを呼び出すときに TMQFORWARD を使用しないアプリケーションでは、自身の応答メカニズムのヒントとして TPQREPLYQOS フラグを使用できます。

TPQREPLYQOS が設定されていない場合は、TPQNOREPLYQOS を設定する必要があります。TPQNOREPLYQOS が設定されている場合、REPLYQUEUE キューのデフォルトの配信方針が応答に対するサービスの配信品質を指定します。デフォルトの配信方針は、メッセージに対する応答がキューに登録されるときに決定される点に注意してください。つまり、元のメッセージがキューに登録されてから応答が登録されるまでの間に、応答キューのデフォルトの配信方針が変更された場合、応答が最後に登録される時点で有効な方針が使用されます。

次は、有効な TPQUEQOS-DELIVERY-FLAG および
TPQUEQOS-REPLY-FLAG のフラグです。

TPQQOSDELIVERYDEFAULTPERSIST
TPQQOSREPLYDEFAULTPERSIST
: このフラグは、ターゲットまたは応答のキューで指定されているデフォルトの配信方針を使用して、メッセージが配信されるように指定します。

TPQQOSDELIVERYPERSISTENT
TPQQOSREPLYPERSISTENT
: このフラグは、メッセージがディスク・ベースの配信方法を使用して、永続的な記憶域に配信されることを指定します。このフラグが設定されていると、ターゲットまたは応答のキューに指定されたデフォルトの配信方針がこのフラグで上書きされます。

TPQQOSDELIVERYNONPERSISTENT
TPQQOSREPLYNONPERSISTENT
: このフラグは、メッセージがメモリ・ベースの配信方法を使用して、非永続的な記憶域に配信されることを指定します。メッセージは、キューから取り出されるまでメモリに登録されたままになります。このフラグが設定されていると、ターゲットまたは応答のキューに指定されたデフォルトの配信方針がこのフラグで上書きされます。

  呼び出し元がトランザクション・モードの場合、一時的メッセージは呼び出し元のトランザクション内でキューに登録されます。ただし、システムがシャットダウンしたりクラッシュしたりした場合、またはキュー・スペースとしての IPC 共用メモリが除去された場合は、一時的メッセージは失われます。

TPQEXPTIME-ABS
: この値が設定されていると、メッセージに有効期限の絶対時間が適用されます。これは、キューからメッセージが削除される絶対時間です。
有効期限の絶対時間は、キュー・マネージャ・プロセスが存在するマシン・クロックによって決定されます。

  有効期限の絶対時間は、EXP-TIME に格納された値で示されます。
  EXP-TIME は、time(2) または mktime(3C) によって生成された絶対時

間に設定されなければなりません (世界協定時 (UTC) 1970 年 1 月 1 日 00:00:00 から経過した秒数)。

キューへの登録操作の時間より早い絶対時間が指定されると、操作は成功しますが、メッセージはしきい値の計算の対象になりません。有効期限の時間がメッセージの使用可能時間より前の場合、使用可能時間が有効期限の切れる時間より前になるようにいずれかの時間を変更しない限り、メッセージをキューから取り出すことはできません。また、これらのメッセージがキューからの取り出しの対象になったことがなくても、有効期限が切れるとキューから削除されます。トランザクション中にメッセージの期限が切れてもトランザクションは失敗しません。トランザクション内でキューへの登録、またはキューからの取り出し中に有効期限が切れたメッセージは、トランザクションが終了した時点でキューから削除されます。メッセージの有効期限が切れたことの通知は行われません。

TPQEXPTIME-ABS、TPQEXPTIME-REL、TPQEXPTIME-NONE、TPQNOEXPTIME のいずれかを設定しなければなりません。

TPQEXPTIME-REL

この値が設定されていると、メッセージに有効期限の相対時間が適用されます。これは、メッセージがキューに到達してから、キューから削除されるまでの秒数です。有効期限の相対時間は、EXP-TIME に格納された値で示されます。

有効期限の時間がメッセージの使用可能時間より前の場合、使用可能時間が有効期限の切れる時間より前になるようにいずれかの時間を変更しない限り、メッセージをキューから取り出すことはできません。また、これらのメッセージがキューからの取り出しの対象になったことがなくても、有効期限が切れるとキューから削除されます。メッセージがトランザクション内にあるときに期限切れになった場合、それによってトランザクションが異常終了することはありません。トランザクション内でキューへの登録、またはキューからの取り出し中に有効期限が切れたメッセージは、トランザクションが終了した時点でキューから削除されます。メッセージの有効期限が切れたことの通知は行われません。

TPQEXPTIME-ABS、TPQEXPTIME-REL、TPQEXPTIME-NONE、TPQNOEXPTIME のいずれかを設定しなければなりません。

TPQEXPTIME-NONE

この値が設定されていると、メッセージの有効期限が無期限になります。このフラグは、ターゲットのキューに対応付けられているデフォルトの有効期限の方針を上書きします。メッセージを削除するには、管理インターフェイスを介してキューから取り出すか、または削除します。TPQEXPTIME-ABS、TPQEXPTIME-REL、TPQEXPTIME-NONE、TPQNOEXPTIME のいずれかを設定しなければなりません。

TPQNOEXPTIME

この値が設定されていると、ターゲットのキューに対応付けられているデフォルトの有効期限の時間がメッセージに適用されます。TPQEXPTIME-ABS、TPQEXPTIME-REL、TPQEXPTIME-NONE、TPQNOEXPTIME のいずれかを設定しなければなりません。

このほかに、*TPQUEDEF-REC* のメンバ APPL-RETURN-CODE にユーザ戻りコードを設定することができます。この値は、メッセージをキューから取り出すために TPDEQUEUE() を呼び出すアプリケーションに返されます。

TPENQUEUE() からの出力では、次のフィールドが TPQUEDEF-REC 構造体に設定されます。

```
05 DIAGNOSTIC           PIC S9(9) COMP-5.
05 MSGID                PIC X(32).
```

次は、TPENQUEUE()() からの出力情報を制御する *TPQUEDEF-REC* の有効な設定です。TPENQUEUE() の呼び出し時にこの値が設定されている場合、BEA Tuxedo /Q サーバ TMQUEUE(5) は、レコード内の対応する要素にメッセージ識別子を挿入します。TPENQUEUE() の呼び出し時にこの値が設定されていない場合、TMQUEUE() は、レコード内の対応する要素にメッセージ識別子を挿入しません。

TPQMSGID

この値が設定されている場合に、TPENQUEUE() の呼び出しが成功すると、メッセージ識別子が MSGID に格納されます。メッセージ識別子の値は 32 バイト全体が意味を持つので、MSGID に格納される値は、たとえば NULL 文字を埋め込むなどして、完全に初期化する必要があります。初期化に使用される実際の埋め込み文字は、BEA Tuxedo /Q コンポーネントのリリースによって異なります。TPQNOMSGID が設定されている場合、メッセージ識別子は使用できません。

制御構造体の残りのメンバは、TPENQUEUE() への入力では使用されません。

TPENQUEUE() の呼び出しが失敗し、TP-STATUS に TPEDIAGNOSTIC が設定された場合、失敗の原因を示す値が DIAGNOSTIC に返されます。次は、返される値です。

[QMEINVAL]
無効な設定値が指定されています。

[QMEBADRMID]
無効なリソース・マネージャ識別子が指定されています。

[QMENOTOPEN]
リソース・マネージャが現在オープンされていません。

[QMETRAN]
呼び出しがトランザクション・モードではないか、または TPNOTRAN を指定して呼び出しが行われたため、キューにメッセージを登録するトランザクションを開始したときに、エラーが発生しました。この診断は、BEA Tuxedo リリース 7.1 以降のキュー・マネージャでは返されません。

[QMEBADMSGID]
無効なメッセージ識別子が指定されています。

[QMESYSTEM]
システム・エラーが発生しました。エラーの正確な内容がログ・ファイルに書き込まれます。

[QMEOS]
オペレーティング・システムのエラーが発生しました。

[QMEABORTED]
操作がアボートされました。アボートされた操作がグローバル・トランザクション内で実行されていた場合、グローバル・トランザクションは「ロールバックのみ」とマークされます。それ以外の場合、キュー・マネージャは操作をアボートします。

[QMEPROTO]
トランザクションの状態がアクティブではないときに、キューへの登録が行われました。

[QMEBADQUEUE]
無効または削除されたキューの名前が指定されています。

[QMENOSPACE]
キュー上に領域がないなどリソース不足が原因で、サービスの品質 (永続的な記憶域または非永続的な記憶域) が指定されたメッセージがキューに登録されませんでした。次のいずれかのリソース設定を超えると、QMENOSPACE が返されます。(1) キュー・スペースに割り当てられたディスク容量 (永続的)、(2) キュー・スペースに割り当てられたメモリ容量 (非永続的)、(3) 同時にアクティブ状態になるトランザクションの最大数 (キュー・スペースで許容される数であることが必要です)、(4) キュー・スペースに一度に入れることができる最大メッセージ数、(5) キューイング・サービス・コンポーネントが処理できる並列アクションの最大数、または (6) キューイング・サービス・コンポーネントを同時に使用できる認証されたユーザの最大数。

[QMERELEASE]
新機能がサポートされていないバージョンの BEA Tuxedo システムのキュー・マネージャに対して、メッセージのキューへの登録が試みられました。

[QMESHARE]
指定されたキューのメッセージの登録時に、そのキューが別のアプリケーションによって 排他的にオープンされています。別のアプリケーションとは、BEA Tuxedo システム以外の BEA 製品に基づくもので、キューをオープンして、キューイング・サービス API (QSAPI) を使用して読み取りおよび書き込み、またはそのいずれかを排他的に行っています。

キューの順序の無効化

キューの作成時に、管理者が TPENQUEUE() 呼び出しでキュー上のメッセージの順序を無効にできるようにした場合、次の 2 つの方法でこの無効機能を利用できます。この 2 つの方法は、相互に排他的です。TPQTOP を設定すると、メッセージをキューの先頭に置くことができます。または、TPQBEFOREMSGID、MSGID に既存メッセージの ID を設定して、メッセージを特定の既存のメッセージの前に置くこともできます。この方法では、メッセージ ID を使用できるように、以前の呼び出しで取得されたメッセージ ID

が保存されていることが必要です。管理者は、キューでサポートされている方法を通知する必要があります。キューは、この 2 つのいずれかまたは両方を使用できるように、あるいはどちらも使用できないように作成できます。

キューの優先順位の無効化

PRIORITY に値を設定して、メッセージの優先順位を指定することができます。この値は、1 以上 100 以下の範囲でなければなりません。数値が高いほど優先順位も高くなります。つまり、UNIX の nice コマンドで指定される値とは異なります。キューの順序付けパラメータの中に PRIORITY が含まれていない場合、ここで優先順位を設定しても取り出しの順序には影響しません。ただし、優先順位の値は保持されるので、メッセージがキューから取り出されるときに検査されます。

メッセージの使用可能時間の設定

DEQ-TIME に、絶対時間またはキューへの登録が完了してからの相対時間として、メッセージが処理されるまで時間を指定できます。TPQTIME-ABS または TPQTIME-REL のいずれかを設定して、値の処理方法を指定できます。キューは、time を順序付けの基準として作成することができます。その場合、メッセージは使用可能時間によって順序付けされます。

次のコード例は、相対時間を使用して、メッセージをキューに登録する方法を示しています。このメッセージは、60 秒後に処理対象になります。

```
  01  TPQUEDEF-REC.
      COPY TPQUEDEF.
  01  TPTYPE-REC.
      COPY TPTYPE.
  01  TPSTATUS-REC.
      COPY TPSTATUS.
  01  USER-DATA-REC  PIC X(100).
*
*
*
  MOVE LOW-VALUES TO TPQUEDEF-REC.
  MOVE "QSPACE1" TO QSPACE-NAME IN TPQUEDEF-REC.
  MOVE "Q1" TO QNAME IN TPQUEDEF-REC.
  SET TPTRAN IN TPQUEDEF-REC TO TRUE.
  SET TPBLOCK IN TPQUEDEF-REC TO TRUE.
```

```
SET TPTIME IN TPQUEDEF-REC TO TRUE.
SET TPSIGRSTRT IN TPQUEDEF-REC TO TRUE.
SET TPQDEFAULT IN TPQUEDEF-REC TO TRUE.
SET TPQTIME-REL IN TPQUEDEF-REC TO TRUE.
MOVE 60 TO DEQ-TIME IN TPQUEDEF-REC.
SET TPQNOPRIORITY IN TPQUEDEF-REC TO TRUE.
SET TPQNOCORRID IN TPQUEDEF-REC TO TRUE.
SET TPQNOREPLYQ IN TPQUEDEF-REC TO TRUE.
SET TPQNOFAILUREQ IN TPQUEDEF-REC TO TRUE.
SET TPQMSGID IN TPQUEDEF-REC TO TRUE.
MOVE LOW-VALUES TO TPTYPE-REC.
MOVE "STRING" TO REC-TYPE IN TPTYPE-REC.
MOVE LENGTH OF USER-DATA-REC TO LEN IN TPTYPE-REC.
CALL "TPENQUEUE" USING
        TPQUEDEF-REC
        TPTYPE-REC
        USER-DATA-REC
        TPSTATUS-REC.
```

TPENQUEUE() およびトランザクション

TPENQUEUE() の呼び出し元がトランザクション・モードにある場合に、TPTRAN が設定されていると、キューへの登録は呼び出し元のトランザクション内で行われます。呼び出し元は、TPENQUEUE() が成功したか失敗したかによって、メッセージがキューに登録されたかどうかを判断できます。呼び出しが正常に行われると、メッセージがキューに登録されたことが保証されます。呼び出しが失敗すると、メッセージがキューに登録された部分も含めて、トランザクションがロールバックされます。

TPENQUEUE() の呼び出し元がトランザクション・モードにない場合、または TPNOTRAN が設定されている場合、メッセージは呼び出し元のトランザクションとは別のトランザクションでキューに登録されます。tpenqueue() への呼び出しが正常な戻り値を返した場合、メッセージがキューに登録されたことが保証されます。TPENQUEUE() の呼び出しが通信エラーまたはタイムアウトによって失敗した場合は、その障害がメッセージ登録の前に発生したのか後に発生したのか、呼び出し元には判断できません。

呼び出し元がトランザクション・モードにないときに TPNOTRAN を指定しても意味がありません。

メッセージのキューからの取り出し

次は、TPDEQUEUE() の構文です。

```
01 TPQUEDEF-REC.
   COPY TPQUEDEF.
01 TPTYPE-REC.
   COPY TPTYPE.
01 DATA-REC.
   COPY User Data.
01 TPSTATUS-REC.
   COPY TPSTATUS.
CALL "TPDEQUEUE" USING TPQUEDEF-REC TPTYPE-REC DATA-REC TPSTATUS-REC.
```

この呼び出しが行われると、TPQUEDEF-REC 内の QSPACE-NAME で指定されたキュー・スペースの TPQUEDEF-REC の QNAME キューからメッセージを取り出すようにシステムが指示されます。メッセージは、DATA-REC に挿入されます。TPTYPE-REC の LEN にデータ長が設定されます。TPDEQUEUE() から返された LEN が 0 の場合、そのメッセージにはデータ部がないことを示します。TPQUEDEF-REC の設定によって、システムに TPDEQUEUE() 呼び出しの処理方法が通知されます。

TPDEQUEUE() の引数

TPDEQUEUE(3cbl) の処理を制御するいくつかの重要な引数があります。その一部について、以下に説明します。

TPDEQUEUE():TPQUEDEF-REC 引数内の QSPACE-NAME

QSPACE-NAME は、管理者によって既に作成されたキュー・スペースを識別します。TMQUEUE サーバがコンフィギュレーション・ファイルの SERVERS セクションで定義されている場合、そのサーバが提供するサービス名は、実際のキュー・スペース名 (GROUPS セクションの OPENINFO パラメータの一部とし

て指定されます) のエイリアスになります。たとえば、アプリケーションがサーバ TMQUEUE を使用する場合、QSPACE-NAME 引数が指す値は、TMQUEUE が宣言するサービス名になります。サービスのエイリアスが何も定義されていない場合、デフォルトのサービス名は TMQUEUE サーバと同じエイリアスになります。その場合、コンフィギュレーション・ファイルには次の内容が記述されています。

```
TMQUEUE
        SRVGRP = QUE1  SRVID = 1
        GRACE = 0  RESTART = Y CONV = N
        CLOPT = "-A"
または
        CLOPT = "-s TMQUEUE"
```

サーバ・グループ QUE1 のエントリには、OPENINFO パラメータを使用して、リソース・マネージャ、デバイスのパス名、およびキュー・スペース名を指定します。クライアント・プログラムにおける QSPACE-NAME 引数は、次のように記述されます。

```
  01 TPQUEDEF-REC.
     COPY TPQUEDEF.
  01 TPTYPE-REC.
     COPY TPTYPE.
  01 TPSTATUS-REC.
     COPY TPSTATUS.
  01 USER-DATA-REC  PIC X(100).
*
*
*
  MOVE LOW-VALUES TO TPQUEDEF-REC.
  MOVE "TMQUEUE" TO QSPACE-NAME IN TPQUEDEF-REC.
  MOVE "REPLYQ" TO QNAME IN TPQUEDEF-REC.
  SET TPTRAN IN TPQUEDEF-REC TO TRUE.
  SET TPBLOCK IN TPQUEDEF-REC TO TRUE.
  SET TPTIME IN TPQUEDEF-REC TO TRUE.
  SET TPSIGRSTRT IN TPQUEDEF-REC TO TRUE.
  MOVE LOW-VALUES TO TPTYPE-REC.
  MOVE "STRING" TO REC-TYPE IN TPTYPE-REC.
  MOVE LENGTH OF USER-DATA-REC TO LEN IN TPTYPE-REC.
  CALL "TPDEQUEUE" USING
          TPQUEDEF-REC
          TPTYPE-REC
          USER-DATA-REC
          TPSTATUS-REC.
```

TMQUEUE(5) リファレンス・ページの例では、サーバを作成してコンフィギュレーション・ファイルで指定する際に、サービスのエイリアスを指定する方法が示されています。A-1 ページの「サンプル・アプリケーション」のサンプル・プログラムでも、サービスのエイリアスが指定されています。

TPDEQUEUE():TPQUEDEF-REC 引数内の QNAME

キュー・スペース内のキュー名は、そのキュー・スペースにアクセスするアプリケーション間で一貫していなければなりません。これは、応答キューでは特に重要です。QNAME が応答キューを参照する場合、管理者はほかのキューと同じ方法で応答キュー、そして多くの場合、エラー・キューも作成します。*QNAME* には、メッセージまたは応答を取り出すキューの名前が指定されています。

TPDEQUEUE():DATA-REC および TPTYPE-REC 引数内の LEN

この引数は、TPENQUEUE() で使用される場合と若干意味が異なります。DATA-REC は、キューから取り出されたメッセージをシステムが格納する場所を示します。

入力として LEN に 0 が設定されている場合はエラーになります。TPDEQUEUE() が戻ると、LEN には取り出されたデータの長さが格納されます。0 は応答にデータがなかったことを示します。アプリケーションによっては、これは正当で正常な応答です。長さ 0 の応答を受信した場合でも、それをキューに登録された要求の正常処理を示すために使用できます。レコードが TPDEQUEUE() 呼び出しの前と比べて変更されているかどうかを確認する場合は、TPDEQUEUE() 呼び出しの前にデータ長を保存し、それを呼び出しが終了した後で LEN と比較します。応答が LEN より長い場合、*DATA-REC* にはこのレコードに格納できるだけのバイト数が挿入されます。それを超える部分は破棄され、TPTRUNCATE が設定されて TPDEQUEUE() は失敗します。

TPDEQUEUE():TPQUEDEF-REC の設定値

TPQUEDEF-REC の設定値は、TPDEQUEUE() 呼び出しの処理方法を BEA Tuxedo システムに通知するために使用されます。次は、有効な設定です。

TPNOTRAN

呼び出し元がトランザクション・モードにある場合、この設定はキューからのメッセージの取り出しが別のトランザクション内で行われることを指示します。TPNOTRAN または TPTRAN のいずれかを設定しなければなりません。

TPTRAN

呼び出し元がトランザクション・モードにある場合、この設定はキューからのメッセージの取り出しが同じトランザクション内で行われることを指定します。TPNOTRAN または TPTRAN のいずれかを設定しなければなりません。

TPNOBLOCK

ブロッキング状態が存在する場合、メッセージはキューから取り出されません。TPNOBLOCK が設定されている場合に、メッセージの転送先である内部バッファがいっぱいであるなどのブロッキング状態が存在すると、呼び出しは失敗し、tperrno(5) に TPEBLOCK が設定されます。TPNOBLOCK が設定されている場合に、ターゲットのキューが別のアプリケーションによって排他的にオープンされているというブロッキング状態が存在すると、呼び出しは失敗し、tperrno() に TPEDIAGNOSTIC が設定され、TPQCTL 構造体の診断フィールドに QMESHARE が設定されます。後者の場合、BEA Tuxedo システム以外の BEA 製品に基づくほかのアプリケーションが、キューイング・サービス API (QSAPI) を使用して読み取りと書き込み、またはそのいずれかを排他的に行うためにキューをオープンしています。TPNOBLOCK または TPBLOCK のいずれかを設定しなければなりません。

TPBLOCK

TPBLOCK が設定されている場合に、ブロッキング状態が存在すると、その状態が解消されるかタイムアウト (トランザクション・タイムアウトまたはブロッキング・タイムアウト) が発生するまで、呼び出し元はブロックされます。TPQWAIT が設定されている場合、このブロッキング条件にはキュー自体でのブロッキングは含まれません。TPNOBLOCK または TPBLOCK のいずれかを設定しなければなりません。

TPNOTIME

この値が設定されていると、呼び出しにブロッキング・タイムアウトが適用されないことを示します。ただし、トランザクション・タ

イムアウトは発生する可能性があります。TPNOTIME または TPTIME のいずれかを設定しなければなりません。

TPTIME
: この値が設定されていると、呼び出しにブロッキング・タイムアウトが適用されることを示します。TPNOTIME または TPTIME のいずれかを設定しなければなりません。

TPNOCHANGE
: この値が設定されていると、*DATA-REC* のレコード・タイプは変更できません。つまり、受信したレコードのタイプおよびサブタイプは、レコード *DATA-REC* のタイプおよびサブタイプと一致しなければなりません。TPNOCHANGE または TPCHANGE のいずれかを設定しなければなりません。

TPCHANGE
: デフォルトでは、レコード *DATA-REC* とは異なるタイプのレコードが受信されると、受信側が着信レコードのタイプを識別する限り、*DATA-REC* のレコード・タイプは、受信されたレコードのタイプに変更されます。つまり、受信したレコードのタイプおよびサブタイプは、レコード *DATA-REC* のタイプおよびサブタイプと一致しなければなりません。TPNOCHANGE または TPCHANGE のいずれかを設定しなければなりません。

TPSIGRSTRT
: この値が設定されていると、基となるシステム・コールがシグナルによって中断された場合、中断されたシステム・コールが再度呼び出されることを指定します。TPSIGRSTRT または TPNOSIGRSTRT のいずれかを設定しなければなりません。

TPNOSIGRSTRT
: この値が設定されている場合にシグナルが受信されると、呼び出しは失敗し、TP-STATUS に TPEGOTSIG が設定されます。TPSIGRSTRT または TPNOSIGRSTRT のいずれかを設定しなければなりません。

TPQUEDEF-REC 構造体

TPDEQUEUE() の最初の引数は、*TPQUEDEF-REC* 構造体です。*TPQUEDEF-REC* 構造体には、アプリケーションで使用されるメンバと BEA Tuxedo システムで使用されるメンバがあり、アプリケーション・プログラムとキュー機能間の両方向でパラメータがやり取りされます。TPDEQUEUE() を呼び出すクライアントは、設定値を使用して、システム側で入力する必要のあるメンバをマークします。前述のように、この構造体は TPENQUEUE() でも使用されます。一部のメンバは、TPENQUEUE() だけに適用されます。この構造体全体のコード例は、4-8 ページの「TPQUEDEF-REC 構造体」に示されています。

TPDEQUEUE() への入力では、次のフィールドを TPQUEDEF 構造体に設定します。

```
05 MSGID                PIC X(32).
05 CORRID               PIC X(32).
```

次は、TPDEQUEUE() の入力情報を制御する *TPQUEDEF-REC* の有効な値です。

TPQGETNEXT
: この値が設定されていると、デフォルトのキューの順序を使用して、キュー上にある次のメッセージが取り出されます。TPQGETNEXT、TPQGETBYMSGID、TPQGETBYCORRID のいずれかを設定しなければなりません。

TPQGETBYMSGID
: この値が設定されていると、MSGID で識別されるメッセージが取り出されます。メッセージ識別子は、以前に呼び出された TPENQUEUE() 呼び出しによって返されます。メッセージがあるキューから別のキューに移動された場合、メッセージ識別子は無効になります。また、メッセージ識別子の値は 32 バイト全体が意味を持つので、MSGID で識別される値は、たとえば空白を埋め込むなどして、完全に初期化する必要があります。

  TPQGETNEXT、TPQGETBYMSGID、TPQGETBYCORRID のいずれかを設定しなければなりません。

TPQGETBYCORRID
: この値が設定されていると、CORRID で識別されるメッセージが取り出されます。相関識別子は、アプリケーションが tpenqueue() で

キューにメッセージを登録したときに指定されます。相関識別子の値は 32 バイト全体が意味を持つので、CORRID で識別される値は、たとえば空白を埋め込むなどして、完全に初期化する必要があります。

TPQGETNEXT、TPQGETBYMSGID、TPQGETBYCORRID のいずれかを設定しなければなりません。

TPQWAIT

この値が設定されていると、キューが空の場合はエラーが返されません。代わりに、メッセージを取り出すことができるようになるまで、プロセスが待機します。待機しない場合は、TPQNOWAIT を設定します。TPQWAIT が TPQGETBYMSGID または TPQGETBYCORRID と併せて設定されている場合、指定されたメッセージ識別子または相関識別子を持つメッセージがキューに存在しないときは、エラーが返されません。代わりに、基準を満たすメッセージを取り出すことができるようになるまで、プロセスは待機します。プロセスには呼び出し元のトランザクション・タイムアウトの影響を受けます。また、トランザクション・モードでない場合、プロセスは TMQUEUE プロセスで -t オプションによって指定されたタイムアウトの影響を受けます。

基準に一致するメッセージをすぐに取り出すことができない場合に、設定されているアクション・リソースの限界に達すると、TPDEQUEUE は失敗し、TP-STATUS に TPEDIAGNOSTIC が設定され、DIAGNOSTIC に QMESYSTEM が設定されます。

TPQWAIT 制御パラメータを指定する TPDEQUEUE() の各要求では、条件を満たすメッセージがすぐに利用できない場合、キュー・マネージャ (TMQUEUE) のアクション・オブジェクトを使用できる必要があります。アクション・オブジェクトを利用できない場合、TPDEQUEUE() 要求は失敗します。利用できるキュー・マネージャのアクション数は、キュー・スペースの作成時または変更時に指定されます。待機中のキューからの取り出し要求が完了すると、対応するアクション・オブジェクトは別の要求に使用できるようになります。

TPQPEEK

TPQPEEK が設定されていると、指定されたメッセージが読み取られてもキューから削除されません。TPNOTRAN フラグを設定する必要があります。トランザクション内でメッセージをキューに登録、また

はキューから取り出す場合、そのトランザクションが完了する前に、メッセージを読み取ることはできません。

あるスレッドが TPQPEEK を使用してメッセージを破棄せずにキューから取り出す場合、その取り出し要求をシステムが処理している短時間の間、非ブロッキング状態のほかのメッセージ取り出し操作にメッセージが認識されないことがあります。たとえば、特定の選択基準 (メッセージ識別子や相関識別子など) を使用してメッセージをキューから取り出す操作が、破棄せずに取り出しが現在行われているメッセージを探している場合などがあります。

TPDEQUEUE() からの出力時には、次の要素が *TPQUEDEF-REC* に設定されます。

```
05 PRIORITY     PIC S9(9) COMP-5.
05 MSGID        PIC X(32).
05 CORRID       PIC X(32).
05 TPQUEQOS-DELIVERY-FLAG    PIC S9(9) COMP-5.
05 TPQUEQOS-REPLY-FLAG       PIC S9(9) COMP-5.
05 REPLYQUEUE                PIC X(15).
05 FAILUREQUEUE              PIC X(15).
05 DIAGNOSTIC                PIC S9(9) COMP-5.
05 CLIENTID OCCURS 4 TIMES   PIC S9(9) COMP-5
05 APPL-RETURN-CODE          PIC S9(9) COMP-5.
05 APPKEY                    PIC S9(9) COMP-5.
```

次は、TPDEQUEUE() からの出力情報を制御する *TPQUEDEF-REC* の有効な設定です。どの値でも、TPDEQUEUE() の呼び出し時に設定されていると、メッセージがキューに登録されたときに提供された値が、レコード内の対応する要素に格納され、その値は設定されたままになります。値を使用できない場合、つまりメッセージがキューに登録されたときに値が提供されなかった場合、または TPDEQUEUE() の呼び出し時に値が設定されなかった場合、値が設定されない状態で TPDEQUEUE() が完了します。

TPQPRIORITY

この値が設定され、TPDEQUEUE() の呼び出しが成功し、メッセージが明示的な優先順位でキューに登録された場合は、その優先順位が PRIORITY に格納されます。優先順位は 1 以上 100 以内の範囲内で、数値が高いほど優先順位も高くなります。つまり、高い数値のメッセージが低い数値のメッセージよりも先にキューから取り出されます。TPQNOPRIORITY が設定されている場合、優先順位は使用できません。

メッセージのキューへの登録時に優先順位が明示的に指定されていない場合、そのメッセージの優先順位は 50 になります。

TPQMSGID

この値が設定されている場合に、TPDEQUEUE() の呼び出しが成功すると、メッセージ識別子が MSGID に格納されます。メッセージ識別子の値は、32 バイト全体が意味を持ちます。TPQNOMSGID が設定されている場合、メッセージ識別子は使用できません。

TPQCORRID

この値が設定され、TPDEQUEUE() の呼び出しが成功し、メッセージが相関識別子を使用してキューに登録された場合、その相関識別子は CORRID に格納されます。相関識別子の値は、32 バイト全体が意味を持ちます。BEA Tuxedo /Q から渡されるメッセージに対するすべての応答は、元のメッセージの相関識別子を持ちます。TPQNOCORRID が設定されている場合、相関識別子は使用できません。

TPQDELIVERYQOS

この値が設定され、TPDEQUEUE() の呼び出しが成功し、メッセージがサービスの配信品質と共にキューに登録された場合、TPQUEQOS-DELIVERY-FLAG で指定されるフラグ TPQQOSDELIVERYDEFAULTPERSIST、TPQQOSDELIVERYPERSISTENT、または TPQQOSDELIVERYNONPERSISTENT がサービスの配信品質を示します。TPQNODELIVERYQOS が設定されている場合、サービスの配信品質は使用できません。

メッセージのキューへの登録時にサービスの配信品質が明示的に指定されていない場合、ターゲットのキューのデフォルトの配信方針がメッセージに対するサービスの配信品質を指定します。

TPQREPLYQOS

この値が設定され、TPDEQUEUE() の呼び出しが成功し、メッセージがサービスの配信品質と共にキューに登録された場合、TPQUEQOS-REPLY-FLAG で指定されるフラグ TPQQOSREPLYDEFAULTPERSIST、TPQQOSREPLYPERSISTENT、または TPQQOSREPLYNONPERSISTENT がサービスの配信品質を示します。TPQNOREPLYQOS が設定されている場合、サービスの応答品質は使用できません。

メッセージがキューに入れられたときにサービスの応答品質が明示的に指定されていない場合、REPLYQUEUE キューのデフォルトの配信

方針が応答に対するサービスの配信品質を指定します。デフォルトの配信方針は、メッセージに対する応答がキューに登録されるときに決定されます。つまり、元のメッセージがキューに登録されてから応答が登録されるまでの間に、応答キューのデフォルトの配信方針が変更された場合、応答が最後に登録される時点で有効な方針が使用されます。

TPQREPLYQ
: この値が設定され、TPDEQUEUE() の呼び出しが成功し、メッセージが応答キューと共にキューに登録された場合、その応答キューの名前が REPLYQUEUE に格納されます。メッセージへの応答は、要求メッセージと同じキュー・スペース内の指定されたキューに登録されます。TPQNOREPLYQ が設定されている場合、応答キューは使用できません。

TPQFAILUREQ
: この値が設定され、TPDEQUEUE() の呼び出しが成功し、メッセージが異常終了キューと共にキューに登録された場合、その異常終了キューの名前が FAILUREQUEUE に格納されます。異常終了メッセージは、要求メッセージと同じキュー・スペース内の指定された異常終了キューに登録されます。TPQNOFAILUREQ が設定されている場合、異常終了キューは使用できません。

TPQUEDEF-REC の残りの設定は、TPDEQUEUE() が呼び出されると、次の値に設定されます。TPQNOTOP、TPQNOBEFOREMSGID、TPQNOTIME_ABS、TPQNOTIME_REL、TPQNOEXPTIME_ABS、TPQNOEXPTIME_REL、および TPQNOEXPTIME_NONE。

TPDEQUEUE() の呼び出しが失敗し、TP-STATUS に TPEDIAGNOSTIC が設定された場合、失敗の原因を示す値が DIAGNOSTIC に返されます。DIAGNOSTIC の有効な設定値には、4-7 ページの「TPQUEDEF-REC 構造体」に示されている TPENQUEUE() の値 (QMENOSPACE と QMERELEASE を除く) のほかに、次のものがあります。

[QMENOMSG]
: キューから取り出せるメッセージはありません。メッセージがキュー上に存在し、別のアプリケーション・プロセスがこのメッセージをキューから読み取った可能性があることに注意してください。その場合、その別のプロセスがトランザクションをロールバックしたときに、メッセージがキューに戻されます。

[QMEINUSE]

メッセージ識別子または相関識別子を使用してメッセージをキューから取り出す際に、指定されたメッセージが別のトランザクションによって使用されています。それ以外の場合、現在キューにあるすべてのメッセージは、ほかのトランザクションによって使用されています。この診断は、BEA Tuxedo リリース 7.1 以降のキュー・マネージャでは返されません。

TPQWAIT の使用

フラグに TPQWAIT を設定して TPDEQUEUE() を呼び出すと、メッセージをすぐに取り出すことができない場合、TPDEQUEUE() が制御を呼び出し元に返す前に、TMQUEUE サーバは TPDEQUEUE() 要求に一致するメッセージがキューに到着するのを待ちます。TMQUEUE プロセスは待機中の要求を無視し、ほかのプロセスの要求を処理しながら、最初の要求の条件が満たされるのを待ちます。TPQGETBYMSGID や TPQGETBYCORRID も指定されている場合、サーバは指定されたメッセージ識別子や相関識別子を持つメッセージをキューから取り出せるようになるまで待ちます。このどちらのフラグも設定されていない場合、サーバは任意のメッセージがキューに登録されるまで待ちます。呼び出しがトランザクション・モードの場合、サーバの待機時間は、呼び出し元のトランザクション・タイムアウトによって制御されます。呼び出しがトランザクション・モードでない場合、TMQUEUE サーバの CLOPT パラメータに指定される -t オプションで制御されます。

TMQUEUE サーバは、要求を処理するアクション・リソースを利用できる限り、複数の待機中 TPDEQUEUE() 要求を同時に処理できます。十分なアクション・リソースがキュー・スペースに対して設定されていない場合、TPDEQUEUE() は失敗します。使用しているシステムがこれに該当する場合は、キュー・スペースに対するアクション・リソースの数を増やしてください。

TMQFORWARD サービス使用時のエラー処理

キューからのメッセージの取り出しで、エラー処理の最良の方法を考慮する場合、次の 2 種類のエラーを区別しておきます。

- 要求されたサービスに転送するために、TMQFORWARD(5) がメッセージをキューから取り出すときに発生するエラー
- 要求を処理するサービスに発生するエラー

デフォルトでは、メッセージがトランザクション内でキューから取り出され、そのトランザクションがロールバックされると、そのメッセージはキューに戻されて、再度キューからの取り出しと実行が可能になります。一時的な障害が解消されるように (たとえば、データベースのロックを別のトランザクションが解除するように) 少し時間をおいてから、メッセージの取り出しと実行を再度試みます。通常、再試行の回数に上限を設定することも、アプリケーション不備によってリソースを著しく浪費することを防ぐために有用です。管理者がキューを設定するときに、再試行の回数と遅延時間 (秒単位) の両方を指定できます。再試行の回数 0 は、再試行が行われないことを示します。再試行の回数が上限に達すると、管理者がキュー・スペースに設定するエラー・キューにメッセージが移動されます。エラー・キューが設定されていない場合、再試行回数の上限に達したメッセージは単に削除されます。エラー・キューにあるメッセージは、管理者によって処理されなければなりません。管理者は、アプリケーションの要件を満たす方法で発信元に通知する必要があります。選択されたメッセージ処理方法は、メッセージをキューに登録した発信元プログラムにほとんど透過的です。メッセージは一度正常にキューに登録されると、TPENQUEUE() のパラメータおよびキューの属性に従って処理されることが実質的には保証されます。メッセージがエラー・キューに移動されたことの通知は、キューのパラメータを適切に調整されたシステムではほとんどありません。

異常終了キュー (通常、キュー・スペースのエラー・キューとは異なります) は、キューに登録された各メッセージと対応付けられます。このキューは、異常終了メッセージを置く場所として、キューにメッセージを登録する呼び出しで指定されます。特定の要求に対する異常終了メッセージは、アプリケーション生成の相関識別子で識別できます。相関識別子は、キューにメッセージを登録するときに、そのメッセージと対応付けられます。

成功するまで (または事前に定義された回数まで) 再試行を繰り返すというデフォルトの動作は、時間が経過すれば解決されるような一時的な障害が失敗の原因であり、メッセージが後から適切に処理される場合は適しています。

障害が一時的なものではない場合もあります。たとえば、存在しないアカウントでの操作を要求するメッセージがキューに入れられている場合 (アプリケーションも適当な時間に存在していない場合) です。その場合、再試行し

てリソースを浪費しないようにします。アプリケーション・プログラマまたは管理者が、特定の操作の失敗が一時的ではないと判断した場合、単に再試行回数を 0 に設定します。ただし、この場合、このようなメッセージが入るキュー・スペースのエラー・キューを継続的にクリアするメカニズム (キューを定期的に読み取るバックグラウンドのクライアントなど) が必要です。また、同じサービスに対する障害でも、一時的なもの (データベース・ロックの競合など) であったり、永続的なもの (アカウントが存在しないなど) であったりする場合が多くあります。

メッセージが TMQFORWARD によって処理される (キューから取り出され、TPCALL を介してアプリケーションに渡される) 場合、TPCALL が返す情報には、TPESVCFAIL エラーが一時的な障害と永続的な障害のどちらによって生じたものであるかを示すメカニズムはありません。

キューからの取り出しをアプリケーションが処理する場合、簡単な解決方法はたとえ操作が失敗しても、そのサービスに対して成功を返すことです。つまり、TPSUCCESS を設定して TPRETURN を実行します。これにより、トランザクションはコミット可能になり、メッセージはキューから削除されます。応答メッセージが使用されている場合、サービスから返されるバッファの情報によって操作の失敗を示すことができ、メッセージが応答キューに登録されます。TPRETURN の *TPSVCRET-REC* 引数内の APPL-CODE も、アプリケーション固有の情報を返すために使用できます。

サービスが失敗し、トランザクションをロールバックする必要がある場合、TMQFORWARD が処理をそれ以上行わずに、2 番目のトランザクションを実行してキューからメッセージを削除するかどうかは明確ではありません。デフォルトでは、TMQFORWARD は失敗したサービスに対するメッセージを削除しません。TMQFORWARD のトランザクションはロールバックされ、メッセージはキューに格納されます。TMQFORWARD にコマンド行オプションを指定すると、サービスが失敗して 0 より長い応答メッセージが返された場合に、メッセージがキューから削除されます。メッセージは、2 番目のトランザクションで削除されます。この処理を行うには、キューに遅延時間および再試行回数を設定する必要があります。メッセージが失敗したキューに対応付けられている場合、メッセージがキューから削除されたトランザクションと同じトランザクションで、応答データが異常終了キューに登録されます。

TMQFORWARD を通して呼び出されたサービスからの応答をキューから取り出す手順

アプリケーションがキューに登録されたメッセージに対する応答を要求している場合、次の手順に従います。

1. 前提条件として、キュー・スペースに応答キューおよび異常終了キューがあることが必要です。アプリケーションは、相関識別子の内容について一貫していなければなりません。サービスは、論理的な失敗の場合に TPSUCCESS を返し、TPRETURN の *TPSVCRET-REC* 引数内の APPL-CODE に情報を示すコードを返すようにコーディングします。

2. TPENQUEUE() を呼び出してキューにメッセージを登録する際に、次を設定します。

   ```
   TPQCORRID       TPQREPLYQ
   TPQFAILUREQ     TPQMSGID
   ```

   この呼び出しを行う前に、CORRID、REPLYQUEUE、および FAILUREQUEUE の値を設定します。呼び出しから制御が戻ったら、CORRID を保存します。

3. TPDEQUEUE() を呼び出して応答を確認する際に、QNAME に応答キューを指定し、次を設定します。

   ```
   TPQCORRID       TPQREPLYQ
   TPQFAILUREQ     TPQMSGID
   TPQGETBYCORRID
   ```

   この呼び出しを行う前に、保存した相関識別子を使用して CORRID に入力します。TPDEQUEUE() の呼び出しが失敗し、TP-STATUS に TPEDIAGNOSTIC が設定された場合、詳しい情報を DIAGNOSTIC から取得できます。エラー・コード QMENOMSG を受信した場合、キューから取り出すことのできるメッセージがなかったことを示します。

4. もう 1 つの TPDEQUEUE() 呼び出しを設定します。この呼び出しでは、QNAME が異常終了キューの名前を指すようにし、次を設定します。

   ```
   TPQCORRID       TPQREPLYQ
   TPQFAILUREQ     TPQMSGID
   TPQGETBYCORRID
   ```

TPQCORRID に相関識別子を挿入します。呼び出しから制御が戻ったら、LEN を確認してデータを受信したかどうかを確認し、APPL-RETURN-CODE を調べてサービスがユーザ戻り値を返したどうかを確認します。

メッセージの順次処理

メッセージの順次処理は、あるサービスがそのトランザクションがコミットされる前に、連鎖的に次のサービス用にメッセージをキューに登録することによって行われます。最初の発信元のプロセスは、reply_queue に対する一連の TPDEQUEUE() 呼び出しによって順次処理の進行状況をトラッキングできます。ただし、各メンバが同じ相関識別子を使用し、長さ 0 の応答を返すことが必要です。

また、任意通知型通知を使用して、順次処理全体が成功したという通知を最初の発信元に返すこともできます。順次処理の最後のトランザクションが TPCOMMIT で終了したことを確認するには、*TPQUEDEF-REC* 構造体で渡されるクライアント識別子を使用して、TPNOTIFY を呼び出す操作を追加します。最初の発信元であるクライアントは、TPSETUNSOL を呼び出して、使用されている任意通知型メッセージ・ハンドラを指定しておかなければなりません。

ピア・ツー・ピア通信でのキューの使用

キューへのメッセージの登録およびキューからのメッセージの取り出しに関するこれまでの説明では、キューが要求 / 応答の一形態として使用されていることが暗黙の前提になっていました。メッセージ自体はサービス要求である必要はありません。キュー機能は、あるプロセスから別のプロセスに、サービス要求と同じように効果的にデータを転送できます。アプリケーション間またはクライアント間のこの通信方式は、ピア・ツー・ピア通信と呼ばれます。

使用するアプリケーションが、このような目的で BEA Tuxedo /Q を使用することに適している場合は、管理者に別のキューを作成してもらい、そのキューからメッセージを取り出す独自の受信用プログラムをコーディングします。

章

A サンプル・アプリケーション

ここでは、次の内容について説明します。

- 概要
- 前提条件
- qsample とは
- qsample のビルド
- 理解を深めるために

概要

ここでは、1 つのクライアントと 1 つのサーバ、および BEA Tuxedo /Q を使用するアプリケーション `qsample` について説明します。このソフトウェア (対話形式) は、BEA Tuxedo ソフトウェアに同梱されています。

前提条件

このサンプル・アプリケーションを実行するには、BEA Tuxedo ソフトウェアがインストールされてビルドされ、ここで説明するファイルやコマンドを使用できることが必要です。BEA Tuxedo ソフトウェアをインストールする必要がある場合、そのインストール方法については『BEA Tuxedo システムのインストール』を参照してください。

既にインストールされている場合は、ソフトウェアのインストール先のルート・ディレクトリのパス名を確認する必要があります。また、BEA Tuxedo ディレクトリ構造内のディレクトリとファイルに読み取りパーミッションと実行パーミッションを設定し、qsample の各ファイルをコピーしたり、BEA Tuxedo の各コマンドを実行できるようにします。

qsample とは

qsample は、BEA Tuxedo /Q を使用する基本的な BEA Tuxedo アプリケーションです。このアプリケーションは、1 つのクライアントと 1 つのサーバ、および 2 つのシステム・サーバ TMQUEUE(5) と TMQFORWARD(5) を使用します。クライアントは、TMQUEUE を呼び出して、qsample 用に作成されたキュー・スペースにメッセージを登録します。このメッセージは、TMQFORWARD によってキューから取り出され、アプリケーション・サーバに渡されます。アプリケーション・サーバは、文字列を小文字から大文字に変換し、TMQFORWARD に返します。TMQFORWARD は、応答メッセージをキューに登録します。その間に、クライアントは TMQUEUE を呼び出して、応答をキューから取り出します。応答を受信すると、クライアントはその応答をユーザの画面に表示します。

qsample のビルド

以下は、qsample アプリケーションをビルドして実行する手順です。

1. qsample 用にディレクトリを作成し、そのディレクトリに移動 (cd) します。

   ```
   mkdir qsampdir
   cd qsampdir
   ```

   この作業は省略せずに行ってください。この作業を行うと、最初からあった qsample のファイルと、手順に従って作成したファイルを確認できるようになります。C シェル (/bin/csh) を使用せずに、標準シェル (/bin/sh) または Korn シェルを使用してください。

2. qsample ファイルをコピーします。

   ```
   cp $TUXDIR/apps/qsample/* .
   ```

   一部のファイルを編集してから、実行形式ファイルを作成することがあるので、本ソフトウェアで提供されたオリジナルのファイルではなく、コピーしたファイルを用いて作業することをお勧めします。

3. ファイルを一覧表示します。

   ```
   $ ls
   README
   client.c
   crlog
   crque
   makefile
   rmipc
   runsample
   server.c
   setenv
   ubb.sample
   $
   ```

このアプリケーションを構成するファイルは、次のとおりです。

`README`
: アプリケーション、およびそのコンフィギュレーションと実行方法について記述したファイル

`setenv`
: 環境変数を設定するスクリプト

`crlog`
: `TLOG` ファイルを作成するスクリプト

`crque`
: このアプリケーション用のキュー・スペースおよびキューを定義するスクリプト

`makefile`
: このアプリケーションの実行可能ファイルを作成する makefile

`client.c`
: クライアント・プログラムのソース・コード

`server.c`
: サーバ・プログラムのソース・コード

`ubb.sample`
: このアプリケーションに対する ASCII 形式のコンフィギュレーション・ファイル

`runsample`
: このサンプル・アプリケーションのビルドと実行に必要なすべてのコマンドを呼び出すスクリプト

`rmipc`
: キュー・スペースのために IPC 資源を削除するスクリプト

4. setenv ファイルを編集します。

   setenv ファイルを開いて、TUXDIR 値を BEA Tuxedo システムのインストール先のルート・ディレクトリの絶対パス名に変更します。この値を編集するときに、山かっこ (< と >) を削除します。

5. runsample を実行します。

   runsample スクリプトには、いくつかのコマンドが記述されています。各コマンドの前には、そのコマンドを説明するコメント行があります。

```
#set the environment
. ./setenv
chmod +w ubb.sample
uname="`uname -n`"
ed ubb.sample<<!
g;<uname -n>;s;;${uname};
g;<full path of Tuxedo software>;s;;${TUXDIR};
g;<full path of APPDIR>;s;;${APPDIR};
w
q
!
#build the client and server
make client server
#create the tuxconfig file
tmloadcf -y ubb.sample
#create the TLOG
./crlog
#create the QUE
./crque
#boot the application
tmboot -y
#run the client
client
#shutdown the application
tmshutdown -y
#remove the client and server
make clean
#remove the QUE ipc resources
./rmipc
#remove all files created
rm tuxconfig QUE stdout stderr TLOG ULOG*
```

   このスクリプトを実行すると、各種のコマンドから生成される一連のメッセージが画面に表示されます。このメッセージの中に、次の行があります。

```
before: this is a q example
after:THIS IS A Q EXAMPLE
```

before: で始まる行は、client がキューに登録し、server で処理される文字列のコピーです。after: で始まる行は、server から返された文字列です。この 2 行によって、プログラムが正常に動作したことがわかります。

理解を深めるために

runserver を使用してサンプル・アプリケーションのビルドと実行を行うほかに、このアプリケーションの各部分について理解することも有用です。ここでは、参照しておくと役立つ内容について説明します。このアプリケーションについて理解を深めると、ほかのアプリケーションも理解できるようになります。

setenv: 環境の設定

setenv は、BEA Tuxedo の開発でよく使用されるファイルの 1 つです。TUXDIR、APPDIR、および PATH の 3 つの変数が設定されています。これらの変数は、BEA Tuxedo システムで作業しているときに常に必要になります。SUN マシンで実行する場合は、PATH 変数の先頭に別個に bin が必要になるので注意してください。LD_LIBRARY_PATH、SHLIB_PATH、または LIBPATH は、共用ライブラリを使用してシステムをビルドする場合に重要です。使用する必要のある変数は、お使いのオペレーティング・システムに依存します。TUXCONFIG は、システムを起動する前に設定しておく必要があります。QMADMIN は、変数で設定することも、qmadmin(1) コマンド行で指定することもできます。

考慮すべき点は、BEA Tuxedo /Q の作業環境にこのようなファイルが必要とされるかどうか、また、自分の .profile にコマンドを記述して、ログイン時に環境設定を行うかどうかです。

makefile: アプリケーションの作成

makefile は、サーバおよびクライアントをビルドするために、それぞれ buildserver(1) および buildclient(1) を使用することに注意してください。これらのコマンドを個別に実行することも、または make 機能を使用してアプリケーションを常に最新にしておくこともできます。

ここでは makefile について説明していますが、クライアント・プログラムとサーバ・プログラムの .c ファイルも参照しておくことをお勧めします。BEA Tuxedo /Q と関連して特に重要なことは、tpenqueue および tpdequeue の呼び出しです。*qspace* および *qname* 引数の値に特に注意してください。コンフィギュレーション・ファイルを確認すると、これらの値の設定元がわかります。

ubb.sample:ASCII コンフィギュレーション・ファイル

コンフィギュレーション・ファイルの中で、特に直接関係する 3 つのエントリとして、TMQUEUE サーバの CLOPT パラメータ、TMQFORWARD サーバの CLOPT パラメータ、および *GROUPS エントリの OPENINFO パラメータがあります。次のコードは、この 3 つを抽出したものです。

```
# First the CLOPT parameter from TMQUEUE:
        CLOPT = "-s QSPACENAME:TMQUEUE --  "
# Then the CLOPT parameter from TMQFORWARD:
        CLOPT="-- -i 2 -q STRING"
# Finally, the OPENINFO parameter from the QUE1 group:
        OPENINFO = "TUXEDO/QM:<APPDIR pathname>/QUE:QSPACE"
```

TMQUEUE の CLOPT パラメータは、サービスのエイリアスとして QSPACENAME を指定しています。再度 client.c を参照して、tpenqueue および tpdequeue の *qspace* 引数を確認してください。TMQFORWARD の CLOPT パラメータは、-q オプションを使用して STRING サービスを指定しています。これもそのサービス用にメッセージが登録されるキューに指定された名前であり、client.c にある tpenqueue の *qname* 引数として指定されています。

tmloadcf(1) コマンドは、ASCII 形式のコンフィギュレーション・ファイルをコンパイルして TUXCONFIG ファイルを作成するために使用されます。

crlog: トランザクション・ログの作成

crlog のスクリプトは、tmadmin(1) を呼び出して、TLOG 用のデバイス・リスト・エントリを作成し、その後でコンフィギュレーションで指定されているサイトのためのログを作成します。キュー機能のすべてのメッセージは、トランザクション内でキューに登録され、キューから取り出されるので、TMS_QM サーバによって管理されるトランザクションをトラッキングするログが必要になります。

crque: キュー・スペースとキューの作成

crque のスクリプトは、qmadmin(1) を呼び出して、このサンプル・アプリケーションで使用されるキュー・スペースおよびキューを作成します。キュー・スペースは、QSPACE という名前になることに注意してください。この名前は、コンフィギュレーション・ファイルの OPENINFO パラメータの最後の引数として指定される名前でもあります。STRING および RPLYQ という名前のキューが作成されます。このスクリプトの qspacecreate 部分では、エラー・キューの名前が指定されています。ただし、このスクリプトには、そのキューを作成する qcreate コマンドは含まれていません。この部分は、必要に応じて後で変更します。

アプリケーションの起動、実行、およびシャットダウン

アプリケーション・プログラムを作成し、TUXCONFIG をロードし、キュー・スペースおよびキューを作成したら、次にアプリケーションを起動して実行します。アプリケーションを起動するには、次のコマンドを実行します。

```
tmboot -y
```

-y オプションは、tmboot が起動前の確認用プロンプトを表示しないようにします。

サンプル・アプリケーションを実行するには、単に次のコマンドを入力します。

```
client
```

アプリケーションをシャットダウンするには、tmshutdown コマンドを使用します。

終了処理

runsample スクリプトには、スクリプト実行前の環境に戻すためのコマンドが 3 つあります。make clean コマンドでは、make を使用して、クライアント用とサーバ用のオブジェクト・ファイルと実行可能ファイルを削除します。

rmipc コマンドは、キュー・スペースの IPC 資源が tmshutdown を使用しても自動的に削除されないため、使用します。tmshutdown は、アプリケーションで使用される BEA Tuxedo IPC 資源を削除します。rmipc を参照すると、rmipc が qmadmin を呼び出し、qmadmin の ipcrm コマンドを使用していることがわかります。その場合、QSPACE を指定して削除する資源が識別されています。

このスクリプトの最後のコマンドは、rm コマンドです。このコマンドは、アプリケーションで生成された数多くのファイルを削除します。これらのファイルが残っていても問題はありません。実際、このサンプル・アプリケーションで作業を続ける場合、tuxconfig、QUE、および TLOG は再度作成しなくても済むように残しておくと便利です。