BEA Tuxedo®

BEA Tuxedo アプリケーション実行時の管理

BEA Tuxedo 8.1
2003 年 1 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

目次

3. アプリケーションの動的な変更

8. BEA Tuxedo ATMI アプリケーションのチューニング

9. BEA Tuxedo アプリケーションのトラブルシューティング

このマニュアルについて

このマニュアルでは、BEA Tuxedo® システム依存型の Tuxedo ATMI 環境、またはオブジェクト指向の Tuxedo CORBA 環境のいずれかで管理する方法について説明します。

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「アプリケーションの起動とシャットダウン」では、BEA Tuxedo アプリケーションの起動およびシャットダウン方法について説明します。
- 第 2 章「BEA Tuxedo アプリケーションの監視」では、リソース、アクティビティ、およびコンフィギュレーションで発生する可能性のある問題の監視方法について説明します。
- 第 3 章「アプリケーションの動的な変更」では、システムをシャットダウンせずにコンフィギュレーションを変更する方法について説明します。
- 第 4 章「分散アプリケーションでのネットワーク管理」では、分散型 BEA Tuxedo アプリケーションをサポートするためのネットワーク環境の管理方法について説明します。
- 第 5 章「イベント・ブローカについて」では、イベント・ブローカの概要を示します。イベント・ブローカは、BEA Tuxedo アプリケーションで実行中のプロセス間で、アプリケーション・イベントを非同期にルーティングするためのツールです。
- 第 6 章「イベントのサブスクライブ」では、イベント・ブローカ・サーバの設定方法について説明します。
- 第 7 章「アプリケーションの移行」では、設定されたバックアップ・マシンまたは代替マシンに BEA Tuxedo サーバを移行する方法について説明します。

- 第 8 章「BEA Tuxedo ATMI アプリケーションのチューニング」では、Tuxedo ATMI 環境でアプリケーションをスムーズに実行する方法について説明します。
- 第 9 章「BEA Tuxedo アプリケーションのトラブルシューティング」では、BEA Tuxedo システムにおける各種トラブルシューティングの実行方法について説明します。.

対象読者

このマニュアルは、主にミッション・クリティカルな BEA Tuxedo システムをサポートするパラメータの設定を行うシステム管理者を対象としています。また、BEA Tuxedo プラットフォームについて理解してい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/) から無償で入手できます。

関連情報

以下のマニュアルには、BEA Tuxedo ソフトウェアについての関連情報が掲載されています。

- 『BEA Tuxedo システムのインストール』—CD に同梱されているドキュメント
- 『BEA Tuxedo リリース・ノート』—CD に同梱されているドキュメント
- 『BEA Tuxedo アプリケーションの設定』—BEA Tuxedo オンライン・マニュアル CD で参照できます。このガイドには、BEA Tuxedo システムのセットアップおよび管理方法についての情報が記載されています。
- 『BEA Tuxedo Domains コンポーネント』—オンライン・マニュアル CD で参照できます。このガイドには、BEA Tuxedo ドメインの設定方法と管理方法が記載されています。

- 『BEA Tuxedo CORBA アプリケーションのスケーリング、分散、およびチューニング』―BEA Tuxedo オンライン・マニュアル CD で参照できます。このガイドでは、BEA Tuxedo CORBA 環境で実行する CORBA アプリケーションの調整方法を説明しています。

『BEA Tuxedo ATMI のアーキテクチャ』および 『BEA Tuxedo CORBA のアーキテクチャ』環境の設定と管理の詳細については、http://edocs.beasys.co.jp/ の CORBA Bibliography を参照してください。

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`

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*`

規則	項目
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。

1 アプリケーションの起動とシャットダウン

ここでは、次の内容について説明します。

- アプリケーションの起動とシャットダウンに必要なタスク
- 環境変数の設定
- TUXCONFIG ファイルの作成
- アプリケーション固有のディレクトリとファイルを手動で複製転送する
- TLOG デバイスの作成
- アプリケーションの起動
- アプリケーションのシャットダウン

アプリケーションの起動とシャットダウンに必要なタスク

次のフローチャートに、BEA Tuxedo アプリケーションの起動とシャットダウンに必要なタスクを示します。

タスク名をクリックすると、そのタスクの実行方法が表示されます。

図 0-1 起動とシャットダウンのタスク

環境変数の設定

BEA Tuxedo アプリケーションを管理するには、BEA Tuxedo の実行可能ファイルおよびデータ・ライブラリにアクセスできなければなりません。アプリケーションの起動とシャットダウンに必要なコマンドは %TUXDIR%\bin (Windows 2000 ホスト・マシンの場合) および $TUXDIR/bin (UNIX ホスト・マシンの場合) に格納されています。

Windows の場合

Windows 2000 ホスト・マシンの環境を設定するには、コマンド・プロンプトで次のコマンドを入力します。

```
set TUXCONFIG=path_name_of_TUXCONFIG_file
set TUXDIR=path_name_of_BEA_Tuxedo_system_root_directory
set APPDIR=path_name_of_BEA_Tuxedo_application_root_directory
set PATH=%APPDIR%;%TUXDIR%\bin;%PATH%
```

イタリック体の文字列は、インストール先の適切な絶対パス名に置き換えてください。

Windows 2000 は、PATH 変数に設定されたパスを使って、動的に読み込み可能な必須ライブラリ・ファイルにアクセスします。具体的には、次の順序で動的に読み込み可能なライブラリ・ファイルが検索されます。

1. BEA Tuxedo アプリケーションのインストール元ディレクトリ
2. 現在のディレクトリ
3. Windows システム・ディレクトリ (C:\Win2000\System32 など)
4. Windows ディレクトリ (C:\Win2000 など)
5. PATH 環境変数に設定されているディレクトリ

UNIX の場合

UNIX ホスト・マシンの環境を設定するには、次のように環境変数を設定し、エクスポートします。

```
TUXCONFIG=path_name_of_TUXCONFIG_file
TUXDIR=path_name_of_BEA_Tuxedo_system_root_directory
APPDIR=path_name_of_BEA_Tuxedo_application_root_directory
PATH=$APPDIR:$TUXDIR/bin:/bin:$PATH
LD_LIBRARY_PATH=$APPDIR:$TUXDIR/lib:/lib:/usr/lib:$LD_LIBRARY_PATH
export TUXCONFIG TUXDIR APPDIR PATH LD_LIBRARY_PATH
```

プラットフォームの種類	設定
HP-UX (HP 9000)	LD_LIBRARY_PATH の代わりに SHLIB_PATH を使用します。
RS/6000 (AIX)	LD_LIBRARY_PATH の代わりに LIBPATH を使用します。

イタリック体の文字列は、インストール先の適切な絶対パス名に置き換えてください。

注記　アプリケーション管理者は、UBBCONFIG ファイルの MACHINES セクションで、TUXCONFIG、TUXDIR、および APPDIR の 3 つの環境変数を定義するか、またはアプリケーション内の各マシンに対して TM_MIB の T_MACHINE クラスを定義してください。これらの環境変数については、UBBCONFIG(5) または TM_MIB(5) のリファレンス・ページを参照してください。

TUXCONFIG ファイルの作成

各 BEA Tuxedo ドメインは、コンフィギュレーション・ファイルによって制御されます。コンフィギュレーション・ファイルには、インストールに応じて異なるパラメータが定義されています。テキスト形式のコンフィギュレー

ション・ファイルを UBBCONFIG と呼びます。バイナリ形式の UBBCONFIG ファイルは TUXCONFIG と呼びます。UBBCONFIG ファイルと同様、TUXCONFIG ファイルがどのような名前であっても、実際の名前は TUXCONFIG 環境変数で指定されたデバイス・ファイル名またはシステム・ファイル名になります。

注記　コンフィギュレーション・ファイルについては、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の UBBCONFIG(5) を参照してください。

tmloadcf(1) コマンドは、テキスト形式のコンフィギュレーション・ファイルを TUXCONFIG と呼ばれるバイナリ形式のファイルに変換し、TUXCONFIG 変数で指定されている位置に書き込みます。次のようにコマンドを実行します。

```
$ tmloadcf [-n] [-y] [-c] [-b blocks] {UBBCONFIG_file | - }
```

注記　このコマンドを実行するには、MASTER マシンにログインして、コンフィギュレーション・ファイルの所有者としての有効なユーザ ID を取得する必要があります。

各オプションの機能は、次のとおりです。

- -n オプションは、構文チェックのみを行い、エラーを報告します。
- -y オプションは、既存の TUXCONFIG ファイルを上書きします。上書きするかどうかの確認は行いません。
- -c オプションは、コンフィギュレーションの IPC (プロセス間通信) 資源の最小値を計算します。
- -b オプションは、TUXCONFIG ファイルのサイズを制限します。

-c および -n オプションを使用すると、TUXCONFIG ファイルはロードされません。IPC 資源はプラットフォームに固有です。-c オプションを使用する場合は、『BEA Tuxedo Installation Guide』のプラットフォーム・データシートを確認し、IPC 資源を変更する必要があるかどうかを判断してください。IPC 資源を変更する場合は、そのプラットフォームの管理者マニュアルを参照してください。-n オプションにより、コンフィギュレーション・ファイルの構文エラーが見つかった場合は、エラーを修正してから作業を進めてくださいなお、*UBBCONFIG_file* には、コンフィギュレーション・ファイルの完全修飾名を指定します。

-b オプションには、引数として、TUXCONFIG ファイルを格納するためのブロック数の制限値を指定できます。初期化されていない raw ディスク・デバイスに TUXCONFIG をインストールする場合は、このオプションを使用してください。TUXCONFIG が通常の UNIX システム・ファイルに格納されている場合、このオプションはお勧めできません。

全サイトでの tlisten の起動

ネットワーク・アプリケーションを実行するには、リスナ・プロセスを各マシン上で実行している必要があります。ネットワーク・アプリケーションとは、複数のマシン上で稼働するアプリケーションであり、UBBCONFIG ファイルの RESOURCES セクションにある MODEL MP パラメータで設定します。

注記　TUXDIR、TUXCONFIG、APPDIR、およびその他の関連する環境変数を定義してから、tlisten を起動してください。

プロセスが接続をリッスンするポートは、コンフィギュレーション・ファイルの NETWORK セクションにある NLSADDR で指定されているポートと同じでなければなりません。各マシン上で、tlisten(1) コマンドを次のように使用します。

```
tlisten [ -d device ] -l nlsaddr [-u {uid-# | uid-name}] [ -z bits ] [ -Z bits ]
```

例：`tlisten -l //machine1:6500`

tlisten のコマンド・オプション

- -d *device*—ネットワーク・デバイスのフル・パス名。BEA Tuxedo のリリース 6.4 以降では、このオプションは省略可能です。BEA Tuxedo システムの以前のバージョン (リリース 6.3 以前) では、TCP/IP など一部のネットワーク・プロバイダでこの情報が必要になります。
- -l *nlsaddr*—プロセスが接続をリッスンするネットワーク・アドレス。TCP/IP アドレスは次の形式で指定します。

"//hostname:port_number"

"//#.#.#.#:port_number"

最初の形式の場合、tlisten はローカル名を解決する機能 (通常は DNS) を使って hostname のアドレスを検索します。hostname にはローカル・マシン名を指定し、名前解決の機能で hostname をローカル・マシンのアドレスに明確に解決する必要があります。

2 番目の形式の場合、#.#.#.# にはドット区切りの 10 進数を指定します。ドット区切りの 10 進数形式では、それぞれの # に 0 ～ 255 の数字を指定します。このドット区切りの 10 進数は、ローカル・マシンの IP アドレスを表します。どちらの形式の場合も、port_number には tlisten プロセスが接続要求の受信をリッスンする TCP ポート番号を指定します。port_number には、0 ～ 65535 までの数字または名前を指定します。port_number が名前の場合は、ローカル・マシンのネットワーク・サービス・データベースになければなりません。アドレスは、先頭に 0x をつけ、16 進形式で指定することもできます。先頭の 0x に続く文字として、0 ～ 9 までの数字か、または A から F までの文字 (大文字と小文字は区別しない) を指定できます。IPX/SPX や TCP/IP のような任意のバイナリ・ネットワーク・アドレスには、16 進数の形式が便利です。アドレスは、任意の文字列としても指定できます。値は、コンフィギュレーション・ファイル内の NETWORK セクションにある NLSADDR パラメータの値と同じでなければなりません。

tmloadcf(1) は、nlsaddr が MASTER LMID のエントリから欠落している場合は警告を出力し、これ以外のエントリから欠落している場合はエラーを出力します。ただし、nlsaddr が MASTER LMID エントリから欠落している場合には、tmadmin(1) をリモート・マシンから管理者モードで実行することはできません。可能な処理は、読み取りのみの操作だけです。これは、MASTER サイトで障害が発生しても、バックアップ・サイトから再起動できないことも意味します。

- -u *uid-#* または *uid-name*—ユーザを指定して tlisten プロセスを実行する場合に使用します。このオプションは、リモート・マシンの root で tlisten(1) コマンドを実行する場合に必要です。
- -z [*bits*]—BEA Tuxedo システムの管理プロセスと tlisten との間のネットワーク・リンクを確立する場合に必要な最低レベルの暗号化を指定します。ゼロ (0) は、暗号化が行われないことを示し、56 および 128

は暗号化キーの長さ (ビット単位) を指定します。ここで指定する最小レベルの暗号化が満たされない場合、リンクの確立は失敗します。デフォルト値はゼロです。

- -Z [*bits*]—BEA Tuxedo システムの管理プロセスと tlisten との間のネットワーク・リンクを確立する場合に可能な最大レベルの暗号化を指定します。ゼロ (0) は、暗号化が行われないことを示し、56 および 128 は暗号化キーの長さ (ビット単位) を指定します。デフォルト値は 128 です。-z オプションおよび -Z オプションは、国際版または米国 / カナダ版の BEA Tuxedo セキュリティ・アド・オン・パッケージがインストールされている場合にのみ使用できます。

アプリケーション固有のディレクトリとファイルを手動で複製転送する

tmboot(1) を実行すると、コンフィギュレーション内のすべてのマシンに TUXCONFIG が自動的に複製転送されます。ただし、手動で複製転送しなければならないファイルもあります。次の表は、ネットワーク・アプリケーションに必要なファイルとディレクトリの一覧です。まず、BEA Tuxedo システムをマシンにインストールしてください。

注記　tlisten プロセスは、アプリケーションを起動する前に、ネットワーク接続された BEA Tuxedo アプリケーションの各マシンで起動する必要があります。tlisten(1) のリファレンス・ページを参照してください。

TUXDIR、TUXCONFIG、APPDIR、およびその他の関連する環境変数を定義してから、tlisten を起動してください。

表 1-1 複製転送するディレクトリとファイル

ディレクトリ / ファイル	説明
APPDIR	各ノードに対して、APPDIR 変数に指定した名前のディレクトリを作成する必要があります。すべてのノードで同じディレクトリ・パス名を使用すると便利です。
実行可能ファイル	プラットフォームごとにアプリケーション・サーバのセットを 1 つ作成し、プラットフォームで実行中のほかのすべてのマシンには、適切なアプリケーション・サーバのセットを手動で複製転送する必要があります (これは自動的には行われません)。実行可能ファイルは APPDIR 変数で指定したディレクトリに格納するか、またはコンフィギュレーション・ファイルの MACHINES セクションの ENVFILES にある PATH 変数で指定されたディレクトリに格納します。
フィールド・テーブル VIEW テーブル	バッファ型として FML または VIEWS を使用する場合は、これらのバッファ型を使用するマシンにフィールド・テーブルと VIEW 記述ファイルを手動で複製転送し、再コンパイルする必要があります。フィールド・テーブル・ファイルからヘッダ・ファイルを作成するには mkfldhdr、mkfldhdr32(1) を使用します。VIEW ファイルをコンパイルするには viewc、viewc32(1) を使用します。FML フィールド・テーブルおよび VIEW 記述ファイルは、FLDTBLDIR、FIELDTBLS、VIEWDIR、VIEWFILES、またはこれらの環境変数の 32 ビット版を通じて使用できます。

TLOG デバイスの作成

分散トランザクション処理を作成するには、トランザクションに参加するマシンにグローバル・トランザクション・ログ (TLOG) を作成する必要があります。TLOG を定義するには、次の手順に従います。

1. 最初に、コンフィギュレーション ファイルの MACHINES セクションで、TLOGDEVICE、TLOGOFFSET、TLOGNAME、および TLOGSIZE パラメータを設定する必要があります。

2. トランザクションに参加する各マシンに対して、TLOGDEVICE の汎用デバイス・リスト・エントリ (UDL) を作成する必要もあります。このタスクは、TUXCONFIG をロードする前でも後でもかまいませんが、システムを起動する前に行う必要があります。TLOG デバイス用のエントリを UDL に作成するには、アプリケーションを起動していない状態の MASTER マシンで tmadmin -c を実行します (-c オプションを指定すると、コンフィギュレーション・モードで tmadmin が呼び出されます)。

3. 次のコマンドを入力します。

   ```
   crdl -z config -b blocks
   ```

   -z *config* には、UDL が作成されるデバイス (TLOG の常駐先デバイス) のフル・パス名を指定します。-b *blocks* には、デバイスに割り当てるブロック数を指定します。*config* の値は、MACHINES セクションの TLOGDEVICE パラメータの値と一致する必要があります。ブロック数は、TLOGSIZE より大きくなければなりません。-z を指定しない場合は、デフォルトとして *config* の値に FSCONFIG 変数の値 (アプリケーションのデータベースを指す) が使用されます。

4. グローバル・トランザクションに参加する各マシンで、手順 1 と 2 を繰り返します。

TLOGDEVICE が 2 つのマシン間でミラーリングされる場合は、片方のマシンに対して手順 4 を実行する必要はありません。TLOG を障害から回復できるようにするには、TLOG をミラーリング可能なデバイス上に配置する必要があります。TLOG は、ディスク・パーティション全体を割り当てるほど大きくないため、(通常 は 100 ページ程度)、一般には BEA Tuxedo /Q データベースと同じ raw ディスク・スライスに格納されます。

アプリケーションの起動

すべての準備が整ったら、tmboot を使ってアプリケーションを起動できます。tmboot(1) を実行できるのは、TUXCONFIG ファイルを作成した管理者だけです。

アプリケーションは通常、コンフィギュレーション・ファイルの RESOURCES セクションで MASTER として指定されたマシン、または MASTER マシンとして動作する BACKUP マシンから起動します。これ以外のマシンから起動する場合は、-b オプションを使用します。tmboot の実行時に実行可能ファイルが検索されるようにするには、BBL などの BEA Tuxedo システムのプロセスが $TUXDIR/bin になければなりません。アプリケーション・サーバは、コンフィギュレーション・ファイルの APPDIR で定義されたディレクトリになければなりません。

tmboot は、アプリケーション・サーバの起動時に、コンフィギュレーション・ファイルの CLOPT、SEQUENCE、SRVGRP、SRVID、および MIN パラメータを使用します。SEQUENCE オプションを指定した場合、アプリケーション・サーバは SEQUENCE パラメータに指定された順序で起動します。SEQUENCE を指定しない場合、サーバはコンフィギュレーション・ファイルに記述されている順序で起動します。コマンド行の形式は、以下のとおりです。

```
$ tmboot  [-g grpname] [-o sequence] [-S] [-A] [-y]
```

表 1-2tmboot のオプション

オプション	処理内容
-g *grpname*	この *grpname* パラメータを使用して、グループ内のすべての TMS とアプリケーション・サーバを起動します。
-o *sequence*	SEQUENCE パラメータで指定された順序ですべてのサーバを起動します。
-s *server-name*	サーバを個別に起動します。
-S	SERVERS セクションに記述されているすべてのサーバを起動します。
-A	MACHINES セクションに記述されているマシンのすべての管理サーバを起動します。このオプションにより、DBBL、BBL、および BRIDGE プロセスは、正しい順序で開始されるようになります。
-y	管理サーバとアプリケーション・サーバをすべて起動するかどうかを確認するプロンプトに対し、自動的に「yes (はい)」で応答します。この応答は、コマンドの範囲を制限するオプション (-g *grpname* など) を指定しない場合にのみ表示されます。

注記　tmboot オプションの総合一覧については、tmboot(1) のリファレンス・ページを参照してください。

小規模アプリケーション (マシン 2 台による構成) で tmboot を実行する

コンフィギュレーション全体を起動するには、次のコマンドを入力します。

```
prompt> tmboot -y
```

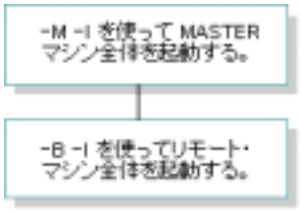
tmboot を実行すると、次の処理が実行されます。

図 0-2 小規模アプリケーションの起動手順 (デフォルト)

大規模アプリケーション (50 台を超えるマシン構成) で tmboot を実行する

大規模アプリケーション (50 台を超えるマシン構成) の場合は、1 つのステップですべてのマシンを起動します。前に述べたデフォルト手順 (マシン 2 台の構成の場合) のすべてのステップは実行しません。次の図は、最適化された起動手順を示しています。

図 0-3 大規模アプリケーションの起動手順



注記　この起動手順では、システム・メッセージの数が大幅に少ないため、大規模アプリケーションをより高速に処理できます。この方法では、起動にかかる時間を半分に短縮できます。ネットワークの速度が遅いコンフィギュレーションでは、MASTER マシンに高速で接続できるマシンを最初に起動することにより、起動時間を短縮できます。

アプリケーションのシャットダウン

tmshutdown(1) コマンドを使用して、BEA Tuxedo アプリケーションの一部または全体をシャットダウンします。このコマンドの使用上の規則は、tmboot(1) と似ています。tmshutdown は tmboot とは逆の処理を行います。

アプリケーション全体がシャットダウンされると、tmshutdown は BEA Tuxedo システムに関連付けられた IPC 資源を削除します。起動する範囲を指定するための tmboot のオプション (-A、-g、-I、-S、-s、-l、-M、-B) は、tmshutdown でも使用できます。MASTER マシン以外から tmboot を使用するた

めの -b オプションは、tmshutdown ではサポートされていません。つまり、tmshutdown コマンドは、必ず MASTER (または BACKUP MASTER) マシンで実行する必要があります。

サーバを移行するには、-R オプションを使用します。このオプションは、掲示板のエントリを削除せずにサーバをシャットダウンします。マシンが分断されている場合、tmshutdown に -P LMID オプションを指定して、分断されたマシン上で実行することにより、そのマシン上のサーバをシャットダウンすることができます。

tmshutdown を実行しても、クライアントの接続先であるマシンの管理サーバの BBL はシャットダウンされません。ただし、-c オプションを使用すると、この特性を上書きできます。直ちにマシンを停止する必要がある場合や、クライアントと通信できない場合に、このオプションを使用してください。

-w delay オプションを使用すると、*delay* で指定した秒数が経過した後で強制的にシャットダウンを実行できます。このオプションにより、すべてのサーバが直ちに中断されます。以降の作業がキューに登録されることはありません。*delay* には、キューに登録済みのリクエストを処理するための時間を指定します。*delay* で指定した秒数が経過すると SIGKILL 信号がサーバに送られます。管理者はこのオプションを使用して、アプリケーション・コードでループ状態にあるかまたはブロックされているサーバをシャットダウンすることができます。

tmshutdown の実行

tmshutdown(1) を実行できるのは、TUXCONFIG ファイルを作成した管理者だけです。アプリケーションのシャットダウンは、コンフィギュレーション・ファイルで MASTER として指定されているマシンからのみ実行できます。BACKUP マシンが MASTER マシンとして動作している場合、シャットダウン・プロセスでは、このマシンが MASTER マシンと見なされます。ただし、分断されたマシンは例外です。-p オプションを使用すると、管理者は分断されたマシンから tmshutdown コマンドを実行して、そのサイトのアプリケーションをシャットダウンできます。

アプリケーション・サーバは、SEQUENCE パラメータまたはコンフィギュレーション・ファイルで指定された順序とは逆の順序でシャットダウンされます。SEQUENCE で順序が指定されたサーバと、順序が指定されていないサーバが混在する場合、まず番号が指定されていないサーバがシャットダウンされ、次に SEQUENCE 番号が指定されたアプリケーション・サーバが (逆の順序で) シャットダウンされます。最後に、管理サーバがシャットダウンされます。

アプリケーションがシャットダウンされる場合と、BEA Tuxedo システムが割り当てた IPC 資源はすべて削除されます。tmshutdown により、DBMS が割り当てた IPC 資源が削除されることはありません。

アプリケーションが正常にシャットダウンできない場合に IPC ツールを使用する

IPC 資源とは、メッセージ・キュー、共用メモリ、セマフォなどのオペレーティング・システムの資源のことです。tmshutdown コマンドを使って BEA Tuxedo アプリケーションが正常にシャットダウンされると、BEA Tuxedo アプリケーションで使用される IPC 資源は、すべてシステムから削除されます。ただし、アプリケーションが正常にシャットダウンされず、システムに IPC 資源が残る場合もあります。このような場合は、アプリケーションを再起動できなくなります。

この問題の解決策として、IPCS コマンドを実行するスクリプトを使用して IPC 資源を削除し、特定のユーザが保有するすべての IPC 資源を解放する方法があります。しかし、この方法では IPC 資源の識別が困難です。たとえば、特定の BEA Tuxedo アプリケーションに属する資源か、BEA Tuxedo システムとは無関係の資源かを識別することができません。誤って IPC 資源を削除するとアプリケーションが破損する可能性があるため、資源の種類を識別できることは重要です。

BEA Tuxedo の IPC ツール (tmipcrm(1) コマンド) を使用すると、稼働中のアプリケーションで BEA Tuxedo システムによって割り当てられている IPC 資源 (コア・システムと Workstation コンポーネントのみ) を削除できます。

IPC 資源を削除するコマンド tmipcrm(1) は、TUXDIR/bin に格納されています。このコマンドは、バイナリ形式のコンフィギュレーション・ファイル (TUXCONFIG) を読み込み、このファイルの情報を使用して掲示板に書き込みます。tmipcrm を使用できるのは、ローカル・サーバ・マシンに対してのみです。BEA Tuxedo のコンフィギュレーションのリモート・マシンにある IPC 資源は削除できません。

このコマンドを実行するには、次のコマンド行を入力します。

```
tmipcrm [-y] [-n] [TUXCONFIG_file]
```

IPC ツールを使用すると、BEA Tuxedo システムで使用されるすべての IPC 資源を一覧表示したり、IPC 資源を削除することができます。

注記　このコマンドは、TUXCONFIG 環境変数を正確に設定するか、またはコマンド行で適切な TUXCONFIG ファイルを指定しないと利用できません。

/Q の IPC 資源を削除するには、qmadmin(1) ipcrm コマンドを使用します。

2 BEA Tuxedo アプリケーションの監視

ここでは、次の内容について説明します。

- アプリケーションの監視方法
- 適切な監視ツールを選択する
- BEA Administration Console を使用してアプリケーションを監視する
- コマンド行ユーティリティを使用してアプリケーションを監視する
- イベント・ブローカを使用してアプリケーションを監視する
- ログ・ファイルを使用してアクティビティを監視する
- MIB を使用してアプリケーションを監視する
- 実行時のトレース機能を使用する
- DBBL および BBL を使用してエラーを管理する
- ATMI を使用してシステム・エラーとアプリケーション・エラーを処理する
- マルチスレッド化またはマルチコンテキスト化されたアプリケーションを監視する

アプリケーションの監視方法

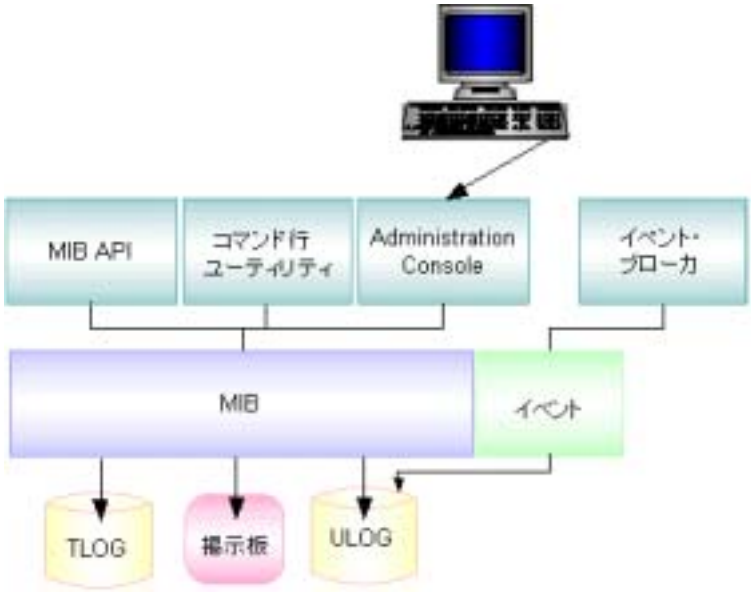
システム管理者は、一度アプリケーションが起動して実行されたら、そのアプリケーションが実行中は企業の求めるパフォーマンス、可用性、セキュリティの各条件を満たすように管理する必要があります。このため、資源 (共用メモリなど)、アクティビティ (トランザクション処理など)、および発生する可能性のある問題 (セキュリティ違反など) について監視し、必要に応じて適切な処理を実行する必要があります。

管理者のこのような責務を実行するため、BEA Tuxedo システムには、システム・イベントおよびアプリケーション・イベントを監視し、システムを再コンフィギュレーションして性能を高めるための方法が用意されています。システムがどのように動作しているかを監視するには、以下の機能を使用します。

- BEA Tuxedo Administration Console
- コマンド行ユーティリティ
- ログ・ファイル
- ATMI
- MIB
- 実行時のトレース機能

これらのツールを使用すると、変化し続けるビジネス上のニーズや、システムで発生した障害に対して、迅速かつ効率的に対応できます。また、これらのツールを使用して、アプリケーションの性能やセキュリティを管理することもできます。

図 2-1 監視ツール


MIB API
コマンド行
ユーティリティ
Administration
Console
イベント・
ブローカ
MIB
イベント
TLOG
掲示板
ULOG

BEA Tuxedo システムには、アプリケーションを監視するための次のようなツールが用意されています。

- BEA Administration Console—アプリケーションの動作を監視し、その操作を動的にコンフィギュレーションするための Web 対応のグラフィカル・ユーザ・インターフェイス。コンフィギュレーション情報の表示や変更を行ったり、システムの各コンポーネントの状態を決定したり、実行済みリクエストやキュー内のリクエストなどに関する統計情報を取得することができます。
- コマンド行ユーティリティ—アプリケーションの活性化、非活性化、コンフィギュレーション、および管理に使用するためのコマンド群。`tmboot(1)`、`tmadmin(1)`、`tmshutdown(1)` などがあります。
- イベント・ブローカ—システム障害や例外的な問題 (ネットワーク障害など) を管理者に通知するメカニズム。クライアントまたはサーバからイベントがポストされると、イベント・ブローカはポストされたイベント名をそのイベント用のサブスクライバ・リストと照合し、サブスクリプションで定義されている適切なアクションを実行します。
- ログ・ファイル—エラー・メッセージ、警告メッセージ、デバッグ・メッセージ、および通知メッセージが格納されたファイル群。システムで発生した問題をトラッキングし、解決するときに参照します。
- MIB—MIB 内の情報にアクセスしたり、これらの情報を変更するためのプロシージャに対するインターフェイス。MIB を使用すると、実行時のアプリケーションを監視するためのプログラムを作成できます。
- 実行時のトレース機能—アプリケーションの実行処理をトラッキングするソフトウェア。トラッキングした情報を使用して、システムの問題を解決できます。

関連項目

- 2-5 ページの「システム・データとアプリケーション・データの監視」
- 2-10 ページの「適切な監視ツールを選択する」
- 2-11 ページの「BEA Administration Console を使用してアプリケーションを監視する」

- 『BEA Tuxedo システム入門』の 4-4 ページの「BEA Tuxedo Administration Console を使用する利点」
- 2-13 ページの「コマンド行ユーティリティを使用してアプリケーションを監視する」
- 2-18 ページの「イベント・ブローカを使用してアプリケーションを監視する」
- 2-19 ページの「ログ・ファイルを使用してアクティビティを監視する」
- 2-33 ページの「ATMI を使用してシステム・エラーとアプリケーション・エラーを処理する」
- 2-26 ページの「MIB を使用してアプリケーションを監視する」
- 『BEA Tuxedo システム入門』の 4-15 ページの「MIB を使用した操作の管理」
- 2-29 ページの「実行時のトレース機能を使用する」
- 『BEA Tuxedo コマンド・リファレンス』の `tmshutdown(1)`
- 『BEA Tuxedo システム入門』の 4-1 ページの「BEA Tuxedo の管理ツール」

システム・データとアプリケーション・データの監視

BEA Tuxedo システムを使用すると、システム・データおよびアプリケーション・データの監視を行うことができます。

システム・データを監視する

実行中のシステムを監視するため、BEA Tuxedo システムでは、次のシステム・コンポーネントに関するパラメータ設定を管理し、統計情報を生成します。

- クライアント
- 会話
- グループ
- メッセージ・キュー
- ネットワーク
- サーバ
- サービス
- CORBA インターフェイス
- トランザクション

これらのコンポーネントには、MIB または `tmadmin` を使用してアクセスできます。また、管理者側で操作は行わず、掲示板の統計情報に基づいてシステム・コンポーネントが動的に変更されるようにシステムを設定することもできます。システムが正しくコンフィギュレーションされている場合は、次の機能を実行できます (掲示板で次の機能が指定されている場合)。

- ロード・バランシングの有効化
- 新しいサーバの起動
- 使用されていないサーバのシャットダウン

システムの管理データを監視することにより、アプリケーションの性能、可用性、およびセキュリティの低下の原因となる問題を防止したり、解決することができます。

システム・データの格納場所

システムを監視するために必要な情報を格納する場所として、BEA Tuxedo システムには次の 3 つのデータ・リポジトリが用意されています。

- 掲示板―ネットワーク上の各マシンにある共用メモリのセグメント。コンフィギュレーションのコンポーネントやアクティビティに関する統計情報はここに書き込まれます。
- ログ・ファイル―システム生成されたメッセージが書き込まれるファイル。
- UBBCONFIG―システムとアプリケーションのパラメータを定義するテキスト形式のファイル。

静的および動的な管理データを監視する

実行中の BEA Tuxedo システムでは、静的データと動的データという 2 種類の管理データを監視できます。

静的データとは

コンフィギュレーションの静的データとは、システムとアプリケーションを最初にコンフィギュレーションしたときに割り当てる設定値のことです。これらの設定値は変更されません (ただし、リアルタイムで変更したり、プログラムを使用して変更する場合は例外)。たとえば、システム全体に関するパラメータ (使用するマシン数など) やローカル・マシンのシステムに割り当てられた IPC 資源 (共用メモリなど) の容量は静的データです。静的データは、UBBCONFIG ファイルと掲示板に格納されます。

静的データの確認

コンフィギュレーションに関する静的データの確認が必要な場合があります。たとえば、コンフィギュレーション (または掲示板のマシン・テーブル) で許可されている最大マシン数の範囲内で、できるだけ多くのマシンを追

加したい場合などです。この場合、システム全体に関するパラメータ (MAXMACHINES など) の現在の値を確認して、マシンの最大数を調べることができます。

システムをチューニングして、アプリケーションの性能を向上することもできます。チューニングが必要かどうかを決定するには、現在利用可能なローカル IPC 資源の容量を確認します。

動的データとは

コンフィギュレーションの動的データとは、アプリケーションの実行中にリアル・タイムで変更される情報のことです。たとえば、負荷 (サーバに送信されたリクエスト数) やコンフィギュレーションでのコンポーネント (サーバなど) の状態は、頻繁に変化します。動的データは、掲示板に格納されます。

動的データの確認

コンフィギュレーションの動的データは、管理上の問題を解決する場合に便利です。次に、具体的な例を 2 つ挙げます。

たとえば、スループットが低下し、現在接続中のクライアント数に対応できる十分なサーバが稼動しているかどうかを調べるとします。この場合、まず、実行中のサーバ数、接続されているクライアント数、および 1 つまたは複数のサーバの負荷を確認します。これらの値を確認すると、サーバの追加により性能を向上できるかどうかを判断できます。

また、アプリケーションに対して特定の要求を行ったときの応答が遅いというクレームを複数受け取ったとします。この場合は、負荷に関する統計情報を確認することにより、BLOCKTIME パラメータの値を増やして応答時間を短縮できるかどうかを判断することができます。

起動とシャットダウンに関する一般的な問題

BEA Tuxedo システムが正常に動作しているかどうかを確認する場合は、起動とシャットダウンに関する次の問題を考慮し、定期的にシステムを監視する必要があります。

起動時の一般的な問題

- アプリケーション・サーバが正常に起動しない、または初期化中にコア・ダンプが発生する。
- アプリケーション・サーバ・ファイルが見つからない、または実行できない。
- サーバ・グループが自動的に移行される。
- デフォルトの起動手順が最適でない。
- 環境変数が設定されていない、または正しく設定されていない。
- `IPCKEY` が既に使用されている。
- ネットワーク・アドレスが無効である。
- `UBBCONFIG` ファイルで指定されている上限値に到達した。
- ネットワーク・ポートが既に使用されている。
- システム・リソースの制限値に到達した。
- サーバ起動時の依存関係に問題がある。
- `TLOG` ファイルが作成されない。

シャットダウン時の一般的な問題

- クライアントが接続されたままの状態になっている。
- サーバが停止している。
- シャットダウンの順序に問題がある。

適切な監視ツールを選択する

実行中のアプリケーションを監視するには、コンフィギュレーションの動的な局面を継続的にトラッキングし、静的なデータを不定期的に調査する必要があります。つまり、基本的には掲示板の情報に注目し、必要に応じて UBBCONFIG ファイルを参照します。次の要素を考慮した上で、適切な方法を選択してください。

- BEA Tuxedo システムの管理者としての経験。管理者としての経験が十分あり、シェル・プログラミングの専門知識がある場合は、頻繁に実行するコマンドを自動化するプログラムを作成して、アプリケーションを監視します。
- オペレーティング・システムの使用経験。十分な経験がない場合は、BEA Administration Console の使用をお勧めします。
- 表示したい情報の種類。tmadmin コマンドを実行して UBBCONFIG ファイルの RESOURCES セクションを調べ、アプリケーションを監視する場合、アクセスできるのは現在値だけです。

次の表では、各監視ツールの使用方法を説明します。

監視ツールの種類	操作方法
BEA Administration Console	グラフィカル・インターフェイスを使用します。
txrpt や tmadmin などのコマンド行ユーティリティ	プロンプトの後にコマンドを入力します。

監視ツールの種類	操作方法
イベント・ブローカ	サーバの異常終了やネットワーク障害などの BEA Tuxedo システムのイベントをサブスクライブします。
ログ・ファイル (ULOG、TLOG など)	任意のテキスト・エディタで ULOG を表示し、ULOG で tlisten を参照します。次に、tmadmin dumptlog (TLOG をテキスト形式にダウンロード) を実行して、バイナリ形式の TLOG をテキスト形式のファイルに変換します。
MIB	実行時アプリケーションを監視するプログラムを作成します。
実行時のトレース機能	トレース時の表現 (カテゴリ、フィルタ表現、アクション) を指定し、TMTRACE 環境変数を有効にします。詳細については、2-29 ページの「実行時のトレース機能を使用する」を参照してください。

BEA Administration Console を使用してアプリケーションを監視する

BEA Administration Console は、アプリケーションをチューニングしたり、変更するときに使用する MIB 用のグラフィカル・ユーザ・インターフェイスです。World Wide Web 経由でアクセスし、Web ブラウザを使用して表示します。管理者は、サポートされているブラウザを使用して BEA Tuxedo アプリケーションを監視できます。

ツールバーを使用してアクティビティを監視する

ツールバーには、管理や監視を行うツールを実行するための 12 のボタンが用意されています。各ボタンには、アイコンと名前が付いています。監視用のボタンは次のとおりです。

- Logfile—アクティブ・ドメイン内の特定のマシンから ULOG ファイルを表示します。
- Event Tool—システム・イベントを監視します。[Event Tool] ボタンをクリックすると、次の 4 つのオプションがあるウィンドウが表示されます。[subscribe] は、指定されたシステム・イベントの通知を要求します。[unsubscribe] は、指定されたシステム・イベントの通知を拒否します。[snapshot] は、Event Tool で現在保留中のデータのレコードを作成します。[select format] は、Event Tool で収集された情報に関するパラメータを選択します。
- Stats—BEA Tuxedo システムのアクティビティをグラフィカルに表示します。
- Search—ツリー内で特定のオブジェクト・クラスまたはオブジェクトを検索します。

関連項目

- 『BEA Tuxedo システム入門』の 4-4 ページの「BEA Tuxedo Administration Console を使用した管理操作」

コマンド行ユーティリティを使用してアプリケーションを監視する

コマンド行インターフェイスからアプリケーションを監視するには、tmadmin(1) コマンドまたは txrpt(1) コマンドを使用します。

tmadmin を使用してコンフィギュレーションを調べる

tmadmin コマンドは、掲示板とその関連エンティティを表示したり、変更するための 53 種類のコマンドのインタプリタです。tmadmin コマンドを使用すると、サービスの状態などのシステムの統計情報、実行済みのリクエスト数、キューに登録されているリクエスト数などを監視できます。

tmadmin コマンドを使用すると、BEA Tuxedo システムを動的に変更できます。たとえば、システムの実行中に次のような変更を行うことができます。

- サービスの中断および再開
- サービスの宣言および宣言解除
- サービス・パラメータの変更
- AUTOTRAN のタイムアウト値の変更

tmadmin セッションの開始時には、そのセッションの動作モードを選択できます。動作モードには、動作モード (デフォルト)、読み取り専用モード、コンフィギュレーション・モードがあります。

- デフォルトの動作モードでは、管理者権限 (管理者用の有効な UID または GID) があれば、tmadmin セッション中に掲示板のデータを表示したり、変更することができます。
- 読み取り専用モードでは、掲示板のデータを表示することはできますが、データを変更することはできません。読み取り専用モードの利点は、管

理者のプロセスが tmadmin によって拘束されないことです。tmadmin プロセスはクライアントとして掲示板を参照するため、管理者用のスロットはほかの作業に使用できます。

- コンフィギュレーション・モードでは、掲示板のデータを表示できます。BEA Tuxedo アプリケーションの管理者であれば、データを変更することもできます。コンフィギュレーション・モードでの tmadmin セッションの開始は、どのマシン (非アクティブなマシンも含む) からでも行えます。非アクティブなマシンでは通常、tmadmin を実行するためコンフィギュレーション・モードである必要があります。(コンフィギュレーション・モードを指定せずに tmadmin セッションを開始できる非アクティブなマシンは、MASTER マシンだけです)。

注記　BEA Tuxedo のバージョンとライセンス数に関するレポートを作成することもできます。

txrpt を使用してサーバとサービスに関するレポートを作成する

txrpt コマンドを実行すると、BEA Tuxedo サーバの標準エラー出力の分析が行われ、そのサーバでサービス処理にかかる時間のサマリが生成されます。レポートには、指定した期間内に各サービスのディスパッチが何回行われたか、また、各サービスではリクエストの処理にどれだけの時間がかかったかが記録されます。txrpt は、標準入力または入力用にリダイレクトされた標準エラー・ファイルからの入力を読み込みます。標準エラー・ファイルを作成するには、servopts(5) の選択肢から -r オプションを使用してサーバを起動します。-e servopts オプションを使用すると、ファイル名を指定できます。複数のファイルを、txrpt 用に 1 つの入力ストリームに連結することができます。

サービス X と、そのサービスが格納されたサーバ Y の情報がファイルに蓄積されます。txrpt を使用してこのファイルを処理すると、サービスのアクセス状況とサーバの処理時間に関するレポートが生成されます。

関連項目

- 2-2 ページの「アプリケーションの監視方法」
- 2-15 ページの「tmadmin セッションのしくみ」
- 2-17 ページの「tmadmin コマンドを使用してシステムを監視する」
- 『BEA Tuxedo システム入門』の 4-11 ページの「コマンド行ユーティリティを使用した操作の管理」

tmadmin セッションのしくみ

`tmadmin` コマンドは、掲示板とその関連エンティティを表示したり、変更するための 53 種類のコマンドのインタプリタです。次の図は、典型的な `tmadmin` セッションの動作の例です。

図 2-2 典型的な tmadmin セッション

tmadmin コマンドを使用してシステムを監視する

以下は、tmadmin コマンドを使用して監視できる、実行時のシステムの機能の一覧です。

- サービスにインストールされたサーバの数
- 負荷分散が適切であるかどうか
- 特定のサービスが実行中かどうか
- 非アクティブなクライアントがあるかどうか
- 作業は、システム全体でスムーズに流れるように分散されているか
- クライアントが接続を占有しているために、サーバがほかのクライアントの処理を行えないかどうか
- ネットワークが安定しているかどうか
- 手動でトランザクションをコミットまたはアボートする必要があるかどうか
- ローカル・マシン上の共用メモリやセマフォなどのオペレーティング・システムの資源が十分かどうか

関連項目

- 『BEA Tuxedo コマンド・リファレンス』の tmadmin(1)

イベント・ブローカを使用してアプリケーションを監視する

BEA Tuxedo のイベント・ブローカは、実行中のアプリケーションで発生するイベントを監視します。イベントとは、たとえば、クライアントの状態がアクティブから非アクティブに変わる、という MIB オブジェクトの状態の変化のことです。イベント・ブローカがイベントを検出すると、そのイベントはレポート (ポスト) され、イベントの発生がサブスクライバに通知されます。MIB オブジェクトを表す FML データ・バッファを受け取って、MIB でのイベントの発生通知を自動的に受け取ることもできます。イベントをポストしてサブスクライバにレポートする場合、イベント・ブローカは tppost(3c) を使用します。管理者もアプリケーション・プロセスもイベントをサブスクライブできます。

イベント・ブローカは、MIB オブジェクトの 100 種類以上の状態の変化をシステム・イベントとして認識します。ポストされたシステム・イベントは、イベントが発生した MIB オブジェクトの表現と、発生したイベントを識別するイベント固有のフィールドで構成されます。たとえば、マシンが分断された場合、ポストされるイベントには次の情報が含まれます。

- T_MACHINE クラスで指定されている影響を受けるマシンの名前、およびそのマシンのすべての属性
- イベントをマシンの分断として識別するいくつかのイベント属性

システム・イベントにサブスクライブするだけで、イベント・ブローカを使用できます。

関連項目

- 『BEA Tuxedo システム入門』の 4-19 ページの「イベント・ブローカを使用したイベントの管理」

ログ・ファイルを使用してアクティビティを監視する

エラー状況を迅速かつ正確に識別するため、BEA Tuxedo システムには次のログ・ファイルが用意されています。

- トランザクション・ログ (TLOG)—トランザクション・マネージャ・サーバ (TMS: Transaction Manager Server) で使用されるバイナリ形式のファイルです。通常、ユーザや管理者が読むことはありません。TLOG は、BEA Tuxedo のグローバル・トランザクションに参加したマシンでのみ作成されます。
- ユーザ・ログ (ULOG)—アプリケーションの実行中に BEA Tuxedo システムが生成するメッセージのログ・ファイルです。

これらのログ・ファイルの管理と更新は、アプリケーションの実行中に常に行われます。

関連項目

- 2-20 ページの「トランザクション・ログ (TLOG) とは」
- 2-2 ページの「アプリケーションの監視方法」
- 2-22 ページの「ログを使用してエラーを検出する」
- 2-26 ページの「アプリケーション・サービス・ログを使用してサービスの作業負荷を予測する」

トランザクション・ログ (TLOG) とは

トランザクション・ログ (TLOG) には、コミット・フェーズのグローバル・トランザクションが記録されます。2 フェーズ・コミット・プロトコルのうち、第 1 フェーズの終了時に、グローバル・トランザクションのパーティシパントは、グローバル・トランザクションをコミットするか、またはロールバックするかを決める応答を発行します。この応答は、TLOG に記録されます。

TLOG ファイルを使用するのは、グローバル・トランザクションを調整するトランザクション・マネージャ・サーバ (TMS) だけです。管理者は参照しません。TLOG のロケーションとサイズは、UBBCONFIG ファイルの MACHINES セクションにある 4 つのパラメータで指定します。

グローバル・トランザクションに参加する各マシンに TLOG を作成する必要があります。

関連項目

- 2-22 ページの「ログを使用してエラーを検出する」

ユーザ・ログ (ULOG) とは

ユーザ・ログ (ULOG) は、BEA Tuxedo システムによって生成されるすべてのメッセージ、つまりエラー・メッセージ、警告メッセージ、情報メッセージ、デバッグ・メッセージが書き込まれるファイルです。アプリケーションのクライアントおよびサーバも、ユーザ・ログへの書き込みが可能です。ログは毎日新しく作成されます。そのため、マシンごとにログが異なる場合もあります。ただし、リモート・ファイル・システムが使用されている場合、ULOG を複数のマシンで共有できます。

ULOG によって管理者に提供されるシステム・イベントの記録から、BEA Tuxedo システムおよびアプリケーションのほとんどの障害の原因を特定できます。ULOG はテキスト・ファイルなので、任意のテキスト・エディタで表示できます。ULOG には、tlisten プロセスによって生成されるメッセージも挿入されています。tlisten プロセスにより、アプリケーション内のほかのマシンにあるリモート・サービスに接続できます。マスタ・マシンを含め各マシンで tlisten プロセスが実行されていることが必要です。

ログを使用してエラーを検出する

BEA Tuxedo のログ・ファイルを使用して、アプリケーション・エラーおよびシステム・エラーを検出できます。

トランザクション・ログ (TLOG) を分析する

TLOG は、コミット処理中のグローバル・トランザクションに関するメッセージのみを含むバイナリ形式のファイルです。TLOG を表示するには、まずテキスト形式に変換する必要があります。BEA Tuxedo システムでは、tmadmin を使った 2 つの方法で TLOG を変換できます。

- dumptlog (dl) を実行します。バイナリ形式の TLOG はテキスト・ファイルにダウンロード (ダンプ) されます。
- loadtlog を実行します。テキスト形式の TLOG は、既存のバイナリ形式の TLOG にアップロード (ロード) されます。

dumptlog コマンドおよび loadtlog コマンドは、サーバ・グループやマシンを移行するときに、マシン間で TLOG を移動する場合にも便利です。

トランザクション・エラーを検出する

MIB T_TRANSACTION クラスを使用して、システム内の実行時のトランザクション属性を取得できます。この情報は、tmadmin コマンドの printtrans (pt) を使用して表示することもできます。トランザクション内の各グループに関する情報は、tmadmin を冗長モードで実行した場合にのみ表示されます。冗長モードで実行するには、それ以前に verbose (v) コマンドを実行しておく必要があります。

トランザクションのコミット・プロセスの間に発生した重大なエラー (TLOG への書き込みの失敗など) は、USERLOG に書き込まれます。

ユーザ・ログ (ULOG) を分析する

BEA Tuxedo システムでは、アプリケーション内のアクティブなマシンにログ・ファイルが置かれます。このログ・ファイルには、BEA Tuxedo システムのエラー・メッセージ、警告メッセージ、デバッグ・メッセージ、またはそれ以外の役立つ情報が含まれています。このファイルをユーザ・ログ (ULOG) と呼びます。ULOG を使用すると、BEA Tuxedo ATMI によって返されるエラーを簡単に見つけることができます。また、ULOG は、BEA Tuxedo システムとアプリケーションで生成されたエラー情報の主要な格納先となります。

ULOG の情報を使用すると、システムやアプリケーションの障害の原因を特定できます。ユーザ・ログには、1 つの問題に対して複数のメッセージを記録できます。一般的には、先に記述されたメッセージの方が有益な診断情報を説明しています。

ULOG メッセージの例

次の例では、LIBTUX_CAT カタログのメッセージ 358 が、その後のメッセージで報告される問題の原因を説明しています。つまり、UNIX システム・セマフォの不足が、アプリケーションを起動できない原因であることを示しています。

コード リスト 2-1　ULOG **メッセージの例**

```
151550.gumby!BBL.28041.1.0:LIBTUX_CAT:262: std main starting
151550.gumby!BBL.28041.1.0:LIBTUX_CAT:358: reached UNIX limit on semaphore ids
151550.gumby!BBL.28041.1.0:LIBTUX_CAT:248:fatal: system init function ...
151550.gumby!BBL.28040.1.0:CMDTUX_CAT:825: Process BBL at SITE1 failed ...
151550.gumby!BBL.28040.1.0:WARNING: No BBL available on site SITE1.
Will not attempt to boot server processes on that site.
```

注記　ユーザ・ログ・メッセージの詳細と、問題に対する適切な対応策については、『システム・メッセージ』を参照してください。

ULOG 内の tlisten メッセージを分析する

ULOG には、tlisten プロセスのエラー・メッセージも記録されます。tlisten プロセスのメッセージは、任意のテキスト・エディタを使用して表示できます。MASTER マシンを含む各マシン上で、個別に tlisten プロセスが実行されます。それぞれの tlisten プロセスのログは、各マシン上の ULOG に記録されますが、リモート・ファイル・システムから共有で使用することもできます。

ULOG は、tlisten プロセスのエラーを記録します。tlisten が使用されるのは、tmboot を使用してシステムを起動するとき、およびアプリケーションの実行中に tmadmin を実行するときです。tlisten のメッセージは、tlisten プロセスの実行後、直ちに作成されます。tlisten プロセスでエラーが発生すると、メッセージが ULOG に記録されます。

注記　アプリケーションの管理者には ULOG の tlisten メッセージを分析する責任がありますが、プログラマも ULOG のメッセージを活用できます。

BEA Tuxedo System Messages の「CMDTUX カタログ £ には、tlisten のメッセージに関する次の情報が説明されています。

- すべてのメッセージの説明
- メッセージ内で報告されたエラー状況に対し、管理者 (またはプログラマ) が行うよう推奨されている処理

tlisten メッセージの例

ULOG 内の次の tlisten メッセージを例に取ります。

```
121449.gumby!simpserv.27190.1.0:LIBTUX_CAT:262:std main starting
```

ULOG のメッセージは、タグとテキストで構成されています。タグは、次の情報で構成されています。

- 時刻を表す 6 桁の文字列 (hhmmss)。先頭から順に、時、分、秒を表します。
- マシン名 (UNIX システムで `uname -n` コマンドを実行すると返される名前)。
- メッセージを記録しているプロセスの名前と識別子。このプロセス ID には、オプションでトランザクション ID を含めることもできます。スレッド ID (1) と コンテキスト ID (0) も含まれます。

  注記　シングルスレッドおよびシングルコンテキストのアプリケーションの場合、`thread_ID` フィールドと `context_ID` フィールドにプレースホルダが出力されます。アプリケーションがマルチスレッドであるかどうかは、複数のスレッドを使用するまでわかりません。

テキストは、次の情報で構成されています。

- メッセージ・カタログの名前
- メッセージ番号
- BEA Tuxedo システム・メッセージ

注記　このメッセージについては、BEA Tuxedo System Messages の「LIBTUX カタログ」を参照してください。

関連項目

- 1-9 ページの「TLOG デバイスの作成」
- 1-10 ページの「アプリケーションの起動」
- 『BEA Tuxedo システム入門』の 3-14 ページの「BEA Tuxedo トランザクション管理サーバ」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 1-18 ページの「トランザクション」

アプリケーション・サービス・ログを使用してサービスの作業負荷を予測する

BEA Tuxedo アプリケーション・サーバでは、処理対象のサービス要求のログを生成することができます。このログは、サーバの標準出力 (stdout) に表示されます。各レコードには、サービス名、開始時刻、および終了時刻が記録されます。

このログは、サーバの起動後に要求できます。txrpt 機能を使用してサーバの実行時間に関するサマリを生成し、ログの出力結果を分析できます。このデータを使用すると、サービスごとの相対的な作業負荷を見積もることができ、MIB 内の該当するサービスに対して作業負荷パラメータを適切に設定できます。

MIB を使用してアプリケーションを監視する

MIB を使用すると、2 種類の操作を実行できます。1 つ目は、get 操作を使用した MIB からの情報の取得です。2 つ目は、set 操作を使用した MIB の情報の更新です。これらの操作は、ATMI 関数 (tpalloc(3c)、tprealloc(3c)、tpcall(3c)、tpacall(3c)、tpgetrply(3c)、tpenqueue(3c)、tpdequeue(3c) など) を使用していつでも実行できます。

get 操作を使って MIB を照会すると、MIB は一致したオブジェクトをいくつか返し、そのほかの候補数を示します。MIB は、残りのオブジェクトを取得するためのハンドル (カーソル) を返します。次のオブジェクトのセットを取得する操作は、getnext です。照会したオブジェクトが複数のバッファに渡ると、3 つ目の操作が発生します。

制限を指定して MIB を照会する

仮想データベースである MIB を照会するとは、データベース・テーブルからレコードのセットを選択することです。データベース・テーブルのサイズを制限する方法は 2 つあります。つまり、照会に対して返すオブジェクト数を制限するか、または各オブジェクトの情報量を制限します。キー・フィールドとフィルタを使用して、リクエストの範囲を必要なデータだけに制限することもできます。制限を多く指定すればするほど、アプリケーションから要求される情報は少なくなり、応答は早くなります。

グローバル・データおよびローカル・データを照会する

MIB のデータは、いくつかの場所に分けて保存されています。分散アプリケーションでは、複数のマシンに複製されているデータもあります。また、複製はされず、データの属性やデータが表すオブジェクトにより、特定のマシンのローカル・データになるデータもあります。

グローバル・データとは

グローバル・データとは、サーバなどのアプリケーション・コンポーネントに関する情報で、アプリケーション内のすべてのマシン上に複製されます。サーバに関するほとんどのデータ (コンフィギュレーションや状態に関する情報など) は、アプリケーションにグローバルに複製され、特にすべての掲示板に複製されます。BEA Tuxedo アプリケーションでは、どこからでもこの情報にアクセスできます。

たとえば、管理者は、Customer Orders という名前のアプリケーション内の任意のマシンから、サーバ B6 が Group1 に属しており、CustOrdA というマシン上で起動しており、アクティブであることを確認できます。

ローカル・データとは

ローカル・データとは、グローバルには複製されない、エンティティに対してローカルなデータ (サーバの統計など) のことです。ローカルな属性の例は TA_TOTREQC です。この属性は、特定のサーバ上でサービスが処理される回数を定義します。この統計は、ホスト・マシン上のサーバに保存されます。サーバがサービス要求を受信し、そのサービスが処理されると、カウンタの値が増えます。この種の情報はローカルで管理されるため、複製するとシステムの性能が低下します。

MIB には、ローカル (クライアントなど) にしかないクラスもあります。クライアントがログインすると、掲示板にクライアント用のエントリが作成され、ここにクライアントに関するトラッキング情報が記録されます。MIB は、このエントリを調べることにより、いつでもクライアントの状態を判別できます。

tpadmcall を使用して情報にアクセスする

BEA Tuxedo システムには、アプリケーションが稼動していない場合に MIB に直接アクセスするためのプログラミング・インターフェイスが用意されています。これが tpadmcall 関数であり、MIB を構成するデータにアプリケーションから直接アクセスするときに使用します。tpadmcall を使用すると、ローカル情報のサブセットにアクセスできます。

tpadmcall は、システムが稼動していないときに、システムを照会したり、管理上の変更を行う場合に使用してください。tpadmcall は、リクエストを送信する代わりに TUXCONFIG ファイルを照会します。送信用と受信用のデータ・バッファ (照会と応答の内容を格納) は、まったく同じです。

関連項目

- 『BEA Tuxedo システム入門』の 4-15 ページの「MIB を使用した操作の管理」
- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の MIB(5)

ud32 を使用して MIB の照会と更新を行う

ud32 は、BEA Tuxedo システムに付属のクライアント・プログラムであり、FML バッファ型のテキスト表現で構成される入力情報を読み込みます。ud32 は、MIB に対して特殊な照会を行ったり、MIB を更新するときに使用できます。ud32 を実行すると、FML32 バッファが作成され、バッファを指定したサービス呼び出しが行われます。サービス呼び出しに対する応答 (FML32 バッファ型) を受信すると、結果は画面上またはテキスト・ファイルに出力されます。

ud32 は、テキスト形式の FML フィールドと値を使用して FML32 型のバッファを作成し、バッファ内で指定されたサービスを呼び出して、応答を待ちます。応答は、レポートとして FML32 形式で返されます。MIB は FML32 をベースにしているため、ud32 は MIB のスクリプト作成用ツールになります。

たとえば、次のテキストを含む小さいファイルを作成するとします。

```
service name=.tmib and ta_operation=get, TACLASSES=T_SERVER
```

ud32 にこのファイルを入力すると、サーバに関するすべてのシステム・データが一覧表示された FML 出力バッファが返されます。

実行時のトレース機能を使用する

BEA Tuxedo システムには、分散型ビジネス・アプリケーションの実行をトラッキングする実行時のトレース機能があります。システムには、さまざまなカテゴリの関数の呼び出しを記録するためのトレース・ポイントが組み込

まれています。たとえば、アプリケーションによって発行された ATMI 関数、X/Open 準拠のリソース・マネージャに対する BEA Tuxedo システムからの XA 関数などです。

トレース機能を有効にするには、カテゴリ、フィルタ式、およびアクションを含むトレース式を指定する必要があります。カテゴリには、トレース対象の関数の種類 (ATMI など) を指定します。フィルタ式には、アクションをトリガした特定の関数を指定します。アクションには、BEA Tuxedo システムによって指定された関数に対する応答を指定します。たとえば、ULOG へのレコードの書き込み、システム・コマンドの実行、トレース処理の終了、などのアクションを指定できます。クライアント・プロセスでは、リクエストを使用してトレース機能を複製転送することもできます。この機能は「ダイイング」と呼ばれます。つまり、トレース処理により、クライアントから呼び出されたすべてのサービスを「色付けする」ことを意味します。

トレース式は、TMTRACE 環境変数を設定するか、またはサーバ環境で式を指定することにより設定できます。

- トレース式を簡単に指定するには、クライアント環境で TMTRACE=on と定義します。この式により、クライアントまたはクライアントの代わりにサービスを実行する任意のサーバで、ATMI 関数をトレースできます。トレース・レコードは ULOG ファイルに書き込まれます。
- また、サーバ環境でトレース式を指定することもできます。たとえば、TMTRACE=atmi:/tpservice/ulog のように入力します。. この設定をサーバ環境内でエクスポートすると、サーバ上でサービスが実行されるたびに ULOG ファイルにレコードが記録されます。

tmadmin の changetrace コマンドを使用すると、トレース・オプションをアクティブにしたり非アクティブにできます。このコマンドを使用すると、アクティブなクライアント・プロセスまたはサーバ・プロセスのトレース式を上書きできます。管理者は、すべてのクライアントとサーバに対してグローバルなトレース機能を許可することも、特定のマシン、グループ、またはサーバだけがトレース機能を実行できるように設定することもできます。

関連項目

- 2-2 ページの「アプリケーションの監視方法」

- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の tmtrace(5)

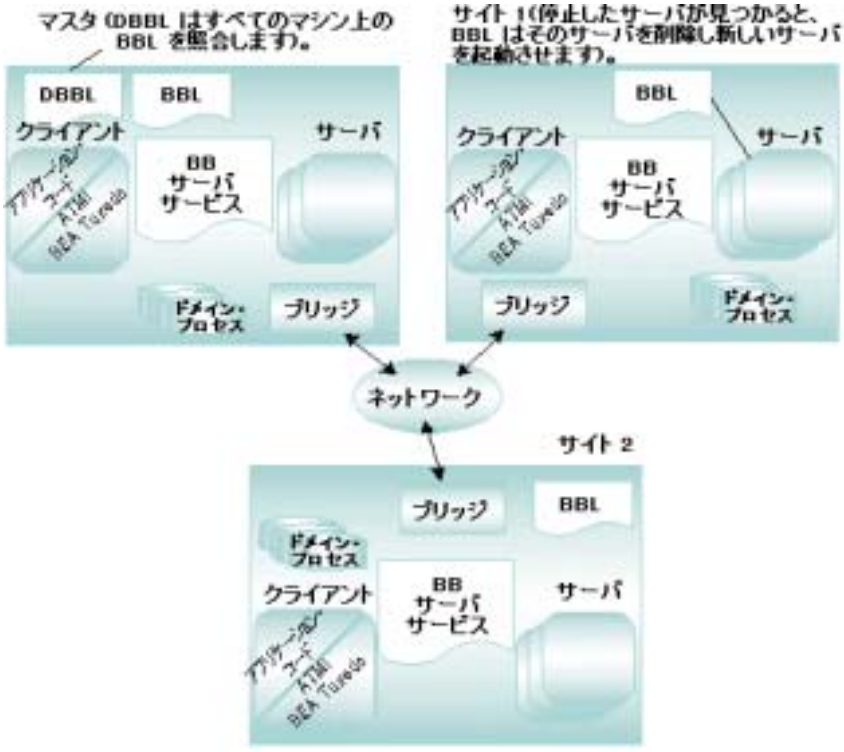
DBBL および BBL を使用してエラーを管理する

BEA Tuxedo システムでは、次の 2 つの管理サーバを使用して、掲示板にある情報をアプリケーション内のすべてのマシンに分散します。

- DBBL—特別掲示板連絡 (DBBL) サーバは、MIB に対するグローバルな変更を複製転送し、MIB の静的な部分を保ちます。DBBL の特徴は次のとおりです。
  - MASTER マシン上に置かれ (各アプリケーションに 1 つ)、すべての BBL に対して周期的にステータス・リクエストを送信します。
  - 掲示板の更新、さまざまなマシンの状態、および BBL に対する照会を調整します。
  - サーバの移行を調整します。
  - 障害から回復させるために他のマシンに移行できます。
- BBL—Bulletin Board Liaison の略。BBL サーバは、ホスト・マシン上の掲示板を管理します。また、ローカルな MIB に対する変更を調整し、マシン上でアクティブな状態のアプリケーション・プログラムの整合性を検証します。掲示板の特徴は次のとおりです。
  - アプリケーション内の各 BEA Tuxedo マシン上に置かれ、DBBL からの要求を実行します。また、サービス要求のタイムアウト、リクエストへの応答、およびトランザクションを管理します。
  - 検出したサーバの障害に対して、ユーザ定義された回復処理を開始し、自動的にサーバを再起動します。
  - クライアントの障害を検出します。

- クライアントとサーバのエントリ、および掲示板での会話型通信をクリーンナップします。
- DBBL の障害を検出し、回復します (MASTER マシンの BBL の場合)。

図 2-3DBBL と BBL を使用した診断および修復



どちらのサーバにも障害を管理する役割があります。DBBL はアプリケーション内のほかのアクティブ・マシンの状態を調整します。各 BBL は、MIB の状態の変更についての通信を行い、不定期的に DBBL にホスト・マシン上のすべてが順調であることを示すメッセージを送信します。

BEA Tuxedo のランタイム・システムは、ユーザ・ログ (ULOG) にイベント (システム・エラー、警告、トレース・イベントも含む) を記録します。プログラマは、ULOG を使用して、アプリケーションをデバッグしたり、認可の失敗などの特別な状況や検出した状態を管理者に通知します。

ATMI を使用してシステム・エラーとアプリケーション・エラーを処理する

ATMI を使用すると、プログラマは通信に関するよりグローバルな問題を管理できます。ATMI には、アプリケーションとシステムの両方に関連するエラーを処理する機能があります。サービス・ルーチンで、アプリケーション・エラー (無効なアカウント番号が使用された場合など) が発生すると、アプリケーション・エラーが返され、サービスは実行されたがリクエストは完了しなかったことがクライアントに通知されます。

システム障害 (リクエストの実行中にサーバがクラッシュするなど) が発生すると、システム・エラーが返され、サービス・ルーチンによりタスクが実行されなかったことがクライアントに通知されます。BEA Tuxedo システムは、アプリケーションの動作およびシステム自体の動作を監視し、プログラムに対してシステム・エラーを発行します。

設定可能なタイムアウトのメカニズムを使用する

リクエストの処理中に、サービスが無限ループに陥る場合があります。この場合、クライアント側で待機しても応答は戻りません。クライアントが無限に待機しないようにするため、BEA Tuxedo システムには、ブロッキング・タイムアウトとトランザクション・タイムアウトという 2 つの設定可能なタイムアウトのメカニズムがあります。これらのタイムアウト・メカニズムの詳細については、『BEA Tuxedo Domains コンポーネント』の Domains のトランザクション・タイムアウトとブロッキング・タイムアウトの指定を参照してください。

ブロッキング・タイムアウトは、ブロックされたプログラムが、何らかのイベント発生後に指定の時間を超えて待機しないようにするメカニズムです。いったんタイムアウトが検出されると、待機中のプログラムは、ブロッキング・タイムアウトが発生したことを通知するシステム・エラーを受信します。ブロッキング・タイムアウトは、サービス要求の有効期間、つまりアプ

リケーションがサービス要求に対する応答を待機する時間を定義します。タイムアウト値は、TUXCONFIG ファイルの RESOURCES セクションの BLOCKTIME フィールドで定義されるグローバルな値です。

トランザクション・タイムアウトは、アクティブなトランザクションでリソースが集中することが原因で発生するタイムアウトです。トランザクション・タイムアウトは、トランザクション (その中で複数のサービス要求が行われる場合もあり) の有効期間を定義します。このタイムアウト値は、tpbegin(3c) を呼び出して、トランザクションを開始するときに定義されます。トランザクション・タイムアウトは、リソースを最大化する場合に便利です。たとえば、トランザクション処理中にデータベースがロックされた場合に、アプリケーションのトランザクション用のリソースが停止状態になる時間を制限することができます。トランザクション・タイムアウトは、常にブロッキング・タイムアウトを上書きします。

UBBCONFIG ファイルのトランザクション・ライムアウト・パラメータには次の 2 つがあります。

- UBBCONFIG の SERVICES セクションで指定する TRANTIME。特定の AUTOTRAN サービスのタイムアウト値を制御します。
- UBBCONFIG の RESOURCES セクションで指定する MAXTRANTIME。管理者が、tpbegin(3c) または AUTOTRAN サービスの呼び出しによって開始したトランザクションのタイムアウト値の上限値を設定するために使用します。

これらのトランザクション・タイムアウト・パラメータの詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の UBBCONFIG(5) を参照してください。

冗長サーバを設定して障害を処理する

冗長サーバと自動再起動機能をアプリケーションに設定して、障害に対処することもできます。冗長サーバを設定すると、可用性が高まり、大量の処理、サーバ障害、またはマシン障害を処理できるようになります。BEA Tuxedo システムは、アクティブなサーバの状態を定期的に調べ、再起動可能なサーバの障害を検出すると、自動的にサーバの新しいインスタンスを生成します。

サーバに自動再起動のプロパティを設定すると、個々のサーバの障害を処理できます。また、システムによる再起動の回数を指定することもできます。この機能により、サーバの再起動回数の制限が原因で度々発生するアプリケーション・エラーを回避できます。

BEA Tuxedo システムは、アクティブなマシンの可用性を頻繁に調べます。マシンにアクセスできない場合は、そのマシンを「分断されたマシン(partitioned)」とマークします。この場合は、システム・イベントが生成されます。マシンの分断は、ネットワーク障害、マシン障害、またはサーバの性能低下が原因で発生する場合もあります。

関連項目

- 『BEA Tuxedo システム入門』の 3-1 ページの「BEA Tuxedo システムの管理とサーバ・プロセス」
- 2-5 ページの「システム・データとアプリケーション・データの監視」
- 2-7 ページの「静的および動的な管理データを監視する」

マルチスレッド化またはマルチコンテキスト化されたアプリケーションを監視する

- マルチスレッド化されたアプリケーションを監視する間、管理者は個々のスレッドを認識できません。
- `tmadmin(1)` コマンド・インタプリタを実行すると、マルチスレッド化またはマルチコンテキスト化されたアプリケーションのさまざまな点に関する MIB の統計レポートを取得できます。たとえば、マルチスレッド化されたアプリケーションに関する以下の情報を取得できます。

- クライアント・プロセスあたりのクライアント・コンテキストの数および各クライアント・コンテキストの個別のエントリ (`tmadmin pclt` コマンドを実行して取得)。
- サーバ・プロセスあたりのディスパッチ済みサービスの数。オプションで、各コンテキストの情報を取得することもできます (`tmadmin/psr` を実行して取得。冗長モードでも可)。

■ BBL は、クライアントを調べる場合に、プロセスが実行中であることを確認します。プロセスが異常終了すると、BBL はプロセスの異常終了を検出します。ただし、プロセス内の個々のスレッドが異常終了しても、BBL はスレッドの異常終了を検出しません。

したがって、プログラマは、プロセス内の個々のスレッドが異常終了した可能性があることを認識する必要があります。1 つのスレッドが異常終了してシグナルが出されると、通常、そのスレッドが属する全体のプロセスが異常終了しているため、BBL により異常終了が検出されます。

ただし、スレッドの exit 関数を誤って呼び出したためにスレッドが異常終了した場合、シグナルは生成されません。スレッドが `tpterm()` を呼び出す前にこの種の異常終了が発生すると、BBL では異常終了を検出できず、異常終了したスレッドに関連するコンテキストの登録テーブル・スロットの割り当ては解除されません (スレッドの異常終了を検出できても、BBL は通常この登録テーブルのスロットの割り当て解除は行いません。スレッドが異常終了すると別のスレッドがコンテキストに関連付けられる、というアプリケーション・モデルがあるためです)。

この制限事項には適切な対応策がないため、プログラマはこの点を考慮してアプリケーションを設計する必要があります。

MIB を使用してマルチスレッド化またはマルチコンテキスト化されたアプリケーションのデータを取得する

注記　ここで説明する機能は、使用する管理ツールに関係なく、マルチスレッド化またはマルチコンテキスト化されたすべてのアプリケーションに適用されます。ここでは、MIB の呼び出しを使用する管理

者の観点で機能を説明しています。MIB 用のインターフェイス (tmadmin(1) または BEA Administration Console) を使用する管理者も適用できます。

マルチスレッド化またはマルチコンテキスト化されたアプリケーションのデータは、次の方法で取得できます。

- MIB に対して呼び出しを行う
- tmadmin コマンドを実行する

取得した情報は、以下の場所に格納されます。

- 掲示板のレジストリのクライアント・セクション。コンテキストごとのエントリがあります。エントリは、TPMULTICONTEXTS モードで tpinit() が呼び出され、新しいコンテキストが作成されるたびに、BEA Tuxedo システムによって自動的に作成されます。
- TM_MIB の T_SERVERCTXT クラス。複数のサーバ・ディスパッチ・スレッドが同時に実行されている場合、このクラスの 14 のフィールドには、複数のインスタンスが提供されます。つまり、T_SERVERCTXT セクションには、実行中のサーバ・ディスパッチ・スレッドごとに次のフィールドが用意されています。
  - TA_CONTEXTID (キー・フィールド)
  - TA_SRVGRP (キー・フィールド)
  - TA_SRVID (キー・フィールド)
  - TA_CLTLMID
  - TA_CLTPID
  - TA_CLTREPLY
  - TA_CMTRET
  - TA_CURCONV
  - TA_CURREQ
  - TA_CURRSERVICE
  - TA_LASTGRP
  - TA_SVCTIMEOUT
  - TA_TIMELEFT

- TA_TRANLEV

たとえば、12 のサーバ・ディスパッチ・スレッドが同時に実行されている場合、このアプリケーション用の MIB の T_SERVERCTXT クラスには、12 のオカレンスを含む TA_CONTEXTID フィールド、12 のオカレンスを含む TA_SRVGRP フィールド、などが設定されます。

T_SERVER クラス内のフィールドに複数のインスタンスがあり、これらのインスタンスの値がマルチコンテキスト化されたサーバのコンテキストごとに異なる場合、T_SERVER クラスのフィールドにはダミーの値が指定され、T_SERVERCTXT フィールドには、各コンテキストの実際の値が指定されます。

関連項目

- 『BEA Tuxedo コマンド・リファレンス』の tmadmin(1)
- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の TM_MIB(5)
- 『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』の 10-1 ページの「マルチスレッドおよびマルチコンテキスト・アプリケーションのプログラミング」

3 アプリケーションの動的な変更

ここでは、次の内容について説明します。

- アプリケーションの動的な変更
- tmconfig を使用したコンフィギュレーションへの永続的な変更
- tmconfig の実行
- tmconfig を使用したコンフィギュレーションの一時的な変更
- tmconfig を使用した動的な変更での制限
- tmadmin を使用したコンフィギュレーションへの一時的な変更

アプリケーションの動的な変更

システム管理者は、一度アプリケーションが起動して実行されたら、そのアプリケーションが実行中は企業の求めるパフォーマンス、可用性、セキュリティの各条件を満たすように管理する必要があります。BEA Tuxedo システムでは、システムをシャットダウンせずにコンフィギュレーションを変更することができます。つまり、ユーザの作業を中断することなく、以下の作業を行うことができます。

- コンフィギュレーション・ファイルの既存のエントリに変更を加えることができます。つまり、`TUXCONFIG` を編集できます。

- コンフィギュレーション・ファイルにエントリを追加することによって、アプリケーションにコンポーネントを追加できます。
- サービスの宣言、宣言の取り消し、一時停止または再開、サービス・パラメータ (LOAD、PRIORITY など) の変更を行うことによって、アプリケーションに一時的な変更を加えることができます。

注記　実行中のアプリケーションのコンフィギュレーション・ファイルを変更する場合は、次のいずれかの作業を行います。

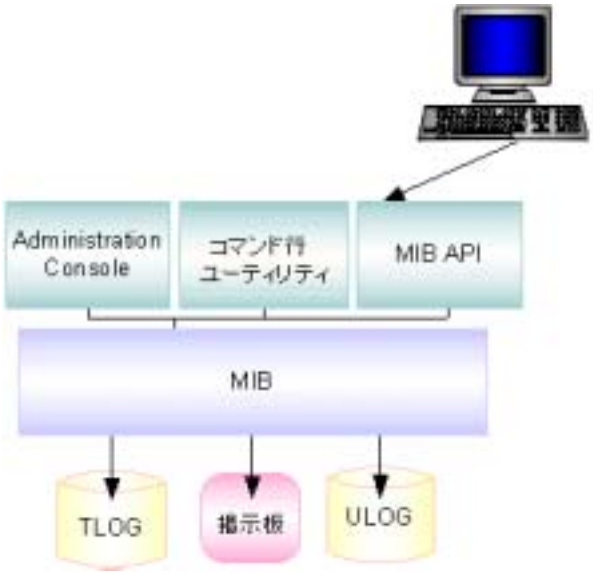
- まず、アプリケーションをシャットダウンします。次に、コンフィギュレーション・ファイルを変更し、アプリケーションを再起動します。
- tmconfig(1) コマンド (tmconfig、wtmconfig(1) リファレンス・ページを参照) を実行し、コンフィギュレーション・ファイルを動的に変更します。

このようにアプリケーションを一時的または永続的に変更すると、現在の条件または継続的に必要とされる条件を反映するようにシステムを変更できます。一時的な変更は、掲示板でのみ反映されます。永続的な変更は、TUXCONFIG ファイルを編集して行います。TUXCONFIG はバイナリ・ファイルなので、通常のテキスト・エディタで編集することができます。

アプリケーションを変更するためのツール

BEA Tuxedo システムには、アプリケーションを動的に変更するための 3 つのツール、BEA Administration Console、コマンド行ユーティリティ、および管理情報ベース (MIB: Management Information Base) API が提供されています。これらのツールを使用すると、業務のニーズやシステムの障害に対応してアプリケーションを変更しなければならないときに、即座に効率的に対応できます。これらのツールを使用すると、アプリケーションを安定した最良の状態で高速に機能させることができます。

図 3-1 動的に変更を加えるためのツール



- BEA Administration Console―Web ベースのグラフィカル・ユーザ・インターフェイス (GUI) であり、アプリケーションを動的にコンフィギュレーションする場合に使用します。このツールでは、コンフィギュレーション情報の表示と変更、システムの各コンポーネントの状態の確認、実行された要求やキューにある要求などの統計情報の取得を行うことができます。
- コマンド行ユーティリティ―動的な変更を行う場合に必要とされる機能は、`tmadmin` および `tmconfig` の 2 つのコマンドで実行できます。`tmadmin` はシェル・レベルのコマンドで、70 以上のサブコマンドがあり、システムの動的な変更をはじめとする各種の管理タスクを行うことができます。`tmconfig` もシェル・レベルのコマンドで、このコマンドを使用すると、システムの実行中でもコンフィギュレーション・エントリの追加や変更を行うことができます。
- MIB API―この管理情報ベース API を使用すると、システムを監視したり、システムに動的な変更を加えるための独自のプログラムを作成することができます。

どの管理タスクに対しても、このいずれかのツールを選択して使用できます。ただし、動的な変更や再コンフィギュレーションを行う場合は、BEA Administration Console を使うと便利です。BEA Administration Console の全機能については、GUI からヘルプにアクセスして参照してください。

コマンド行で操作する場合は、単に `tmadmin` または `tmconfig` コマンドを実行します。

注記　コンフィギュレーション・パラメータと再コンフィギュレーションでの制限については、『BEA Tuxedo コマンド・リファレンス』の `tmconfig`、`wtmconfig(1)`、および『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の `TM_MIB(5)` を参照してください。

関連項目

- 3-5 ページの「tmconfig を使用したコンフィギュレーションへの永続的な変更」
- 『BEA Tuxedo システム入門』の 4-4 ページの「BEA Tuxedo Administration Console を使用した管理操作」
- 『BEA Tuxedo システム入門』の 4-15 ページの「MIB を使用した操作の管理」
- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の `APPQ_MIB(5)`、`DM_MIB(5)`、`MIB(5)`、および `TM_MIB(5)`

tmconfig を使用したコンフィギュレーションへの永続的な変更

tmconfig コマンドを使用すると、コンフィギュレーション・ファイル (マスタ・マシン上の TUXCONFIG) とその構成要素を確認して変更したり、アプリケーションの実行中にそのアプリケーションに新しいコンポーネント (マシンやサーバなど) を追加したりできます。tmconfig を使用してコンフィギュレーション・ファイル (マスタ・マシン上の TUXCONFIG) を変更すると、以下の処理が行われます。

- その時点でアプリケーションで使用されているすべてのマシン上の TUXCONFIG ファイルが更新されます。
- マシンが起動するたびに、その新しいマシンに TUXCONFIG ファイルが複製転送されます。

  注記　tmconfig コマンドは、BEA Tuxedo システムのクライアントとして実行します。

tmconfig は BEA Tuxedo クライアントとして実行するので、以下の条件があります。

- TPINIT 型付きバッファを割り当てることができない場合、tmconfig は失敗します。
- クライアントに対応する *username* は、ユーザのログイン名になります。ユーザのログイン名が確認できない場合、tmconfig は失敗します。
- コンフィギュレーション・ファイルで SECURITY パラメータが設定されたセキュリティ付きアプリケーションの場合、tmconfig の使用時にアプリケーション・パスワードの入力が求められます。アプリケーション・パスワードが入力されなかった場合、tmconfig は失敗します。
- tmconfig がクライアントとして登録できなかった場合、*tperrno* を含むエラー・メッセージが表示されて、tmconfig は終了します。このエラー・メッセージが表示された場合は、ユーザ・ログを確認してエラーの原因を特定します。このエラーの主な原因は、次のとおりです。

- TUXCONFIG 環境変数が正しく設定されていません。
- tmconfig を実行しているマシンとは別のマシン上で、システムが起動しました。

■ tmconfig は、すべての任意通知型メッセージを無視します。

■ printclient (tmadmin コマンド) からの出力に表示される tmconfig プロセスに対するクライアント名は、tpsysadm になります。

tmconfig のしくみ

コマンド行で tmconfig と入力すると、一連のメニューやプロンプトが表示されます。これらのメニューやプロンプトを介して、コンフィギュレーション・ファイルのレコードを表示したり変更することができます。tmconfig はメニューで選択された内容を収集し、要求された操作を実行し、別のメニューを表示して次の操作を指定するように求めます。このコマンドは、メニューから QUIT が選択されてセッションが終了するまで、新しいメニューを表示して操作を実行するように繰り返し要求します。

次に示す例は、tmconfig コマンド・セッションを開始すると表示されるメニューとプロンプトです。

注記　この例では、わかりやすいように行番号を付けてあります。実際の tmconfig セッションでは、これらの番号は表示されません。

コード リスト 3-1　tmconfig **セッションで表示されるメニューとプロンプト**

```
1   $ tmconfig
2   Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
3    5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
4    10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
5
6   Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
7    6) CLEAR BUFFER 7) QUIT [1]:
8   Enter editor to add/modify fields [n]?
9   Perform operation [y]?
```

この例に示したように、次の 4 つの内容が確認されます。

- コンフィギュレーション・ファイルのどのセクションのレコードを表示、追加、または変更するのか
- 指定したコンフィギュレーション・ファイルのセクションで、どの操作を実行するのか
- テキスト・エディタをその場で開き、レコードのフィールドの追加や変更を行うかどうか
- tmconfig を実行して、指定された操作をすぐに行うかどうか

コンフィギュレーション・ファイルのセクションの選択

tmconfig セッションを開始すると、次のメニューが表示されます。各項目は、アプリケーションのコンフィギュレーション・ファイル TUXCONFIG のセクションです。

```
Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
```

注記　各セクションのコンフィギュレーション・パラメータなど、これらのセクションの詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の TM_MIB(5) を参照してください。TM_MIB には、tmconfig コマンドのセッションで表示されるフィールド名、各フィールドに設定できる値の範囲、各セクションのキー・フィールド、および各セクションのフィールドの制限や更新が定義されています。

- セクションを選択するには、メニュー・プロンプトでそれに対応する番号を入力します。たとえば、MACHINES セクションを選択するには、次に示すように「2」と入力します。

  ```
  10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]: 2
  ```

- デフォルトのセクションである RESOURCES では、アプリケーション全体に適用されるパラメータが定義されています。デフォルトのセクション (角かっこで囲まれて表示される番号) を選択する場合は、単に Enter キーを押します。

  ```
  10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
  ```

tmconfig タスクの選択

コンフィギュレーション・ファイルのセクションを選択すると、tmconfig で実行できるタスクのメニューが表示されます。

```
Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
6) CLEAR BUFFER 7) QUIT [1]:
```

操作を選択するには、メニュー・プロンプトでそれに対応する番号を入力します。たとえば、CLEAR BUFFER セクションを選択するには、次に示すように「6」と入力します。

```
6) CLEAR BUFFER 7) QUIT [1]: 6
```

次の表は、各タスクの定義を示しています。

表 3-1tmconfig の各タスク

タスク番号	タスク	タスクの内容
1	FIRST	指定されたセクションの最初のレコードを表示します。キー・フィールドは必要ありません入力バッファに格納されていても、無視されます。 FIRST を使用すると、必要なデータをすべて入力する必要がなくなります。セクションに新しいレコードを追加する場合は、FIRST を使用して UBBCONFIG セクションの既存のレコードを取得します。このようにすると、必要なフィールド名や値をすべて入力する必要がありません。次に、ADD を選択し、テキスト・エディタを使用して、新しく作成したレコードのパラメータ値を変更します。
2	NEXT	入力バッファのキー・フィールドに基づいて、指定されたセクションの次のレコードを表示します。
3	RETRIEVE	要求されたレコード (適切なキー・フィールドで指定されたレコード) を指定されたセクションから表示します。
4	ADD	選択されたセクションに指定されたレコードを追加します。指定されていない省略可能なフィールドに対しては、TM_MIB(5) に指定されているデフォルト値が使用されます。tmloadcf(1) で使用されるすべてのデフォルト値と妥当性検査が適用されます。すべてのフィールドの現在の値は、出力バッファに返されます。この操作は、BEA Tuxedo アプリケーション管理者だけが行うことができます。

表 3-1tmconfig の各タスク

タスク番号	タスク	タスクの内容
5	UPDATE	指定されたセクションの入力バッファで指定されたレコードを更新します。入力バッファ内で指定されていないフィールドは変更されません。tmloadcf(1) で使用されるすべてのデフォルト値と妥当性検査が適用されます。すべてのフィールドの現在の値は、入力バッファに返されます。この操作は、BEA Tuxedo アプリケーション管理者だけが行うことができます。
6	CLEAR BUFFER	入力バッファをクリアします。すべてのフィールドは削除されます。この操作の後で、tmconfig によって再度セクションを指定するように求められます。
7	QUIT	tmconfig を終了します。つまり、クライアントが終了します。tmconfig は、「q」と入力しても終了させることができます。

tmconfig タスクの出力内容

tmconfig によってタスクが完了すると、戻り値と出力バッファの内容が結果として画面上に出力されます。

- 操作は正常に行われたが更新が何も行われなかった場合は、次のメッセージが表示されます。

  ```
  Return value TAOK
  ```

  TA_STATUS フィールドには次のメッセージが表示されます。

  ```
  Operation completed successfully.
  ```

- 操作が正常に行われて更新が行われた場合は、次のメッセージが表示されます。

  ```
  Return value TAUPDATED
  ```

  TA_STATUS フィールドには次のメッセージが表示されます。

  ```
  Update completed successfully.
  ```

- 操作が失敗した場合は、エラー・メッセージが表示されます。

- コンフィギュレーション・パラメータではなく、パーミッションまたは BEA Tuxedo システムとの通信に問題がある場合は、TAEPERM、TAEOS、TAESYSTEM、TAETIME のいずれかの戻り値が出力されます。
- 実行中のアプリケーションのコンフィギュレーション・パラメータに問題がある場合は、TA_BADFLDNAME ファイルの値としてそのパラメータの名前が表示されます。そして、出力バッファの TA_STATUS フィールドの値として問題が示されます。このような問題が発生した場合は、TAERANGE、TAEINCONSIS、TAECONFIG、TAEDUPLICATE、TAENOTFOUND、TAEREQUIRED、TAESIZE、TAEUPDATE、または TAENOSPACE のいずれかの戻り値が出力されます。

tmconfig エラー・メッセージによって示されるシステムの状態

次は、エラー・メッセージで示されるシステムの状態です。

TAEPERM
: UPDATE または ADD が選択されているが、BEA Tuxedo アプリケーション管理者が tmconfig を実行していません。

TAESYSTEM
: BEA Tuxedo システムのエラーが発生しました。エラーの内容は、ユーザ・ログに書き込まれます。『BEA Tuxedo C リファレンス』の userlog(3c) を参照してください。

TAEOS
: オペレーティング・システムのエラーが発生しました。エラーの内容は、ユーザ・ログに書き込まれます。

TAETIME
: ブロッキング・タイムアウトが発生しました。出力バッファは更新されないので、取得操作に対して情報が返されません。更新操作のステータスは、更新されたレコードを取得すると確認できます。

TAERANGE
: フィールド値が範囲外であるか無効です。

TAEINCONSIS
: たとえば、既存の RQADDR 値または SRVGRP/SERVERNAME のあるエントリが、SRVGRP/SERVERNAME の別のエントリに対して指定されています。

TAECONFIG
: TUXCONFIG ファイルの読み取り中にエラーが発生しました。

TAEDUPLICATE
: 重複するレコードを追加しようとしました。

TAENOTFOUND
: 操作対象として指定されたレコードが見つかりませんでした。

TAEREQUIRED
: フィールド値が必要ですが、指定されていません。

TAESIZE
: 文字列フィールドの値が長すぎます。

TAEUPDATE
: 実行できない更新を行おうとしました。

TAENOSPACE
: 更新を行おうとしましたが、TUXCONFIG ファイルまたは掲示板に十分な容量がありませんでした。

tmconfig の実行

tmconfig を正常に実行するには、必要な環境変数を設定する必要があります。また、tmconfig をまだ実行していない場合は、通常の tmconfig セッションを実行し、コンフィギュレーション・ファイルのエントリを変更してください。

tmconfig の環境変数の設定

tmconfig セッションを開始するには、必要な環境変数やパーミッションを設定する必要があります。また、必要に応じて、デフォルトではないテキスト・エディタを選択して使用することもできます。

tmconfig を実行する前に、次の手順に従って作業環境を正しく設定します。

1. TUXCONFIG にエントリを追加したり、既存のエントリを変更する場合は、BEA Tuxedo アプリケーション管理者としてログインします。既存のコンフィギュレーション・ファイルのエントリを参照するだけで、変更や追加を行わない場合は、管理者としてログインする必要はありません。
2. TUXCONFIG および TUXDIR の 2 つの必須の環境変数に値を設定します。
   - TUXCONFIG の値としては、必ず tmconfig を実行しているマシン上のバイナリ・コンフィギュレーション・ファイルの絶対パス名を指定します。
   - TUXDIR の値としては、必ず BEA Tuxedo システムのバイナリ・ファイルが置かれたルート・ディレクトリの絶対パス名を指定します。(tmconfig では、$TUXDIR/udataobj/tpadmin からフィールド名と識別子を抽出できることが必要です。
3. EDITOR 環境変数を設定します (省略可能)。EDITOR の値としては、パラメータ値を変更する場合に使用するテキスト・エディタの名前を指定します。デフォルト値は、ed (UNIX システムのコマンド行エディタ) です。

   注記　TERM 環境変数が設定されていないと、大半の全画面表示のエディタは正しく機能しません。

tmconfig の全セッションの実行

tmconfig のサンプル・セッションを実行するには、次の手順に従います。

1. シェル・プロンプトで「tmconfig」と入力します。

   ```
   $ tmconfig
   ```

   注記　セクション・メニュー・プロンプトで「q」と入力すると、いつでもセッションを終了することができます。

   TUXCONFIG ファイルにあるセクションのメニューが表示されます。

   ```
   Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
   5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
   10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
   ```

2. メニュー番号を入力して、変更するセクションを選択します。たとえば、MACHINES セクションを選択する場合は、「2」と入力します。デフォルト値は RESOURCES セクションで、手順 1 で示したリストの最後にその番号である [1] が表示されています。デフォルト値を使用せずに別のセクションを指定すると、それ以外のセクションを選択するまでそのセクションが新しいデフォルト値となります。

   実行可能な操作のメニューが表示されます。

   ```
   Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
   6) CLEAR BUFFER 7) QUIT [1]:
   ```

   上記の各操作は、コンフィギュレーション・ファイルの 1 つのセクションで、一度に 1 つのレコードに対して実行されます。ほとんどの操作には、操作の内容がわかる名前 (FIRST および NEXT) が付いています。FIRST を選択すると、コンフィギュレーション・ファイルの指定されたセクションの最初のレコードが画面に表示されます。NEXT を選択すると、指定されたセクションの 2 番目のレコードでバッファの内容が置き換わり、バッファの新しい内容が画面に表示されます。NEXT を繰り返し選択すると、コンフィギュレーション・ファイルの特定のセクションにあるすべてのレコードをリストされている順に参照することができます。

3. 実行する操作を選択します。

   デフォルトでは FIRST が選択されており、手順 2 に示したように、リストの最後にこの操作を示す [1] が表示されます。

テキスト・エディタを開いて、手順 2 で選択した TUXCONFIG セクションに変更を加えるかどうかが確認されます。

```
Enter editor to add/modify fields [n]?
```

4. y (yes) または n (no) を選択します。デフォルトでは no が選択されており、メッセージの最後に n と表示されます。

   yes (y) を選択すると指定されたエディタが開き、フィールドの追加や編集を行うことができます。各フィールドは、次のような形式になっています。

   field_name<tabs>*field_value*

   フィールド名とフィールド値は、1 つ以上のタブで区切られています。

   ほとんどの場合、フィールド名は UBBCONFIG ファイルの対応する *KEYWORD* と同じであり、接頭語として TA_ が付いています。

   注記　有効な入力の詳細については、3-15 ページの「tmconfig 入力バッファの注意事項」を参照してください。UBBCONFIG ファイルの各セクションに対応するフィールド名の説明については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の TM_MIB(5) を参照してください。

   入力バッファの編集が終了すると、tmconfig がその内容を読み取ります。エラーが検出されると、構文エラー・メッセージが表示されてそのエラーを修正するかどうかが確認されます。

   ```
   Enter editor to correct?
   ```

5. n または y を選択します。

   エラーを修正しない (n を入力した) 場合は、入力バッファには何もフィールドが格納されません。エラーを修正する場合は、エディタが再度開きます。

   入力バッファの編集を終了するとメッセージが表示され、手順 3 で指定した操作をすぐに実行するかどうかが確認されます。

   ```
   Perform operation [y]?
   ```

6. n または y を選択します。デフォルトでは no が選択されており、メッセージの最後に y と表示されます。

- no を選択すると、セクション・メニューが再度表示されます。手順 2 に戻ります。
- yes を選択すると、指定された操作が tmconfig によって実行され、次の確認メッセージが表示されます。

  ```
  Return value TAOK
  ```

  操作の結果が画面上に出力されます。

  TUXCONFIG の 1 つのセクションでの操作が完了しました。この後、同じセクションまたは別のセクションでほかの操作を実行することができます。新しい操作を実行できるように、手順 1 で表示された TUXCONFIG セクションのメニューが再度表示されます。

注記　すべての出力バッファ・フィールドは、入力バッファがクリアされない限り、入力バッファに格納されます。

7. 操作を指定して tmconfig セッションを続けるか、またはセッションを終了します。
   - 操作を指定するには、手順 2 に戻ります。
   - tmconfig セッションを終了するには、手順 3 で示したように操作メニューから QUIT を選択します。
8. tmconfig セッションを終了した後で、編集した TUXCONFIG ファイルのバックアップ・コピーをテキスト形式で作成できます。次の例では、管理者が yes を指定してバックアップを作成すること (デフォルト)、そしてバックアップ・ファイル (UBBCONFIG) のデフォルト名を別の名前 (backup) で上書きすることを指定しています。

   ```
   Unload TUXCONFIG file into ASCII backup [y]?
   Backup filename [UBBCONFIG]? backup
   Configuration backed up in backup
   ```

tmconfig 入力バッファの注意事項

以下は、tmconfig で入力バッファを操作する場合の注意事項です。

- フィールドに入力した値が 1 行を超える場合、次の行にわたって入力を続けられます。ただし、2 行目の先頭には 1 つ以上のタブを挿入します。

入力した内容が TUXCONFIG に読み込まれた時点で、タブ文字は削除されます。

- 1 つの改行文字だけから構成される空の行は無視されます。
- 特定のフィールドに対して複数の行がある場合、最初の行が使用され、ほかの行は無視されます。

- 表示できない文字をフィールド値として入力したり、タブをフィールドの先頭文字にするには、バックスラッシュを入力し、その後に必要な文字を 2 文字の 16 進表現で入力します。16 進数に対応する ASCII 文字については、UNIX システムのリファレンス・マニュアルの ASCII(5) を参照してください。次に例を示します。
  - 空白を挿入するには、次のように入力します。
    ```
    \20
    ```
  - バックスラッシュを挿入するには、次のように入力します。
    ```
    \\
    ```

tmconfig を使用したコンフィギュレーションの一時的な変更

コンフィギュレーションの各要素は、動的に変更することができます。この節では、以下に示す作業を行うための手順を示します。

- 3-18 ページの「新しいマシンの追加」
- 3-22 ページの「サーバの追加」
- 3-24 ページの「新しく設定したマシンのアクティブ化」
- 3-27 ページの「新しいグループの追加」
- 3-28 ページの「アプリケーションに対するデータ依存型ルーティング (DDR) の変更」
- 3-30 ページの「インターフェイスのファクトリ・ベース・ルーティング (FBR) の変更」
- 3-32 ページの「アプリケーション全体に関するパラメータの変更」
- 3-35 ページの「アプリケーション・パスワードの変更」

新しいマシンの追加

1. 「tmconfig」と入力します。
2. コンフィギュレーション・ファイルの MACHINES セクションを指定するには、セクション・リストの最後に表示されるプロンプトで「2」と入力します (以下に示す例の 2 ～ 4 行目を参照)。
3. Enter キーを押して、デフォルトの操作を選択します。デフォルト値は 1) FIRST です。これは、指定されたセクションの最初のレコードを表示します。その場合、最初のレコードは MACHINES セクションで最初にあるマシンに対するレコードです (6 行目を参照)。
4. Enter キーを押して、エディタの起動と指定された操作の実行に関して、デフォルト値の no と yes をそれぞれ選択します。指定された内容に従って、MACHINES セクションの最初のレコードが表示されます。これは、以下に示す例で、SITE1 という名前のマシンのレコードです (10 ～ 35 行目を参照)。
5. セクション・メニューの最後で Enter キーを押し、MACHINES セクションを再度選択します (36 ～ 38 行目を参照)。
6. 操作メニューの最後で「4」と入力し、ADD を選択します (39 ～ 40 行目を参照)。
7. プロンプトで「y」と入力して、テキスト・エディタを起動します (41 行目を参照)。
8. 適宜パス名を変更し、次の 4 つのキー・フィールドに新しい値を設定します。
   - TA_TLOGSIZE (50 ～ 51 行目を参照)
   - TA_PMID (52 ～ 53 行目を参照)
   - TA_LMID (54 ～ 55 行目を参照)
   - TA_TYPE (56 ～ 57 行目を参照)

9. 入力した内容を書き込み (保存し)、エディタを終了します (58 ～ 60 行目を参照)。
10. プロンプトで「y」と入力し、tmconfig で操作 (マシンの追加) を実行します (61 行目を参照)。

次の例は、マシンを追加する tmconfig セッションを示しています。

コード リスト 3-2 **マシンの追加**

```
1 $ tmconfig
2 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
3 5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
4  10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:2
5 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
6  6) CLEAR BUFFER 7) QUIT [1]:
7 Enter editor to add/modify fields [n]?
8 Perform operation [y]?
9 Return value TAOK
10 Buffer contents:
11 TA_OPERATION             4
12 TA_SECTION               1
13 TA_OCCURS                1
14 TA_PERM                  432
15 TA_MAXACCESSERS          40
16 TA_MAXGTT                20
17 TA_MAXCONV               10
18 TA_MAXWSCLIENTS          0
19 TA_TLOGSIZE              100
20 TA_UID                   4196
21 TA_GID                   601
22 TA_TLOGOFFSET            0
23 TA_TUXOFFSET             0
24 TA_STATUS                LIBTUX_CAT:1137: Operation completed successfully
25 TA_PMID                  mchn1
26 TA_LMID                  SITE1
27 TA_TUXCONFIG             /home/apps/bank/TUXCONFIG
28 TA_TUXDIR                /home/tuxroot
29 TA_STATE                 ACTIVE
30 TA_APPDIR                /home/apps/bank
31 TA_TYPE                  3B2
32 TA_TLOGDEVICE            /home/apps/bank/TLOG
33 TA_TLOGNAME              TLOG
34 TA_ULOGPFX               /home/apps/bank/ULOG
35 TA_ENVFILE               /home/apps/bank/ENVFILE
36 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
37  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
38  10) NETGROUPS 11) NETMAPS 12) INTERFACES [2]:
39 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
40  6) CLEAR BUFFER 7) QUIT [1]: 4
41 Enter editor to add/modify fields [n]? y
42 491
43 g/home/s//usr/p
```

```
44 TA_TUXCONFIG             /usr/apps/bank/TUXCONFIG
45 TA_TUXDIR                /usr/tuxroot
46 TA_APPDIR                /usr/apps/bank
47 TA_TLOGDEVICE            /usr/apps/bank/TLOG
48 TA_ULOGPFX               /usr/apps/bank/ULOG
49 TA_ENVFILE               /usr/apps/bank/ENVFILE
50 /100/s//150/p
51 TA_TLOGSIZE              150
52 /mchn1/s//mchn2/p
53 TA_PMID                  mchn2
54 /SITE1/s//SITE3/p
55 TA_LMID                  SITE3
56 /3B2/s//SPARC/p
57 TA_TYPE                  SPARC
58 w
59 412
60 q
61 Perform operation [y]?
62 Return value TAUPDATED
63 Buffer contents:
64 TA_OPERATION             2
65 TA_SECTION               1
66 TA_OCCURS                1
67 TA_PERM                  432
68 TA_MAXACCESSERS          40
69 TA_MAXGTT                20
70 TA_MAXCONV               10
71 TA_MAXWSCLIENTS          0
72 TA_TLOGSIZE              150
73 TA_UID                   4196
74 TA_GID                   601
75 TA_TLOGOFFSET            0
76 TA_TUXOFFSET             0
77 TA_STATUS                LIBTUX_CAT:1136: Update completed successfully
78 TA_PMID                  mchn2
79 TA_LMID                  SITE3
80 TA_TUXCONFIG             /usr/apps/bank/TUXCONFIG
81 TA_TUXDIR                /usr/tuxroot
82 TA_STATE                 NEW
83 TA_APPDIR                /usr/apps/bank
84 TA_TYPE                  SPARC
85 TA_TLOGDEVICE            /usr/apps/bank/TLOG
86 TA_TLOGNAME              TLOG
87 TA_ULOGPFX               /usr/apps/bank/ULOG
88 TA_ENVFILE               /usr/apps/bank/ENVFILE
```

サーバの追加

1. 「tmconfig」と入力します。
2. コンフィギュレーション・ファイルの SERVERS セクションを指定するには、セクション・メニューの最後で「4」と入力します (3 行目を参照)。
3. 操作メニューの最後で「6」と入力し、CLEAR BUFFER 操作を選択します (5 行目を参照)。
4. Enter キーを押して、デフォルトのセクション SERVERS を選択します (7 ～ 9 行目を参照)。
5. 操作メニューの最後で「4」と入力し、ADD 操作を選択します (10 ～ 11 行目を参照)。
6. プロンプトで「y」と入力して、テキスト・エディタを起動します (12 行目を参照)。
7. 次の 3 つのキー・フィールドに新しい値を設定します。
   - TA_SERVERNAME (15 行目を参照)
   - TA_SRVGRP (16 行目を参照)
   - TA_SRVID (17 行目を参照)
8. 入力した内容を書き込み (保存し)、エディタを終了します (19 ～ 21 行目を参照)。
9. プロンプトで「y」と入力し、tmconfig で操作 (サーバの追加) を実行します (22 行目を参照)。

次の例は、サーバを追加する tmconfig セッションを示しています。

コード リスト 3-3 サーバの追加

```
1 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
2  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
3  10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:4
```

```
4 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
5  6) CLEAR BUFFER 7) QUIT [4]: 6
6 Buffer cleared
7 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
8  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
9  10) NETGROUPS 11) NETMAPS 12) INTERFACES [4]:
10 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
11    6) CLEAR BUFFER 7) QUIT [6]: 4
12 Enter editor to add/modify fields [n]? y
13 1
14 c
15 TA_SERVERNAME            XFER
16 TA_SRVGRP                BANKB1
17 TA_SRVID                 5
18 .
19 w
20 28
21 q
22 Perform operation [y]?
23 Return value TAOK
24 Buffer contents:
25 TA_OPERATION             3
26 TA_SECTION               3
27 TA_OCCURS                1
28 TA_SRVID                 5
29 TA_SEQUENCE              0
30 TA_MIN                   1
31 TA_MAX                   1
32 TA_RQPERM                432
33 TA_RPPERM                432
34 TA_MAXGEN                5
35 TA_GRACE                 86400
36 TA_STATUS                LIBTUX_CAT:1137: Operation completed successfully
37 TA_SYSTEM_ACCESS         FASTPATH
38 TA_ENVFILE
39 TA_SRVGRP                BANKB1
40 TA_SERVERNAME            XFER
41 TA_CLOPT                 -A
42 TA_CONV                  N
43 TA_RQADDR
44 TA_REPLYQ                Y
45 TA_RCMD
46 TA_RESTART               Y
```

新しく設定したマシンのアクティブ化

1. 「tmconfig」と入力します。
2. コンフィギュレーション・ファイルの MACHINES セクションを指定するには、セクション・メニューの最後で「2」と入力します (以下に示す例の 1 ～ 3 行目を参照)。
3. MACHINES セクションで適切なレコードを選択するには、マシン・レコードのリストを切り替えて表示する必要があります。最初のマシン・レコードを表示するには、操作メニューの最後で Enter キーを押して FIRST を選択します (4 ～ 5 行目を参照)。最初のマシン・レコードを選択しない場合、操作メニューの最後で「2」と入力して NEXT を選択し、次のマシン・レコードを表示します。
4. Enter キーを押して、エディタの起動と指定された操作の実行に関して、デフォルト値の no と yes をそれぞれ選択します。MACHINES セクションの指定されたレコードが表示されます。これは、以下に示す例で、SITE3 という名前のマシンのレコードです。(9 ～ 34 行目を参照)。
5. セクション・メニューの最後で Enter キーを押し、MACHINES セクションを再度選択します (35 ～ 37 行目を参照)。
6. 操作メニューの最後で「5」と入力し、UPDATE を選択します (38 ～ 39 行目を参照)。
7. プロンプトで「y」と入力して、テキスト・エディタを起動します (40 行目を参照)。
8. TA_STATE フィールドの値を NEW から ACTIVE に変更します (42 ～ 45 行目を参照)。
9. 入力した内容を書き込み (保存し)、エディタを終了します (46 ～ 48 行目を参照)。
10. プロンプトで「y」と入力し、tmconfig で操作 (新しく設定したマシンのアクティブ化) を実行します (49 行目を参照)。

11. 指定されたマシンに対して変更されたレコードが表示されます。変更内容を確認し、必要に応じて編集します。

12. 変更内容を編集しない場合、操作メニューの最後で「7」と入力し、tmconfig セッションを終了します。

次の例は、サーバをアクティブにする tmconfig セッションを示しています。

コード リスト 3-4 新しいサーバのアクティブ化

```
1  Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
2   5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
3   10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:2
4  Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
5   6) CLEAR BUFFER 7) QUIT [1]:
6  Enter editor to add/modify fields [n]?
7  Perform operation [y]?
8  Return value TAOK
9  Buffer contents:
10 TA_OPERATION             4
11 TA_SECTION               1
12 TA_OCCURS                1
13 TA_PERM                  432
14 TA_MAXACCESSERS          40
15 TA_MAXGTT                20
16 TA_MAXCONV               10
17 TA_MAXWSCLIENTS          0
18 TA_TLOGSIZE              150
19 TA_UID                   4196
20 TA_GID                   601
21 TA_TLOGOFFSET            0
22 TA_TUXOFFSET             0
23 TA_STATUS                LIBTUX_CAT:1175: Operation completed successfully
24 TA_PMID                  mchn2
25 TA_LMID                  SITE3
26 TA_TUXCONFIG             /usr/apps/bank/TUXCONFIG
27 TA_TUXDIR                /usr/tuxroot
28 TA_STATE                 NEW
29 TA_APPDIR                /usr/apps/bank
30 TA_TYPE                  SPARC
31 TA_TLOGDEVICE            /usr/apps/bank/TLOG
32 TA_TLOGNAME              TLOG
33 TA_ULOGPFX               /usr/apps/bank/ULOG
34 TA_ENVFILE               /usr/apps/bank/ENVFILE
35 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
36  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
```

```
37  10) NETGROUPS 11) NETMAPS 12) INTERFACES [2]:
38 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
39  6) CLEAR BUFFER 7) QUIT [1]: 5
40 Enter editor to add/modify fields [n]? y
41 491
42 /TA_STATE
43 TA_STATE                 NEW
44 s/NEW/ACTIVE
45 TA_STATE                 ACTIVE
46 w
47 412
48 q
49 Perform operation [y]?
50 Return value TAUPDATED
51 Buffer contents:
52 .
53 .
54 .
```

新しいグループの追加

1. 「tmconfig」と入力します。
2. コンフィギュレーション・ファイルの GROUPS セクションを指定するには、セクション・メニューの最後に表示されるプロンプトで「3」と入力します (以下に示す例の 1 ～ 3 行目を参照)。
3. 操作メニューの最後で「6」と入力し、CLEAR BUFFER 操作を選択します。(5 行目を参照)。
4. Enter キーを押して、デフォルトのセクション GROUPS を選択します (7 ～ 9 行目を参照)。
5. 操作メニューの最後で「4」と入力し、ADD 操作を選択します (10 ～ 11 行目を参照)。
6. プロンプトで「y」と入力して、テキスト・エディタを起動します (12 行目を参照)。
7. 次の 3 つのキー・フィールドに新しい値を設定します。
   - TA_LMID (15 行目を参照)
   - TA_SRVGRP (16 行目を参照)
   - TA_GRPNO (17 行目を参照)

次の例は、グループを追加する tmconfig セッションを示しています。

コード リスト 3-5 **グループの追加**

```
1 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
2  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
3  10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:3
4 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
5    6) CLEAR BUFFER 7) QUIT [4]: 6
6 Buffer cleared
7 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
8  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
```

```
9  10) NETGROUPS 11) NETMAPS 12) INTERFACES [3]:
10 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
11   6) CLEAR BUFFER 7) QUIT [6]: 4
12 Enter editor to add/modify fields [n]? y
13 1
14 c
15 TA_LMID                  SITE3
16 TA_SRVGRP                GROUP3
17 TA_GRPNO                 3
18 .
19 w
20 42
21 q
22 Perform operation [y]?
23 Return value TAUPDATED
24 Buffer contents:
25 TA_OPERATION             2
26 TA_SECTION               2
27 TA_OCCURS                1
28 TA_GRPNO                 3
29 TA_TMSCOUNT              0
30 TA_STATUS                LIBTUX_CAT:1136: Update completed successfully
31 TA_LMID                  SITE3
32 TA_SRVGRP                GROUP3
33 TA_TMSNAME
34 TA_OPENINFO
35 TA_CLOSEINFO
```

アプリケーションに対するデータ依存型ルーティング (DDR) の変更

アプリケーションのデータ依存型ルーティングを変更するには、次の手順に従います。

1.「tmconfig」と入力します。

2. コンフィギュレーション・ファイルの ROUTING セクションを指定するには、セクション・メニューの最後に表示されるプロンプトで「7」と入力します。
3. FIRST および NEXT を選択して、ROUTING セクションのエントリを切り替えて表示します。FIRST では最初のエントリ、NEXT では次のエントリが表示されます。DDR を変更するエントリを選択します。
4. 操作メニューから 5)UPDATE を選択します。

5. プロンプトで「y」と入力して、テキスト・エディタを起動します

```
Do you want to edit(n)? y
```

6. 関連するフィールドの値を次の表の「サンプル値」の値に変更します。

フィールド	サンプル値	説明
TA_ROUTINGNAME	account_routing	ルーティング・セクションの名前
TA_BUFTYPE	FML	バッファ・タイプ
TA_FIELD	account_ID	ルーティング・フィールドの名前
TA_RANGES	1-10:group1,*:*	使用されているルーティング基準。ここで示すように、account_ID の値が 1 ～ 10 (10 を含む) の場合、要求はグループ 1 のサーバに送信されます。それ以外の場合は、要求はコンフィギュレーション内の任意のサーバに送信されます。

注記　詳細については、『BEA Tuxedo コマンド・リファレンス』の tmconfig、wtmconfig(1) を参照してください。

インターフェイスのファクトリ・ベース・ルーティング (FBR) の変更

注記　BEA Tuxedo CORBA の分散アプリケーションのファクトリ・ベース・ルーティングについては、『BEA Tuxedo CORBA アプリケーションのスケーリング、分散、およびチューニング』を参照してください。

CORBA インターフェイスのファクトリ・ベース・ルーティングを変更するには、次の手順に従います。

1. tmconfig セッションを開始します。

2. コンフィギュレーション・ファイルの ROUTING セクションを選択します (コンフィギュレーション・ファイル・セクションのメニューで 7 を選択します)。
3. FIRST および NEXT の操作を使用して、FBR を変更するエントリを選択します。
4. UPDATE 操作を選択します。
5. 編集を開始するかどうかを確認するメッセージが表示されたら、y (はい) を入力します。

   ```
   Do you want to edit(n)? y
   ```

6. 関連するフィールドを、次の表の中央の列に示す値に変更します。

フィールド	サンプル値	説明
TA_ROUTINGNAME	STU_ID	ルーティング・セクションの名前。
TA_FIELD	student_id	このフィールドの値は、TA_RANGES フィールドで指定された基準に適用されるので、これによってルーティングの結果が決まります。
TA_RANGES	100001-100050:ORA_GRP1, 100051-*:ORA_GRP2	使用されているルーティング基準。

TA_RANGES フィールドの値がルーティング基準です。たとえば、更新前の学生登録のルーティング基準では、学生番号 100001&ends;100005 が ORA_–、100006&e–0010 が ORA_GRP2 に送信されるとします。上の表に示すように変更すると、student_id の値が 100001&en–050 (100001 と 100050 を含む) の場合、要求は ORA_GRP1 のサーバに送信されます。それ以外の要求は ORA_GRP2 に送信されます。

注記　tmconfig によってルーティング・パラメータを動的に変更すると、後続の呼び出しに適用されますが、未処理の呼び出しには適用されません。

INTERFACES セクションで TA_FACTORYROUTING を動的に変更することもできます。次に例を示します。

1. tmconfig セッションを開始します。
2. コンフィギュレーション・ファイルの INTERFACES セクションを選択します (コンフィギュレーション・ファイル・セクションのメニューで 12 を選択します)。
3. FIRST および NEXT 操作を使用して、FBR を変更するインターフェイス・エントリを選択します。たとえば、ROUTING セクションに CAMPUS という名前の新しいファクトリ・ベース・ルーティング基準を定義した場合、この基準に Registrar インターフェイスを再び割り当てることが出来ます。
4. UPDATE 操作を選択します。
5. 編集を開始するかどうかを確認するメッセージが表示されたら、y (はい) を入力します。

```
Do you want to edit(n)? y
```

アプリケーション全体に関するパラメータの変更

一部の実行時パラメータは、コンフィギュレーションのすべてのコンポーネント (マシン、サーバなど) に関連しています。これらのパラメータは、コンフィギュレーション・ファイルの RESOURCES セクションにリストされています。

RESOURCES セクションのパラメータを知る簡単な方法は、そのセクションの最初のエントリを表示することです。その場合、次の手順に従います。

1.「tmconfig」と入力します。
2. セクション・メニューの最後で Enter キーを押し、デフォルト値の RESOURCES セクションを選択します (以下に示す例の 1 ～ 3 行目を参照)。

3. 操作メニューの最後で Enter キーを押し、デフォルト値の FIRST セクションを選択します (4 ～ 5 行目を参照)。
4. 編集するかどうか確認されたら、Enter キーを押してデフォルト値の n を選択します。

   ```
   Do you want to edit(n)?
   ```

5. 指定された操作 (FIRST) を実行するかどうかを確認されたら、Enter キーを押してデフォルト値の y を選択します。

   ```
   Perform operation [y]?
   ```

次の例は、RESOURCES セクションの最初のエントリを表示する tmconfig セッションを示しています。

コード リスト 3-6　RESOURCES **セクションの最初のエントリの表示**

```
1 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
2  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
3  10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
4 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
5  6) CLEAR BUFFER 7) QUIT [1]:
6 Enter editor to add/modify fields [n]?
7 Perform operation [y]?
8 Return value TAOK
9 Buffer contents:
10 TA_OPERATION       1
11 TA_SECTION         0
12 TA_STATUS          Operation completed successfully
13 TA_OCCURS          1
14 TA_PERM            432
15 TA_BBLQUERY        30
16 TA_BLOCKTIME       6
17 TA_DBBLWAIT        2
18 TA_GID             10
19 TA_IPCKEY          80997
20 TA_LICMAXUSERS     1000000
21 TA_MAXACCESSERS    100
22 TA_MAXBUFSTYPE     32
23 TA_MAXBUFTYPE      16
24 TA_MAXCONV         10
25 TA_MAXDRT          0
26 TA_MAXGROUPS       100
27 TA_MAXGTT          25
28 TA_MAXMACHINES     256
```

```
29 TA_MAXQUEUES       36
30 TA_MAXRFT          0
31 TA_MAXRTDATA       8
32 TA_MAXSERVERS      36
33 TA_MAXSERVICES     100
34 TA_MIBMASK         0
35 TA_SANITYSCAN      12
36 TA_SCANUNIT        10
37 TA_UID             5469
38 TA_MAXACLGROUPS    16384
39 TA_MAXNETGROUPS    8
40 TA_MAXINTERFACES   150
41 TA_MAXOBJECTS      1000
42 TA_SIGNATURE_AHEAD 3600
43 TA_SIGNATURE_BEHIND 604800
44 TA_MAXTRANTIME     0
45 TA_STATE           ACTIVE
46 TA_AUTHSVC
47 TA_CMTRET          COMPLETE
48 TA_DOMAINID
49 TA_LDBAL           Y
50 TA_LICEXPIRE       2003-09-15
51 TA_LICSERIAL       1234567890
52 TA_MASTER          SITE1
53 TA_MODEL           SHM
54 TA_NOTIFY          DIPIN
55 TA_OPTIONS
56 TA_SECURITY        NONE
57 TA_SYSTEM_ACCESS   FASTPATH
58 TA_USIGNAL         SIGUSR2
59 TA_PREFERENCES
60 TA_COMPONENTS      TRANSACTIONS,QUEUE,TDOMAINS,TxRPC,
61 EVENTS,WEBGUI,WSCOMPRESSION,TDOMCOMPRESSION
62 TA_SIGNATURE_REQUIRED
63 TA_ENCRYPTION_REQUIRED
64 TA_SEC_PRINCIPAL_NAME
65 TA_SEC_PRINCIPAL_LOCATION
66 TA_SEC_PRINCIPAL_PASSVAR
```

アプリケーション・パスワードの変更

1.「tmconfig」と入力します。
2. セクション・メニューの最後で Enter キーを押し、RESOURCES セクションを選択します (以下に示す例の 2 ～ 4 行目を参照)。
3. 操作メニューの最後で「6」と入力し、CLEAR BUFFER 操作を選択します (6 行目を参照)。
4. セクション・メニューの最後で Enter キーを押し、RESOURCES セクションを再度選択します (8 ～ 10 行目を参照)。
5. 操作メニューの最後で「5」と入力し、UPDATE を選択します (11 ～ 12 行目を参照)。
6. プロンプトで「y」と入力して、テキスト・エディタを起動します (13 行目を参照)。
7. (バッファに) 次のように入力します。

   TA_PASSWORD *new_password*
8. 入力した内容を書き込み (保存し)、エディタを終了します (18 ～ 20 行目を参照)。

次の例は、アプリケーション・パスワードを neptune に変更している tmconfig セッションを示しています。

コード リスト 3-7 **アプリケーション・パスワードの変更**

```
1 $ tmconfig
2 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
3  5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
4  10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
5 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
6    6) CLEAR BUFFER 7) QUIT [4]: 6
7 Buffer cleared
8 Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
9  5)SERVICES 6) NETWORK 7) ROUTING q) QUIT 9) WSL
```

```
10 10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
11 Operation: 1) FIRST 2) NEXT 3) RETRIEVE 4) ADD 5) UPDATE
12    6) CLEAR BUFFER 7) QUIT [6]: 5
13 Enter editor to add/modify fields [n]? y
14 1
15 c
16 TA_PASSWORD         neptune
17 .
18 w
19 49
20 q
21 Perform operation [y]?
22 Return value TAUPDATED
23 Buffer contents:
24 TA_OPERATION       1
25 TA_SECTION         0
26 TA_STATUS          Operation completed successfully
27 TA_OCCURS          1
28 TA_PERM            432
29 TA_BBLQUERY        30
30 TA_BLOCKTIME       6
31 TA_DBBLWAIT        2
32 TA_GID             10
33 TA_IPCKEY          80997
34 TA_LICMAXUSERS     1000000
35 TA_MAXACCESSERS    100
36 TA_MAXBUFSTYPE     32
37 TA_MAXBUFTYPE      16
38 TA_MAXCONV         10
39 TA_MAXDRT          0
40 TA_MAXGROUPS       100
41 TA_MAXGTT          25
42 TA_MAXMACHINES     256
43 TA_MAXQUEUES       36
44 TA_MAXRFT          0
45 TA_MAXRTDATA       8
46 TA_MAXSERVERS      36
47 TA_MAXSERVICES     100
48 TA_MIBMASK         0
49 TA_SANITYSCAN      12
50 TA_SCANUNIT        10
51 TA_UID             5469
52 TA_MAXACLGROUPS    16384
53 TA_MAXNETGROUPS    8
54 TA_MAXINTERFACES   150
55 TA_MAXOBJECTS      1000
56 TA_PASSWORD        neptune
57 TA_STATE           ACTIVE
58 TA_AUTHSVC
```

```
59 TA_CMTRET           COMPLETE
60 TA_DOMAINID
61 TA_LDBAL            Y
62 TA_LICEXPIRE        1998-09-15
63 TA_LICSERIAL        1234567890


64 TA_MASTER           SITE1
65 TA_MODEL            SHM
66 TA_NOTIFY           DIPIN
67 TA_OPTIONS
68 TA_SECURITY         NONE
69 TA_SYSTEM_ACCESS    FASTPATH
70 TA_USIGNAL          SIGUSR2
71 TA_PREFERENCES
72 TA_COMPONENTS       TRANSACTIONS,QUEUE,TDOMAINS,TxRPC,EVENTS,WEBGUI,
73                     WSCOMPRESSION,TDOMCOMPRESSION
```

tmconfig を使用した動的な変更での制限

tmconfig を使用してアプリケーションを動的に変更する場合は、以下の制限に注意してください。簡単には変更できないパラメータは、慎重に設定してください。

- 各セクションには対応するキー・フィールドがあります。これらのキー・フィールドは、操作を実行するレコードを識別するものです。詳細については、『BEA Tuxedo コマンド・リファレンス』の tmconfig、wtmconfig(1) を参照してください。キー・フィールドは、アプリケーションの実行中は変更できません。通常、新しいエントリとそのキー・フィールドを追加し、それを古いエントリの代わりに使用します。その場合、新しいエントリだけが使用されます。つまり、管理者はコンフィギュレーションの古いエントリを起動しません。
- 通常、パラメータに対応付けられたコンフィギュレーション・コンポーネントが起動しているときは、そのパラメータを更新することはできま

せん。たとえば、MACHINES セクションのエントリに対応するマシンが起動しているときは、そのエントリを変更することはできません。特に、

- グループ内のサーバが起動しているときは、そのグループのエントリを変更することはできません。
- サーバが起動しているときは、その名前、タイプ (会話型またはそれ以外)、そのメッセージ・キューに関連するパラメータを変更することはできません。ほかのサーバ・パラメータは変更できます。ただし、サーバが次回に起動するまで変更内容は適用されません。
- SERVICES エントリは随時変更できます。ただし、サーバが次回に宣言されるまで変更内容は適用されません。
- RESOURCES セクションへの更新には、次の制限があります。UID、GID、PERM、MAXACCESSERS、MAXGTT、および MAXCONV パラメータは、RESOURCES セクションでは更新できません。ただし、マシン単位で更新できます。IPCKEY、MASTER、MODEL、OPTIONS、USIGNAL、MAXSERVERS、MAXSERVICES、MAXBUFTYPE、および MAXBUFSTYPE パラメータは、動的には変更できません。

■ 作業しているコンフィギュレーション・ファイルのセクションに注意してください。tmconfig では、間違ったセクションに操作を行っても警告は表示されません。たとえば、RESOURCES セクションを作業している際に、MACHINES セクションの ENVFILE パラメータの更新を試みると、操作は成功したように見えます。つまり、tmconfig で TAOK が返されます。しかし、UBBCONFIG ファイルをアップロードしても、変更内容は適用されていません。更新が行われたのは、TAUPDATED ステータス・メッセージが表示された場合だけです。

複数のマシンがあるコンフィギュレーションでは、常に次の操作を行ってください。

■ MIGRATE オプションを指定してマスタ・マシンのバックアップを指定します。これは、アプリケーション・サーバの移行が必要ない場合でも行います。

■ 上限を指定する MAXSERVERS、MAXSERVICES などのパラメータには、コンフィギュレーションの拡張に対応できるだけの十分な値を設定します。アプリケーションを当初 1 つのマシンだけで実行している場合でも、複数マシンにコンフィギュレーションが拡張される可能性がある場合は、

MP モデルを使用します。その場合、LAN オプション、および初期マシンとしてネットワーク・エントリを指定します。

- MACHINES セクションのパラメータは慎重に設定します。これらのパラメータを更新する場合は、マシンをシャットダウンする必要があるからです。また、マスタ・マシンの場合は、マスタをバックアップに切り替える必要があります。

実行中のシステムに行うことのできない作業

BEA Tuxedo システムのほとんどの要素は、手動でも自動的にも、動的に変更することができます。たとえば、新しいサーバやマシンを増やしたり、タイムアウト・パラメータを変更することができます。ただし、一部のパラメータはシステムの実行中には変更することができません。

- 掲示板のサイズに影響するパラメータは、動的には変更できません。これらのパラメータの大半は、MAXGTT など、MAX という文字列で開始されます。これらのパラメータは、BEA Tuxedo システムで同時に処理できるトランザクションの最大数を示します。
- 実行中のアプリケーションで使用されているマシン名は、動的には変更できません。新しいマシン (つまり、新しい名前のマシン) を追加することはできますが、既存のマシン名を変更することはできません。
- サーバの実行可能ファイルがマスタとバックアップの両方のマシンで実行するように一度割り当てられると、マスタとバックアップの割り当てを変更することはできません。

注記　サーバの実行可能ファイルの新しいコピーを作成し、別のマシンで実行するように設定できます。ただし、一意な識別子を持つ既存のサーバを変更することはできません。

tmadmin を使用したコンフィギュレーションへの一時的な変更

tmconfig コマンドを使用して、TUXCONFIG ファイルとそれに対応する掲示板を更新する場合、行った変更は永続的に適用されます。つまり、システムをシャットダウンして再起動しても変更内容が保持されます。

ただし、実行中のアプリケーションに一時的な変更を加える場合もあります。たとえば、次のような操作を一時的に行う場合があります。

- Tuxedo ATMI サービスまたはサーバの一時停止
- Tuxedo ATMI サービスまたはサーバの再開
- サービスまたはサーバの宣言
- サービスまたはサーバの宣言の取り消し
- サービス・パラメータの変更
- CORBA インターフェイス・パラメータの変更
- タイムアウト値の変更
- CORBA インターフェイスの一時停止
- CORBA インターフェイスの再開

これらの操作は tmadmin コマンドで実行します。この節では、これらの操作を tmadmin コマンドで行う手順を示します。

tmadmin の環境変数の設定

tmadmin セッションを開始する前に、環境変数と必要なパーミッションを設定する必要があります。また、必要に応じて、デフォルトではないテキスト・エディタを選択して使用することもできます。

tmadmin を実行する前に、次の手順に従って作業環境を正しく設定します。

1. TUXCONFIG にエントリを追加したり、既存のエントリを変更する場合は、BEA Tuxedo アプリケーション管理者としてログインします。既存のコンフィギュレーション・ファイルのエントリを参照するだけで、変更や追加を行わない場合は、管理者としてログインする必要はありません。
2. TUXCONFIG および TUXDIR の 2 つの必須の環境変数に値を設定します。
   - TUXCONFIG の値としては、必ず tmconfig を実行しているマシン上のバイナリ・コンフィギュレーション・ファイルの絶対パス名を指定します。
   - TUXDIR の値としては、必ず BEA Tuxedo システムのバイナリ・ファイルが置かれたルート・ディレクトリを指定します。(tmconfig では、$TUXDIR/udataobj/tpadmin からフィールド名と識別子を抽出できることが必要です。

Tuxedo ATMI サービスまたはサーバの一時停止

Tuxedo ATMI サービスまたはサーバを一時停止するには、tmadmin および susp (suspend の省略形) コマンドを次のように入力します。

```
$ tmadmin
> susp
```

susp コマンドでは、次のいずれかが非アクティブになります。

- 1 つのサービス
- 特定のキューにあるすべてのサービス
- 特定のグループ ID またはサーバ ID の組み合わせに対応するすべてのサービス

サービスまたはサーバを一時停止しても、そのサービスまたはサーバに対するキュー内の要求は処理されます。ただし、一時停止したサーバにルーティングされた新しいサービスの要求は処理されません。グループ ID またはサーバ ID の組み合わせが指定され、それが MSSQ セットの一部である場合は、その MSSQ セット内のすべてのサービスは指定されたサービスに対して非アクティブになります。

Tuxedo ATMI サービスまたはサーバの再開

Tuxedo ATMI サービスまたはサーバを再開するには、`tmadmin` および `resume` (`res` の省略形) コマンドを次のように入力します。

```
$ tmadmin
> res
```

`res` コマンドは `susp` コマンドの処理を取り消します。つまり、次のいずれかをキューでアクティブにします。

- 1 つのサービス
- 特定のキューにあるすべてのサービス
- 特定のグループ ID またはサーバ ID の組み合わせに対応するすべてのサービス

グループ ID またはサーバ ID が MSSQ セットの一部である場合、その MSSQ セット内のすべてのサーバは指定されたサービスに対してアクティブになります。

サービスまたはサーバの宣言

サービスまたはサーバを宣言するには、次のコマンドを入力します。

```
$ tmadmin
> adv [{[-q queue_name] | [-g grpid] [-i srvid]}] service
```

サービスの宣言を取り消す場合は、その前に一時停止しておくことが必要です。ただし、再度宣言する場合は、一時停止を解除する必要はありません。以前に宣言が取り消されて現在一時停止しているサービスは、単に宣言するだけで一時停止が解除されます。

サービスまたはサーバの宣言の取り消し

サービスまたはサーバの宣言を取り消すには、次のコマンドを入力して一時停止する必要があります。

```
$ tmadmin
> unadv [{[-q queue_name] | [-g grpid] [-i srvid]}] service
```

サービスの宣言を取り消すことは、一時停止することよりも影響が大きくなります。サービスの宣言を取り消すと、そのサービスに対するサービス・テーブルのエントリの割り当てが解除されます。そのため、サービス・テーブルでクリアされた領域がほかのサービスで利用可能になります。

Tuxedo ATMI サーバのサービス・パラメータの変更

tmadmin コマンドを使用すると、特定のグループ ID または サーバ ID の組み合わせ、または特定のキューに対するサービス・パラメータの値を動的に変更できます。

次の表は、サービス・パラメータを変更するために使用できる tmadmin コマンドを示しています。

変更対象	入力するコマンド
ロード (LOAD)	$tmadmin >chl -s *service_name*
キューから取り出す優先順位 (PRIO)	$tmadmin >chp -s *service_name*
トランザクション・タイムアウト値	$tmadmin >chtt -s *service_name*

-s オプションは、tmadmin default コマンド行、または tmadmin chl、chp、chtt のいずれかのコマンド行で、必ず指定してください。default コマンド行で -s オプションを設定できるので、-s オプションは chl、chp、および chtt コマンド行で省略可能と見なされます。

Tuxedo CORBA サーバのインターフェイス・パラメータの変更

tmadmin コマンドを使用すると、特定のグループ ID または サーバ ID の組み合わせ、または特定のキューに対するインターフェイス・パラメータの値を動的に変更できます。

次の表は、インターフェイス・パラメータを変更するために使用できる tmadmin コマンドを示しています。

変更対象	入力するコマンド
ロード (LOAD)	$tmadmin >chl -I *interface_name*

変更対象	入力するコマンド
キューから取り出す優先順位 (PRIO)	$tmadmin >chp -I *interface_name*
トランザクション・タイムアウト値	$tmadmin >chtt -I *interface_name*

-I オプションは、tmadmin default コマンド行、または tmadmin chl、chp、chtt のいずれかのコマンド行で、必ず指定してください。default コマンド行で -I オプションを設定できるので、-I オプションは chl、chp、および chtt コマンド行で省略可能と見なされます。

AUTOTRAN タイムアウト値の変更

トランザクション・タイムアウト (TRANTIME) を AUTOTRAN フラグが設定されたインターフェイスまたはサービスに対して変更するには、changetrantime (chtt) コマンドを次のように実行します。

```
$ tmadmin
chtt [-m machine] {-q qaddress [-g groupname] [-i srvid]
        [-s service] | -g groupname -i srvid -s service |
        -I interface [-g groupname]} newtlim
```

tpbegin() または tx_set_transaction_timeout() を使用して、アプリケーション・クライアントによって開始されたトランザクション・タイムアウト値を変更することはできません。

Tuxedo CORBA インターフェイスの一時停止

注記 suspend コマンドの実行は、サーバの一時停止と比べて、BEA Tuxedo のシステム・リソースに対する影響を最小限に抑えることができます。

インターフェイスを一時停止するには、suspend (または susp) コマンドを入力します。次に例を示します。

```
tmadmin
>susp -i IDL:beasys.com/Simple:1.0
```

インターフェイスを一時停止すると、再開されるまでクライアントはそのインターフェイス上のメソッドを呼び出すことができません。

Tuxedo CORBA インターフェイスの再開

注記 resume コマンドの実行は、サーバの一時停止と比べて、BEA Tuxedo のシステム・リソースに対する影響を最小限に抑えることができます。

インターフェイスを再開するには、resume (または res) コマンドを入力します。次に例を示します。

```
tmadmin
>res -i IDL:beasys.com/Simple:1.0
```

一時停止されていたインターフェイスを再開すると、クライアントはそのインターフェイス上のメソッドを呼び出すことができます。

4 分散アプリケーションでのネットワーク管理

ここでは、次の内容について説明します。

- 分散アプリケーション用のネットワークの構築
- ネットワーク・データの圧縮
- ネットワーク要求のロード・バランシング
- データ依存型ルーティングの使用
- ネットワーク・コンフィギュレーションの変更

分散アプリケーション用のネットワークの構築

分散アプリケーション用のネットワークを構築する作業は、主にコンフィギュレーション・フェーズまたはセットアップ・フェーズで行われます。ネットワークを定義し、アプリケーションを起動すると、ネットワークは自動的に構築されます。

この章では、ネットワークを介してデータを移動する方法と、ネットワークに関する操作を制御するコンフィギュレーション・ファイルのパラメータを設定する方法について説明します。

ネットワーク・データの圧縮

BEA Tuxedo システムでは、アプリケーションのプロセス間で送信されるデータを圧縮できます。データ圧縮は便利な機能であり、特に、大規模なコンフィギュレーションを運用する場合に重要です。データ圧縮は、同じマシン上でメッセージの送受信を行う場合 (ローカル・データ圧縮)、または異なるマシン間でメッセージの送受信を行う場合 (リモート・データ圧縮) に使用できます。データ圧縮には次の利点があります。

- プロセス間通信 (IPC) のキューを使用してメッセージが送信されるため、ローカル・データ圧縮を行うと、IPC 資源の消費を抑えることができます。
- ネットワーク経由でメッセージが送信されるため、リモート・データ圧縮を行うと、ネットワーク帯域の消費を抑えることができます。

圧縮レベルの設定

データ圧縮を使用するには、コンフィギュレーション・ファイルの MACHINES セクションの CMPLIMIT パラメータを次のように設定する必要があります。

```
CMPLIMIT=string_value1[,string_value2]
```

このパラメータの文字列値には、リモート・プロセス (*string_value1*) およびローカル・プロセス (*string_value2*) で処理されるメッセージのサイズのしきい値を指定します。必須パラメータは最初の文字列値 (string_value1) のみです。どちらの文字列も、デフォルトは MAXLONG パラメータの値です。

TMCMPPRFM パラメータを設定して、圧縮と CPU のパフォーマンスとのバランスを調整することもできます。圧縮レベルを高くすると、圧縮速度は下がりますがネットワーク帯域幅を効率的に利用できます。圧縮レベルを低くすると、圧縮速度は上がり、CPU 使用率を節約できます。

圧縮レベルを指定するには、次の手順に従います。

1. UBBCONFIG コンフィギュレーション・ファイルの CMPLIMIT パラメータを使用して、圧縮のしきい値を設定します。
2. (オプション) TMCMPPRFM 環境変数を設定します。TMCMPPRFM の値には、1 ～ 9 の範囲の 1 桁の数値を指定します。デフォルト値は 1 です (省略可能)。

   圧縮レベルの最低値 1 を指定すると、圧縮は最も早い速度で行われます。圧縮レベルの最高値 9 を指定すると、圧縮は最も遅い速度で行われます。つまり、低い数値が指定されると、圧縮ルーチンの処理速度は速くなります。

TMCMPPRFM 変数の設定方法の詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の tuxenv(5) を参照してください。

データ圧縮のしきい値の選択

メッセージに対して圧縮のしきい値を指定することができます。つまり、指定したしきい値を超えるメッセージは圧縮されます。圧縮のしきい値を指定するには、CMPLIMIT パラメータを設定します。設定の手順については、4-2 ページの「圧縮レベルの設定」を参照してください。

データ圧縮のしきい値を選択する場合は、次の基準を参考にしてください。

- BEA Tuxedo リリース 4.2.1 以降が稼動中のサイトでは、リモート・データ圧縮を使用します。設定は、ネットワークの速度によって異なります。たとえば、イーサネット (高速ネットワーク) と X.25 (低速ネットワーク) には、別の設定を割り当てます。
  - 高速ネットワークの場合は、BEA Tuxedo で生成されるファイル転送処理に対し、下限値を指定してリモート・データ圧縮を設定します (ファイル転送については、後の注記を参照してください)。つまり、送信サイトまたは受信サイトで、ファイル転送の対象となるメッセージだけが圧縮されます。アプリケーション内の各マシンの制限値は異なる場合があります。このような場合は、各マシンに対して下限値を選択してください。

- 低速ネットワークの場合は、すべてのマシンに対してリモート・データ圧縮のしきい値をゼロに設定します。つまり、すべてのアプリケーションとシステム・メッセージは圧縮されます。

■ BEA Tuxedo リリース 4.2.1 以降が稼動中のサイトでは、リリース 4.2.1 以前のサイトと相互運用している場合も、常にローカル・データ圧縮を使用します。これにより、少ない IPC 資源で処理を実行できます。また、この設定により多くのファイル転送が不要になり、ファイル転送が必要な場合もファイルのサイズを大幅に縮小できます。詳細については、『BEA Tuxedo システムのインストール』の-4 ページの「メッセージ・キューとメッセージ」を参照してください。

ローカル・データ圧縮の場合、アプリケーション内の各マシンに対して、異なるしきい値を割り当てることができます。このような場合は、常に各マシンに対して下限値を選択する必要があります。

注記　IPC キューのブロッキングにより、タイムアウトとメッセージの破棄が頻繁に発生する高トラフィックのアプリケーションでは、ローカル圧縮が常に行われるように設定し、IPC のキューイング・サブシステム上のアプリケーションからの要求を少なくすることができます。

圧縮方法は転送されるデータの種類によって異なるため、環境に応じた最適な設定を行うことをお勧めします。

関連項目

■ 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の `DMCONFIG(5)`

■ 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の `tuxenv(5)`

■ 『BEA Tuxedo システム入門』の 2-30 ページの「データ圧縮」

ネットワーク要求のロード・バランシング

ロード・バランシングがオンになっている場合、つまりアプリケーション・コンフィギュレーション・ファイルの RESOURCES セクションで LDBAL が Y に設定されている場合、BEA Tuxedo システムは、リクエストをネットワーク全体に均衡化します。ロード情報はグローバルには更新されないため、各サイトにはリモート・サイトにおけるロードの独自のビューを持ちます。

コンフィギュレーション・ファイルの MACHINES セクションの NETLOAD パラメータ、または TMNETLOAD 環境変数を使用すると、より多くのリクエストをローカル・キューに送るよう強制できます。このパラメータの値はリモート・キューのロードに追加されるため、リモート・キューに実際より多くの作業があるように見えます。そのため、ロード・バランシングがオンになっていても、ローカル要求はリモート・キューよりもローカル・キューに頻繁に送られます。

たとえば、サーバ A とサーバ B がロード・ファクタ 50 のサービスを提供するとします。サーバ A は呼び出し側クライアント (ローカル) と同じマシン上で稼働しており、サーバ B は別のマシン (リモート) 上で稼働しています。NETLOAD を 100 に設定すると、1 つのリクエストがサーバ B に送られる度に約 3 つのリクエストがサーバ A に送られます。

ロード・バランシングに影響するもう 1 つのメカニズムは、アイドル状態のローカル・サーバが優先されるという点です。クライアントと同じマシン上にあるサーバが、希望するサービスを提供し、アイドル状態にある場合、このサーバにリクエストが送られます。ローカル・サーバは直ちに使用可能になるため、この機能はどんなロード・バランシングの設定より優先されます。

関連項目

- 『BEA Tuxedo システム入門』の 2-39 ページの「ロード・バランシング」

データ依存型ルーティングの使用

データ依存型ルーティングは、クライアントが次のものにサービスを要求した場合に有用です。

- 水平分離型データベース
- ルール・ベース・サーバ

水平分離型データベースは、情報を格納しておくリポジトリです。情報はカテゴリ別に保存されます。これは、各本棚に異なるカテゴリ (伝記、フィクションなど) の本が収納されている図書館に似ています。

ルール・ベース・サーバとは、サービス要求をサービス・ルーチンに転送する前に、サービス要求が特定のアプリケーション固有の条件を満たしているかどうかを判定するサーバです。ルール・ベース・サーバは、ほとんど同じ複数の要求に対して、ビジネス上の理由で多少異なる処理を行う場合に使用すると有用です。

注記　分散型 BEA Tuxedo CORBA アプリケーションのファクトリ・ベース・ルーティングについては、『BEA Tuxedo CORBA アプリケーションのスケーリング、分散、およびチューニング』を参照してください。

水平分離型データベースを使ったデータ依存型ルーティングの例

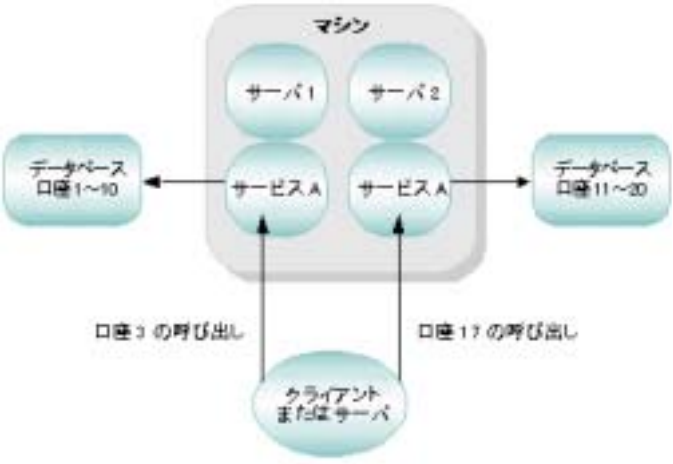
銀行取引アプリケーションで 2 つのクライアントが口座 3 と口座 17 という 2 つの口座の現在の残高を照会する要求を発行したとします。口座 3 と口座 17 という 2 つの口座の現在の残高を照会する要求を発行したとします。アプリケーションでデータ依存型ルーティングが使用されている場合、BEA Tuxedo システムでは次の処理が行われます。

1. 2 つのサービス要求で指定された口座番号 (3 と 17) を取得します。

2. どのサーバがどのデータ範囲の処理を行うかを示す、BEA Tuxedo の掲示板のルーティング・テーブルを調べます (この例では、サーバ 1 は口座 1 ～ 10 の範囲のすべてのリクエストを処理し、サーバ 2 は口座 11 ～ 20 の範囲のすべてのリクエストを処理します)。
3. それぞれの要求を該当するサーバに送信します。つまり、口座 3 への要求をサーバ 1 に転送し、口座 17 への要求をサーバ 2 に転送します。

次の図は、このプロセスを示しています。

図 4-1 水平分離型データベースを使ったデータ依存型ルーティング



ルール・ベース・サーバでのデータ依存型ルーティングの例

次の規則を持つ銀行取引アプリケーションがあるとします。

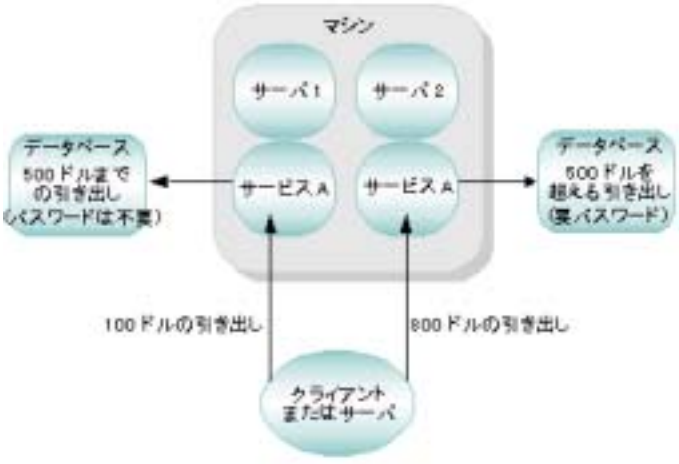
- 顧客は、特別なパスワードを入力しなくても、500 ドルまで引き出すことができます。
- 500 ドルを超える額を引き出すには、特別なパスワードを入力する必要があります。

2 つのクライアントが 1 万円と 8 万円の引き出しを要求したとします。引き出しの規則でデータ依存型ルーティングが有効になっている場合、BEA Tuxedo では次のような処理が行われます

1. 2 つのサービス要求で指定された引き出し額 (100 ドルと 800 ドル) を取得します。
2. どのサーバがいくらのリクエストの処理を行うかを示す、BEA Tuxedo の掲示板のルーティング・テーブルを調べます。この例では、500 ドルまでのすべての引き出し要求をサーバ 1 が処理し、500 ドルを超えるすべての引き出し要求をサーバ 2 が処理しています。
3. それぞれの要求を該当するサーバに送信します。つまり、100 ドルの要求をサーバ 1 に転送し、800 ドルの要求をサーバ 2 に転送します。

次の図は、このプロセスを示しています。

図 4-2 ルール・ベース・サーバでのデータ依存型ルーティング



関連項目

- 『BEA Tuxedo システム入門』の 2-30 ページの「データ依存型ルーティング」

- 『BEA Tuxedo アプリケーションの設定』の第 7 章「ネットワークへの ATMI アプリケーションの分散」
- 『BEA Tuxedo アプリケーションの設定』の第 8 章「分散型の ATMI アプリケーション用のコンフィギュレーション・ファイルの作成」
- 『BEA Tuxedo アプリケーションの設定』の第 9 章「分散アプリケーションのネットワーク設定」
- 『BEA Tuxedo CORBA アプリケーションのスケーリング、分散、およびチューニング』

ネットワーク・コンフィギュレーションの変更

アプリケーションの稼働中にコンフィギュレーション・パラメータを変更するには、tmconfig(1) を実行します。このコマンドは、BEA Tuxedo システム管理情報ベース (MIB) へのシェル・レベルのインターフェイスです。

tmconfig を使用すると、システムを停止せずに TUXCONFIG ファイルの表示や変更を行うことができます。たとえば、アプリケーションの起動中にマシンやサーバなどの新しいコンポーネントを追加することができます。

関連項目

- 『BEA Tuxedo システム入門』の 4-12 ページの「コマンド行ユーティリティを使用したアプリケーションの操作」
- 『BEA Tuxedo コマンド・リファレンス』の tmconfig、wtmconfig(1)
- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の MIB(5)
- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の TM_MIB(5)
- 『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』の 2-41 ページの「リンク・レベルの暗号化の管理」
- 『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』の 2-49 ページの「公開鍵セキュリティの管理」

5 イベント・ブローカについて

ここでは、次の内容について説明します。

- イベント
- アプリケーション定義のイベントとシステム定義のイベントの違い
- イベント・ブローカとは
- イベント・ブローカのしくみ
- ブローカ・イベントの利点

イベント

イベントとは、ネットワークの突然の切断など、実行中のアプリケーションで起こった状態の変更などで、オペレータ、システム管理者、またはソフトウェアによる操作を必要とするものです。BEA Tuxedo システムでは、次の 2 種類のイベントが通知されます。

- システム定義のイベント。これは、BEA Tuxedo システムによって定義される状況 (主に障害) です。たとえば、特定のシステムでの許容量の超過、サーバの終了、セキュリティの侵害、ネットワーク障害などがあります。

- アプリケーション定義のイベント。これは、カスタマ・アプリケーションによって定義される状況です。たとえば、次の表に示すようなものがあります。

ビジネス・アプリケーションの分野	「イベント」として定義される状況
株式売買	株式が指値で、またはそれよりも高く取り引きされました。
銀行業務	限度額を超えて引き出しまたは預金が行われました。
	現金自動預け払い機の現金が取り扱い額よりも少なくなりました。
製造業	在庫がなくなりました。

アプリケーション・イベントはアプリケーション定義のイベントの発生、システム・イベントはシステム定義のイベントの発生を意味します。BEA Tuxedo のイベント・ブローカ・コンポーネントでは、アプリケーション・イベントとシステム・イベントの両方の受信および配信が可能です。

アプリケーション定義のイベントとシステム定義のイベントの違い

アプリケーション定義のイベントは、アプリケーションの設計者によって定義されます。そのため、アプリケーション固有のものです。アプリケーションに対して定義されたすべてのイベントは、アプリケーションで実行されているクライアント・プロセスおよびサーバ・プロセスによってトラッキングされます。

システム定義のイベントは、BEA Tuxedo システムのコードによって定義されます。通常、TM_MIB(5) で定義されたオブジェクトが対応付けられています。システム定義の全イベントのリストは、EVENTS(5) リファレンス・ページに記載されています。これらすべてのイベントは、BEA Tuxedo システムのユーザによってトラッキングされます。

BEA Tuxedo のイベント・ブローカでは、アプリケーション定義のイベントとシステム定義のイベントの両方がポストされます。そして、アプリケーションではこの両方のイベントをサブスクライブできます。この 2 種類のイベントは、名前から区別できます。システム定義のイベントはピリオド (.) で開始され、アプリケーション定義のイベントはピリオド (.) 以外の文字で開始されます。

イベント・ブローカとは

BEA Tuxedo のイベント・ブローカは、BEA Tuxedo アプリケーションで実行中のプロセス間で、アプリケーション・イベントを非同期にルーティングするためのツールです。また、このツールでは、システム・イベントを受信する必要のあるアプリケーション・プロセスに、システム・イベントを配信することもできます。

イベント・ブローカには、次の機能があります。

- イベントをモニタし、イベントが tppost(3c) を介してポストされると、サブスクライバに通知します。
- アプリケーションでの変更をシステム管理者に通知します。
- システム全体に関するイベントのサマリを提供します。
- イベントが各種の通知を開始するためのツールを提供します。
- フィルタ機能を提供し、ポストされたイベントのバッファに条件を追加します。

注記　デモとしてコピーして実行できるサンプル・アプリケーションについては、『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 3-1 ページの「bankapp (複雑な C 言語アプリケーション) のチュートリアル」を参照してください。

イベント・ブローカは、MIB オブジェクトの 100 種類以上の状態の変化をシステム・イベントとして認識します。システム・イベントをポストすると、イベントが発生した現在の MIB としてのオブジェクト、および発生したイベントを識別するイベント固有のフィールドの情報が提供されます。たとえば、マシンが分断された場合、ポストされるイベントには次の情報が含まれます。

- `T_MACHINE` クラスで指定されている影響を受けるマシンの名前、およびそのマシンのすべての属性
- イベントを「分断されたマシン」として識別する属性

イベント・ブローカは、システム・イベントをサブスクライブするだけで使用できます。以降、MIB レコードを要求しなくても、MIB オブジェクトを表す `FML` データ・バッファを受け取ることによって、MIB でイベントが発生したことが自動的に通知されます。

イベント・ブローカのしくみ

BEA Tuxedo のイベント・ブローカは、任意の数のイベント通知の「サプライヤ」が任意の数の「サブスクライバ」にメッセージをポストするためのツールです。イベント通知のサプライヤは、クライアントまたはサーバとして動作しているアプリケーション・プロセスまたはシステム・プロセスです。イベント通知のサブスクライバは、クライアントまたはサーバとして動作しているアプリケーション・プロセス、またはシステム管理者です。

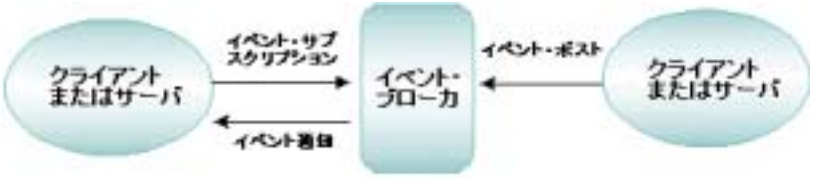
イベント・ブローカを使用するクライアント・プロセスとサーバ・プロセスは、「サブスクリプション」に基づいて相互に通信します。各プロセスでは、そのプロセスが受け取る必要のあるイベント・タイプが識別され、1 つ以上のサブスクリプション要求がイベント・ブローカに送信されます。イベント・ブローカでは、購読料を支払っている購読者だけに新聞を配達するよう

に、その要求を処理します。そのため、イベント・ブローカが基づいているパラダイムは、「パブリッシュ・アンド・サブスクライブ」通信と呼ばれます。

イベント・サプライヤ (クライアントまたはサーバ) は、イベントが発生するたびにイベント・ブローカに通知します。このような通知は、イベントの「ポスト」と呼ばれます。イベント・サプライヤがイベントをポストすると、イベント・ブローカはポストされたイベント・タイプと要求が合致するサブスクライバを探します。サブスクライバは、システム管理者またはアプリケーション・プロセスです。合致するサブスクライバが検出されると、イベント・ブローカは各サブスクリプションに対して指定された処理を行います。つまり、サブスクライバへの通知など、サブスクライバによって指定された処理が開始されます。

次の図は、イベント・ブローカがイベント・サブスクリプションを処理してポストするプロセスを示しています。

図 5-1 イベントのポストとサブスクライブ

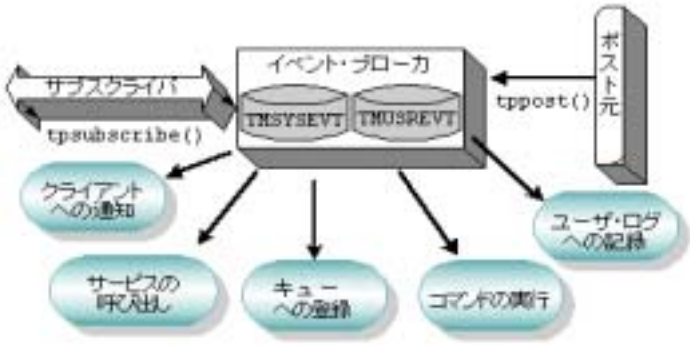


BEA Tuxedo アプリケーション管理者は、クライアント・プロセスとサーバ・プロセスの代わりに、EVENT_MIB(5) に関する追加情報 の T_EVENT_COMMAND クラスへの呼び出しを行って、サブスクリプションを要求できます。イベント・ブローカを使用して、プログラムで tpsubscribe(3c) 関数を呼び出してイベントをサブスクライブすることもできます。

イベントの通知

イベント・ブローカのサブスクリプションでは、次の図に示すいずれかの通知方法が指定されます。

図 5-2 サポートされている通知方法



- クライアントへの通知―イベント・ブローカは、クライアントが通知を必要とする特定のイベントをトラッキングします。そして、そのようなイベントが発生すると、自動的にクライアントに通知します。この方法は「任意通知型通知」と呼ばれます。
- サービスの呼び出し―サブスクライバがイベント通知をサービス呼び出しに渡す必要がある場合、サブスクライバ・プロセスは `tpsubscribe()` 関数を呼び出して、呼び出すサービスの名前を渡します。
- 安定記憶域のキューへのメッセージの登録―サブスクリプションで、イベント通知を安定記憶域のキューに送信することが要求されている場合、イベント・ブローカはキュー・スペース、キュー名、相関識別子を取得します。サブスクライバは、イベントをサブスクライブする際に、キュー名を指定します。相関識別子は、同じイベントの記述とフィルタ・ルールが定義され、格納先が同じキューである複数のサブスクリプションを区別するために使用されます。
- コマンドの実行―イベントがポストされると、それに対応付けられたバッファがシステム・コマンドに変換されて実行されます。たとえば、電子メール・メッセージを送信するシステム・コマンドに変換されます。このプロセスは、MIB を介して行われる必要があります。
- ユーザ・ログへのメッセージの記録―イベントが発生し、イベント・ブローカによって合致する要求があることが検出されると、指定されたメッセージはユーザ・ログ (`ULOG`) に書き込まれます。このプロセスは、MIB を介して行われる必要があります。

システム・イベントの重要度レベル

イベント・ブローカでは、サーバの終了やネットワーク障害などのシステム・イベントに、次の 3 つの重要度のいずれかが割り当てられます。

重要度のレベル	イベント・ブローカに通知される内容
ERROR	サーバの終了やネットワークの突然の切断など、異常が発生しました。
INFO (Information の省略形)	プロセス、またはコンフィギュレーションの変更が原因で、状態が変化しました。
WARN (Warning の省略形)	認証に失敗し、クライアントがアプリケーションに参加できませんでした。アプリケーションのパフォーマンスに影響を与えるようなコンフィギュレーションの変更が行われました。

ブローカ・イベントの利点

- 無名通信―イベント・ブローカを使用すると、BEA Tuxedo の各プログラムで要求されるイベントをそのプログラムがサブスクライブできるようになります。また、すべてのサブスクリプションをトラッキングできるようになります。そのため、あるイベントのサブスクライバは、どのプログラムが同じイベントをサブスクライブするのか知る必要がありません。そして、イベントのポスト元は、ほかのどのプログラムがそのイベントをサブスクライブするのか知る必要がありません。この「無名」処理により、サブスクライバはイベントのポスト元と同期を取る必要がありません。
- 例外条件の分離―パブリッシュ・アンド・サブスクライブ通信モデルでは、例外条件を検出するソフトウェアと、例外条件を処理するソフトウェアを分離することができます。
- BEA Tuxedo システムとの統合化―イベント・ブローカには、メッセージ・バッファ、メッセージ・パラダイム、分散トランザクション、およ

びイベント・ポストに対する ACL パーミッション・チェックなどの機能があります。

- 各種の通知方法―クライアントまたはサーバがシステム・イベント (サーバの終了など) またはアプリケーション・イベント (ATM マシンでのメモリ不足など) をサブスクライブしている場合、そのイベントの発生が通知されたときにイベント・ブローカが行う処理はクライアントまたはサーバによって定義されています。

  サブスクライバが BEA Tuxedo クライアントの場合、サブスクライブ時に次のいずれかの処理が行われます。

  - 任意通知型通知を要求します。
  - 呼び出すサービス・ルーチン名を指定します。
  - 後で処理できるように、イベント・ブローカがデータを格納するアプリケーション・キューを指定します。

  サブスクライバが BEA Tuxedo サーバの場合、サブスクライブ時に次のいずれかの処理が行われます。

  - サービス要求を指定します。
  - イベント・ブローカがデータを格納するアプリケーション・キューを指定します。

関連項目

- 6-1 ページの「イベントのサブスクライブ」
- 『BEA Tuxedo システム入門』の 6-4 ページの「ATMI および EVENT_MIB を使用したイベントのサブスクライブ、ポスト、サブスクライブの取り消し」
- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の `EVENT_MIB(5)` に関する追加情報
- 『BEA Tuxedo C リファレンス』の `tpsubscribe(3c)`
- 『BEA Tuxedo C リファレンス』の `tpunsubscribe(3c)`

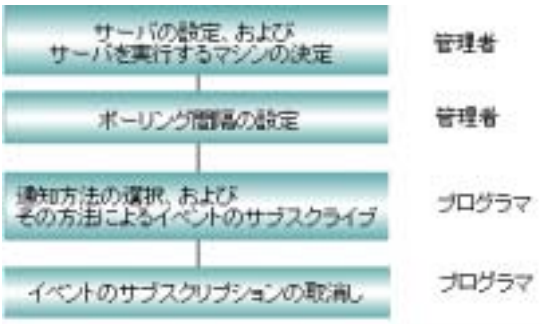
6 イベントのサブスクライブ

ここでは、次の内容について説明します。

- イベント・ブローカを使用するためのプロセス
- イベント・ブローカ・サーバの設定
- ポーリング間隔の設定
- ATMI および EVENT_MIB を使用したイベントのサブスクライブ、ポスト、サブスクライブの取り消し
- 通知方法の選択
- イベントへのサブスクリプションの取り消し
- トランザクションでのイベント・ブローカの使用

イベント・ブローカを使用するためのプロセス

イベント・ブローカを使用するには、準備作業を行う必要があります。次の図は、それらの準備作業、およびアプリケーション管理者またはプログラマのどちらがその作業を行うのかを示しています。



これらの作業を行う手順に関しては、図のボックスをクリックしてください。

注記　イベント・ブローカで行われる処理を理解するには、bankapp を実行します。これは、BEA Tuxedo システムに同梱のサンプル・アプリケーションです。bankapp のコピー方法とデモとしての実行方法については、『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 3-1 ページの「bankapp (複雑な C 言語アプリケーション) のチュートリアル」を参照してください。

イベント・ブローカ・サーバの設定

クライアントがイベント・ブローカにアクセスするには、BEA Tuxedo システムで提供されている 2 つのサーバのいずれかを使用します。つまり、アプリケーション・イベントを処理する TMUSREVT(5)、またはシステム・イベントを処理する TMSYSEVT(5) を使用します。このサーバは両方ともイベントを処理し、サブスクライバへの通知を開始します。

ご使用のシステムに BEA Tuxedo のイベント・ブローカを設定するには、次の例に示すように、この 2 つのサーバのいずれかまたは両方を UBBCONFIG ファイルの SERVERS セクションで設定します。

```
*SERVERS
  TMSYSEVT SRVGRP=ADMIN1 SRVID=100 RESTART=Y GRACE=900 MAXGEN=5
    CLOPT="-A --"
```

```
TMSYSEVT SRVGRP=ADMIN2 SRVID=100 RESTART=Y GRACE=900 MAXGEN=5
  CLOPT="-A -- -S -p 90"

TMUSREVT SRVGRP=ADMIN1 SRVID=100 RESTART=Y
  MAXGEN=5 GRACE=3600
  CLOPT="-A --"
TMUSREVT SRVGRP=ADMIN2 SRVID=100 RESTART=Y
  MAXGEN=5 GRACE=3600
  CLOPT="-A -- -S -p 120"
```

どちらのサーバもネットワーク上の任意の場所に配置できます。ただし、MASTER サイトには主要サーバを割り当てることをお勧めします。

注記　イベント・ポストとイベント通知に起因するネットワーク・トラフィックは、ネットワーク上の別のマシンにセカンダリ・サーバを割り当てると減らすことができます。

ポーリング間隔の設定

セカンダリ・サーバは、プライマリ・サーバをポーリングして現在のサブスクリプション・リストを取得します。このリストには、フィルタと通知に関するルールが定義されています。デフォルトでは、ポーリングは 30 秒間隔で行われます。必要に応じて、この間隔は変更することができます。

ポーリング間隔 (秒単位) を設定するには、コンフィギュレーション・ファイルの TMUSREVT(5) または TMSYSEVT(5) エントリに -p コマンド行オプションを使用します。

-p *poll_seconds*

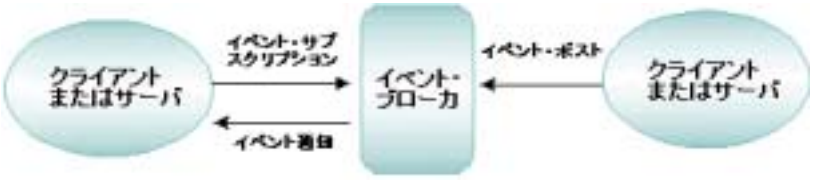
サブスクリプションが追加されている間と、セカンダリ・サーバが更新されている間は、イベント・メッセージが失われたように見えます。

ATMI および EVENT_MIB を使用したイベントのサブスクライブ、ポスト、サブスクライブの取り消し

BEA Tuxedo アプリケーション管理者は、クライアント・プロセスとサーバ・プロセスの代わりに、EVENT_MIB(5) に関する追加情報 の T_EVENT_COMMAND クラスへの呼び出しを行って、サブスクリプションを要求できます。プログラムで tpsubscribe(3c) 関数を呼び出して、イベントをサブスクライブすることもできます。

次の図は、クライアントとサーバがイベント・ブローカを使用して、イベントのサブスクライブ、ポスト、サブスクライブの取り消しを行うプロセスを示しています。

図 6-1 イベントのサブスクライブ



eventexpr およびフィルタを使用したイベント・カテゴリの識別

クライアントまたはサーバがイベントをサブスクライブするには、tpsubscribe(3c) を呼び出します。tpsubscribe() の引数は *eventexpr* (必須) だけです。*eventexpr* の値はワイルドカード文字列で、ユーザに通知する必要があるイベント名を識別します。ワイルドカード文字列については、『BEA Tuxedo C リファレンス』の tpsubscribe(3c) リファレンス・ページを参照してください。

たとえば、UNIX システム・プラットフォーム上のユーザに、ネットワーキング・カテゴリに関するすべてのイベントを通知する必要があるとします。その場合、*eventexpr* に次の値を指定します。

```
\.SysNetwork.*
```

ピリオド (.) の前のバックスラッシュは、ピリオドがリテラルであることを示します。ピリオドの前にバックスラッシュがない場合は、ピリオドは改行文字以外の任意の文字として解釈されます。`\.SysNetwork.*` の最後にある `.*` は、改行文字以外の 0 個以上の任意の文字として解釈されます。

また、クライアントまたはサーバがイベント・データをフィルタするには、`tpsubscribe()` の呼び出し時に *filter* 引数 (省略可能) を指定します。*filter* の値は、ブール値のフィルタ・ルールを含む文字列です。イベント・ブローカがイベントをポストする場合、このルールが正常に評価されることが必要です。

たとえば、ユーザに重要度レベルが `ERROR` であるシステム・イベントだけを通知する必要があるとします。その場合、*filter* に次の値を指定します。

```
"TA_EVENT_SEVERITY='ERROR'"
```

イベント名がポストされ、それが *eventexpr* に合致すると、イベント・ブローカは *eventexpr* に対応付けられているフィルタ・ルールでポストされたデータをテストします。データがフィルタ・ルールに違反していない場合、またはイベントに対するフィルタ・ルールが存在しない場合、サブスクライバはイベント通知およびイベントと共にポストされたすべてのデータを受け取ります。

イベント・ブローカへのアクセス

アプリケーションからイベント・ブローカにアクセスするには、ATMI または `EVENT_MIB(5)` に関する追加情報 を使用します。次の表は、この 2 つの方法について説明しています。

方法	関数	目的
ATMI	tppost(3c)	イベント・ブローカに通知を行うか、またはイベントとそれに伴うデータをポストします。イベント名は *eventname* 引数と *data* 引数で指定され、NULL 以外の場合はデータを指します。ポストされたイベントとそのデータは、BEA Tuxedo のイベント・ブローカによって、*eventname* に対して正常に評価されるサブスクリプション、および *data* に対して正常に評価されるフィルタ・ルール (省略可能) が設定されたすべてのサブスクライバにディスパッチされます。
	tpsubscribe(3c)	*eventexpr* で指定されたイベントをサブスクライブします。サブスクリプションは BEA Tuxedo のイベント・ブローカによって保持され、tppost() を介してイベントがポストされたときにサブスクライバに通知するために使用されます。各サブスクリプションには、クライアントへの通知、サービスの呼び出し、安定記憶域のキューへの登録、コマンドの実行、ユーザ・ログへの記録のいずれかの方法が指定されています。通知方法は、サブスクライバのプロセス・タイプ (つまり、プロセスがクライアントであるか、サーバであるか)、および tpsubscribe() に渡される引数によって決定されます。
	tpunsubscribe(3c)	BEA Tuxedo のイベント・ブローカのアクティブ・サブスクリプションのリストから、イベント・サブスクリプションを削除します。*subscription* は、tpsubscribe() で返されるイベント・サブスクリプション・ハンドルです。*subscription* をワイルドカード値の -1 に設定すると、呼び出し元プロセスが以前に行ったすべての非永続的なサブスクリプションが tpunsubscribe によって取り消されます。非永続的なサブスクリプションとは、tpsubscribe() の ctl->flags パラメータに TPEVPERSIST ビットが設定されずに行われたサブスクリプションです。持続タイプのサブスクリプションは、tpsubscribe() から返されたハンドルを使用することによってのみ削除できます。

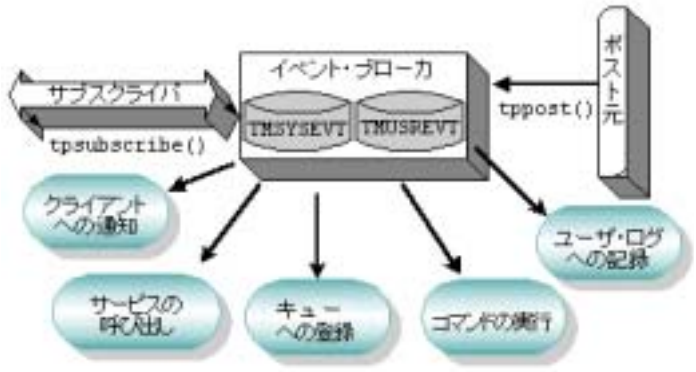
方法	関数	目的
EVENT_MIB(5) に関する追加情報	N/A	EVENT_MIB は、サブスクリプション情報とフィルタ・ルールを格納する管理情報ベース (MIB: Management Information Base) です。EVENT_MIB を使用しても、BEA Tuxedo のイベント・ブローカに新しいイベントをアプリケーションで定義することはできません。ただし、イベント・ブローカをカスタマイズして、イベントをトラッキングし、アプリケーションに要求されるイベントの発生をサブスクライバに通知することができます。 EVENT_MIB を使用して、イベントのサブスクライブまたはサブスクリプションの変更や取り消しを行うことができます。

注記　tppost(3c)、tpsubscribe(3c)、および tpunsubscribe(3c) は C 言語の関数です。これらの関数に相当するルーチン (TPPOST(3cbl)、TPSUBSCRIBE(3cbl)、および TPUNSUBSCRIBE(3cbl)) が COBOL プログラマ用に提供されています。詳細については、『BEA Tuxedo C リファレンス』および『BEA Tuxedo COBOL リファレンス』を参照してください。

通知方法の選択

イベント・ブローカでは、次の図に示すように、各種のイベント通知方法がサポートされています。

図 6-2 イベント・ブローカでサポートされている通知方法



どの通知方法を選択しても、そのインプリメント方法は同じです。`tpsubscribe()` への呼び出しで、`TPEVCTL` 型の構造体を参照する引数を指定します。

引数の値が NULL の場合、イベント・ブローカは任意通知型メッセージをサブスクライバに送信します。通知をサービスに送信する方法と、安定記憶域のキューに送信する方法では、クライアントが通知を直接要求することはできません。サブスクライブするには、クライアントがサービス・ルーチンを呼び出す必要があります。

各サブスクリプションに対して、次の通知方法を選択することができます。つまり、イベント・ブローカでは以下を行うことができます。

- クライアントへの通知―イベント・ブローカは、クライアントへの通知が必要なイベントをトラッキングします。そして、そのようなイベントが発生すると、任意通知型通知をクライアントに送信します。匿名でポストされるイベントもあります。ほかのクライアントがサブスクライブしているかどうかにかかわらず、クライアントはアプリケーションに参加し、イベント・ブローカにイベントをポストできます。イベント・ブローカは、これらのイベントをサブスクリプションのデータベースと照合し、該当するクライアントに任意通知型通知を送信します。『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の `EVENT_MIB(5)` に関する追加情報 エントリの `T_EVENT_CLIENT` クラスの定義を参照してください。

- サービスの呼び出し―サブスクライバがイベント通知をサービス呼び出しに渡す必要がある場合、ctl パラメータが有効な TPEVCTL 構造体を指していることが必要です。『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の EVENT_MIB(5) に関する追加情報 エントリの T_EVENT_SERVICE クラスの定義を参照してください。
- 安定記憶域のキューへのメッセージの登録―安定記憶域のキューへのサブスクリプションでは、eventexpr および filter の値のほかに、キュー・スペース、キュー名、相関識別子が指定されています。これは、合致を検出する場合に使用されます。相関識別子は、同じイベント記述とフィルタ・ルールが定義され、格納先が同じキューである複数のサブスクリプションを区別するために使用されます。『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の EVENT_MIB(5) に関する追加情報 エントリの T_EVENT_QUEUE クラスの定義を参照してください。
- コマンドの実行―EVENT_MIB の T_EVENT_COMMAND クラスを使用して、サブスクライバは実行可能プロセスを呼び出します。合致するデータが検出されると、そのデータは要求されたオプションと、実行可能プロセスの名前として使用されます。『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の EVENT_MIB(5) に関する追加情報 エントリの T_EVENT_COMMAND クラスの定義を参照してください。
- ユーザ・ログ (ULOG) へのメッセージの記録―EVENT_MIB の T_EVENT_USERLOG クラスを使用して、サブスクライバはシステム USERLOG メッセージを書き込みます。イベントが検出されて合致すると、これらのイベントは USERLOG に書き込まれます。『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の EVENT_MIB(5) に関する追加情報 エントリの T_EVENT_USERLOG クラスの定義を参照してください。

イベントへのサブスクリプションの取り消し

クライアントが tpterm(3c) を呼び出してアプリケーションを終了すると、サブスクリプションが永続として指定されていない限り、すべてのサブスクリプションは取り消されます。永続として指定されている場合、クライアントが tpterm() を実行した後でも、サブスクリプションは引き続きポストを受け取ります。クライアントが後でアプリケーションに再度参加し、これらのサブスクリプションを新しくする場合、再度サブスクライブする必要があります。

良く設計されたクライアントは、tpterm() を呼び出す前にサブスクライブを取り消します。その場合、アプリケーションを終了する前に、tpunsubscribe(3c) を呼び出します。

トランザクションでのイベント・ブローカの使用

トランザクションでイベント・ブローカを使用する場合は、次の内容に注意してください。

- トランザクションでイベント・ブローカを使用する前に、イベント・ブローカを実行しているサーバ・グループで、TMUSREVT(5) サーバに NULL_TMS パラメータを設定する必要があります。
- トランザクションにイベントをポストする長所は、トランザクションが成功すると、ポストに関係のない処理も含めて、すべての処理が必ず完了することです。トランザクション内で行われたいずれかの処理が失敗すると、トランザクション内で行われたすべての処理は必ずロールバックされます。短所は、なんらかの理由でトランザクションがアボートすると、ポストが失われる可能性があることです。

- サブスクリプションがトランザクションの一部であることを指定するには、tpsubscribe(3c) に TPEVTRAN フラグを設定します。サブスクリプションがトランザクションに関与して行われた場合、イベントに応答して行われる処理は、呼び出し元のトランザクションの一部として行われます。

  注記　この方法は、BEA Tuxedo サービスが呼び出されるサブスクリプション、またはレコードを永続的なキューに登録するサブスクリプションに対してのみ使用できます。

イベント・ブローカでのトランザクションのしくみ

ポスト元とサブスクライバがトランザクションをリンクすることに合意すると、ボーティング・フォームが作成されます。ポスト元では、なんらかの条件が合致するとアサーションを出力し、メッセージをトランザクションに対応付けます。つまり、元のプロセスを離れたメッセージは、トランザクションと対応付けられていると見なされます。そのトランザクションはイベント・ブローカに渡されます。

サービスの呼び出しやサブスクライバに対してメッセージをキューに格納することなど、イベント・ブローカの処理も同じトランザクションの一部です。実行中のサービス・ルーチンがエラーを検出すると、そのルーチンはトランザクションを失敗させることができます。その場合、トランザクションに関与するほかのすべてのサブスクリプションおよびポストされた元のトランザクションなど、ほかのサービスを呼び出していたり、ほかのデータベース処理を行っていたすべての処理がロールバックされます。ポスト元は (「ポスト元」がこの処理を行うという) アサーションを出力し、データを提供し、そのデータをトランザクションとリンクさせます。

数多くの無名サブスクライバ、つまりポスト元が認識していないサブスクライバは、トランザクションに関与して呼び出されます。サブスクライバがその処理をポスト元の処理とリンクできなかった場合、トランザクション全体がロールバックされます。トランザクションに関与するすべてのサブスクライバは、その処理をポスト元の処理とリンクする必要があります。リンクしないと、すべての処理がロールバックされます。ポスト元がそのトランザクションでポストすることを許可していない場合、イベント・ブローカによって別のトランザクションが開始され、トランザクションに関与するすべてのサブスクリプションがそのトランザクションに集められます。これらのトランザクションのいずれかが失敗すると、トランザクションに関与するサブス

クリプションの代わりに行われたすべての処理はロールバックされます。ただし、ポスト元のトランザクションはロールバックされません。このプロセスは、TPEVTRAN フラグによって制御されます。

トランザクションでのイベント・ブローカの使用例

株式売買が、取引売買アプリケーションで完了しようとしています。取引トランザクションで、各種のサービスによって数多くのデータベース・レコードが更新されました。イベントがポストされると、取引が今行われようとしていることが通知されます。

このような取引の監査を記録するアプリケーションは、このイベントをサブスクライブしています。つまり、このアプリケーションでは、このようなイベントがポストされたら、指定されたキューに必ずレコードを格納するように要求しています。サービス・ルーチンは取引を実行できるかどうかを決定し、このようなイベントをサブスクライブしています。また、そのような取引が行われる場合は通知されます。

すべての処理が問題なく行われると、取引が完了し、監査が記録されます。

キューでエラーが発生し、監査が記録されなかった場合、株式取引全体がロールバックされます。同様に、サービス・ルーチンが失敗すると、トランザクションがロールバックされます。すべての処理が正常に行われると、取引が行われてトランザクションがコミットされます。

関連項目

- 5-3 ページの「イベント・ブローカとは」
- 『BEA Tuxedo システム入門』の 4-19 ページの「イベント・ブローカを使用したイベントの管理」
- 『BEA Tuxedo システム入門』の 4-19 ページの「イベント・ブローカを使用したイベントの管理」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 1-15 ページの「イベント・ベースの通信」
- 『BEA Tuxedo C リファレンス』の `tppost(3c)`、`tpsubscribe(3c)`、および `tpunsubscribe(3c)`

- 『BEA Tuxedo COBOL リファレンス』の TPPOST(3cbl)、TPSUBSCRIBE(3cbl)、および TPUNSUBSCRIBE(3cbl)
- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の EVENT_MIB(5) に関する追加情報、EVENTS(5)、TMSYSEVT(5)、および TMUSREVT(5)

章

7 アプリケーションの移行

ここでは、次の内容について説明します。

- アプリケーションの移行について
- 移行の種類
- Master マシンと Backup マシンの切り替え
- サーバ・グループの移行
- 1 つのマシンから別のマシンへのサーバ・グループの移行
- 移行の取り消し
- Backup マシンへのトランザクション・ログの移行

アプリケーションの移行について

通常、管理者は、設定済みの MASTER マシンで日常的な管理タスクを行います。MASTER マシン上の DBBL は、コンフィギュレーション内のほかのマシンを監視し、コンフィギュレーションを更新し、TMIB への動的な変更をブロードキャストします。マシンのクラッシュ、データベースの破壊、BEA Tuxedo システムの問題、ネットワークの分断、アプリケーションの障害などが原因で MASTER マシンに障害が発生しても、アプリケーションは停止しません。クライアントのアプリケーションへの参加、サーバからのサービス

要求の発行、およびローカル・マシンでの名前指定は引き続き可能です。ただし、MASTER マシンが復元されない限り、サーバをアクティブまたは非アクティブにしたり、管理者側でシステムを動的に再コンフィギュレーションすることはできません。

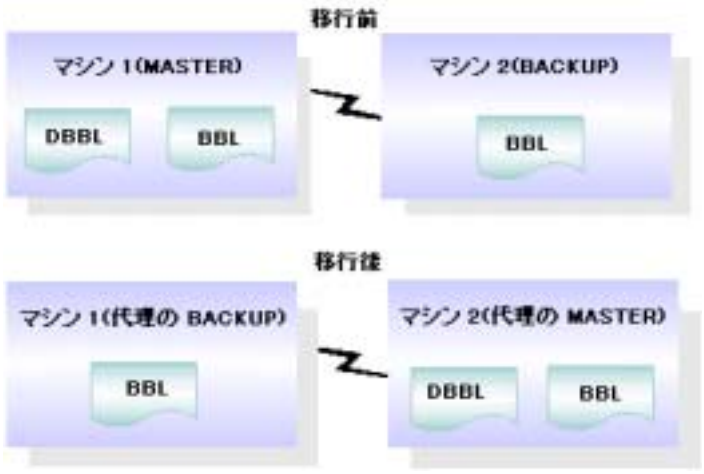
同様のことがアプリケーション・サーバにも当てはまります。アプリケーション・サーバは、特定のマシン上でクライアントのリクエストを処理するようにコンフィギュレーションされていますが、マシンに障害が発生したり、シャットダウンする必要が生じると、そのマシン上のサーバは使用できなくなります。このような場合は、設定済みの BACKUP マシンまたは代替マシンにサーバを移行することができます。

シャットダウンの準備 (サービスまたはアップグレードのため) として移行を行う場合、マシン障害に固有の問題は管理者に通知されません。したがって、この場合の管理者は、高レベルの権限で移行作業を管理できます。

Master マシンの移行

Master マシンの移行とは、設定済みの MASTER マシンから設定済みの BACKUP マシンに DBBL を移動するプロセスのことです。このプロセスにより、設定済みの MASTER マシンがダウンしていても、サーバは引き続きサービスを提供することができます。移行を行うには、まず、設定済みの BACKUP マシンを代理の MASTER マシンとし、設定済みの MASTER マシンを代理の BACKUP マシンとするよう、管理者側から要求します。これで、すべての管理機能は、代理の MASTER マシンで行われます。つまり、コンフィギュレーション内のほかのマシンを監視し、コンフィギュレーションの動的な変更を受け付けます。

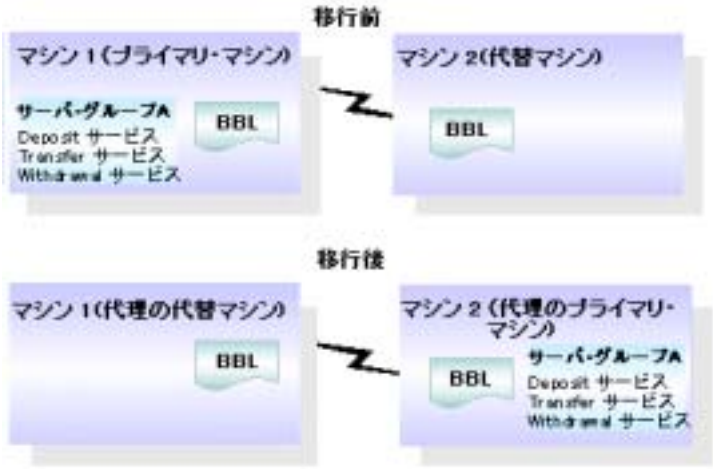
次の図では、マシン 2 (設定済みの BACKUP マシン) が代理の MASTER マシンになり、マシン 1 (設定済みの MASTER マシン) が代理の BACKUP マシンになっています。設定済みの MASTER マシンが利用できるようになると、代理の MASTER マシン (設定済みの BACKUP マシン) から再びアクティブにされます。設定済みの MASTER マシンには、代理の MASTER マシンを制御する権限が再び与えられます。



サーバ・グループの移行

管理者は、各サーバ・グループに対してプライマリ・マシンと代替マシンを指定します。サーバ・グループの移行プロセスでは、代替マシン上のサーバ・グループをアクティブにします。

次の図では、グループ A がマシン 1 (プライマリ・マシン) に割り当てられ、マシン 2 がグループ A の代替マシンとして割り当てられています。移行後、グループ A はマシン 2 でアクティブ化されます。つまり、このグループ内のすべてのサーバおよび関連するサービスは、マシン 2 (代理のプライマリ・マシン) で提供されます。



マシンの移行

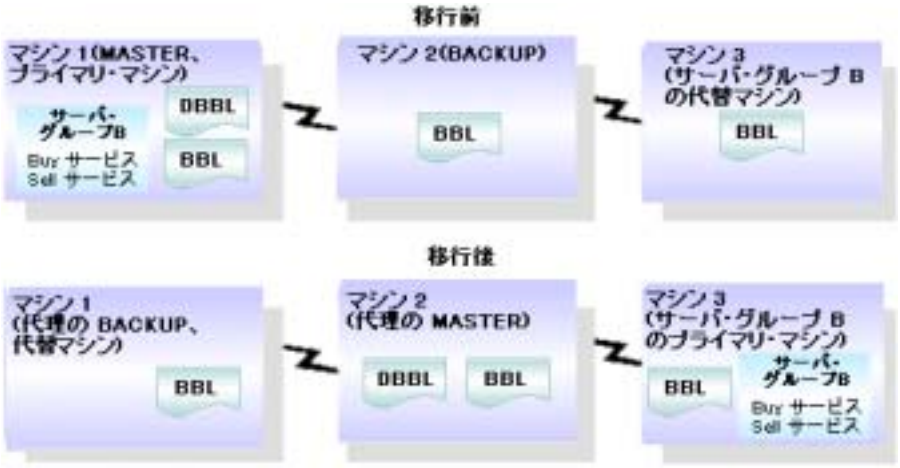
単に 1 つのサーバ・グループを移行することも便利ですが、実際には、マシン全体の移行の方が必要です。たとえば、コンピュータに障害が発生した場合にこのタイプの移行が必要になります。マシンの移行には、そのマシン上で稼動する各サーバ・グループを移行することも含まれます。各サーバ・グループには、代替マシンを設定する必要があります。

スケジュール設定された移行の実行

制御された状況 (コンピュータをオフラインにしたり、アップグレードする必要がある場合など) では、管理者は、サーバやサービスの現在のコンフィギュレーション情報を保存し、これらの情報を、代替マシンでサーバをアクティブにするときに使用できます。コンフィギュレーション情報をこのように使用できるのは、サーバが非アクティブになり、移行の要求に応答できなくなっても、サーバ・エントリがプライマリ・マシンに保持されるためです。

移行処理では、サーバ・グループ全体を移行することも、マシン全体を移行することもできます。マシン全体の移行は、プライマリ・マシン上のすべてのサーバ・グループに対して同じマシンが代替マシンとして設定されている場合に可能です。これ以外の場合 (プライマリ・マシン上の異なるサーバ・グループに対して別々の代替マシンが設定されている)、サーバはマシン単位ではなく、グループ単位で移行しなければなりません。

次の図では、マシン 1 が設定済みの MASTER マシンおよびグループ B のプライマリ・マシンであり、マシン 2 が設定済みの BACKUP です。サーバ・グループ B には、プライマリ・マシンとしてマシン 1 が設定されており、代替マシンとしてマシン 3 が設定されています。マシン 1 がダウンすると、マシン 2 が代理の MASTER マシンとなります。サーバ・グループ B は非アクティブになり、代替マシンであるマシン 3 に移行され、再びアクティブになります。



グループ内のすべてのサーバを非アクティブにしたら、代理のプライマリ・マシンから代理の代替マシンにグループを移行できます。実行中のサーバ、現在宣言されているサービス、および動的に変更されたコンフィギュレーション内容を指定する必要はありません。設定済みの代替マシンは、設定済みのプライマリ・マシン上で利用可能なサーバ (非アクティブの場合) のコンフィギュレーション情報から、これらの情報を取得します。使用されているデータ依存型ルーティングが代替マシンでも適用される場合、サービスは、ルーティング先のグループ名 (マシン名ではない) に基づいてルーティングされます。

移行の対象がアプリケーションの一部であっても、全体であっても、サービスの中断を最小限に抑えるように変更処理を行ってください。アプリケーションのマシン、ネットワーク、データベース、およびその他のコンポーネントの整合性は、そのままにしておく必要があります。BEA Tuxedo システムにはアプリケーションの移行時に、すべてのコンポーネントの整合性を保持するための方法が用意されています。

移行の種類

BEA Tuxedo システムでの移行作業には、次の種類があります。

- MASTER マシンと BACKUP マシンを切り替える
- プライマリ・マシン上の 1 つのサーバ・グループを代替マシンへ移行する
- プライマリ・マシン上のすべてのサーバ・グループを代替マシンへ移行する
- トランザクション・ログを移行する

移行をキャンセルすることもできます。

上記のアプリケーション・コンポーネントの組み合わせに従って移行を行い、分断されたネットワークを回復するためのユーティリティを使用すると、マシン全体を移行できます。

Master マシンと Backup マシンの切り替え

MASTER マシンを保守のためにシャットダウンする場合、または予想外の問題が原因でアクセスできなくなった場合 (ネットワークの分断など) は、MASTER マシンでの作業を設定済みの BACKUP マシンに移行する必要があります。

注記　MASTER マシンと BACKUP マシンで同じリリースの BEA Tuxedo ソフトウェアを実行していることを確認してから MASTER マシンを移行してください。

この種の移行は、DBBL を MASTER マシンから BACKUP マシンへ移行することによって実現できます。DBBL を移行するには、次のコマンドを入力します。

```
tmadmin master
```

ほとんどの場合、アプリケーション・サーバを代替サイトに移行するか、または MASTER マシンを復元する必要があります tmadmin コマンドの詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の tmadmin(1) リファレンス・ページを参照してください。

MASTER マシンと BACKUP マシンの切り替え例

次の 2 つの tmadmin セッションの例では、MASTER マシンと BACKUP マシンの切り替え方法を示しています。ここでは、BACKUP マシンから MASTER にアクセスできるかどうかは無関係です。最初の例では、BACKUP マシンから MASTER にアクセスできるため、DBBL プロセスは MASTER マシンから BACKUP マシンへ移行されます。

コード リスト 7-1 MASTER **マシンと** BACKUP **マシンの切り替え** (BACKUP **マシンから** MASTER **マシンにアクセスできる場合**)

```
$ tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL.  All rights reserved.
> master
are you sure? [y,n] y
Migrating active DBBL from SITE1 to SITE2, please wait...
DBBL has been migrated from SITE1 to SITE2
> q
```

2 番目の例では、BACKUP マシンから MASTER マシンにアクセスできないため、DBBL プロセスは BACKUP マシン上に作成されます。

コード リスト 7-2 MASTER **マシンと** BACKUP **マシンの切り替え** (BACKUP **マシンから** MASTER **マシンにアクセスできない場合**)

```
$ tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL.  All rights reserved.
TMADMIN_CATT:199:Cannot become administrator.  Limited set of commands available.
> master
are you sure? [y,n]  y
Creating new DBBL on SITE2, please wait... New DBBL created on SITE2
> q
```

サーバ・グループの移行

1. UBBCONFIG ファイルの GROUPS セクションで、移行されるサーバ・グループの LMID パラメータに代替ロケーションを指定します。そのグループ内のサーバには RESTART=Y を指定し、UBBCONFIG ファイルの RESOURCES セクションで MIGRATE オプションを指定する必要があります。
2. サーバ・グループを移行する場合は、次のコマンドを実行して、グループ内のサーバをシャットダウンします。

   ```
   tmshutdown -R -g groupname
   ```

3. 次のコマンドを入力して tmadmin セッションを開始します。

   ```
   tmadmin
   ```

4. tmadmin のプロンプトで、次のどちらかのコマンドを入力します。
   - 1 つのグループ内のすべてのサーバを移行するには、次のコマンドを入力します。

     ```
     migrategroup(migg)
     ```

     このコマンドには、1 つのサーバ・グループの名前を引数として指定します。
   - マシン上のすべてのサーバ・グループ (LMID で指定) を移行するには、次のコマンドを入力します。

```
migratemach(migm)
```

5. 移行するサーバ・グループに含まれるサーバに対してトランザクションが記録されている場合は、TLOG を BACKUP マシンに移動し、それをロードしてウォームスタートを行います。

サーバ・グループの移行 (プライマリ・マシンから代替マシンにアクセスできる場合)

プライマリ・マシンから代替マシンにアクセスできる場合は、次の手順でサーバ・グループを移行します。

1. 次のコマンドを入力して、MASTER マシンをシャットダウンします。

   ```
   tmshutdown -R -g groupname
   ```

2. 次のコマンドを入力して、プライマリ・マシンで tmadmin セッションを開始します。

   ```
   tmadmin
   ```

3. 次のコマンドを入力して、適切なグループを移行します。

   ```
   migrategroup groupname
   ```

4. 必要に応じてトランザクション・ログを移行します。
5. 必要に応じてアプリケーション・データを移行します。

サーバ・グループの移行 (プライマリ・マシンから代替マシンにアクセスできない場合)

プライマリ・マシンから代替マシンにアクセスできない場合は、必要に応じて次の手順で MASTER マシンと BACKUP マシンを切り替えます。

1. 代替マシンで次のコマンドを入力して tmadmin セッションを開始します。

   ```
   tmadmin
   ```

2. 次のコマンドを入力して、クリーンナップと再起動が必要なプライマリ・マシン上のサーバを指定し、処理を実行します。

   ```
   pclean primary_machine
   ```

3. 次のコマンドを入力して、適切なサーバ・グループを設定済みの代替マシンに転送します。

   ```
   migrate groupname
   ```

4. 次のコマンドを入力して、移行後のサーバ・グループを起動します。

   ```
   boot -g groupname
   ```

サーバ・グループの移行例

次の 2 つのセッション例では、サーバ・グループの移行方法を示しています。ここでは、プライマリ・マシンから代替マシンにアクセスできるかどうかは無関係です。最初の例では、プライマリ・マシンから代替マシンにアクセスできます。

コード リスト 7-3 **サーバ・グループの移行 (プライマリ・マシンから代替マシンにアクセスできる場合)**

```
$ tmshutdown -R -g GROUP1
Shutting down server processes...
Server ID = 1 Group ID = GROUP1 machine = SITE1: shutdown succeeded
1 process stopped.
$ tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL.
> migg GROUP1
migg successfully completed
> q
```

2 番目の例では、プライマリ・マシンから代替マシンにアクセスできません。

コード リスト 7-4 サーバ・グループの移行 (プライマリ・マシンから代替マシンにアクセスできない場合)

```
$ tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL.
> pclean SITE1
Cleaning the DBBL.
Pausing 10 seconds waiting for system to stabilize.
3 SITE1 servers removed from bulletin board
> migg GROUP1
migg successfully completed.
> boot -g GROUP2
Booting server processes ...
exec simpserv -A :
on SITE2 -> process id=22699 ... Started.
1 process started.
> q
```

1 つのマシンから別のマシンへのサーバ・グループの移行

1. LMID を使用して、サーバ・グループが稼働するプロセッサを指定します。LMID のすべてのサーバ・グループに対して同じ代替ロケーションを指定します。
2. UBBCONFIG ファイルの RESOURCES セクションで、次のようにパラメータを設定します。
   - LMID で指定したマシン上の各サーバに対して、RESTART=Y を設定します。
   - MIGRATE オプションを指定します。

3. 次のコマンドを入力して、すべてのサーバ・グループをシャットダウンし、グループ内のサーバが再起動可能になるように設定します。

   ```
   tmshutdown -R
   ```

4. プライマリ・マシンを保守のためにシャットダウンする場合、またはプライマリ・マシンにアクセスできなくなった場合は、tmadmin(1) migratemach (migm) コマンドを使用してすべてのサーバ・グループを 1 つのマシンから別のマシンに移行します。このコマンドでは、引数として 1 つの論理マシン識別子を指定します。

マシンの移行 (プライマリ・マシンから代替マシンにアクセスできる場合)

プライマリ・マシンから代替マシンにアクセスできる場合は、次の手順でマシンを移行します。

1. マシン上で次のコマンドを入力して、MASTER マシンをシャットダウンします。

   ```
   tmshutdown -R -1 primary_machine
   ```

2. 次のコマンドを入力して、MASTER マシンで tmadmin セッションを開始します。

   ```
   tmadmin
   ```

3. tmadmin のプロンプトで、次のコマンドを入力して適切なマシンを移行します。

   ```
   migratemach primary_machine
   ```

4. 必要に応じてトランザクション・ログを移行します。
5. 必要に応じてアプリケーション・データを移行します。

マシンの移行 (プライマリ・マシンから代替マシンにアクセスできない場合)

プライマリ・マシンから代替マシンにアクセスできない場合は、必要に応じて次の手順で MASTER マシンと BACKUP マシンを切り替えます。

1. 代替マシンで次のコマンドを入力して tmadmin セッションを開始します。

   ```
   tmadmin
   ```

2. 次のコマンドを入力して、クリーンナップと再起動が必要なプライマリ・マシンを指定し、処理を実行します。

   ```
   pclean primary_machine
   ```

3. 次のコマンドを入力して、適切なサーバ・グループを設定済みの代替マシンに転送します。

   ```
   migratemach primary_machine
   ```

4. 次のコマンドを入力して、移行後のサーバ・グループを起動します。

   ```
   boot -l alternate_machine
   ```

マシンの移行例

次のセッションの例は、マシンの移行方法を示しています。最初の例では、プライマリ・マシンから代替マシンにアクセスできます。

コード リスト 7-5 **マシンの移行 (プライマリ・マシンから代替マシンにアクセスできる場合)**

```
$ tmshutdown -R -l SITE1
Shutting down server processes...
Server ID = 1 Group ID = GROUP1 machine = SITE1: shutdown
succeeded 1 process stopped.
$ tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL.
> migm SITE1
migm successfully completed
> q
```

2 番目の例では、プライマリ・マシンから代替マシンにアクセスできません。

コード リスト 7-6 **マシンの移行 (プライマリ・マシンから代替マシンにアクセスできない場合)**

```
$ tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL.
>pclean SITE1
Cleaning the DBBL.
Pausing 10 seconds waiting for system to stabilize.
3 SITE1 servers removed from bulletin board
> migm SITE1
migm successfully completed.
> boot -l SITE2
Booting server processes ...
exec simpserv -A :
on SITE2 -- process id=22782 ... Started.
1 process started.
>q
```

移行の取り消し

サーバ・グループまたはマシンを非アクティブにした後で移行作業を中止したい場合は、再びサーバ・グループまたはマシンをアクティブにする前に移行を取り消すことができます。ネーム・サーバ内にある、非アクティブにされたサーバとサービスの情報はすべて削除されます。

シャットダウンを実行しても、`migrate` コマンドを実行する前であれば、次のコマンドを使用して移行を取り消すことができます。

取り消しの対象	**コマンド**	**結果**

サーバの移行	`tmadmin migrategroup -cancel`
	または
	`tmadmin migg -cancel`
		掲示板からサーバ・エントリが削除されます。移行が取り消された場合は、サーバを再起動する必要があります。

取り消しの対象	**コマンド**	**結果**

マシンの移行	`tmadmin migratemach -cancel`
	または
	`tmadmin migm -cancel`
		移行は中止されます。

移行の取り消し例

次の `tmadmin` セッションの例は、サーバ・グループとマシンが、対応するプライマリ・マシンと代替マシン間で移行される手順を示しています。

コード リスト 7-7 **サーバ・グループ** GROUP1 **の移行の取り消し**

```
$tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL.
> psr -g GROUP1

a.out Name  Queue Name    Grp Name   ID RqDone Ld Done Current Service
----------  ----------    --------   -- ------ ------- ---------------
simpserv     00001.00001   GROUP1     1    -    -       (DEAD MIGRATING)
> psr -g GROUP1
TMADMIN_CAT:121: No such server
migg -cancel GROUP1
>boot -g GROUP1
Booting server processes...
exec simpserv -A:
on SITE1 ->process id_27636 ...Started. 1 process started.
> psr -g GROUP1

a.out Name  Queue Name    Grp Name   ID RqDone Ld Done Current Service
----------  ----------    --------   -- ------ ------- ---------------
simpserv    00001.00001    GROUP1     1    -    -           ( - )
> q
```

Backup マシンへのトランザクション・ログの移行

トランザクション・ログを BACKUP マシンに移行するには、次の手順に従います。

1. 次のコマンドを入力して tmadmin セッションを開始します。

   ```
   tmadmin
   ```

2. すべてのグループ内のサーバをシャットダウンし、ログへの書き込みを中止します。

3. 次のコマンドを実行して、テキスト・ファイルに TLOG をダンプします。

   ```
   dumptlog [-z config] [-o offset] [-n filename] [-g groupname]
   ```

   注記　TLOG は、引数 *config* および *offset* を使用して指定します。*offset* はデフォルトで 0 に設定されます。*name* はデフォルトで TLOG に設定されます。-g オプションを選択すると、TMS によって調整された、*groupname* のレコードだけがダンプされます。

4. *filename* を BACKUP マシンにコピーします。

5. 次のコマンドを入力して、指定したマシンの既存の TLOG にファイルを読み込みます。

   ```
   loadtlog -m machine filename
   ```

6. 次のコマンドを入力して、TLOG を強制的にウォームスタートさせます。

   ```
   logstart machine
   ```

   TLOG の情報が読み込まれ、この情報を使用して共用メモリ内のトランザクション・テーブルにエントリが作成されます。

7. サーバを BACKUP マシンに移行します。

8 BEA Tuxedo ATMI アプリケーションのチューニング

ここでは、次の内容について説明します。

- MSSQ セットの使用
- ロード・バランシングの有効化
- サービスの実行時間の測定
- サービスへの優先順位の割り当て
- 複数のサービスをサーバへバンドルする
- システム全体のパフォーマンスの向上
- システムの IPC 要件の決定
- IPC パラメータの調整
- システム・トラフィックの測定

注記　BEA Tuxedo CORBA 環境でのアプリケーションのチューニングについては、『BEA Tuxedo CORBA アプリケーションのスケーリング、分散、およびチューニング』を参照してください。

MSSQ セットの使用

注記　BEA Tuxedo CORBA サーバでは、MSSQ セット (複数サーバ、単一キュー) がサポートされていません。

BEA Tuxedo ATMI 環境で MSSQ のスキーマを使用すると、ロード・バランシングを行うことができます。1 つのキューには、常に同じサービスを提供する複数のサーバが対応しています。ある要求がサーバのキューに送信され、そのキューが MSSQ セットの一部である場合、メッセージがキューから取り出されて、使用可能な最初のサーバに送られます。このようにして、個々のキューのレベルでロード・バランシングが行われます。

MSSQ を構成するサーバには、そのサーバ固有の応答キューを設定する必要があります。このサーバがほかのサーバに対して要求を行った場合、その応答は要求元のサーバに返されなければなりません。つまり、MSSQ 内のほかのサーバによってキューから取り出されないようにする必要があります。

MSSQ セットを動的に設定し、キューの負荷状況に応じてサーバの生成や削除を自動的に行うことができます。

次の表は、どのような場合に MSSQ セットの使用が適しているかを示しています。

MSSQ セットの使用が適している場合	MSSQ セットの使用が適さない場合
2 ～ 12 台のサーバがある。	サーバ数が多すぎる (ただし、MSSQ セットを多数使用して対処することも可能)。
バッファ・サイズが適度な大きさである (1 つのキューに十分収まる程度)。	バッファ・サイズが 1 つのキューでいっぱいになる場合。
すべてのサーバが同じサービスのセットを提供する。	サーバごとにサービスが異なる場合。
メッセージのサイズが比較的小さい。	キューがいっぱいになるほどサイズの大きいメッセージがサービスに渡される。キューがいっぱいになると、非ブロッキング送信が失敗するか、またはブロッキング送信がブロックされます。

MSSQ セットの使用が適している場合	MSSQ セットの使用が適さない場合
サービスのターンアラウンド・タイムが最適であり、一貫性がある。	

次は、MSSQ セットの使用が適している場合と適さない場合の例を日常生活の中から示しています。

- MSSQ セットが適用できる例は銀行です。銀行には、同じサービスを提供する窓口が複数あり、顧客は 1 列に並んで空いた窓口へ順に進みます。次に空く窓口が、常に列の次の顧客に応対します。この例では、各窓口がすべての顧客サービスを行うことができなければなりません。これを BEA Tuxedo 環境に当てはめると、複数のサーバが単一のキューを共有する場合、すべてのサーバで常に同じサービスが提供できなければならないことを示します。MSSQ セットの使用により、各キューのレベルでロード・バランシングを実現できます。
- MSSQ セットが適用できない例は、スーパーマーケットのレジでのサービスです。クレジット・カードを受け付けるレジや、現金のみを受け付けるレジなど、レジごとに支払方法が異なるという図式は、MSSQ セットを適用できない BEA Tuxedo アプリケーションに似ています。

ロード・バランシングの有効化

システム・トラフィックが原因でアプリケーションの性能が低下するのを避けるため、アプリケーション全体に対してロード・バランシング・アルゴリズムを実行できます。ロード・バランシングを使用すると、ロード・ファクタがシステム内の各サービスに適用され、各サーバの負荷の合計を監視できます。各サービス要求は、負荷が最も少ない適切なサーバに送信されます。

システム全体に対してロード・バランシングを実行するには、次の手順に従います。

1. アプリケーションを長時間実行します。
2. 各サービスの実行にかかる平均時間を記録します。

3. コンフィギュレーション・ファイルの RESOURCES セクションを次のように設定します。
   - LDBAL を Y に設定します。
   - 平均値に近い時間がかかるサービスの場合は、LOAD 値に 50 (LOAD=50) を割り当てます。
   - 平均値より長い時間がかかる場合は、LOAD>50 を設定します。平均値より短い時間で済む場合は LOAD<50 を設定します。

注記　このアルゴリズムは効果的ですが、コストがかかるため必要な場合にのみ使用してください。アルゴリズムが必要となるのは、2 つ以上のキューを使用する複数のサーバによってサービスが提供される場合のみです。1 つのサーバのみによって提供されるサービス、または MSSQ (複数サーバ、単一キュー) にある複数のサーバによって提供されるサービスは、ロード・バランシングを必要としません。

サービスの実行時間の測定

サービスの実行時間は、以下のどちらかの方法で測定できます。

- 管理パラメータを使用―コンフィギュレーション・ファイルで、サービスのログが標準エラーに書き込まれるように設定できます。SERVICES セクションで次を指定します。

  ```
  servopts -r
  ```

  ログの情報を分析するには、txrpt(1) コマンドを実行します。

  servopts(5) および txrpt(1) については、それぞれ『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』と『BEA Tuxedo コマンド・リファレンス』を参照してください。

- プログラムを使用―サービス・ルーチンの始めと終わりに time() への呼び出しを挿入します。実行時間が最も長いサービスでは負荷が最大になり、実行時間が最も短いサービスでは負荷が最小になります。time() の詳細については、C 言語ライブラリのマニュアルを参照してください。

サービスへの優先順位の割り当て

優先順位を割り当てることにより、アプリケーション内のデータの流れを強力に制御できます。つまり、重要度の高いリクエストに迅速にサービスを提供し、重要度の低いリクエストには後でサービスを提供することができます。また、特定のユーザや特定の状況に応じて、いつでも優先順位を設定することができます。

BEA Tuxedo サービスに対する優先順位は、以下のどちらかの方法で割り当てることができます。

- 管理パラメータを使用—コンフィギュレーション・ファイルの SERVICES セクションで定義されている各サービスに対し、PRIO パラメータを指定します。
- プログラムを使用—適切なクライアント・アプリケーションとサーバ・アプリケーションに対して tpsprio() 関数への呼び出しを追加し、指定されたクライアントとサーバだけが優先順位を動的に変更できるようにします。ただし、選ばれたクライアントだけがサービスの優先順位を高く設定できるようにするべきです。サーバがサービス要求を行うシステムでは、サーバ側から tpsprio() を呼び出し、サービス呼び出しの優先順位を上げることができます。これによりユーザは、必要なサービス要求を待たずに済みます。

優先順位の設定例

たとえば、サーバ 1 が、サービス A、B、および C を提供するとします。サービス A および B の優先順位は 50、サービス C の優先順位は 70 です。サービス C に対するリクエストは、サービス A または B に対するリクエストより常に先にキューから取り出されます。サービス A および B に対するリクエストは、互いに等しくキューから取り出されます。システムは、10 回に 1 回の割合で FIFO (first-in、first-out) 順序でリクエストをキューから取り出し、メッセージがキューで無制限に待機しないようにします。

PRIO パラメータを使用して性能を高める

PRIO は、サーバのキュー内にあるサービスの優先順位を決定するパラメータです。ただし、このパラメータの使用には注意が必要です。いったん優先順位が割り当てられると、キューからのメッセージの取り出しに時間がかかる場合があります。キューのメッセージの順序によっては、頻繁に取り出されないメッセージもあります。たとえば、キュー内に A、B、C の 3 つのメッセージがあり、C に対して 10 回以上のリクエストが送信されると、A と B は、10 回中 1 回しかキューから取り出されません。つまり、サービスの性能が低下し、ターンアラウンド・タイムが遅くなる可能性があります。

PRIO パラメータを使用するかを決定する場合は、以下の点を考慮してください。

- 高い方の優先順位が最優先されるため、通常は、呼び出し回数の少ないサービスに高い優先順位を割り当てます。
- メッセージは、FIFO に基づき 10 回に 1 回取り出されるため、優先順位の低いメッセージがキューに無制限にとどまることはありません。サービスに対して低い優先順位を割り当てる場合は、これらが応答時間を考慮する必要のないものであることを確認しておく必要があります。

複数のサービスをサーバへバンドルする

複数のサービスをサーバにパッケージ化する最も簡単な方法は、まったくパッケージしないことです。しかし、サービスをパッケージ化しないと、サーバ、メッセージ・キューおよびセマフォの数が許容範囲を超える原因となります。サービスをまったくバンドルしない (まとめない) 場合と細かくバンドルする (まとめる) 場合では、それぞれ長所と短所があります。

サービスのバンドルが必要な場合

次のうち、いずれかの状況または条件に当てはまる場合は、サービスをバンドルする (まとめる) ことをお勧めします。

- サービスの機能が類似している―アプリケーション内に類似するサービスがある場合は、それらのサービスを同じサーバにバンドルできます。アプリケーション側は、指定された時点で、すべてのサービスを提供するか、またはサービスをまったく提供しないかのどちらかを行います。たとえば、`bankapp` アプリケーションの例では、`WITHDRAW`、`DEPOSIT`、`INQUIRY` の 3 つのサービスを、窓口のオペレーションを扱うサーバにグループ化できます。このように内容の似たサービスをまとめると、サービスを管理しやすくなります。
- 類似するライブラリがある―同じライブラリを使用するサービスをバンドルすると、ディスク領域を節約できます。たとえば、同じ 100K のライブラリを使用する 3 つのサービスと、異なる 100K のライブラリを使用する 3 つのサービスがある場合、最初の 3 つのサービスをバンドルすると 200K の領域を節約できます。たいていの場合、同等の機能を持つサービスには類似するライブラリがあります。
- キュー―キューで処理できる範囲のサービスのみをサーバにバンドルしてください。空の MSSQ セットに追加された各サービスは、サーバの実行可能モジュールのサイズにはあまり影響しません。また、システム上のキューの数にはまったく影響しません。ただし、キューがいっぱいになるとシステムの性能は低下するため、対応策として別のサーバの実行可能モジュールを作成する必要があります。

同じサーバに、お互いを呼び出すサービス (呼び出し依存型サービス) を 2 つ以上設定しないでください。このサービスを設定すると、サーバは自身を呼び出し、デッドロックの原因となります。

システム全体のパフォーマンスの向上

パフォーマンスを向上するための以下の方法は、BEA Tuxedo リリース 8.0 以降で適用できます。

- サービスとインターフェイスをキャッシュする
- 認可および監査によるセキュリティを削除する
- マルチスレッド処理をオフにする
- XA トランザクションをオフにする

サービスとインターフェイスをキャッシュする

BEA Tuxedo リリース 8.0 以降では、サービス・エントリとインターフェイス・エントリをキャッシュして、掲示板をロックせずに、キャッシュされたサービスやインターフェイスのコピーを使用することができます。これによってパフォーマンスが大幅に向上します。特に、クライアント数が多く、サービスの数が少ないシステムで有効です。

コンフィギュレーション・ファイルの MACHINCES および SERVERS セクションに追加された SICACHEENTRIESMAX オプションを使用すると、プロセスやサーバ側で保持できるサービス・キャッシュ・エントリの最大数を指定できます。

クライアントやアプリケーションによっては、キャッシュが効果的でない場合があるため、キャッシュ・サイズを制御する環境変数 TMSICACHEENTRIESMAX が追加されています。また、TMSICACHEENTRIESMAX にはデフォルト値が設定されているため、以前のバージョンからアップグレードする場合は新たに設定を変更する必要がありません。キャッシュ・サイズの肥大化はクライアントにとって望ましくないため、TMSICACHEENTRIESMAX を使用してキャッシュ・エントリ数を制御することもできます。

サービス・キャッシュの制限事項

キャッシュ機能には以下の制限事項があります。

- サービスにルーティング基準がある場合、そのサービスはキャッシュされません。
- サービスにバッファ型の制限がある場合、そのサービスはキャッシュされません。

- サービスのグループがあらかじめ指定されている場合 (TMS サービス)、そのサービスはキャッシュされません。
- サービス・エントリの数がゼロの場合、キャッシュ処理は実行されません。

注記　SICACHEENTRIESMAX オプションの詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の UBBCONFIG(5) および TM_MIB(5) のセクションを参照してください。

TMSICACHEENTRIESMAX 変数の詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の tuxenv(5) のセクションを参照してください。

認可および監査によるセキュリティを削除する

BEA Tuxedo リリース 7.1 では、AAA (Authentication、Authorization、Auditing) セキュリティ機能が追加され、AAA プラグイン関数を使用したインプリメンテーションでは BEA Tuxedo の管理オプションによるセキュリティをベースにする必要がなくなりました。この結果、BEA Tuxedo 7.1 の主要なコード・パスでは、BEA Engine の AAA セキュリティ関数が常に呼び出されます。しかし、多くのアプリケーションではセキュリティ機能が使用されないため、BEA Engine のセキュリティ呼び出しによるオーバーヘッドは不要です。

BEA Tuxedo リリース 8.0 以降では、コンフィギュレーション・ファイルの RESOURCES セクションの OPTIONS パラメータに、NO_AA オプションが追加されています。NO_AA オプションを使用すると、認可および監査に関するセキュリティ関数の呼び出しを回避できます。認証はほとんどのアプリケーションで必要であるため、この機能をオフにすることはできません。

NO_AA オプションを有効にすると、以下の SECURITY パラメータ が影響を受けます。

- パラメータ NONE、APP_PW、および USER_AUTH は正常に動作しますが、認可または監査は行われません。

- ACL および MANDATORY_ACL パラメータは正常に動作しますが、デフォルトの BEA セキュリティ・メカニズムだけが使用されます。

注記　NO_AA オプションの詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の UBBCONFIG(5) および TM_MIB(5) のセクションを参照してください。

マルチスレッド化されたブリッジを使用する

ブリッジ・プロセスは、マルチ・マシン Tuxedo ドメイン内のホスト・マシンごとに 1 つずつ実行されます。このため、あるホスト・マシンからドメイン内のほかのホスト・マシンへのトラフィックは、すべて同じブリッジ・プロセスを通して送信されます。ブリッジ・プロセスでは、シングルスレッド実行機能とマルチスレッド実行機能がサポートされています。マルチスレッド化されたブリッジ処理を使用すると、データ・スループットを向上できる場合があります。マルチスレッド化されたブリッジ処理を有効にするには、UBBCONFIG ファイルの MACHINES セクションの BRTHREADS パラメータを設定します。

BRTHREADS=Y と設定すると、ブリッジ・プロセスはマルチスレッドで実行されます。BRTHREADS=N と設定するか、デフォルトの N のままにすると、ブリッジ・プロセスはシングルスレッドで実行されます。

ローカル・マシンを BRTHREADS=Y に、リモート・マシンを BRTHREADS=N に設定することも可能ですが、この場合のマシン間のスループットはシングルスレッド化されたブリッジ・プロセスより劣ります。

BRTHREADS パラメータを使用する際は、以下の点にも注意してください。

- BRTHREADS=Y に設定する意味があるのはマシンが複数の CPU を備えている場合のみですが、複数の CPU を備えていなくても BRTHREADS=Y に設定することは可能です。
- UBBCONFIG ファイルの RESOURCES セクションの MODEL パラメータを SHM に設定した場合、BRTHREADS パラメータは無視されます。
- BRTHREADS=Y に設定し、ブリッジ環境に TMNOTHREADS=Y が含まれている場合、ブリッジはスレッド・モードで起動し、警告メッセージがログに

記録されます。基本的には、BRTHREADS によって TMNOTHREADS がオーバライドされ、ブリッジが TMNOTHREADS の設定を無視したことを示す警告メッセージが記録されます。

注記　Tuxedo マルチ・マシン・ドメインでは、BRTHREADS=Y に設定しても、旧バージョンの Tuxedo が稼働するマシンには影響しません。

マルチスレッド化されたブリッジの詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の UBBCONFIG(5) で説明する MACHINES セクションの BRTHREADS パラメータを参照してください。

マルチスレッド処理をオフにする

BEA Tuxedo は、汎用的なスレッド機能を備えています。ただし、アーキテクチャの汎用性により、重要な状態情報を保護するため、すべての ATMI 呼び出しでミューテックス関数を呼び出す必要がありました。また、ライブラリで使用されるエンジンやキャッシュ・スキーマの階層構造により、さらにミューテックスが必要になります。スレッドを使用しないアプリケーションでは、これらのオプションをオフにすると、アプリケーション・コードを変更しなくても大幅にパフォーマンスを向上させることができます。

マルチスレッド処理をオフにするには、TMNOTHREADS 環境変数を使用します。この環境変数を設定することにより、新たに API 関数やフラグを導入しなくても、個々のプロセスでスレッドをオンまたはオフにすることができます。

TMNOTHREADS=Y に設定すると、ミューテックス関数の呼び出しが回避されます。

注記　TMNOTHREADS の詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の tuxenv(5) のセクションを参照してください。

XA トランザクションをオフにする

すべての BEA Tuxedo アプリケーションで XA トランザクションを使用するのではないにもかかわらず、すべてのプロセスで、内部トランザクション関数の呼び出しによってトランザクション処理の負担がかかります。BEA Tuxedo リリース 8.0 以降では、XA トランザクションを使用しないアプリケーションでパフォーマンスを向上させるため、コンフィギュレーション・ファイルの RESOURCES セクションの OPTIONS パラメータに NO_XA フラグが追加されています。

NO_XA フラグをセットすると、XA トランザクションは許可されなくなります。NO_XA オプションが指定されている場合、GROUPS セクションで TMS サービスを設定しようとすると失敗するので注意してください。

注記　NO_XA オプションの詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の UBBCONFIG(5) および TM_MIB(5) のセクションを参照してください。

システムの IPC 要件の決定

システムの IPC 要件は、以下のシステム・パラメータを使用して決定します。

- MAXACCESSERS
- REPLYQ
- RQADDR
- MAXSERVERS
- MAXSERVICES
- MAXGTT

tmboot -c コマンドを使用すると、コンフィギュレーションの IPC に関する最低条件を表示できます。

次の表は、このようなシステム・パラメータの説明です。

表 8-1IPC 資源のチューニング・パラメータ

パラメータ	説明
MAXACCESSSERS	セマフォの数を指定します。 メッセージ・キューの数は、「MAXACCESSERS + 応答キューを持つサーバ数 (MSSQ セットのサーバ数 + MSSQ セット数)」とほぼ同じです。
MAXSERVERS、MAXSERVICES、および MAXGTT	MAXSERVERS、MAXSERVICES、MAXGTT の 3 つのパラメータ、および ROUTING、GROUP、NETWORK の 3 つのセクションの全体のサイズは、共用メモリのサイズに影響します。これらのパラメータの相関関係を計算式に表そうとすると複雑になりますが、単純に tmboot -c または tmloadcf -c を実行すると、アプリケーションで最低限必要な IPC の要件を計算することができます。
キューに関連するカーネル・パラメータ	クライアントとサーバ間のバッファ・トラフィックのフローを管理するためにチューニングします。キューの最大サイズ (バイト単位) は、アプリケーションで最も大きいメッセージを処理できる大きさに設定します。通常、キューの 75 ～ 85% は使用中です。この割合が低いとキューが無駄になり、割合が高いとメッセージ送信が頻繁にブロックされる原因になります。 アプリケーションが送信するメッセージに対して最大のバッファを処理できるように最大サイズを設定します。 最大待ち行列長 (キューに同時にとどまることができる最大メッセージ数) は、アプリケーションにおけるオペレーションに適した大きさに設定します。 キューの平均使用率と平均長を測定するには、アプリケーションをシミュレートするか、または実行します。これにより、アプリケーションの実行前にチューニング可能なパラメータを評価でき、アプリケーションの性能分析後にアプリケーションを正しくチューニングできます。 大規模なシステムでは、オペレーティング・システム・カーネルのサイズに対するパラメータ設定の効果を分析します。効果が認められない場合には、アプリケーション・プロセスの数を減らすか、またはアプリケーションをさらに多くのマシンに分散して MAXACCESSERS を減らします。

IPC パラメータの調整

以下のアプリケーション・パラメータを設定すると、システムの効率を高めることができます。

- MAXACCESSERS、MAXSERVERS、MAXINTERFACES、および MAXSERVICES
- MAXGTT、MAXBUFTYPE、および MAXBUFSTYPE
- SANITYSCAN、BLOCKTIME、および個々のトランザクション・タイムアウト値
- BBLQUERY および DBBLWAIT

MAXACCESSERS、MAXSERVERS、MAXINTERFACES、および MAXSERVICES パラメータの設定

MAXACCESSERS、MAXSERVERS、MAXINTERFACES、および MAXSERVICES の各パラメータを設定すると、セマフォと共用メモリにかかる負担が増えます。これらのパラメータを使用する前にはこの点を考慮し、システムのニーズを満たす最適な値を設定する必要があります。将来システムを移行する場合に備え、余分な資源を確保しておくことも必要です。また、システムに同時にアクセスできるクライアント数を設定できるようにしておく必要もあります。これらのパラメータには、デフォルトで IPC 資源が適度に割り当てられていますが、必要最低限の値を設定するのが賢明です。

MAXGTT、MAXBUFTYPE、および MAXBUFSTYPE パラメータの設定

デフォルト値がアプリケーションで適切かどうかを判断するには、システム内のクライアント数とそれらのクライアントがトランザクションにコミットする時間の割合を掛けた値を算出します。この値が 100 に近い場合は、MAXGTT パラメータ値を増やす必要があります。MAXGTT パラメータの値を増やすと、次のような結果になります。

- コミットの速度によっては、クライアントの数を増やす必要があります。
- マシンによっては、TLOGSIZE の値を増やす必要があります。
- 分散型のトランザクションを使用しないアプリケーションでは、MAXGTT を 0 に設定する必要があります。

アプリケーションで使用するバッファ型の数を制限するには MAXBUFTYPE パラメータを設定し、バッファのサブタイプの数を制限するには MAXBUFSTYPE パラメータを設定します。MAXBUFTYPE の現在のデフォルト値は 16 です。ユーザ定義のバッファ型を 8 つ以上作成する予定がある場合は、MAXBUFTYPE に高い数値を設定してください。このパラメータの値を特に指定しない場合は、デフォルト値が使用されます。

MAXBUFSTYPE の現在のデフォルト値は 32 です。多くの VIEW サブタイプを使用する予定がある場合は、このパラメータに高い数値を設定してください。

SANITYSCAN、BLOCKTIME、BBLQUERY、および DBBLWAIT パラメータの設定

システムが遅いプロセッサで稼動している場合 (使用率が高い場合など) 時間間隔を調整するタイミング・パラメータである SANITYCAN、BLOCKTIME、およびトランザクションのタイムアウト値を増やします。

ネットワークの動作が遅い場合は、BLOCKTIME、BBLQUERY および DBBLWAIT パラメータの値を増やします。

チューニング関連パラメータの推奨値

次の表は、アプリケーションのチューニングに使用するシステム・パラメータと推奨値です。

パラメータ	目的
MAXACCESSERS、MAXSERVERS、MAXINTERFACES、および MAXSERVICES	必要最低限の値を設定します (IPC 資源を確保するため)。クライアントを追加できるように設定します。
MAXGTT、MAXBUFTYPE、および MAXBUFSTYPE	MAXGTT の値を増やして、多数のクライアントを許容します。トランザクション処理を行わないアプリケーションでは、MAXGTT を 0 に設定します。 ユーザ定義のバッファ型を 8 つ以上作成する場合に限り、MAXBUFTYPE を使用します。 多くの種類の VIEW サブタイプを使用する場合は、MAXBUFSTYPE の値を増やします。
BLOCKTIME、TRANTIME、および SANITYSCAN	システムの動作が遅い場合は、値を増やします。
BLOCKTIME、TRANTIME, BBLQUERY、および DBBLWAIT	ネットワークの動作が遅い場合は、値を増やします。

システム・トラフィックの測定

一定の交通量がある道路では渋滞が起こるように、システムでも同様にボトルネックが発生します。実際の高速道路を走る車両数は、道路に渡されたケーブルによってカウントされます。つまり、このケーブルの上を車両が通過するたびに、カウンタの値が増加します。

これと同様に、サービスのトラフィックを測定することができます。たとえば、サーバの起動時 (tpsvrinit() の呼び出し時) に、グローバル・カウンタを初期化して開始時間を記録することができます。以降、特定のサービスが呼び出されるたびにカウンタ値は増加します。tpsvrdone() 関数を呼び出してサーバをシャットダウンすると、最終カウンタ値と終了時間が記録されます。このメカニズムにより、一定期間に特定サービスのビジー状態がどの程度であるかを判断できます。

BEA Tuxedo システムでは、問題のあるデータ・フローのパターンが原因でボトルネックが生じます。ボトルネックを検出する最も早い方法は、関連するサービスに要する時間をクライアント側から測定することです。

システム・ボトルネックの検出方法

たとえば、クライアント 1 が結果を出力すると 4 秒かかるとします。time() を呼び出すと、サービス A に対する tpcall が動作を 3.7 秒遅らせていることがわかり、さらにサービス A を開始点と終了点で監視すると、その動作に 0.5 秒かかっています。tmadmin の pq コマンドを使用すると、キューがいっぱいであることがわかります。

一方、サービス A の実行に 3.2 秒かかるとします。サービス A の個々の部分はまとめて測定できます。サービス A がサービス B に対して tpcall を発行し、この動作に 2.8 秒かかっていることも考えられます。この場合、キュー時間またはメッセージ送信ブロッキング時間を分離できます。適切な時間を識別すると、アプリケーションはトラフィックの処理に戻されます。

time() を使用すると、次の項目の所要時間を測定できます。

- クライアント・プログラム全体
- 単一のクライアント・サービス要求
- サービス機能全体
- サービス要求を行うサービス機能 (ある場合)

UNIX プラットフォームにおけるボトルネックの検出

UNIX システムの sar(1) コマンドを使用すると、システムのボトルネックの検出に役立つ性能に関する情報が表示されます。sar(1) は、次の目的に使用できます。

- あらかじめ決められた間隔でオペレーティング・システムの累積アクティビティ・カウンタをサンプリングする。
- システム・ファイルからデータを抽出する。

次の表は、sar(1) コマンドのオプションの説明です。

オプション	目的
-u	CPU の使用率を示します。システムが、ユーザ・モード、システム・モード、待機状態 (ブロック I/O を待つプロセスを持ったままアイドル状態)、およびアイドル状態である時間を割合で示します。
-b	バッファのアクティビティ (システム・バッファとディスク (またはほかのブロック・デバイス) との間で送信される 1 秒あたりのデータ転送数など) を報告します。
-c	すべての種類のシステム・コールと、fork(2) や exec(2) などの特定のシステム・コールのアクティビティを報告します。
-w	システムのスワッピング・アクティビティ (スワップインとスワップアウトの転送数など) を監視します。
-q	専有時のキューの長さの平均および専有時間の割合を報告します。
-m	メッセージとシステム・セマフォのアクティビティ (1 秒あたりのプリミティブの数) を報告します。
-p	ページング・アクティビティ (アドレス変換ページ障害、ページ障害と保護エラー、およびフリー・リストに再利用された有効ページなど) を報告します。
-r	未使用のメモリ・ページとディスク・ブロック (ユーザ・プロセスで使用できる平均ページ数、プロセス・スワッピングに対して使用できるディスク・ブロックなど) を報告します。

注記　UNIX システムの中には、sar(1) コマンドの代わりに別のコマンドが用意されているものもあります。たとえば、BSD には iostat(1) コマンド、Sun には perfmeter(1) が用意されています。

Windows 2000 プラットフォームにおけるボトルネックの検出

Windows 2000 プラットフォームでは、パフォーマンス・モニタを使用して、システム情報を収集しボトルネックを検出します。パフォーマンス・モニタを開くには、[スタート] メニューから次の項目を選択します。

[スタート] —> [プログラム] —> [管理ツール] —> [パフォーマンス モニタ]

関連項目

- 『BEA Tuxedo アプリケーションの設定』の 8-1 ページの「分散型の ATMI アプリケーション用のコンフィギュレーション・ファイルの作成」
- 『BEA Tuxedo アプリケーションの設定』の 9-1 ページの「分散アプリケーションのネットワーク設定」
- 4-1 ページの「分散アプリケーションでのネットワーク管理」
- 『BEA Tuxedo CORBA アプリケーションのスケーリング、分散、およびチューニング』

章

9 BEA Tuxedo アプリケーションのトラブルシューティング

ここでは、次の内容について説明します。

- 障害の種類の判別
- 任意通知型メッセージのブロードキャスト
- システム・ファイルの保守
- 障害回復時の注意事項
- 分断されたネットワークの修復
- 障害が発生したマシンの復元
- システム・コンポーネントの置換
- アプリケーション・コンポーネントの置換
- 手動によるサーバのクリーンナップと再起動
- トランザクションのアボートとコミット
- トランザクション使用時における障害回復
- アプリケーションが正しくシャットダウンされない場合の IPC ツールの使用

- マルチスレッド化またはマルチコンテキスト化されたアプリケーションのトラブルシューティング

障害の種類の判別

トラブルシューティングの最初のステップとして、まずどこで障害が発生したかを判別します。ほとんどのアプリケーションでは、障害の発生源として以下の 6 か所が考えられます。

- アプリケーション
- BEA Tuxedo システム
- データベース管理ソフトウェア
- ネットワーク
- オペレーティング・システム
- ハードウェア

障害が発生した場所を特定できたら、各箇所の担当者と協力して障害を解決してください。たとえば、ネットワーキングに関する障害が発生した場合は、ネットワーク管理者と協力して解決してください。

アプリケーション障害の原因の判別

アプリケーション障害の原因を検出するには、次の手順に従います。

1. ユーザ・ログ (ULOG) 内で、BEA Tuxedo システムの警告メッセージとエラー・メッセージを調べます。
2. 発生した問題を反映していると思われる内容のメッセージをいくつか選択します。各メッセージのカタログ名とメッセージ番号を書き留めておき、該当するメッセージを『システム・メッセージ』で調べます。マニュアルの各項目には次の情報が含まれています。
   - メッセージが示すエラーの詳細内容

- エラーを解決するための推奨措置

3. ULOG 内でアプリケーションの警告メッセージとエラー・メッセージを調べます。
4. アプリケーション・サーバおよびアプリケーション・クライアントによって生成された警告とエラーを調べます。このようなメッセージは通常、標準出力ファイルおよび標準エラー・ファイルに送られます (デフォルト名はそれぞれ stdout および stderr)。
   - stdout ファイルおよび stderr ファイルは APPDIR 変数で定義されたディレクトリにあります。
   - クライアント側とサーバ側に設定されている stdout ファイルおよび stderr ファイルの名前は、変更されている場合があります。stdout ファイルと stderr ファイルの名前を変更するには、コンフィギュレーション・ファイル内で、該当するクライアントとサーバの定義にそれぞれ -e または -o を指定します。詳細については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の servopts(5) を参照してください。
5. APPDIR 変数で定義されたディレクトリにコア・ダンプがないかどうかを調べます。dbx などのデバッガを使用してスタック・トレースを実行します。コア・ダンプが見つかった場合は、アプリケーション開発者に通知します。
6. sar(1) などのコマンドを実行してシステム・アクティビティ・レポートを表示し、システムが正しく機能しない原因を調べます。以下の原因が考えられます。
   - システムのメモリが不足している。
   - カーネルが正しくチューニングされていない。

BEA Tuxedo システムでの障害の原因の判別

システム障害の原因を検出するには、次の手順に従います。

1. ユーザ・ログ (ULOG) 内で、BEA Tuxedo システムの警告メッセージとエラー・メッセージを調べます。

- TPEOS メッセージは、オペレーティング・システムにエラーが発生したことを示します。
- TPESYSTEM メッセージは、BEA Tuxedo システムにエラーが発生したことを示します。

2. 発生した問題を反映していると思われる内容のメッセージをいくつか選択します。各メッセージのカタログ名とメッセージ番号を書き留めておき、該当するメッセージを『システム・メッセージ』で調べます。マニュアルの各項目には次の情報が含まれています。
   - メッセージが示すエラーの詳細内容
   - エラーを解決するための推奨措置
3. 次の手順でデバッグの準備をします。
   - suspend サービスをシャットダウンします。
   - tmboot -n -s(server) -d1 を実行します(このコマンドを実行してもサーバは起動しませんが、サーバを起動するためのコマンド行が出力されます)。dbx などのデバッガを指定して、このコマンド行を使用してください。

任意通知型メッセージのブロードキャスト

イベント・ブローカのイベント・サマリ (システム全体に渡るサマリ) 生成機能とメカニズム (イベントにより通知を発行) により、トラブルシューティングの機能が拡張されます。イベント・ブローカは、BEA Tuxedo のシステム・イベント (サーバの停止やネットワークの障害など) やアプリケーション・イベント (ATM 機のメモリ不足など) の詳細を報告します。あるイベントに関する任意通知型通知を受信した BEA Tuxedo クライアントは、呼び出すサービス・ルーチンを指定したり、以降のデータの格納先とするアプリケーション・キューを指定することができます。また、任意通知型通知を受信した BEA Tuxedo サーバは、サービス要求を指定したり、データの格納先とするアプリケーション・キューを指定できます。

1. 任意通知型メッセージを送信するには、次のコマンドを入力します。

   `broadcast (bcst) [-m machine] [-u usrname] [-c cltname] [text]`

   注記　デフォルトでは、メッセージはすべてのクライアントに送信されます。

2. ただし、メッセージの送信先を次のいずれかに制限することもできます。
   - 特定のマシン (`-m machine`)
   - 特定のクライアント・グループ (`-c client_group`)
   - 特定のユーザ (`-u user`)

メッセージは、80 文字以内で記述します。システムは `STRING` バッファ型でメッセージを送信します。つまり、クライアントの任意通知型メッセージ処理関数 (`tpsetunsol(0)` で指定) では、この型のメッセージを処理できなければなりません。この場合、`tptypes()` 関数を使用すると便利です。

関連項目

- 『BEA Tuxedo システム入門』の 2-19 ページの「任意通知型通信」
- 『BEA Tuxedo システム入門』の 4-19 ページの「イベント・ブローカを使用したイベントの管理」

システム・ファイルの保守

ファイル・システムを保守するため、次のタスクを定期的に行う必要があります。

- 汎用デバイス・リスト (UDL: Universal Device List) の出力
- VTOC 情報の出力
- デバイスの再初期化

- デバイス・リストの作成
- デバイス・リストの破棄

注記　このファイル形式は、TUXCONFIG、TLOG、および /Q で使用されます。

汎用デバイス・リスト (UDL: Universal Device List) の出力

UDL を出力するには、次の手順に従います。

1. tmadmin -c を実行します。
2. 次のコマンドを入力します。
   ```
   lidl
   ```
3. UDL を取得するデバイスを指定する方法は、次の 2 つです。
   - lidl コマンド行で次のように指定します。
     ```
     -z device_name [devindx]
     ```
   - 希望するデバイスの名前に環境変数 FSCONFIG を指定します。

VTOC 情報の出力

VTOC 情報を出力するには、次の手順に従います。

1. tmadmin -c を実行します。
2. すべての VTOC テーブル・エントリの情報を取得するには、次のコマンドを入力します。
   ```
   livtoc
   ```
3. VTOC を取得するデバイスを指定する方法は、次の 2 つです。
   - lidl コマンド行で次のように指定します。

-z *device name* [*devindx*]

- 希望するデバイスの名前に環境変数 FSCONFIG を指定します。

デバイスの再初期化

デバイス・リスト内のデバイスを再初期化するには、次の手順に従います。

1. tmadmin -c を実行します。
2. 次のコマンドを入力します。

   initdl [-z *devicename*] [-yes] *devindx*

   注記　*devindx* には、破棄するファイルのインデックスを指定します。
3. 次の方法のいずれかでデバイスを指定できます。
   - -z オプションの後にデバイス名を指定します (上記を参照)。
   - デバイス名に環境変数 FSCONFIG を指定します。
4. コマンド行で -yes オプションを指定すると、ファイルを削除するかどうかを確認するプロンプトは表示されません。

デバイス・リストの作成

デバイス・リストを作成するには、次の手順に従います。

1. tmadmin -c を実行します。
2. 次のコマンドを入力します。

   crdl [-z *devicename*] [-b *blocks*]

   - *devicename* [*devindx*] には、希望するデバイス名を指定します。新しいデバイスに名前を割り当てる別の方法として、環境変数 FSCONFIG に希望するデバイス名を設定することもできます。
   - *blocks* には、必要なブロックの数を指定します。デフォルト値は 1000 です。

注記　TLOG に関連する管理オーバーヘッド用に 35 ブロックが必要なため、TLOG を作成する場合は、35 より大きい値を割り当てます。

デバイス・リストの破棄

devindx のインデックスを持つデバイス・リストを破棄するには、次の手順に従います。

1. `tmadmin -c` を実行します。
2. 次のコマンドを入力します。

   `dsdl [-z devicename] [yes] [devindx]`

   注記　*devindx* には、破棄するファイルのインデックスを指定します。
3. 次の方法のいずれかでデバイスを指定できます。
   - `-z` オプションの後にデバイス名を指定します (上記を参照)。
   - デバイス名に環境変数 `FSCONFIG` を指定します。
4. コマンド行で yes オプションを指定すると、ファイルを破棄するかどうかを確認するプロンプトは表示されません。

障害回復時の注意事項

BEA Tuxedo システムの機能性を最適にするには、環境がある程度安定している必要があります。BEA Tuxedo の管理サブシステムには、ネットワーク、マシン、およびアプリケーション・プロセスの障害から回復するための優れた機能が用意されていますが、これらも完璧ではありません。BEA Tuxedo システムが以下のように機能することを理解しておいてください。

`SYSTEM_ACCESS` の `FASTPATH` モデル (デフォルト) を使用するアプリケーション・クライアントおよびアプリケーション・サーバは、BEA Tuxedo 共有データ構造へのダイレクト・メモリ・アクセスが可能です。`FASTPATH` モデルを使用すると、BEA Tuxedo システムの性能を最大限に引き出すことがで

きます。BEA Tuxedo システムでは、オペレーティング・システムが提供する IPC (InterProcess Communication) 機能とファイル・システム機能を使用します。

アプリケーションで、これらの機能を誤って使用して BEA Tuxedo 共用メモリまたは BEA Tuxedo ファイル記述子に書き込みを行った場合や、誤ってほかの BEA Tuxedo システム・リソースを使用した場合は、データが破壊されたり、BEA Tuxedo の機能が損なわれたり、アプリケーションが停止したりするおそれがあります。

アプリケーション・クライアント、アプリケーション・サーバ、および BEA Tuxedo 管理プロセスは、重要な部分 (つまり、共用メモリ内の共有情報の更新) で実行されている可能性があるため、ユーザや管理者がこれらのプロセスを直接終了するのは適切ではありません。メモリの更新中に重要な部分への割り込みを行うと、内部データ構造に矛盾が生じるおそれがあります。これは、BEA Tuxedo システム固有の問題ではなく、共有データを使用するすべてのシステムに共通の問題です。BEA Tuxedo ユーザ・ログ内のエラー・メッセージがロックやセマフォを参照している場合、このようなデータの破壊が発生したことを示している可能性があります。

アプリケーションの可用性を最大にするには、BEA Tuxedo システムの冗長性管理機能 (複数サーバ、マルチ・マシン、マルチ・ドメインなどの機能) を活用します。アプリケーションの機能を分散しておくと、1 つの領域で障害が発生した場合でもオペレーションを継続できます。

分断されたネットワークの修復

この節では、ネットワーク分断の原因を見つけ、回復するためのトラブルシューティングについて説明します。1 つまたは複数のマシンから MASTER マシンにアクセスできない場合は、ネットワークが分断されています。アプリケーションの管理者は、ネットワークの分断を見つけ、回復する責任があります。

ネットワークの分断は、次のいずれかの原因で発生します。

- ネットワークの障害。一時的な障害と重大な障害があります。一時的な障害は、発生後、数分で自動的に回復しますが、重大な障害では、分断されたマシンをネットワークから外す必要があります。
- MASTER マシンまたは非マスタ・マシンで発生するマシンの障害。
- BRIDGE の障害。

分断されたネットワークの回復手順は、分断された原因によって異なります。

分断されたネットワークの検出

分断されたネットワークは、次のいずれかの方法で検出できます。

- ユーザ・ログ (ULOG) 内のメッセージを調べ、問題の原因に相当する内容を探します。
- この目的に沿った tmadmin のコマンドを実行して、ネットワーク、サーバ、およびサービスに関する情報を収集します。

ULOG のチェック

ネットワークで問題が発生すると、BEA Tuxedo システムの管理サーバは ULOG へメッセージを送り始めます。ULOG がリモート・ファイル・システム上で設定されている場合、すべてのメッセージは同じログに書き込まれます。したがって、1 つのファイルで tail(1) コマンドを実行し、画面に表示される障害メッセージを調べることができます。

ただし、リモート・ファイル・システムが、障害のあるネットワークを使用している場合、リモート・ファイル・システムが使用できない場合があります。

コード リスト 9-1 ULOG のエラー・メッセージの例

```
151804.gumby!DBBL.28446:... : ERROR: BBL partitioned, machine=SITE2
```

ネットワーク、サーバ、およびサービスに関する情報の収集

次のリストは、分断されたネットワークおよびそのネットワーク上のサーバとサービスに関する情報を収集する tmadmin セッションの例です。次の 3 つの tmadmin コマンドが実行されます。

- pnw (printnetwork コマンド)
- psr (printserver コマンド)
- psc (printservice コマンド)

コード リスト 9-2 tmadmin **セッションの例**

```
$ tmadmin
> pnw SITE2
Could not retrieve status from SITE2

> psr -m SITE1
a.out Name   Queue Name   Grp Name   ID   Rq Done   Load Done   Current Service
BBL          30002.00000  SITE1       0      -         -            ( - )
DBBL         123456       SITE1       0     121       6050          MASTERBB
simpserv     00001.00001  GROUP1      1      -         -            ( - )
BRIDGE       16900672     SITE1       0      -         -            ( DEAD )
>psc -m SITE1
Service Name Routine Name a.out    Grp Name ID Machine # Done Status
------------ ------------ -------- -------- -- ------- ------------
ADJUNCTADMIN ADJUNCTADMIN BBL      SITE1     0   SITE1      - PART
ADJUNCTBB    ADJUNCTBB    BBL      SITE1     0   SITE1      - PART
TOUPPER      TOUPPER      simpserv GROUP1    1   SITE1      - PART
BRIDGESVCNM  BRIDGESVCNM  BRIDGE   SITE1     1   SITE1      - PART
```

ネットワーク接続の回復

この節では、一時的なネットワーク障害および重大なネットワーク障害からの回復方法を説明します。

一時的なネットワーク障害からの回復

一時的なネットワーク障害は、BRIDGE によって自動的に回復され、再接続が行われるため、通常ユーザ側ではこの障害の発生はわかりません。ただし、一時的なネットワーク障害を手動で回復する必要がある場合は次の手順に従います。

1. MASTER マシンで tmadmin(1) セッションを開始します。
2. reconnect コマンド (rco) を実行し、分断されていないマシンと分断されたマシンの名前を指定します。

   ```
   rco non-partioned_node1 partioned_node2
   ```

重大なネットワーク障害からの回復

重大なネットワーク障害を回復するには、次の手順に従います。

1. MASTER マシンで tmadmin セッションを開始します。
2. pclean コマンドを実行して、分断されたマシンの名前を指定します。

   ```
   pcl partioned_machine
   ```

3. アプリケーション・サーバを移行するか、または障害の修復後にマシンを再起動します。

障害が発生したマシンの復元

障害が発生したマシンを復元する手順は、マシンが MASTER マシンであるかどうかによって異なります。

障害が発生した MASTER マシンの復元

障害が発生した MASTER マシンを復元するには、次の手順に従います。

1. BEA Tuxedo のプロセスに対する IPC 資源がすべて除去されていることを確認します。
2. ACTING MASTER (SITE2) で tmadmin セッションを開始します。

   ```
   tmadmin
   ```

3. 次のコマンドを入力して MASTER (SITE1) で BBL を起動します。

   ```
   boot -B SITE1
   ```

   SITE1 で pclean を実行していない場合、BBL は起動できません。
4. tmadmin セッションで、以下のように入力して、MASTER サイト (SITE1) で再度 DBBL を実行します。

   ```
   MASTER
   ```

5. 障害が発生したマシンからアプリケーション・サーバとデータを移行した場合は、再起動するか、または移行を元に戻します。

障害が発生した非マスタ・マシンの復元

障害が発生した非マスタ・マシンを復元するには、次の手順に従います。

1. MASTER マシンで tmadmin セッションを開始します。
2. pclean を実行し、分断されたマシンをコマンド行で指定します。
3. マシンの問題を修正します。
4. MASTER マシンから BBL を起動して、障害の発生したマシンを復元します。
5. 障害が発生したマシンからアプリケーション・サーバとデータを移行した場合は、再起動するか、または移行を元に戻します。

次のリストでは、非マスタ・マシン SITE2 を復元する例を示します。

コード リスト 9-3 障害が発生した非マスタ・マシンの復元の例

```
$ tmadmin
tmadmin - Copyright © 1987-1990 AT&T; 1991-1993 USL. All rights reserved

> pclean SITE2
Cleaning the DBBL.

Pausing 10 seconds waiting for system to stabilize.
3 SITE2 servers removed from bulletin board

> boot -B SITE2
Booting admin processes ...

Exec BBL -A :

on SITE2 -> process id=22923 ... Started.
1 process started.
> q
```

システム・コンポーネントの置換

BEA Tuxedo システムのコンポーネントを置換するには、次の手順に従います。

1. 置換したい BEA Tuxedo システム・ソフトウェアをインストールします。
2. 変更によって影響を受ける次のアプリケーションの部分をシャットダウンします。
   - ライブラリが更新された場合は、BEA Tuxedo システム・サーバをシャットダウンします。
   - 関連する BEA Tuxedo システムのヘッダ・ファイルまたは静的ライブラリが置換された場合は、アプリケーションのクライアントとサーバをシャットダウンして再度ビルドする必要があります。ただし、BEA Tuxedo システムのメッセージ・カタログ、システム・コマンド、管

理サーバ、または共用オブジェクトが置換された場合、アプリケーションのクライアントとサーバのビルドは必要ありません。

3. 関連する BEA Tuxedo システムのヘッダ・ファイルおよび静的ライブラリが置換される場合は、アプリケーションのクライアントとサーバを再度ビルドします。
4. シャットダウンしたアプリケーション部分を再起動します。

アプリケーション・コンポーネントの置換

アプリケーションのコンポーネントを置換するには、次の手順に従います。

1. アプリケーション・ソフトウェアをインストールします。アプリケーション・ソフトウェアの構成要素には、アプリケーション・クライアント、アプリケーション・サーバ、および管理ファイル (FML フィールド・テーブルなど) があります。
2. 置換するアプリケーション・サーバをシャットダウンします。
3. 必要に応じて、新しいアプリケーション・サーバをビルドします。
4. 新しいアプリケーション・サーバを起動します。

手動によるサーバのクリーンナップと再起動

デフォルトでは、BEA Tuxedo システムはデッド・プロセスに関連付けられた資源 (キューなど) をクリーンナップし、BBL スキャン中に定期間隔で掲示板 (BB) から再起動可能な停止したサーバを再起動します。ただし、ほかの時点でもクリーンナップを要求できます。

デッド・プロセスに関連付けられた資源のクリーンナップ

デッド・プロセスに関連付けられた資源を直ちにクリーンナップするには、次の手順に従います。

1. `tmadmin` セッションを開始します。
2. `bbclean` *`machine`* を入力します。

`bbclean` コマンドには、オプションの引数として、クリーンナップされるマシンの名前を指定できます。

引数	**結果**
指定しない場合	デフォルト設定のマシン上の資源がクリーンナップされます。
マシンを指定する場合	指定したマシン上の資源がクリーンナップされます。
DBBL	DBBL (Distinguished Bulletin Board Liaison) およびすべてのサイトの掲示板の資源がクリーンナップされます。

ほかの資源のクリーンナップ

ほかの資源をクリーンナップするには、次の手順に従います。

1. `tmadmin` セッションを開始します。
2. `pclean` *`machine`* を入力します。

注記　*`machine`* には値を指定する必要があります。これは必須引数です。

指定するマシン	結果
分断されていないマシン	`pclean` は `bbclean` を呼び出します。
分断されたマシン	`pclean` は、分断されていないすべての掲示板からサーバとサービスの全エントリを削除します。

このコマンドは、不意に分断が生じた後でシステムを正常に復元する場合に便利です。

BEA Tuxedo CORBA サーバの起動順序の確認

BEA Tuxedo CORBA アプリケーションの起動に失敗した場合、テキスト・エディタでアプリケーションの `UBBCONFIG` ファイルを開き、`SERVERS` セクションでサーバが正しい順序で起動されているかどうかを確認します。BEA Tuxedo CORBA 環境での正しいサーバの起動順序は以下のとおりです。この規則に違反した場合、BEA Tuxedo CORBA アプリケーションは起動しません。

サーバの起動順序

1. システムのイベント・ブローカ、TMSYSEVT。
2. -N および -M オプションが設定された TMFFNAME サーバ。NameManager サービスを (MASTER として) 起動します。このサービスは、アプリケーション側で提供される名前とオブジェクト参照のマッピングを維持します。
3. -N オプションのみ設定された TMFFNAME サーバ。スレーブ NameManager サービスを起動します。
4. -F オプションが設定された TMFFNAME サーバ。FactoryFinder を起動します。
5. ファクトリを宣言するアプリケーション・サーバ。

詳細については、『BEA Tuxedo アプリケーションの設定』の 3-85 ページの「CORBA C++ サーバの起動順序」を参照してください。

BEA Tuxedo CORBA サーバのホスト名形式と大文字 / 小文字の確認

プログラマが Bootstrap オブジェクト・コンストラクタまたは TOBJADDR で指定したネットワーク・アドレスは、サーバ・アプリケーションの UBBCONFIG ファイルにあるネットワーク・アドレスと完全に一致していなければなりません。アドレスの形式や、大文字 / 小文字も識別されます。アドレスが一致しない場合、Bootstrap オブジェクト・コンストラクタの呼び出しが失敗し、一見無関係と思われる以下のエラー・メッセージが表示されます。

```
ERROR: Unofficial connection from client at
<tcp/ip address>/<port-number>:
```

たとえば、ネットワーク・アドレスが、サーバ・アプリケーションの UBBCONFIG ファイルにある ISL コマンド行のオプション文字列で //TRIXIE:3500 に指定されている場合、Bootstrap オブジェクト・コンストラクタや TOBJADDR で //192.12.4.6:3500 または //trixie:3500 のいずれかを指定すると、接続が失敗します。

UNIX システムでは、ホスト・システムで uname -n コマンドを使用して大文字 / 小文字を指定します。Windows 2000 システムでは、ホスト・システムでコントロール パネルの [ネットワーク] を使用して大文字 / 小文字を指定します。

BEA Tuxedo CORBA クライアントの起動失敗の原因

BEA Tuxedo CORBA アプリケーションを実行する Windows 2000 サーバで、一部のインターネット ORB 間プロトコル (IIOP) クライアントを起動した後、//host:port が正しく指定されているにもかかわらず、一部のクライアントで Bootstrap オブジェクトを作成できず、InvalidDomain メッセージが返される場合は、以下の手順を実行できます。関連情報については、9-18 ページの「BEA Tuxedo CORBA サーバのホスト名形式と大文字 / 小文字の確認」を参照してください。

1. regedt32 (レジストリ・エディタ) を起動します。
2. [ローカル マシン上の HKEY_LOCAL_MACHINE] ウィンドウを表示します。
3. 以下のキーを選択します。

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Afd\Parameters
```

4. [編集] メニューの [値の追加] を使って以下の値を追加します。

```
DynamicBacklogGrowthDelta:REG_DWORD :0xa
EnableDynamicBacklog:REG_DWORD:0x1
MaximumDynamicBacklog:REG_DWORD:0x3e8
```

```
MinimumDynamicBacklog:REG_DWORD:0x14
```

5. Windows 2000 を再起動し、変更を有効にします。

これらの値を追加することにより、5 つの保留接続が保持される静的接続キュー (バックログ) が、動的接続バックログに置き換えられます。この動的接続バックログには、最小 20 エントリ (最小 0x14)、最大 1000 エントリ (最大 0x3e8) を格納でき、最小値から最大値の間で 10 ずつ増加できます (0xa 間隔)。

これらの設定は、システムが受信した接続にのみ適用されますが、IIOP リスナでは受け付けられません。マイクロソフトでは、最小値 20、間隔 10 という値を推奨しています。最大値はマシンによって異なりますが、マイクロソフトでは Windows 2000 サーバの場合 5000 を超えない値を推奨しています。

トランザクションのアボートとコミット

この節では、トランザクションのアボートとコミットの方法について説明します。

トランザクションのアボート

トランザクションをアボートするには、次の手順に従います。

1. 次のコマンドを入力します。

   ```
   aborttrans (abort) [-yes] [-g groupname] tranindex
   ```

2. *tranindex* の値を決定するために、printtrans コマンド (tmadmin コマンド) を実行します。

3. *groupname* が指定されると、メッセージがそのグループの TMS に送信され、そのグループのトランザクションに「アボート済み」のマークが付けられます。グループが指定されないと、メッセージはトランザクショ

ン・コーディネータの TMS に送信され、トランザクションのアボートを要求します。アボート処理を制御するには、トランザクション内のすべてのグループにアボート・メッセージを送信する必要があります。

このコマンドは、トランザクション・コーディネータのサイトが分断されているか、またはクライアントがコミットまたはアボートを呼び出す前に終了してしまった場合に便利です。タイムアウト値が大きいと、トランザクションは、アボートされるまでトランザクション・テーブルに残ります。

トランザクションのコミット

トランザクションをコミットするには、次の手順に従います。

1. 次のコマンドを入力します。

   ```
   committrans (commit) [-yes] [-g groupname] tranindex
   ```

   注記　*groupname* および *tranindex* は必須引数です。

トランザクションがプリコミットされていないか、またはアボート済みのマークが付いている場合、操作は失敗します。トランザクションを完全にコミットするため、このメッセージはすべてのグループに送信しなければなりません。

committrans コマンドの使用上の注意

committrans コマンドの使用には注意が必要です。このコマンドは、次の 2 つの条件に合致する場合にのみ実行できます。

- すべてのグループがコミット・メッセージを受信する前に、トランザクション・コーディネータの TMS がダウンした。
- トランザクション・コーディネータの TMS が、ある期間トランザクションを回復できなくなる。

また、クライアントが tpcommit() でブロックされることもありますが、これはタイムアウトになります。管理コミットを実行する場合には、そのことをクライアントに必ず知らせてください。

トランザクション使用時における障害回復

管理対象のアプリケーションにデータベース・トランザクションが含まれる場合は、ディスク破損の障害後に回復したデータベースにアフター・イメージ・ジャーナル (AIJ) を適用する必要があります。または、この回復アクティビティのタイミングをサイトのデータベース管理者 (DBA) と調整することが必要になる可能性もあります。通常はエラーが発生すると、データベース管理ソフトウェアが自動的にトランザクションのロールバックを行います。ただしデータベース・ファイルを格納するディスクが永久的に破損した場合は、システム管理者または DBA が介入してロールフォワード・オペレーションを行うことが必要になることもあります。

データベースの一部を含むディスクが、水曜日の午後 3 時に破損したと想定します。この例では、シャドウ・ボリューム (ディスク・ミラーリング) が存在しないとします。

1. BEA Tuxedo アプリケーションをシャットダウンします。シャットダウン方法については、『BEA Tuxedo アプリケーションの設定』の 1-1 ページの「アプリケーションの起動とシャットダウン」を参照してください。
2. データベースの最新のフル・バックアップを取得し、ファイルを復元します。たとえば、先週の日曜日午前 12 時 1 分現在のデータベースのフル・バックアップ・バージョンを復元します。
3. 月曜日、火曜日、の順に、インクリメンタル・バックアップ・ファイルを適用します。たとえば、この場合は火曜日の午後 11 時までのデータベースを復元するとします。
4. 火曜日の午後 11 時 15 分から水曜日の午後 2 時 50 分までのトランザクションを含むトランザクション・ジャーナル・ファイル (AIJ) を適用します。
5. データベースを再度オープンします。
6. BEA Tuxedo アプリケーションを再起動します。

データベースのロールフォワード・プロセスの具体的な説明については、リソース・マネージャ (データベース製品) のマニュアルを参照してください。

アプリケーションが正しくシャットダウンされない場合の IPC ツールの使用

IPC 資源とは、メッセージ・キュー、共用メモリ、セマフォなどのオペレーティング・システムの資源のことです。tmshutdown コマンドを使って BEA Tuxedo アプリケーションが正常にシャットダウンされると、IPC 資源はすべてシステムから削除されます。ただし、アプリケーションが正常にシャットダウンされず、システムに IPC 資源が残る場合もあります。このような場合は、アプリケーションを再起動できなくなります。

この問題の解決策として、IPCS コマンドを実行するスクリプトを使用して IPC 資源を削除し、特定のユーザが保有するすべての IPC 資源を解放する方法があります。しかし、この方法では IPC 資源の識別が困難です。たとえば、BEA Tuxedo アプリケーションの資源か、特定の BEA Tuxedo アプリケーションに属する資源か、または BEA Tuxedo システムとは無関係の資源かを識別することができません。誤って IPC 資源を削除するとアプリケーションが破損する可能性があるため、資源の種類を識別できることは重要です。

BEA Tuxedo の IPC ツール (tmipcrm コマンド) を使用すると、実行中のアプリケーションで BEA Tuxedo システムによって割り当てられている IPC 資源 (コア・システムと Workstation コンポーネントのみ) を削除できます。

IPC 資源を削除するコマンドは、TUXDIR/bin に格納されている tmipcrm です。このコマンドは、バイナリ形式のコンフィギュレーション・ファイル (TUXCONFIG) を読み込み、このファイルの情報を使用して掲示板に書き込みます。tmipcrm を使用できるのは、ローカル・サーバ・マシンに対してのみです。BEA Tuxedo のコンフィギュレーションのリモート・マシンにある IPC 資源は削除できません。

このコマンドを実行するには、次のコマンド行を入力します。

```
tmipcrm [-y] [-n] [TUXCONFIG_file]
```

IPC ツールを使用すると、BEA Tuxedo システムで使用されるすべての IPC 資源を一覧表示したり、IPC 資源を削除することができます。

注記　このコマンドは、TUXCONFIG 環境変数を正確に設定するか、またはコマンド行で適切な TUXCONFIG ファイルを指定しないと利用できません。

マルチスレッド化またはマルチコンテキスト化されたアプリケーションのトラブルシューティング

マルチスレッド化またはマルチコンテキスト化されたアプリケーションのデバッグ

シングルスレッドのアプリケーションと比較すると、マルチスレッド化されたアプリケーションのデバッグは困難です。そのため、管理者側で、マルチスレッド化されたアプリケーションの作成基準となる方針を決めておくと便利です。

マルチスレッド化されたアプリケーションでの保護モードの制限

実行中のアプリケーションが保護モードの場合、そのアプリケーションは ATMI 呼び出しが実行されたときのみ共用メモリにアタッチされます。保護モードにしておくと、BEA Tuxedo の共用メモリが不正なアプリケーション・ポインタによって誤って上書きされるのを防ぐことができます。

保護モードで実行中のマルチスレッド・アプリケーションには、アプリケーション・コードを実行するスレッドと、BEA Tuxedo の関数呼び出しにより掲示板の共用メモリにアタッチされるスレッドが混在している場合があります。したがって、ATMI 呼び出しで掲示板にアタッチされたスレッドが少なくとも 1 つあると、保護モードを実行しても、アプリケーション・コードを実行するスレッドを不正なアプリケーション・ポインタから保護できず、BEA Tuxedo の共用メモリが上書きされる場合があります。つまり、マルチスレッド化されたアプリケーションでは、保護モードの効果が制限されています。

この制限を解除する方法はありません。マルチスレッド化されたアプリケーションを実行するときは、シングルスレッドのアプリケーションのときほど、保護モードを信頼できない点に注意してください。