BEA Tuxedo®

BEA Tuxedo アプリケーションの設定

BEA Tuxedo 8.1
2003 年 1 月

目次

4. トランザクションについて

5. トランザクション対応の ATMI アプリケーションのコンフィギュレーション

6. CORBA インターフェイス・リポジトリの管理

10. ワークステーション・クライアントについて

11. ワークステーション・クライアントの設定

12. BEA Tuxedo CORBA リモート・クライアント・アプリケーションの管理

このマニュアルについて

このマニュアルでは、データ依存型の ATMI 環境またはオブジェクト指向の CORBA 環境で、BEA Tuxedo® システムを計画、設計、および設定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「管理タスクと管理ツール」では、BEA Tuxedo の管理者タスクについて説明します。
- 第 2 章「コンフィギュレーション・ファイルについて」では、BEA Tuxedo のコンフィギュレーション・ファイルと、アプリケーションの起動と実行に必要なすべての情報が格納されているリポジトリの概要を示します。また、CORBA 環境の NameManager についても説明します。
- 第 3 章「コンフィギュレーション・ファイルの作成」では、ATMI 環境または CORBA 環境で動作するシングル・マシン用のコンフィギュレーション・ファイルを作成する方法について説明します。
- 第 4 章「トランザクションについて」では、BEA Tuxedo トランザクションの概要を示します。
- 第 5 章「トランザクション対応の ATMI アプリケーションのコンフィギュレーション」では、BEA Tuxedo の ATMI 環境でトランザクションを設定する方法について説明します。
- 第 6 章「CORBA インターフェイス・リポジトリの管理」では、BEA Tuxedo ドメインでインプリメントされる CORBA オブジェクトのインターフェイス・リポジトリを作成する方法について説明します。
- 第 7 章「ネットワークへの ATMI アプリケーションの分散」では、単一の BEA Tuxedo コンフィギュレーション・ファイルを使用して、複数の

マシンにわたる 1 つ以上のサーバ経由でローカル・クライアントまたはリモート・クライアントに Tuxedo ATMI アプリケーションを配布する方法について説明します。

- 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」では、ATMI 環境でマルチ・マシン用のコンフィギュレーション・ファイルを作成する方法について説明します。
- 第 9 章「分散アプリケーションのネットワーク設定」では、BEA Tuxedo システムで分散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設定方法について説明します。
- 第 10 章「ワークステーション・クライアントについて」では、BEA Tuxedo システムの BEA Tuxedo ワークステーション・コンポーネントについて説明します。
- 第 11 章「ワークステーション・クライアントの設定」では、Tuxedo ワークステーション・クライアントをアプリケーションに参加させるための BEA Tuxedo 環境の設定方法について説明します。
- 第 12 章「BEA Tuxedo CORBA リモート・クライアント・アプリケーションの管理」では、標準インターネット ORB 間プロトコル (IIOP) によるリモート CORBA クライアント・アプリケーションから CORBA オブジェクトへの接続を設定する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 アプリケーション実行時の管理』—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 管理タスクに対するコマンド行インターフェイスからのアクセスについて説明しています。
- 『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 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 CORBA トランザクション』—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 環境における CORBA トランザクションの設定方法が記載されています。

BEA Tuxedo ATMI のアーキテクチャ および BEA Tuxedo CORBA 環境の設定と管理の詳細については、BEA Tuxedo オンライン・マニュアルの 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 管理タスクと管理ツール

ここでは、次の内容について説明します。

- 管理者が実行するタスク
- アプリケーションの設計計画を立てる
- アプリケーションの管理用ツール

管理者が実行するタスク

管理者の作業は、次の 2 つのタスクに大きく分類できます。

- セットアップ・タスク—アプリケーションを起動する前にシステム側で準備する必要のあるすべてのタスク
- 実行時の管理—起動後のアプリケーション上で実行するタスク

セットアップ・タスク

管理者は、セットアップ時に、BEA Tuxedo システムの計画、設計、インストール、セキュリティの設定、およびコンフィギュレーションを行う責任があります。次の表は、セットアップ時の必須タスクとオプションのタスクを示します。

セットアップのタスク	**必須**	**オプション**
アプリケーションの設計者、プログラマ、およびビジネス・ユーザから情報を収集する。	X	
ハードウェアとソフトウェアの設定、および BEA Tuxedo システムとアプリケーションのインストールを行う (インストール)。	X	
BEA Tuxedo システムのパラメータを設定する (コンフィギュレーション)。これらのパラメータにより、アプリケーションで各コンポーネントがどのように使用されるかが決定されます。	X	
ドメイン、マシン、グループ、インターフェイス、サービス、およびその他の必須コンポーネントに対応したトランザクションを設定する (コンフィギュレーション)。	X	
アプリケーションとデータを保護するためのセキュリティ方式を選択および実装する。	X	
CORBA 環境でインターネット ORB 間プロトコル (IIOP) のリスナ / ハンドラを設定し、マシンのコンフィギュレーションを変更する。	X	
ルーティング・ツールを使用して分散アプリケーションを設定する。CORBA 環境ではファクトリ・ベース・ルーティングを、ATMI 環境ではデータ依存型ルーティングを使用する。		X
ネットワーク・アプリケーションを設定する。		X

セットアップのタスク	**必須**	**オプション**
ローカル・ドメインおよびリモート・ドメインをコンフィギュレーションする。		X
ワークステーション・クライアントを設定する。環境テーブルとワークステーション・リスナを追加し、マシンのコンフィギュレーションを変更する。		X
アプリケーションのキュー・スペースを作成し、キューに登録されたメッセージがサポートされるようにコンフィギュレーションを変更する。		X

実行時のタスク

BEA Tuxedo システムをインストールし、`TUXCONFIG` ファイルをロードすると、アプリケーションの起動準備は完了です。アプリケーションを起動したら、アプリケーションのアクティビティを監視し、問題が発生していないかどうか、または問題の原因となり得る現象がないかどうかを確認する必要があります。次の表は、実行時の必須タスクとオプションのタスクを示します。

実行時のタスク	**必須**	**オプション**
アプリケーションを起動およびシャットダウンする。	X	
バッファを管理する。	X	
アプリケーションのセキュリティを管理する。	X	
アプリケーションのアクティビティ、問題、および性能を監視する。	X	
ATMI 環境でトランザクションを管理する。		X
CORBA 環境でインターフェイスを管理する。		X

実行時のタスク	必須	オプション
ネットワーク・アプリケーションを管理する。		X
リモート・ワークステーション・クライアントを管理する。		X
イベントをサブスクライブする。		X
キュー機能を使用する。		X
問題が発生した場合に、問題の内容を明らかにし、解決する (トラブルシューティング)。		X
MASTER マシンで問題が発生した場合に、アプリケーションの基本的な役割を MASTER マシンから代替 (BACKUP) マシンに再び割り当てる (移行)。		X
ニーズの変化に応じて、システム・パラメータやサービスの種類を変更する (動的な変更)。		X
新しいマシンやサーバなどの追加コンポーネントの導入に合わせてアプリケーションを調整する (動的な再コンフィギュレーション)。		X

実行時には、問題の原因となり得る現象や、アプリケーション要件の変化にすばやく対応する必要があります。そのため、BEA Tuxedo Administration Console、コマンド行インターフェイス、および AdminAPI の 3 つのツールが用意されています。次の表は、いくつかの処理と、それらの実行方法を示します。

目的	実行方法
性能を最大化するには	サービスのロード・バランシングを行うか、またはサービスに優先順位を設定します。
MASTER マシン上で発生する可能性のある問題を修復するには	指定した BACKUP マシンに置換します。

目的	実行方法
処理およびリソースの使用率に関する要件を変更するには	マシン、サーバ、クライアント、インターフェイス、サービスなどを追加します。

関連項目

- 1-7 ページの「アプリケーションの設計計画を立てる」
- 1-10 ページの「アプリケーションの管理用ツール」

BEA Tuxedo における ATMI 環境と CORBA 環境の違い

BEA Tuxedo CORBA 環境では、BEA Tuxedo の管理機能により、オブジェクト・リクエスト・ブローカ (ORB: Object Request Broker) と TP フレームワークのコンテキスト内で実行するアプリケーションの管理がサポートされます。

BEA Tuxedo CORBA 環境の `UBBCONFIG` コンフィギュレーション・ファイルには、クライアント・アプリケーションとサーバ・アプリケーションのコンフィギュレーションをサポートする以下のセクションがあります。

- `RESOURCES` セクション。アプリケーション全体にわたるデフォルト値を設定し、掲示板のテーブル・サイズを指定します。
- `MACHINES` セクション。プロセッサ固有の値を指定し、テーブルのサイズ指定に適用できます。
- `INTERFACES` セクション。アプリケーションで使用される CORBA インターフェイスに関する情報を指定できます。
- `ROUTING` セクション。Tuxedo CORBA 環境で使用される各種ルーティング基準をサポートします。BEA Tuxedo の ATMI データ依存型ルーティ

ングのパラメータを指定する既存の ROUTING セクションをそのまま使用することもできます。

- BEA Tuxedo の ATMI 環境では、クライアント・アプリケーションからサーバ・アプリケーションへの接続用にワークステーション・ハンドラとリスナを設定します。管理者側から見ると、このタスクは BEA Tuxedo の CORBA 環境 でのタスクと似ています。

  ただし、BEA Tuxedo の CORBA 環境 では、異なる通信プロトコルを使用して、リモートおよび外部クライアントから BEA Tuxedo サーバ・アプリケーションに接続します。このプロトコルが、標準インターネット ORB 間プロトコル (IIOP) です。BEA Tuxedo のワークステーション・ハンドラ (WSH) プロセスとワークステーション・リスナ (WSL) プロセスの代わりに、CORBA 環境では、そのゲートウェイ・プロセスである IIOP ハンドラ (ISH) と IIOP リスナ (ISL) を呼び出します。この場合、構文がわずかに異なり、各アプリケーションの UBBCONFIG コンフィギュレーション・ファイルにある SERVERS セクションで、WSL の代わりに ISL を使用します。

全体的に、BEA Tuxedo の CORBA 環境と ATMI 環境の管理タスクはよく似ていますが、以下のような相違点があります。

- どちらの環境でも、ルーティング基準を使用して特定のサーバ・グループに処理を分散します。BEA Tuxedo の CORBA 環境のルーティング・メカニズムは、ファクトリ・ベース・ルーティングと呼ばれます。このメカニズムは、BEA Tuxedo ATMI のデータ依存型ルーティング・メカニズムとは基本的に異なります。

  BEA Tuxedo の ATMI 環境では、サービス呼び出しに使用される FML フィールドをチェックして、データ依存型のルーティング基準を確認することができます。これに対し BEA Tuxedo の CORBA 環境 の場合、システム設計者は、CORBA インターフェイスのルーティング基準に個別にアクセスする必要があります。BEA Tuxedo の CORBA 環境 には、ルーティングに使用するサービス要求メッセージ・データや関連バッファ情報が用意されていません。これは、CORBA 環境のルーティングが、ターゲットの CORBA オブジェクトのメソッド呼び出しではなく、ファクトリで実行されるためです。

- 実行時に CORBA インターフェイスを動的に宣言することはできません。ただし、CORBA インターフェイスを一時停止して、再びアクティブにすることは可能です。
- CORBA インターフェイスを ACL で直接制御することはできません。管理者レベルでサーバントを制御することはできません。UBBCONFIG コンフィギュレーション・ファイルの MANDATORY_ACL パラメータから SECURITY パラメータまでは無視されます。
- LDAP の単一のセキュリティ管理機能は CORBA インターフェイスではサポートされていません。

注記　管理情報ベース (MIB) で一連のクラスを定義し、そのクラスによってアプリケーションの基本的な側面を設定および管理することができます。MIB クラスによって、BEA Tuxedo CORBA および ATMI 環境への管理用プログラミング・インターフェイスが提供されます。

アプリケーションの設計計画を立てる

管理者は、顧客のビジネス上の要件を把握し、ソフトウェアがどのように使用されるかを知っておく必要があります。これらのニーズを理解しておけば、管理者は、システム設計者やアプリケーション開発者と協力して、要件を満たすアプリケーションのコンフィギュレーションを実現できます。

アプリケーションの設計を計画する前に、以下の事項を確認してください。

1. 使用するマシンの台数 ____________________
2. クライアント・アプリケーションは、サーバ・アプリケーション側から見てリモートのマシンに存在するかどうか ________________________
3. ATMI 環境のアプリケーションで提供するサービス
   ____
   ____

4. CORBA 環境のクライアントまたはサーバ・アプリケーションで使用するインターフェイス

   ____
   ____

5. アプリケーションで使用されるリソース・マネージャ (データベース) の種類と場所

   ____
   ____

6. リソース・マネージャに必要な OPENINFO 文字列

   ____
   ____

7. RDBMS に必要なセットアップ情報

   ____
   ____
   ____

8. トランザクションの分散を行うかどうか ________________

9. アプリケーションでグローバル・トランザクションを使用するかどうか ________________

10. 使用するバッファ・タイプの種類

    ____

11. データを複数のマシンに分散するかどうか

    ____

12. サービスのエクスポートやインポートを行うときの、エクスポート先またはインポート元となる外部ドメイン

    ____
    ____
    ____
    __

13. アプリケーションでファクトリ・ベース・ルーティングまたはデータ依存型ルーティングを使用するかどうか

    ____

14. CORBA インターフェイスまたは ATMI サービスの名前

15. インターフェイスまたはサービスの優先順位

16. 信頼性の要件。冗長なリスナ・ポートおよびハンドラ・ポートは必要かどうか。サーバ・アプリケーションの複製は必要かどうか

17. CORBA 環境のドメインでインターフェイス・リポジトリ (IR) データベースが必要かどうか。必要な場合、ドメインに IR の複製を作成した方がよいかどうか。また、いくつの IR サーバ・アプリケーションを定義する必要があるか

18. 会話型サービスがあるかどうか。ある場合、どのリソース・マネージャにアクセスするか。また、どのバッファ型を使用するか

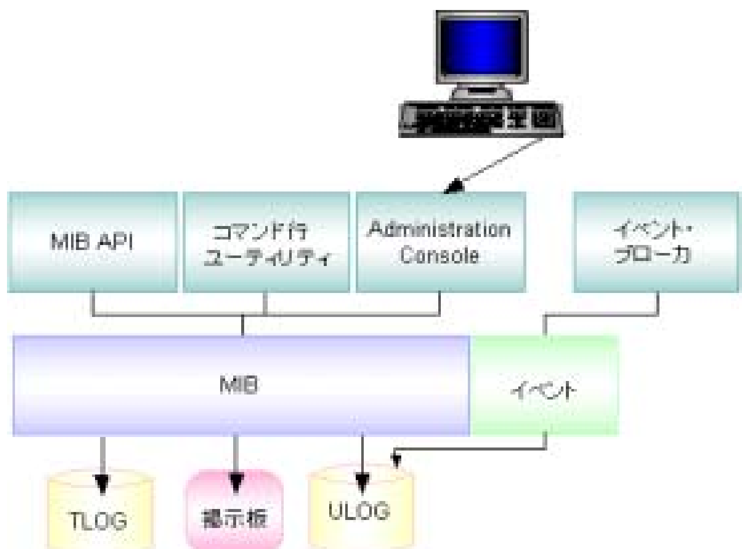
関連項目

- 1-10 ページの「アプリケーションの管理用ツール」

アプリケーションの管理用ツール

BEA Tuxedo システムには、BEA Tuxedo ATMI または CORBA 環境のいずれかで同じ種類の管理タスクを実行する方法がいくつか用意されています。グラフィカル・ユーザ・インターフェイスに慣れている場合も、シェル・プロンプトでのコマンド入力に慣れている場合も、BEA Tuxedo アプリケーションの管理作業がしやすい方法が用意されています。次の図は、コンフィギュレーション・ファイルへの書き込みや、実行時の BEA Tuxedo アプリケーション管理に使用できるツールを示します。

図 1-1 管理ツール



- BEA Tuxedo Administration Console―アプリケーションを監視し、アプリケーション操作を動的にコンフィギュレーションするための Web ベースのツールです。
- BEA Tuxedo MIB アプリケーション・プログラミング・インターフェイス―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用のインターフェイスです。

- コマンド行ユーティリティーアプリケーションの管理 (`tmadmin(1)`)、起動 (`tmboot(1)`)、コンフィギュレーション (`tmconfig`、`wtmconfig(1)`)、および終了 (`tmshutdown(1)`) を実行するためのコマンド群です。詳細については、『BEA Tuxedo コマンド・リファレンス』を参照してください。

ツールの種類	使用方法
BEA Tuxedo Administration Console	グラフィカル・ユーザ・インターフェイス (GUI) を使用して、`TUXCONFIG` ファイルを作成および編集します。GUI から直接ヘルプにアクセスすると、GUI の詳しい説明を参照できます。
BEA Tuxedo MIB アプリケーション・プログラミング・インターフェイス	管理者用に `TUXCONFIG` ファイルを変更するプログラムを作成します。
コマンド行インターフェイス	1. テキスト・エディタで、`UBBCONFIG` ファイル (テキスト形式の `TUXCONFIG`) を作成および編集します。 2. `tmloadcf` を実行して、`UBBCONFIG` ファイルをバイナリ形式の `TUXCONFIG` ファイルに変換します。 `tmloadcf` コマンド・オプションの詳細については、『BEA Tuxedo コマンド・リファレンス』の `tmloadcf(1)` を参照してください。

関連項目

- 『BEA Tuxedo システム入門』の「BEA Tuxedo Administration Console を使用した管理操作」
- 『BEA Tuxedo システム入門』の「MIB を使用した操作の管理」
- 『BEA Tuxedo システム入門』の「コマンド行ユーティリティを使用した操作の管理」
- 1-1 ページの「管理者が実行するタスク」

- 『BEA Tuxedo システム入門』の「BEA Tuxedo ATMI のアーキテクチャ」
- 『BEA Tuxedo CORBA アプリケーション入門』の「BEA Tuxedo CORBA プログラミング環境」
- 『BEA Tuxedo のファイル形式とデータ記述方法』の `ACL_MIB(5)`、`APPQ_MIB(5)`、`EVENT_MIB(5)` に関する追加情報、`MIB(5)`、`TM_MIB(5)`、`WS_MIB(5)`、および `UBBCONFIG(5)`
- 『BEA Tuxedo コマンド・リファレンス』の `tmshutdown(1)`

2 コンフィギュレーション・ファイルについて

ここでは、次の内容について説明します。

- コンフィギュレーション・ファイルとは
- コンフィギュレーション・ファイルの内容

コンフィギュレーション・ファイルとは

BEA Tuxedo の各アプリケーションのコンフィギュレーションは、管理者の主要なタスクです。管理者は、コンフィギュレーション・ファイルにパラメータ値を設定して、アプリケーションを記述します。これらのパラメータ値により、実行可能なアプリケーションが作成されます。コンフィギュレーション・ファイルとは、アプリケーションの起動と実行に必要なすべての情報 (アプリケーションのリソース、マシン・グループ、サーバ、および利用可能なサービスなどの仕様) が格納されたファイルです。

テキスト形式とバイナリ形式のコンフィギュレーション・ファイル

コンフィギュレーション・ファイルには、次の 2 つの形式があります。

- UBBCONFIG ファイルは、テキスト形式のコンフィギュレーション・ファイルであり、任意のエディタで作成したり、編集できます。BEA Tuxedo のサンプル・アプリケーションには、コンフィギュレーション・ファイルのサンプルが用意されていますが、UBBCONFIG そのものは用意されていません。UBBCONFIG ファイルは、アプリケーションごとに作成してください。コンフィギュレーション・ファイルのエントリで使用する構文については、『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5) を参照してください。

  注記　BEA Tuxedo ソフトウェアには、bankapp および simpapp アプリケーションの一部として 3 つの UBBCONFIG ファイルのサンプル (ubbshm、ubbmp、ubbsimple) が用意されています (『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』を参照してください)。

- TUXCONFIG ファイルは、バイナリ形式のコンフィギュレーション・ファイルであり、tmloadcf(1) コマンドにより、テキスト形式のファイルから作成されます。tmloadcf を実行する前に、TUXCONFIG をロードするデバイスまたはシステム・ファイルの絶対パス名に環境変数 TUXCONFIG を設定しておく必要があります。TUXCONFIG のパラメータの多くは、必要に応じて、アプリケーションの実行中に tmconfig、wtmconfig(1)、または MIB を使って変更できます。

コンフィギュレーション・ファイルの内容

次の表は、コンフィギュレーション・ファイルの 9 つのセクションと、各セクションの目的を説明します。

セクション	**必須 / オプション**	**説明**
RESOURCES	必須	すべてのシステム・パラメータを定義します。
MACHINES	必須	アプリケーションで使用されるすべてのマシンを指定します。
GROUPS	必須	アプリケーション内のすべてのグループ、グループ名、および グループ ID を定義します。
SERVERS	オプション	システム内で起動するサーバの初期条件を指定します。
SERVICES	オプション	アプリケーションで使用されるサービスに関する情報を提供します。
INTERFACES	オプション	CORBA 環境のアプリケーションで使用されるインターフェイスに対し、アプリケーション全体にわたるデフォルト・パラメータに関する情報を指定します。
NETWORK	オプション	LAN 環境のネットワーク設定を記述します。
NETGROUPS	オプション	LAN 環境で使用可能なネットワーク・グループを記述します。
ROUTING	オプション	FML バッファおよび VIEW を使用するサービス要求のデータ依存型ルーティングに関する情報を提供します。

コンフィギュレーション・ファイルには、少なくとも 9 つのパラメータが必要です。パラメータは 80 種類あります。最初のセクション以外のすべてのセクションには、複数のエントリを指定でき、各エントリには、独自のパラメータを指定できます。RESOURCES セクション以外のすべてのセクションでは、デフォルト値を使用して、複数のエントリに含まれるパラメータを指定できます。

コマンド行インターフェイスまたは BEA Tuxedo Administration Console を使用して、バイナリ形式のコンフィギュレーション・ファイル (TUXCONFIG) を作成することができます。まず、そのファイルで定義するコンフィギュレーションの種類を決定します。

- 1 台のマシンで構成するアプリケーション—1 つ以上のローカル・クライアントまたはリモート・クライアントが、同じマシン上にある 1 つ以上のサーバと通信します。
- 複数のマシンで構成 (分散) するアプリケーション—1 つ以上のローカル・クライアントまたはリモート・クライアントが、複数のマシン上に分散する 1 つ以上のサーバと通信します。
- 複数のドメインにまたがるアプリケーション—複数のアプリケーションが、BEA Tuxedo の拡張機能である Domains 機能を利用して相互に通信します。このコンフィギュレーションに含まれる各アプリケーションをドメインと呼びます。

CORBA 環境の管理上の要件とパフォーマンス

この節では、BEA Tuxedo システムで CORBA 環境を適切に管理する方法について説明します。

NameManager の設定

CORBA 環境を適切に管理するには、以下の要件を満たす必要があります。

- NameManager は、BEA Tuxedo のイベント・ブローカを使用して相互にアクティビティを調整し合い、管理者やオペレータの介入が不要でなければなりません。イベント・ブローカは、サーバが NameManager サービスを提供する前に起動しておく必要があります。アプリケーションでイベント・ブローカが設定されていないため、NameManager サービスの起動時にイベント・ブローカが実行されない場合、NameManager は起動処理をアボートし、ユーザ・ログにエラー・メッセージが記録されます。
- 少なくとも 2 つのサーバが、アプリケーションの一部として NameManager サービスを実行するように設定されていなければなりません。これにより、常に「名前 - IOR」マッピングのコピーが使用可能になります。複数のマシン上に複数のサーバが存在し、1 台のマシンがクラッシュした場合、そのマシンとアプリケーションを再起動する際に、新しい NameManager がほかの NameManager からマッピングを取得します。アプリケーションが 1 台のマシン上にのみ存在し、そのマシンがクラッシュした場合は、アプリケーションを再起動しなければならないので、アプリケーションの起動処理の一部として NameManager が再起動されます。NameManager サービスの起動時に、アプリケーションに 2 つの NameManager が設定されていない場合、NameManager は起動処理をアボートし、ユーザ・ログにエラー・メッセージが記録されます。
- NameManager はマスタとスレーブのどちらにも指定できますが、デフォルトはスレーブです。アプリケーションでマスタの NameManager サー

バが設定されていないため、スレーブ NameManager サーバの起動時にマスタ NameManager サーバが実行されていない場合、スレーブ・サーバの起動時にサーバ自体が終了し、ユーザ・ログにエラー・メッセージが記録されます。

- FactoryFinder サービスの起動時に、アプリケーションで NameManager サービスが設定されていない場合、FactoryFinder は起動処理をアボートし、ユーザ・ログにエラー・メッセージが記録されます。FactoryFinder は、アプリケーションから “find” 要求を受信した場合のみ NameManager と通信するので、FactoryFinder サービスの前に NameManager サービスを起動する必要はありません。これに対し、NameManager は起動時に相互通信を試みます。FactoryFinder は、リモート・ドメインにあるファクトリの検索要求を受け取らない限り、相互に通信することはありません。
- BEA Tuxedo のイベント・ブローカ、NameManager、および FactoryFinder サービスは、アプリケーション固有のサーバを起動する前に起動しておく必要があります。ただし、アプリケーションに複数のイベント・ブローカが設定されている場合、すべてのセカンダリ・イベント・ブローカは、アプリケーション・サーバがすべて起動された後に起動しなければなりません。アプリケーション・サーバにはこの起動順序を設定するシステム・プロトコルがないので、すべてのセカンダリ・イベント・ブローカをアプリケーション・サーバの後に配置する必要があります。
- アプリケーション・サーバからファクトリ・オブジェクトへのオブジェクト・リファレンスを登録するには、マスタ NameManager を起動し、実行しておく必要があります。スレーブ NameManager を実行しておくだけでは不十分です。

信頼性に関する要件

この節では、CORBA 環境の信頼性を向上させる方法について説明します。

ファクトリ・エントリの管理

アプリケーション・サーバがダウンすると、NameManager に登録されたファクトリを削除できなくなります。また、FactoryFinder が、アクティブではなくなったファクトリのオブジェクト・リファレンスを返す場合もあります。これは、非アクティブになったファクトリを含むサーバが使用不能になった、ファクトリを NameManager から登録解除できない、または、ファクトリ用のインターフェイスを提供するサーバがほかに存在しないことが原因で発生します。

通常、直後にアプリケーション・ファクトリを再起動してファクトリを提供できますが、ファクトリ・エントリがいつまでも残ったままにならないように、アプリケーション・サーバがダウンした場合は NameManager に通知されます。この通知を受信すると、NameManager は現在アクティブなサーバではサポートされていないファクトリ・エントリを削除することができます。

複数の NameManager と FactoryFinder の設定

少なくとも、マスタとスレーブの 2 つの NameManager をアプリケーションで (可能であれば異なるマシンに) 設定し、FactoryFinder で照会機能を実行できるようにします。アプリケーションには、複数の FactoryFinder を設定することも可能です。

マスタ NameManager の指定

マスタ NameManager は、`UBBCONFIG` ファイルで指定する必要があります。マスタ NameManager には、登録に関するあらゆるアクティビティが送信されます。内容が更新されると、マスタ NameManager からスレーブ NameManager に通知されます。マスタ NameManager がダウンすると、マスタが再起動されるまで、ファクトリの登録と削除はできません。

パフォーマンスに関するヒント

FactoryFinder と NameManager を別々のマシン上で実行するより、同じマシン上の別のサーバで実行した方が、パフォーマンスを最適化することができます。これにより、マシン間の通信が不要になるので、より迅速な応答が可能になります。

関連項目

- 3-1 ページの「コンフィギュレーション・ファイルの作成」
- 3-4 ページの「複数のマシンで構成 (分散) する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」
- 『BEA Tuxedo システム入門』の「BEA Tuxedo Domains (マルチ・ドメイン) サーバ」
- 『BEA Tuxedo アプリケーション実行時の管理』の 1-4 ページの「TUXCONFIG ファイルの作成」
- 分散型の BEA Tuxedo CORBA アプリケーションについては、『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。

3 コンフィギュレーション・ファイルの作成

ここでは、次の内容について説明します。

- コンフィギュレーション・ファイルの作成方法
- 1 台のマシンで構成する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
- 複数のマシンで構成 (分散) する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
- 複数のドメインにまたがる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
- コンフィギュレーション・ファイルの RESOURCES セクションの作成方法
- コンフィギュレーション・ファイルの MACHINES セクションの作成方法
- コンフィギュレーション・ファイルの GROUPS セクションの作成方法
- コンフィギュレーション・ファイルの NETWORK セクションの作成方法
- コンフィギュレーション・ファイルの NETGROUPS セクションの作成方法

- コンフィギュレーション・ファイルの SERVERS セクションの作成方法
- コンフィギュレーション・ファイルの SERVICES セクションの作成方法
- コンフィギュレーション・ファイルの INTERFACES セクションの作成方法
- コンフィギュレーション・ファイルの ROUTING セクションの作成方法
- スレッドを利用する BEA Tuxedo システムの設定方法
- コンフィギュレーション・ファイルのコンパイル方法

コンフィギュレーション・ファイルの作成方法

コンフィギュレーション・ファイルに指定する内容は、作成するアプリケーションによって異なります。以下は、さまざまなアプリケーションの作成手順です。

- 1 台のマシンで構成する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
- 複数のマシンで構成 (分散) する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
- 複数のドメインにまたがる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
- スレッドを利用する BEA Tuxedo システムの設定方法

関連項目

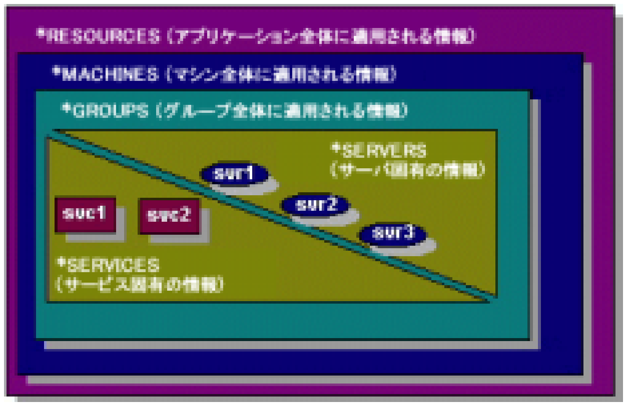
- 2-1 ページの「コンフィギュレーション・ファイルについて」
- 『BEA Tuxedo のファイル形式とデータ記述方法』の `UBBCONFIG(5)`

1 台のマシンで構成する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

1 台のマシンで構成するコンフィギュレーションでは、コンフィギュレーション・ファイルに、次のセクションを作成する必要があります。各作業をクリックすると、その作業を行う手順が表示されます。

1. コンフィギュレーション・ファイルの RESOURCES セクションを作成します。
2. コンフィギュレーション・ファイルの MACHINES セクションを作成します。
3. コンフィギュレーション・ファイルの GROUPS セクションを作成します。
4. コンフィギュレーション・ファイルの SERVERS セクションを作成します。
5. コンフィギュレーション・ファイルの SERVICES セクションを作成します。
6. コンフィギュレーション・ファイルの INTERFACES セクションを作成します (CORBA のみ)。
7. コンフィギュレーション・ファイルの ROUTING セクションを作成します。

次の図中をクリックすると、クリックしたセクションの作成方法が表示されます。



複数のマシンで構成 (分散) する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

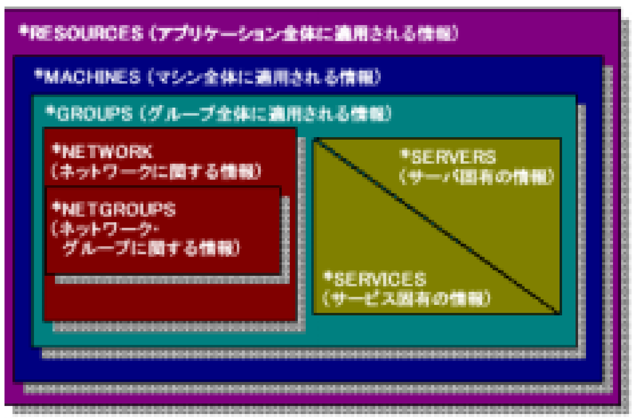
分散型の ATMI アプリケーションでは、コンフィギュレーション・ファイルに次のセクションを作成する必要があります。各作業をクリックすると、その作業を行う手順が表示されます。

1. コンフィギュレーション・ファイルの RESOURCES セクションを作成します。
2. コンフィギュレーション・ファイルの MACHINES セクションを作成します。
3. コンフィギュレーション・ファイルの GROUPS セクションを作成します。
4. コンフィギュレーション・ファイルの NETWORK セクションを作成します。
5. コンフィギュレーション・ファイルの NETGROUPS セクションを作成します。

6. コンフィギュレーション・ファイルの `SERVERS` セクションを作成します。
7. コンフィギュレーション・ファイルの `SERVICES` セクションを作成します。
8. コンフィギュレーション・ファイルの `ROUTING` セクションを作成します (オプション)。

注記　BEA Tuxedo システムで分散型 CORBA アプリケーションのコンフィギュレーション・ファイルを作成する方法については、『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。

次の図中をクリックすると、クリックしたセクションの作成方法が表示されます。



複数のドメインにまたがる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

複数のドメインにまたがるコンフィギュレーションでは、ドメインごとに次の 2 つのコンフィギュレーション・ファイルを作成する必要があります。

- UBBCONFIG—アプリケーションのコンフィギュレーション・ファイル。
- DMCONFIG—ドメインのコンフィギュレーション・ファイル。

たとえば、ローカル・ドメイン (lapp) とリモート・ドメイン (rapp) で構成されるアプリケーションの場合は、以下の作業が必要です。

各作業をクリックすると、その作業を行う手順が表示されます。

図 3-1 複数のドメインにまたがるサンプル・アプリケーションのコンフィギュレーション

次の図は、2 つのドメインにまたがるアプリケーションの `UBBCONFIG` ファイルおよび `DMCONFIG` ファイルで設定するセクションを示しています。片方はローカル・ドメインであり、もう一方はリモート・ドメインです。

次の図中をクリックすると、クリックしたコンフィギュレーション・ファイルのセクションの作成方法が表示されます。

図 3-2 複数のドメインにまたがるアプリケーションのコンフィギュレーション

関連項目

- 『BEA Tuxedo Domains コンポーネント』の 1-1 ページの「Domains について」
- 『BEA Tuxedo Domains コンポーネント』の「ATMI Domains の計画とコンフィギュレーション」
- 『BEA Tuxedo のファイル形式とデータ記述方法』の DMCONFIG(5)

コンフィギュレーション・ファイルの RESOURCES セクションの作成方法

コンフィギュレーション・ファイルの先頭には、必ず RESOURCES セクションを指定します。このセクションで定義されるパラメータは、アプリケーション全体を制御し、システムのデフォルト値になります。RESOURCES セクションのパラメータ値は、MACHINES セクションのマシンごとの値を変更することによって上書きできます。

次の表では、RESOURCES セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

RESOURCES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
IPC (プロセス間通信) 資源の一意のアドレス	IPCKEY (必須)	共用メモリのアドレス
セキュリティ・アクセス	UID、GID、および PERM (オプション)	セキュリティ・アクセス
掲示板に同時に接続できるプロセスの最大数	MAXACCESSERS (オプション)	IPC 資源の上限値

RESOURCES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
掲示板のサーバ・テーブル・エントリの最大数	MAXSERVERS (オプション)	IPC 資源の上限値
掲示板のサービス・テーブル・エントリの最大数	MAXSERVICES (オプション)	IPC 資源の上限値
CORBA インターフェイスの最大数	MAXINTERFACES (オプション)	IPC 資源の上限値
CORBA オブジェクトの最大数	MAXOBJECTS (オプション)	IPC 資源の上限値
起動、シャットダウン、およびその他の管理タスクが実行される DBBL (Distinguished Bulletin Board Liaison) の場所	MASTER (必須)	マスタ・プロセッサ
掲示板のアーキテクチャ	MODEL (SHM または MP)、および OPTIONS (LAN または MIGRATE)(必須)	アプリケーション・タイプ
セキュリティのレベル	SECURITY、AUTHSVC (オプション)	セキュリティのレベル
ID 確認のためのプロセスのプリンシパル名、プリンシパル・ユーザの秘密鍵の場所、およびパスワードを格納する環境変数	SEC_PRINCIPAL_NAME、SEC_PRINCIPAL_LOCATION、および SEC_PRINCIPAL_PASSVAR	セキュリティ属性
クライアントが任意通知型メッセージを検出する場合のデフォルトの検出方法	NOTIFY、USIGNAL (オプション)	任意通知型メッセージ
共用メモリの保護	SYSTEM_ACCESS (オプション)	共用メモリの保護
サーバのロード・バランシングの有効化	LDBAL (オプション)	ロード・バランシング
バッファのタイプとサブタイプの最大数	MAXBUFTYPE、MAXBUFSTYPES (オプション)	バッファのタイプとサブタイプ

RESOURCES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
1 台のマシンで受け付ける会話の最大数	MAXCONV (オプション)	会話数の上限値
ネットワーク・グループの最大数	MAXNETGROUPS (オプション)	ネットワーク・グループ
正常性チェックの間隔とブロッキング呼び出しの許容時間	SCANUNIT、SANITYSCAN、BLOCKTIME (オプション)	正常性チェックの間隔とブロッキング・タイムアウト

RESOURCES セクションの例

以下は、コンフィギュレーション・ファイルの RESOURCES セクションの例です。

```
*RESOURCES
IPCKEY         39211
UID            0
GID            1
PERM           0660
MAXACCESSERS   75
MAXSERVERS     40
MAXSERVICES    55
MASTER         SITE1, SITE2
MODEL          MP
OPTIONS        LAN, MIGRATE
SECURITY       APP_PW
AUTHSVC        "AUTHSVC"
NOTIFY         DIPIN
SYSTEM_ACCESS  PROTECTED, NO_OVERRIDE
LDBAL           Y
```

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5)
- 3-34 ページの「コンフィギュレーション・ファイルの MACHINES セクションの作成方法」

アプリケーション・タイプを定義する

BEA Tuxedo のアプリケーションのアーキテクチャについて、以下の事項を決定する必要があります。

- 単一プロセッサとグローバル共用メモリを備えたマルチプロセッサのどちらでアプリケーションを実行するか。

- アプリケーションをネットワークに接続するかどうか。
- サーバの移行をサポートするかどうか。

アプリケーション・タイプを定義するには、MODEL パラメータおよび OPTIONS パラメータを使用します。

MODEL パラメータは、単一のプロセッサ上でアプリケーションを実行するかどうかを指定します。ユニプロセッサ、およびグローバル共用メモリを備えたマルチプロセッサの場合、このパラメータには SHM が設定されます。共用メモリを備えていないマルチプロセッサ、およびネットワーク接続されたアプリケーションの場合は、MODEL に MP が設定されます。これは、必須パラメータです。

OPTIONS パラメータは、アプリケーションのコンフィギュレーションに関するオプションをカンマ区切りの形式で指定したリストです。指定できるオプションは、LAN (ネットワーク構成であることを示す) および MIGRATE (アプリケーション・サーバの移行が可能であることを示す) の 2 つです。

MODEL および OPTIONS パラメータの特性

パラメータ	説明
MODEL	これは必須パラメータです。SHM は、グローバル共用メモリを備えた単一のマシンを示します。MP は、グローバル共用メモリを備えていない複数のマシン、またはネットワーク接続されたアプリケーションを示します。
OPTIONS	アプリケーションのコンフィギュレーションに関するオプションを、カンマ区切りの形式で指定したリストです。LAN は、ローカル・エリア・ネットワークを示します。MIGRATE は、サーバの移行を有効にします。 RESOURCES セクションの例では、MODEL が MP が設定され、OPTIONS が LAN および MIGRATE に設定されています。

設定例

以下は、コンフィギュレーション・ファイルの RESOURCES セクションの設定例です。

```
*RESOURCES
     MODEL   MP
     OPTIONS LAN, MIGRATE
```

バッファのタイプとサブタイプの数を制御する

アプリケーションで受け付けられるバッファのタイプおよびサブタイプの数は、MAXBUFTYPE パラメータおよび MAXBUFSTYPE パラメータで制御できます。ユーザ定義のバッファ・タイプが多数指定されていない限り、MAXBUFTYPE は省略できます。何種類もの VIEW タイプを使用する予定がある場合は、MAXBUFSTYPE に現在設定されているデフォルト値を増やしておきます。

MAXBUFTYPE および MAXBUFSTYPE パラメータの特性

パラメータ	説明
MAXBUFTYPE	システムで受け付けるバッファ・タイプの最大数。ユーザ定義のバッファ・タイプを 8 つ以上作成する場合にのみ使用します。MAXBUFTYPE には、0 より大きく 32,768 未満の値を指定します。指定しない場合は、デフォルト値の 16 が設定されます。 例 :MAXBUFTYPE 20

パラメータ	説明
MAXBUFSTYPE	システムで受け付けるバッファのサブタイプの最大数。MAXBUFSTYPE には、0 より大きく 32,768 未満の値を指定します。指定しない場合は、デフォルト値の 32 が設定されます。 例 :MAXBUFSTYPE　40

設定例

この例では、バッファのタイプの最大数として 20、サブタイプの最大数として 40 が指定されています。

```
*RESOURCES
       MAXBUFTYPE   20
       MAXBUFSTYPE  40
```

会話の数を制御する

MAXCONV パラメータを使用すると、マシン上で同時に実行できる会話数を指定できます。MAXCONV には、0 より大きく 32,768 未満の値を指定します。

MAXCONV パラメータの特性

MAXCONV パラメータには、以下の特性があります。

- 各マシンで同時に実行できる会話の最大数を定義します。
- SERVERS セクションにリストされた会話型サーバを備えるアプリケーションのデフォルト値は 10 です。これ以外の場合、デフォルト値は 1 です。

- MACHINES セクションで、該当するマシンの値を変更すると、このパラメータを上書きできます。

設定例

この例では、各マシンで同時に実行できる会話の最大数が 15 に設定されています。

```
*RESOURCES
       MAXCONV   15
```

IPC 資源の上限値を定義する

ほとんどの IPC および共用メモリの掲示板テーブルは、高速処理用に静的に割り当てられているため、正しくチューニングすることが重要です。値の設定が大きすぎると、メモリと IPC 資源が無駄に消費されます。逆に、値の設定が小さすぎると、IPC 資源の上限を超えた時点でプロセスが異常終了します。`tmloadcf -c` コマンドを使用すると、アプリケーションに必要な IPC 資源の上限を確認できます。『BEA Tuxedo コマンド・リファレンス』の `tmloadcf(1)` を参照してください。

MAXACCESSERS、MAXSERVERS、MAXSERVICES、MAXINTERFACES、および MAXOBJECTS は、IPC 資源のサイズを決める、調整可能なパラメータです。アプリケーションに割り当てられる共用メモリの容量は、MAXGTT および MAXCONV パラメータによって制御されます。

MAXACCESSERS、MAXSERVERS、MAXSERVICES、MAXINTERFACES、および MAXOBJECTS パラメータの特性

パラメータ	説明
MAXACCESSERS	BEA Tuxedo アプリケーションの特定のサイトで、掲示板に同時に接続できるプロセスの最大数。この数には、クライアント、システム提供のサーバ、およびアプリケーション・サーバがすべて含まれますが、BBL や tmadmin() など、掲示板にアクセス・スロットが予約されている管理プロセスは含まれません。 MAXACCESSERS には、0 より大きく 32,768 未満の値を指定します。指定しない場合は、デフォルト値の 50 が設定されます。MACHINES セクションでは、マシンごとに MAXACCESSERS を上書きできます。
MAXSERVERS	アプリケーションで使用できるサーバ・プロセスの最大数。この数には、システム提供されるサーバおよびアプリケーション・サーバがすべて含まれます。 MAXSERVERS には、0 より大きく 8.192 未満の値を指定します。指定しない場合は、デフォルト値の 50 が設定されます。
MAXSERVICES	アプリケーションで宣言できる BEA Tuxedo サービスの最大数。MAXSERVICES には、0 より大きく 32,768 未満の値を指定します。指定しない場合は、デフォルト値の 100 が設定されます。 注記　CORBA 環境では、各 CORBA インターフェイスが BEA Tuxedo サービスにマップされます。生成されるサービスの数を考慮して設定してください。

パラメータ	説明
MAXINTERFACES	CORBA 環境のアプリケーションで宣言できる CORBA インターフェイスの最大数。MAXINTERFACES には、0 より大きく 32,766 未満の値を指定します。指定しない場合は、デフォルト値の 100 が設定されます。 注記　インターフェイスのすべてのインスタンスは、掲示板のインターフェイス・テーブルで同じスロットを占有し、再利用します。たとえば、サーバ SVR1 がインターフェイス IF1 と IF2 を、サーバ SVR2 がインターフェイス IF2 と IF3 を、サーバ SVR3 がインターフェイス IF3 と IF4 を宣言した場合、MAXINTERFACES を計算すると、インターフェイス数は 6 ではなく 4 になります。
MAXOBJECTS	CORBA 環境のアプリケーションで有効にできる CORBA オブジェクトの最大数。MAXOBJECTS には、0 より大きく 32,766 未満の値を指定します。指定しない場合は、デフォルト値の 100 が設定されます。

注記　システム提供されるサーバには、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS、TMS_QM、GWTDOMAIN、および WSL があります。

MAXACCESSERS の値を増やすと、サイトごと、および、クライアントまたはサーバ・プロセスごとに、別のセマフォが必要になります (アクセサについては注記を参照)。MAXACCESSERS の値を増やすことによって追加されるセマフォのほか、システム・プロセスでもわずかなセマフォのオーバーヘッドが発生します。一方、MAXSERVERS および MAXSERVICES の値を増やしても、各サーバ、サービス、またはクライアント・エントリに用意されている程度の少量の共用メモリしか消費されません。MAXSERVERS および MAXSERVICES パラメータは、アプリケーションの拡張に対応するために用意されています。これらのパラメータより、MAXACCESSERS を詳細に調べる方が重要です。

注記　システムでは、掲示板へのアクセス・スロットごとに 1 つのセマフォが割り当てられます。セマフォは、複数のプロセスが、掲示板の共用メモリに同時にアクセスしないようにするラッチ回路です。

BEA Tuxedo のリリース 7.1 より前の MAXACCESSERS および MAXSERVERS パラメータは、ユーザ・ライセンス数をチェックするしくみと関連付けられていました。つまり、アプリケーションで実行中の 1 台以上のマシンの MAXACCESSERS の数と、特定のマシンの MAXACCESSERS の数の合計が、MAXSERVERS の数とユーザ・ライセンス数の合計より大きい場合、マシンを起動することはできませんでした。したがって、アプリケーションの MAXACCESSERS パラメータには、MAXSERVERS の数とユーザ・ライセンス数の合計か、またはそれより小さい値を指定しなければなりませんでした。

BEA Tuxedo のリリース 7.1 以降では、アプリケーションに設定されているユーザ・ライセンスの数と、現在使用されているユーザ・ライセンスの数に基づいて、ライセンスのチェックが行われます。すべてのユーザ・ライセンスが使用中になると、アプリケーションに新しいクライアントが参加することはできなくなります。

設定例

この例では、最大 75 のプロセス (クライアントおよびサーバ) が同時にシステムにアクセスできます。掲示板では、40 サーバが 55 のサービスを宣言できます。

```
*RESOURCES
      MAXACCESSERS   75
      MAXSERVERS    40
      MAXSERVICES   55
```

ロード・バランシングを有効にする

BEA Tuxedo では、アプリケーション全体に対し、ロード・バランシングのアルゴリズムを使用するかどうかを制御できます。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。

ロード・バランシングを使用するかどうかを指定するには、LDBAL パラメータに Y (はい) または N (いいえ) を設定します。デフォルトでは N が設定されます。

ロード・バランシングは、必要な場合のみ、つまり、複数のキューを使用するサーバによってサービスが提供される場合にのみ使用します。1 つのサーバによって提供されるサービス、または MSSQ セット (複数サーバ、単一キュー) のサーバによって提供されるサービスに対してロード・バランシングを設定する必要はありません。コンフィギュレーションにこのようなタイプのサービスしかない場合、LDBAL パラメータを N に設定します。LDBAL が N に設定されているときに複数のキューが同じサービスを提供すると、最初に使用可能なキューが選択されます。

LDBAL パラメータの特性

LDBAL パラメータには、以下の特性があります。

- LDBAL に Y を設定すると、ロード・バランシングが使用されます。
- LDBAL に Y を設定し、アプリケーションがネットワーク接続されている場合は、TMNETLOAD 環境変数を使用してローカル・サイトの優先順位を設定できます。
- LDBAL に N を設定すると、割り当てられたサーバが最初に使用可能なサーバになります。
- デフォルトは N です。

- LDBAL を設定するとオーバーヘッドが発生するため、必要な場合にのみ使用してください。
- すべての BEA Tuxedo サービスが 1 台のサーバによってのみ提供される場合、ロード・バランシングは使用しないでください。
- すべての BEA Tuxedo サービスが 1 つの MSSQ サーバ・セットによって提供される場合、ロード・バランシングは使用しないでください。

設定例

この例では、アプリケーションでロード・バランシングが有効になっています。

```
*RESOURCES
     LDBAL   Y
```

関連項目

- 『BEA Tuxedo システム入門』の「ロード・バランシング」

マスタ・マシンを識別する

MASTER マシンは、アプリケーションの起動およびアプリケーション全体の管理を制御します。各アプリケーションでは、必ず MASTER パラメータを設定して MASTER マシンを指定する必要があります。MASTER には、該当するコンピュータの論理マシン識別子 (LMID) を指定します。LMID は、管理者が指定する英数字の文字列であり、MACHINES セクションの LMID パラメータに割り当てられます。したがって、LMID パラメータの値が SITE1 の場合、MASTER の値も SITE1 でなければなりません。

アプリケーションをシャットダウンせずに MASTER マシンを終了するには、MASTER を移行できるようにしておく必要があります。移行を行うには、LMID に、プライマリ MASTER とバックアップ MASTER を指定する必要があります。

MASTER パラメータの特性

MASTER パラメータには、以下の特性があります。

- アプリケーションの起動と管理を制御する必須パラメータです。
- マシンを移行するには、バックアップ・マシンを作成するため、2 種類の LMID を指定する必要があります。
- RESOURCES セクションの例では、マスタ・サイトには SITE1、バックアップ・サイトには SITE2 が指定されています。

設定例

Site1 は MASTER マシン、SITE2 はバックアップ・マシンです。

```
*RESOURCES
     MASTER SITE1, SITE2
```

ネットワーク・グループの最大数を指定する

設定済みのネットワーク・グループの最大数を指定するには、MAXNETGROUPS パラメータを設定します。値には、1 以上 8192 未満を指定します。デフォルト値は 8 です。これは、オプション・パラメータです。

正常性チェックおよびブロッキング・タイムアウトを指定する

BBL は、定期的 (デフォルトでは 120 秒ごと) にマシン上のサーバの正常性をチェックします。ただし、この間隔は、SCANUNIT および SANITYSCAN パラメータを設定して変更することができます。さらに、BLOCKTIME パラメータを設定して、メッセージ、トランザクション、およびその他のシステム・アクティビティをブロッキングする際のタイムアウト値を設定することもできます。値は、5 の正の倍数で指定します。

サーバの正常性チェックを行うときに、次のチェックまでに繰り返す SCANUNIT の回数を指定するには、SANITYSCAN パラメータを使用します。現在のデフォルト値は、SANITYSCAN × SCANUNIT が約 120 秒になるように設定されています。

SCANUNIT、SANITYSCAN、および BLOCKTIME パラメータの特性

パラメータ	説明
SCANUNIT	チェック間隔とタイムアウトの粒度を制御します。SCANUNIT には、0 〜 60 の範囲の 5 の倍数を秒単位で指定します。 例 : SCANUNIT 10 デフォルト値は 10 です。

パラメータ	説明
SANITYSCAN	サーバの正常性チェックを行うときに、次のチェックまでに繰り返すスキャン単位 (SCANUNIT) の回数を指定します。 SANITYSCAN には、32,767 までの任意の数を指定できます。 デフォルト値は、SCANUNIT × SANITYSCAN が約 120 秒になるように設定されています。
BLOCKTIME	メッセージをブロックしておく期間を指定します。この期間を経過すると、タイムアウトが発生します。 SCANUNIT × BLOCKTIME の値は、32,767 以下でなければなりません。 デフォルト値は、SCANUNIT × BLOCKTIME が約 60 秒になるように設定されています。

ATMI 操作のブロックとタイムアウト

タイムアウトとは、集合的に、次の期間を示します。

- クライアントが要求キューへのメッセージ送信を待機する期間
- クライアントが応答キューからのメッセージを待機する期間
- サーバがクライアントを処理する期間
- クライアントがネットワーク上を移動する期間

ブロッキング・タイムアウトとは、ブロックされているクライアント要求が、ブロック解除されるまでの期間のことです。非同期型のサービス要求および会話のブロッキング・タイムアウトは、個々の送受信操作に対して適用されます。tpacall (3c)、tpconnect (3c)、または tpsend (3c) を使用してメッセージが送信される場合は、キューがいっぱいであり、送信されたメッセージがキューへの登録を待機する場合にのみタイムアウトが適用されます。メッセージを受信するための tpgetrply (3c) または tprecv(3c) 呼び出しがクライアント・プロセスから発行される場合、タイムアウトは、キューが空の場合にクライアントがメッセージの受信を待機する期間になります。

設定例

この例では、正常性チェックが 30 秒 ごとに実行されます。要求が 10 秒以上ブロックされることはありません。SCANUNIT に 10 が指定され、SANITYSCAN に 3 が指定されているため、10 秒の 3 倍、つまり 30 秒が経過すると、次のチェックが BBL により行われます。

```
*RESOURCES
       SCANUNIT     10
       SANITYSCAN    3
       BLOCKTIME     1
```

オペレーティング・システム・レベルのセキュリティを設定する

UID、GID、および PERM の 3 つのパラメータを設定すると、BEA Tuxedo の管理機能へのアクセスを、認可された管理者だけに制限できます。

UID および GID のデフォルト値は、アプリケーションのコンフィギュレーションの際に tmloadcf(1) コマンドを実行したユーザのユーザ ID (UID の場合) およびグループ ID (GID の場合) になります。ただし、MACHINES セクションの値を上書きして、この設定を変更することもできます。

UID、GID、および PERM パラメータの特性

パラメータ	説明
UID	管理者のユーザ ID。値は数値で表され、システムの起動とシャットダウンを担当するユーザの UNIX システム・ユーザ ID に対応しています。 デフォルト値は、`tmloadcf(1)` コマンドを実行するユーザの ID です。 例 : `UID=3002` 注記　Windows 2000 では、この値を 0 に設定してください。
GID	管理者のグループ ID を示す数値。 デフォルト値は、`tmloadcf(1)` コマンドを実行するユーザの ID です。 例 : `GID=100` 注記　Windows 2000 では、この値を 0 に設定してください。
PERM	アプリケーションの起動時に作成される IPC 資源に割り当てるパーミッションを指定する 8 進数値。このパラメータは、BEA Tuxedo システムの IPC 構造を不正なアクセスから保護する、第 1 レベルのセキュリティを指定します。これらの値は、実働アプリケーションに対して指定する必要があります。 デフォルト値は 0666 であり、これは、すべてのユーザに対して読み込み権と書き込み権を許可することを示します。 例 : `PERM=0660`

注記　リモート・マシンに対して割当てられている上記のパラメータは、上書きできます。リモート・マシンと `MASTER` マシンのユーザ ID およびグループ ID は、同じでなくてもかまいません。デフォルト値を上書きするには、コンフィギュレーション・ファイルの `MACHINES` セクションで、別のユーザ ID およびグループ ID を指定します。特に値を指定しない場合は、`RESOURCES` セクションで指定した値が使用されます。

セキュリティ・レベルを指定する

以下の 3 つのセキュリティ・レベルを設定できます。

- `PERM` パラメーターーアプリケーションのキューに対する書き込み権を制限することにより、必要最低限のセキュリティを設定します。
- `SECURITY` パラメーターーより高いセキュリティ・レベルを設定します。このパラメータが設定されている場合、クライアントはアプリケーションに参加するときにパスワードを入力しなければなりません。このパスワードは、`UBBCONFIG` ファイルから `TUXCONFIG` ファイルが生成されたときに管理者が割り当てたパスワードと照合されます。
- `AUTHSVC` パラメーターー最もレベルの高いセキュリティを設定します。このパラメータが設定されていると、アプリケーションへの参加を要求するクライアント要求は、認証サービスに送信されます。認証サービスは、BEA Tuxedo システムに用意されているデフォルトのサービスか、またはサード・パーティのベンダによるサービス (Kerberos サービスなど) になります。このセキュリティ・レベルは、`SECURITY` パラメータが設定されていないと使用できません。
- LDAP の単一のセキュリティ管理を有効にするには、UBBCONFIG ファイルの SERVERS セクションで LAUTHSVR を設定する必要があります。

SECURITY および AUTHSVC パラメータの特性

パラメータ	説明
SECURITY	アプリケーションに参加するときにパスワードの入力を求めるセキュリティ・レベル。有効な値は、NONE (デフォルト値)、APP_PW、USER_AUTH□AACL、および MANDATORY_ACL です。 LDAP の単一のセキュリティ管理を有効にするには、SECURITY レベルを USER_AUTH、MANDATORY_ACL、または ACL に設定する必要があります。 デフォルト値は NONE です。 例 : SECURITY APP_PW
AUTHSVC	認証サービスの名前。 SECURITY APP_PW またはそれ以上を指定する必要があります。 デフォルト値は、認証サービスなしです。 Kerberos によるクライアント認証が可能です。 例 : AUTHSVC “AUTHSVC’’

関連項目

- 『BEA Tuxedo のセキュリティ機能』の「ATMI のセキュリティについて」
- 『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』
- 『BEA Tuxedo のファイル形式とデータ記述方法』
- 『BEA Tuxedo コマンド・リファレンス』

サーバのセキュリティ属性を定義する

SEC_PRINCIPAL_NAME、SEC_PRINCIPAL_LOCATION、および SEC_PRINCIPAL_PASSVAR パラメータを使用すると、認証用のサーバのセキュリティ属性を指定できます。

- SEC_PRINCIPAL_NAME—さまざまなセキュリティ操作で使用するサーバのプリンシパル名を指定します。
- SEC_PRINCIPAL_LOCATION—プリンシパル・ユーザの秘密鍵の場所を指定します。
- SEC_PRINCIPAL_PASSVAR—プリンシパル・ユーザの秘密鍵をオープンするためのパスワードを含む環境変数を指定します。

セクション名	指定内容	上書きされるセクション
RESOURCES	ドメイン内で起動するすべてのシステム・サーバ	N/A
MACHINES	マシン上で起動するすべてのシステム・サーバ	RESOURCES
GROUPS	グループ内で起動するすべてのシステム・サーバおよび相互運用するアプリケーション・サーバ	MACHINES
SERVERS	サーバ上で起動するすべてのシステム・サービスおよび相互運用するアプリケーション・サービス	GROUPS

注記　上記の方針は、ワークステーション・ハンドラ、Domains のゲートウェイ処理、および相互運用するアプリケーション・サーバに適用されます。

関連項目

- 『BEA Tuxedo のセキュリティ機能』の「ATMI のセキュリティについて」
- 『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』の「セキュリティの管理」

共用メモリを保護する

SYSTEM_ACCESS パラメータを使用すると、共用メモリにあるシステム・テーブルを、アプリケーション・クライアントとアプリケーション・サーバから保護できます。このパラメータを指定しておくと、アプリケーションの開発中に、欠陥のあるアプリケーション・コードが不当なポインタによって誤って共用メモリの内容を破壊することを防ぐことができます。アプリケーションが完全にデバッグされ、テストされたら、パラメータの値を変更して、応答が早く返されるように設定できます。このパラメータの有効な値は、次のとおりです。

- PROTECTED—アプリケーション・コードと共にコンパイルされた BEA Tuxedo ライブラリは、システム・コードの実行中には共用メモリにアタッチされません。
- FASTPATH—BEA Tuxedo ライブラリは、常に共用メモリにアタッチされます。

値を選択した後で NO_OVERRIDE を指定すると、選択したオプションをクライアントや管理者が変更できなくなります (クライアントの場合は tpinit() 呼び出しの TPINIT 構造体、管理者の場合は SERVERS セクションで指定)。

PROTECTED、FASTPATH、および NO_OVERRIDE パラメータの特性

パラメータ	説明
PROTECTED	共用メモリの内部構造体が、アプリケーション・プロセスによって誤って破壊されるのを防ぎます。
FASTPATH（デフォルト）	アプリケーション・プロセスは、常に共用メモリへのアクセス権を保持した状態でアプリケーションに参加します。
NO_OVERRIDDE	指定されたオプション (PROTECTED または FASTPATH) は変更できません。

設定例

```
SYSTEM_ACCESS PROTECTED、NO_OVERRIDE
```

アプリケーションのシステム資源のアドレスを設定する

共用メモリのアドレスを設定するには、IPCKEY パラメータを設定します。このパラメータにより、BEA Tuxedo システムは、アプリケーションに対して IPC 資源を割り当てます。これにより、新たにアプリケーションに参加したプロセスは IPC 資源を簡単に見つけることができます。キーの値は、新しいアプリケーション・プロセスに対して掲示板、メッセージ・キュー、およびセマフォを割り当てるために内部で使用されます。単一プロセッサ・モードでは、このキーが掲示板を指定します。マルチプロセッサ・モードでは、このキーが DBBL のメッセージ・キューを指定します。

IPCKEY パラメータの特性

IPCKEY パラメータには、以下の特性があります。

- 必須パラメータです。
- 掲示板およびその他の IPC 資源へのアクセスを設定するために使用されます。
- 32,769 ～ 262,144 の範囲の整数を設定します。
- システム上のほかのアプリケーションの IPCKEY パラメータに同じ値を使用することはできません。値は、すべてのアプリケーションで一意でなければなりません。

設定例

```
*RESOURCES
   IPCKEY 39211
```

任意通知型メッセージの受信方法を指定する

NOTIFY パラメータを使用すると、クライアントが任意通知型メッセージを受信するデフォルトの方法を設定できます。ただし、クライアントは、tpinit() を呼び出してこの設定を上書きできます。

以下の 4 つの方法が可能です。

- IGNORE―クライアントは、任意通知型メッセージを無視します。
- DIPIN―クライアントは、tpchkunsol() を呼び出したとき、または ATMI 呼び出しを行ったときにのみ任意通知型メッセージを受信します。

- SIGNAL―クライアントは、シグナル・ハンドラに関数を呼び出させるシグナルをシステムに生成させ、つまり、tpsetunsol() を設定して、任意通知型メッセージを受信します。

  注記　この方法は、マルチスレッド化されたアプリケーションまたはマルチコンテキスト化されたアプリケーションでは使用できません。

- THREAD―任意通知型メッセージは、BEA Tuxedo システムがこの目的で管理する別のスレッドによって処理されます。

USIGNAL パラメータは、SIGNAL ベースの通知を使用する場合に使用されるシグナルを指定します。SIGUSR1 と SIGUSR2 の 2 種類のシグナルが生成されます。デフォルト設定は SIGUSR2 です。この方法を使用すると、直ちに通知を返すことができますが、ネイティブ・クライアントを実行しているときは、制限があります。つまり、送信プロセス側のユーザ ID が同じでなければなりません。ただし、ワークステーション・クライアントには、この制限はありません。

注記　この方法を適用できないプラットフォームもあります。

NOTIFY および USIGNAL パラメータの特性

パラメータ	説明
NOTIFY	IGNORE は、クライアントが任意通知型メッセージを無視することを示します。 DIPIN は、クライアントが、tpchkunsol() を呼び出すとき、または ATMI 呼び出しを行うときにのみ任意通知型メッセージを受信することを示します。 SIGNAL は、クライアントがシグナルにより任意通知型メッセージを受信することを示します。 デフォルト値は DIPIN です。 例 : NOTIFY SIGNAL

パラメータ	説明
USIGNAL	SIGUSR1 および SIGUSR2 は、このタイプのシグナルでクライアントに通知することを示します。 デフォルト値は SIGUSR2 です。 例 : USIGNAL SIGUSR1

コンフィギュレーション・ファイルの MACHINES セクションの作成方法

コンフィギュレーション・ファイルの 2 番目のセクションには、必ず MACHINES セクションを指定します。MACHINES セクションでは、アプリケーション内の各マシンに対するパラメータを定義します。これらのパラメータは、次の情報を指定します。

- マシンのアドレスと論理識別子 (LMID) のマッピング
- コンフィギュレーション・ファイル (TUXCONFIG) のロケーション
- インストールされている BEA Tuxedo ソフトウェアのロケーション (TUXDIR)
- アプリケーション・サーバのロケーション (APPDIR)
- アプリケーション・ログ・ファイル (ULOGPFX) のロケーション
- 環境ファイル (ENVFILE) のロケーション

注記　UID、GID、PERM、MAXACCESSERS、MAXOBJECTS、MAXCONV、および MAXGTT は、上書きできます。MAXGTT 以外のパラメータについては、RESOURCES セクションを参照してください。

次の表では、MACHINES セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

MACHINES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
SECURITY が ACL または MANDATORY_ACL に設定されている場合のキャッシュ内の ACL 用エントリ数	MAXACLCACHE (オプション)	キャッシュ内の ACL 用エントリ
このマシンから別のマシンにサービス要求を送信するコストを計算するときに追加する負荷	NETLOAD (オプション)	負荷の追加
アドレス (物理プロセッサの名前) とコンピュータの論理名。ほかのすべてのエントリは、このアドレスに指定されたマシンに関する情報を指定します。コンピュータの論理名は、LMID パラメータで指定します。	LMID (必須)	アドレスおよびマシン ID
UNIX セマフォ上のプロセスをブロックするまでに実行する、ユーザ・レベルでの掲示板のロック回数	SPINCOUNT (オプション)	掲示板のロックの上限値
マシンをクラスにグループ化するときに使用する値	TYPE (オプション)	クラスごとのグループ化の値
バイナリ形式の TUXCONFIG ファイルが置かれているマシン上のファイルまたはデバイスの絶対パス名 注記　このパラメータに指定するパス名は、TUXCONFIG 環境変数に指定したパス名 (大文字小文字の区別も含む) と完全に一致しなければなりません。一致しない場合は、tmloadcf(1) を正しく実行できません。	TUXCONFIG (必須)	コンフィギュレーション・ファイルのロケーション
特定のマシン上のプロセスが同時に参加できる会話の最大数	MAXCONV (オプション)	会話数の上限値
このマシンの DTP トランザクション・ログのサイズ (ページ単位)	TLOGSIZE (オプション)	DTP TLOG のサイズ

MACHINES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
このマシンの DTP トランザクション・ログの名前	`TLOGNAME` (オプション)	DTP トランザクション・ログの名前
マシン上のすべてのクライアントとサーバを、指定したファイルの環境で実行する値	`ENVFILE` (オプション)	環境変数の設定
このマシンの DTP トランザクション・ログ (`TLOG`) を格納する BEA Tuxedo のファイルシステム	`TLOGDEVICE` (オプション)	`TLOG` を格納するファイルシステム
このプロセッサ上で、掲示板に同時にアクセスできるプロセスの最大数	`MAXACCESSERS` (オプション)	IPC 資源の上限値
CORBA 環境で、このプロセッサのアクティブ・オブジェクト・テーブルに同時に対応できる CORBA オブジェクトの最大数	`MAXOBJECTS` (オプション)	IPC 資源の上限値
特定のマシンが同時に関与できるグローバル・トランザクションの最大数	`MAXGTT` (オプション)	同時に関与できるグローバル・トランザクションの上限値
ワークステーション・クライアント用に予約されている、このプロセッサのアクセサのエントリ数。このパラメータは、BEA Tuxedo システムのワークステーション・コンポーネントが使用される場合のみ指定されます。	`MAXWSCLIENTS` (オプション)	ワークステーション・アクセサのエントリの上限値
ブリッジ・プロセスで送信されるのを待つメッセージに対して割り当てられる領域の上限	`MAXPENDINGBYTES` (オプション)	メッセージ領域の制限
デバイスの先頭から BEA Tuxedo ファイルシステムの開始点 (このマシンの DTP トランザクション・ログを格納) までのページ単位の数値のオフセット	`TLOGOFFSET` (オプション)	DTP TLOG を含む数値のオフセット

MACHINES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
デバイスの先頭から BEA Tuxedo ファイルシステムの開始点 (このマシンの TUXCONFIG ファイルを格納) までのページ単位の数値のオフセット	TUXOFFSET (オプション)	TUXCONFIG を含む数値のオフセット
掲示板用に作成された IPC 構造体に関連付けるグループ ID の数値。有効な値は 0 〜 2147483647 です。値を指定しない場合、デフォルトで RESOURCES セクションの値が指定されます。	GID (オプション)	セキュリティ・アクセス
掲示板をインプリメントする IPC 構造体に関連付けるパーミッションを示す数値。このパラメータは、通常の UNIX システムの規則に従って (8 進数の 0600 など)、プロセスの読み取り権または書き込み権を指定します。値は 0001 以上 0777 以下の範囲で指定できます。値を指定しない場合、デフォルトで RESOURCES セクションの値が指定されます。	PERM (オプション)	セキュリティ・アクセス
掲示板用に作成された IPC 構造体に関連付けるユーザ ID の数値。有効な値は 0 〜 2147483647 です。値を指定しない場合、デフォルトで RESOURCES セクションの値が指定されます。	UID (オプション)	セキュリティ・アクセス
ID 確認のためのプロセスのプリンシパル名、プリンシパル・ユーザの秘密鍵の場所、およびパスワードを格納する環境変数	SEC_PRINCIPAL_NAME、SEC_PRINCIPAL_LOCATION、SEC_PRINCIPAL_PASSVAR	セキュリティ属性

MACHINES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
アプリケーション・ディレクトリ (APPDIR) の絶対パス名。このディレクトリは、このマシンで起動するすべてのアプリケーションおよび管理サーバのカレント・ディレクトリになります。また、BEA Tuxedo システム・ソフトウェアのインストール先ディレクトリの絶対パス名でもあります。	`TUXDIR` (必須)	システム・ソフトウェアおよびアプリケーション・ソフトウェアのロケーション
リモート・プロセス (*string_value1*) およびローカル・プロセス (*string_value2*) に送信される、自動圧縮の対象となるメッセージのしきい値。このしきい値を超えると、メッセージは圧縮されます。	`CMPLIMIT` (オプション)	メッセージ・サイズのしきい値
このマシンの `userlog`(3c) メッセージ・ファイル名の接頭辞として使用される絶対パス名。	`ULOGPFX` (オプション)	ULOG のパス名

MACHINES セクションの例

以下は、ATMI 環境のコンフィギュレーション・ファイルの MACHINES セクションの例です。

```
*MACHINES
gumby          LMID=SITE1
               TUXDIR=”/tuxdir”
               APPDIR=”/home/apps/mortgage”
               TUXCONFIG=”/home/apps/mortgage/tuxconfig”
               ENVFILE=”/home/apps/mortgage/ENVFILE”
               ULOGPFX=”/home/apps/mortgage/logs/ULOG”
               MAXACCESSERS=100
                MAXCONV=15
```

以下は、CORBA 環境のコンフィギュレーション・ファイルの MACHINES セクションの例です。

```
*MACHINES
gumby          LMID=SITE1
               TUXDIR=”/tuxdir”
               APPDIR=”/home/apps/mortgage”
               TUXCONFIG=”/home/apps/mortgage/tuxconfig”
               ENVFILE=”/home/apps/mortgage/ENVFILE”
               MAXOBJECTS=700
               ULOGPFX=”/home/apps/mortgage/logs/ULOG”
               MAXACCESSERS=100
```

MACHINES セクションの例のパラメータ

上の MACHINES セクションの例では、以下のパラメータと値が指定されています。

パラメータ	説明
gumby	UNIX システムで uname -n コマンドを実行すると返されるマシン名。Windows 2000 システムの場合は、[コントロール パネル] の [ネットワーク] にある [コンピュータ名] で指定します。名前はすべて大文字で指定します。
LMID=SITE1	マシン gumby の論理マシン識別子。

パラメータ	説明
TUXDIR	インストールされている BEA Tuxedo ソフトウェアへの絶対パス。二重引用符で囲みます。
APPDIR	アプリケーション・ディレクトリへの絶対パス。二重引用符で囲みます。
TUXCONFIG	コンフィギュレーション・ファイルの絶対パス名。二重引用符で囲みます。 注記　このパラメータに指定するパス名は、TUXCONFIG 環境変数に指定したパス名 (大文字小文字の区別も含む) と完全に一致しなければなりません。パス名が一致していない場合、tmloadcf(1) は正常に実行されません。
ENVFILE	環境情報を含むファイルの絶対パス名。二重引用符で囲みます。
ULOGPFX	ログ・ファイルの接頭辞として使用する名前の絶対パス名。二重引用符で囲みます。
MAXACCESSERS	このマシンに対し、システム全体にわたる値 (RESOURCES セクションで定義済み) を 100 に上書きします。
MAXOBJECTS	CORBA サンプルで使用。このマシンに対し、システム全体にわたる値 (RESOURCES セクションで定義済み) を 700 に上書きします。
MAXCONV	このマシンに対し、システム全体にわたる値 (RESOURCES セクションで定義済み) を 15 に上書きします。

MACHINES セクションの例のカスタマイズ方法

MACHINES セクションをカスタマイズするには、以下を指定します。

- マシン名。gumby で指定します。

注記　Windows 2000 システムの場合、マシン名は大文字で指定する必要があります。

- BEA Tuxedo ソフトウェアが格納されているディレクトリの絶対パス名。TUXDIR で指定します。
- アプリケーション・ディレクトリの絶対パス名。APPDIR で指定します。
- ENVFILE、TUXCONFIG、および ULOGPFX のシステム上での絶対パス名。

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5)
- 3-55 ページの「コンフィギュレーション・ファイルの GROUPS セクションの作成方法」

キャッシュ内の ACL 用エントリの最大数を指定する

SECURITY が ACL または MANDATORY_ACL に設定されている場合は、MAXACLCACHE パラメータを使用して、キャッシュ内の ACL 用エントリの数を指定できます。このパラメータに適切な値を設定すると、以下のことを実現できます。

- 共用メモリの資源を節約できます。
- ACL をチェックするためのディスクへのアクセス数を少なくすることができます。

値には、10 ～ 30,000 を指定します。デフォルト値は 100 です。

サービス要求の負荷を定義する

NETLOAD パラメータを使用すると、1 つのマシンから別のマシンへサービス要求を送信するときにかかるコストを計算する際に追加する負荷を指定できます。値には、0 以上 32,768 未満を指定します。デフォルト値は 0 です。

関連項目

- 『BEA Tuxedo システム入門』の「ロード・バランシング」

物理アドレスとマシン ID を予約する

まず、アドレスを指定する箇所に、MASTER マシンのアドレスを定義します。これが、MACHINES セクションのエントリの基本となります。エントリ内のその他のすべてのパラメータは、このアドレスで指定されたマシンに関する情報を指定します。UNIX システムの場合は、uname -n コマンドを呼び出し、出力された値にアドレスを設定します。Windows 2000 システムの場合は、[コントロール パネル] の [ネットワーク] にある [コンピュータ名] を参照してください。

LMID パラメータは必須パラメータです。このパラメータは、アドレスを指定したばかりのコンピュータを示す論理名を指定します。論理名には、アプリケーション内のマシン間で一意な英数字を指定します。

アドレスと LMID パラメータの特性

アドレスおよびマシン ID には、以下の特性があります。

- アドレスおよびマシン ID は、次の形式で指定します。

  `address LMID=logical_machine_name`

  アドレス (address) は、物理プロセッサ名を示します。

- `LMID` は、次の形式で指定します。

  `LMID=logical_machine_name`

  `LMID` は物理プロセッサに対する論理マシン名です。論理マシン名には、`MACHINES` セクションで一意な英数字を指定します。

ロック・スピンの回数を設定する

BEA Tuxedo システムの一部の操作 (サービス名のルックアップやトランザクションなど) では、掲示板をロックし、掲示板へのアクセスを 1 つのプロセスだけに制限する場合があります。プロセスまたはスレッドの処理中に、別のプロセスまたはスレッドによって掲示板がロックされていることがわかると、その処理は再試行されるか、または `SPINCOUNT` で指定された回数のロック・スピンが行われます。指定された回数のロック・スピンが行われた後、処理はキューでスリープ状態になります。スリープ状態は資源を消費するため、一定のロック・スピンを行ってからスリープ状態になるように設定しておく方が効率的です。

SPINCOUNT パラメータの特性

`SPINCOUNT` パラメータの値は、アプリケーションおよびシステムによって異なりますが、以下の基本的なガイドラインを覚えておくと便利です。

- ユニプロセッサ・システム上のプロセスは、ロック・スピンができません。ユニプロセッサ・システムのプロセスがアクセスした掲示板がロックされている場合、そのプロセスは、できるだけ早く実行状態に戻されなければなりません。ただし、これは、新しいプロセスが直ちに中断されなければ実現できません。
- ユニプロセッサの場合、`SPINCOUNT` の適切な値は 1 です。
- マルチプロセッサの場合、`SPINCOUNT` を 5,000 から始めるのが適切ですが、場合によっては 100,000 が適切です。
- `SPINCOUNT` の値を設定した後で、アプリケーションのスループットを確認します。`SPINCOUNT` の値は、`TMIB` を使用してシステムの実行中に調整できます。

マシンをタイプ別に指定する

TYPE パラメータを使用すると、マシンをクラスごとにグループ化できます。TYPE には、15 文字以下の任意の文字列を設定できます。

TYPE パラメータの特性

- 2 つのマシンに対して同じ TYPE が指定されている場合、これらのマシン間では、データの符号化および復号化は行われません。
- TYPE には、任意の文字列値を指定できます。このパラメータは、比較のために使用します。
- TYPE パラメータは、アプリケーションが異機種ネットワークのマシンで構成されている場合、またはネットワーク内のマシン上でさまざまなコンパイラを使用している場合に使用します。
- このパラメータを指定しない場合は、デフォルトでヌル文字列になり、値が指定されないすべてのエントリが対象となります。

コンフィギュレーション・ファイルのロケーションを識別する

コンフィギュレーション・ファイルのロケーションおよびマシンを識別するエントリ・ファイル名を指定するには、必須パラメータの TUXCONFIG を設定します。TUXCONFIG パラメータには、最大 64 文字までの絶対パス名を指定し、値は二重引用符で囲みます。

注記　このパラメータに指定するパス名は、TUXCONFIG 環境変数に指定したパス名 (大文字小文字の区別も含む) と完全に一致しなければなりません。パス名が一致していない場合、tmloadcf(1) は正常に実行されません。

TUXCONFIG パラメータの特性

TUXCONFIG パラメータには、以下の特性があります。

- TUXCONFIG パラメータの構文は、TUXCONFIG="*full_path_of_tuxconfig*" です。
- このパラメータは、コンフィギュレーション・ファイルのロケーションおよびファイル名を識別します。
- TUXCONFIG には、64 文字以下の文字列を指定できます。
- TUXCONFIG の値は、TUXCONFIG 環境変数の値と一致していなければなりません。

DTP トランザクション・ログのサイズを指定する

TLOGSIZE パラメータは、このマシンの DTP トランザクション・ログのサイズをページ単位で指定します。オペレーティング・システムのファイルシステム上の空き容量に応じ、0 より大きく 2048 以下の値を指定します。デフォルトは 100 ページです。

DTP トランザクション・ログの名前を定義する

TLOGNAME パラメータは、このマシンの DTP トランザクション・ログの名前を定義します。デフォルト値は、TLOG です。1 つの TLOGDEVICE に複数の TLOG がある場合、各 TLOG の名前は一意でなければなりません。TLOGNAME には、TLOG テーブルの作成先である TLOGDEVICE の VTOC (ボリューム一覧) 内のテーブル名とは異なる名前を指定する必要があります。TLOGNAME には、30 文字以下の英数字を指定します。

環境変数の設定を指定する

ENVFILE パラメータを使用すると、BEA Tuxedo システムが起動するすべてのプロセスに対する環境変数を格納したファイルを指定できます。各プロセスの TUXDIR および APPDIR は、システム側で設定されます。したがって、これらのパラメータはファイル内で指定しないでください。

ただし、アプリケーション操作に関連する次のパラメータについては、環境設定を指定できます。

- FIELDTBLS、FLDTBLDIR
- VIEWFILES、VIEWDIR
- TMCMPLIMIT
- TMNETLOAD

ENVFILE パラメータの特性

ENVFILE は、以下の特性を持つオプション・パラメータです。

- ENVFILE パラメータは、ENVFILE="*envfile*" の形式で、文字列を二重引用符で囲んで示します。
- ENVFILE は、BEA Tuxedo システムが起動するすべてのプロセスに対する環境変数を格納したファイルです。UBBCONFIG ファイルは、同様の方法、つまり完全に限定されたパス名を使用して警告を生成します。
- FIELDTBLS、FLDTBLDIR などを設定しますが、TUXDIR と APPDIR は設定しません。
- 設定はすべてハード・コーディングする必要があります。FLDTBLDIR=$APPDIR などの評価を伴う設定はできません。
- ファイルのエントリの形式は、*VARIABLE*=string です。

環境変数の設定の詳細については、『BEA Tuxedo のファイル形式とデータ記述方法』の tuxenv(5) を参照してください。

TLOG を含む BEA Tuxedo ファイルシステムを定義する

TLOGDEVICE パラメータは、該当するマシンの DTP トランザクション・ログ (TLOG) を含む BEA Tuxedo のファイルシステムを指定します。TLOG は、BEA Tuxedo システムの VTOC テーブルとしてデバイスに格納されています。TLOGDEVICE の値は、64 文字以下の文字列でなければなりません。

このパラメータを指定しない場合、そのマシンには TLOG がないものと見なされます。

マシンで同時に実行できるグローバル・トランザクションの最大数を指定する

MAXGTT パラメータは、特定のマシンが同時に関与できるグローバル・トランザクションの最大数を指定します。このパラメータには、0 以上 32,768 未満の値を指定します。RESOURCES セクションで指定した値は、MACHINES セクションでマシンごとに上書きできます。

ワークステーション・クライアントのアクセサ・エントリ数を定義する

MAXWSCLIENTS パラメータは、ワークステーション・クライアント用に確保しておくマシン上のエントリ数を定義します。MAXWSCLIENTS 用に確保しておくアクセサ・スロットの数は、慎重に設定してください。この値は、このマシンの MAXACCESSERS で指定したアクセサ総数のうちの一部になります。このマシンのほかのクライアントやサーバは、MAXWSCLIENTS 用に確保されたアクセサ・スロットを使用できません。ワークステーション・クライアントからシステムへのアクセスは、BEA Tuxedo システムに組み込まれている代理プロセス、つまり BEA Tuxedo ワークステーション・ハンドラ (WSH) によって多重化されます。そのため、このパラメータを適切に設定すると、IPC 資源を節約できます。

MAXWSCLIENTS には、0 以上 32,768 未満の値を指定します。値を指定しない場合は、デフォルト値の 0 が設定されます。この値を MAXACCESSERS の値より大きい値に設定すると、エラーが返されます。

注記　MAXWSCLIENTS の値は、ライセンス供与されているユーザ数によって制限されます。

BRIDGE 経由で送信されるメッセージ用の領域を定義する

MAXPENDINGBYTES パラメータは、BRIDGE プロセスによる送信を待機するメッセージに割り当てる領域の制限を定義します。値には、100,000 から MAXLONG までの値を指定します。

MAXPENDINGBYTES が重要になるのは、以下の 2 つの状況です。

- BRIDGE が非同期接続を要求する場合
- すべての回線がビジー状態の場合

メモリおよびディスク領域のサイズが大きい大型コンピュータの場合は MAXPENDINGBYTES の値を大きくし、小型のコンピュータの場合は MAXPENDINGBYTES の値を小さくします。

DTP トランザクション・ログのオフセットを指定する

どの BEA Tuxedo ファイルシステムにもボリューム一覧 (VTOC: Volume Table of Contents) があります。これは、汎用デバイス・リスト (UDL: Universal Device List) で指定されたデバイス上のファイルの一覧です。UDL には、BEA Tuxedo システム・テーブルの物理的な格納位置が指定されています。BEA Tuxedo システムのアプリケーションでは、すべてのシステム・ファイルを同じ raw ディスクやオペレーティング・システムのファイルシステム上に格納することができます。

TLOGOFFSET パラメータは、このマシンのデバイスの先頭から BEA Tuxedo ファイルシステムの開始点 (このマシンの DTP トランザクション・ログを格納) までのオフセットをページ単位で指定します。オフセットには、0 以上でデバイス上のページ数より小さい値を指定します。デフォルト値は 0 です。

TUXCONFIG のオフセットを定義する

どの BEA Tuxedo ファイルシステムにもボリューム一覧 (VTOC: Volume Table of Contents) があります。これは、汎用デバイス・リスト (UDL: Universal Device List) で指定されたデバイス上のファイルの一覧です。UDL には、BEA Tuxedo システム・テーブルの物理的な格納位置が指定されています。BEA Tuxedo システムのアプリケーションでは、すべてのシステム・ファイルを同じ raw ディスクやオペレーティング・システムのファイルシステム上に格納することができます。

TUXOFFSET パラメータは、このマシンのデバイスの先頭から BEA Tuxedo ファイルシステムの開始点 (このマシンの TUXCONFIG を格納) までのオフセットをページ単位で定義します。この値が環境でどのように使用されるかについては、MACHINES セクションの ENVFILE を参照してください。

TUXOFFSET パラメータの特性

- オフセットには、0 以上でデバイス上のページ数より小さい値を指定します。
- デフォルトのオフセットは 0 です。
- 0 以外の TUXOFFSET の値は、マシン上で起動するすべてのサーバの環境に設定されます。

システム・ソフトウェアおよびアプリケーション・サーバ・ソフトウェアのロケーションを識別する

サーバをサポートするアプリケーション内の各マシンは、BEA Tuxedo システム・ソフトウェアおよびアプリケーション・サーバのコピーを備えている必要があります。システム・ソフトウェアのロケーションは、TUXDIR パラメータで指定します。アプリケーション・ソフトウェアのロケーションは、APPDIR パラメータで指定します。どちらも必須パラメータです。APPDIR パラメータで指定したディレクトリは、すべてのプロセスの現在の作業ディレクトリになります。BEA Tuxedo ソフトウェアは、TUXDIR/bin および APPDIR 内で実行可能ファイルを検索します。

APPDIR および TUXDIR パラメータの特性

パラメータ	説明
APPDIR	構文は、APPDIR=“*APPDIR*” の形式で指定し、絶対パス名を二重引用符で囲んで示します。 APPDIR は、アプリケーション・ソフトウェアのロケーションを示します。 APPDIR は必須パラメータです。 APPDIR は、サーバ・プロセスの現在の作業ディレクトリになります。
TUXDIR	構文は、TUXDIR=“*TUXDIR*” の形式で指定し、絶対パス名を二重引用符で囲んで示します。 TUXDIR は、BEA Tuxedo ソフトウェアのロケーションを示します。 TUXDIR は必須パラメータです。

圧縮するメッセージのしきい値を指定する

CMPLIMIT パラメータは、リモート・プロセス (*string_value1*) およびローカル・プロセス (*string_value2*) に送信される、自動圧縮の対象となるメッセージのしきい値を定義します。このしきい値を超えると、メッセージは圧縮されます。

どちらの値も、負以外の整数または文字列 MAXLONG でなければなりません。値を指定しない場合は、デフォルトの MAXLONG,MAXLONG が指定されます。

注記　CMPLIMIT の値を設定した後で、アプリケーションのスループットを確認します。CMPLIMIT の値は TMIB を使用してシステムの実行中に調整できます。

例

CMPLIMIT=*string_value1*,*string_value2*

ULOG のパス名を指定する

ULOGPFX パラメータを設定すると、マシン上の userlog(3c) メッセージ・ファイルの接頭辞として使用する名前の絶対パス名を指定できます。指定したマシンの ULOGPFX の値は、このマシン上で実行されるすべてのサーバ、クライアント、および管理プロセスに関する userlog(3c) メッセージ・ファイルを作成するために使用されます。このパラメータが指定されない場合、APPDIR 環境変数で指定されたパスが使用されます。*mmddyy* (月、日、年) が接頭辞に追加されると、完全なログ・ファイル名になります。

ULOGPFX パラメータの特性

ULOGPFX パラメータには、以下の特性があります。

- ULOGPFX パラメータは、ULOGPFX="*ULOGPFX*" の形式で、文字列を二重引用符で囲んで示します。
- アプリケーションのログ・ファイルには、TPESYSTEM および TPEOS エラーに関するすべてのメッセージが記録されます。
- ユーザ・ログを使用して、アプリケーション・エラーをログ・ファイルに記録できます。
- ULOGPFX は、*APPDIR*/ULOG にデフォルト設定されます。
- たとえば、BANKLOG.022667 というファイルの場合、userlog の接頭辞は、次のように指定されます。
  ULOGPFX="/mnt/usr/appdir/logs/BANKLOG"

関連項目

- 3-55 ページの「コンフィギュレーション・ファイルの GROUPS セクションの作成方法」

コンフィギュレーション・ファイルの GROUPS セクションの作成方法

GROUPS セクションでは、複数のサーバを論理的にグループ化することができます。グループ化したサーバを使用すると、リソース・マネージャにアクセスしたり、サーバ・グループの移行を簡単に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 GROUPS セクションでは、サーバ・グループが定義されています。アプリケーション・サーバをマシン上で実行するには、マシンに対して少なくとも 1 つのサーバ・グループを定義する必要があります。ただし、マシンに対してサーバ・グループが定義されていなくても、アプリケーションの一部にグループを含め、そのサイトから管理コマンド `tmadmin(1)` を実行できます。

トランザクションに関与しない非分散型のシステムでは、グループのコンフィギュレーションは比較的単純です。各グループに対して、グループ名をグループ番号および論理マシン ID にマッピングするだけです。分散トランザクション・システムをサポートする柔軟性も備えられています。

次の表では、GROUPS セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

GROUPS **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
グループの論理名	*GROUPNAME* (必須)	グループ名。

GROUPS セクションで指定する情報	パラメータ (必須 / オプション)	参照先 (クリックするとリンク先にジャンプ)
このサーバ・グループに関連付けられたグループ番号 0 より大きく、30000 未満の番号を指定します。番号は、GROUPS セクションのすべてのエントリの中で一意でなければなりません。	GRPNO (必須)	グループ番号
リソース・マネージャをクローズするときに必要な、リソース・マネージャに依存する情報	CLOSEINFO (オプション)	リソース・マネージャをクローズするための情報
リソース・マネージャをオープンするときに必要な、リソース・マネージャに依存する情報	OPENINFO (オプション)	リソース・マネージャをオープンする情報
TMSNAME が指定されている場合に、関連するグループに対して起動するトランザクション・マネージャ・サーバの数	TMSCOUNT (オプション)	グループ内の TM サーバの数
ID 確認のためのプロセスのプリンシパル名、プリンシパル・ユーザの秘密鍵の場所、およびパスワードを格納する環境変数	SEC_PRINCIPAL_NAME、SEC_PRINCIPAL_LOCATION、SEC_PRINCIPAL_PASSVAR	セキュリティ属性
グループ内のすべてのサーバを、指定したファイルの環境で実行する値	ENVFILE (オプション)	サーバ・グループ環境
このサーバ・グループが MACHINES セクションの *string_value1* でシンボリックに指定されているマシン上、または、SHM モードのデフォルトの値に存在することを示す値	LMID (必須)	サーバ・グループのロケーション
このグループに関連付けられているトランザクション・マネージャ・サーバの名前。	TMSNAME (オプション)	グループのトランザクション・マネージャ・サーバ

ATMI の GROUPS セクションの例

以下は、ATMI 環境のコンフィギュレーション・ファイルの GROUPS セクションの例です。

```
##EVBGRP1 LMID=SITE1      GRPNO=104

DEFAULT:TMSNAME=TMS_SQL TMSCOUNT=2 LMID=SITE1
BANKB1GRPNO=1 OPENINFO="TUXEDO/SQL:APPDIR1/bankdl1:bankdb:readwrite"
BANKB2GRPNO=2 OPENINFO="TUXEDO/SQL:APPDIR1/bankdl2:bankdb:readwrite"
BANKB3GRPNO=3 OPENINFO="TUXEDO/SQL:APPDIR1/bankdl3:bankdb:readwrite"
```

CORBA 環境の GROUPS セクションの例

以下は、Tuxedo CORBA University の Production サンプル・アプリケーションの UBBCONFIG ファイルにある、GROUPS セクションの例です。この例では、UBBCONFIG ファイルの ROUTING セクションにある RANGES 識別子で指定されたグループを識別し、設定する必要があります。

Production サンプルでは、ORA_GRP1、ORA_GRP2、APP_GRP1、および APP_GRP2 の 4 つのグループが指定されています。ここでは、これらのグループを設定し、実行するマシンを識別しなければなりません。

```
*GROUPS

APP_GRP1
   LMID = SITE1
   GRPNO = 2
   TMSNAME = TMS

APP_GRP2
   LMID = SITE1
   GRPNO = 3
   TMSNAME = TMS

ORA_GRP1
   LMID = SITE1
   GRPNO = 4

OPENINFO = "ORACLE_XA:Oracle_XA+Acc=P/scott/tiger+SesTm=100+LogDir=.+MaxCur=5"
```

```
   CLOSEINFO = ""
   TMSNAME = "TMS_ORA"

ORA_GRP2
   LMID = SITE1
   GRPNO = 5

OPENINFO = "ORACLE_XA:Oracle_XA+Acc=P/scott/tiger+SesTm=100+LogDir=.+MaxCur=5"

CLOSEINFO = ""
TMSNAME = "TMS_ORA"
```

上の例は、ORA_GRP1、ORA_GRP2、APP_GRP1、および APP_GRP2 グループの設定方法を示しています。GROUPS セクションのグループ名と、ROUTING セクションで指定されたグループ名の対応については、3-123 ページの「University Production サンプル・アプリケーションの CORBA ファクトリ・ベース・ルーティング」を参照してください。ルーティング機能を正しく動作するには、両方のグループ名を一致させる必要があります。また、アプリケーションでグループを設定する際にグループ名を変更した場合は、ROUTING セクションにも反映しなければなりません。

注記　BEA Tuxedo ソフトウェアに収録されている Production サンプル・アプリケーションは、1 台のマシンで実行するように設定されていますが、LMID パラメータでほかのマシンを指定することにより、複数のマシンで実行するように設定することもできます。この方法は、RESOURCES セクションで MODEL MP パラメータが指定されていることを前提としています。

関連項目

- 3-77 ページの「コンフィギュレーション・ファイルの SERVERS セクションの作成方法」

グループ名、グループ番号、および LMID を指定する

GROUPS セクションの基本であるグループ名は、グループを識別する名前であり、英数字で指定します。グループ名により、グループの論理名 (*string_value*) が決まります。各グループには、一意なグループ番号 (GRPNO) が指定されます。これは、必須パラメータです。各グループは、すべて 1 つの論理マシン (LMID) 上に常駐する必要があります。

LMID は、このサーバ・グループが MACHINES セクションの *string_value1* というシンボリック名で指定されているマシンに存在することを示します。

グループ名、グループ番号、および LMID の特性

パラメータ	説明
Group_name required_ parameters [*optional_ parameters*]	必須パラメータです。 グループを識別する英数字名です。 一意であり、グループの論理名を指定します。
GRPNO (グループ番号)	必須パラメータであり、一意。
LMID=*string_value1* [,*string_value2*]	必須パラメータです。 LMID 値には、30 文字以下の英数文字列を指定します。 論理マシン名は 2 つまで指定できます。2 つ目の論理名が指定され、サーバ・グループが移行できる場合、サーバ・グループが関連付けられているマシンは移行できます。

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の `UBBCONFIG(5)`
- 3-65 ページの「コンフィギュレーション・ファイルの NETWORK セクションの作成方法」

TMS の名前および各グループの TMS の数を指定する

分散トランザクション、つまり、複数のリソース・マネージャまたはマシンにわたるトランザクションに参加するサーバを含むサーバ・グループのエントリには、トランザクション・マネージャ・サーバ (TMS: Transaction Manager Server) を指定する必要があります。TMS を指定するには、`TMSNAME` パラメータを設定します。このパラメータは、サーバ・グループの起動時に `tmboot(1)` が実行するファイル (*`string_value`*) を指定します。

値「TMS」は、ヌル XA インターフェイスの使用を示すために予約されています。このインターフェイスは、リソース・マネージャのないサーバ・グループ用に使用できます。ただし、リソース・マネージャがない場合は、TMS も必要ありません。このサーバ・グループは、トランザクションに関与するメッセージの影響を受けます。「TMS」以外で、さらに空ではない値が指定された場合は、このエントリの `LMID` 値に関連するマシンに対して、`TLOGDEVICE` を指定する必要があります。各 TM サーバには、一意なサーバ識別子が自動的に割り当てられます。サーバは何度でも再起動できます。

`TMSNAME` が指定されている場合は、`TMSCOUNT=`*`number`* も指定し、関連するサーバ・グループ用の TMS の数を設定する必要があります。`TMSCOUNT` のデフォルト値は 3 です。このパラメータに 0 以外の値を指定する場合、指定できる最小値は 2、最大値は 256 です。サーバは、自動的に MSSQ セットに設定されます。

グループ内のサーバの環境ファイルの場所を識別する

`ENVFILE` 環境変数 (`ENVFILE=`*`string_value`*) に不正なファイル名が指定された場合、環境には何も値が設定されません。環境ファイルの各行は、*`ident=value`* の形式で指定します。*`ident`* は下線 (_) または英数字で構成します。

`value` 内の `${`*`env`*`}` という形式の文字列は、ファイルの処理時に、環境内の既存の変数を使用して展開されます。前方参照はサポートされていません。値が設定されていない場合、変数は空の文字列に置換されます。バックスラッシュ (\) を使用すると、ドル記号 ($) およびバックスラッシュ自体をエスケープできます。その他のシェルのクォーテーションおよびエスケープのメカニズムは無視され、展開された value がそのまま環境に組み込まれます。

環境ファイルは、コンフィギュレーション・ファイル内の少なくとも 2 つのセクションに用意されています。BEA Tuxedo システムでは、次の順序で環境ファイルが読み取られます。

1. `MACHINES` セクションの `ENVFILE`
2. `GROUPS` セクションの `ENVFILE`
3. `SERVERS` セクションの `ENVFILE` (オプション)

`SERVERS` セクションの値は、`GROUPS` セクションの値を上書きします。`GROUPS` セクションの値は、`MACHINES` セクションの値を上書きします。

リソース・マネージャをオープンおよびクローズするときに必要な情報を定義する

`OPENINFO` および `CLOSEINFO` パラメータには、256 以下の英数字で構成する文字列を二重引用符で囲んで指定します。これらの設定は、このグループ (グループ名) のリソース・マネージャをオープンおよびクローズするときに必要な、リソース・マネージャに依存する情報を指定します。

このグループの `TMSNAME` パラメータが設定されていないか、または `TMS` が設定されている場合、この値は無視されます。`TMSNAME` パラメータに `TMS` 以外の値が設定されているが、`OPENINFO` 文字列にヌル文字列 (`""`) が設定されているか、または何も設定されていない場合は、このグループ用のリソース・マネージャは存在するが、`open` 操作を実行するための情報は不要であることを示します。`TMSNAME` パラメータに `TMS` 以外の値が設定されているが、`CLOSEINFO` 文字列にヌル文字列 (`""`) が設定されているか、または何も設定されていない場合は、このグループ用のリソース・マネージャは存在するが、close 操作を実行するための情報は不要であることを示します。

`OPENINFO` 文字列の形式は、基となるリソース・マネージャのベンダごとに異なります。ベンダ固有の情報の先頭には、トランザクション・インターフェイス (XA インターフェイス) の公開名とコロン (:) が付きます。

たとえば、BEA Tuxedo /Q データベースの場合、`OPENINFO` は次のような形式になります。

- UNIX の場合
  `OPENINFO = "TUXEDO/QM:`*`qmconfig`*`:`*`qspace`*`"`
- Windows 2000 の場合
  `OPENINFO = "TUXEDO/QM:`*`qmconfig`*`;`*`qspace`*`"`

上記の設定では、TUXEDO/QM が BEA Tuxedo /Q XA インターフェイスの公開名です。*qmconfig* は、キュー・スペースを設定する QMCONFIG (『BEA Tuxedo コマンド・リファレンス』の qmadmin(1) を参照) の名前です。*qspace* はキュー・スペースの名前です。Windows 2000 では、*qmconfig* の後に指定する区切り文字として、セミコロン (;) を使用します。

注記　BEA Tuxedo /Q データベースでは、CLOSEINFO 文字列は使用されません。

その他のベンダのデータベースでは、OPENINFO 文字列の形式は、基となるリソース・マネージャのベンダごとに異なります。たとえば、次の OPENINFO 文字列は、Oracle のリソース・マネージャをオープンするときに必要な情報を示します。

```
OPENINFO="Oracle_XA:Oracle_XA+Acc=P/Scott/*****+SesTm=30+LogDit=/
tmp"
```

Oracle_XA は、Oracle XA インターフェイスの公開名です。OPENINFO 文字列内の 5 つの連続するアスタリスク (*) は、暗号化されたパスワードを示します。次に、パスワードについて説明します。

リソース・マネージャに渡される OPENINFO 文字列内のパスワードは、平文または暗号化された形式で格納されます。パスワードを暗号化するには、まず、OPENINFO 文字列内のパスワードが必要な場所に、5 つ以上の連続するアスタリスクを入れます。次に、tmloadcf(1) コマンドを実行して、UBBCONFIG ファイルをロードします。tmloadcf() は、アスタリスクの文字列を検出すると、パスワードの作成をユーザに要求します。次に例を示します。

```
tmloadcf -y /usr5/apps/bankapp/myubbconfig
Password for OPENINFO (SRVGRP=BANKB3):
password
```

tmloadcf() は、暗号化したパスワードを TUXCONFIG ファイルに格納します。tmunloadcf(1) を使用して TUXCONFIG ファイルから UBBCONFIG ファイルを再び生成すると、パスワードは、暗号化された形式 (@@ で区切られる) で UBBCONFIG ファイルに出力されます。次に例を示します。

```
OPENINFO="Oracle_XA:Oracle_XA+Acc=P/Scott/@@A0986F7733D4@@+SesTm=
30+LogDit=/tmp"
```

tmloadcf() の実行時に、tmunloadcf() によって生成された UBBCONFIG ファイル内で暗号化されたパスワードが検出されても、ユーザに対してパスワードの作成は要求されません。

コンフィギュレーション・ファイルの NETWORK セクションの作成方法

分散アプリケーションが複数のマシンで構成されている場合は、コンフィギュレーション・ファイルに NETWORK セクションを作成する必要があります。このセクションは、マシン間の通信を設定します。ネットワーク・グループを作成するには、アプリケーションの UBBCONFIG ファイルで、NETGROUPS セクションおよび NETWORK セクションを設定します。

次の表では、NETWORK セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

NETWORK **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
LMID 上に置かれた BRIDGE プロセスがネットワークにアクセスするのに使用するデバイス名	BRIDGE (オプション)	BRIDGE デバイス名
BRIDGE プロセスで使用される完全なネットワーク・アドレス (LMID の接続指示受け付けアドレス)	NADDR (必須)	BRIDGE ネットワーク・アドレス
このマシンに対してネットワーク・リンクを確立するときに必要な最低限の暗号化レベル	MINENCRYPTBITS (オプション)	暗号化レベル
ネットワーク・リンクを確立するときに実行できる最高の暗号化レベル	MAXENCRYPTBITS (オプション)	暗号化レベル
このネットワーク・エントリに関連付けられたネットワーク・グループ指定しない場合、デフォルト値は DEFAULTNET と見なされます。DEFAULTNET が設定されない場合、NETGROUPS セクションで指定したグループ名をこのパラメータに定義する必要があります。	NETGROUP (オプション)	ネットワーク・グループ

NETWORK セクションで指定する情報	パラメータ (必須 / オプション)	参照先 (クリックするとリンク先にジャンプ)
LMID が指定するノード上でネットワークにサービスを提供する tlisten(1) プロセス用のネットワーク・アドレス	NLSADDR (オプション)	tlisten ネットワーク・アドレス

NETWORK セクションの例

以下は、2 サイトを含むコンフィギュレーション・ファイルの NETWORK セクションの例です。

```
   *NETWORK
       SITE1   NADDR="//mach1:80952"
               NLSADDR="//mach1:serve"
#      SITE2   NADDR="//mach386:80952"
               NLSADDR="//mach386:serve"
```

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5)
- 3-71 ページの「コンフィギュレーション・ファイルの NETGROUPS セクションの作成方法」

BRIDGE プロセス用のデバイス名を指定する

LMID 上に置かれた BRIDGE プロセスで使用するデバイス名を指定し、ネットワークにアクセスするには、BRIDGE パラメータを次のように設定します。

BRIDGE=*string_value*

TCP/IP を使用する場合は、BRIDGE にデバイス名を指定する必要はありません。

ネットワーク・トランスポートのエンドポイント・ファイルのパス名は、次の形式で指定します。

/dev/*provider_name*

BRIDGE のネットワーク・アドレスを割り当てる

LMID の接続指示受け付けアドレスとして、LMID 上に置かれている BRIDGE プロセスで使用する完全なネットワーク・アドレスを指定するには、NADDR パラメータを次のように設定します。

NADDR = *string_value*

BRIDGE の接続指示受け付けアドレスは、アプリケーションの別の BRIDGE プロセスがその BRIDGE プロセスにアクセスするロケーションです。

BRIDGE の接続指示受け付けアドレスは、次の 3 つのうち、いずれかの形式で指定できます。

- *//host.name:port_number*
- *//#.#.#.#:port_number*
- *0xhex-digits* または *xhex-digits*

最初の形式では、*host.name* は、アドレスがバインドされるときに TCP/IP のアドレスに解決されます。この形式は、オペレーティング・システムのコマンドによる、ローカル名の解決機能に基づいています。*port_number* には、シンボリック名または 10 進数を指定します。

2 番目の形式の文字列 *#.#.#.#* は、ドット区切りの 4 つの 10 進数 (0 〜 255 を指定) を示します。*port_number* は、0 〜 65,535 の 10 進数です (指定された文字列の 16 進表現)。*port_number* には、シンボリック名または 10 進数を指定します。

3 番目の形式の文字列 *0xhex-digits* または *xhex-digits* には、有効な 16 進数の偶数を含めます。2 つのうち、どちらかの形式で表現された文字列は、内部で TCP/IP アドレスを含む文字配列に変換されます。

注記　プラットフォームによっては、システム側で、上記より小さい数値が予約されている場合があります。

暗号化レベルを割り当てる

このマシンに対してネットワーク・リンクを確立するときに必要な暗号化の最低レベルを設定するには、MINENCRYPTBITS パラメータを設定します。有効な値は、0、56、および 128 です。0 は暗号化を行わないことを示し、56 および 128 は暗号化キーの長さをビット単位で指定します。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 0 です。

ネットワーク・リンクを確立するときの暗号化の最大レベルを設定するには、MAXENCRYPTBITS パラメータを設定します。有効な値は、0、56、および 128 です。0 は暗号化を行わないことを示し、56 および 128 は暗号化キーの長さをビット単位で指定します。デフォルト値は 128 です。

例

```
MAXENCRYPTBITS=128
MINENCRYPTBITS=0
```

関連項目

- 『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』の「リンク・レベルの暗号化」

tlisten のネットワーク・アドレスを割り当てる

LMID が指定するマシン上でネットワークにサービスを提供する tlisten(1) プロセス用のネットワーク・アドレスを指定するには、NLSADDR パラメータを次のように指定します。

NLSADDR=*string_value*

string には、NADDR パラメータと同じ形式でネットワーク・アドレスを指定します。

NLSADDR の tlisten アドレスは、次の 3 つのうち、いずれかの形式で指定できます。

- *//host.name:port_number*
- *//#.#.#.#:port_number*
- *0xhex-digits* または *xhex-digits*

最初の形式では、*host.name* は、アドレスがバインドされるときに TCP/IP のアドレスに解決されます。この形式は、オペレーティング・システムのコマンドによる、ローカル名の解決機能に基づいています。*port_number* には、シンボリック名または 10 進数を指定します。

2 番目の形式の文字列 *#.#.#.#* は、ドット区切りの 4 つの 10 進数 (0 ～ 255 を指定) を示します。*port_number* は、0 ～ 65,535 の 10 進数です (指定された文字列の 16 進表現)。*port_number* には、シンボリック名または 10 進数を指定します。

3 番目の形式の文字列 `0xhex-digits` または `\\xhex-digits` には、有効な 16 進数の偶数を含めます。2 つのうち、どちらかの形式で表現された文字列は、内部で TCP/IP アドレスを含む文字配列に変換されます。

`NLSADDR` が `MASTER LMID` 以外のマシンのエントリにない場合、`tmloadcf(1)` はエラーを出力します。`NLSADDR` が `MASTER LMID` エントリにない場合、`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、読み取り専用の操作だけです。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。

コンフィギュレーション・ファイルの NETGROUPS セクションの作成方法

UBBCONFIG ファイルの NETGROUPS セクションでは、LAN 環境で使用可能なネットワーク・グループの情報を設定します。1 組のマシンに割り当てるネットワーク・グループの数に制限はありません。ネットワーク・グループを構成する各ネットワークの通信方式は、優先順位のメカニズム (NETPRIO) によって決まります。

すべての LMID は、デフォルトのネットワーク・グループ (DEFAULTNET) のメンバでなければなりません。デフォルトのネットワーク・グループのグループ番号 (NETGRPNO) は、0 です。ただし、DEFAULTNET に設定されているデフォルトの優先順位は変更できます。リリース 6.4 より前の BEA Tuxedo システムで定義されているネットワークは、DEFAULTNET ネットワーク・グループに割り当てられています。

次の表では、NETGROUPS セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

NETGROUPS **セクションで指定する情報 (オプション)**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
デフォルトのネットグループ数 (8) 以上のグループを定義します。この値は、RESOURCES セクションで指定されています。	MAXNETGROUPS (オプション)	ネットグループの最大数
ネットワークが使用可能になるのを待つデータの最大サイズ。この値は、MACHINES セクションで指定されています。	MAXPENDINGBYTES (オプション)	メッセージ領域の制限
このネットワーク・エントリに関連付けられたネットワーク・グループ	NETGROUP (必須)	ネットワーク・グループ名
フェイルオーバーおよびフェイルバック用に割り当てる必要のある一意のネットワーク・グループ番号	NETGRPNO (必須)	ネットワーク・グループ番号

NETGROUPS **セクションで指定する情報 (オプション)**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
このネットワーク・グループの優先順位	NETPRIO (オプション)	ネットワーク・グループの優先順位

ネットワーク・グループ構成の例

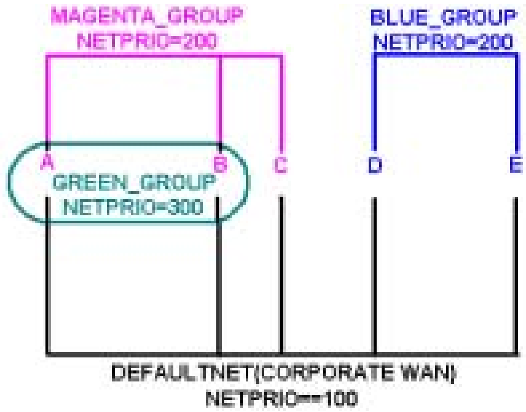
ネットワーク・アドレスは、ネットワーク・グループに関連付けられます。この機能の例を以下に示します。

たとえば、First State Bank には 5 台のマシン (A ～ E) で構成されるネットワークがあるとします。各マシンは、以下のように定義した 4 つのネットグループの中の 2 つまたは 3 つに属します。

- DEFAULTNET (デフォルトのネットワークである企業 WAN)
- MAGENTA_GROUP (LAN)
- BLUE_GROUP (LAN)
- GREEN_GROUP (メンバのマシン間に、高速な光ファイバのポイント・ツー・ポイント・リンクを提供する専用 LAN)

すべてのマシンは、DEFAULTNET (企業 WAN) に属します。さらに、各マシンは MAGENTA_GROUP または BLUE_GROUP に関連付けられます。MAGENTA_GROUP LAN の一部のマシンは、GREEN_GROUP という特殊なグループに属しています。次の図は、ネットワーク内のネットワーク・アドレスを持つマシン A ～ E を示します。

図 3-3 ネットワークのグループ化の例



次の表は、マシンと、対応するグループのアドレスを示します。

マシン	マシンが属するグループのアドレス
A および B	DEFAULTNET (企業 WAN) MAGENTA_GROUP (LAN) GREEN_GROUP (LAN)
C	DEFAULTNET (企業 WAN) MAGENTA_GROUP (LAN)
D および E	DEFAULTNET (企業 WAN) BLUE_GROUP (LAN)

注記　ローカル・エリア・ネットワークはロケーション間でルーティングされないため、マシン D (BLUE_GROUP LAN 内) は、共通の単一アドレス、つまり企業 WAN ネットワーク・アドレスのみを使用してマシン A (GREEN_GROUP LAN 内) と通信します。

UBBCONFIG ファイルにネットグループを設定する

前の節で説明したコンフィギュレーションを設定するには、First State Bank のシステム管理者は、次のコンフィギュレーション・ファイルの例に示すように、UBBCONFIG ファイルの NETGROUPS セクションで各グループを定義します。

コード リスト 3-1　NETGROUPS **および** NETWORK **セクションの例**

```
*NETGROUPS

DEFAULTNET    NETGRPNO = 0          NETPRIO = 100 #default
BLUE_GROUP    NETGRPNO = 9          NETPRIO = 200
MAGENTA_GROUP NETGRPNO = 125        NETPRIO = 200
GREEN_GROUP   NETGRPNO = 13         NETPRIO = 300
```

```
*NETWORK

A      NETGROUP=DEFAULTNET         NADDR="//A_CORPORATE:5723”
A      NETGROUP=MAGENTA_GROUP      NADDR="//A_MAGENTA:5724"
A      NETGROUP=GREEN_GROUP        NADDR="//A_GREEN:5725"

B      NETGROUP=DEFAULTNET         NADDR="//B_CORPORATE:5723"
B      NETGROUP=MAGENTA_GROUP      NADDR="//B_MAGENTA:5724"
B      NETGROUP=GREEN_GROUP        NADDR="//B_GREEN:5725"

C      NETGROUP=DEFAULTNET         NADDR="//C_CORPORATE:5723"
C      NETGROUP=MAGENTA_GROUP      NADDR="//C_MAGENTA:5724"

D      NETGROUP=DEFAULTNET         NADDR="//D_CORPORATE:5723"
D      NETGROUP=BLUE_GROUP         NADDR="//D_BLUE:5726"
E      NETGROUP=DEFAULTNET         NADDR="//E_CORPORATE:5723"
E      NETGROUP=BLUE_GROUP         NADDR="//E_BLUE:5726"
```

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5)
- 3-77 ページの「コンフィギュレーション・ファイルの SERVERS セクションの作成方法」
- 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」

ネットワーク・グループに名前を割り当てる

ネットワーク・グループに名前を割り当てるには、NETGROUP パラメータを次のように設定します。

NETGROUP *required_parameters* [*optional_parameters*]

NETGROUP に DEFAULTNET を設定すると、エントリには、デフォルトのネットワーク・グループに関する情報が指定されます。NETGROUP パラメータが DEFAULTNET に設定されているネットワーク・エントリは、すべて TM_MIB の T_MACHINE クラスで表されますが、ほかの NETGROUP に関連付けられた NETWORK エントリは、TM_MIB の T_NETMAP クラスで表され、以前のリリースとの相互運用が可能です。

ネットワーク・グループ番号を割り当てる

フェイルオーバーおよびフェイルバックに対応するには、NETGRPNO パラメータを次のように設定します。

NETGRPNO=*numeric_value*

このエントリが DEFAULTNET を説明している場合、NETGRPNO の値はゼロにする必要があります。

ネットワーク・グループに優先順位を割り当てる

複数のネットワーク・グループにある、同じ優先順位が指定された 1 組のマシンは、優先順位が最も高い回線上で同時に通信できます。ネットワーク・グループの優先順位を割り当てるには、NETPRIO パラメータを使用します。特定の優先順位を持つすべてのネットワーク回線が、管理者またはネットワークによって切断された場合は、1 つ下の優先順位の回線が使用されます。優先順位の高い回線に対しては、接続の再試行が行われます。NETPRIO パラメータには、0 より大きく 8,192 未満の値を指定します。デフォルト値は 100 です。

コンフィギュレーション・ファイルの SERVERS セクションの作成方法

コンフィギュレーション・ファイルの SERVERS セクションには、サーバ・プロセスに固有な情報が含まれています。このセクションは必須ではありません。ただし、このセクションがないということは、アプリケーション・サーバがなく、機能もほとんどないことを意味します。このセクションの各エントリは、アプリケーションで起動するサーバ・プロセスを表し、次の情報を指定します。

- サーバ名、サーバ・グループ、および数値で指定するサーバ識別子 (SRVGRP、SRVID)
- servopts で定義するサーバのコマンド行オプション (CLOPT)
- サーバの起動順序および起動するサーバ数 (SEQUENCE、MIN、MAX)
- サーバ固有の環境ファイル (ENVFILE)
- サーバのキューに関する情報 (RQADDR、RQPERM、REPLYQ、RPPERM)
- 再起動に関する情報 (RESTART、RCMD、MAXGEN、GRACE)
- 会話型サーバとするかどうかの指定 (CONV)
- システム全体にわたる共用メモリ・アクセスを上書きするかどうかの指定 (SYSTEM_ACCESS)
- IIOP リスナ (ISL) サーバのセキュリティ・パラメータの設定

注記　BEA Tuxedo システムでサポートされるコマンド行オプションについては、『BEA Tuxedo のファイル形式とデータ記述方法』の servopts(5) を参照してください。

次の表では、SERVERS セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

SERVERS セクションで指定する情報 (オプション)	パラメータ (必須 / オプション)	参照先 (クリックするとリンク先にジャンプ)
サーバが会話型サーバであるかどうか。接続は、会話型サーバに対してのみ行うことができます。tpacall(3c) または tpcall(3c) を使った rpc 要求は、非会話型サーバに対してのみ行うことができます。	CONV (オプションの実行時パラメータ)	会話型サーバ
ID 確認のためのプロセスのプリンシパル名、プリンシパル・ユーザの秘密鍵の場所、およびパスワードを格納する環境変数	SEC_PRINCIPAL_NAME、SEC_PRINCIPAL_LOCATION、SEC_PRINCIPAL_PASSVAR	セキュリティ属性
ほかのサーバとの関連で、このサーバをいつ起動またはシャットダウンするか	SEQUENCE (オプションのブート・パラメータ)	サーバの起動順序
tmboot によって起動されるサーバのオカレンスの最小数	MIN (オプションのブート・パラメータ)	サーバの起動順序
起動できるサーバのオカレンスの最大数。	MAX (オプションのブート・パラメータ)	サーバの起動順序
起動時にサーバ・プロセスに渡される servopts(5) オプションのリスト。何も指定しない場合、デフォルトの -A が使用されます。*string_value* は 256 文字まで指定できます。	CLOPT (オプションのブート・パラメータ)	サーバのコマンド行オプション
初期化時に、該当するファイルの値をサーバ環境に追加する要求。サーバが、別のマシンに移行可能なサーバ・グループに関連付けられている場合は、移行元マシンと移行先マシンの同じ場所に ENVFILE を格納する必要があります。	ENVFILE (オプションの実行時パラメータ)	サーバ環境ファイル

SERVERS **セクションで指定する情報 (オプション)**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
サーバが所属するサーバ・グループの名前。*string_value* には、GROUPS セクションのサーバ・グループに関連付けられた論理名を指定します。	SRVGRP (必須)	サーバ・グループ
サーバ・グループ内のサーバを一意に識別する整数。識別子は、1 以上 30,000 以下でなければなりません。	SRVID (必須)	サーバ ID
プロセスの要求キューのシンボリック名	RQADDR (オプションの実行時パラメータ)	サーバ・キュー情報
要求キューに対するパーミッション (数値で指定)	RQPERM (オプションの実行時パラメータ)	サーバ・キュー情報
プロセスに対して応答キューを確立するかどうか	REPLYQ (オプションの実行時パラメータ)	サーバ・キュー情報
応答キューに対するパーミッション (数値で指定)	RPPERM (オプションの実行時パラメータ)	サーバ・キュー情報
再起動可能なプロセスが異常終了した場合に実行するコマンド	RCMD (オプションの実行時パラメータ)	サーバの再起動に関する情報
再起動可能なプロセスの再起動回数。GRACE で指定された時間内に再起動できる最大回数から 1 を引いた数を指定します。	MAXGEN (オプションの実行時パラメータ)	サーバの再起動に関する情報
再起動可能なプロセスの再起動回数。指定した秒内で MAXGEN 回まで再起動できることを指定します。	GRACE (オプションの実行時パラメータ)	サーバの再起動に関する情報
プロセスを再起動できるかどうか。デフォルト値は N です。サーバを移行できる場合は、RESTART を Y に設定します。SIGTERM シグナルで終了されたサーバは再起動する必要があります。	RESTART (オプションの実行時パラメータ)	サーバの再起動に関する情報

SERVERS **セクションで指定する情報** (**オプション**)	**パラメータ** (**必須** / **オプション**)	**参照先** (**クリックするとリンク先にジャンプ**)
アプリケーション・プロセス内で、BEA Tuxedo のシステム・ライブラリから内部テーブルへのアクセスを実現するデフォルト・モード。	SYSTEM_ACCESS (オプションの実行時パラメータ)	サーバへのシステム・アクセス
最初にサーバを起動するときのサーバ・ディスパッチ・スレッドの最小数。MAXDISPATCHTHREADS が 1 より小さい場合 (MAXDISPATCHTHREADS>1) は、別のディスパッチ・スレッドが使用されます。このディスパッチ・スレッドは、MAXDISPATCHTHREADS で指定した数には含まれません。MAXDISPATCHTHREADS は、MINDISPATCHTHREADS と同じか、またはそれ以上でなければなりません (MINDISPATCHTHREADS<= MAXDISPATCHTHREADS)。このパラメータのデフォルト値は 0 です。	MINDISPATCHTHREADS	スレッド
各サーバ・プロセスで生成される、同時に実行できるディスパッチ・スレッドの最大数 MAXDISPATCHTHREADS が 1 より大きい場合 (MAXDISPATCHTHREADS>1) は、別のディスパッチ・スレッドが使用されます。このディスパッチ・スレッドは、パラメータで指定した数には含まれません。MAXDISPATCHTHREADS は、MINDISPATCHTHREADS と同じか、またはそれ以上でなければなりません (MINDISPATCHTHREADS<= MAXDISPATCHTHREADS)。このパラメータのデフォルト値は 1 です。	MAXDISPATCHTHREADS	スレッド

SERVERS **セクションで指定する情報** (**オプション**)	**パラメータ** (**必須** / **オプション**)	**参照先** (**クリックするとリンク先にジャンプ**)
最初のスレッド以降の各サーバ・スレッドのスタック・サイズ (バイト単位)。このパラメータを指定しないか、または 0 の場合は、オペレーティング・システムのデフォルト値が使用されます。このオプションは、MAXDISPATCHTHREADS に 1 より大きい値が指定された場合のみサーバに影響を与えます。	THREADSTACKSIZE	スレッド
WebLogic Server の組み込み LDAP ベース認証サーバ。System /T で用意されたサーバであり、ユーザのセキュリティ情報が WebLogic Server にあるときに認証サービスを提供します。このサーバは、クライアントがアプリケーションに参加するときにユーザ単位の認証を提供するために、セキュリティ付きアプリケーションで使用されます。 SECURITY USER_AUTH またはそれ以上を指定する必要があります。 デフォルトでは $TUXDIR/udataobj/tpldap ファイルを使用して、LDAP のコンフィギュレーション情報を取得します。 例 : LAUTHSVR SRVGRP= “AUTH’’SRVID=100 CLOPT=”-A-- -f/usr/tuxedo/udataobj/tpldap”	LAUTHSVR (オプション)	LAUTHSVR

SERVERS セクションの例

以下は、コンフィギュレーション・ファイルの SERVERS セクションの例です。

```
*SERVERS
DEFAULT:          RESTART=Y MAXGEN=5 GRACE=3600
                  REPLYQ=N CLOPT=”-A”
                  ENVFILE=”/usr/home/envfile”
                  SYSTEM_ACCESS=PROTECTED

RINGUP1           SRVGRP=GROUP1 SRVID=1 MIN=3
                  RQADDR=”ring1"
RINGUP2           SRVGRP=GROUP1 SRVID=4 MIN =3
                  RQADDR=”ring2"
```

注記　この例では、SEQUENCE (1 〜 6 の順に起動)、REPLYQ と RPPERM (サーバは応答を受け取らない)、RCMD (再起動時のコマンド不要)、および CONV (非会話型サーバ) が省略されています。特定のサーバに対して特に設定を行わない限り、すべてのサーバにはデフォルト設定が適用されます。

SERVERS セクションのパラメータ

上の SERVERS セクションの例では、以下のパラメータと値が指定されています。

パラメータ	説明
RESTART=Y (デフォルト)	サーバを再起動します。
MAXGEN=5 (デフォルト)	MAXGEN パラメータには、GRACE パラメータで指定した期間内にサーバが起動できる回数を制御する、0 より大きく 256 よりも小さい数値を指定します。デフォルト値は 1 です。サーバが再起動可能な場合、このパラメータには 2 以上 (MAXGEN >= 2) を指定します。再起動は、指定した値から 1 を引いた回数行うことができます。RESTART は、Y に指定します。指定しないと、MAXGEN は無視されます。

パラメータ	説明
GRACE=3600 (デフォルト)	RESTART が Y の場合、GRACE パラメータを使用して、サーバの再起動を行える期間 (秒単位) を指定することができます。再起動は、MAXGEN - 1 回行うことができます。秒数として 0 以上 2,147,483,648 未満 (または 68 年強) の値を指定します。GRACE を指定しない場合は、デフォルトの 86,400 秒 (24 時間) が指定されます。GRACE 期間が終了すると、次の期間が開始します。このパラメータに 0 を設定すると、すべての制限が解除されます。つまり、サーバの再起動回数が制限されなくなります。
REPLYQ=N (デフォルト)	応答キューはありません。
CLOPT=”-A” (デフォルト)	各サーバのコマンド行に -A を指定します。
ENVFILE=”/usr/home/envfile” (デフォルト)	ENVFILE に指定したファイルから環境設定を読み取ります。
SYSTEM_ACCESS=PROTECTED (デフォルト)	システム・コード外から内部テーブルへのアクセスを拒否します。
RINGUP1	最初に起動するサーバのサンプル名
SRVGRP=GROUP1 SRVID=1 MIN=3 RQADDR=”ring1"	サンプル・サーバの 3 つのインスタンスが、サーバ・グループ GROUP1 で起動します。サーバ ID は、1、2、および 3 です。3 つのサーバは、MSSQ セットを構成し、キュー *ring1* からの要求を読み取ります。 注記　RQADDR は、このサーバの要求キューにシンボリック名を割り当てます。MSSQ セットを作成するには、複数のサーバに対して同じシンボリック・キュー名を指定し、これらのサーバの実行可能ファイルに同じ名前を割り当てます。MIN には、1 より大きい値を指定します。
RINGUP2	2 番目に起動するサンプル・サーバの名前

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5)
- 3-101 ページの「コンフィギュレーション・ファイルの SERVICES セクションの作成方法」

サーバを会話型として指定する

サーバが会話型サーバである (つまり、クライアントと専用サーバの間に双方向接続を確立する) 場合、CONV は必須パラメータであり Y を設定する必要があります。デフォルト値は N で、この場合はそのサーバが会話に含まれないことを示します。

CONV パラメータの特性

CONV パラメータには、以下の特性があります。

- Y は、サーバが会話型であることを示します。N は、サーバが会話型ではないことを示します。
- サーバで会話型の要求を受け取る場合は、Y を設定する必要があります。
- デフォルトは N です。

サーバを起動する順序を設定する

サーバを起動する順序を指定するには、各サーバの SEQUENCE パラメータを設定します。SEQUENCE には、1 から 30,000 までの任意の数字を指定します。SEQUENCE パラメータの値が小さいサーバは、値が大きいサーバより先に起

動します。SEQUENCE パラメータをまったく指定しない場合、サーバは SERVERS セクションにリストされている順序で起動します。順序が指定されたサーバと指定されていないサーバが混在する場合は、順序が指定されたサーバが先に起動します。サーバは、起動する順序とは逆の順序でシャットダウンされます。

SEQUENCE は、オプション・パラメータです。このパラメータは、大規模なアプリケーションで、サーバを起動する順序を指定する必要がある場合に役立ちます。

警告 : CORBA 環境では、システムのイベント・ブローカ、FactoryFinder オブジェクト、およびアプリケーション ファクトリを起動する順序が厳密に規定されています。この順序に従わないと、CORBA アプリケーションは起動しません。詳細については、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。

複数のサーバを起動するには、MIN パラメータを設定して、起動作業を簡略化できます。この場合、すべてのサーバに同じオプションが適用されます。RQADDR を指定すると、サーバは MSSQ セットを生成します。MIN のデフォルト値は 1 です。

起動するサーバの最大数を指定するには、MAX パラメータを設定します。tmboot(1) コマンドを実行すると、実行時に MIN で指定した数のサーバが起動します。続いて、MAX で指定した数までのサーバが起動します。デフォルトは、MIN で指定した数のサーバです。

MIN および MAX パラメータは、大規模なアプリケーションのコンフィギュレーション・ファイルを管理しやすいサイズに保つ上で役立ちます。MAX の最大値は、IPC 資源を割り当てて設定します。MIN および MAX パラメータは、会話型サービスやサーバの自動生成でも使用されます。

CORBA C++ サーバの起動順序

BEA Tuxedo CORBA 環境でのサーバの正しい起動順序は以下のとおりです。この順序に従わないと、CORBA アプリケーションは起動しません。

1. システムのイベント・ブローカである `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 (マスタとして) 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` オプションのみが設定された `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 オブジェクトを起動します。
5. ファクトリを宣言するアプリケーション C++ サーバ。

コード リスト 3-2 は、BEA Tuxedo ソフトウェアに収録されているサンプル・アプリケーションの 1 つである、BEA Tuxedo CORBA University Basic アプリケーションの起動順序を示します。この `SERVERS` セクションは、`ubb_b.nt` コンフィギュレーション・ファイルを編集したものから抜粋しています。

コード リスト 3-2 University 用の UBBCONFIG サンプル・ファイルの SERVERS セクションを編集した例

```
*SERVERS
    # デフォルトでは、サーバがクラッシュした場合、24 時間以内に最大 5 回まで
    # 再起動します。
#
    DEFAULT:
        RESTART = Y
        MAXGEN  = 5

    # BEA Tuxedo System のイベント・ブローカを起動します。このイベント・ブ
ローカは、
    # NameManager サービスを提供するサーバを起動する前に
    # 起動しておかなければなりません。
    #
    TMSYSEVT
        SRVGRP  = SYS_GRP
        SRVID   = 1

    # TMFFNAME は、BEA Tuxedo CORBA 環境で提供される、
```

```
# NameManager および FactoryFinder サービスを実行するサーバです。

# NameManager は、BEA Tuxedo CORBA 環境固有のサービスで、
 # アプリケーションが提供する名前とオブジェクト・リファレンスの
# マッピングを管理します。

# NameManager Service (-N オプション ) を起動します。この NameManager
は
# マスタとして起動されます (-M オプション )。
#
TMFFNAME
    SRVGRP  = SYS_GRP
    SRVID   = 2
    CLOPT   = "-A -- -N -M"

# スレーブ NameManager サービスを起動します。
#
TMFFNAME
    SRVGRP  = SYS_GRP
    SRVID   = 3
    CLOPT   = "-A -- -N"

# FactoryFinder (-F) サービスを起動します。
#
TMFFNAME
    SRVGRP  = SYS_GRP
    SRVID   = 4
    CLOPT   = "-A -- -F"

# インターフェイス・リポジトリ・サーバを起動します。
#
TMIFRSVR
    SRVGRP  = SYS_GRP
    SRVID   = 5

# University サーバを起動します。
#
univb_server
    SRVGRP  = ORA_GRP
    SRVID   = 6
    RESTART = N



# IIOP クライアントのリスナを起動します。
#
# サーバ・マシンのホスト名とポート番号を
```

```
# 指定します。通常、ポート番号は 2500 です。
#
ISL
    SRVGRP  = SYS_GRP
    SRVID   = 7
    CLOPT   = "-A -- -n //TRIXIE:2500"
```

この例では、TMSYSEVT サーバと TMFFNAME サーバを起動した後、以下のコンポーネントが起動します。

- インターフェイス・リポジトリ。この機能とコマンド行オプション (CLOPT パラメータ) については、第 6 章「CORBA インターフェイス・リポジトリの管理」を参照してください。
- University Basic サンプル・アプリケーションの univb_server。サンプル・アプリケーションの詳細については、*『BEA Tuxedo CORBA University サンプル・アプリケーション』*を参照してください。
- インターネット ORB 間プロトコル (IIOP) サーバ・リスナ (ISL)。この機能と CLOPT パラメータについては、第 12 章「BEA Tuxedo CORBA リモート・クライアント・アプリケーションの管理」を参照してください。

注記　何らかの理由でグループやマシンを移行またはシャットダウンするとき、アクティブなスレーブ NameManager が別のグループにある場合は、マスタ NameManager が アクティブになる前に FactoryFinder やスレーブ NameManager が再起動されないように、UBBCONFIG ファイルを設定してください。たとえば、FactoryFinder がマスタ NameManager と同じグループにある場合、マスタ NameManager が先に起動されるように、UBBCONFIG ファイルでサーバの起動順序を変更します。

SEQUENCE、MIN、および MAX パラメータの特性

パラメータ	説明
SEQUENCE	オプション・パラメータです。値には 1 ～ 10,000 の範囲の数値を設定します。 小さい値が指定されたサーバは、大きい値が指定されたサーバより先に起動します。 このパラメータが設定されていないサーバは、SERVERS セクションにリストされている順序で起動します。 順序が指定されているサーバは、指定されていないサーバより先に起動します。
MIN	実行時に起動するサーバの最小数を指定します。 RQADDR が指定されており、MIN が 1 より大きい場合 (MIN>1) は、MSSQ セットが作成されます。 すべてのインスタンスには、同じサーバ・オプションが適用されます。 値には、0 ～ 1000 を指定します。 デフォルト値は 1 です。
MAX	起動するサーバの最大数を指定します。 MAX には 0 ～ 1000 の値を指定します。MAX を指定しない場合、デフォルトで MIN の値が指定されます。

サーバのコマンド行オプションを指定する

BEA Tuxedo システムでは、サーバが要求を処理するときに使用するオプションを指定できます。これらのオプションは、servopts で定義します。これを実行すると、サーバ・プロセス用の実行時オプションがリストされます。サーバ側では、コマンド行からの情報の取得が必要な場合があります。CLOPT パラメータを使用すると、コマンド行オプションを指定して、サーバに設定されたデフォルト値を変更したり、ユーザ定義のオプションを tpsvrinit() 関数に渡すことができます。

標準の main() を使用すると、引数 -- までのオプションのセットが解析され、以降のオプションは tpsvrinit() に渡されます。CLOPT のデフォルトのオプション -A は、buildserver(1) または buildobjserver(1) によって組み込まれたすべてのサービスを宣言するようサーバに通知します。次の表は、使用可能なオプションの一部です。

オプション	目的
-o *filename*	標準出力を *filename* ファイルにリダイレクトします。
-e *filename*	標準エラーを *filename* ファイルにリダイレクトします。
-s *services*	サービスを宣言します。たとえば、-s x,y,z は、サービス x、y、および z を宣言します。
-s x,y,z:*funcname*	サービス x、y、および z を宣言しますが、関数 *funcname* を使用してこれらのサービスの要求を処理します。これは、関数名のエイリアスと呼ばれます。
-r	実行したサービスがログに記録されるようにサーバを指定します。
-v	標準出力にサービス名と関数名のリストを出力します。 このオプションは、UBBCONFIG の CLOPT では使用できません。これは、サーバを手動で起動するときに使用します。

注記　その他の標準的な main() オプションについては、『BEA Tuxedo のファイル形式とデータ記述方法』の servopts(5) を参照してください。

CLOPT パラメータの特性

- 構文は、CLOPT="*servopts* -- *application_opts*" です。
- オプション・パラメータであり、デフォルトで -A が設定されます
- main() および tpsvrinit() では、サーバのコマンド行オプションが使用されます。
- servopts(5) オプションは main() に渡されます。
- アプリケーションのオプションは tpsvrinit() に渡されます。

サンプルの BANKAPP アプリケーションでは、次のようにコマンド行オプションを指定しています。

```
CLOPT="-A -- -T 10"
```

これは、サーバに対し、すべてのサービスを宣言するオプション (-A) と 10 という窓口 ID が指定されていることを示します。オペレーションが発生すると、特定の窓口のレコードは更新されます。このオプション、特に tpsvrinit() に渡されるオプションを使用する場合、システム管理者とアプリケーション・プログラマは、よく相談する必要があります。

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の servopts(5)

サーバ環境ファイルのロケーションを識別する

環境設定を指定するには、MACHINES セクションにある ENVFILE パラメータを使用します。また、特定のサーバ・プロセスに同じパラメータを指定できますが、セマンティクスは同じです。MACHINES セクションの ENVFILE と、SERVERS セクションの ENVFILE が指定されると、両方とも有効になります。MACHINES および SERVERS セクションの両方に同じ変数が定義された場合は、SERVERS セクションの設定が優先されます。

サーバ環境ファイルの特性

ENVFILE は、サーバ環境ファイルを定義するパラメータであり、以下の特性があります。

- オプション・パラメータであり、MACHINES セクションの ENVFILE パラメータと同じセマンティクスを含みます。ただし、定義するのは 1 つのサーバだけです。
- 変数が重複する場合、ENVFILE の SERVERS セクションの設定が、ENVFILE の MACHINES セクションと GROUPS セクションの設定を上書きします。

環境変数の設定の詳細については、『BEA Tuxedo のファイル形式とデータ記述方法』の tuxenv(5) を参照してください。

サーバ名、サーバ・グループ、および サーバ ID を定義する

まず、SERVERS セクションでサーバに名前を割り当てます。指定する名前は、以下のいずれかのコマンドを使用して作成した実行可能ファイルです。

- buildserver(1) (ATMI アプリケーションの場合)
- buildobjserver(1) (CORBA C++ サーバ・アプリケーションの場合)

また、各サーバにグループ識別子 (SRVGRP) を指定する必要があります。SRVGRP には、GROUPS セクションのエントリの最初に指定した名前を設定します。最後に、指定されたグループ内の各サーバ・プロセスに一意の数値識別子 (SRVID) を指定します。サーバ・エントリには、必ず SRVGRP パラメータと SRVID パラメータが必要です。エントリでは、アプリケーションのほか、起動するマシンも記述されるため、複数のエントリに同じサーバ名が表示される場合もあります。

サーバ名、SRVGRP、および SRVID パラメータの特性

パラメータ	説明
Server_name	起動する実行可能ファイルを識別します。 ATMI では buildserver(1) で作成されます。 CORBA では buildobjserver(1) で作成されます。 必須パラメータですが、サーバ・グループ内で一意でなくてもかまいません。

パラメータ	説明
SRVGRP (サーバ・グループ)	グループとの関係を識別します。 グループ名は、GROUPS セクションのエントリに設定されているものを指定します。 必須パラメータです。
SRVID (サーバ ID)	数値で指定します。 必須パラメータであり、サーバ・グループ内で一意な値を設定します。

サーバ・キュー情報を識別する

サーバ・キュー情報を指定すると、サーバ・メッセージ・キューを作成し、このキューへのアクセスを制御できます。BEA Tuxedo システムでは、RQADDR パラメータを使用して複数サーバ、単一キュー (MSSQ) セットを作成できます。どのサーバに対しても、このパラメータの値に英数字を設定できます。同じサービスを提供するすべてのサーバに対して同じ RQADDR 値を設定すると、サービスを 1 つのメッセージ・キューにまとめて MSSQ セットを作成することができ、ロード・バランシングを実現できます。

MSSQ の例

MSSQ セットは、銀行の窓口担当者の業務に似ています。たとえば、窓口が 4 つあり、顧客は 1 列に並んで順番にサービスを受けるとします。顧客は、空いた窓口へ順番に進みます。当然ながら、融資担当の窓口は別です。融資担当の窓口では、預金の預け入れや引き出しを処理できません。また、融資を希望する顧客は限られています。同様に、MSSQ セットの場合も、提供するサービスが異なるサーバは、同じ MSSQ セットに参加できません。

RQPERM パラメータを使用すると、UNIX システムの規則に従った、サーバ要求キューのパーミッションを指定できます (0666 など)。この設定により、要求キューへのアクセスを制御できます。

MSSQ サーバ内のサービス・ルーチンでサービス要求が発行されると、それらの要求に対する応答を応答キューで受け取る必要があります。このような応答キューは、REPLYQ=Y で指定することができます。デフォルトでは、REPLYQ に N が設定されています。REPLYQ が Y に設定されている場合、RPPERM パラメータを使ってその応答キューにパーミッションを割り当てることもできます。

RQADDR、RQPERM、REPLYQ、および RPPERM パラメータの特性

パラメータ	説明
RQADDR	MSSQ セットを作成できるようにする英数値です。値は、MSSQ セットのすべてのメンバに対して同じです。MSSQ のすべてのメンバは、同じサービスのセットを提供する必要があり、MSSQ セットのサーバは同じ実行可能ファイル名を持っていなければなりません。複数のサーバを起動するには、Min パラメータに 1 より大きい値を設定します。
RQPERM	要求キューのパーミッションを表します。パラメータを指定しない場合、RESOURCES セクションの PERM で指定された掲示板のパーミッションが使用されます。値が指定されない場合は、デフォルト値の 0666 が使用されます。デフォルトを使用すると、システム上にログインしたユーザがすべてアプリケーションにアクセスできるようになります。

パラメータ	説明
REPLYQ	このサーバに対して、応答キュー (要求キューとは別) を設定するかどうかを指定します。要求キューを使用しているサーバが 1 つだけの場合、応答は問題なく要求キューから受け取られます。BEA Tuxedo システムでは、サーバが MSSQ セットのメンバであり、サービスが応答メッセージを受け取るようにプログラミングされている場合は、REPLYQ を Y に設定して、このサーバに対して応答キューが個別に作成されるようにします。そうでない場合、応答は、MSSQ セット内の全サーバが共用する要求キューに送信されてしまい、応答が要求元のサーバに返されるかどうかは保証されません。マルチスレッド化されたサーバの場合は、このパラメータが設定されていなくても、自動的に REPLYQ が作成されます。
RPPERM	応答キューのパーミッションを割り当てます。このパラメータは、REPLYQ=Y の場合のみ有効です。要求と応答が同じキューから読み出される場合、必要なのは RQPERM のみで、RPPERM は無視されます。

サーバの再起動に関する情報を定義する

適切にデバッグされたサーバは、自動的に終了しません。デフォルトでは、アプリケーションの実行中に終了したサーバは、自動的に再起動しません。サーバを再起動する場合は､RESTART パラメータに Y を設定します。RESTART が Y の場合、RCMD、MAXGEN、または GRACE パラメータを設定できます。

RCMD パラメータを使用すると、サーバの再起動時に並行して実行されるコマンドを指定できます。たとえば、サーバ開発者や、サーバのアクティビティを監査する担当者に電子メールを送信する、という操作を実行できます。

MAXGEN パラメータは、GRACE で指定された期間内にサーバを再起動できる回数を示します。サーバは、GRACE で指定された秒数の間に MAXGEN-1 回だけ再起動できます。GRACE がゼロに設定されると、サーバの再起動は無制限に

行われます。MAXGEN はデフォルトで 1 に設定されており、256 より大きい値は設定できません。GRACE には、ゼロまたはゼロより大きい値を設定します。ただし、2,147,483,647 (2^{31} - 1) 以上の値は設定できません。

注記　完全にデバッグされたサーバを再起動する必要はありません。RESTART パラメータおよび関連するパラメータには、テスト用とプロダクション用の 2 つの値を設定する必要があります。

RESTART、RCMD、MAXGEN、および GRACE パラメータの特性

パラメータ	説明
RESTART	Y を設定すると、サーバを再起動できるようになります。 デフォルトは N です。
RCMD	再起動時に実行される実行可能ファイルを指定します。 サーバの再起動時にアクションを行えるようにします。
MAXGEN	指定された時間の間隔内でのサーバ再起動の最大数を表します。 デフォルト値は 1 で、最大値は 256 です。
GRACE	MAXGEN に使用される時間間隔を表します。 ゼロの場合は、起動回数に制限がないことを表します。 0 ～ 2,147,483,647 (2^{31} - 1) の範囲の値である必要があります。 デフォルト値は 24 時間です。

共用メモリに対するサーバ・アクセスを定義する

SYSTEM_ACCESS パラメータは、サーバ・プロセスを共用メモリにアタッチし、システム・コードの外から内部テーブルへのアクセスを許可するかどうかを決定します。アプリケーション開発時には、このようなアクセスを許可しないように設定しておく (PROTECTED) ことをお勧めします。アプリケーションをテストした後で、SYSTEM_ACCESS の値を FASTPATH に変更すると、性能を高めることができます。

このパラメータの設定は、RESOURCES セクションで NO_OVERRIDE が指定されない限り、RESOURCES セクションの値を上書きします。RESOURCES セクションで NO_OVERRIDE が指定されている場合、このパラメータは無視されます。NO_OVERRIDE 値は、このセクションでは使用されません。

SYSTEM_ACCESS パラメータの特性

SYSTEM_ACCESS パラメータには、以下の特性があります。

- PROTECTED は、サーバがシステム・コードの外から共用メモリにアタッチしないことを示します。
- FASTPATH は、サーバが常に共用メモリにアタッチされることを示します。
- NO_OVERRIDE セクションで RESOURCES が指定されている場合、このパラメータは無視されます。
- デフォルト値は、RESOURCES セクションの SYSTEM_ACCESS パラメータに指定した値です。
- 値 PROTECTED が設定されていると、BEA Tuxedo システムの処理速度は低くなります。

サーバ・ディスパッチ・スレッドを定義する

MAXDISPATCHTHREADS は、各サーバ・プロセスで生成される、同時に実行できるディスパッチ・スレッドの最大数です。MAXDISPATCHTHREADS が 1 より大きい場合 (MAXDISPATCHTHREADS>1) は、別のディスパッチ・スレッドが使用されます。このディスパッチ・スレッドは、パラメータで指定した数には含まれません。MAXDISPATCHTHREADS は、MINDISPATCHTHREADS と同じか、またはそれ以上でなければなりません (MINDISPATCHTHREADS<=MAXDISPATCHTHREADS)。指定しない場合、このパラメータのデフォルト値は 1 です。

MINDISPATCHTHREADS は、最初にサーバを起動するときのサーバ・ディスパッチ・スレッドの最小数です。MAXDISPATCHTHREADS が 1 より大きい場合 (MAXDISPATCHTHREADS>1) は、別のディスパッチ・スレッドが使用されます。このディスパッチ・スレッドは、MAXDISPATCHTHREADS で指定した数には含まれません。MAXDISPATCHTHREADS は、MINDISPATCHTHREADS と同じか、またはそれ以上でなければなりません (MINDISPATCHTHREADS<=MAXDISPATCHTHREADS)。このパラメータのデフォルト値は 0 です。

最初のスレッド以降の各サーバ・スレッドのスタック・サイズをバイト数で指定する必要があります。このパラメータを指定しないか、または 0 の場合は、オペレーティング・システムのデフォルト値が使用されます。このオプションは、MAXDISPATCHTHREADS に 1 より大きい値が指定された場合のみサーバに影響を与えます。

ISL サーバのセキュリティ・パラメータの設定

CORBA 環境では、IIOP リスナ (ISL) プロセスが、リモート・クライアントの接続要求をリッスンします。ISL プロセスは、BEA Tuxedo システム側で提供されるサーバとして、1 つのエントリで指定されます。

IIOP を使用する場合、Secure Socket Layer (SSL) プロトコルによってプロセス間の安全な通信方法を定義します。ISL コマンドの -s オプションを使用して、必須パラメータを設定します。これらのパラメータは、SSL プロトコルを使用する場合のみ設定する必要があります。SSL プロトコルは、BEA Tuxedo Security Pack にインストールされています。

表 0-1 は、SSL パラメータの特性を示します。

表 0-1ISL **および** SSL **パラメータの特性**

パラメータ	説明
`SEC_PRINCIPAL_NAME`	IIOP リスナ / ハンドラの ID を指定します。
`SEC_PRINCIPAL_LOCATION`	IIOP リスナ / ハンドラの秘密鍵の場所を指定します。
`SEC_PRINCIPAL_PASSWORD`	IIOP リスナ / ハンドラの秘密鍵の文字列を指定します。

これらのパラメータ設定の詳細については、『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』を参照してください。

コンフィギュレーション・ファイルの SERVICES セクションの作成方法

コンフィギュレーション・ファイルの SERVICES セクションでは、アプリケーションで提供されるサービスに関する詳しい情報を指定できます。トランザクションに関与しない非分散型のアプリケーションの場合、これらの情報の指定は簡単です。SERVICES セクションには、以下の種類の情報が含まれます。

- ロード・バランシング情報 (SRVGRP)
- サービスへの優先順位の割り当て
- 個々のサーバ・グループに対する異なるサービス・パラメータ
- バッファ・タイプの情報 (BUFTYPE)

サービスには必須パラメータはありません。サービスをリストする必要があるのは、オプション・パラメータを設定する場合のみです。

次の表では、SERVICES セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

SERVICES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
まだトランザクション・モードでない状態で要求のメッセージを受け取った場合に、トランザクションが自動的に開始されるかどうか	AUTORAN (DTP アプリケーションのみ)	トランザクションの自動開始
このサービスで受け付けるデータ・バッファのタイプおよびサブタイプのリスト。このパラメータには、最大 32 のタイプとサブタイプの組み合わせを最大 256 文字で指定できます。	BUFTYPE (オプション)	バッファ型
SVCNAM によってシステムに指定されるロード・ファクタ	LOAD (オプション)	ロード・バランシング

SERVICES **セクションで指定する情報**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
データ依存型ルーティングを行うときに、このサービスに使用されるルーティング基準名	ROUTING (オプション)	ルーティング条件の名前
SVCNAM がグループ・パラメータのすべての設定を取得するサーバ・グループ名	SRVGRP (オプション)	サーバ・グループ・パラメータ
SVCNM をキューから取り出すときの優先順位	PRIO (オプション)	サービスの優先順位
特定のサービスの処理にかかる時間 (秒単位)	SVCTIMEOUT (オプション)	サービスの処理時間
関連するサービスに対するトランザクションを自動的に開始するまでのデフォルトのタイムアウト値 (秒単位)	TRANTIME (DTP アプリケーションのみ)	トランザクションのタイムアウト値

SERVICES セクションの例

以下は、コンフィギュレーション・ファイルの SERVICES セクションの例です。

```
*SERVICES
#
DEFAULT:  LOAD=50 PRIO=50
RINGUP    BUFTYPE=”VIEW:ringup”
```

この例では、サービスのデフォルトのロードおよび優先順位の値は `50` です。宣言されている 1 つのサービスは、`RINGUP` サービスであり、バッファ・タイプ `ringup` `VIEW` が指定されています。

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の `UBBCONFIG(5)`
- 3-118 ページの「コンフィギュレーション・ファイルの ROUTING セクションの作成方法」

トランザクションの自動開始とタイムアウト間隔を指定する

`AUTOTRAN ={Y|N}` パラメータをコーディングして、要求メッセージが既にトランザクション・モードの場合に、トランザクションを自動的に開始するかどうかを指定できます。デフォルトは `N` です。

あるサービスに対するトランザクションが開始した後でそのトランザクションが失敗した場合、ロールバックされるまでのタイムアウト間隔を指定することができます。タイムアウト間隔が自動的に実行されるように指定するには、`TRANTIME` パラメータを次のように設定します。

`TRANTIME=`*`number`*

デフォルト値は 30 秒です。0 (タイムアウトの最大値) をコンピュータに指定すると、トランザクションでタイムアウトは発生しません。

UBBCONFIG ファイルの RESOURCES セクションでは、MAXTRANTIME という追加のトランザクション・タイムアウト・プロパティを使用できます。MAXTRANTIME タイムアウト値が TRANTIME タイムアウト値またはトランザクションを開始する際に tpbegin(3c) 呼び出しで渡されたタイムアウト値よりも小さい場合、トランザクションのタイムアウトは MAXTRANTIME 値まで減少します。

注記　MAXTRANTIME は BEA Tuxedo 8.0 以前を実行するマシン上で開始されるトランザクションには影響を与えません。ただし、BEA Tuxedo 8.1 以降を実行するマシンがトランザクションの影響を受ける場合は、そのノードに対して設定されている MAXTRANTIME 値までトランザクション タイムアウト値が制限 (必要に応じて減少) されます。

関連項目

- 『BEA Tuxedo Domains コンポーネント』
- MAXTRANTIME の詳細については、UBBCONFIG(5) の RESOURCES セクションにある MAXTRANTIME、または TM_MIB(5) の T_DOMAIN クラスにある TA_MAXTRANTIME を参照してください。

サービスで受け付けるバッファ・タイプのリストを指定する

BUFTYPE パラメータを使用すると、サービス・コードとは関係なく、バッファ・タイプをチェックするサービスを調整できます。このパラメータを設定する場合は、サービスで受け付けるバッファ・タイプのリストを次の形式で指定します。

type[:*subtype*[,*subtype*]]

すべての *subtype* を受け入れるには、サブタイプの値に * を設定します。

サービスの BUFTYPE パラメータの値が ALL の場合、このサービスはすべてのバッファ・タイプを受け付けます。デフォルト値は ALL です。

BUFTYPE パラメータの例

BUFTYPE の例	説明
`BUFTYPE="FML;VIEW:aud,aud2"`	サブタイプが aud および aud2 の FML および VIEW バッファ・タイプを受け付けます。
`BUFTYPE="FML;VIEW:*"`	すべての FML および VIEW バッファ・タイプを受け付けます。
`BUFTYPE=ALL`	すべてのバッファ・タイプを受け付けます (デフォルト)。

要求を処理する時間を指定する

リクエストの処理中に、予期しないシステム・エラーが発生し、サービスが停止 (フリーズ) したり、制御不可能になる場合があります。このようなプロセスは削除することが望ましいですが、実際にこれらのエラーを検出したり、エラー原因を追跡することは困難です。BEA Tuxedo システムには、エラーを識別できなくても、このようなプロセスを終了できるメカニズムが組み込まれています。このメカニズムを使用するには、SVCTIMEOUT パラメータを設定します。

SVCTIMEOUT パラメータは、サービスが要求を処理できる時間 (秒) を指定します。このパラメータで定義された間隔が経過してもサービスが要求を処理できなかった場合、その要求の処理は強制終了されます。本質的に、サービス・タイムアウトは、停止または制御不能状態にあるアプリケーション・

サーバのスカベンジ機能として働きます。デフォルトでは、BEA Tuxedo システムはサービス・プロセスを終了しません。この機能を有効にするには、SVCTIMEOUT を設定する必要があります。

UBBCONFIG ファイルの SVCTIMEOUT パラメータに値を割り当てるか、または TM_MIB の TA_SVCTIMEOUT 属性を動的に変更することによって設定できます。SVCTIMEOUT または TA_SVCTIMEOUT の値には、サービスが 1 つのリクエストの処理に費やす時間の 2 ～ 3 倍の時間を指定しておくことをお勧めします。このような値を設定しておくと、BEA Tuxedo システムは、停止状態のプロセスだけを削除します。

この節では、サービス・タイムアウト・エラーの原因と結果、および BEA Tuxedo システムによるエラーのレポート方法を説明します。また、エラーの処理方法に関するアドバイスも掲載しています。

タイムアウト発生時の処理

タイムアウトが発生すると、BEA Tuxedo システムは、停止されたサービスの実行元であるサーバ・プロセスを終了します。ただし、子プロセスは終了しません。次に、TPESVCERR エラーを返し、サービスの処理中に未知のエラーが発生したことを示します。会話型サービスでは、会話型イベントである TPEV_SVCERR が返されます。

サービスのタイムアウトの通知方法

BEA Tuxedo システムでは、以下の 3 つの機能を使用してサービス・タイムアウトを通知します。

- TPED_SVCTIMEOUT—tpstrerror(3c) より詳細なタイムアウト・エラーの情報を提供します。
- .SysServiceTimeout—システム・イベント
- .SysServiceTimeout に関する ULOG 情報

SVCTIMEOUT 値が設定可能になったため、クライアント・アプリケーション側では、SVCTIMEOUT の値を超過して発生した TPESVCERR とそれ以外の原因で発生した TPESVCERR を簡単に識別できるようにしておくことが必要です。この情報は、ULOG にも記録されていますが、クライアント・プログラムによって抽出するのは困難です。サービス・タイムアウトの TPESVCERR をその他のエラーと区別するため、プログラムで TPESVCERR を検出した後に tperrordetail(3c) ルーチンを呼び出し、サービス・タイムアウトが発生したときに TPED_SVCTIMEOUT を生成することができます。

また、サービス・タイムアウトが発生すると、システム・イベントである .SysServiceTimeout が生成されます。.SysServiceTimeout イベントが発生すると、ULOG に次のように記録されます。

ERROR:.SysServiceTimeout:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID サーバがサービス・タイムアウトのため強制終了しました。

サービス・タイムアウトの制御方法

- アプリケーション管理者は、UBBCONFIG ファイルの SERVICES セクションにある SVCTIMEOUT パラメータを変更するか、または TM_MIB の T_SERVER クラスまたは T_SERVICE クラスの TA_SVCTIMEOUT 属性を変更してサービス・タイムアウトを制御することができます。また、ULOG ファイルを監視して、サービス・タイムアウトの動作を確認することもできます。
- ULOG ファイルを監視する方法に加え、アプリケーション・オペレータは、サービス・タイムアウトの発生を警告する .SysServiceTimeout イベントをサブスクライブできます。
- アプリケーション・プログラマは、tperrordetail(3c) や tpstrerrordetail(3c) といった関数、および TPED_SVCTIMEOUT エラー詳細コードを使用できます。また、サービス・タイムアウトの発生時に生成されるシステム・イベント .SysServiceTimeout の 1 つまたは複数のサブスクリプションが可能です。

ロード・バランシングを有効にする

ロード・バランシングを有効にするには、RESOURCES セクションの LDBAL パラメータを Y に設定します。実行するサービスには、LOAD パラメータの値に基づいてロード・ファクタが割り当てられます。BEA Tuxedo システムは、各サーバの負荷の合計を監視します。各サービス要求は、負荷の合計が最も低いサーバにルーティングされます。ルーティング先のサーバの負荷は、ルーティングされたサービスのロード・ファクタ分 (LOAD) だけ増加します。

負荷の情報は、サービス要求のルーティング元サイトにのみ保存されます。BEA Tuxedo システムでは、分散アプリケーションのすべてのサイトに負荷の情報を常に伝達するのは非効率的です。このような環境でロード・バランシングを行う場合、各サイトは、そのサイトで発生した負荷だけを認識し、必要に応じてロード・バランシングを行います。つまり、各サイトは、指定されたサーバ (またはキュー) に関し、それぞれ異なる負荷の統計情報を持ちます。したがって、負荷が最も低いと見なされるサーバは、サイトごとに異なります。

ロード・バランシングが無効であり、複数のサーバが同じサービスを提供する場合は、使用可能な最初のキューが要求を受け取ります。

LDBAL パラメータの特性

LDBAL パラメータには、以下の特性があります。

- RESOURCES セクションの LDBAL パラメータが Y に設定されている場合、ロード・バランシングが行われます。
- ロード・ファクタは、サーバの負荷の合計に追加されます。
- ロード値は、サービスどうしの関連で決められます。

ルーティング基準名を定義する

データ依存型ルーティングを使用するときは、サービス用のルーティング基準を指定する必要があります。このような基準を指定するには、ROUTING パラメータを次のように設定します。

ROUTING=*string_value*

このパラメータが指定されないと、サービスはデータ依存型ルーティングを実行しません。

文字列の最大値は 15 文字です。サービスには、1 つ以上の ROUTING パラメータの値は割り当てられません。1 つのサービスに複数のエントリがあり、それらのエントリに異なる SRVGRP パラメータが含まれている場合でも、ROUTING の値はすべてのエントリで同じでなければなりません。

異なるサーバ・グループに対してサービス・パラメータを指定する

複数のグループに同じサービスを割り当てたり、異なるグループのサービス・エントリに対して設定したさまざまなサービス固有のパラメータに異なる値を割り当てることができます。そのためには、SRVGRP パラメータにグループ固有の値を指定して、各グループのサービスに対して個別のエントリを作成してください。

サービスの優先順位を指定してデータ・フローを制御する

PRIO パラメータを使用して、サービスの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・フローを有効に制御できます。PRIO には、0 〜 100 の数値を指定する必要があります。数値が大きいほど割り当てられる

サービスの優先順位も高くなります。優先順位の高いサービスは優先順位の低いサービスより先にキューから取り出されますが、システムは、10 回に 1 回ごとに FIFO 順序で要求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することがないようにしています。

たとえば、サーバ 1 はサービス A、B、および C を提供します。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 に対するサービス要求は、常に A または B に対する要求の前にキューから取り出されます。A および B に対する要求は、互いに等しくキューから取り出されます。

注記　`tpsprio()` 呼び出しを使用すると、優先順位を動的に変更できます。

PRIO パラメータの特性

PRIO パラメータには、以下の特性があります。

- サーバのキューにおけるサービスの優先順位を決めます。
- 最も高い優先順位が最優先されます。
- 10 回に 1 回ごとに要求は FIFO でキューから取り出されます。

優先順位の異なる SERVICES セクションの例

次の SERVICES セクションの例は、サービスに対する優先順位の割り当て方を示します。

```
*SERVICES
A  SRVGRP=GRP1 PRIO=50 LOAD=60
A  SRVGRP=GRP2  PRIO=70 LOAD=30
```

この例では、2 つのサーバ・グループに対して異なるサービス固有のパラメータが割り当てられます。サーバ・グループ `GRP1` のサービス A には、優先順位 50、およびロード 60 が割り当てられ、サーバ・グループ `GRP2` のサービス A に優先順位 70、およびロード 30 が割り当てられます。

サービスの処理時間を指定する

サービスの処理にかかる最大時間を秒単位で指定するには、SVCTIMEOUT パラメータを次のように設定します。

SVCTIMEOUT=*number*

値は 0 以上でなければなりません。この値が 0 の場合は、サービスに対してタイムアウトが適用されないことを示します。サービス要求を処理するサーバは SIGKILL シグナルによって終了します。このパラメータのデフォルト値は 0 です。

コンフィギュレーション・ファイルの INTERFACES セクションの作成方法

注記　このセクションは、BEA Tuxedo CORBA 環境にのみ適用されます。

コンフィギュレーション・ファイルの INTERFACES セクションは、BEA Tuxedo システムの CORBA 環境用のパラメータを定義するためのセクションです。このセクションでは、アプリケーションで使用される CORBA インターフェイスに対する、アプリケーション全体のデフォルト・パラメータを指定します。ファクトリ・ベース・ルーティングに参加する CORBA インターフェイスに対しては、インターフェイス名を定義し、Tuxedo CORBA 環境で各インターフェイスに適用するルーティング基準の名前を指定します。ファクトリ・ベース・ルーティングは、プロセスを特定のサーバ・グループに分散するための機能です。

ファクトリ・ベース・ルーティングをインプリメントする場合、INTERFACES セクションだけでなく、ROUTING セクションでルーティング基準を定義し、GROUPS セクションでグループ名を定義する必要があります。関連パラメータとファクトリ・ベース・ルーティングの詳細については、この章の 「コンフィギュレーション・ファイルの ROUTING セクションの作成方法」を参照してください。

INTERFACES セクションで CORBA インターフェイスを指定する

アプリケーションで使用する CORBA インターフェイス固有の情報は、コンフィギュレーション・ファイルの INTERFACES セクションで指定します。必須パラメータはありません。オプションのパラメータが不要の場合、CORBA インターフェイスをリストする必要はありません。INTERFACES セクションには、以下の種類の情報が含まれます。

- トランザクションを自動的に開始するかどうか (AUTOTRAN) (CORBA のみ)
- この CORBA インターフェイスでファクトリ・ベース・ルーティングに使用されるルーティング基準 (FACTORYROUTING) (CORBA のみ)
- ロード・バランシング情報 (LOAD)
- インターフェイスに対する優先順位の割り当て (PRIO)
- 個々のサーバ・グループに対する異なるサービス・パラメータ (SRVGRP)
- CORBA インターフェイスに関連付けられたトランザクションのタイムアウト値 (TRANTIME)
- この CORBA インターフェイスのメソッド処理のタイムアウト値 (TIMEOUT)

表 0-2 に、AUTOTRAN、FACTORYROUTING、LOAD、PRIO、SRVGRP、TRANTIME、および TIMEOUT パラメータの特性を示します。

表 0-2INTERFACES **セクションのパラメータの特性**

パラメータ	特性
`AUTOTRAN = {Y | N }`	操作の呼び出しを受け取るときにトランザクションを自動的に開始する場合は、CORBA インターフェイスごとに `AUTOTRAN` を `Y` に設定します。インターフェイスが既にトランザクション・モードになっている場合は、`AUTOTRAN=Y` を指定しても無効です。デフォルトは `N` です。 `AUTOTRAN` の設定が有効かどうかは、システム設計者がインプリメンテーション・コンフィギュレーション・ファイル (ICF) またはサーバ記述ファイル (XML) でそのインターフェイスに対して指定したトランザクション・ポリシーによって異なります。このトランザクション・ポリシーが、実行時に、関連する `T_IFQUEUE MIB` オブジェクトのトランザクション・ポリシーの属性になります。この値が実際にアプリケーションの動作に影響するのは、システム設計者が トランザクション・ポリシーを「オプション」に指定した場合だけです。 注記　このパラメータを正しく機能させるには、システム設計者とシステム管理者の間のコミュニケーションが重要です。プログラマが設定した ICF または XML パラメータを知らずにシステム管理者がこの値を `Y` に設定すると、実行時にパラメータが適切に動作しない可能性があります。
`FACTORYROUTING = criterion-name`	この CORBA インターフェイスのファクトリ・ベース・ルーティングで使用するルーティング基準の名前を指定します。ファクトリ・ベース・ルーティングを要求するインターフェイスには、`FACTORYROUTING` パラメータを指定する必要があります。
`LOAD = number`	CORBA インターフェイスがシステムに与える相対的な負荷を示す、1 ～ 100 の任意の数です。この値は、このアプリケーションで使用されるほかの CORBA インターフェイスに割り当てられた `LOAD` 値との関係で相対的に決まります。デフォルトは 50 です。この値は、BEA Tuxedo システムが要求のルーティング先サーバを選択する際に使用されます。
`PRIO = number`	CORBA インターフェイスのすべてのメソッドに対して、キューから取り出す優先順位を指定します。この値には、1 ～ 100 の値を指定します。100 は、優先順位が最も高いことを示します。デフォルト値は 50 です。

表 0-2INTERFACES **セクションのパラメータの特性 （続き）**

パラメータ	特性
SRVGRP = server-group-name	SRVGRP を使用して、INTERFACES セクションのこの部分で定義されたパラメータを、指定したサーバ・グループに適用します。この機能を使用すると、指定された CORBA インターフェイスで、サーバ・グループごとに異なるパラメータを定義することができます。
TRANTIME = number	AUTOTRAN が Y に設定されている場合は、TRANTIME パラメータを設定する必要があります。TRANTIME は、対象トランザクションのタイムアウト値 (秒単位) を指定するためのパラメータです。この値には、0 〜 2,147,483,647 (231 - 1、つまり約 70 年) の値を指定します。0 は、トランザクションにタイムアウトが設定されていないことを示します。デフォルト値は 30 秒です。
TIMEOUT=number	この CORBA インターフェイスでメソッドの処理に使用できる時間を秒単位で指定します。値は 0 以上でなければなりません。この値が 0 の場合は、インターフェイスに対してタイムアウトが適用されないことを示します。タイムアウト・メソッドにより、インターフェイスに対してメソッドを処理するサーバが SIGKILL イベントと共に終了します。このため、インターフェイスに対する処理に時間のかかるメソッドの場合は、タイムアウト値を指定することをお勧めします。

ファクトリ・ルーティング基準を指定する

各 CORBA インターフェイスに対し、INTERFACES セクションでインターフェイスのルーティング基準を指定します。INTERFACES セクションでは、FACTORYROUTING 識別子を使用してルーティング基準を指定します。

University のサンプル

University Production サンプル・アプリケーションでは、ファクトリ・ベース・ルーティングのコード作成方法を示しています (コード リスト 3-3 参照)。このサンプルの UBBCONFIG ファイル (ubb_p.nt または ubb_p.mk) は、BEA Tuxedo ソフトウェアのインストール・ディレクトリに収められています。\samples\corba\university\production サブディレクトリを検索してください。

コード リスト 3-3　Production **サンプルの** INTERFACES **セクション**

```
*INTERFACES

   "IDL:beasys.com/UniversityP/Registrar:1.0"
        FACTORYROUTING = STU_ID

   "IDL:beasys.com/BillingP/Teller:1.0"
        FACTORYROUTING = ACT_NUM
```

上の例では、University Production サンプルの 2 つのインターフェイスに、完全修飾されたインターフェイス名が使用されています。また、FACTORYROUTING 識別子によって、ルーティング値の名前がそれぞれ STU_ID と ACT_NUM に指定されています。

INTERFACES FACTORYROUTING パラメータと ROUTING セクションの関連性については、3-123 ページの「University Production サンプル・アプリケーションの CORBA ファクトリ・ベース・ルーティング」を参照してください。

Bankapp のサンプル

コード リスト 3-4 は、サンプル・アプリケーション Bankapp でのファクトリ・ベース・ルーティングの指定例を示しています。

コード リスト 3-4　Bankapp **サンプルのファクトリ・ベース・ルーティング**

```
*INTERFACES
        "IDL:BankApp/Teller:1.0"
         FACTORYROUTING=atmID

*ROUTING
     atmID
            TYPE = FACTORY
            FIELD = "atmID"
            FIELDTYPE = LONG
            RANGES = "1-5:BANK_GROUP1,
                      6-10: BANK_GROUP2,
                        *:BANK_GROUP1
```

この例では、IDL:Bankapp/Teller インターフェイスが、ROUTING セクションで定義された atmID という名前のファクトリ・ベース・ルーティング基準を使用しています。この例の ROUTING セクションでは、2 つのグループに処理を分散するように指定しています。atmID フィールドが 1 〜 5 または 10 より大きい場合は、BANK_GROUP1 がアプリケーションで使用されるインターフェイスを処理し、atmID が 6 〜 10 の場合は、BANK_GROUP2 がアプリケーションで使用されるインターフェイスを処理します。

ロード・バランシングを有効化する

BEA Tuxedo の CORBA 環境では、ロード・バランシングは常に有効です。

各 CORBA インターフェイスには、起動するたびに LOAD ファクタが割り当てられます。これで、各サーバ・プロセスの実行結果として CORBA インターフェイス全体にかかる負荷がトラッキングされます。各インターフェイス要求は、負荷の合計が最も低いサーバにルーティングされます。ルーティング先のサーバの負荷は、ルーティングされた CORBA インターフェイスのロード・ファクタ分 (LOAD) だけ増加します。ロード・バランシングが無効であり、複数のサーバが同じ CORBA インターフェイスを提供する場合は、使用可能な最初のキューが要求を受け取ります。

BEA Tuxedo CORBA 環境でのロード・バランシングの詳細については、『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』の「システム制御のロード・バランシングの有効化」を参照してください。

BEA Tuxedo のリリース 8.0 では、CORBA 環境における並列オブジェクトがサポートされるようになりました。これにより、ローカル・ドメイン内の複数サーバ間でロード・バランシングが可能になります。BEA Tuxedo CORBA 環境の並列オブジェクトについては、『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』の「パラレル・オブジェクトの使用」を参照してください。

インターフェイスの優先順位を指定してデータ・フローを制御する

PRIO パラメータを使用してインターフェイスの優先順位を割り当てることにより、BEA Tuxedo のクライアントまたはサーバ・アプリケーションのデータ・フローを制御することができます。たとえば、サーバ 1 がインターフェイス A、B、および C を提供し、インターフェイス A と B の優先順位が 50、インターフェイス C の優先順位が 70 であるとします。インターフェイス C に対する要求は、常に A または B に対する要求より先にキューから取り出されます。A および B に対する要求がキューから取り出される優先順位は同等です。システムは、10 回に 1 回の割合で FIFO 順序で要求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 パラメータには、以下の特性があります。

- サーバのキューにおける CORBA インターフェイスの優先順位を指定します。
- 最も高い優先順位が最優先されます。
- 10 回に 1 回ごとに要求は FIFO でキューから取り出されます。

サーバ・グループごとに異なるインターフェイス・パラメータを指定する

ロード・バランシングや優先順位など、インターフェイス固有のパラメータは、サーバ・グループごとに変えて設定することができます。この場合、SRVGRP パラメータに指定したグループごとに、インターフェイスの設定を繰り返す必要があります。

コンフィギュレーション・ファイルの ROUTING セクションの作成方法

UBBCONFIG の ROUTING セクションでは、SERVICES セクション (ATMI のデータ依存型ルーティングの場合) や INTERFACES セクション (CORBA のファクトリ・ベース・ルーティングの場合) で指定されたルーティング基準を詳細に定義することができます。

注記　CORBA 環境でのファクトリ・ベース・ルーティングの設定については、『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。

次の表では、ROUTING セクションの各パラメータを説明し、参照先へのリンクやその他の情報を示します。

ROUTING **セクションで指定する情報** (**オプション**)	**パラメータ** (**必須** / **オプション**)	**参照先** (**クリックするとリンク先にジャンプ**)
ルーティング・フィールドの範囲および関連するサーバ・グループを指定します。	RANGES (必須)	範囲基準
この値は、最大 15 文字の文字列です。 ATMI の場合は、データ依存型ルーティング用に SERVICES セクションの ROUTING パラメータで指定したルーティング基準名になります。 CORBA の場合は、ファクトリ・ベース・ルーティング用に、INTERFACES セクションの FACTORYROUTING パラメータで指定したルーティング基準名になります。	criterion_name (必須)	

ROUTING **セクションで指定する情報 (オプション)**	**パラメータ (必須 / オプション)**	**参照先 (クリックするとリンク先にジャンプ)**
ルーティングのタイプを指定します。 ATMI の場合、デフォルトは TYPE=SERVICE になり、Tuxedo ATMI 環境で使用される既存の UBBCONFIG ファイルが引き続き正常に動作することが確認されます。 CORBA では、CORBA インターフェイスに対してファクトリ・ベース・ルーティングをインプリメントする場合は TYPE=FACTORY を使用します。	TYPE	
ルーティング・フィールドの名前。値は、FML バッファ、XML の要素または要素の属性、FML フィールド・テーブルで指定された VIEW フィールド名 (FLDTBLDIR および FIELDTBLS 環境変数を使用)、または FML の VIEW テーブル (VIEWDIR および VIEWFILES 環境変数を使用) になります。この情報は、メッセージの送信時に、データ依存型ルーティングに関連するフィールド値を取得するために使用されます。 CORBA のファクトリ・ベース・ルーティングでは、この値によってルーティング・フィールドの名前が指定されます。最大 30 文字まで指定できます。この値は、そのインターフェイスの TP::create_object_reference (C++) または com.beasys.Tobj.TP::create_object_reference (Java) に対するファクトリの呼び出しで、ファクトリ・ベース・ルーティングに指定されたフィールド名と対応していなければなりません。	FIELD (必須)	ルーティング・バッファのフィールドとタイプ
このルーティング・エントリで有効なデータ・バッファのタイプとサブタイプのリストを指定します。このパラメータには、最大 32 のタイプとサブタイプの組み合わせを最大 256 文字で指定できます。	BUFTYPE (必須)	バッファのタイプとサブタイプ

ROUTING セクションの例

以下は、コンフィギュレーション・ファイルの ROUTING セクションの例です。

```
BRNCH FIELD=B_FLD
RANGES="0-2:DBG1,3-5:DBG2,6-9:DBG3"
BUFTYPE="FML"
```

ルーティング・バッファ・フィールドとフィールド・タイプを定義する

次の表は、ルーティング・バッファ・フィールドとフィールド・タイプの説明です。

パラメータ	説明
FIELD	ルーティングが行われるバッファ・フィールドの名前。30 文字まで使用できます。 BEA Tuxedo のデータ依存型ルーティングでは、このパラメータの値は、FML フィールドの名前 (FML バッファ)、XML の要素または属性、FML フィールド・テーブルで指定された VIEW フィールド名 (FLDTBLDIR および FIELDTBLS 環境変数を使用)、FML の VIEW テーブル (VIEWDIR および VIEWFILES 環境変数を使用) のいずれかになります。この情報は、メッセージの処理時に、データ依存型ルーティングに関連するフィールド値を取得するために使用されます。FML32 バッファ内のフィールドがルーティングに使用される場合は、8191 以下の数値をフィールド番号として指定します。 XML 文書のルーティングでは、FIELD 構文には、ルーティング 要素のタイプ (または名前) または ルーティング要素の属性名が含まれます。FIELD パラメータは、次の構文で定義します。 `root_element[/child_element][/child_element][/..` `.][/@attribute_name]` 要素は、XML 文書またはデータグラムの要素のタイプ (要素名) または要素の属性名と見なされます。この情報は、文書またはデータグラムの送信時に、データ依存型ルーティングに関連する要素の内容または属性値を取得するために使用されます。BEA Tuxedo システムでは、インデックス指定がサポートされていないため、データ依存型ルーティングで XML バッファを処理する場合は、指定された要素のタイプの最初のオカレンスだけが認識されます。 CORBA のファクトリ・ベース・ルーティングでは、この値によってルーティング・フィールドの名前が指定されます。最大 30 文字まで指定できます。この値は、そのインターフェイスの TP::create_object_reference (C++) または com.beasys.Tobj.TP::create_object_reference (Java) に対するファクトリの呼び出しで、ファクトリ・ベース・ルーティングに指定されたフィールド名と対応していなければなりません。
FIELDTYPE	このパラメータは、XML バッファをルーティングする場合にのみ使用されます。このパラメータは、FIELD で指定されたルーティング・フィールドのタイプを示します。構文は次のとおりです。 FIELDTYPE=*type* *type* は string、char、hort、long、float、または double のいずれかです。 ルーティング・フィールドのデフォルトのデータ型は string です。

範囲の基準を指定する

RANGES パラメータでは、フィールド値を次のようにグループ名にマップできます。

RANGES=”[*val1*[-*val2*]:*group1*] [,*val3*[-*val4*]:*group2*]...[,*:*groupn*]”

val1、*val2* などはフィールドの値であり、*groupn* はグループ名または任意のグループが選択されることを示すワイルドカード文字 (*) です。最後に *val* の代わりに指定されている * は「catch-all」、つまり、データが指定された範囲内にない場合はデフォルトのグループに割り当てられることを示します。また、データが範囲内にあっても、その範囲のエントリに関連する有効なサーバがグループ内にない場合、サービス要求は、ワイルドカード「*」で指定したデフォルトのグループに転送されます。*val1* には、以下の値を使用できます。

- 数値 (数値フィールドで使用)
- STRING バッファまたは CARRAY バッファ (一重引用符で囲む)
- MIN または MAX (マシンのデータの最小値または最大値)

範囲はいくつでも指定できますが、ルーティング情報は共用メモリに保存されるため、システム・リソースを消費します。

注記　範囲は重複してもかまいません。ただし、値が 2 つの範囲に当てはまる場合、その値は最初のグループにマッピングされます。たとえば、RANGES が RANGES=”0-5:Group1,3-5:Group2" に指定されている場合、4 は Group1 にルーティングされます。

バッファ・タイプを定義する

BEA Tuxedo のデータ依存型ルーティングでは、BUFTYPE パラメータを使用して、バッファ・タイプを指定できます。このパラメータは、特定のバッファ・タイプとサブタイプに対してルーティング基準を適用するため、

SERVICES セクションの対応するパラメータと似ています。ルーティングに使用できるのは、FML、XML、および VIEW 型だけです。構文は、SERVICES セクションと同じく、*type:subtype[,subtype]* をセミコロンで区切ってリストします。ルーティング基準には 1 つのタイプしか指定できません。この制限により、サービスのルーティング時に受け付けられるバッファ・タイプの数が限定されます。

University Production サンプル・アプリケーションの CORBA ファクトリ・ベース・ルーティング

CORBA University Production サンプル・アプリケーションでは、BEA Tuxedo でファクトリ・ベース・ルーティングをインプリメントする方法を示します。このサンプルの UBBCONFIG ファイル (ubb_p.nt または ubb_p.mk) は、BEA Tuxedo ソフトウェアのインストール・ディレクトリに収められています。\samples\corba\university\production サブディレクトリを検索してください。

ubb_b.nt コンフィギュレーション・ファイルから抜粋した以下の INTERFACES、ROUTING、および GROUPS セクションは、BEA Tuxedo の CORBA アプリケーションでファクトリ・ベース・ルーティングをインプリメントする方法を示しています。

INTERFACES セクションには、ファクトリ・ベース・ルーティングを有効にするインターフェイスの名前がリストされています。各インターフェイスに対し、このセクションでインターフェイスのルーティング基準を指定します。このセクションでは、コード リスト 3-5 の例に示すように、識別子 FACTORYROUTING を使用してルーティング基準を指定します。

コード リスト 3-5 Production **サンプルの** INTERFACES **セクション**

```
*INTERFACES

   "IDL:beasys.com/UniversityP/Registrar:1.0"
        FACTORYROUTING = STU_ID

   "IDL:beasys.com/BillingP/Teller:1.0"
        FACTORYROUTING = ACT_NUM
```

上の例では、ファクトリ・ベース・ルーティングを使用する Production サンプルで、2 つのインターフェイスに完全修飾されたインターフェイス名が使用されています。また、FACTORYROUTING 識別子によって、ルーティング値の名前がそれぞれ STU_ID と ACT_NUM に指定されています。

ROUTING セクションでは、ルーティング値ごとに以下のデータを指定します。

- TYPE パラメータでは、ルーティングのタイプを指定します。Production サンプルでは、ルーティングのタイプはファクトリ・ベース・ルーティングです。このため、このパラメータは FACTORY に定義されます。
- FIELD パラメータでは、ファクトリがルーティング値として挿入する変数名を指定します。Production サンプルでは、このフィールド・パラメータがそれぞれ student_id と account_number に設定されています。
- FIELDTYPE パラメータでは、ルーティング値のデータ型を指定します。Production サンプルでは、student_id と account_number のフィールド型が long 型に設定されています。
- RANGES パラメータでは、サーバ・グループと、各ルーティング値の有効範囲のサブセットを関連付けます。

コード リスト 3-6 は、Production サンプル・アプリケーションで使用される UBBCONFIG ファイルの ROUTING セクションを示しています。

コード リスト 3-6 Production **サンプルの** ROUTING **セクション**

```
*ROUTING

   STU_ID
      FIELD     = "student_id"
      TYPE      = FACTORY
      FIELDTYPE = LONG
      RANGES    = "100001-100005:ORA_GRP1,100006-100010:ORA_GRP2"

   ACT_NUM
      FIELD     = "account_number"
      TYPE      = FACTORY
      FIELDTYPE = LONG
      RANGES    = "200010-200014:APP_GRP1,200015-200019:APP_GRP2"
```

上の例では、ID が特定の範囲に含まれる学生の Registrar オブジェクトが 1 つのサーバ・グループに、別の範囲に含まれる学生の Registrar オブジェクトが別のグループにインスタンス化されています。同様に、特定の範囲に含まれる口座の Teller オブジェクトが 1 つのサーバ・グループに、別の範囲に含まれる口座の Teller オブジェクトが別のグループにインスタンス化されています。

`UBBCONFIG` ファイルの `ROUTING` セクションで `RANGES` 識別子によって指定されたグループを、識別して設定する必要があります。たとえば、Production サンプルでは、`ORA_GRP1`、`ORA_GRP2`、`APP_GRP1`、および `APP_GRP2` の 4 つのグループが指定されていますが、ここでこれらのグループを設定し、実行するマシンを識別する必要があります。

コード リスト 3-7 は、Production サンプルの `UBBCONFIG` ファイルにある `GROUPS` セクションの例を示しています。`GROUPS` セクションの名前と、`ROUTING` セクションで指定された名前の関係に注意してください。ファクトリ・ベース・ルーティングを正しく機能させるには、両方の名前を一致させる必要があります。また、アプリケーションでグループを設定する際にグループ名を変更した場合は、`ROUTING` セクションにも反映しなければなりません。BEA Tuxedo ソフトウェアに収録されている Production サンプルは、1 台のマシンで実行するように設定されていますが、複数のマシンで実行するように設定することもできます。

コード リスト 3-7 Production **サンプルの** GROUPS **セクション**

```
*GROUPS

APP_GRP1
   LMID = SITE1
   GRPNO = 2
   TMSNAME = TMS

APP_GRP2
   LMID = SITE1
   GRPNO = 3
   TMSNAME = TMS

ORA_GRP1
   LMID = SITE1
   GRPNO = 4
OPENINFO = "ORACLE_XA:Oracle_XA+Acc=P/scott/tiger+SesTm=100+LogDir=.+MaxCur=5"
   CLOSEINFO = ""
   TMSNAME = "TMS_ORA"

ORA_GRP2
   LMID = SITE1
   GRPNO = 5
OPENINFO = "ORACLE_XA:Oracle_XA+Acc=P/scott/tiger+SesTm=100+LogDir=.+MaxCur=5"
CLOSEINFO = ""
TMSNAME = "TMS_ORA"
```

Bankapp サンプル・アプリケーションの CORBA ファクトリ・ベース・ルーティング

コード リスト 3-8 は、INTERFACES セクションを使用して Bankapp サンプル・アプリケーションを拡張し、ファクトリ・ベース・ルーティングを使用する方法を示しています。BEA Tuxedo ソフトウェアに収録されているサンプル・アプリケーションでは、以下のパラメータは設定されていません。

コード リスト 3-8 Bankapp **サンプルの** INTERFACES **セクション**

```
*INTERFACES
        "IDL:BankApp/Teller:1.0"
         FACTORYROUTING=atmID
*ROUTING
     atmID
            TYPE = FACTORY
            FIELD = "atmID"
            FIELDTYPE = LONG
            RANGES = "1-5:BANK_GROUP1,
                      6-10: BANK_GROUP2,
                      *:BANK_GROUP1
*GROUPS
      SYS_GRP
             LMID        = SITE1
                  GRPNO       = 1
      BANK_GROUP1
                  LMID        = SITE1
                  GRPNO       = 2
      BANK_GROUP2
                  LMID        = SITE1
                  GRPNO       = 3
```

この例では、IDL:Bankapp/Teller インターフェイスが、ROUTING セクションで定義された atmID という名前のファクトリ・ベース・ルーティング基準を使用しています。この例では、以下の 2 つのサーバ・グループに処理が分散されます。

- BANK_GROUP1 は、atmID フィールドが 1 ～ 5 の間または 11 以上の場合に、アプリケーションで使用されるインターフェイスを処理します。
- BANK_GROUP2 は、atmID が 6 ～ 10 の間の場合にアプリケーションで使用されるインターフェイスを処理します。

スレッドを利用する BEA Tuxedo システムの設定方法

マルチコンテキスト化されたアプリケーションをコンフィギュレーションするには、UBBCONFIG ファイルを通常どおりに編集し、アプリケーションに必要な次の表のパラメータを追加します。編集には、テキスト・エディタまたは BEA Tuxedo Administration Console を使用します。

表 0-3 スレッドを使用するコンフィギュレーション・ファイルのパラメータの設定

セクション	パラメータ	注意事項
RESOURCES	MAXACCESSERS	オプション・パラメータ。ただし、50 アクセサ (デフォルト値) より大きい値を指定する必要があります。 ライセンスの制限のため、マルチコンテキスト化されたクライアントの各コンテキストは個別にカウントされます。
	NOTIFY	オプション・パラメータ。任意通知型通知に適用するデフォルトの方法を定義します。マルチコンテキスト化されたアプリケーションの有効値は、次のとおりです。 ■ DIPIN ■ THREAD ■ IGNORE
MACHINES	MAXACCESSERS	オプション・パラメータ。ただし、50 アクセサ (デフォルト値) より大きい値を指定する必要があります。 ライセンスの制限のため、マルチコンテキスト化されたクライアントの各コンテキストは個別にカウントされます。
	MAXWSCLIENTS	オプション・パラメータ。 マルチコンテキスト化されたワークステーション・クライアントの各コンテキストは、ライセンス制限のため、個別にカウントされます。デフォルト値は 0 であるため、ワークステーション・クライアントが定義されているマシンを介してシステムにアクセスする場合は、このパラメータを設定する必要があります

表 0-3 スレッドを使用するコンフィギュレーション・ファイルのパラメータの設定 （続き）

セクション	パラメータ	注意事項
SERVERS	MINDISPATCHTHREADS	オプション・パラメータ。
	MAXDISPATCHTHREADS	マルチスレッド化されたサーバの必須パラメータ。 既存のサーバをマルチスレッド化する場合、経験のあるプログラマは、サーバのソース・コードがスレッド・セーフな方法で作成されているかどうかを検証しなければなりません。つまり、コンフィギュレーション・ファイルの MAXDISPATCHTHREADS の値を増やすだけで、静的変数でシングルスレッド化されたサーバをマルチスレッド化されたサーバに変換することはできません。このようなサーバは、マルチスレッド用として作成する必要があります。
	THREADSTACKSIZE	オプション・パラメータ。 サーバ・ディスパッチ・スレッドに対してサイズの大きいスタックが必要な場合に設定します。 デフォルト値は 0 です。ほとんどのアプリケーションでは、デフォルト値で十分です。オペレーティング・システムに 0 が渡されると、オペレーティング・システムに設定されているデフォルト値が呼び出されます。

コンフィギュレーション・ファイルのコンパイル方法

コンフィギュレーション・ファイルのコンパイルとは、テキスト形式の UBBCONFIG ファイルを、バイナリ形式の TUXCONFIG ファイルに変換することです。コンフィギュレーション・ファイルをコンパイルするには、tmloadcf コマンドを実行します。tmloadcf により UBBCONFIG ファイルが解析され、バイナリ形式のファイルがロードされます。

tmloadcf は、ファイル (UBBCONFIG の構文に記述された標準入力) を読み取って構文をチェックし、成功すると、バイナリ形式のコンフィギュレーション・ファイルである TUXCONFIG ファイルをロードします。TUXCONFIG 環境変数および TUXOFFSET 環境変数 (オプション) により、TUXCONFIG ファイルとオフセットが指定され、情報はこの場所に格納されます。tmloadcf は、-c または -n オプションが指定されていない限り、UBBCONFIG ファイルの RESOURCES セクションで指定されている MASTER マシンでのみ実行できます。

注記　UBBCONFIG ファイルの RESOURCES セクションに UID が指定されている場合、この値は、tmloadcf を実行する担当者のユーザ識別子 (UID) と同じでなければなりません。

TUXCONFIG 環境変数に指定されたパス名は、UBBCONFIG ファイルの MACHINES セクションに指定されている TUXCONFIG パラメータの値と完全に一致しなければなりません (大文字と小文字の区別も含む)。パス名が一致していない場合、tmloadcf(1) は正常に実行されません。

4 トランザクションについて

ここでは、次の内容について説明します。

- トランザクションとは
- トランザクションの利点
- グローバル・トランザクションの例
- BEA Tuxedo トランザクション・マネージャ (TM) とは
- 分散トランザクション処理をトラッキングする
- 2 フェーズ・コミットを使用してトランザクションをコミットする

注記　BEA Tuxedo CORBA 環境でトランザクションを使用する方法については、『BEA Tuxedo CORBA トランザクション』を参照してください。

トランザクションとは

トランザクションとは、関連するアクションの集まりです。グローバル・トランザクションとは、複数のプログラムおよび複数のリソース・マネージャにまたがる関連アクションの集まりです。この章で使用する「トランザクション」は、グローバル・トランザクションを意味します。

トランザクションの単純な例として、銀行口座からの引き出し処理があります。この処理では、一定の金額を引き出して預金残高の状態を変更する、という一連のアクションが行われます。このトランザクションでは、次の 3 つの操作を含む手順を実行する必要があります。

トランザクションの手順	銀行口座の引き出し処理の手順
1. 実行するアクティビティを確認します。	1. 引き出し処理を行うかどうかを確認します。
2. トランザクションの処理を実行します。	2. 口座から指定された金額を引き出します。
3. 完了した処理の永続的なレコードを作成します。	3. 預金残高のレコードを更新します。

これらの処理は、このトランザクション専用に作成されたソフトウェア・モジュールによって実行されます。このモジュールには、トランザクションを開始および終了するコードも必要です。トランザクションの開始および終了を示すコード部分が、トランザクションのソフトウェア・モジュールの主要な部分ではない場合、これらのコード部分は通常、別のモジュールにパッケージ化されます。

トランザクション・コーディネータは、トランザクションに参加するリソース間で、トランザクションを管理するロジックを実行するソフトウェア・モジュールです。

ACID 特性とは

銀行口座からの引き出し処理などのトランザクションでは、トランザクションを構成する操作がすべて成功するか、またはすべて失敗しなければなりません。たとえば、トランザクションを構成する操作のうち、1 つの操作が成功し、別の操作が失敗するとします。つまり、銀行口座から預金を引き出しても、引き出し後の金額が預金残高に反映されないとすれば、その銀行は営業を続けることができません。

トランザクションを構成する操作は、すべて成功するか、または失敗しなければならない、という特性を「原子性」と言います。BEA Tuxedo システムのすべてのトランザクションは、原子性という特性を持ち、さらに、「一貫

性」、「独立性」、および「持続性」という関連する特性を備えています。BEA Tuxedo システムで実行されるトランザクションのこうした 4 つの属性を ACID 特性と呼びます。

表 0-4BEA Tuxedo **トランザクションの** ACID **特性**

特性	**意味**
原子性	トランザクションは個別の作業単位であり、トランザクションを構成する操作はすべて成功するか、または失敗しなければなりません。この種の操作には、キューへのメッセージの登録、データベースの更新、およびトランザクション結果の画面表示などがあります。
一貫性	トランザクションは、正しい状態のままシステムを終了するか、またはアボートしなければなりません。トランザクションの状態が不安定な場合は、初期状態に戻らなければなりません。
独立性	トランザクションの動作は、同時に実行されているほかのトランザクションによる影響を受けません。トランザクションでは、共用リソースへのアクセスをシリアル化しなければならず、同時に実行されるプログラムがお互いの操作を妨害しないことが保証されなければなりません。
持続性	コミットされたトランザクションの結果は永続的です。システムが異常終了しても、トランザクションによって行われた変更は永続し、持続します。

トランザクションの成功と失敗

トランザクションの成功または失敗は、原子性の要件によって決まります。

状況	結果
トランザクション内のいずれかの操作が何らかの原因で失敗した場合	■ トランザクションはアボート、つまり直ちに終了します。 ■ トランザクションはロールバックします。つまり、このトランザクションで実行した処理を元に戻し、状態をトランザクションの実行前に戻します。たとえば、銀行口座からの金額の引き出しに失敗し、処理がロールバックされると、口座には、トランザクションの実行前と同じ金額が保持され、残高としてトランザクションの実行前と同じ金額が表示されます。
トランザクション内のすべての操作が成功した場合	クライアントは、トランザクションをコミットします。つまり、トランザクションを終了できる状態になり、結果を保存する必要があることを正式に通知します。たとえば、発注データベースの更新が永続化されると、出荷部門に送られた注文は、その部門のキューに永続的なレコードとして登録されます。

トランザクションの利点

BEA Tuxedo システム、およびその通信 API とプロトコルは、トランザクションを使用するように設計されています。BEA Tuxedo の通信呼び出しを使用すると、トランザクションを簡単に作成することができ、分散アプリケーションの作成には不可欠なツールです。

トランザクションを使用すると、以下のことが可能になります。

- 分散アプリケーションを簡単に作成できます。
- 通信の結果を 1 つの単位としてコミットできます。

- マシン、プログラム、およびネットワークに関する障害など、分散環境で発生する可能性がある問題に対して、すばやく対処できます。
- エラーが発生しても、簡単なプログラムを使って、処理を元に戻すことができます。

グローバル・トランザクションの例

電子商取引を行う小売店が、CUST_ORDER というサービスを使用するとします。顧客がその小売店の Web サイトで注文を行うと、CUST_ORDER サービスにより次の 2 つの操作が実行されます。

- 小売店の発注データベースを更新します。
- 新しい注文を出荷部門に送信し、出荷待ちのキューに登録します。

小売店側では、CUST_ORDER サービスを原子性の原則に従って実行する必要があります。つまり、CUST_ORDER が実行された場合、データベースの更新および出荷部門のキューへの要求の登録が両方とも成功しなければなりません。CUST_ORDER サービスが常に原子性に基づいて処理されるようにするため、CUST_ORDER を呼び出すクライアントは、要求をグローバル・トランザクションに関連付けます。

クライアントは、サービスをグローバル・トランザクションに関連付けるため、次の手順に従います。

1. tpbegin() を呼び出してトランザクションを開始します。
2. サービス要求を発行します。
3. tpcommit() を呼び出してトランザクションを終了します。

操作は、1 つの作業単位で、グローバル・トランザクションの一環として実行されます。CUST_ORDER サービスが呼び出されると、サーバにクライアントのトランザクションが複製転送されます。その結果、発注データベースへのアクセスと、出荷用キューへの注文の登録は、クライアントのトランザクションの一部となります。

何らかの原因でどちらかの操作が失敗すると、システム・エラーまたはアプリケーション・エラーが発生し、トランザクションの作業は元に戻され (ロールバックされ) ます。つまり、トランザクションは初期状態に戻ります。

一方、2 つの操作が両方とも成功すると、クライアントはトランザクションをコミットします。つまり、トランザクションの結果を確定することを正式に通知します。これで、発注データベースへの更新は永続的となり、出荷部門に送信された注文は、その部門のキューに登録されます。

BEA Tuxedo トランザクション・マネージャ (TM) とは

リソース・マネージャ (RM: Resource Manager) は、データベース管理システムや Application Queuing Manager のようなデータ・リポジトリであり、データにアクセスするためのツールを備えています。BEA Tuxedo システムは、1 つまたは複数の RM を使用してアプリケーションの状態を管理します。たとえば、銀行の預金残高のレコードは、RM に保存されています。預金引き出しサービスによってアプリケーションの状態が変わると、変更後の預金残高が、適切な RM に記録されます。

BEA Tuxedo システムは、XA インターフェイス対応の RM が関与するトランザクションを管理するのに役立ちます。BEA Tuxedo システムは、トランザクション・マネージャ (TM: Transaction Manager) として動作し、トランザクションに関連するすべての操作およびすべてのモジュールを調整します。

TM は、システム全体に渡るリソースが関与するグローバル・トランザクションを調整します。個々のリソースは、ローカルのリソース・マネージャ (RM) によって管理されます。トランザクション・マネージャ・サーバ (TMS: Transaction Manager Server) は、複数のリソースに関与するトランザクションの開始、コミット、およびアボートを行います。アプリケーション・コードは、RM に対して標準の埋め込み型 SQL インターフェイスを使用し、読み取りや更新を行います。TMS は、RM に対して XA インターフェイスを使用し、グローバル・トランザクション操作を実行します。

次の表は、個々のトランザクションの代わりに TM が行うアクションをまとめたものです。

表 0-5 トランザクション・マネージャが実行するアクション

状況	トランザクション・マネージャのアクション
アプリケーションがトランザクションを開始した場合	グローバル・トランザクション識別子 (GTRID) をトランザクションに割り当てます。
ほかのプロセスが、トランザクションを開始したプロセスと通信する場合	通信相手のプロセスをトラッキングします。
トランザクション処理の一環として RM へのアクセスが行われる場合	RM に適切な GTRID を渡します。この結果、RM は、トランザクション処理でアクセスされるデータベース・レコードを監視できます。
アプリケーションからトランザクションのコミットが通知された場合	2 フェーズ・コミット・プロトコルを実行します。具体的には、以下を実行します。 (a) フェーズ 1 の実行中に、通信相手とコンタクトします。(b) フェーズ 1 が成功したことをログに記録します。 (c) フェーズ 2 で通信相手とコンタクトします。
アプリケーションからトランザクションのアボートが通知された場合	ロールバック手順を実行します。
障害が発生した場合	障害を回復します。

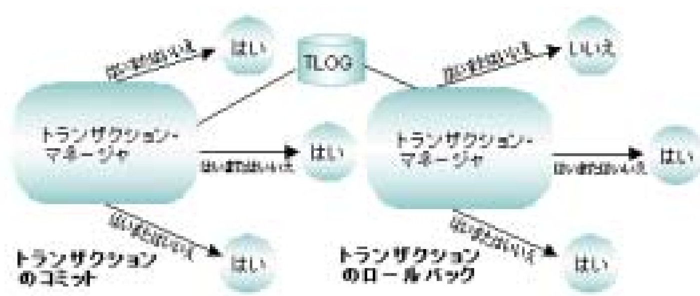
分散トランザクション処理をトラッキングする

BEA Tuxedo のトランザクションは、分散型のアーキテクチャで使用できます。たとえば、トランザクションに関与するローカル・マシンがリモート・マシンと通信し、このリモート・マシンが、さらに別のリモート・マシンと通信することができます。このようなしくみで実行されるトランザクション処理を、分散トランザクション処理と呼びます。

システムでは、いつでもトランザクションをロールバック (初期状態に戻す) できるように、トランザクション情報を保持しておく必要があるため、分散トランザクション処理 (DTP) のトラッキングは複雑になります。この処理を正しく実行するため、BEA Tuxedo システムは、トランザクション・ログ (`TLOG`) と呼ばれる専用のファイルに、トランザクションのパーティシパントすべてのトラッキング情報を格納します。

次の図は、2 つのトランザクション・マネージャ (TM) を使用するアプリケーションの例を示しています。2 つの TM はどちらも同じ `TLOG` にトラッキング・データを記録します。

図 4-1 トランザクション管理



トランザクションをコミットする前に、TM は処理を続行するかどうかを尋ねる質問に繰り返し答える必要があります。必要に応じて、TM はロールバックを行います。

グローバル・トランザクション識別子 (GTRID) を使用してトラッキングする

BEA Tuxedo システムは、同時に実行されるトランザクションを含め、分散システム内で実行されるすべてのトランザクションのフローをトラッキングします。トランザクションのコミット時には、コーディネータ側でトランザクションに参加する RM を認識していなければならないため、各トランザクションは識別できる状態でなければなりません。このため、BEA Tuxedo システムは、各トランザクションにグローバル・トランザクション識別子 (`GTRID`) を割り当てます。

BEA Tuxedo システムは、XA インターフェイスを介して、アプリケーションがアクセスする RM と通信します。RM は、ローカル・トランザクション識別子を割り当てることによってトランザクションをトラッキングし、グローバル識別子をローカル識別子にマッピングします。

トランザクション・ログ (TLOG) を使用してトラッキングする

グローバル・トランザクションは、コミットされるプロセスに含まれる場合にのみ、トランザクション・ログ (`TLOG`) に記録されます。`TLOG` には、2 フェーズ・コミット・プロトコルの第 1 フェーズの最後に、グローバル・トランザクションのパーティシパントからの応答が記録されます。

`TLOG` の記録があるということは、グローバル・トランザクションをコミットしなければならないことを示します。ロールバックされるトランザクションは、`TLOG` に記録されません。

最初の「プリコミット」フェーズでは、各リソース・マネージャでトランザクション要求の実行をコミットします。すべてのリソース・マネージャでトランザクションがコミットされると、トランザクション・マネージャは、第 2 フェーズを実行します。つまり、トランザクションはコミットされ、終了します。アプリケーション障害またはシステムの障害により、どちらかのタスクが失敗すると、タスクは両方共失敗し、実行済みの処理は取り消される、つまり初期状態に「ロールバック」されます。

グローバル・トランザクションの調節を行う TMS では、TLOG ファイルが使用されます。各マシンには、専用の TLOG が必要です。

アプリケーションで Domains コンポーネントを使用している場合は、Domains ゲートウェイが TMS の機能を果たします。ただし、Domains では、Domains 固有の情報のほか、TLOG の内容に似た情報を含む独自のトランザクション・ログを使用します。

2 フェーズ・コミットを使用してトランザクションをコミットする

2 フェーズ・コミットとは、トランザクションのコミットを確実に行うためのアルゴリズムです。

このアルゴリズムのしくみをわかりやすく説明するため、次の状況を例にとります。6 人の仲間が、家を 1 週間借りて休暇を過ごす予定を立てているとします。6 人共、それぞれ賃料の 6 分の 1 の金額しか負担できません。つまり、6 人のうち誰かが参加できないと、家を借りることはできなくなります。

1. この計画の第 1 フェーズでは、幹事が出席を確認し、すべてのメンバから 6 分の 1 の賃料を集金します。1 人でも欠席者が出たら、幹事はグループのすべてのメンバに連絡し、家を借りることができなくなったことを伝えます。ただし、すべてのメンバの出席が確認でき、6 人分の賃料を徴収できれば、第 1 フェーズは成功します。

2. 第 2 フェーズでは、幹事はグループのすべてのメンバに対し、計画が予定どおりに行われることを通知します。

2 フェーズ・トランザクションのコミット処理も、この計画と同じように実行されます。

1. 第 1 フェーズでは、トランザクション・コーディネータがトランザクションに参加する可能性があるパーティシパントにコンタクトします。すべてのパーティシパントは、トランザクションの結果の確定に同意し

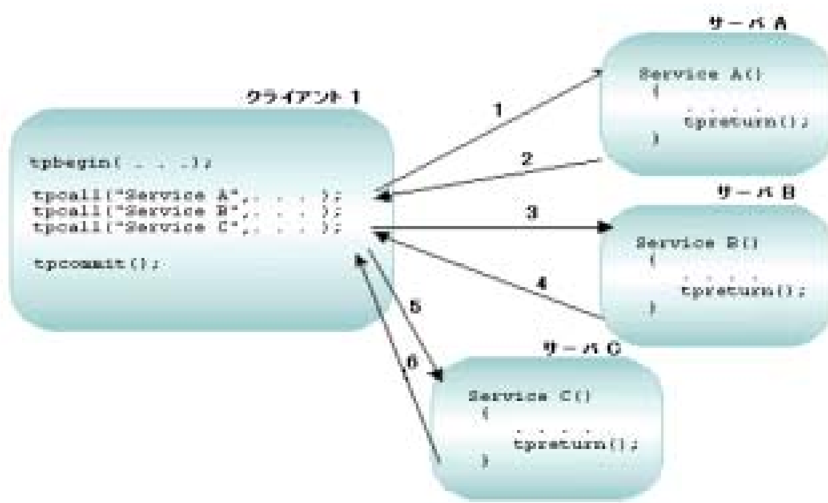
ますが、すぐには行いません。パーティシパントは、情報をディスクに記録し、第 2 フェーズを完了できるようにします。すべてのパーティシパントがコミットに同意すると、トランザクション・コーディネータはそれを記録し、結果は確定されます。パーティシパントの同意がログに記録されると、第 1 フェーズは終了します。

2. 第 2 フェーズでは、コーディネータは各パーティシパントに決定を通知し、リソースへの更新を確定します。

トランザクションの影響を処理する

アプリケーション・モジュールが別のモジュールから呼び出され、トランザクションに参加することを、トランザクションの影響といいます。アプリケーション・モジュールが影響を受けると、BEA Tuxedo システムはすべてのパーティシパントをトラッキングして、第 2 フェーズに関連付けるパーティシパントを決定します。次の図は、システムがパーティシパントをトラッキングする方法を示します。

図 4-2 トランザクション上の影響



この図では、クライアント 1 がトランザクションを開始し、A、B、C の 3 つのサービスを呼び出しています。サービス A、B、C は、トランザクションの開始後に呼び出されたため、トランザクションの影響を受けています。サーバ A、B、C によって実行される作業はすべて、クライアント 1 によって開始されたトランザクションの一部です。すべての作業は、1 つの単位で実行され、すべてが成功するか、または失敗して `tpabort` の呼び出しによりロールバックされるかのどちらかになります。トランザクションが失敗すると、初期状態に戻り、リソース・マネージャ上のトランザクションの結果は元に戻されます。トランザクション内で認識されていないリソース・マネージャや、トランザクション外からアクセスされたリソース・マネージャは、ロールバックできません。

ATMI を使用して 2 フェーズ・コミットの前にトランザクションの整合性を確保する

トランザクションに関わる各リソースで実行される作業は、2 フェーズ・コミットを開始する前に、すべて終了していなければなりません。ATMI を使用すると、2 フェーズ・コミット・プロトコルの開始時に、すべてのトランザクション作業を中止できます。

次に、2 フェーズ・コミットを実行する前に、ATMI を使用してトランザクション処理を中止する手順を示します。

1. Client_1 は `tpbegin()` を使ってトランザクションを開始します。
2. Client_1 は、`tpcall()` を使用して Service_A を呼び出します。Service_A の特徴は、以下のとおりです。
   a. トランザクションの影響を受けます。
   b. 操作を実行します。
   c. `tpreturn()` を呼び出します。
   d. トランザクション処理を終了します。
3. Client_1 は、`tpcall()` を使用して Service_B を呼び出します。Service_B の特徴は、以下のとおりです。

   a. トランザクションの影響を受けます。
   b. 操作を実行します。
   c. `tpreturn()` を呼び出します。
   d. トランザクション処理を終了します。

4. Client_1 は、`tpcall()` を使用して Service_C を呼び出します。Service_C の特徴は、以下のとおりです。
   a. トランザクションの影響を受けます。
   b. 操作を実行します。
   c. `tpreturn()` を呼び出します。
   d. トランザクション処理を終了します。
5. Client_1 は、`tpcommit()` を使用して、コミットのプロセスを開始します。

トランザクションの処理中に、呼び出されたサービスが別のサービスを実行しているか、またはオープン中の会話に関連付けられている場合、ATMI はそのアクティビティをトラッキングし、アクティビティが完了するまでプロセスがコミットされないようにします。

ATMI は、呼び出されたすべてのサービスによるトランザクション処理が成功した場合にのみ、トランザクションをコミットします。すべての処理が成功すると、トランザクション・マネージャは、トランザクション内で実行されたすべての更新が確定されたことをリソース・マネージャに通知します。

関連項目

- 5-2 ページの「UBBCONFIG ファイルを ATMI トランザクションに対応させて変更する」
- 5-12 ページの「トランザクションをサポートするように Domains コンフィギュレーション・ファイルを変更する」
- 5-16 ページの「例 : トランザクションを使用する分散アプリケーション」

- 『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「グローバル・トランザクションのコーディング」
- 『BEA Tuxedo システム入門』の「ATMI の使用」
- BEA Tuxedo CORBA 環境でトランザクションを使用する方法については、『BEA Tuxedo CORBA トランザクション』を参照してください。

5 トランザクション対応の ATMI アプリケーションのコンフィギュレーション

ここでは、次の内容について説明します。

- UBBCONFIG ファイルを ATMI トランザクションに対応させて変更する
- RESOURCES セクションでグローバル・トランザクションのパラメータを指定する
- MACHINES セクションでトランザクション・ログ (TLOG) を作成する
- GROUPS セクションでリソース・マネージャとトランザクション・マネージャ・サーバを定義する
- SERVICES セクションでサービスに対するトランザクションの開始を有効にする
- トランザクションをサポートするように Domains コンフィギュレーション・ファイルを変更する
- 例 : トランザクションを使用する分散アプリケーション

注記　BEA Tuxedo CORBA 環境でトランザクションを使用する方法については、『BEA Tuxedo CORBA トランザクション』を参照してください。

UBBCONFIG ファイルを ATMI トランザクションに対応させて変更する

アプリケーションの UBBCONFIG ファイルをトランザクションに対応させるには、RESOURCES、MACHINES、GROUPS、および SERVICES セクションを変更する必要があります。

セクション名	変更内容
RESOURCES	アプリケーションに設定できるトランザクションの数、およびコミット制御フラグの値を指定します。
MACHINES	各マシンの TLOG 情報を指定します。
GROUPS	各リソース・マネージャおよびトランザクション・マネージャ・サーバに関する情報を指定します。
SERVICES	自動トランザクション・オプションを有効にします。

RESOURCES セクションでグローバル・トランザクションのパラメータを指定する

次の表では、RESOURCES セクションに指定するトランザクション関連のパラメータを説明します。

パラメータ名	**説明**
MAXGTT	一度に 1 台のマシン上で設定できるグローバル・トランザクション識別子 (GTRID) の数を制限します。設定可能な最大値は 2048、最小値は 0、デフォルト値は 100 です。MAXGTT の値は、MACHINES セクションでマシンごとに上書きできます。 グローバル・トランザクションがアクティブな間だけ、テーブル内にそのエントリがあるため、このパラメータには同時トランザクションの数に制限を設ける効果があります。

パラメータ名	**説明**
CMTRET	TP_COMMIT_CONTROL 特性の初期設定が次のいずれかであることを示します。 ■ LOGGED—TP_COMMIT_CONTROL 特性は TP_CMT_LOGGED に設定されます。すべてのパーティシパントが正常にプリコミットを行った場合に tpcommit() が返されることを示します。 ■ COMPLETE—TP_COMMIT_CONTROL 特性は TP_CMT_COMPLETE に設定されます。すべてのパーティシパントが正常にコミットするまで tpcommit() が返されないことを示します。 デフォルト値は COMPLETE です。 適切な設定値については、リソース・マネージャ (RM) のベンダに問い合わせてください。XA 標準のレイト・コミット実装を使用するリソース・マネージャがある場合、COMPLETE に設定します。すべてのリソース・マネージャがアーリー・コミット実装を使用する場合は、性能上の理由により LOGGED に設定します。この設定は tpscmt() を使用して上書きできます。
MAXTRANTIME	トランザクションの最長のタイムアウト値を指定します。 有効な値は 0 ～ 2,147,483,647 の範囲です。 0 はトランザクションのタイムアウト値に制限がないことを表します。 デフォルトは 0 です 注記　MAXTRANTIME の詳細については、UBBCONFIG(5) の RESOURCES セクションにある MAXTRANTIME、または TM_MIB(5) の T_DOMAIN クラスにある TA_MAXTRANTIME を参照してください。

MACHINES セクションでトランザクション・ログ (TLOG) を作成する

`TLOG` を作成するには、以下のタスクを実行します。

- 汎用デバイス・リスト (UDL: Universal Device List) を作成します。
- `MACHINES` セクションでトランザクション関連のパラメータを定義します。
- Domains トランザクション・ログを作成します。

UDL を作成する

汎用デバイス・リスト (UDL: Universal Device List) は、BEA Tuxedo ファイルシステムの地図のようなものです。UDL は、アプリケーションの起動後、共用メモリにロードされます。`TLOG` は、トランザクションが終了するまで、トランザクションに関する情報が保持されているログを参照します。UDL 内に `TLOG` デバイス用のエントリを作成するには、グローバル・トランザクションを使用して各マシンに UDL を作成します (`TLOGDEVICE` が 2 台のマシン間でミラーリングされる場合は、一方のマシンに UDL を作成するだけで十分です)。起動時に、BBL は `TLOG` を初期化し、オープンします。

UDL を作成するには、アプリケーションを起動する前に次のコマンドを実行します。

```
tmadmin -c crdl -z config -b blocks
```

注記　デバイスが既に存在する場合、このコマンドは失敗します。

`config` の値は、UDL の作成先デバイスの絶対パス名です。これは、コンフィギュレーション・ファイルの `MACHINES` セクションにある `TLOGDEVICE` パラメータの値と一致している必要があります。*`blocks`* には、デバイス上に割り当てるブロックの数を指定します。

注記　*blocks* の値が TLOGSIZE の値より小さいと、性能が低下する可能性があります。したがって、*blocks* には TLOGSIZE より大きい値を指定してください。たとえば、TLOGSIZE に 200 ブロックが指定されている場合、-b500 を指定すると、性能は低下しません。

TLOG の保存方法については、『BEA Tuxedo システムのインストール』を参照してください。

MACHINES セクションでトランザクション関連のパラメータを定義する

グローバル・トランザクション・ログ (TLOG) を定義するには、UBBCONFIG ファイルの MACHINES セクションでパラメータをいくつか設定する必要があります。

これらのパラメータのうち、TLOGDEVICE については、TLOG が必要な各マシン上に TLOGDEVICE のデバイス・リスト・エントリを手動で作成しなければなりません。これは、TUXCONFIG がロードされる前でも後でも作成できますが、必ず、システムを起動する前に作成してください。

次の表では、MACHINES セクションに指定するトランザクション関連のパラメータを説明します。

パラメータ名	説明
TLOGNAME	このマシンの DTP トランザクション・ログの名前を指定します。
TLOGDEVICE	このマシンの DTP トランザクション・ログ (TLOG) を含む BEA Tuxedo ファイルシステムを指定します。このパラメータを指定しないと、マシンには TLOG がないと見なされます。値に使用できる文字数は、最大 64 文字です。

パラメータ名	説明
TLOGSIZEE	TLOG ファイルのサイズを指定します (物理ページ単位)。この値には、1 〜 2048 の範囲の値を設定します。デフォルト値は 100 です。指定した時点で、未処理トランザクションを十分保持できるマシン上のサイズを指定します。1 ページにつき 1 つのトランザクションがログに記録されます。ほとんどのアプリケーションではデフォルト値で十分です。
TLOGOFFSET	TLOGDEVICE の始めから、このマシンのトランザクション・ログを含む VTOC の開始までのページのオフセット (ページ数)。この値は 0 以上で、デバイス上のページ数より小さい値でなければなりません。デフォルト値は 0 です。 TLOGOFFSET が必要になることはほとんどありません。ただし、2 つの VTOC が同じデバイスを共有したり、VTOC が別のアプリケーションに共有されるデバイス (ファイルシステムなど) 上に格納される場合は、TLOGOFFSET を使用してデバイスのアドレスに関連する開始アドレスを指定できます。

Domains トランザクション・ログを作成する

Domains ゲートウェイ・グループを起動する前に、Domains トランザクション・ログを作成する必要があります。Domains トランザクション・ログは、現在のマシン (DMADM を実行中のマシン) の指定されたローカル・ドメインに作成してください。ログを作成するには、次のコマンドを入力します。

```
dmadmin crdmlog crdlog -d local_domain_name
```

このコマンドでは、DMCONFIG ファイルで指定したパラメータが使用されます。指定したローカル・ドメインが現在のマシン上でアクティブであるか、またはログがすでに存在する場合、このコマンドは失敗します。トランザクション・ログがない場合は、Domains ゲートウェイ・グループの起動時に作成されます。

関連項目

- 『BEA Tuxedo アプリケーション実行時の管理』の 2-20 ページの「トランザクション・ログ (TLOG) とは」

GROUPS セクションでリソース・マネージャとトランザクション・マネージャ・サーバを定義する

`GROUPS` セクションのパラメータにより、特定のグループに対してトランザクション・マネージャ・サーバ (TMS) およびリソース・マネージャ (RM) の属性を定義できます。

- TMS は、グローバル・トランザクションを制御するほとんどの作業を行うサーバです。このサーバに対しては、次のパラメータを定義します。
  - `TMSNAME` には、エントリで定義済みのグループに関連付けられた、トランザクション・マネージャ・サーバ用の実行可能ファイル名を指定します。BEA Tuxedo システムには、トランザクションに参加しても RM は使用しないグループ用に、ヌル・トランザクション・マネージャ・サーバ (TMS) が用意されています。この TMS サーバは、どの RM とも通信しません。RM とは通信せず、トランザクションの管理だけを行います。
  - `TMSCOUNT` には、起動する TMS の数 (最小 2、最大 10、デフォルト 3) を指定します。
- 各リソース・マネージャに対しては、`OPENINFO` および `CLOSEINFO` パラメータを定義します。これらのパラメータの値は、リソース・マネージャをオープンまたはクローズするために必要な情報を含む文字列です。適切なパラメータ値は、リソース・マネージャのベンダ側で用意されます。たとえば、リソース・マネージャとして Oracle データベースを使用している場合、次のような値を指定します。

```
OPENINFO=”ORACLE_XA:
Oracle_XA+Acc=P/Scott/*****+SesTm=30+LogDit=/tmp”
```

GROUPS セクションの例

以下は、BEA Tuxedo システムに同梱されている銀行業務サンプル・アプリケーション bankapp の GROUPS セクションの例です。

```
BANKB1 GRPNO=1 TMSNAME=TMS_SQL TMSCOUNT=2
OPENINFO=”TUXEDO/SQL:APPDIR/bankdl1:bankdb:readwrite”
```

GROUPS セクションの例でのトランザクション値

次の表では、GROUPS セクションの例のトランザクション値を説明します。

トランザクション値	目的
BANKB1 GRPNO=1 TMSNAME=TMS_SQL TMSCOUNT=2	トランザクション・マネージャ・サーバの名前 (TMS_SQL) と、グループ BANKB1 で起動するサーバの数 (2) を指定します。
TUXEDO/SQL	リソース・マネージャの公開名です。
APPDIR/bankdl1	デバイス名です。
bankdb	データベース名です。
readwrite	アクセス・モードです。

TMSNAME、TMSCOUNT、OPENINFO、および CLOSEINFO パラメータの特性

次の表では、TMSNAME、TMSCOUNT、OPENINFO、および CLOSEINFO パラメータの特性を説明します。

パラメータ名	説明
TMSNAME	実行可能なトランザクション・マネージャ・サーバの名前です。 トランザクション対応のアプリケーションでは、必須パラメータです。 TMS は、ヌル・トランザクション・マネージャ・サーバです。
TMSCOUNT	トランザクション・マネージャ・サーバの数。2 〜 10 の範囲の値を設定します。 デフォルト値は 3 です。このパラメータは省略できます。
OPENINFO、CLOSEINFO	リソース・マネージャをオープンまたはクローズするために必要な情報です。 内容は、リソース・マネージャによって異なります。 値はリソース・マネージャの名前で始まります。 この値を省略すると、RM がオープンまたはクローズするための情報を必要としないことを示します。

SERVICES セクションでサービスに対するトランザクションの開始を有効にする

状況によっては、SERVICES セクションに AUTOTRAN、TRANTIME、および ROUTING という 3 つのトランザクション関連パラメータを設定しなければならない場合があります。

- クライアントの代わりに、サービスにトランザクションを開始させる場合は、AUTOTRAN フラグを Y に設定する必要があります。この設定は、サービスが大規模なトランザクションの一部でない場合や、トランザクションの決定に関わるクライアント側の負担を軽減したい場合に役立ち

ます。既存のトランザクションがある場合にサービスが呼び出されると、この呼び出しは既存のトランザクションの一部になります。デフォルト値は N です。

注記　一般的に、サービスは大規模なトランザクションに参加するため、トランザクションのイニシエータとしてはクライアントが最も適しています。

- AUTOTRAN が Y に設定されている場合、TRANTIME パラメータを設定する必要があります。これはトランザクション作成時のタイムアウト値です。0 ～ 2,147,483,647 (2^{31} - 1、つまり約 70 年) の値を指定してください。0 は、トランザクションにタイムアウトが設定されていないことを示します。デフォルト値は 30 秒です。
- データ依存型ルーティングを使用するトランザクションには、ROUTING パラメータを定義する必要があります。

AUTOTRAN、TRANTIME、および ROUTING パラメータの特性

次の表では、AUTOTRAN、TRANTIME、および ROUTING パラメータの特性を説明します。

パラメータ名	説明
AUTOTRAN	トランザクションのイニシエータとしてサービスを指定します。 このパラメータを正しく機能させるには、アプリケーション設計者とアプリケーション管理者の間のコミュニケーションが重要です。開発者が設定した ICF パラメータを知らずに管理者がこの値を Y に設定すると、アプリケーションの誤動作や失敗が生じる可能性があります。 トランザクションがすでに存在している場合、新しいトランザクションは開始しません。 デフォルト値は N です。

パラメータ名	説明
TRANTIME	AUTOTRAN トランザクションのタイムアウト値を指定します。 有効な値は 0 ～ 2,147,483,647 の範囲です。 0 は、タイムアウトが発生しないことを示します。 デフォルト値は 30 秒です。
ROUTING	このサービスを要求するトランザクションにデータ依存型ルーティングが指定されている ROUTING セクション内のエントリを指します。

トランザクションをサポートするように Domains コンフィギュレーション・ファイルを変更する

ドメインを介してトランザクションを有効にするには、Domains コンフィギュレーション・ファイル (DMCONFIG) の DM_LOCAL および DM_IMPORT セクション内のパラメータを設定する必要があります。DM_LOCAL セクションのエントリでは、ローカル・ドメインの特性を定義します。DM_IMPORT セクションのエントリでは、インポートされたサービスや、リモート・ドメインで使用可能なサービスを定義します。

DMTLOGDEV、DMTLOGNAME、DMTLOGSIZE、MAXRAPTRAN、および MAXTRAN パラメータの特性

Domains コンフィギュレーション・ファイルの DM_LOCAL セクションでは、ローカル・ドメインおよびそれに関連するゲートウェイ・グループを指定します。ゲートウェイ・グループ (ローカル・ドメイン) ごとに、そのグループで実行するドメイン・ゲートウェイ・プロセスに必要なパラメータを指定するエントリを作成する必要があります。

次の表では、このセクションの 5 つのトランザクション関連パラメータ (DMTLOGDEV、DMTLOGNAME、DMTLOGSIZE、MAXRAPTRAN、および MAXTRAN) を説明します。

パラメータ名	説明
DMTLOGDEV	このマシンの Domains トランザクション・ログ (DMTLOG) を含む BEA Tuxedo ファイルシステムを指定します。DMTLOG は、TLOGDEVICE (BEA Tuxedo ファイルシステム) 上に BEA Tuxedo の VTOC テーブルとして格納されます。このパラメータを指定しない場合、Domains ゲートウェイ・グループは要求をトランザクション・モードで処理できません。同じマシン上で実行するローカル・ドメインは、同じ DMTLOGDEV ファイルシステムを共用できますが、ローカル・ドメインごとに個別のログ (DMTLOGDEV) を作成する必要があります。各ログの名前は、DMTLOGNAME パラメータによって決まります。
DMTLOGNAME	このドメインの Domains トランザクション・ログの名前を指定します。このドメインがほかのローカル・ドメインと同じファイルシステム上にある場合 (DMTLOGDEV の値が同じ)、DMTLOGNAME の値はログごとに一意でなければなりません。値には 30 文字以内の文字を使用できます。デフォルトは DMTLOG です。

パラメータ名	説明
DMTLOGSIZE	このマシンの Domains トランザクション・ログのサイズを指定します (ページ数単位)。0 より大きく、BEA Tuxedo ファイルシステム上の空き領域より小さい値を指定します。デフォルトは 100 ページです。 注記　トランザクション内のドメインの数により、DMTLOGSIZE パラメータで指定するページ数が決まります。トランザクションとログのページ数は、1 対 1 でマッピングされません。
MAXRAPTRAN	トランザクションに含めることのできるドメインの最大数を指定します。ゼロより大きく、32,768 未満の値を指定します。デフォルト値は 16 です。
MAXTRAN	このローカル・ドメイン上で同時に実行できるグローバル・トランザクションの最大数を指定します。0 以上で、コンフィギュレーション・ファイルに定義されている MAXGTT パラメータ以下の値を指定します。デフォルト値は、MAXGTT の値です。

AUTOTRAN および TRANTIME パラメータの特性

Domains コンフィギュレーション・ファイルの DM_IMPORT セクションでは、インポートされ、リモート・ドメインで使用可能になったサービスに関する情報を提供します。各リモート・サービスは、特定のリモート・ドメインに関連付けられています。

DM_IMPORT セクションに、トランザクションをサポートする AUTOTRAN パラメータと TRANTIME パラメータを設定することもできます。次の表では、これらのパラメータを説明します。

パラメータ名	説明
AUTOTRAN	リモート・サービスのトランザクションを自動的に開始 / 終了するためにゲートウェイが使用します。この機能は、リモート・サービスに対して、信頼性のあるネットワーク通信を行う場合に必要です。この機能を要求するには、対応するリモート・サービスのエントリで AUTOTRAN パラメータを Y に設定します。
TRANTIME	定義されたサービスに対して自動的に開始されるトランザクションのタイムアウト値を秒単位で指定します。0 以上 2,147,483,648 未満の値を指定します。0 は、マシンの最大タイムアウト値を示します。デフォルト値は 30 秒です。

UBBCONFIG ファイルの RESOURCES セクションでは、MAXTRANTIME という追加のトランザクション・タイムアウト・プロパティも使用できます。MAXTRANTIME タイムアウト値が TRANTIME タイムアウト値、またはトランザクションを開始する際に tpbegin(3c) 呼び出しで渡されたタイムアウト値よりも小さい場合、トランザクションのタイムアウトは MAXTRANTIME 値まで減少します。MAXTRANTIME は BEA Tuxedo 8.0 以前を実行するマシン上で開始されるトランザクションには影響を与えません。ただし、BEA Tuxedo 8.1 以降を実行するマシンがトランザクションの影響を受ける場合は、そのノードに対して設定されている MAXTRANTIME 値までトランザクション・タイムアウト値が制限 (必要に応じて減少) されます。

Domains コンフィギュレーションの場合は、以下のようなトランザクション処理のシナリオが考えられます。

- ドメイン間トランザクションで MAXTRANTIME パラメータを認識しないノードが影響を受ける場合、または MAXTRANTIME パラメータを認識するがパラメータが設定されていない場合、トランザクションのタイムアウト値は TRANTIME、またはトランザクションを開始した tpbegin() 呼び出しで渡されたタイムアウト値によって決まります。TRANTIME または tpbegin() のタイムアウト値を超えると、トランザクションの影響を受けるすべての BEA Tuxedo ノード (トランザクションを開始したノードも含む) は TMS タイムアウト・メッセージを生成します。
- ドメイン間トランザクションで MAXTRANTIME パラメータを認識するノードが影響を受け、そのノードに対してパラメータが設定されている場合、トランザクションのタイムアウト値はそのノードの MAXTRANTIME 値と同じ値に減少されます。

  TRANTIME または tpbegin() のタイムアウト値が MAXTRANTIME 以下の場合、トランザクション処理のシナリオは前述のようになります。

  TRANTIME または tpbegin() のタイムアウト値が MAXTRANTIME より大きい場合、影響を受けるノードはトランザクションのタイムアウト値を MAXTRANTIME まで減少します。MAXTRANTIME タイムアウト値を超えると、影響を受けるノードは TMS タイムアウト・メッセージを生成します。

MAXTRANTIME の詳細については、UBBCONFIG(5) の RESOURCES セクションにある MAXTRANTIME、または TM_MIB(5) の T_DOMAIN クラスにある TA_MAXTRANTIME を参照してください。

例 : トランザクションを使用する分散アプリケーション

この節では、3 つのサイトに分散される、トランザクション対応の bankapp アプリケーションを定義するコンフィギュレーション・ファイル例を示します。このアプリケーションには、次の特徴があります。

- ACCOUNT_ID に対してデータ依存型ルーティングが実行されます。
- データは 3 つのデータベースに分散されます。
- ATMI インターフェイスを使用してシステムと通信する BRIDGE プロセスが実行されます。
- 1 つのサイトからアプリケーションが管理されます。

ファイルは、7 つのセクション (RESOURCES、MACHINES、GROUPS、NETWORK、SERVERS、SERVICES、および ROUTING) で構成されます。

RESOURCES セクションの例

次のリストは、RESOURCES セクションの例です。

コード リスト 5-1 RESOURCES **セクションの例**

```
*RESOURCES
#
IPCKEY       99999
UID          1
GID          0
PERM         0660
MAXACCESSERS 25
MAXSERVERS   25
MAXSERVICES  40
MAXGTT       20
MASTER       SITE3, SITE1
SCANUNIT     10
SANITYSCAN   12
BBLQUERY     180
BLOCKTIME    30
DBBLWAIT     6
OPTIONS      LAN, MIGRATE
MODEL        MP
LDBAL        Y
```

このリストでは、次の点に注意してください。

- MAXSERVERS、MAXSERVICES、および MAXGTT は、デフォルト値より小さい値に設定されています。これは、掲示板のサイズを小さくするためです。
- MASTER は SITE3 で、バックアップ・マスタは SITE1 です。
- MODEL が MP に設定されており、OPTIONS が LAN、MIGRATE に設定されているため、ネットワーク接続されたコンフィギュレーションを使用して移行を行うことができます。
- BBLQUERY が 180 に設定されており、SCANUNIT が 10 に設定されているため、DBBL はリモートの BBL を 1800 秒おき (30 分おき) にチェックします。

MACHINES セクションの例

次のリストは、MACHINES セクションの例です。

コード リスト 5-2 MACHINES **セクションの例**

```
*MACHINES
giselle       LMID=SITE1
              TUXDIR=”/usr/tuxedo”
              APPDIR=”/usr/home”
              ENVFILE=”/usr/home/ENVFILE”
              TLOGDEVICE=”/usr/home/TLOG”
              TLOGNAME=TLOG
              TUXCONFIG=”/usr/home/tuxconfig”
              TYPE=”3B600”

romeo         LMID=SITE2
              TUXDIR=”/usr/tuxedo”
              APPDIR=”/usr/home”
              ENVFILE=”/usr/home/ENVFILE”
              TLOGDEVICE=”/usr/home/TLOG”
              TLOGNAME=TLOG
              TUXCONFIG=”/usr/home/tuxconfig”
              TYPE=”SEQUENT”

juliet        LMID=SITE3
              TUXDIR=”/usr/tuxedo”
              APPDIR=’/usr/home”
              ENVFILE=”/usr/home/ENVFILE”
              TLOGDEVICE=”/usr/home/TLOG”
              TLOGNAME=TLOG
              TUXCONFIG=”/usr/home/tuxconfig”
              TYPE=”AMDAHL”
```

このリストでは、次の点に注意してください。

- TLOGDEVICE および TLOGNAME が指定され、トランザクションが行われることを示します。

- TYPE パラメータはすべて異なりますが、これはマシン間で送られるすべてのメッセージに符号化 / 復号化が行われることを示します。

GROUPS セクションおよび NETWORK セクションの例

次のリストは、GROUPS セクションおよび NETWORK セクションの例です。

コード リスト 5-3 GROUPS **セクションおよび** NETWORK **セクションの例**

```
*GROUPS
DEFAULT:         TMSNAME=TMS_SQL       TMSCOUNT=2
BANKB1           LMID=SITE1            GRPNO=1
 OPENINFO=”TUXEDO/SQL:/usr/home/bankdl1:bankdb:readwrite”
BANKB2           LMID=SITE2            GRPNO=2
 OPENINFO=”TUXEDO/SQL:/usr/home/bankdl2:bankdb:readwrite”
BANKB3            LMID=SITE3            GRPNO=3
 OPENINFO=”TUXEDO/SQL:/usr/home/bankdl3:bankdb:readwrite”

*NETWORK
SITE1             NADDR=”0X0002ab117B2D4359”
                  BRIDGE=”/dev/tcp”
                  NLSADDR=”0X0002ab127B2D4359”

SITE2             NADDR=”0X0002ab117B2D4360”
                  BRIDGE=”/dev/tcp”
                  NLSADDR=”0X0002ab127B2D4360”

SITE3             NADDR=”0X0002ab117B2D4361”
                  BRIDGE=”/dev/tcp”
                  NLSADDR=”0X0002ab127B2D4361”
```

このリストでは、次の点に注意してください。

- TMSCOUNT は 2 に設定されますが、これはグループあたり 2 つの TMS_SQL トランザクション・マネージャ・サーバだけが起動されることを示します。
- OPENINFO 文字列は、アプリケーションがデータベース・アクセスを行うことを示します。

SERVERS、SERVICES、および ROUTING セクションの例

次のリストは、SERVERS、SERVICES、および ROUTING セクションの例です。

コード リスト 5-4 SERVERS、SERVICES、**および** ROUTING **セクションの例**

```
*SERVERS
DEFAULT: RESTART=Y MAXGEN=5 REPLYQ=N CLOPT=”-A”
TLR      SRVGRP=BANKB1    SRVID=1    CLOPT=”-A -- -T 100"
TLR      SRVGRP=BANKB2    SRVID=3    CLOPT=”-A -- -T 400"
TLR      SRVGRP=BANKB3    SRVID=4    CLOPT=”-A -- -T 700"
XFER     SRVGRP=BANKB1    SRVID=5    REPLYQ=Y
XFER     SRVGRP=BANKB2    SRVID=6    REPLYQ=Y
XFER     SRVGRP=BANKB3    SRVID=7    REPLYQ=Y

*SERVICES
DEFAULT:    AUTOTRAN=N
WITHDRAW        ROUTING=ACCOUNT_ID
DEPOSIT         ROUTING=ACCOUNT_ID
TRANSFER        ROUTING=ACCOUNT_ID
INQUIRY         ROUTING=ACCOUNT_ID

*ROUTING
ACCOUNT_ID      FIELD=ACCOUNT_ID     BUFTYPE=”FML”
                     RANGES=”MON - 9999:*,
                     10000 - 39999:BANKB1
                     40000 - 69999:BANKB2
                     70000 - 100000:BANKB3
                        “”
```

このリストでは、次の点に注意してください。

- TLR サーバが tpsvrinit() 関数を呼び出す場合は、-T オプションに 100、400、または 700 を指定します。
- すべてのサービス要求は ACCOUNT_ID フィールドでルーティングされます。

- `AUTOTRAN` モードではサービスは行われません。

関連項目

- 4-1 ページの「トランザクションとは」
- 『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』の 9-1 ページの「グローバル・トランザクションのコーディング」
- 『BEA Tuxedo システム入門』の「ATMI の使用」
- BEA Tuxedo CORBA 環境でトランザクションを使用する方法については、『BEA Tuxedo CORBA トランザクション』を参照してください。

6 CORBA インターフェイス・リポジトリの管理

この章は、BEA Tuxedo CORBA 環境にのみ適用されます。以下の節で構成されます。

- 管理上の注意
- 管理コマンドを使用してインターフェイス・リポジトリを管理する
- 1 つ以上のインターフェイス・リポジトリ・サーバを起動するように UBBCONFIG ファイルを設定する

インターフェイス・リポジトリには、BEA Tuxedo ドメインでインプリメントされる CORBA オブジェクトのインターフェイス記述が格納されています。インターフェイス・リポジトリは、BEA Tuxedo CORBA サーバ固有のツールを使用して管理します。これらのツールを使用すると、インターフェイス・リポジトリを作成したり、OMG (Object Management Group) の IDL (インターフェイス定義言語) の定義に準拠した設定を行ったり、インターフェイスの削除を行うことができます。インターフェイス・リポジトリ・サーバをシステムに追加するには、アプリケーションの `UBBCONFIG` ファイルにエントリを追加することが必要になる場合があります。

プログラミングの関連情報については、『BEA Tuxedo CORBA プログラミング・リファレンス』を参照してください。

管理上の注意

管理者は、インターフェイス・リポジトリが必要かどうかを判断する必要があります。インターフェイス・リポジトリは、すべてのシステムで必要なわけではないためです。インターフェイス・リポジトリが必要な場合は、まずリポジトリ・データベースを作成し、必要な設定を行います。リポジトリ・データベースの作成と設定には、`idl2ir` コマンドを使用します。

インターフェイス・リポジトリが必要な場合は、以下の点を確認してください。

- 必要なインターフェイス・リポジトリ・サーバの数
- インターフェイス・リポジトリ・データベースを複製するかどうか
- インターフェイス・リポジトリ・データベースへの共有アクセスを許可するかどうか
- インターフェイス・リポジトリの更新方法

システムには、1 つ以上のインターフェイス・リポジトリ・サーバを設定することができます。動的起動インターフェイス (DII) または ActiveX を使用するクライアントがある場合は、少なくとも 1 つのインターフェイス・リポジトリ・サーバを設定する必要があります。

複数のサーバを設定すると、パフォーマンスと障害耐久性を高めることができます。パフォーマンスを向上させるには、インターフェイス・リポジトリ・サーバの数を DDI および ActiveX クライアントの数に基づいて決めます。また、障害耐久性を高めるには、システムのコンフィギュレーションと要求される障害耐久性のレベルに基づいて、インターフェイス・リポジトリ・サーバの数を決める必要があります。

複数のインターフェイス・リポジトリ・サーバが設定されたシステムでは、データベースを複製するかどうか、データベースを共有するかどうか、またはそれらを両方とも行うかどうかを決定する必要があります。どちらの設定方法にもそれぞれメリットとデメリットがあります。インターフェイス・リポジトリ・データベースを複製すると、ローカル・ファイル・アクセスが可能になるので、パフォーマンスを向上できます。

しかし、複製したデータベースをどのように更新するかが重大な問題となります。すべてのデータベースは常に同一でなければならず、そのためにインターフェイス・リポジトリ・サーバの起動と停止が必要になります。インターフェイス・リポジトリ・データベースをマウントして共有するとこの問題を解決できますが、パフォーマンスに影響し、シングル・ポイント障害が発生する可能性があります。複製と共有の 2 つの方法を組み合わせることも可能です。

管理コマンドを使用してインターフェイス・リポジトリを管理する

以下のコマンドを使用して、BEA Tuxedo ドメインのインターフェイス・リポジトリを管理します。

- `idl2ir`
- `ir2idl`
- `irdel`

前提条件

コマンドを実行する前に、定義したパスに `bin` ディレクトリがあることを確認します。以下はパスの例です。

Windows 2000 の場合 :

```
set path=%TUXDIR%\bin;%path%
```

UNIX の場合 :

C シェル (csh) の場合 :`set path = ($TUXDIR/bin $path)`

Bourne (sh) または Korn (ksh) シェルの場合 :
```
PATH=$TUXDIR/bin:$PATH
export PATH
```

環境変数を設定するには :

Windows 2000 の場合 :

```
set var=value
```

UNIX の場合 :

C シェルの場合 :

```
setenv var value
```

Bourne および Korn シェル (sh/ksh) の場合 :

```
var=value
export var
```

インターフェイス・リポジトリを作成して設定する

インターフェイス・リポジトリを作成し、インターフェイス定義をロードするには、idl2ir コマンドを使用します。このコマンドは、リポジトリ・ファイルが存在しない場合は作成し、既に存在する場合は指定されたインターフェイス定義をそのファイルにロードします。このコマンドの形式は次のとおりです。

```
idl2ir [options] definition-filename-list
```

このコマンドの詳細については、BEA Tuxedo のオンライン・マニュアルの『BEA Tuxedo のファイル形式とデータ記述方法』を参照してください。

注記　変更結果を確認するには、インターフェイス・リポジトリ・サーバを再起動する必要があります。

インターフェイス・リポジトリの内容を表示または抽出する

インターフェイス・リポジトリの内容を表示するには、`ir2idl` コマンドを使用します。このコマンドを使用して、複数のインターフェイスの OMG IDL ステートメントをファイルに抽出することもできます。このコマンドの形式は次のとおりです。

```
ir2idl [options] [interface-name]
```

このコマンドの詳細については、BEA Tuxedo のオンライン・マニュアルの『BEA Tuxedo のファイル形式とデータ記述方法』を参照してください。

インターフェイス・リポジトリからオブジェクトを削除する

インターフェイス・リポジトリから指定したオブジェクトを削除するには、`irdel` コマンドを使用します。削除できるのは、ほかのインターフェイスから参照されていないインターフェイスだけです。デフォルトのリポジトリ・ファイルは `repository.ifr` です。このコマンドの形式は次のとおりです。

```
irdel [-f repository-name]  [-i id] object-name
```

このコマンドの詳細については、BEA Tuxedo のオンライン・マニュアルの『BEA Tuxedo のファイル形式とデータ記述方法』を参照してください。

注記　変更結果を確認するには、インターフェイス・リポジトリ・サーバを再起動する必要があります。

1 つ以上のインターフェイス・リポジトリ・サーバを起動するように UBBCONFIG ファイルを設定する

1 つ以上のインターフェイス・リポジトリを使用する各アプリケーションに対し、Tuxedo CORBA のインターフェイス・リポジトリ・サーバを 1 つ以上起動する必要があります。サーバ名は TMIFRSVR です。この TMIFRSVR のエントリを、アプリケーションの UBBCONFIG ファイルの SERVERS セクションに追加します。

デフォルトでは、TMIFRSVR サーバは、APPDIR 環境変数で指定された最初のパス名にあるインターフェイス・リポジトリ・ファイル repository.ifr を使用します。このデフォルト設定は、コマンド行オプション (CLOPT) パラメータで -f filename オプションを指定して上書きすることができます。

次の例は、UBBCONFIG サンプル・ファイルの SERVERS セクションを示しています。この例では、アプリケーションのデフォルトのインストール先ディレクトリ ($APPDIR) にあるデフォルト・ファイル repository.ifr を使用する代わりに、別のファイルと場所 (/usr/repoman/myrepo.ifr) を指定しています。

注記　この例では、BEA Tuxedo CORBA アプリケーションでサーバの起動順序が重要であることを示すため、ほかのサーバ・エントリも記載しています。この順序に従わないと、BEA Tuxedo CORBA アプリケーションは起動しません。

詳細については、第 3 章「コンフィギュレーション・ファイルの作成」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。

インターフェイス・リポジトリ・サーバの TMIFRSVR は、5 番目に起動されます。

```
*SERVERS

    # BEA Tuxedo System Event Broker を起動します。
    TMSYSEVT
        SRVGRP  = SYS_GRP
        SRVID   = 1

    # NameManager ( マスタ ) を起動します。
        SRVGRP  = SYS_GRP
        SRVID   = 2
        CLOPT   = "-A -- -N -M"

    # NameManager ( スレーブ ) を起動します。
    TMFFNAME
        SRVGRP  = SYS_GRP
        SRVID   = 3
        CLOPT   = "-A -- -N"

    # FactoryFinder (-F) を起動します。
    TMFFNAME
        SRVGRP  = SYS_GRP
        SRVID   = 4
        CLOPT   = "-A -- -F"

    # インターフェイス・リポジトリ・サーバを起動します。
    TMIFRSVR
        SRVGRP  = SYS_GRP
        SRVID   = 5
        RESTART=Y
        MAXGEN=5
        GRACE=3600
        CLOPT="-A -- -f /usr/repoman/myrepo.ifr"
```

TMIFRSVR -f filename パラメータの詳細については、『BEA Tuxedo のファイル形式とデータ記述方法』を参照してください。TMIFRSVR パラメータには、CLOPT -f filename パラメータのほか、アプリケーションの UBBCONFIG コンフィギュレーション・ファイルの SERVERS セクションで、その他のパラメータ (BEA Tuxedo システム固有ではないパラメータ) を設定することもできます。

`SRVGRP`、`SRVID`、`RESTART`、`MAXGEN`、`GRACE` などのパラメータの詳細については、第 3 章「コンフィギュレーション・ファイルの作成」の 3-101 ページの「コンフィギュレーション・ファイルの SERVICES セクションの作成方法」を参照してください。

7 ネットワークへの ATMI アプリケーションの分散

ここでは、次の内容について説明します。

- 分散型の ATMI アプリケーションとは
- ATMI アプリケーションをネットワークに分散する理由

注記　ネットワークに BEA Tuxedo CORBA アプリケーションを分散する方法については、『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。

分散型の ATMI アプリケーションとは

分散アプリケーションとは、1 つ以上のローカルまたはリモートのクライアントが、ネットワーク接続された複数のマシン上の 1 つ以上のサーバと通信するアプリケーションです。分散アプリケーションでは、どの場所からでも業務処理を行うことができます。たとえば、企業では、次のような処理を国内各地や海外に分散できます。

- 売上の予測

- 商品の発注
- 商品の製造、出荷、および請求処理
- 社内データベースの更新

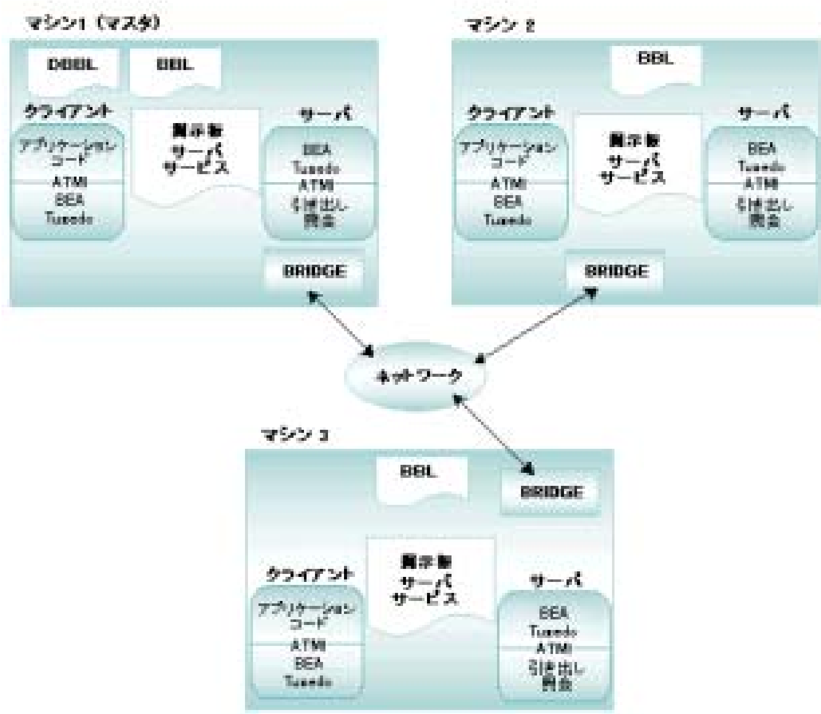
最新の電気通信技術やデータのネットワーク化により、この種の分散処理は一般的になりつつあります。このような企業戦略を実現するために開発されたアプリケーションを使用すると、コストを削減し、世界中の顧客にサービスを提供することができます。

BEA Tuxedo システムは、分散アプリケーションの管理タスクを簡略化することにより、このようなアーキテクチャをサポートします。アプリケーションが 1 台のコンピュータで構成されている場合でも、ネットワーク接続された数千台のコンピュータで構成されている場合でも、アプリケーションのすべての構成要素 (クライアントやサーバ、およびそれらを接続するネットワークなど) は、1 つの BEA Tuxedo コンフィギュレーション・ファイルによって管理されます。

分散アプリケーションの例

次の図は、3 台のマシンに分散されたアプリケーションの基本部分を示しています。

図 7-1 分散アプリケーションの例



分散アプリケーションの実装

分散アプリケーションは、コンフィギュレーション・ファイルの NETWORK セクション (およびオプションで NETGROUPS) に定義されているネットワーク上で実装されます。多くの場合、分散アプリケーションでは、コンフィギュレーション・ファイルの ROUTING セクションに定義されたデータ依存型ルーティングが使用されます。分散アプリケーションの設計では、サーバ・グループ、プロセス、トランザクション・マネージャ・サーバ (TMS)、およびリソース・マネージャ (RM) の配置が重要です。

ネットワーク上に分散アプリケーションを設定する場合、アプリケーション管理者はネットワーク管理者と協力する必要があります。大抵の場合、アプリケーション管理者がコンフィギュレーション・ファイルの RESOURCES、MACHINES、GROUPS、SERVICES、および ROUTING セクションに分散アプリケーション用のパラメータを定義し、ネットワーク管理者または MIS 部門の担当者がネットワーク関連のセクションを定義または担当します。

関連項目

- 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」
- 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」
- 『BEA Tuxedo アプリケーション実行時の管理』の 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』

ATMI アプリケーションをネットワークに分散する理由

分散アプリケーションには、いくつかの重要な利点があります。初期のビジネス・アプリケーションは、1 台の大型メインフレーム・コンピュータ上で実行するように開発されたものでした。コンピュータ処理はすべて 1 台のマシン上で行われたため、障害が発生するとシステム全体の停止につながりました。しかし、分散アプリケーションの普及により、このようなシステム障害による危険は少なくなりつつあります。

また、アプリケーションを分散することにより、1 つのアプリケーションを論理グループに分割し、これらのグループをそれぞれ適切な場所に配置できるという利点もあります。たとえば、サーバ・グループを作成して、大規模なアプリケーションを管理しやすいサイズの業務別コンポーネントに分割し、最適な場所に配置できます。

分散アプリケーションにより、以下のことが可能になります。

- データ依存型の分離を実行できます。
- 複数のリソースを管理できます。
- クライアント / サーバ型のモデルを拡張できます。
- BEA Tuxedo システムのサービスに対し、透過的にアクセスできます。
- 複数のサーバ・グループを確立できます。
- 1 つのアプリケーション作業を複数のコンピュータで同時に行うことにより、スループットと応答時間を向上できます。
- 利用頻度の高いリソースを複製できます。

分散アプリケーションの特長

- 自律的なアクションの調整—自律的なアクションとは、複数のサーバ・グループと複数のリソース・マネージャ・インターフェイスのどちらか、または両方が関係するアクションです。BEA Tuxedo システムでは、個別のアプリケーション間の自律的なアクションを、1 つの論理作業単位として調整することができます。
- フォールト・トレランス—複数のマシンのうち、1 台で障害が発生しても、残りのマシンが処理を続行します。同様に、サーバ・グループ内のどれか 1 つのサーバで障害が発生しても、残りのサーバが作業を続行します。
- スケーラビリティ—アプリケーションの負荷や容量を、以下の方法で増加できます。
  - グループにサーバを追加します。
  - アプリケーションにマシンを追加し、複数のマシンにグループを再分散します。
  - マシン上のサーバ・グループを別のマシン上に複製し、ロード・バランシングを使用します。
  - 特定の基準を満たすグループにデータ依存型ルーティングを使用して、データベースをセグメントに分割します。

関連項目

- 3-4 ページの「複数のマシンで構成 (分散) する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」
- 『BEA Tuxedo システム入門』の「ロード・バランシング」
- 『BEA Tuxedo システム入門』の「データ依存型ルーティング」
- 『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』

8 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

このマニュアルでは、以下の内容について説明します。

- 分散型の BEA Tuxedo ATMI アプリケーション用のコンフィギュレーション・ファイルの要件
- RESOURCES セクションの作成
- MACHINES セクションの作成
- GROUPS セクションの作成
- SERVICES セクションの作成
- ROUTING セクションの作成
- 分散アプリケーション用のコンフィギュレーション・ファイルの例
- ルーティングをサポートするためのドメイン・ゲートウェイ・コンフィギュレーション・ファイルの変更

注記　分散型の BEA Tuxedo CORBA アプリケーション用のコンフィギュレーション・ファイルを作成する方法については、『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。

分散型の BEA Tuxedo ATMI アプリケーション用のコンフィギュレーション・ファイルの要件

分散型の BEA Tuxedo ATMI アプリケーションは、1 つ以上のローカル・クライアントまたはリモート・クライアントで構成され、ネットワーク接続された複数のマシン上にある 1 つ以上のサーバと通信します。すべての構成要素は、BEA Tuxedo コンフィギュレーション・ファイルで 1 つのエンティティとして管理されます。分散型のコンフィギュレーションを設定するには、コンフィギュレーション・ファイルを作成し、次のセクションを設定する必要があります。

- RESOURCES セクション
- MACHINES セクション
- GROUPS セクション
- NETGROUPS セクション (オプション)
- NETWORK セクション
- SERVICES セクション
- ROUTING セクション (データ依存型ルーティングを使用する場合)

コンフィギュレーションに複数のドメインが含まれており、データ依存型ルーティングを使用する場合は、ルーティング機能をサポートするためのドメイン・ゲートウェイ・コンフィギュレーション・ファイル (DMCONFIG) を変更する必要もあります。

RESOURCES セクションの作成

RESOURCES セクションでは、アプリケーションの最大サーバ数など、システム全体に関するリソースの制御パラメータを定義します。このセクションのパラメータ設定は、すべて、アプリケーション全体に適用されます。

注記　このトピックの表で説明するパラメータは、分散アプリケーション用です。どの BEA Tuxedo アプリケーションにも適用できる基本的なパラメータの説明については、『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5) を参照してください。

表 8-1RESOURCES **セクションのパラメータ**

パラメータ	説明
BBLQUERY (オプション)	BBLQUERY は、DBBL がすべての BBL の状態をチェックする間隔を SCANUNIT の乗数で指定します。DBBL は、すべての BBL の状態が BBLQUERY で指定した期間内に報告されるようにします。BBL からの報告がない場合、DBBL はその BBL にメッセージを送信し、状態を照会します。応答がない場合、BBL は分断されます。 BBLQUERY には、0 より大きい値を指定する必要があります。このパラメータを指定しない場合、デフォルト値が設定され、SCANUNIT × BBLQUERY が、約 300 秒になります。
BLOCKTIME (オプション)	BLOCKTIME は、ブロッキング呼び出し (たとえば応答の受信) の後でタイムアウトが発生するまでの時間を SCANUNIT の乗数で指定します。 BLOCKTIME には、0 より大きい値を指定する必要があります。このパラメータを指定しない場合、デフォルト値が設定され、SCANUNIT × BBLQUERY が、約 60 秒になります。

表 8-1RESOURCES **セクションのパラメータ （続き）**

パラメータ	説明
DBBLWAIT (オプション)	DBBLWAIT は、DBBL が、すべての BBL からの応答を待機する期間を SCANUNIT の倍数で指定します。この期間を経過すると、タイムアウトが発生します。DBBL は、BBL に要求を転送した後、すべての BBL から肯定応答を受信した後で要求元に応答を返します。このパラメータを使用して、動作不能な BBL または異常な BBL を検出することもできます。 DBBLWAIT には、0 より大きい値を指定する必要があります。このパラメータを指定しない場合、デフォルト値が設定され、SCANUNIT × DBBLWAIT が SCANUNIT より大きい値か、または 20 秒になります。
IPCKEY (必須)	IPCKEY は、掲示板の数値キーを指定します。単一プロセッサ環境では、このキーにより掲示板の名前が指定されます。マルチプロセッサ環境では、このキーにより DBBL のメッセージ・キューが指定されます。また、このキーは、既知のアドレスのほか、マルチプロセッサ全体の掲示板などのリソース名の基準としても使用されます。 IPCKEY の値は、32,768 より大きく 262,143 未満でなければなりません。
MASTER (必須)	MASTER (*string_value1*[,*string_value2*]) は、TUXCONFIG のマスタ・コピーが置かれているマシンの LMID を指定します。また、アプリケーションが MP モードで実行されている場合、MASTER は DBBL が実行されるマシンを指定します。*string_value2* は、プロセスの再配置および起動時に、使用される LMID の代替位置を指定します。本来の位置が使用できない場合、DBBL はこの代替位置で起動し、代替の TUXCONFIG ファイルが使用されます。 *string_value1* および *string_value2* の値は、どちらも MACHINES セクションで定義されたマシンの LMID でなければなりません。各文字列には、最大 30 文字まで指定できます。

表 8-1RESOURCES **セクションのパラメータ （続き）**

パラメータ	説明
MAXGROUPS (オプション)	MAXGROUPS は、掲示板のグループ・テーブルに対応した、設定済みのサーバ・グループの最大数を指定します。 MAXGROUPS には、100 以上 32,768 未満の値を指定する必要があります。デフォルト値は 100 です。
MAXSERVERS (オプション)	MAXSERVERS は、掲示板のサーバ・テーブルに対応するサーバの最大数を指定します。 MAXSERVERS には、0 以上 8192 未満の値を指定する必要があります。デフォルト値は 50 です。
MAXSERVICES (オプション)	MAXSERVICES は、掲示板のサービス・テーブルに対応するサービスの最大数を指定します。 MAXSERVICES には、0 以上 32.768 未満の値を指定する必要があります。デフォルト値は 100 です。
SANITYSCAN (オプション)	SANITYSCAN は、システムの正常性チェックを行う間隔を、SCANUNIT の乗数で指定します。 SCANUNIT には、0 より大きい値を指定する必要があります。デフォルト値が設定されると、SCANUNIT × SANITYSCAN が約 120 秒になります。 正常性チェックは、掲示板のデータ構造体のほか、サーバに対しても実行されます。
SCANUNIT (オプション)	SCANUNIT は、BBL が、サービス要求内でタイムアウトされたトランザクションやブロッキング呼び出しをスキャンする間隔 (秒単位) を指定します。この値は BBL によるスキャン処理の基本単位として使用されます。この値は、tpbegin(3c) で指定できるトランザクション・タイムアウト値と、BLOCKTIME パラメータで指定されるブロッキング・タイムアウト値に影響します。SANITYSCAN パラメータ、BBLQUERY パラメータ、DBBLWAIT パラメータ、および BLOCKTIME パラメータは、システム内のその他のタイムアウト値を指定するパラメータであり、SCANUNIT の倍数で指定されます。 SCANUNIT には、0 より大きく 60 秒以下の値を 5 の倍数で指定する必要があります。デフォルト値は 10 秒です。

MACHINES セクションの作成

MACHINES セクションでは、コンフィギュレーション内のすべての物理マシン (マルチプロセッサ・マシンのすべての処理機能も含む) に論理名を割り当て、各マシンに対して個別にパラメータを定義します。次の表では、分散アプリケーションに参加する各マシンに対し、マシン名やマシン固有の情報を定義するためのパラメータを説明します。

表 8-2MACHINES セクションのパラメータ

パラメータ	説明
ENVFILE (オプション)	ENVFILE は、マシン上のすべてのクライアントとサーバの実行環境を定義するファイルを指定します。 各行は、*ident=value* の形式で指定します。*ident* には下線 (_) か英数字、またはその両方を指定し、先頭には下線 (_) または英文字を指定します。 ENVFILE に無効なファイル名が指定されると、環境に値は追加されません。
MAXACCESSERS (オプション)	MAXACCESSERS は、このプロセッサの掲示板に同時にアクセスできるプロセスの最大数を指定します。適切な値を計算する場合、BBL や tmadmin などのシステム管理プロセスをカウントする必要はありませんが、アプリケーションのサーバとクライアント、および TMS サーバはすべてカウントする必要があります。 MAXACCESSERS には、0 より大きく 32,768 未満の値を指定する必要があります。デフォルトは、RESOURCES セクションで指定した値です。
MAXCONV (オプション)	MAXCONV は、特定のマシン上のプロセスで同時に実行できる会話の最大数を指定します。 MAXCONV には、0 より大きく 32,768 未満の値を指定する必要があります。サーバあたりの同時会話の最大数は、64 です。デフォルトは、RESOURCES セクションで指定した値です。

表 8-2MACHINES **セクションのパラメータ**

パラメータ	**説明**
MAXWSCLIENTS (オプション)	MAXWSCLIENTS は、このプロセッサが受け付ける、ワークステーション・クライアント専用のアクセサ・エントリ数を指定します。このパラメータは、BEA Tuxedo システムの Workstation コンポーネントが使用される場合にのみ指定されます。このパラメータで指定された数は、MAXACCESSERS で指定したアクセサ・スロットの総数に含まれます。ワークステーション・クライアントからシステムへのアクセスは、BEA Tuxedo システムに組み込まれている代理プロセス、つまりワークステーション・ハンドラによって多重化されます。そのため、このパラメータを適切に設定すると、IPC 資源を節約できます。 MAXWSCLIENTS には、0 以上 32,768 未満の値を指定する必要があります。MAXACCESSERS より大きい値を指定することはできません。MAXACCESSERS より大きい値を MAXWSCLIENTS に割り当てると、エラーが発生します。デフォルト値は 0 です。

GROUPS セクションの作成

GROUPS セクションでは、アプリケーション内の各サーバ・グループを識別し、特定のサーバ・グループ内のサーバに要求がルーティングされるようにします。

GROUPS セクションでは、アプリケーションに必要なサーバ・グループの数を設定します。サーバ・グループは、すべて同じサイトに設定することも (SHM モード)、分散アプリケーションで複数のサイトに分散することも (MP モード) できます。

GROUPS セクションのパラメータは、分散トランザクション処理に関する次の 2 つの側面を実現します。

- パラメータにより、サーバ・グループが特定の LMID およびリソース・マネージャの特定のインスタンスに関連付けられます。
- サーバ・グループに対して、代替マシンを示す別の LMID を関連付けることにより、サーバ・グループを代替マシンに移行できます (ただし、MIGRATE オプションが指定されている場合)。

次の表は、GROUPS セクションのパラメータの説明です。

表 8-3GROUPS **セクションのパラメータ**

パラメータ	説明
ENVFILE	ENVFILE は、グループ内のすべてのサーバの実行環境を定義するファイルを指定します。 各行は、*ident=value* の形式で指定します。*ident* には下線 (_) か英数字、またはその両方を指定します。 ENVFILE に無効なファイル名が指定されると、環境に値は追加されません。
GRPNO (必須)	GRPNO は、特定のサーバ・グループに対して番号を関連付けます。 0 より大きく、30,000 未満の番号を指定する必要があります。番号は、GROUPS セクションのエントリの中で一意でなければなりません。
LMID (必須)	LMID は、定義されたサーバ・グループが実行されるマシンを識別します。値の後に、カンマで区切って別の LMID を指定し、代替マシンを設定することもできます。MIGRATE オプションが指定されている場合は、サーバ・グループをこの代替マシンに移行できます。GROUPS セクションで RESTART=Y が指定されているサーバは移行できます。 LMID の値は、MACHINES セクションの LMID パラメータに割り当てられている値と同じでなければなりません。

SERVICES セクションの作成

SERVICES セクションのパラメータは、アプリケーション・サービスの処理方法を決定します。このセクションのエントリの各行は、識別子名によってサービスと関連付けられます。

SERVICES セクションでは、各サーバ・グループで提供されるサービスを識別する必要があります。SRVGRP パラメータは、1 つのサービスを複数のサーバにリンクするために用意されています。このパラメータは、サービスのインスタンスを示すパラメータを特定のサーバ・グループに関連付けます。

次の表では、分散アプリケーションの定義用の SERVICES セクションのパラメータを説明します。

表 8-4SERVICES **セクションのパラメータ**

パラメータ	**説明**
LOAD (オプション)	LOAD は、システムの SVCNM による負荷の大きさを指定します。 LOAD には、1 ～ 32,767 の値を指定する必要があります。数値が高くなるほど負荷も大きくなります。デフォルト値は 50 です。
PRIO (オプション)	PRIO は、SVCNM のキューからの取り出し優先順位を指定します。 PRIO には、0 より大きく、100 以下の値を指定する必要があります。優先順位の最大値は 100 です。デフォルト値は 50 です。
ROUTING (オプション)	ROUTING は、データ依存型ルーティングを行うときに、このサービスに使用されるルーティング基準名を指定します。このパラメータを指定しないと、このサービスに対してデータ依存型ルーティングは実行されません。 ROUTING には、15 文字までの文字列を指定できます。サービス名は同じで、異なる SRVGRP パラメータをもつ複数のエントリがある場合、ROUTING パラメータはこれらすべてのエントリに対して同じでなければなりません。

表 8-4SERVICES **セクションのパラメータ**

パラメータ	説明

SRVGRP (オプション)	SRVGRP は、SVCNM によって指定され、このセクションのパラメータ・セットによって制御される、サービスのホスト・サーバ・グループを指定します。
	SRVGRP の設定により、複数のサーバで提供される 1 つのサービスに対して、異なるパラメータを設定することができます。たとえば、GROUP1 と GROUP2 の 2 つのサーバ・グループが、両方とも WITHDRAW サービスを提供するとします。SRVGRP を設定すると、このサービスに対して、次のように異なるロード・ファクタを割り当てることができます。
	`WITHDRAW  ROUTING=123 LOAD=60 SRVGRP=GROUP1`
	`WITHDRAW  ROUTING=123 LOAD=60 SRVGRP=GROUP2`
	SRVGRP には、30 文字までの文字列を指定できます。

SVCTIMEOUT (オプション)	SVCTIMEOUT は、指定したサービスの処理にかかる時間を秒単位で指定します。タイムアウトが発生すると、サービス要求を処理するサーバは、SIGKILL シグナルと共に終了します。 SVCTIMEOUT には、0 以上の値を指定する必要があります。0 は、サービスにタイムアウトを適用しないことを示します。デフォルト値は 0 です。

アプリケーションにトランザクション処理が含まれる場合は、SERVICES セクションでさらに 3 つのパラメータ、AUTOTRAN、ROUTING、および TRANTIME を設定します。これらのパラメータについては、5-1 ページの「トランザクション対応の ATMI アプリケーションのコンフィギュレーション」で説明します。

以下は、SERVICES セクションのサンプルです。

```
*SERVICES

WITHDRAW  ROUTING=ACCOUNT_ID
DEPOSIT   ROUTING=ACCOUNT_ID
OPEN_ACCT ROUTING=BRANCH_ID
```

ROUTING セクションの作成

ROUTING セクションでは、データ依存型ルーティングの実行時に使用される基準を指定します。サービスが複数のエントリに属しており、それぞれに対して異なる SRVGRP パラメータの値が設定されている場合、各エントリの ROUTING パラメータには同じ値を指定する必要があります。ROUTING パラメータに同じ値が指定されないと、サービスに対して一貫したルーティングが行われません。サービスがルーティングされるのは、1 つのフィールド上でのみです。したがって、そのフィールドのサービスに対する値は、すべてのエントリで同じでなければなりません。

コンフィギュレーション・ファイルに ROUTING セクションを追加し、データ範囲とグループのマッピングを示すことができます。このセクションの情報により、システムは指定されたグループ内のサーバに要求を送ることができます。ROUTING セクションの各項目には、SERVICES セクションで使用される識別子が含まれています。

ROUTING セクションの行の形式は次のとおりです。

CRITERION_NAME required_parameters

CRITERION_NAME は、データ依存型ルーティングの SERVICES セクションで指定したルーティング・エントリ名です。*CRITERION_NAME* の値は、15 文字以内の文字列でなければなりません。

次の表は、ROUTING セクションのパラメータの説明です。

表 8-5ROUTING セクションのパラメータ

パラメータ	説明
RANGES	ルーティング・フィールドの範囲および関連するサーバ・グループを指定します。
FIELD	ルーティング・フィールドの名前を指定します。値は、FML バッファ、XML の要素または要素の属性、FML フィールド・テーブルで指定された VIEW フィールド名 (FLDTBLDIR および FIELDTBLS 環境変数を使用)、FML の VIEW テーブル (VIEWDIR および VIEWFILES 環境変数を使用) のいずれかになります。この情報は、メッセージの送信時に、データ依存型ルーティングに関連するフィールド値を取得するために使用されます。
BUFTYPE	このルーティング・エントリで有効なデータ・バッファのタイプとサブタイプのリストを指定します。 このパラメータには、最大 256 文字を指定できます。タイプとサブタイプの組み合わせは最大 32 あります。

関連項目

- 3-4 ページの「複数のマシンで構成 (分散) する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」
- 『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5)
- 『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』

分散アプリケーション用のコンフィギュレーション・ファイルの例

以下は、データ依存型ルーティングを使用する BEA Tuxedo アプリケーションの UBBCONFIG サンプル・ファイルからの抜粋です。GROUPS、SERVICES、ROUTING の 3 つのセクションを示しています。

```
*GROUPS
BANKB1            GRPNO=1
BANKB2            GRPNO=2
BANKB3            GRPNO=3
#
*SERVICES
WITHDRAW          ROUTING=BY_ACCOUNT_ID
DEPOSIT           ROUTING=BY_ACCOUNT_ID
INQUIRY           ROUTING=BY_ACCOUNT_ID
OPEN_ACCT         ROUTING=BY_BRANCH_ID
CLOSE_ACCT        ROUTING=BY_BRANCH_ID
#
*ROUTING
BY_ACCOUNT_ID     FIELD=ACCOUNT_ID BUFTYPE=”FML”
                   RANGES=”MIN - 9999:*,
                      10000-49999:BANKB1,
                      50000-79999:BANKB2,
                      80000-109999:BANKB3,
                      *:*”
BY_BRANCH_ID      FIELD=BRANCH_ID BUFTYPE=”FML”
                   RANGES=”MIN - 0:*,
                      1-4:BANKB1,
                      5-7:BANKB2,
                      8-10:BANKB3,
                        *:*”
```

ルーティングをサポートするためのドメイン・ゲートウェイ・コンフィギュレーション・ファイルの変更

ドメイン・ゲートウェイ・コンフィギュレーションに関するすべての情報は、バイナリ形式の BDMCONFIG ファイルに格納されています。このファイルは、まず DMCONFIG というテキスト形式のコンフィギュレーション・ファイルを作成し、次にバイナリ形式の BDMCONFIG にコンパイルして作成します。コンパイル済みの BDMCONFIG ファイルは、実行中のシステムで dmadmin(1) コマンドを使用して更新できます。BEA Tuxedo のマニュアルでは、これらのコンフィギュレーション・ファイルを DMCONFIG および BDMCONFIG と呼んでいますが、任意の名前を付けることもできます。

Domains 機能を追加する各 BEA Tuxedo アプリケーションには、BDMCONFIG ファイルを 1 つ作成する必要があります。BDMCONFIG ファイルへのアクセスは、Domains 管理サーバである DMADM(5) から行います。ゲートウェイ・グループが起動されると、ゲートウェイ管理サーバである GWADM(5) は、そのグループに必要なコンフィギュレーションのコピーを DMADM サーバに要求します。また、GWADM サーバと DMADM サーバは、コンフィギュレーションにおける実行時の変更が、対応するドメイン・ゲートウェイ・グループに反映されるようにします。

注記　DMCONFIG ファイルの詳細については、『BEA Tuxedo のファイル形式とデータ記述方法』の DMCONFIG(5) を参照してください。

DMCONFIG の ROUTING セクションのパラメータ

DM_ROUTING セクションでは、型付きバッファである FML、XML、VIEW、X_C_TYPE、および X_COMMON を使用したサービス要求のデータ依存型ルーティングに関する情報を提供します。DM_ROUTING セクション内にある各行の形式は次のとおりです。

CRITERION_NAME required_parameters

CRITERION_NAME は、SERVICES セクションで指定されたルーティング・エントリの名前です。*CRITERION_NAME* の値は、15 文字以内の文字列でなければなりません。

次の表は、DM_ROUTING セクションのパラメータの説明です。

パラメータ	説明
FIELD (オプション)	ルーティング・フィールドの名前を指定します。値は、FML バッファ、XML の要素または要素の属性、FML フィールド・テーブルで指定された VIEW フィールド名 (FLDTBLDIR および FIELDTBLS 環境変数を使用)、FML の VIEW テーブル (VIEWDIR および VIEWFILES 環境変数を使用) のいずれかになります。この情報は、メッセージの送信時に、データ依存型ルーティングに関連するフィールド値を取得するために使用されます。 FML32 バッファ内のフィールドがルーティングに使用される場合は、8191 以下の数値をフィールド番号として指定します。

パラメータ	説明
RANGES (オプション)	ルーティング・フィールドの範囲と関連付けられたリモート・ドメイン名 (RACCESSPOINT) を指定します。RANGES には、二重引用符で囲まれた文字列を指定します。文字列は、範囲 /RACCESSPOINT のペアをカンマで区切って順番に並べます。 範囲 (*range*) は、単一の値 (符号付き数値または一重引用符で囲んだ文字列)、または *lower - upper* (*lower* と *upper* は共に符号付き数値または一重引用符で囲んだ文字列) の形式で表します。 *lower* には、*upper* 以下の値を設定します。文字列値に一重引用符を埋め込むには、一重引用符の前にバックスラッシュを 2 つ挿入します。たとえば、「O'Brien」は「O\\'Brien」になります。 関連する FIELD のデータ型の最小値を示すには、MIN を使用します。文字列と carray の最小値にはヌル文字列を指定します。文字フィールドの最小値には、0 を指定します。数値の場合、これはフィールドに格納できる最小値です。 関連する FIELD のデータ型の最大値を示すには、MAX を使用します。文字列と carray の最大値には、8 進数値の 255 文字の無限文字列を指定します。文字フィールドの最大値には、単一の 8 進数値の 255 文字を指定します。数値の場合は、数値としてフィールドに格納できる最大値です。したがって、MIN - -5 は -5 以下のすべての数を表し、6 - MAX は 6 以上のすべての数を表します。 範囲 (range) 内のメタキャラクタ * (ワイルドカード) は、既にエントリとして指定した範囲では使用されなかった任意の値を示します。各エントリでは 1 つのワイルドカードによる範囲指定だけが可能です。* は最後に指定します。続けて範囲を指定すると無視されます。

パラメータ	説明
BUFTYPE (オプション)	BUFTYPE は、このルーティング・エントリを適用できるデータ・バッファのタイプとサブタイプを一覧表示します。有効なタイプは、FML、VIEW、X_C_TYPE、および X_COMMON です。FML ではサブタイプは指定されません。それ以外のタイプにはサブタイプが必要です。ただし * は使用できません。タイプとサブタイプのペアのうち、重複するものは同じルーティング基準名として指定できません。タイプとサブタイプのペアが一意の場合、複数のルーティング・エントリは同じ基準名を持つことができます。 単一のルーティング・エントリに複数のバッファ・タイプが指定される場合、各バッファ・タイプに対するルーティング・フィールドのデータ型は同じでなければなりません。フィールド値が設定されていないか (FML バッファの場合)、または特定の範囲と一致しておらず、ワイルドカードの範囲が指定されていない場合、リモート・サービスの実行を要求したアプリケーション・プロセスに対してエラーが返されます。

ルーティング・フィールドの説明

ルーティング・フィールドの値には、FML または VIEW でサポートされているデータ型を使用できます。数値の範囲または文字列の範囲も使用できます。文字列、CARRAY、および文字フィールド型の文字列の範囲値には、以下の規則が適用されます。

- 一重引用符で囲み、先頭に正の符号や負の符号を付けることはできません。
- short 型または long 型の整数値は数字の文字列であり、オプションで先頭に正の符号または負の符号を付けることができます。
- 浮動小数点数は、C コンパイラまたは atof() で受け付けられる形式で指定します。つまり、正の符号または負の符号、数字の文字列 (オプションで小数点を追加)、e または E (オプション)、符号またはスペース (オプション)、整数という形式で指定します。

- フィールド値が範囲と一致するときに、関連する *RACCESSPOINT* 値には、要求がルーティングされるリモート・ドメインを指定します。*RACCESSPOINT* 値に * を指定すると、ゲートウェイ・グループが認識する任意のリモート・ドメインに要求が送られることを示します。範囲 /RACCESSPOINT のペアでは、範囲 (range) と *RACCESSPOINT* はコロン (:) で区切られます。

ルーティングを使用した 5 サイトのドメイン・コンフィギュレーションの例

以下は、2 つのドメインが 5 サイトに分散されているアプリケーションのコンフィギュレーション・ファイルの例です。5 つのサイトは、Central Bank Office と 4 つの支店を示します。4 つの支店のうち、3 つは BEA Tuxedo ドメインに属しています。残りの支店は、別の TP ドメインに属し、そのドメインとの通信には OSI-TP が使用されています。

この例では、Central Bank から見た BEA Tuxedo システムのドメイン・ゲートウェイ・コンフィギュレーション・ファイルを示しています。`DM_TDOMAIN` セクションは、`b01` をミラーリングしたゲートウェイを示しています。

コード リスト 8-1 5 サイト用のドメイン・コンフィギュレーション・ファイル

```
# Central Bank 用の BEA Tuxedo ドメイン・コンフィギュレーション・ファイル。
#
#
*DM_LOCAL
# local_domain_name Gateway_Group_name domain_type domain_ID log_device
#                [audit log] [blocktime]
#                [log name] [log offset] [log size]
#                [maxaccesspoint] [maxraptran] [maxtran]
#                [maxdatalen] [security]
#                [tuxconfig] [tuxoffset]

#
#
DEFAULT: SECURITY = NONE
c01     GWGRP = bankg1
        TYPE = TDOMAIN
        ACCESSPOINTID = "BA.CENTRAL01"
```

```
        DMTLOGDEV = "/usr/apps/bank/DMTLOG"
        DMTLOGNAME = "DMTLG_C01"
c02     GWGRP = bankg2
        TYPE = OSITP
        ACCESSPOINTID = "BA.CENTRAL01"
        DMTLOGDEV = "/usr/apps/bank/DMTLOG"
        DMTLOGNAME = "DMTLG_C02"
        NWDEVICE = "OSITP"
        URCH = "ABCD"
#
*DM_REMOTE
#remote_domain_name  domain_type domain_ID
#
b01     TYPE = TDOMAIN
        ACCESSPOINTID = "BA.BANK01"
b02     TYPE = TDOMAIN
        ACCESSPOINTID = "BA.BANK02"
b03     TYPE = TDOMAIN
        ACCESSPOINTID = "BA.BANK03"
b04     TYPE = OSITP
        ACCESSPOINTID = "BA.BANK04"
        URCH = "ABCD"
#
*DM_TDOMAIN
#
#     local_or_remote_domain_name network_address [nwdevice]
#
# Local network addresses
c01     NWADDR = "//newyork.acme.com:65432"     NWDEVICE ="/dev/tcp"
c02     NWADDR = "//192.76.7.47:65433"     NWDEVICE ="/dev/tcp"
# Remote network addresses: second b01 specifies a mirrored gateway
b01     NWADDR = "//192.11.109.5:1025" NWDEVICE = "/dev/tcp"
b01     NWADDR = "//194.12.110.5:1025" NWDEVICE = "/dev/tcp"
b02     NWADDR = "//dallas.acme.com:65432"  NWDEVICE = "/dev/tcp"
b03     NWADDR = "//192.11.109.156:4244" NWDEVICE = "/dev/tcp"
#
*DM_OSITP
#
#local_or_remote_domain_name apt aeq
#                              [aet] [acn] [apid] [aeid]
#                              [profile]
#
c02     APT = "BA.CENTRAL01"
        AEQ = "TUXEDO.R.4.2.1"
        AET = "{1.3.15.0.3},{1}"
        ACN = "XATMI"
b04     APT = "BA.BANK04"
        AEQ = "TUXEDO.R.4.2.1"
        AET = "{1.3.15.0.4},{1}"
```

```
        ACN = "XATMI"
*DM_EXPORT
#service_name  [Local_Domain_name] [access_control] [exported_svcname]
#                [inbuftype] [outbuftype]
#
open_act        ACL = branch
close_act       ACL = branch
credit
debit
balance
loan            LACCESSPOINT = c02      ACL = loans
*DM_IMPORT
#service_name   [Remote_domain_name] [local_domain_name]
#               [remote_svcname] [routing] [conv]
#               [trantime] [inbuftype] [outbuftype]


#
tlr_add   LACCESSPOINT = c01  ROUTING = ACCOUNT
tlr_bal   LACCESSPOINT = c01  ROUTING = ACCOUNT
tlr_add   RACCESSPOINT = b04  LACCESSPOINT = c02 RNAME ="TPSU002"
tlr_bal   RACCESSPOINT = b04  LACCESSPOINT = c02 RNAME ="TPSU003"
*DM_ROUTING
# routing_criteria    field typed_buffer ranges
#
ACCOUNT FIELD = branchid  BUFTYPE ="VIEW:account"
        RANGES ="MIN - 1000:b01, 1001-3000:b02, *:b03"
*DM_ACCESS_CONTROL
#acl_name    Remote_domain_list
#
branch  ACLIST = b01, b02, b03
loans   ACLIST = b04
```

関連項目

- 『BEA Tuxedo Domains コンポーネント』の 1-19 ページの「Domains コンフィギュレーション・ファイル」
- 『BEA Tuxedo Domains コンポーネント』の 2-29 ページの「Domains コンフィギュレーションの設定」
- 『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』

9 分散アプリケーションのネットワーク設定

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設定
- ネットワーク上のデータの流れ
- パラレル・ネットワーク上でのデータの流れ
- 単純な分散アプリケーションのネットワーク設定例
- ネットワーク・データのスケジューリングでのフェイルオーバーとフェイルバック
- 複数のネットグループによる設定例

分散アプリケーション用のネットワーク設定

分散アプリケーションとは、BEA Tuxedo システムがインストールされた複数のコンピュータ上で動作するアプリケーションのことです。コンピュータどうしは接続されており、ハードウェアやソフトウェア、アクセス方式、および通信プロトコルにより、ネットワーク経由で通信できます。BEA Tuxedo システムは、メッセージの符号化、ルーティング、および復号化を行い、処理されたメッセージをネットワーク経由でマシンに送信します。これらのタスクは自動的に実行されます。

分散アプリケーションに必要なネットワーク機能を設定するには、コンフィギュレーション・ファイルに次のエントリを指定します。

セクション名	パラメータ名	目的
RESOURCES	MODEL (必須)	MP を指定します。このパラメータを指定すると、ほかのネットワーク・パラメータが有効になります。これは、ネットワーク接続されたマシン用のパラメータです。1 台のマシンによるコンフィギュレーションでは、マルチプロセッサ・マシンの場合でも SHM を使用します。
	OPTIONS (必須)	LAN (ローカル・エリア・ネットワーク) を指定すると、同一マシン上のプロセッサ間ではなく、異なるマシン間で通信が行われます。
	MAXNETGROUPS (オプション)	NETGROUPS に定義できるグループ数の上限値を指定します。デフォルト値は 8 で、上限値は 8192 です。

セクション名	パラメータ名	目的

MACHINES	TYPE=*string*（オプション）	2 台のマシン間でメッセージを交換する場合に、メッセージの符号化が必要かどうかを指定します。TYPE パラメータは、定義されたマシンで使用されるデータ表現を指定します。データ表現が異なるマシン間でメッセージを送受信する場合は、送信メッセージを符号化してから送信し、受信時に復号化します。
		データ表現が同じマシン間でメッセージを送受信する場合、符号化 / 復号化の処理は省略されます。
		
		例 1
		`LMID_1 TYPE = “abc”`
		`LMID_2 TYPE = “abc”`
		この場合、復号化は行われません。
		
		例 2
		`LMID_1 TYPE = “HP”`
		`LMID_2 TYPE = “SUN”`
		この場合、符号化が行われます。
		
		メッセージを交換するすべてのマシンのデータ表現が同じであれば、このパラメータを設定する必要はありません。パラメータを設定する必要があるのは、データ表現が異なるマシンのみです。たとえば、9 台の SPARC マシンと 1 台の HP マシンがある場合は、HP マシンにのみ TYPE=*string* を指定します。SPARC マシンは、デフォルトのヌル文字列により、同じタイプであると見なされます。

セクション名	パラメータ名	目的
	CMPLIMIT=*remote* [, *local*] (オプション)	圧縮のしきい値、つまり、リモートやローカルの宛先に送信するメッセージのうち、圧縮の対象とするメッセージの最小値をバイト数で指定します。*remote* および *local* に指定できる値は、0 から MAXLONG までの数値です。CMPLIMIT に 1 つだけ値を指定すると、指定した値は *remote* 引数と見なされ、ローカルの宛先に送信されるメッセージは圧縮されません。 たとえば、CMPLIMIT=1024 と指定すると、1024 バイトより大きいリモート・ロケーション宛てのメッセージは圧縮されます。 TMCMPLIMIT 変数を使用して圧縮のしきい値を指定することもできます。TMCMPPRFM 変数については、tuxenv(5) の説明を参照してください。この変数は、1 〜 9 の範囲で圧縮のレベルを設定します。
	NETLOAD=*number* (オプション)	リモート・サービスの LOAD の値に、アプリケーション固有の数値を追加します。結果は、要求をローカルで処理するか、またはリモート・マシンに送信するかを決定するために使用されます。NETLOAD の値が大きいほど、リモート・マシンに送信されるトラフィックは少なくなります。
NETGROUPS (オプション)	NETGROUP (必須)	アプリケーションにより、特定のマシン・グループに割り当てられる名前を指定します。名前には、最大 30 文字まで指定できます。ネットワーク上のすべてのマシンを含むグループが 1 つの場合、DEFAULTNET という名前に設定する必要があります。
	NETGRPNO=*number* (必須)	マシン・グループをシステム側で識別するための番号を指定します。1 〜 8192 の任意の数値を指定できます。DEFAULTNET の場合、NETGRPNO を 0 に指定する必要があります。
	NETPRIO=*number* (オプション)	NETGROUP に優先順位を割り当てます。このパラメータは、システム側でどのネットワーク接続を使用するかを決定するのに役立ちます。0 〜 8192 の数値を指定する必要があります。高速回線に高い優先順位を割り当てます。DEFAULTNET には最も低い順位を割り当てます。

セクション名	パラメータ名	目的
NETWORK (オプション)	LMID (必須)	指定したマシンを、MACHINES セクション内のエントリの 1 つにマッピングします。
	NADDR=*string* (必須)	この LMID の BRIDGE プロセスに対する接続指示受け付けアドレスを指定します。このネットワーク・アドレスを指定する有効な形式は 4 つあります。詳細については、UBBCONFIG(5) の NETWORK セクションを参照してください。
	NLSADDR=*string* (必須)	この LMID の tlisten プロセスに対するネットワーク・アドレスを指定します。有効な形式は、NADDR の形式と同じです。
	NETGROUP=*string* (オプション)	NETWORK グループ名を指定します。*string* には、NETGROUPS セクションで指定したグループ名を指定しなければなりません。デフォルト値は DEFAULTNET です。

ネットワーク上のデータの流れ

分散アプリケーションでは、データはネットワーク上で次のように送信されます。

- 送信側では、BRIDGE により、メッセージが *destination_machine* に送信されます。BRIDGE は、メッセージをバーチャル回線に書き込み、オペレーティング・システムにその送信処理を委託することにより、送信を実現します。保留メッセージのコピーは、オペレーティング・システム側で保持されます。ただし、ネットワーク・エラーが発生すると、保留メッセージは失われます。
- 受信側では、BRIDGE プロセスが、特定のネットワーク・アドレスでメッセージの受信をリッスンします。

パラレル・ネットワーク上でのデータの流れ

分散アプリケーションでは、ネットワーク上でのデータの送受信にパラレル・データ回線を使用すると、次のような利点を活用できます。

- 複数のアドレスでリッスンできるため、BRIDGE を頻繁に使用できます。
- パラレル・データ回線上で同時にデータを送信することにより、従来のネットワークでの制限が解消され、BRIDGE でより高いスループットを実現できます。
- パラレル・データ回線を設定すると、最初の送信先である回線がビジー状態でも、メッセージの送信処理は失敗しません。トラフィックは、ネットワーク・グループ番号 (NETGRPNO) が最も大きい回線に流れるように、システム側でスケジューリングされます。この回線がビジー状態の場合、トラフィックは次の回線、つまりネットワーク・グループ番号が 2 番目に大きい回線に自動的にスケジューリングされます。すべての回線がビジー状態の場合は、回線が使用可能になるまでデータはキューに入れられます。

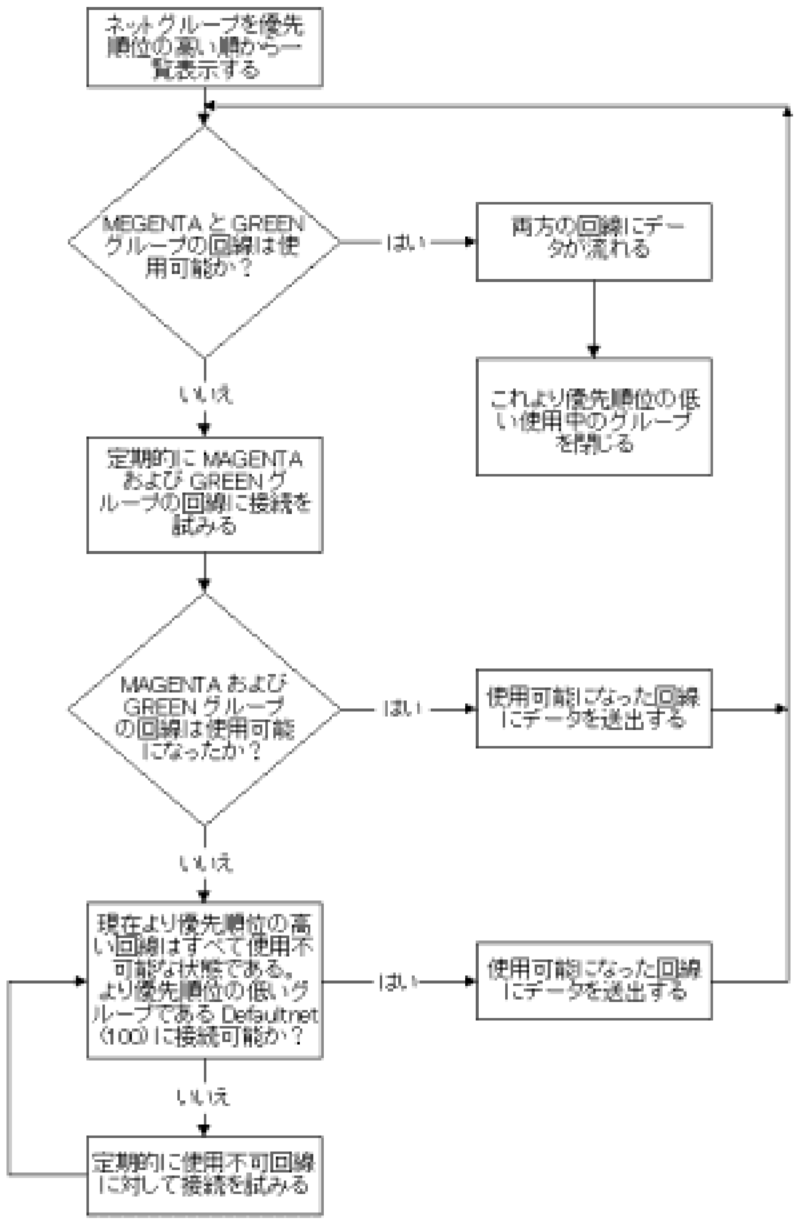
ただし、パラレル・データ回線を使用するかどうかを決定する前に、メッセージを順番に処理する必要があるかどうかを判断してください。システムでは、1 つの特定のデータ回線に会話接続をバインドすることにより、会話型メッセージの正しい順序が確保されます。

すべてのメッセージが順番に送信される必要がある場合、非会話型メッセージの順序もトラッキングされるようにアプリケーションをプログラミングします。この方法を使用する場合、パラレル・データ回線は設定しない方がよいでしょう。

次の図は、あるマシンから別のマシンへのデータの流れを示します。この図は、マシン A とマシン B で構成される例に基づいています。まず、BRIDGE は、両方のマシンに共通のネットワーク・グループである、MAGENTA_GROUP、GREEN_GROUP、および DEFAULTNET を識別します。

優先順位が同じネットワーク・グループ (NETPRIO パラメータの値が同じグループ) には、データは並列に流れます。優先順位が異なるネットワーク・グループは、フェイルオーバー用です。

図 9-1BRIDGE 上のデータの流れ



単純な分散アプリケーションのネットワーク設定例

次の例は、単純なネットワークの設定方法を示しています。

```
# 以下は 2 サイトを含むコンフィギュレーション・ファイルの
# Network セクションの例です。

*NETWORK
   SITE1   NADDR="//mach1:80952”
           NLSADDR="//mach1:serve"
#
   SITE2   NADDR="//mach386:80952"
           NLSADDR="//mach386:serve"
```

ネットワーク・データのスケジューリングでのフェイルオーバーとフェイルバック

データは使用可能な回線のうち、優先順位が最も高い回線を流れます。すべてのネットワーク・グループに同じ優先順位が与えられている場合、データはすべてのネットワークに同時に送信されます。現在優先されているすべての回線で障害が発生すると、データは次に優先順位が高い回線に送信されます。このプロセスを「フェイルオーバー」と呼びます。フェイルオーバーが発生すると、失敗した接続は定期的に再試行されます。

優先順位の高いネットワーク接続が再確立されると、「フェイルバック」が発生し、優先順位の低い接続にはデータがスケジューリングされなくなります。優先順位がより低い接続は、通常の方法で切断されます。

すべてのネットワーク・アドレスへの接続の再試行が失敗すると、次にマシン間でのアプリケーション・データやシステム・データの送信が必要になったときに接続が再試行されます。

複数のネットグループによる設定例

First State Bank という架空の銀行で、5 台のマシン (A-E) が構成されるネットワークを想定してください。これらのマシンは 4 つのネットワーク・グループに構成され、それぞれのマシンが 2 つから 3 つのネットワーク・グループで使用されています。

注記　複数のネットワーク・グループ (NETGROUPS) を含むコンフィギュレーションでは、ハードウェアおよびソフトウェアに対する要件があります。ただし、このマニュアルでは説明していません。たとえば、マシンが、複数の物理ネットワークに参加しなければならない場合があります。各マシンの TCP/IP シンボリック・アドレスは、/etc/hosts ファイルまたは DNS (ドメイン・ネーム・サービス) で識別する必要があります。

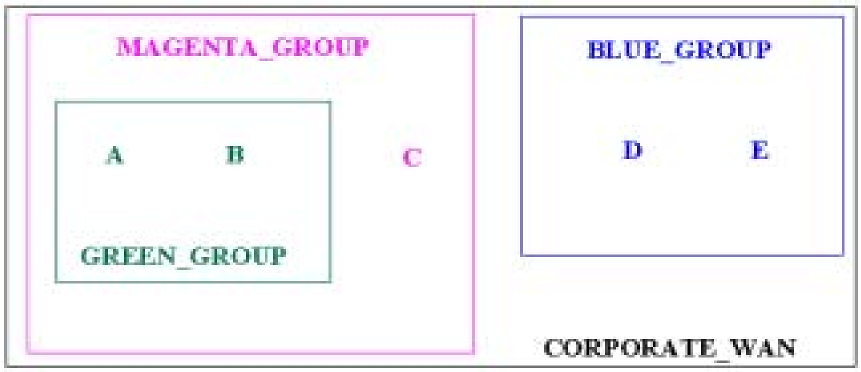
次の例の //A_CORPORATE:5345 という形式のアドレスの文字列 A_CORPORATE は、/etc/hosts ファイルまたは DNS (ドメイン・ネーム・サービス) で指定されていることを想定しています。

First State Bank には、次の 4 つのネットワーク・グループがあります。

- DEFAULTNET (デフォルトのネットワークである企業 WAN)
- MAGENTA_GROUP (LAN)
- BLUE_GROUP (LAN)
- GREEN_GROUP (メンバのマシン間に、高速な光ファイバのポイント・ツー・ポイント・リンクを提供する専用 LAN)

すべてのマシンは、DEFAULTNET (企業 WAN) に属します。さらに、各マシンは MAGENTA_GROUP または BLUE_GROUP に関連付けられます。MAGENTA_GROUP の一部のマシンは GREEN_GROUP に属しています。次の図は、ネットワークにおけるグループの構成を示しています。

図 9-2 ネットワーク・グループの例



この例では、マシン A と B は以下のグループに対するアドレスを持ちます。

- DEFAULTNET (企業 WAN)
- MAGENTA_GROUP (LAN)
- GREEN_GROUP (LAN)

マシン C は以下のグループに対するアドレスを持ちます。

- DEFAULTNET (企業 WAN)
- MAGENTA_GROUP (LAN)

マシン D と E は以下のグループに対するアドレスを持ちます。

- DEFAULTNET (企業 WAN)
- BLUE_GROUP (LAN)

ローカル・エリア・ネットワークはロケーション間でルーティングされないため、マシン D (BLUE_GROUP LAN 内) は、共通の単一のアドレス、つまり企業 WAN ネットワーク・アドレスのみを使用してマシン A (GREEN_GROUP LAN 内) とやり取りします。

サンプル・ネットワークのコンフィギュレーション・ファイル

前の節で説明したネットワークを設定するため、First State Bank の管理者は、以下に示すように、UBBCONFIG ファイルの NETGROUPS および NETWORK セクションで各グループを定義します。

```
*NETGROUPS

DEFAULTNET    NETGRPNO = 0        NETPRIO = 100 #default
BLUE_GROUP    NETGRPNO = 9        NETPRIO = 200
MAGENTA_GROUP NETGRPNO = 125      NETPRIO = 200
GREEN_GROUP   NETGRPNO = 13       NETPRIO = 300

*NETWORK

A      NETGROUP=DEFAULTNET          NADDR="//A_CORPORATE:5723"
A      NETGROUP=MAGENTA_GROUP       NADDR="//A_MAGENTA:5724"
A      NETGROUP=GREEN_GROUP         NADDR="//A_GREEN:5725"

B      NETGROUP=DEFAULTNET          NADDR="//B_CORPORATE:5723"
B      NETGROUP=MAGENTA_GROUP       NADDR="//B_MAGENTA:5724"
B      NETGROUP=GREEN_GROUP         NADDR="//B_GREEN:5725"

C      NETGROUP=DEFAULTNET          NADDR="//C_CORPORATE:5723"
C      NETGROUP=MAGENTA_GROUP       NADDR="//C_MAGENTA:5724"

D      NETGROUP=DEFAULTNET          NADDR="//D_CORPORATE:5723"
D      NETGROUP=BLUE_GROUP          NADDR="//D_BLUE:5726"

E      NETGROUP=DEFAULTNET          NADDR="//E_CORPORATE:5723"
E      NETGROUP=BLUE_GROUP          NADDR="//E_BLUE:5726"
```

各ネットワーク・グループへの優先順位の割り当て

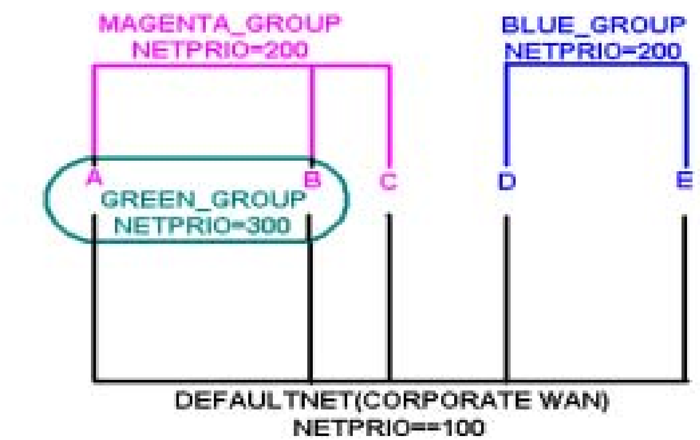
ネットワークの BRIDGE プロセスの機能を最大限に生かすには、各ネットワーク・グループを表す NETGROUP に適切に優先順位を割り当てます。NETGROUP の優先順位を割り当てる場合は、以下の点を念頭に置いてください。

- データは、最も優先順位が高い使用可能な回線上にのみ流れます。
- ネットワーク・グループすべてに同じ優先順位が設定されている場合、データはすべてのネットワーク上を同時に流れます。
- 現在優先されているすべての回線で障害が発生すると、データは次に優先順位が高い回線に送信されます。
- 優先順位がより高い回線が使用可能になると、データは再びこの回線に流れます。
- 優先順位がより高く、使用不可能なすべての回線は定期的に再試行されます。
- すべてのネットワーク・アドレスへの接続の試みが失敗すると、次にマシン間のデータ送信が必要になったときに接続が再試行されます。
- NETPRIO のデフォルト値は 100 です。

ネットワーク・グループへの優先順位の割り当て例

次の図は、First State Bank の管理者が使用可能なネットワーク・グループに優先順位を割り当てる方法を示しています。

図 9-3 ネットワーク・グループへの優先順位の割り当て



次のように優先順位が割り当てられます。

- BLUE_GROUP=200
- DEFAULTNET=100
- GREEN_GROUP=300
- MAGENTA_GROUP=200

NETGROUPS セクションおよび NETWORK セクションの例

最も優先順位の低いネットワーク・グループは、デフォルトのネットワーク・グループとして予約されます。つまり、ほかのネットワーク・グループがすべて使用不可能にならない限り、このグループは使用されません。したがって、分単位で課金される衛星回線など、特定のネットワークの使用を制限する場合は、そのネットワークをデフォルトのネットワーク・グループとして指定します。

デフォルトのネットワーク・グループに優先順位を割り当てるには、ほかのグループと同じように、DEFAULTNET の NETPRIO パラメータを設定します。DEFAULTNET の NETPRIO を指定しないと、以下の例に示すようにデフォルト値の 100 が使用されます。

```
*NETGROUP
DEFAULTNET  NETGRPNO = 0   NETPRIO = 100
```

DEFAULTNET では、ネットワーク・グループ番号 (NETGRPNO) の値を 0 に指定しなければなりません。これ以外の値は無効です。NETGRPNO の値は、エントリごとに一意でなければなりません。

一方、複数のネットワーク・グループに対して、同じ NETPRIO の値を割り当てることもできます。たとえば、First State Bank のコンフィギュレーション・ファイルで、MAGENTA_GROUP と GREEN_GROUP に同じネットワーク優先順位 (NETPRIO=200) を割り当てることができます。

各ネットワーク・アドレス (NETWORK) は、デフォルトで DEFAULTNET ネットワーク・グループに関連付けられます。このパラメータは、エントリ間の統一性を保ちたい場合、または定義されるネットワーク・アドレスを 2 つ目のネットワーク・グループに関連付けたい場合に、明示的に指定することができます。

```
*NETWORK
D      NETGROUP=BLUE_GROUP NADDR="//D_BLUE:5726"
```

10 ワークステーション・クライアントについて

ここでは、次の内容について説明します。

- Workstation コンポーネントとは
- 4 つのワークステーション・クライアントが接続されたアプリケーションの例
- ワークステーション・クライアントのアプリケーションへの接続方法

Workstation コンポーネントとは

BEA Tuxedo システムの Workstation コンポーネントを使用すると、サーバ側の機能がフル・インストールされていないマシン、つまり管理サーバもアプリケーション・サーバもサポートしないマシンに、アプリケーション・クライアントを収容できます。クライアントとアプリケーション・サーバ間の通信は、すべてネットワーク経由で行われます。

ワークステーション・クライアントのプロセスは、Windows 98、Windows 2000、または UNIX プラットフォーム上で実行できます。また、このクライアントは、ATMI にアクセスすることもできます。ユーザは、発行した要求

が背後のネットワークでどのように処理されているかを意識する必要はありません。ワークステーション・クライアントは、ワークステーション・ハンドラ (WSH) を介してシステムに登録し、ネイティブ・クライアントが使用できる機能と同じ機能にアクセスします。

ワークステーション・クライアントとアプリケーション・サーバ間の通信はすべて、ワークステーション・ハンドラ (WSH) プロセスを介して行われます。

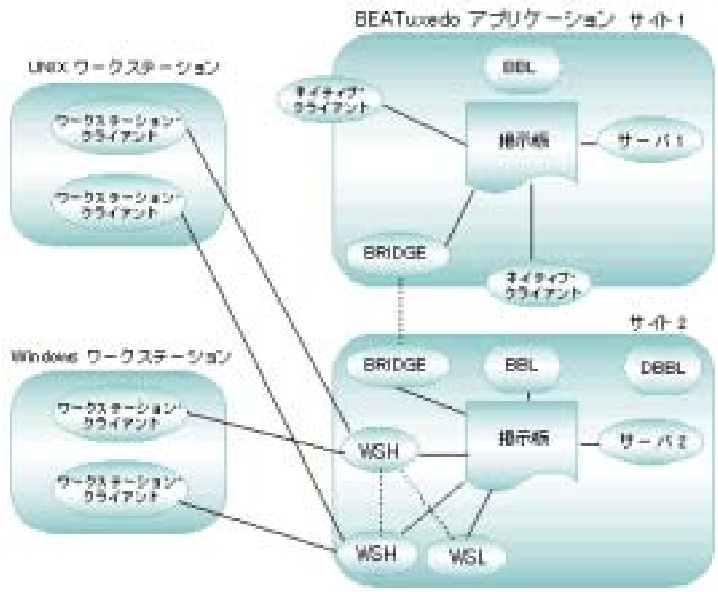
ワークステーション・クライアントは、ネットワーク・クライアントで実行できる機能とほぼ同じ機能を実行できます。たとえば、次のような機能があります。

- メッセージの送受信
- トランザクションの開始、終了、またはコミット
- 任意通知型メッセージの送受信
- BEA Tuxedo クライアントに組み込まれているすべてのセキュリティ・メカニズム

4 つのワークステーション・クライアントが接続されたアプリケーションの例

次の図は、4 つのワークステーション・クライアントが接続されたアプリケーションの例です。

図 10-14 つのワークステーション・クライアントが接続された銀行業務アプリケーション



4 つのうち、2 つのワークステーション・クライアントは、UNIX システム上で実行されています。残りの 2 つのワークステーション・クライアントは、Windows 2000 上で実行されています。すべてのワークステーション・クライアントは、まず、ワークステーション・リスナ (WSL) からアプリケーションに参加します。ワークステーション・リスナは、それ以降の通信処理をワークステーション・ハンドラ (WSH) に任せます。このプロセスは、ネイティブ・クライアントがアプリケーションに参加するプロセスとは異なります。後者の場合、ネイティブ・クライアントは直接掲示板にアタッチして参加します。

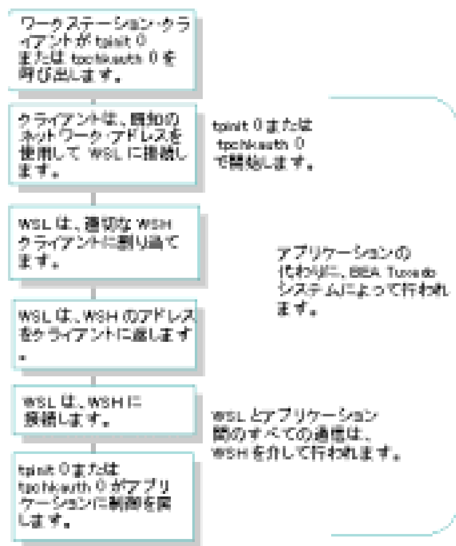
管理サーバとアプリケーション・サーバはすべて `SITE1` と `SITE2` に配置されています。ワークステーション・クライアントからアプリケーションへのサービス要求は、ネットワーク経由で WSH に送られます。WSH は、要求を該当するサーバに転送し、サーバから応答を受け取り、その応答をワークステーション・クライアントに返します。

注記　リソース・マネージャとは、BEA Tuxedo アプリケーションでトランザクション処理を行い、操作を実行するための XA 標準インターフェイスのインプリメンテーションのことです。リソース・マネージャの代表的な例はデータベースです。リソース・マネージャは、グローバル・トランザクション内でアクセスされ、制御されます。

この例では、アプリケーションは 2 台のマシンに分散されているため、MP モードで動作します。ワークステーション・クライアントは、要求を 1 つのワークステーション・ハンドラに送信します。ワークステーション・ハンドラはその要求を BRIDGE プロセスに転送し、次に BRIDGE プロセスにより、要求が正しいマシンに転送されます。

ワークステーション・クライアントのアプリケーションへの接続方法

次の流れ図は、ワークステーション・クライアントからアプリケーションに接続する方法を示しています。



クライアントは、既定のネットワーク・アドレスを使用して WSL プロセスに接続します。クライアントによって `tpchkauth()` または `tpinit()` が呼び出されると、接続を確立するプロセスが開始します。WSL は WSH のアドレスをクライアントに返し、ワークステーション・ハンドラのプロセスに接続要求のプロセスを通知します。WSC は WSH に接続します。以降の WSC とアプリケーション間の通信はすべて WSH を通じて行われます。

11 ワークステーション・クライアントの設定

ここでは、次の内容について説明します。

- ワークステーション・クライアントの定義
- ワークステーション・クライアントの最大数の設定
- ワークステーション・リスナ (WSL) をサーバとして定義する
- ネットワーク障害の検出
- ワークステーション・クライアントをサポートするコンフィギュレーション・ファイルの例

ワークステーション・クライアントの定義

ワークステーション・クライアントを BEA Tuxedo アプリケーションに参加させるには、対応するアプリケーション環境が必要です。BEA Tuxedo システムには、次の表に示すような、環境設定用の変数が用意されています。

TUXDIR および WSNADDR は必須の変数であり、それ以外は、オプションの変数です。WSENVFILE 以外のパラメータには、デフォルト値が用意されています。

説明	設定する環境変数
アプリケーション・パスワード。パスワードによるセキュリティ機能を実装するアプリケーションでのみ有効です。スクリプトから実行するクライアントは、この変数を使用してアプリケーション・パスワードを取得できます。	APP_PW (オプション)
リンク・レベルの暗号化に使用する暗号化キーの最上位ビットの最大数。設定できる値は、0 (暗号を使用しない場合) か、40、56、または 128 (指定された番号が暗号化キーの最上位ビット数の場合) です。	TMMAXENCRYPTBITS (オプション)
リンク・レベルの暗号化に使用する暗号化キーの最上位ビットの最小数。設定できる値は、0 (暗号を使用しない場合) か、40、56、または 128 (指定された番号が暗号化キーの最上位ビット数の場合) です。	TMMINENCRYPTBITS (オプション)
WSRPLYMAX 値に達した場合に応答を格納するディレクトリ。デフォルト値は作業ディレクトリです。	TMPDIR (オプション)
ワークステーション・マシンが、割り当てられた MBSTRING 型付きバッファに格納するコードセットの符号化名を指定します。 TPMBENC にはデフォルト値はありません。MBSTRING 型付きバッファを使用するワークステーション・クライアントの場合、ワークステーション・マシン上で TPMBENC を定義する必要があります。	TPMBENC (オプション)
ワークステーション・マシンが、受信した MBSTRING バッファのデータを、TPMBENC で定義された符号化に自動的に変換するかどうかを指定します。デフォルトでは自動変換は無効になっています。受信した MBSTRING バッファのデータは、符号化の変換はされないままでワークステーション・クライアントに配信されます。TPMBACONV を Y (yes) などの値に設定すると、自動変換が有効になります。	TPMBACONV (オプション)

説明	設定する環境変数
このワークステーション上での BEA Tuxedo システム・ソフトウェアの場所。この環境変数を設定しないと、クライアントは接続を確立できません。	TUXDIR (必須)
ワークステーション・マシンが文書型定義 (DTD)、XML スキーマ、およびエンティティ・ファイルをキャッシュするかどうかを指定します。デフォルトでは、キャッシングは有効になっています (Y)。URLENTITYCACHING を N (no) に設定すると、キャッシングは無効になります。	URLENTITYCACHING (オプション)
ワークステーション・マシンが DTD、スキーマ、およびエンティティ・ファイルをキャッシュするディレクトリを指定します。URLENTITYCACHEDIR 変数には、キャッシュされるファイルの絶対パス名を指定します。URLENTITYCACHEDIR を指定しない場合、デフォルトのディレクトリは URLEntityCachedir になります。このディレクトリはワークステーション・クライアント・プロセスの現在の作業ディレクトリ内に作成され、適切な書き込み権が設定されています。	URLENTITYCACHEDIR (オプション)
使用するネットワーク・デバイス。デフォルト値は空文字列です。	WSDEVICE (オプション)
すべての環境変数を設定するファイルの名前。この変数のデフォルト値はありません。	WSENVFILE (オプション)
ワークステーション・クライアントがワークステーション・リスナまたはワークステーション・ハンドラに接続するのに使用するネットワーク・アドレス。この変数および WSFRANGE 環境変数は、TCP/IP ポート (ワークステーション・クライアントがアウトバウンド接続を行う前にバインドするときのバインド先) の範囲を指定します。このアドレスは、TCP/IP アドレスでなければなりません。	WSFADDR (オプション)
ワークステーション・クライアントのプロセスが、アウトバウンド接続を確立する前にバインドする TCP/IP ポートの範囲。範囲のベースとなるアドレスは、WSFADDR パラメータで指定します。	WSFRANGE (オプション)

説明	設定する環境変数
クライアントがコンタクトする WSL の 1 つまたは複数のネットワーク・アドレスのリスト。このアドレスは、アプリケーション・コンフィギュレーション・ファイルにある WSL プロセスのアドレスと一致する必要があります。	WSNADDR (必須)
アプリケーションの応答を一時的に保存するためのコア・メモリのサイズ。デフォルト値は 256,000 バイトです。	WSRPLYMAX (オプション)
マシンのタイプ。WSL が動作するマシンのコンフィギュレーション・ファイルで指定されている TYPE の値と同じ値を WSTYPE に指定すると、符号化 / 復号化は行われません。デフォルト値は空文字列です。	WSTYPE (オプション)

ワークステーション・クライアントの最大数の設定

アプリケーションにワークステーション・クライアントを参加させるには、UBBCONFIG ファイルの MACHINES セクションで MAXWSCLIENTS パラメータを指定する必要があります。

MAXWSCLIENTS は、Workstation 機能専用の唯一のパラメータです。MAXWSCLIENTS に指定された値は、BEA Tuxedo システムの起動時に通知され、ワークステーション・クライアント用に確保するアクセサ・スロットの数が決まります。ネイティブ・クライアントの場合、各アクセサ・スロットに必要なセマフォは 1 つです。一方、ワークステーション・ハンドラ・プロセス (ワークステーション・クライアントの代わりにネイティブ・プラットフォーム上で実行するプロセス) は、ワークステーション・クライアントのアクセスを単一のアクセサ・スロットに多重化するので、必要なセマフォは 1 つだけです。この点も、Workstation コンポーネントの利点の 1 つです。多

くのクライアントをネイティブ・プラットフォームからワークステーションに移動することによって、アプリケーションで使用する IPC 資源を減らすことができます。

MAXWSCLIENTS は、MAXACCESSERS に指定された数のうち、指定された数のアクセサ・スロットを使用します。MAXWSCLIENTS の設定時には、ネイティブ・クライアントとサーバを収容するために必要な数のスロットを残しておく必要があります。MAXWSCLIENTS 値が MAXACCESSERS 値を超えると、tpinit() の実行時にネイティブ・クライアントとサーバが失敗します。次の表は、MAXWSCLIENTS パラメータの説明です。

パラメータ	**説明**
MAXWSCLIENTS	マシンに接続する WSC の最大数を指定します。 構文は MAXWSCLIENTS=*number* です。デフォルト値は 0 です。 MAXWSCLIENTS を指定しないと、WSC は指定されたマシンに接続できません。

ワークステーション・リスナ (WSL) をサーバとして定義する

ワークステーション・クライアントがアプリケーションにアクセスするときは、1 つの WSL プロセスと 1 つ以上の WSH プロセスのサービスが介在します。WSL は、複数のワークステーション・クライアントをサポートできます。つまり、すべてのワークステーション・クライアントは、WSL のコマンド行で指定される同一のネットワーク・アドレスでアプリケーションと接続します。リスナは、1 つまたは複数のワークステーション・ハンドラ・プロセスをスケジューリングします。

WSH プロセスは、アプリケーションの管理ドメインで、リモート・ワークステーション上のクライアントの代理として機能します。WSH は多重化スキーマによって、複数のワークステーション・クライアントを同時にサポートします。

アプリケーションにワークステーション・クライアントを参加させるには、UBBCONFIG ファイルの SERVERS セクションにワークステーション・リスナ (WSL) のプロセスを指定する必要があります。サーバを指定する場合と同じ構文を使用してください。

WSL プロセスに情報を渡す

WSL プロセスに情報を渡すには、コマンド行オプション文字列 (CLOPT) を使用します。CLOPT パラメータの形式は、次のとおりです。

```
CLOPT="[ -A ] [servopts_options] -- -n netaddr [-d device]
       [-w WSHname][-t timeout_factor][-T Client_timeout]
       [-m minh][-M maxh][-x mpx_factor ]
       [-p minwshport][-P maxwshport]
       [-I init_timeout][-c compression_threshold]
       [-k compression_threshold]
       [-z bits][-Z bits][-H external_netaddr]
       [-N network_timeout][-K{client|handler|both|none}]"
```

-A オプションを指定すると、WSL は起動時にすべてのサービスを提供します。デフォルトでは、このオプションが用意されていますが、ここでは、システム提供のサーバとアプリケーション・サーバの違いを強調するために示してあります。アプリケーション・サーバの起動時には、利用可能なサービスのサブセットだけが提供される場合もあります。

二重ダッシュ (--) は、起動後に WSL に渡されるパラメータ・リストの開始位置を示します。

CLOPT でのコマンド行オプションの使用

以下のコマンド行オプションは、CLOPT 文字列の二重ダッシュ (--) の後に指定することができます。

注記　CLOPT コマンド行オプションの総合一覧については、『BEA Tuxedo のファイル形式とデータ記述方法』の servopts(5) を参照してください。

コマンド行オプション	説明
-n *netaddr* (必須)	WSC がリスナに接続するためのネットワーク・アドレス。WSC は、-n の後に指定されている値に対し、適切な環境変数 (WSNADDR) を設定する必要があります。
[-d *device*] (一部のトランスポート・インターフェイスでは必須)	ネットワーク・デバイス名を指定します。 これは、一部のトランスポート・インターフェイスでのみ必要なオプション・パラメータです。たとえば、ソケットではこのパラメータは必要ありません。
[-t *timeout*]	クライアントが WSH に接続するまでの時間。 システムは timeout の値と SCANUNIT パラメータの値を乗算して、接続までの時間を算出します。 デフォルト値は、セキュリティなしのアプリケーションでは 3、セキュリティ付きアプリケーションでは 6 です。このコンテキストでは、次のいずれかのパラメータが設定されているアプリケーションは、セキュリティ付きであると見なされます。 ■ USER_AUTH ■ ACL ■ MANDATORY_ACL ■ APP_PW
[-w name]	このリスナに対して起動する WSH プロセスの名前。デフォルト値は WSH、つまり提供されているハンドラの名前です。buildwsh(1) コマンドで別のハンドラ・プロセスを作成した場合は、その名前をここで指定します。
[-m *number*]	起動し、常に利用可能にしておくハンドラの最小数。デフォルト値は 0 です。

コマンド行オプション	説明
[-M *number*]	起動できるハンドラの最大数。デフォルト値は、マシンの MAXWSCLIENTS を多重係数 (-x で指定) で割った値です。
[-x *number*]	WSH が同時に多重化できるクライアントの最大数。この値は 0 より大きくなければなりません。デフォルト値は 10 です。
[-T client_timeout]	クライアントが、接続を切断されないままアイドル状態でいられる時間 (分)。クライアントがこの時間内に要求を行わなかった場合、WSH はクライアント接続を切断します。この引数を指定しない場合、または 0 が指定されている場合、タイムアウトは発生しません。
[-p *minwshport*] and [-P maxwshport]	このリスナ・サーバに関連付けられている WSH が使用できるポート番号の範囲。0 〜 65535 の範囲のポート番号を指定してください。デフォルト値は、*minwshport* が 2048、*maxwshport* が 65535 です。
[-z] および [-Z]	WSL 側で、リンク・レベルの暗号化のために使用できるビットの範囲ビットの最小値を指定するには -z を使用し、最大値を指定するには -Z を指定します。
[-N *network_timeout*]	ワークステーション・クライアントが WSL/WSH からの応答を受け取るまでの最小待機時間 (秒)。デフォルト値は 0、つまりタイムアウトが発生しないことを示します。
[-K {client | handler | both | none}]	ワークステーション・ハンドラとワークステーション・クライアントの間で、指定された時間内にトラフィックが発生しなかった場合、これらの間で接続を確立するかどうかを指定します。

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の servopts(5)

ネットワーク障害の検出

Workstation コンポーネントには、ネットワーク接続が切断されてもハング状態が無期限に続かないようにするための、WSL 用の管理オプションが用意されています。これにより、次のことを実現できます。

- クライアント接続を定期的にチェックできます (keep-alive オプション)。
- WSH からの応答をクライアントが待機する時間を制限できます。この時間を経過すると、WSH との接続は切断されます (ネットワーク・タイムアウト・オプション)。

keep-alive オプションの使用

keep-alive は、ワークステーション・ハンドラとワークステーション・クライアントの間で一定期間トラフィックが発生しなかった場合に、その接続の実行可能性を定期的にチェックするネットワーク機能です。

keep-alive オプションを要求するには、UBBCONFIG ファイルの SERVERS セクションで、WSL CLOPT エントリに -K オプションを追加します。-K オプションは、client、handler、both、または none という引数をとります。

オプション	目的
-K client	クライアント・マシンから keep-alive メッセージを生成します。keep-alive メッセージが認識されない場合、クライアント・マシンはネットワークがダウンしていると見なします。以降の ATMI 呼び出しは、TPESYSTEM の tperrno で異常終了します。
-K handler	ハンドラ・マシンから keep-alive メッセージを生成します。keep-alive メッセージが認識されない場合、ハンドラ・マシンはネットワークがダウンしているものとみなします。ハンドラは、応答しないクライアントに関連するエントリをクリーンアップします。この結果、ワークステーションが同時に多重化できるクライアント (-x で指定) を、ハンドラが消費しないようにすることができます。
-K both	クライアント・マシンとハンドラ・マシンの両方から、keep-alive メッセージを生成します。このコンポーネントの利用度とタイムアウトのしきい値は、オペレーティング・システムの調整可能なパラメータによって決まります。
-K none	keep-alive オプションをオフにします。この設定は、-K をまったく指定しない場合と同じ結果になります。

UBBCONFIG ファイル内のエントリは、次のようになります。

```
WSL SRVGRP="WSLGRP" SRVID=1000 RESTART=Y GRACE=0
CLOPT="-A -- -n //ws.beasys.com:5120 -d /dev/tcp -K both"
```

この例では、-K により、ワークステーション・クライアントとワークステーション・サーバの両方の keep-alive チェックが有効になります。

UBBCONFIG の WSL エントリの詳細については、『BEA Tuxedo のファイル形式とデータ記述方法』の WSL(5) を参照してください。

注記　指定したタイムアウト値は、システム全体に適用されます。1 つのアプリケーションに対してタイムアウト値を指定し、その値を後で変更すると、keep-alive を使用するすべてのアプリケーションに影響します。

keep-alive オプションの制約

keep-alive オプションは、ソケットを使用する BEA Tuxedo システムを搭載したプラットフォームでのみサポートされます。

- Tru64 UNIX
- HP UX
- Windows

このオプションは、上記以外のプラットフォームでは使用できません。BEA Tuxedo システムでは、どのサーバ・マシンにも -K オプションを指定できますが、上記以外のプラットフォームでは正しく実行されません。上記以外のプラットフォームで keep-alive 機能を実行すると失敗し、ユーザ・ログにメッセージが書き込まれます (WSH のプロセスごとに 1 回)。処理は、通常どおり続行します。

注記　keep-alive 機能は、TCP/IP 通信でのみ有効です。

ネットワーク・タイムアウト・オプションの使用

ネットワーク・タイムアウトとは、ワークステーション・クライアントによる操作を待機する時間を指定するオプションです。ここで指定した時間を経過すると、操作に対する要求は、ネットワーク上でキャンセル (タイムアウト) されます。

ネットワーク・タイムアウトは、`WSL` の管理オプションである `-N` を使用して要求できます。`-N` オプションは、ネットワーク・タイムアウトを使用して、ワークステーション・クライアントのデータを受信します。

ネットワーク・タイムアウトのしくみ

ネットワーク・タイムアウト・オプションにより、ワークステーション・クライアントがネットワークからデータを受信するための BEA Tuxedo 操作を待機する期間 (秒単位) が指定されます。この期間を過ぎると、操作は失敗し、クライアントはアプリケーションから切断されます。値 0 (デフォルト) はタイムアウトが発生しないことを示します。

注記　この値を低く設定しすぎると、接続が頻繁に切断されます。

タイムアウトが発生すると、各 ATMI 関数はエラーを返します。リンクでタイムアウトが発生すると、アプリケーションに通知されます。この場合、既存のエラー・コードが使用されます。特定のエラー・コードの詳細については、`tperrordetail(3c)` を呼び出して参照してください。ネットワーク・タイムアウトが発生すると、未処理の操作は信頼できない状態になります。つまり、トランザクションが途中で終了したり、応答が失われたりします。アプリケーションへの接続を安全に終了するには、`WSH` と通信せずに、`tpterm(3c)` の呼び出しと同じ操作を行います。

操作が返される時点では、クライアントは BEA Tuxedo アプリケーションの一部ではなくなっています。クライアントを再度アプリケーションに参加させるには、次のどちらかの方法を使用します。

- `tpinit(3c)` を呼び出す
- 暗黙的な接続を使用する (セキュリティが設定されていない場合)

ネットワーク・タイムアウト・オプションの制限

- ネットワークの送信操作に対してネットワーク・タイムアウトは設定できません。
- ネットワーク・タイムアウトの値がトランザクション・タイムアウトまたはブロッキング・タイムアウトの値より小さいと、クライアントは、要求の処理が完了する前に切断される場合があります。
- ネットワーク・タイムアウトが発生すると、接続がまだ有効でもワークステーション・クライアントの接続は切断されます。

ネットワーク・タイムアウト・オプションの設定

BEA Tuxedo アプリケーションにネットワーク・タイムアウト・オプションを指定するには、`WSL CLOPT` の引数に `-N` オプションを追加します。

ワークステーション・クライアントをサポートするコンフィギュレーション・ファイルの例

以下のサンプル・コンフィギュレーション・ファイルからの抜粋は、`bankapp` アプリケーションにワークステーション・コンポーネントを追加する方法を示しています。ここでは、`MACHINES` セクションと `SERVERS` セクションが変更されています。

コード リスト 11-1　**ワークステーション・クライアントをサポートする** UBBCONFIG **ファイルの例**

```
*MACHINES
SITE1
              ...
              MAXWSCLIENTS=150

              ...
SITE2

              ...
              MAXWSCLIENTS=0
              ...

*SERVERS
              ...
WSL SRVGRP="BANKB1" SRVID=500 RESTART=Y
            CLOPT="-A -- -n //ws.beasys.com:5120 -m 5 -M 30 -x 5"

              ...
```

MACHINES セクションおよび SERVERS セクションの変更

MACHINES セクションおよび SERVERS セクションが次のように変更されています。

- MACHINES セクションでは、MAXWSCLIENTS のデフォルト値が 2 つのサイトのエントリで上書きされています。SITE1 ではデフォルト値が 150 に引き上げられ、SITE2 では 0 に引き下げられています。値 0 の場合は、ワークステーションが接続されません。
- SERVERS セクションでは、WSL プロセスはグループ BANKB1 に指定されています。WSL のサーバ ID は 500 です。再起動可能と指定されています。
- コマンド行オプションでは、以下の値が指定されています。
  - WSL はすべてのサービスを宣言します (-A)。

- WSL はネットワーク・アドレス `//ws.beasys.com:5120` で接続指示を受け付けます (`-n`)。
- 最低 5 つの WSH が起動します (`-m`)。
- 最大 30 の WSH が起動します (`-M`)。
- 各ハンドラは同時に最大 5 つのクライアントを接続できます (`-x`)。

12 BEA Tuxedo CORBA リモート・クライアント・アプリケーションの管理

この章では、標準のインターネット ORB 間プロトコル (IIOP: Internet Inter-ORB Protocol) を使用して BEA Tuxedo CORBA のリモート・クライアント・アプリケーションから CORBA オブジェクトへの接続を設定する方法について説明します。この章は、BEA Tuxedo CORBA サーバにのみ適用されます。

ここでは、次の内容について説明します。

- CORBA オブジェクト関連の用語
- CORBA リモート・クライアントの概要
- CORBA リモート・クライアントの環境変数を設定する
- CORBA リモート・クライアントの最大数を設定する
- CORBA リモート・クライアントのリスナを設定する
- CORBA リモート・クライアントをサポートするようにコンフィギュレーション・ファイルを変更する

- リモート共同クライアント / サーバのアウトバウンド IIOP を設定する
- ISL コマンドを使用してアウトバウンド IIOP のサポートを設定する

CORBA オブジェクト関連の用語

以下は、この章で使用する用語の一覧です。

DLL

ダイナミック・リンク・ライブラリ (DLL: Dynamic Link Library)。ロード・モジュールにグループ化された関数の集合。Windows アプリケーションでは、実行時に実行形式プログラムに動的にリンクされます。

IIOP

インターネット ORB 間プロトコル (IIOP: Internet Inter-ORB Protocol)。TCP/IP 準拠の通信プロトコルで、共通のバックボーン・プロトコルとして CORBA 定義のメッセージ送受信をサポートします。

ISH

IIOP サーバ・ハンドラ (ISH: IIOP Server Handler)。アプリケーション・サイト上で実行されるクライアント・プロセス。リモート・クライアントの代理として機能します。

ISL

IIOP サーバ・リスナ (ISL: IIOP Server Listener)。アプリケーション・サイト上で実行されるサーバ・プロセス。リモート・クライアントの接続要求をリッスンします。

サーバ

BEA Tuxedo ドメインのマシンで管理されるサーバ。BEA Tuxedo CORBA サーバは、BEA Tuxedo CORBA の `buildobjserver` コマンドを使用して作成されます。CORBA サーバは、セキュリティ、トランザクション、オブジェクトの状態管理など、BEA Tuxedo の機

能をインプリメントします。サーバは、BEA Tuxedo ドメイン内外の任意のサーバを呼び出すことができます。

ネイティブ・クライアント

BEA Tuxedo ドメイン内のクライアント。CORBA ORB を使用して、BEA Tuxedo ドメイン内外のオブジェクトを呼び出します。ネイティブ・クライアントのホストには、BEA Tuxedo の管理コンポーネントとインフラストラクチャ・コンポーネントがあります。たとえば、`tmadmin`、FactoryFinder、ISL/ISH などです。ネイティブ・クライアントは、環境オブジェクトを使用して CORBA オブジェクトにアクセスします。ネイティブ C++ クライアントは `buildobjclient` コマンドを使用して作成し、ネイティブ Java クライアントはサード・パーティ ORB 提供のツールを使用して作成します。

リモート・クライアント

BEA Tuxedo ドメイン外のクライアント。リモート・クライアントは、CORBA ORB を使用して、BEA Tuxedo ドメイン内外のオブジェクトを呼び出すことができます。リモート・クライアントのホストには、`tmadmin`、FactoryFinder、ISL/ISH など、BEA Tuxedo の管理コンポーネントやインフラストラクチャ・コンポーネントはなく、リモート・クライアントがオブジェクトを呼び出すためのサポート・ソフトウェア (CORBA ORB) があります。リモート・クライアントは、環境オブジェクトを使用して CORBA オブジェクトにアクセスします。リモート C++ クライアントは `buildobjclient` コマンドを使用して作成し、リモート Java クライアントはサード・パーティ ORB 提供のツールを使用して作成します。

ネイティブ共同クライアント / サーバ

(1) ビジネス上の処理のスタータとなるコードを実行し、(2) オブジェクトを呼び出すためのメソッド・コードを実行する、という 2 つの目的を持つプロセス。共同クライアント / サーバは、BEA Tuxedo ドメイン内にあります。ネイティブ・ジョイント C++ クライアント / サーバは、`buildobjclient` コマンドを使用して作成します。Java ネイティブ共同クライアント / サーバはサポートされていません。

注記　ネイティブ共同クライアント / サーバのサーバ・ロールは、通常のサーバと比べて大幅に弱くなります。tmadmin、FactoryFinder、ISL/ISH などの BEA Tuxedo CORBA の管理コンポーネントやインフラストラクチャ・コンポーネントはなく (BEA Tuxedo のスケーラビリティと信頼性に関する属性もなし)、BEA Tuxedo の TP Framework も使用しないため、クライアントと ORB 間でより直接的なインタラクションが必要になります。

リモート共同クライアント / サーバ

(1) ビジネス上の処理のスタータとなるコードを実行し、(2) オブジェクトを呼び出すためのメソッド・コードを実行する、という 2 つの目的を持つプロセス。共同クライアント / サーバは、BEA Tuxedo ドメイン外にあります。共同クライアント / サーバは BEA Tuxedo TP Framework を使用しないので、クライアントと ORB 間でより直接的なインタラクションが必要です。リモート・ジョイント C++ クライアント / サーバは `buildobjclient` コマンドを使用して作

成し、リモート Java クライアント / サーバはサード・パーティ ORB 提供のツールを使用して作成します。

注記　共同クライアント / サーバは、サーバ・ロールの一部としてクライアントの役割を果たすサーバとは異なります。サーバが呼び出し処理を完了すると、休止状態に戻ります。共同クライアント / サーバは常にアクティブ・モードで動作し、サーバ・ロールとは関連のないコードを実行します。このためサーバ・ロールによって一時的にアクティブなクライアント・ロールが中断されますが、クライアント・ロールは常に再開されます。

注記　リモート共同クライアント / サーバのサーバ・ロールは、通常のサーバと比べて大幅に弱くなります。クライアントにもサーバにも、tmadmin、FactoryFinder、ISL/ISH など、BEA Tuxedo の管理コンポーネントやインフラストラクチャ・コンポーネントはありません (BEA Tuxedo のスケーラビリティや信頼性に関する属性もなし)。

BEA Tuxedo CORBA オブジェクト

TP Framework を使用してインプリメントされ、セキュリティ、トランザクション、およびオブジェクトの状態管理をインプリメントする CORBA オブジェクト。CORBA オブジェクトは、BEA Tuxedo CORBA サーバでインプリメントされます。つまり、これらのオブジェクトは BEA Tuxedo ドメインの一部であり、BEA Tuxedo インフラストラクチャを使用します。

コールバック・オブジェクト

ターゲット オブジェクトでのクライアントの呼び出しで、パラメータとして提供される CORBA オブジェクト。ターゲット・オブジェクトは、そのターゲット・オブジェクトの実行時、または後で (ターゲット オブジェクトの呼び出しが完了した後でも可) コールバック・オブジェクトを呼び出すことができます。コールバック・オブジェクトは、BEA Tuxedo ドメイン内外のどちらにも配置できます。

CORBA リモート・クライアントの概要

この節での「リモート・クライアント」は、BEA Tuxedo CORBA サーバ・ソフトウェアがフル・インストールされていないシステムに配置された CORBA クライアント・アプリケーションを意味します。つまり、管理サーバやアプリケーション・サーバが実行されず、掲示板もないことを意味します。クライアントとアプリケーション間の通信は、すべてネットワーク経由で行われます。

クライアントのタイプは以下のとおりです。

- CORBA C++ クライアント
- CORBA Java クライアント
- ActiveX クライアント

クライアント・プロセスは、UNIX または Microsoft Windows 上で実行できます。また、クライアントは CORBA ORB インターフェイスにアクセスすることができます。ユーザは、呼び出しが背後のネットワークでどのように処理されているかを意識する必要はありません。クライアント・プロセスはシステムに登録され、ネイティブ・クライアントと同じ状態になります。

クライアントには以下の特徴があります。

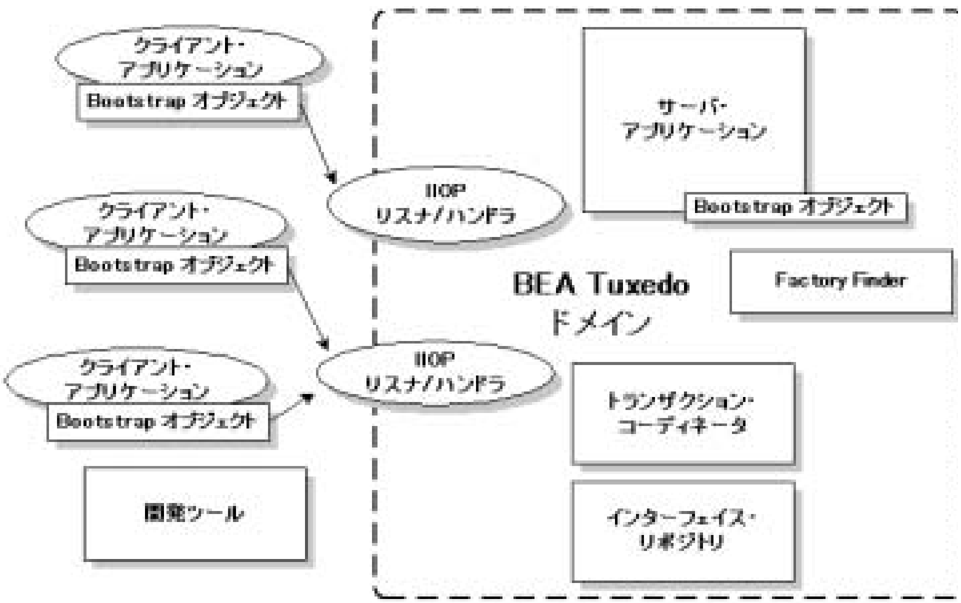
- リモート CORBA オブジェクトでメソッドを呼び出します。
- トランザクションの開始、ロールバック、またはコミットを行います。
- アプリケーション・セキュリティにパスする必要があります。

注記　クライアント・プロセスは、ISH を通してネイティブ・ドメインと通信します。

CORBA リモート・クライアントが接続されたアプリケーションの例

図 12-1 は、リモート・クライアントが接続されたアプリケーションの例です。リモート・クライアントから CORBA サーバ・アプリケーションへのアクセス要求は、ネットワーク経由で ISH に送信されます。このプロセスがアプリケーション・サーバに要求を送信し、その応答をリモート・クライアントに返します。

図 12-1 リモート・クライアントが接続された銀行業務アプリケーション



リモート・クライアントからアプリケーションへの接続方法

クライアントは、既知のネットワーク・アドレスを使用して IIOP リスナ / ハンドラで ISL プロセスに接続します。この接続は、クライアントが Bootstrap オブジェクトのコンストラクタを呼び出すと開始されます。ISL プロセスは、オペレーティング・システム固有の機能を使用して、選択された ISH プロセスに直接接続を渡します。クライアント・アプリケーション側から見ると、接続は 1 つだけです。クライアント・アプリケーションは、現在 ISH プロセスに接続されていることを認識せず、また、認識する必要もありません。

CORBA リモート・クライアントの環境変数を設定する

CORBA C++ クライアントでは、以下に示す環境変数を使用してシステムに情報を渡すことができます。

- `TUXDIR`—リモート・クライアント上の BEA Tuxedo CORBA クライアント・ソフトウェアの場所。クライアントを接続するにはこの値が設定されていなければなりません。
- `TOBJADDR`—クライアントがコンタクトする ISL のネットワーク・アドレス。この値は、アプリケーションのコンフィギュレーション・ファイルで指定された ISL プロセスのアドレスと一致していなければなりません。

  注記　プログラマが Bootstrap 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` ファイルで指定されたネットワーク・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。これらのアドレスが一致しないと、Bootstrap 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、サーバ・アプリケーションの UBBCONFIG ファイルの ISL コマンド行オプションで、ネットワーク・アドレスが //TRIXIE:3500 に指定されている場合、Bootstrap コンストラクタまたは TOBJADDR で //192.12.4.6:3500 や //trixie:3500 を指定すると、接続が失敗します。

UNIX システムでは、ホスト・システムの 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 システムの場合は、ホスト・システムの [コントロール パネル] の [ネットワーク] で大文字 / 小文字を指定してください。または、環境変数 COMPUTERNAME を使用します。次に例を示します。

```
echo %COMPUTERNAME%
```

CORBA リモート・クライアントの最大数を設定する

アプリケーションにリモート・クライアントを参加させるには、UBBCONFIG ファイルの MACHINES セクションで MAXWSCLIENTS パラメータを指定する必要があります。

MAXWSCLIENTS に指定された値は、BEA Tuxedo システムの起動時に通知され、リモート・クライアント用に確保するアクセサ・スロットの数が決まります。ネイティブ・クライアントの場合、各アクセサ・スロットに必要なセマフォは 1 つです。一方、ISH プロセス (リモート・クライアントの代わりにネイティブ・プラットフォーム上で実行するプロセス) は、リモート・クライアントのアクセスを単一のアクセサ・スロットに多重化するので、必要なセマフォは 1 つだけです。この点も、リモート機能の別の利点の 1 つで

す。多くのクライアントをネイティブ・プラットフォームからリモート・システムに移動することによって、アプリケーションで使用する IPC 資源を減らすことができます。

MAXWSCLIENTS は、MAXACCESSERS に指定された数のうち、指定された数のアクセサ・スロットを使用します。MAXWSCLIENTS の設定時には、ネイティブ・クライアントとサーバを収容するために必要な数のスロットを残しておく必要があります。MAXWSCLIENTS に MAXACCESSERS より大きい値を指定しないでください。次の表は、MAXWSCLIENTS パラメータの説明です。

パラメータ	**説明**
MAXWSCLIENTS	マシンに接続するリモート・クライアントの最大数を指定します。 デフォルト値は 0 です。この値を指定しないと、リモート・クライアントから指定されたマシンに接続できません。 構文は MAXWSCLIENTS=*number* です。

CORBA リモート・クライアントのリスナを設定する

リモート・クライアントは、1 つの ISL プロセスと、1 つ以上の ISH プロセスのサービスを介してアプリケーションにアクセスします。ISL は、BEA Tuxedo システム側で提供されるサーバとして、1 つのエントリで指定されます。ISL は複数のリモート・クライアントを扱うことができます。これらのリモート・クライアントは 唯一の通信ポイントである ISL を経由して特定のネットワーク・アドレス (ISL コマンド行で指定) のアプリケーションに接続します。リスナは、1 つまたは複数のリモート・ハンドラ・プロセスをスケジューリングします。ISH プロセスは、アプリケーションの管理ドメインで、リモート・システム上のリモート・クライアントの代理として機能します。ISH は多重化スキーマによって、複数のワークステーション・クライアントを同時にサポートします。

リモート・クライアントをアプリケーションに参加させるには、`UBBCONFIG` ファイルの `SERVERS` セクションで ISL プロセスをリストする必要があります。この場合、サーバのリストと同じ構文を使用します。

CLOPT パラメータの形式

以下の `ISL` コマンド行オプション (`CLOPT`) を使用して、リモート・クライアントの ISL プロセスに情報を渡します。`CLOPT` パラメータの形式は、次のとおりです。

```
ISL SRVGRP=”identifier”
    SRVID="number"
    CLOPT="[ -A ] [ servopts options  ] -- -n netaddr
    [ -C {detect|warn|none} ]
    [ -d device ]
    [ -K {client|handler|both|none} ]
    [ -m minh ]
    [ -M maxh ]
    [ -T client-timeout]
    [ -x mpx-factor ]
    [ -H external-netaddr"
```

CLOPT コマンド行オプションの詳細については、『BEA Tuxedo コマンド・リファレンス』の ISL コマンドを参照してください。

CORBA リモート・クライアントをサポートするようにコンフィギュレーション・ファイルを変更する

コード リスト 0-1 は、リモート・クライアントをサポートする UBBCONFIG ファイルの例です。以下の特徴があります。

- MACHINES セクションでは、2 つのサイトでデフォルトの MAXWSCLIENTS が上書きされています。SITE1 ではデフォルト値が 150 に引き上げられ、SITE2 では 0 に引き下げられています。値 0 の場合は、リモート・クライアントが接続されません。
- SERVERS セクションには、BANKB1 グループの ISL プロセスがリストされています。サーバ ID は 500 で、再起動可能と指定されています。
- コマンド行オプションでは、以下の値が指定されています。
  - IIOP リスナ / ハンドラはすべてのサービスを宣言します (-A)。
  - IIOP リスナ / ハンドラは、ポート 2500 のホスト TRIXIE でリッスンします。

- ネットワーク・プロバイダは /dev/tcp (-d) です。
- 起動に必要な ISH プロセス数の最小値は 5 です (-m)。
- 起動に必要な ISH プロセス数の最大値は 30 です (-M)。
- 各ハンドラは同時に最大 5 つのクライアントを接続できます (-x)。

コード リスト 0-1 UBBCONFIG **ファイルの設定例**

```
*MACHINES
SITE1
              ...
              MAXWSCLIENTS=150
              ...
SITE2
              ...
              MAXWSCLIENTS=0
              ...
*SERVERS
              ...
ISL SRVGRP=”BANKB1" SRVID=500 RESTART=Y
           CLOPT=”-A -- -n //TRIXIE:2500 -d /dev/tcp
                 -m 5 -M 30 -x 5"
          ..
```

リモート共同クライアント / サーバのアウトバウンド IIOP を設定する

アウトバウンド IIOP のサポートにより、ネイティブ・クライアントと、ネイティブ・クライアントとして機能するサーバは、BEA Tuxedo ドメイン外のリモート・オブジェクト・リファレンスを呼び出すことができます。つまり、コールバックに登録されているリモート・クライアントを呼び出して、リモート・サーバ内のオブジェクトにアクセスすることができます。

アウトバウンド IIOP サポート・コンポーネントと直接通信できるのは管理者だけです。管理者は、適切な起動パラメータを使用して ISL を起動し、接続が確立されていないクライアントのオブジェクトに対して、アウトバウンド IIOP を実行できるようにします。また、管理者は、起動する ISL の数や各種起動パラメータを調整し、システムの負荷状況に応じた最適な設定を行う必要もあります。

デフォルトのパラメータを使用して ISL を起動することも可能です。ただし、BEA Tuxedo ISL のデフォルトの起動パラメータでは、IIOP のアウトバウンドを有効にできません。

注記　アウトバウンド IIOP は、トランザクションおよびセキュリティ機能ではサポートされていません。

機能説明

クライアント・コールバックをサポートするには、アウトバウンド IIOP のサポートが必要です。BEA WebLogic Enterprise のバージョン 4.0 および 4.1 では、ISL/ISH はインバウンドのハーフ・ゲートウェイでしたが、アウトバウンド IIOP のサポートにより、ISL/ISH にアウトバウンドのハーフ・ゲートウェイが追加されています (図 12-2 を参照してください)。

ネイティブ・サーバおよびリモート共同クライアント / サーバ・アプリケーションでサポートされる GIOP のバージョンによって、以下に示す 3 種類のアウトバウンド IIOP 接続を使用できます。

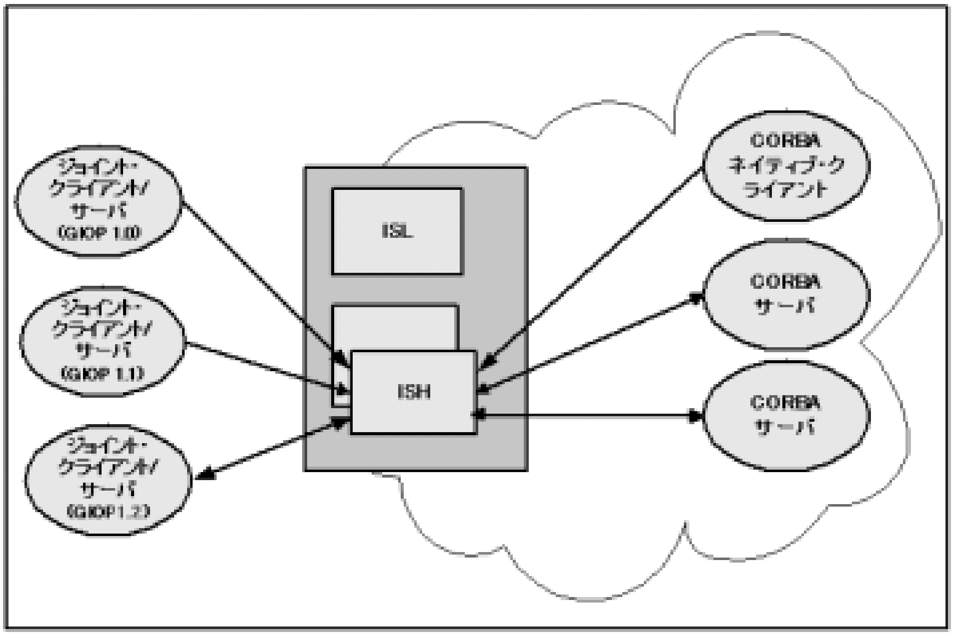
- 双方向一接続を再利用するアウトバウンド IIOP (BEA WebLogic Enterprise のリリース 4.2 以降の C++ GIOP 1.2 サーバ、クライアント、および共同クライアント / サーバでのみサポート)
- 非対称―2 番目の接続経由のアウトバウンド IIOP (GIOP 1.0、GIOP 1.1、GIOP 1.2 のサーバ、クライアント、および共同クライアント / サーバ・アプリケーションでサポート)。
- デュアル・ペア接続―アウトバウンド IIOP (GIOP 1.0、GIOP 1.1、GIOP 1.2 のサーバ、クライアント、および共同クライアント / サーバ・アプリケーションでサポート)。

注記　GIOP 1.2 は、BEA WebLogic Enterprise リリース 4.2 (以降) と BEA Tuxedo リリース 8.0 (以降) の C++ クライアント、サーバ、および共同クライアント / サーバでのみサポートされます。BEA WebLogic Enterprise リリース 4.0 と 4.1 の C++ クライアントとサーバでは、GIOP のバージョン 1.0 と 1.1 はサポートしていますが、GIOP 1.2 はサポートしていません。Java クライアント、サーバ、および共同クライアント / サーバは、GIOP 1.0 のみサポートしています。

双方向およびデュアル・ペア接続のアウトバウンド IIOP では、ISH に接続された共同クライアント / サーバに存在するオブジェクト・リファレンスへのアウトバウンド IIOP 接続が可能です。非対称のアウトバウンド IIOP では、ISH に接続された共同クライアント / サーバに存在しないオブジェクト・リファレンスへのアウトバウンド IIOP 接続が可能です。また、現在 ISH に接続されているクライアント上のオブジェクト・リファレンスだけでなく、任意のオブジェクト・リファレンスを BEA Tuxedo CORBA クライアントによって呼び出すことができます。

以下の節では、それぞれのアウトバウンド IIOP について詳しく説明します。

図 12-2 サポートされる共同クライアント / サーバ IIOP 接続



双方向のアウトバウンド IIOP

双方向のアウトバウンド IIOP では、以下の操作が実行されます (図 12-3 参照)。

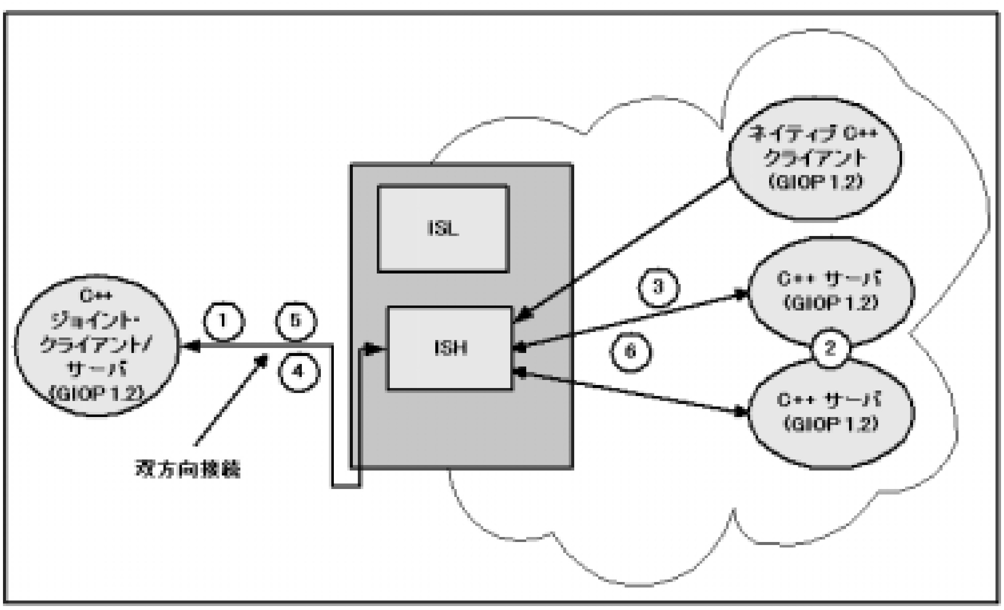
1. クライアントはオブジェクト・リファレンスを作成し、BEA Tuxedo CORBA サーバを呼び出します。クライアント ORB は、サービス・コンテキストを使用して、その接続が双方向であることを認識します。このサービス・コンテキストは、メッセージと共に BEA Tuxedo CORBA サーバに送信されます。

2. オブジェクト・リファレンスのアンマーシャリングで、BEA Tuxedo CORBA サーバはサービス・コンテキストのホスト / ポートをオブジェクト・リファレンスのホスト / ポートと比較します。これらが一致すると、ORB によって ISH へのルーティングに必要な ISH クライアント情報が

追加されます。このクライアント情報は、オブジェクト・リファレンスが BEA Tuxedo CORBA サーバに渡される場合、常にオブジェクト・リファレンスと共に送信されます。

3. 適切な時点で、BEA Tuxedo CORBA サーバまたはネイティブ・クライアントがオブジェクト・リファレンスを呼び出します。クライアント情報が指定されている場合、ルーティング・コードによって適切な ISH が呼び出されます。
4. ISH は、同じクライアント接続を使用して要求をクライアントに送信します。
5. クライアントはメソッドを実行し、クライアント接続を使用して ISH に応答を返します。
6. ISH は、応答を受信して BEA Tuxedo CORBA サーバに転送します。

図 12-3 双方向接続

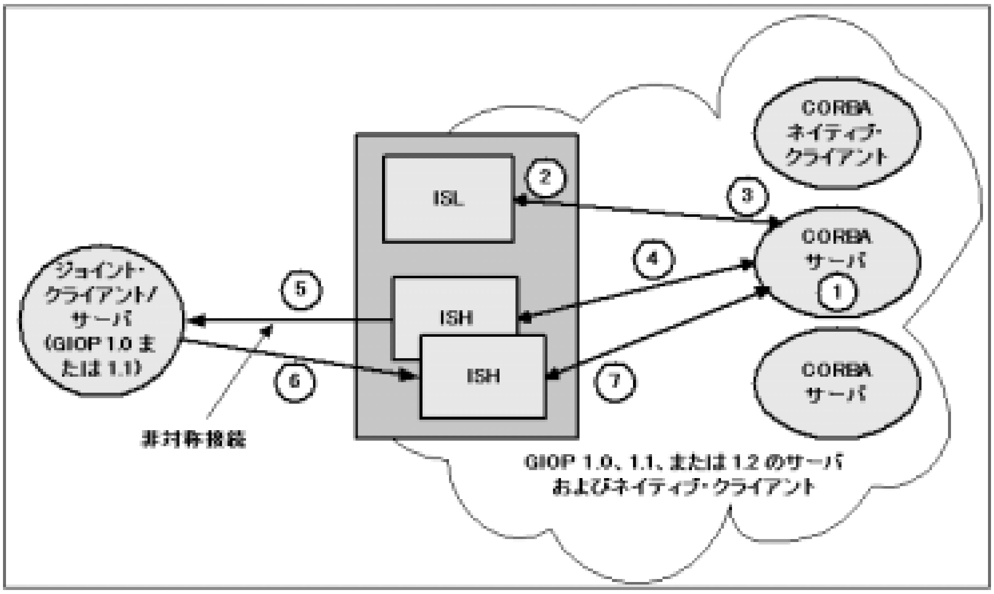


非対称のアウトバウンド IIOP

非対称のアウトバウンド IIOP では、以下の操作が実行されます (図 12-4 参照)。

1. サーバは、ソースからオブジェクト・リファレンスを取得します。ネーミング・サービスや `string_to_object` を取得したり、クライアントを介して取得することもできますが、クライアント自体に存在するものは対象外です。ISH に接続されたクライアントにはオブジェクト・リファレンスが存在しないため、双方向接続を使用して送信時呼び出しを行うことはできません。BEA Tuxedo CORBA サーバがオブジェクト・リファレンスを呼び出します。
2. 最初の呼び出しで、ルーティング・コードによって ISL のサービスが呼び出され、ホスト / ポートに渡されます。
3. ISL はアウトバウンド呼び出しを処理する ISH を選択し、ISH 情報を BEA Tuxedo CORBA サーバに返します。
4. BEA Tuxedo CORBA サーバは ISH を呼び出します。
5. ISH は、クライアントに対して要求を送信するときに使用する送信時接続を決定します。接続が確立されていない場合、ISH はホスト / ポートへの接続を作成します。
6. クライアントがメソッドを実行し、ISH に応答を返します。
7. ISH は、応答を受信して BEA Tuxedo CORBA サーバに転送します。

図 12-4 非対称のアウトバウンド IIOP



デュアル・ペア接続によるアウトバウンド IIOP

デュアル・ペア接続によるアウトバウンド IIOP では、以下の操作が実行されます (図 12-5 参照)。

1. クライアントは、オブジェクト・リファレンスを作成し、Bootstrap 関数 (`register_callback_port`) を呼び出して、オブジェクト・リファレンスを渡します。
2. ISH が IOR からホスト / ポートを取得し、クライアント・コンテキストに保存します。
3. クライアントは BEA Tuxedo CORBA サーバを呼び出してオブジェクト・リファレンスを渡します。`register_callback_port` 呼び出しによって、ISH がホスト / ポートを含むサービス・コンテキストを作成します。このサービス・コンテキストは、メッセージと共に BEA Tuxedo CORBA サーバに送信されます。

4. オブジェクト・リファレンスのアンマーシャリングで、BEA Tuxedo CORBA サーバはサービス・コンテキストのホスト / ポートをオブジェクト・リファレンスのホスト / ポートと比較します。これらが一致すると、ORB によって ISH クライアント情報がオブジェクト・リファレンスに追加されます。このクライアント情報は、オブジェクト・リファレンスが BEA Tuxedo CORBA サーバに渡される場合、常にオブジェクト・リファレンスと共に送信されます。

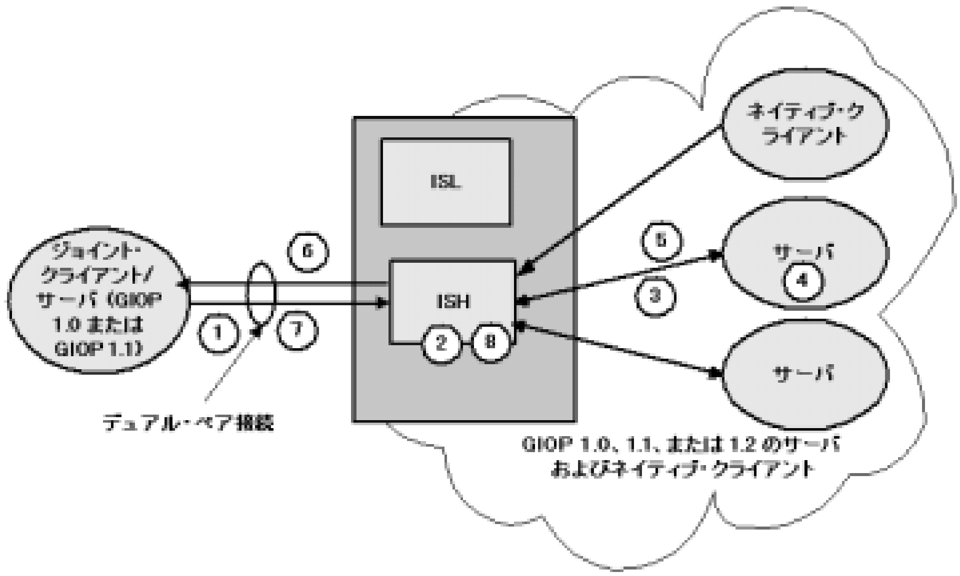
5. 適切な時点で、BEA Tuxedo CORBA サーバまたはネイティブ・クライアントがオブジェクト・リファレンスを呼び出します。ルーティング・コードによって適切な ISH が呼び出され、クライアント情報が渡されます。

6. ISH は、クライアントに対する 2 番目の接続を作成します。ISH は、この 2 番目の接続を使用してクライアントに要求を送信します。

7. クライアントがメソッドを実行し、最初のクライアント接続を使用して ISH に応答を返します。

8. ISH は、応答を受信して BEA Tuxedo CORBA サーバに転送します。クライアントは ISH との最初の接続と 2 番目の接続を切断します。

図 12-5 デュアル・ペア接続によるアウトバウンド IIOP



ルーティング・コードによる ISL の検出

ISL の検出手順は以下のとおりです。

1. 各 ISL でサービスが宣言されます。
2. ルーティング・コードによってサービス名が呼び出されます。

   注記　ISL の検出には通常の BEA Tuxedo ルーティングが使用されます。

3. 同じマシン上でアイドル状態の ISL が使用可能な場合は、その ISL が常に選択されます。アイドルの ISL が使用できない場合、NETLOAD は通常ローカル ISL が選択されていることを確認します。

   注記　ローカル・マシン以外のマシンで ISL が呼び出される場合もあります。

ISL コマンドを使用してアウトバウンド IIOP のサポートを設定する

ネイティブな C++ または Java クライアント、ネイティブ・クライアントとして動作するサーバからリモート・オブジェクト・リファレンスを呼び出す場合は、アウトバウンド IIOP サポートを使用します。ルーティング・コードでは、オブジェクト・リファレンスのソースが BEA Tuxedo CORBA ORB 以外、またはリモートの BEA Tuxedo CORBA 共同クライアント / サーバであることが認識されます。

オブジェクト・リファレンスの種類

リモート・オブジェクト・リファレンスには、次の 2 つの種類があります。

- BEA Tuxedo ドメイン外の BEA Tuxedo CORBA リモート・共同クライアント / サーバによって作成されたオブジェクト・リファレンス
- サードパーティ製のサーバによって作成されたオブジェクト・リファレンス

どちらのオブジェクト・リファレンスも、ルーティング・コードで検出され、アウトバウンド IIOP サポートに送信された後、処理されます。

ユーザ・インターフェイス

アウトバウンド IIOP サポートのユーザ・インターフェイスは、コマンド行インターフェイスであり、これを使用して ISL プロセスを起動します。BEA Tuxedo ソフトウェアの今回のリリースでは、アウトバウンド IIOP 処理を設定するための新しいコマンド行オプションが追加されています。これらのオプションを使用すると、ISH に接続されていないクライアントに存在するオブジェクト・リファレンスへの非対称の IIOP 接続がサポートされます。

以下の ISL コマンド構文は、アウトバウンド IIOP をサポートする新しいオプションを示しています。

```
ISL SRVGRP="identifier"

    SRVID="number"

    CLOPT="[ -A ] [ servopts options ] -- -n netaddr
           [ -C {detect|warn|none} ]
           [ -d device ]
           [ -K {client|handler|both|none} ]
           [ -m minh ]
           [ -M maxh ]
           [ -T Client-timeout]
           [ -x mpx-factor ]
           [-H external-netaddr]
#NEW options for outbound IIOP
           [-O]
           [-o outbound-max-connections]
           [-s Server-timeout]
           [-u out-mpx-users] "
```

CLOPT コマンド行オプションの詳細については、『BEA Tuxedo コマンド・リファレンス』の ISL コマンドを参照してください。