BEA Tuxedo

BEA Tuxedo システム
入門

BEA Tuxedo リリース 8.0 J
8.0 版
2001 年 10 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA Tuxedo システム入門

Document Edition	Date	Software Version
8.0J	2001 年 10 月	BEA Tuxedo リリース 8.0J

目次

このマニュアルについて

このマニュアルでは、BEA Tuxedo ATMI (アプリケーション・トランザクション・モニタ・インターフェイス) プログラミング環境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 BEA Tuxedo システムの基本概念」—BEA Tuxedo プログラミング環境の概要
- 「第 2 章 BEA Tuxedo ATMI のアーキテクチャ」—BEA Tuxedo ATMI 環境の基本的な構成要素。BEA Tuxedo ATMI 環境は、環境への外部インターフェイス、ATMI 層、MIB、システムの各サービス、および標準に準拠したリソース・マネージャとのインターフェイスから構成されます。
- 「第 3 章 BEA Tuxedo ATMI インフラストラクチャに対する 3 つの視点」—BEA Tuxedo ATMI インフラストラクチャの分析。分析は次の 3 つの視点から行います。管理、開発 (ATMI を使用)、および実行時。

対象読者

このマニュアルは、BEA Tuxedo プログラミング環境について学び、BEA Tuxedo 製品を使用して分散 ATMI アプリケーションを作成する必要のあるプログラマを対象としています。

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、Web サイト上にあります。また、マニュアルの CD-ROM にも収められています。この PDF を Adobe Acrobat Reader で開くと、マニュアル全体または一部をブック形式で印刷できます。PDF 形式を利用するには、BEA Tuxedo Documents ページの [PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader をお持ちではない場合は、Adobe Web サイト (http://www.adobe.co.jp/) から無償で入手できます。

関連情報

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

- 『BEA Tuxedo システムのインストール』—CD に同梱されているドキュメント
- 『BEA Tuxedo リリース・ノート』—CD に同梱されているドキュメント
- 『BEA Tuxedo アプリケーションの設定』—BEA Tuxedo オンライン・マニュアル CD で参照できます。このガイドには、BEA Tuxedo システムのセットアップおよび管理方法についての情報が記載されています。
- 『BEA Tuxedo 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 アプリケーションの実行時の管理方法について説明しています。
- 『BEA Tuxedo CORBA アプリケーション入門』—BEA Tuxedo オンライン・マニュアル CD で参照できます。このガイドでは、BEA Tuxedo CORBA 環境で分散 CORBA アプリケーションを開発する方法について説明しています。

BEA Tuxedo ATMI 環境の設定と管理の詳細については、http://edocs.beasys.com/ 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。

規則	項目

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`*
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。

規則	項目
. . .	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 BEA Tuxedo システムの基本概念

ここでは、次の内容について説明します。

- BEA Tuxedo システムとは
- クライアント / サーバ・モデルの分析
- クライアント / サーバ・モデルにおける BEA Tuxedo システム
- BEA Tuxedo 環境におけるクライアント、サーバ、サービス
- BEA Tuxedo システムによって提供されるサービス
- BEA 製品ファミリ

参考文献

BEA Tuxedo システムに関しては、さまざまな書籍やドキュメントが出版されています。次の書籍、ホワイト・ペーパー、およびプレゼンテーションには、クライアント / サーバ・アーキテクチャ、分散ビジネス・アプリケーションの構築と管理、BEA Tuxedo システムを使用したエンタープライズ・アプリケーションの構築と管理などに関する情報が記載されています。

- J. M. Andrade、T. Carges、T. Dwyer、S. Felts 共著『Tuxedo システム－分散ビジネスアプリケーションの構築と管理』Reading, Massachusetts:Addison-Wesley Publshers Japan, Ltd. 渡辺榮一 監訳、漆原茂 / 佐々木政和 訳 1998 年 6 月 30 日
- J. Edwards、DeVoe 共著『3 層 C/S コンピューティング ケーススタディ』株式会社翔泳社、笠野英松 監修、有限会社シンクス 訳、1998 年 3 月 30 日 New York、John Wiley & Sons, Inc.、1997 年 4 月
- J. Edwards、Harkey、R. Orfali 共著 『The Essential Client/Server Survival Guide』New York、John Wiley & Sons, Inc.、1997 年 5 月
- C. Hall 著『Building Client/Server Applications Using Tuxedo - Designing and Building Cost-Effective, High Performance Client/Server Applications Using Tuxedo』Wiley Computer Publishing
- R. Lee 発表『BEA Tuxedo Essentials』1999 年 2 月、ルイジアナ州ニューオーリンズにおける BEA User's Conference でのプレゼンテーション

- R. MacBlane 発表『Managing your BEA Tuxedo Applications Even Over the Internet』1997 年 5 月、カリフォルニア州サンノゼにおける BEA User's Conference でのプレゼンテーション
- R. MacBlane 発表『Tuxedo's Management Information Base』1996 年 2 月、カリフォルニア州サンフランシスコにおける BEA User's Conference でのプレゼンテーション
- 『BEA Tuxedo:The Programming Model』(ホワイト・ペーパー)
- 『BEA Tuxedo and the Component Software Model』(ホワイト・ペーパー)
- 『Inter-Application Transaction Processing with BEA Tuxedo Domains』(ホワイト・ペーパー)
- 『Reliable Queuing Using BEA Tuxedo』(ホワイト・ペーパー)

BEA Tuxedo システムとは

BEA Tuxedo システムは、メッセージ・ベースの通信、および必要に応じて分散トランザクション処理を行って、複数のプラットフォーム、データベース、およびオペレーティング・システムにわたってアプリケーションを分散するミドルウェア製品です。

クライアント / サーバ・アプリケーションでは、ミドルウェアを使用して、複数のサーバ間で処理を分散したり、分散トランザクションを管理したり、複数のデータベース・プラットフォームを統合することができます。ミドルウェア・システムは、「オンライン・トランザクション処理」システム、「OLTP」システムと呼ばれることもあります。

BEA Tuxedo システムは、AT&T、UNIX System Laboratories (USL)、Novell、および BEA 社などのテクノロジ企業のグループが 15 年以上の年月をかけて開発した完成度の高い製品です。このシステムは、開発プラットフォームであると同時に、実行プラットフォームでもあります。BEA Tuxedo システムは、オペレーティング・システムの拡張として機能します。

BEA Tuxedo システムでは、以下が実現されています。

- 異種クライアント / サーバ環境における分散オンライン・トランザクション・アプリケーションの作成と集中管理のための業界標準。
- アプリケーション開発者にとって使いやすいシステム。開発者は、使用するサーバのロケーション、ルーティング、プラットフォームなどの詳細を完全に把握する必要はありません。BEA Tuxedo アプリケーションでは、プログラムのこのような要素が透過的に扱われています。

- 信頼性が高く、パフォーマンスに優れ、管理しやすい分散システムを作成して管理し、維持するための基盤。

BEA Tuxedo システムの機能

BEA Tuxedo システムには、アプリケーションの管理者、構築担当者、およびプログラマのニーズに応える多くの機能が提供されています。

管理に関する機能

- パスワード・セキュリティおよびアクセス制御セキュリティ—パスワード・セキュリティにより、アプリケーション設計者は、初期化の際にパスワードを要求すること (認証) によってアクセスを制御できます。アクセスは、権限によってさらに制御できます。これは、特定のアプリケーション・サービスへのアクセスを制限する手段で、明示的な許可を与えられ、認証を受けたクライアントだけがサービスにアクセスできます。
- システム・イベントの通知—BEA Tuxedo システムでは、サーバの停止やネットワーク障害などのシステム・イベントに関する詳細が提供されます。クライアントまたはサーバによってイベントがポストされると、イベント・ブローカがそのイベントのすべてのサブスクライバを確認し、各サブスクリプションの指定に従って適切な処理を行います。
- MIB (管理情報ベース)—独自のプログラムによってアプリケーションの監視、コンフィギュレーション、チューニングを行うことができる管理用のインターフェイスです。これは、FML 属性として定義され、インプリメンテーションに依存しない管理データベースです。情報を照会したり、変更することができます。
- Web ベースの管理—World Wide Web を通して BEA Tuxedo アプリケーションのコンフィギュレーションと制御を行うグラフィカル・ユーザ・インターフェイスです。

アーキテクチャに関する機能

- 分散サービス—異なるハードウェア・プラットフォーム上に存在するアプリケーション・サービスやシステム・サービスに対して、透過的にアクセスすることができます。
- 高速なコネクションレス型通信—クライアントがサーバではなく掲示板に接続するので、システムのパフォーマンスが向上します。

- スケーラビリティ―サービスおよびサーバを簡単に複製したり分散できるので、要求されるシステム・ロードの変化にすばやく対応してアプリケーションの規模を調整できます。プログラムでしきい値を設定すると、BEA Tuxedo システムで自動的にサーバを新しく作成したり、シャットダウンすることができます。
- サーバの透過性―掲示板上のサービス・ディレクトリによってサービス名がサーバにマップされるので、クライアントがサーバを識別する必要はありません。

プログラミングに関する機能

- 通信方法―Tuxedo システムのアプリケーション・プログラミング・インターフェイス (API) は、X/Open の XATMI インターフェイスのスーパーセットであり、アプリケーション・トランザクション・モニタ・インターフェイス (ATMI) と呼ばれます。Tuxedo の ATMI には、分散アプリケーションを作成するための通信方法が豊富に提供されています。
- 分散トランザクション処理 (DTP)―分散アプリケーション全体で行われている作業をアトミックに完了することができます。これは、どの OLTP システムにおいても重要な特徴の 1 つです。
- 型付きバッファ―異種プラットフォーム間でアプリケーション・データを透過的に処理することができます。
- X/Open TX 準拠―BEA Tuxedo システムは、トランザクションの境界判定に関する X/Open インターフェイス標準に準拠しています。
- X/Open XA 準拠―BEA Tuxedo システムは、トランザクション・データベース・システム (リソース・マネージャ) に関する X/Open インターフェイス標準に準拠しています。そのため、データの整合性を維持しながら、1 つのアプリケーション内で複数のデータベースを混在させ、そのデータベースに合致させることができます。

関連項目

- 第 1 章の 12 ページ「BEA Tuxedo システムによって提供されるサービス」
- 第 1 章の 13 ページ「BEA 製品ファミリ」

クライアント / サーバ・モデルの分析

クライアント / サーバ・アーキテクチャでは、クライアント (サービスを必要とするユーザを表すプログラム) とサーバ (サービスを提供するプログラム) は、それぞれ別個の論理オブジェクトであり、ネットワーク上で通信して共同で処理を行います。クライアントは、サービスを要求し、その要求に対する応答を受け取ります。サーバは、要求を受け取って処理し、要求された応答を送り返します。

クライアント / サーバ・アーキテクチャの特徴

- 非対称のプロトコル—クライアントとサーバには、多対 1 の関係があります。クライアントは、常にサービスを要求することによって会話を開始します。サーバは、受動的にクライアントからの要求を待ちます。
- サービスのカプセル化—サーバは、サービスを要求するメッセージを受け取り、それをどのように処理するかを決定します。サーバとクライアントによって使用されるメッセージ・インターフェイスが変更されない限り、クライアントに影響を与えずにサーバをアップグレードできます。
- 整合性—サーバのコードとデータを集中管理することにより、メンテナンスのコストを抑えることができ、共有データの整合性が保護されます。同時に、クライアントの個別性と独立性を維持できます。
- 位置透過性—サーバは、クライアントと同じマシン、またはネットワーク上のほかのマシンに存在するプロセスです。通常、クライアント / サーバ・ソフトウェアでは、サービス要求をリダイレクトすることによって、サーバのロケーションがクライアントに隠ぺいされます。プログラムは、クライアント、サーバ、またはその両方になることができます。
- メッセージ・ベースの交換—クライアントとサーバは、メッセージを使用して、サービス要求と応答を交換できる疎結合のプロセスです。
- モジュール方式の拡張可能な設計—クライアント / サーバ・アプリケーションはモジュール方式で設計されており、フォルトトレラントにすることができます。フォルトトレラントなシステムでは、障害が発生してもアプリケーション全体がシャットダウンすることはありません。フォルトトレラントなクライアント / サーバ・アプリケーションでは、異常終了したサーバ上で提供されていたサービスがほかのアクティブなサーバ上で利用できる限り、1 つ以上のサーバが異常終了しても、システム全体が停止することはありません。モジュール方式のもう 1 つの利点として、クライアント / サーバ・アプリケーションが、1 つ以上のサービスまたはサーバを追加したりシャットダウンすることによって、システム・ロードの増加や減少に自動的に対応できます。

- プラットフォームからの独立性―理想的なクライアント / サーバ・ソフトウェアは、ハードウェアやオペレーティング・システムのプラットフォームに依存しないので、クライアントとサーバで異なるプラットフォームを使用できます。クライアントとサーバは、それぞれが実行する作業のタイプを最適化して、異なるオペレーティング・システムを使用する異なるハードウェア上で実行できます。
- 再利用可能なコード―サービス・プログラムを複数のサーバ上で使用できます。
- スケーラビリティ―クライアント / サーバ・システムは、水平方向または垂直方向に規模を調整できます。水平方向の調整とは、クライアント・ワークステーションの追加または削除を意味します。パフォーマンスにはわずかにしか影響しません。垂直方向の調整とは、より大型で高速のサーバ・マシンへの移行、またはサーバ・マシンの追加を意味します。
- クライアント / サーバ機能の分離―クライアント / サーバとは、同じマシンまたは別のマシン上で実行されるプロセス間の関係です。サーバ・プロセスは、サービスの供給者です。クライアントは、サービスの要求者です。クライアント / サーバでは、両者の機能がはっきりと区別されます。
- 共有リソース―1 つのサーバが多くのクライアントに対してサービスを同時に提供し、共有リソースに対するアクセスを制御できます。

2 層および 3 層のクライアント / サーバ・アーキテクチャの違い

すべてのクライアント / サーバ・アプリケーションには、次の 3 つの機能単位が含まれています。

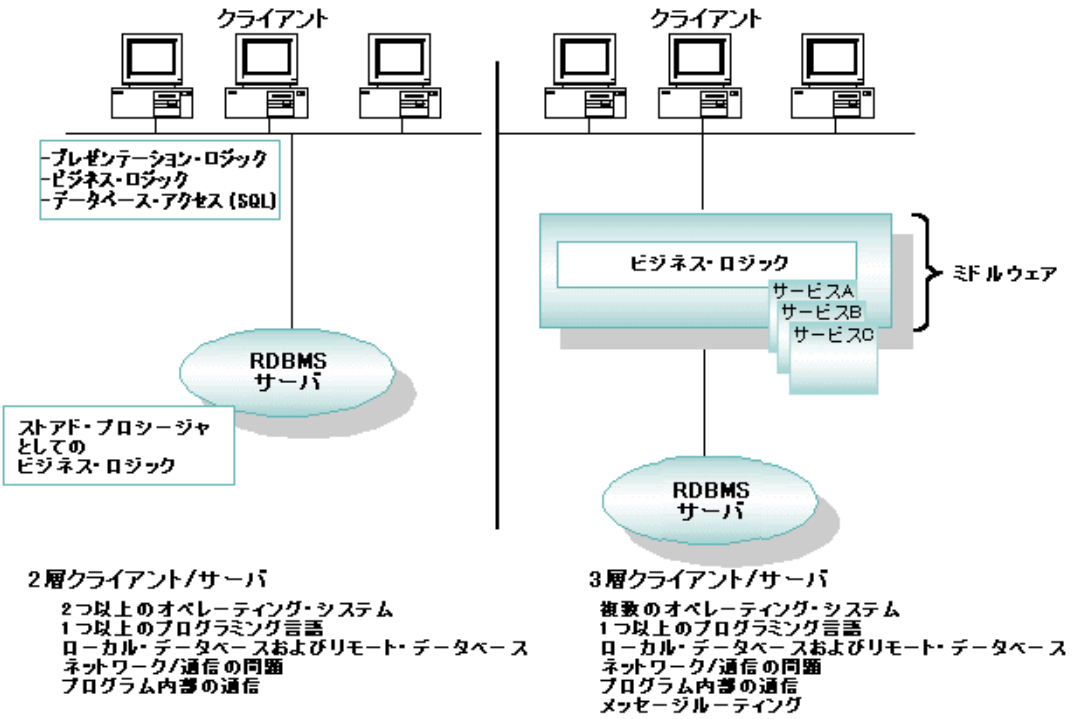
- プレゼンテーション・ロジックまたはユーザ・インターフェイス (たとえば、ATM マシン)
- ビジネス・ロジック (たとえば、顧客が口座残高を照会するためのソフトウェア)
- データ (たとえば、顧客の口座レコード)

これらの機能単位は、クライアントと 1 つ以上のサーバのいずれかに存在します。数多くの種類のアーキテクチャからどれを選択するかは、アプリケーションをどのように分割するか、そして階層間の通信にどのようなミドルウェアを使用するかによって決まります。

2 層クライアント / サーバ・アプリケーションでは、ビジネス・ロジックがクライアント上のユーザ・インターフェイス内に埋め込まれているか、またはサーバ上のデータベース内にストアド・プロシージャとして格納されています。または、ビジネス・ロジックがクライアントとサーバ間で分割されている場合もあります。ストアド・プロシージャを含むファイル・サーバやデータベース・サーバは、2 層アーキテクチャの一例です。

3 層クライアント / サーバ・アプリケーションでは、ビジネス・ロジックがデータやユーザ・インターフェイスから切り離された中間層に存在します。このような方法で、プロセスをユーザ・インターフェイスやデータベースと切り離して管理し、実行することができます。また、3 層システムでは、複数のソースからのデータを統合することもできます。

図 1-1　2 層および 3 層のクライアント / サーバ・モデル



ニーズに応じたクライアント / サーバの各種のアーキテクチャ

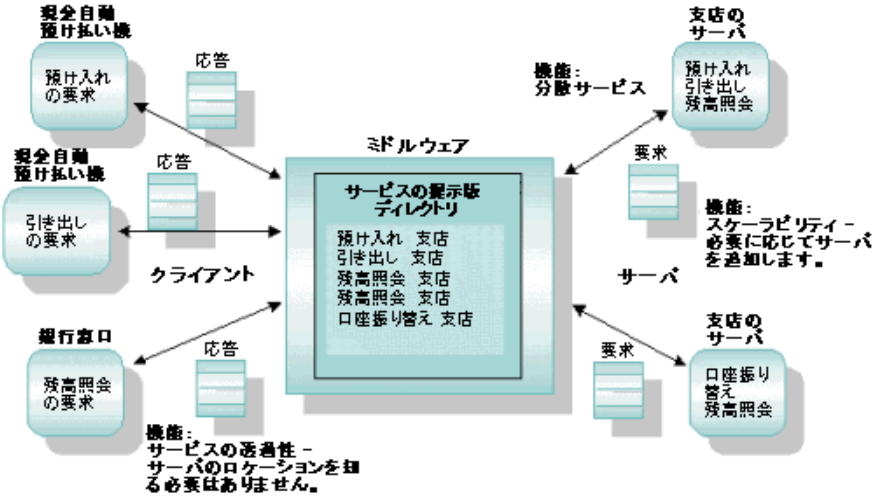
クライアント / サーバ・アーキテクチャは、以下の各状況におけるニーズに対応できます。

- 小規模な事業およびラップトップ―クライアント、ミドルウェア・ソフトウェア、および大部分のビジネス・サービスが同じマシン上で動作します。この方法は、歯科医院、在宅業務、ラップトップ・コンピュータを多用する出張者などの小規模な事業に使用することをお勧めします。
- 中小企業および社内の部署―LAN ベースの単一サーバ・アプリケーションを必要とします。このタイプのアプリケーションのユーザには、複数の医師を擁する病院、複数の部署がある企業、複数の支店を持つ銀行など、中小規模のビジネスが該当します。このタイプのアプリケーションでは、複数のクライアントが 1 つのローカル・サーバと通信します。管理業務は単純です。セキュリティはマシン・レベルでインプリメントされ、障害は簡単に検出できます。
- 大企業―さまざまな機能を提供する複数のサーバを必要とします。インターネット、イントラネット、および社内ネットワーク上に複数のサーバが存在し、そのすべてが高いスケーラビリティを持ちます。サーバは、機能、リソース、またはデータベースごとに分割したり、また障害耐久性やパフォーマンスの向上のために複製することもできます。これは、強力かつ柔軟性の高いモデルです。このクライアント / サーバ・モデルでは、アプリケーションがうまく構築されているかどうかが重要です。作業をサーバ間で分担したり、あるサーバの作業をほかのサーバに委譲する場合もあります。

クライアント / サーバ・モデルにおける BEA Tuxedo システム

BEA Tuxedo システムは、クライアント / サーバ・モデルの中央に配置されます。BEA Tuxedo アプリケーションでは、クライアントがログインし、アプリケーションによって提供されるサービスを要求します。BEA Tuxedo システムでは、透過的な掲示板を通してこれらのサービスが提供されます。掲示板には、サービスを宣言するディレクトリが定義されています。たとえば、銀行業務アプリケーションの場合、掲示板には預け入れ、引き出し、残高照会などのサービスが宣言されます。この後、BEA Tuxedo システムが、要求されたサービスを提供できるサーバ (たとえば、該当する支店など) を見つけます。

図 1-2　銀行業務のサンプル・アプリケーションにおけるクライアントとサーバ



この図には、BEA Tuxedo アプリケーションを構成する基本単位が示されています。

- クライアント―ユーザからの入力を収集し、BEA Tuxedo システムを通して要求をサーバに送信した後、サーバからの応答を収集してユーザに配信するプログラム。
- サーバ―アプリケーションを定義する一連のサービスに、ビジネス・ロジックがカプセル化されたプログラム。
- ミドルウェア―クライアントとサーバ間のやり取りをサポートするために必要なあらゆる分散ソフトウェア。これは、クライアントがサーバからサービスを取得するための手段です。ミドルウェアには、クライアント (要求の発行と応答の受信) およびサーバ (応答の発行) によって使用される API 関数と、クライアントの要求とサーバの応答をネットワーク上で転送するためのメッセージング・パラダイムから構成されます。ミドルウェアには、クライアント上のユーザ・インターフェイス、アプリケーション・ロジック、サーバによって提供されるサービスは含まれません。

この銀行業務のサンプル・アプリケーションでは、クライアント (現金自動預け払い機および銀行窓口) が要求を行い、(支店の) サーバがサービスと応答を提供します。たとえば、ある顧客が、現金自動預け払い機を使って自分の当座預金の残高を調べるとします。現金自動預け払い機 (クライアント) は、残高を取得するためにサーバを呼び出します。サーバは要求を受信し、残高を取得し、その情報を現金自動預け払い機に送信します。

関連項目

- 第 1 章の 5 ページ「クライアント / サーバ・モデルの分析」
- 第 1 章の 10 ページ「BEA Tuxedo 環境におけるクライアント、サーバ、サービス」

BEA Tuxedo 環境におけるクライアント、サーバ、サービス

ここでは、BEA Tuxedo 環境におけるクライアント、サーバ、およびサービスについて説明します。

BEA Tuxedo のクライアントとは

クライアントとは、ユーザからの要求を収集し、その要求を処理できるサーバに渡すプログラムです。アプリケーションのフロント・エンドの一部として、PC またはワークステーション上に置くことができます。また、ATM マシンなどの通信装置を読み取るソフトウェアの中に埋め込むこともできます。このような通信装置からデータが収集され、BEA Tuxedo サーバによって処理される前にフォーマット処理されます。

プログラムがクライアントになるには、アプリケーション・トランザクション・モニタ・インターフェイス (ATMI) と呼ばれる BEA Tuxedo の関数およびプロシージャのライブラリを呼び出せることが必要です。ATMI は、いくつかの言語バインディングでサポートされています。

クライアントは、ATMI のクライアント初期化ルーチンを呼び出すことによって、Tuxedo アプリケーションに参加します。アプリケーションに一度参加すると、クライアントはトランザクション境界を定義したり、アプリケーションのほかのプログラムと通信するための ATMI 関数を呼び出すことができるようになります。クライアントは、ATMI 終了関数を呼び出すことによって、BEA Tuxedo ATMI アプリケーションから分離します。必要な場合だけアプリケーションに参加し、必要なタスクが完了したら分離するようにすると、ほかのクライアントやサーバが使用できるように BEA Tuxedo システムのリソースを解放できます。

分散アプリケーションを構築する場合、ビジネスに関して、処理対象の情報がどのように集められて提供されるかを決定する必要があります。ATMI または CORBA 関数を呼び出す場所やタイミングは、ビジネス・ロジックやビジネス・ルールに従って自由に決めることができます。ある BEA Tuxedo アプリケーションに参加し、いくつかのタスクを実行してこのアプリケーションから分離した後、別の BEA Tuxedo アプリケーションに参加して別のタスクを実行することができます。マルチコンテキスト・アプリケーションを使用している場合は、どのアプリケーションからも分離せずに、複数のアプリケーションでタスクを実行することができます。

BEA Tuxedo サーバとは

BEA Tuxedo サーバとは、一連のサービスを監視し、それらを要求したクライアントに対して自動的にディスパッチするプロセスです。それに対して、サービスとは、ビジネスに必要な特定のタスクを実行するサーバ・プログラム内の関数です。たとえば、銀行には、預け入れを受け取るサービスと、口座残高を報告するサービスが用意されています。あるサーバが、この両方のサービスをクライアントから受け取ったとします。それぞれの要求を該当するサービスにディスパッチするのは、サーバの仕事です。

サービス関数は、SQL などのデータベース・インターフェイスへの呼び出しを通じて、ビジネス・ロジックをインプリメントします。また、ATMI への呼び出しによって、ほかのサービスや問い合わせ、別のリソースにアクセスする場合もあります。これらのサービスが存在するサーバでは、クライアントに応答するか、またはクライアントの要求を別のサービスに転送します。

BEA Tuxedo サービスとは

サービスとは、タスクを実行するアプリケーション・コードのモジュールです。サービスは、コンパイルされてリンクされ、実行可能形式のサーバを生成します。

BEA Tuxedo システムによって提供されるサービス

BEA Tuxedo システムでは、アプリケーションの効率化と管理に役立つ数多くの管理サービスやアプリケーション処理サービスが提供されています。

管理サービス

BEA Tuxedo システムでは、以下の管理タスクを行うためのサービスが提供されています。

- アプリケーション・キュー管理
- 中央集中型のアプリケーション・コンフィギュレーション
- 分散アプリケーション管理
- アプリケーションの動的な再コンフィギュレーション
- イベント管理
- セキュリティ管理
- アプリケーションの起動とシャットダウン
- トランザクション管理
- ワークステーション管理

アプリケーション処理サービス

BEA Tuxedo システムでは、以下の機能をアプリケーションにインプリメントするためのサービスが提供されています。

- データ圧縮
- データ依存型ルーティング
- データ符号化
- データ暗号化
- データ・マーシャリング
- ロード・バランシング

BEA 製品ファミリ

BEA 製品ファミリでは、異種ハードウェアおよびソフトウェア環境でエンド・ツー・エンドの統合を簡単に行うことができ、エンタープライズ規模のトランザクション処理システムを構築できるようになります。BEA 製品は、分散クライアント / サーバ・コンピューティングの柔軟性と、堅牢なミッション・クリティカルなアプリケーションの利点を企業にもたらします。すべての主な業界標準に準拠しているため、開発者はエンタープライズ規模のアプリケーションを 70 以上ものプラットフォームで構築、運用、管理、接続することができます。また、市場をリードするアプリーケーション開発ツール、システム管理ソリューション、およびレガシー・アプリケーションとの完全な統合が可能です。

製品	機能
BEA eLink Adapter for Mainframe	BEA Tuxedo 分散アプリケーションをエンタープライズ・アプリケーションにシームレスに統合するコネクティビティ製品スイート。
BEA Jolt	Java における BEA Tuxedo クライアント API。BEA Jolt は、Java 対応クライアントからの要求を受け取り、BEA Tuxedo アプリケーションの呼び出しに変換します。
BEA Tuxedo and BEA Log Central	BEA Tuxedo および BEA WebLogic Server アプリケーションの管理、統合、運用を行うための完全な環境を提供する BEA アプリケーション管理製品。

製品	機能 (続き)
BEA Tuxedo ATMI の 4 つのコンポーネント ■ BEA Tuxedo ATMI コア・システム ■ Domains 機能 ■ /Q ■ Workstation	■ BEA Tuxedo ATMI コア製品—優れたパフォーマンスと高い信頼性を持つミッション・クリティカルなアプリケーションの構築を可能にします。異機種の分散環境において、スケーラブルな 3 層クライアント / サーバ・アプリケーションを構築するためのフレームワークを提供します。 ■ Domains—BEA Tuxedo クライアント / サーバ・モデルを拡張して、別々に管理されている BEA Tuxedo アプリケーション間にトランザクションの情報交換機能を提供します。 ■ /Q—要求に対して、信頼性の高いキュー処理を行います。 ■ Workstation—各種のオペレーティング・システムに対する完全なクライアント・サポートを提供し、アプリケーションが BEA Tuxedo の完全なインプリメンテーションを必要としないリモート・クライアントを使用できるようにします。
BEA Tuxedo CORBA の 5 つのコンポーネント ■ BEA Tuxedo CORBA コア製品 ■ Bootstrap オブジェクト ■ IIOP リスナ / ハンドラ ■ ORB Client/Server ■ TP Framework	■ BEA Tuxedo CORBA コア製品—BEA Tuxedo の Common Object Request Broker Architecture (CORBA) コンポーネントは、分散型オブジェクト技術を使用して、Object Request Broker (ORB) モデルをオンライン・トランザクション処理 (OLTP) 機能で拡張し、豊富なプログラミング・モデルを提供します。 ■ Bootstrap オブジェクト—クライアント・アプリケーションと BEA Tuxedo ドメインの間の通信を確立します。 ■ IIOP リスナ / ハンドラ—クライアントの要求を取得してサーバ・アプリケーションに送信するプロセス。 ■ ORB—クライアント・アプリケーションがサーバ・アプリケーションに要求を送信するための手段となります。 ■ TP Framework—高レベルのパフォーマンスを達成し、しかもプログラマが複雑な CORBA インターフェイスに煩わされることのないプログラミング・モデルを提供します。
BEA WebLogic Server	Java アプリケーション・サーバ。大規模な分散型の Web アプリケーション、ネットワーク・アプリケーション、およびデータベース・アプリケーションの開発、統合、運用、および管理を行います。

製品	機能 (続き)
BEA WebLogic Collaborate	Busisness-to-Business 電子商取引システムを Web 上で実装するための XML および Java ベースのオープン・マーケット電子商取引プラットフォーム。
BEA WebLogic Commerce Server & Personalization Server	順応性のある個人化された電子商取引アプリケーションを短期間で運用し、顧客および市場の要求に対する迅速な応答を可能にします。
BEA WebLogic Process Integrator	アクティビティのシーケンスを管理し、ビジネス・プロセスのさまざまなアクティビティや手順で要求されるリソースを呼び出して、ビジネス・プロセスを自動化および統合します。

2 BEA Tuxedo ATMI のアーキテクチャ

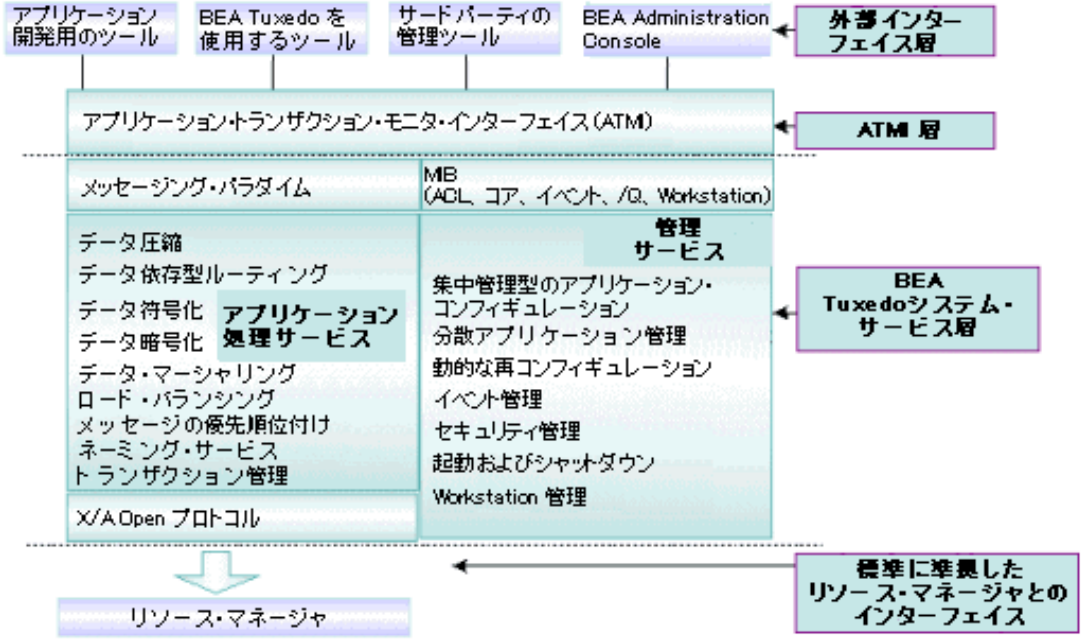
ここでは、次の内容について説明します。

- BEA Tuxedo ATMI 環境の基本アーキテクチャ
- BEA Tuxedo のメッセージング・パラダイム
- BEA Tuxedo ATMI のメッセージ処理
- BEA Tuxedo ATMI アプリケーション処理サービス
- BEA Tuxedo ATMI の管理サービス

BEA Tuxedo ATMI 環境の基本アーキテクチャ

次の図は、BEA Tuxedo ATMI 環境を構成する基本的な要素を示しています。BEA Tuxedo は、システムへの外部インターフェイス、ATMI 層、MIB、BEA Tuxedo システムの各サービス、および標準に準拠したリソース・マネージャとのインターフェイスから構成されます。

図 2-1 BEA Tuxedo ATMI の基本アーキテクチャ



この図で示してあるように、BEA Tuxedo ATMI 環境は次の要素から構成されています。

構成要素	説明
外部インターフェイス層	この層は、ユーザとシステム間のインターフェイスから構成されています。アプリケーション開発ツールと、BEA Administration Console などの管理ツールの両方が含まれています。BEA Administration Console は、標準の管理コンソールとやり取りすることができます。そのため、ユーザは BEA Tuxedo ATMI 環境とネットワークのコンフィギュレーションを 1 つのコンソールから管理できます。また、アプリケーションのアーキテクチャの設計者や開発者は、独自の管理ツールまたはアプリケーション固有のツールや特定分野のツールを MIB の最上位に構築できます。

構成要素	説明 (続き)
アプリケーション・トランザクション・モニタ・インターフェイス (ATMI: Application to Transaction Monitor Interface)	アプリケーションと BEA Tuxedo ATMI 環境間のインターフェイス。ATMI と BEA Tuxedo システムには、トランザクションを行う X/Open DTP モデルがインプリメントされています。ATMI では、抽象的な環境として位置透過性がサポートされ、インプリメンテーションの詳細が隠蔽されます。そのため、プログラマはアプリケーション・コードを変更せずに、複数のプラットフォームに BEA Tuxedo アプリケーションをコンフィギュレーションして運用できます。
メッセージング・パラダイム	クライアントとサーバ間のメッセージ転送に関する各種のモデル。たとえば、要求 / 応答モード、会話モード、イベント、任意通知型通知などがあります。
管理情報ベース (MIB)	BEA Tuxedo ATMI 環境のプログラミングと管理を行うためのインターフェイス。MIB での操作によって、監視、コンフィギュレーション、チューニングなど、すべての管理タスクを行うことができます。MIB では、一度に 1 つのタスクを 1 つのオブジェクトに対して実行したり、ツール・キットを作成してタスクやオブジェクトをバッチ処理することができます。利用可能な MIB については、第 3 章の 3 ページ「使用可能な BEA Tuxedo システムの MIB」を参照してください。
BEA Tuxedo サービス (管理サービスおよびアプリケーション処理サービス)	BEA Tuxedo ATMI 環境のインフラストラクチャによって提供されるアプリケーションの開発と管理のためのサービスや機能。開発者が利用できるアプリケーション処理サービスには、データ圧縮、データ依存型ルーティング、データ符号化、ロード・バランシング、トランザクション管理などがあります。管理サービスには、中央集中型のアプリケーション・コンフィギュレーション、分散アプリケーション管理、ドメインの分割、動的な再コンフィギュレーション、イベントおよび障害の管理、IPC メッセージ・キュー、Workstation 管理などがあります。管理サービスについては、第 3 章の 1 ページ「BEA Tuxedo ATMI インフラストラクチャに対する 3 つの視点」を参照してください。
リソース・マネージャ	データが格納されたソフトウェア製品。データは、ここからアプリケーション・ベースの照会によって取得されます。リソース・マネージャ (RM) は、BEA Tuxedo ATMI 環境とやり取りを行い、XA 標準インターフェイスをインプリメントしています。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・マネージャによって、トランザクション機能とアクションの永続性が可能になります。これらは、グローバル・トランザクション内でアクセスして制御できます。

関連項目

- 第 2 章の 43 ページ「BEA Tuxedo ATMI の管理サービス」
- 第 2 章の 30 ページ「BEA Tuxedo ATMI アプリケーション処理サービス」

ATMI の使用

BEA Tuxedo API である ATMI は、通信、トランザクション、およびデータ・バッファ管理を行うためのインターフェイスで、BEA Tuxedo システムでサポートされているすべての環境で動作します。ATMI によって、アプリケーション・プログラムと BEA Tuxedo システムとが接続されます。ATMI は、広範囲にわたる各種機能に対する 1 つの単純なインタフェースです。ATMI には、トランザクション処理の X/Open DTP モデルがインプリメントされています。

図 2-2 ATMI の使用

ATMI ライブラリには、BEA Tuxedo アプリケーションでグローバル・トランザクションを定義し、制御するための各種の関数が含まれています。グローバル・トランザクションを使用すると、分散アプリケーションにおいて、複数のプログラムとリソース・マネージャにかかわる排他的な操作単位を管理できます。1 つのトランザクションでのすべての操作は、1 つの論理単位として扱われます。そのため、タスクを正常に完了できないプログラムが 1 つでもあると、そのトランザクションではプログラムによってどの操作も実行されません。ほとんどの ATMI 関数では、異なる通信方法がサポートされています。これらの関数は、プログラム間でデータを送受信できるようにして、分散プログラムを互いに結び付けます。すべての ATMI 関数は、型付きバッファでデータを送受信します。次の表は、ATMI 関数 (C および COBOL のバインディング対応) と、それらの関数によって行われる処理を示しています。関数は、タスクごとに分類されています。

表 2-1 ATMI 関数

タスク	C 関数	COBOL 言語の関数	目的
クライアントのメンバーシップ	tpchkauth(3c)	TPCHKAUTH(3cbl)	認証が必要かどうかを確認します。
	tpinit(3c)	TPINITIALIZE(3cbl)	クライアントをアプリケーションに参加させます。
	tpterm(3c)	TPTERM(3cbl)	クライアントをアプリケーションから分離します。
バッファ管理	tpalloc(3c)	N/A	メッセージ・バッファを作成します。
	tprealloc(3c)	N/A	メッセージ・バッファのサイズを変更します。
	tpfree(3c)	N/A	メッセージ・バッファを解放します。
	tptypes(3c)	N/A	メッセージのタイプとサブタイプを取得します。
メッセージの優先順位	tpgprio(3c)	TPGPRIO(3cbl)	最後の要求の優先順位を取得します。
	tpsprio(3c)	TPSPRIO(3cbl)	次の要求の優先順位を設定します。

表 2-1 ATMI **関数 (続き)**

タスク	C 関数	COBOL 言語の関数	目的
要求 / 応答型通信	tpcall(3c)	TPCALL(3cbl)	サービスへの同期要求 / 応答を開始します。
	tpacall(3c)	TPACALL(3cbl)	非同期要求 (ファンアウト) を開始します。
	tpgetrply(3c)	TPGETRPLY(3cbl)	非同期応答を受け取ります。
	tpcancel(3c)	TPCANCEL(3cbl)	非同期要求を取り消します。
会話型通信	tpconnect(3c)	TPCONNECT(3cbl)	サービスとの会話を開始します。
	tpdiscon(3c)	TPDISCON(3cbl)	会話を異常終了します。
	tpsend(3c)	TPSEND(3cbl)	会話中にメッセージを送信します。
	tprecv(3c)	TPRECV(3cbl)	会話中にメッセージを受信します。
高信頼性キュー	tpenqueue(3c)	TPENQUEUE(3cbl)	メッセージをメッセージ・キューに登録します。
	tpdequeue(3c)	TPDEQUEUE(3cbl)	メッセージをメッセージ・キューから取り出します。

表 2-1 ATMI 関数 (続き)

タスク	C 関数	COBOL 言語の関数	目的
イベント・ベースの通信	tpnotify(3c)	TPNOTIFY(3cbl)	クライアントに任意通知型メッセージを送信します。
	tpbroadcast(3c)	TPBROADCAST(3cbl)	複数のクライアントにメッセージを送信します。
	tpsetunsol(3c)	TPSETUNSOL(3cbl)	任意通知型メッセージのコールバックを設定します。
	tpchkunsol(3c)	TPCHKUNSOL(3cbl)	任意通知型メッセージの到着を確認します。
	N/A	TPGETUNSOL(3cbl)	任意通知型メッセージを取得します。
	tppost(3c)	TPPOST(3cbl)	イベント・メッセージをポストします。
	tpsubscribe(3c)	TPSUBSCRIBE(3cbl)	イベント・メッセージをサブスクライブします。
	tpunsubscribe(3c)	TPUNSUBSCRIBE(3cbl)	イベント・メッセージのサブスクリプションを削除します。
トランザクション管理	tpbegin(3c)	TPBEGIN(3cbl)	トランザクションを開始します。
	tpcommit(3c)	TPCOMMIT(3cbl)	現在のトランザクションをコミットします。
	tpabort(3c)	TPABORT(3cbl)	現在のトランザクションをロールバックします。
	tpgetlev(3c)	TPGETLEV(3cbl)	トランザクション・モードであるかどうかを確認します。
	tpsuspend(3c)	TPSUSPEND(3cbl)	現在のトランザクションを一時停止します。
	tpresume(3c)	TPRESUME(3cbl)	トランザクションを再開します。

表 2-1 ATMI **関数 (続き)**

タスク	C 関数	COBOL 言語の関数	目的
サービスの登録と応答	tpsvrinit(3c)	TPSVRINIT(3cbl)	サーバを初期化します。
	tpsvrdone(3c)	TPSVRDONE(3cbl)	サーバを終了します。
	tpservice(3c)	N/A	サービス・エントリ・ポイントのプロトタイプです。
	N/A	TPSVCSTART(3cbl)	サービス情報を取得します。
	tpreturn(3c)	TPRETURN(3cbl)	サービス関数を終了します。
	tpforward(3c)	TPFORWAR(3cbl)	要求を転送します。
動的な宣言	tpadvertise(3c)	TPADVERTISE(3cbl)	サービス名を宣言します。
	tpunadvertise(3c)	TPUNADVERTISE(3cbl)	サービス名の宣言を取り消します。
リソース管理	tpopen(3c)	TPOPEN(3cbl)	リソース・マネージャをオープンします。
	tpclose(3c)	TPCLOSE(3cbl)	リソース・マネージャをクローズします。

注記 ATMI トランザクション管理関数を使用することは必須ではありません。

関連項目

- 『BEA Tuxedo アプリケーション実行時の管理』の第 2 章の 27 ページ「ATMI を使用してシステム・エラーとアプリケーション・エラーを処理する」

BEA Tuxedo のメッセージング・パラダイム

次の表は、アプリケーション開発者が利用できる BEA Tuxedo ATMI のメッセージング・パラダイムを示しています。

表 2-2 BEA Tuxedo ATMI のメッセージング・パラダイム

BEA Tuxedo ATMI のメッセージング・パラダイム	説明
会話型通信	クライアントと専用サーバ間で、双方向で複数のやり取りを行うサービス要求モード
イベント・ベースの通信	パブリッシュ / サブスクライブ・モード
キュー・ベースの通信	配達保証モード
要求 / 応答型通信	同期 (要求元が応答を受け取るまで処理を行わずに待機する) または非同期 (要求元が応答を待つ間も処理を継続する) のサービス要求モード
任意通知型メッセージング	任意のクライアントまたはサーバから任意のクライアントへの通信で、受信側のクライアントが要求または予期していないもの

関連項目

- 第 2 章の 10 ページ「会話型通信」
- 第 2 章の 11 ページ「イベント・ブローカ」
- 第 2 章の 13 ページ「キュー・ベースの通信」
- 第 2 章の 15 ページ「要求 / 応答型通信」
- 第 2 章の 17 ページ「任意通知型通信」
- 第 2 章の 18 ページ「サービス要求の入れ子と転送」

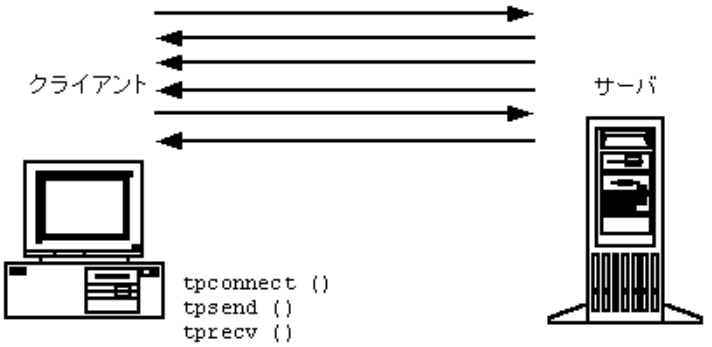
会話型通信

会話型通信は BEA Tuxedo システムのメッセージ交換のパラダイムで、人の会話に似た通信がクライアントとサーバ間でインプリメントされています。この通信方法では、クライアントとサーバ間に仮想の接続が行われます。人が 2 人で会話するように、ある結論に達するまで、2 つのエンティティ間で数多くのメッセージがやり取りされます。通信が行われている間、両者によって会話のポイント (または状態) が「記憶」されます。そのため、一時的な照会、レポート、ファイル転送などの比較的時間のかかる操作をサポートできます。デフォルトで、会話型サーバを使用できます。必要に応じて、自動的にサーバを追加することもできます。

BEA Tuxedo システムでは、アプリケーション内で会話を作成するためのアプリケーション・プログラミング・インターフェイス (API) が提供されています。この API を使用して、クライアントをサーバに接続し、メッセージを送受信し、会話を終了します。

会話は入れ子にすることもできます。ただし、そのためにパフォーマンスが低下する場合があります。会話には、トランザクションまたはサービス要求のいずれかが含まれます。会話型サービスによってサービス呼び出しを行ったり、会話を確立できます。ただし、これらのサービス呼び出しと会話は転送できません。会話は、トランザクションのスコープ内に置いて、トランザクションによって制御されます。

図 2-3 会話型通信



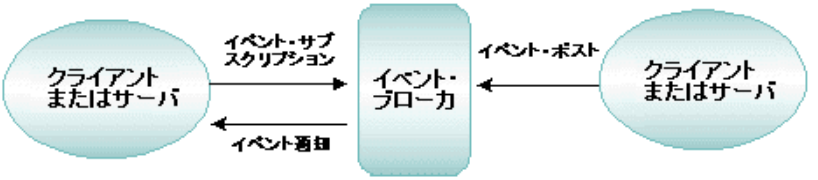
関連項目

- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 11 ページ「会話型通信」

イベント・ブローカ

BEA Tuxedo イベント・ブローカは、任意の数の供給者が任意の数のサブスクライバにメッセージをポストできる通信パラダイムを提供します。イベント・ブローカを使用するクライアントとサーバのプロセスでは、サブスクリプションに基づいて相互に通信が行われるため、このパラダイムはパブリッシュ・アンド・サブスクライブ型の通信と呼ばれます。イベント・ブローカでは、購読料を支払っている購読者だけに新聞を配達するように、処理が行われます。

図 2-4　イベントのポストとサブスクライブ



イベントの発信元 (クライアントまたはサーバのいずれか) は、変更や問題が起こると、イベント・ブローカに通知します。このプロセスはイベントのポストと呼ばれます。通知を受けたイベント・ブローカは、そのイベントの名前をサブスクライバのリストに定義されたイベント名と照合し、合致した場合はそのサブスクライバにイベントを通知します。

関連項目

- 第 2 章の 12 ページ「通知されるイベントのタイプ」
- 第 2 章の 12 ページ「イベントの通知」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 14 ページ「イベント・ベースの通信」

通知されるイベントのタイプ

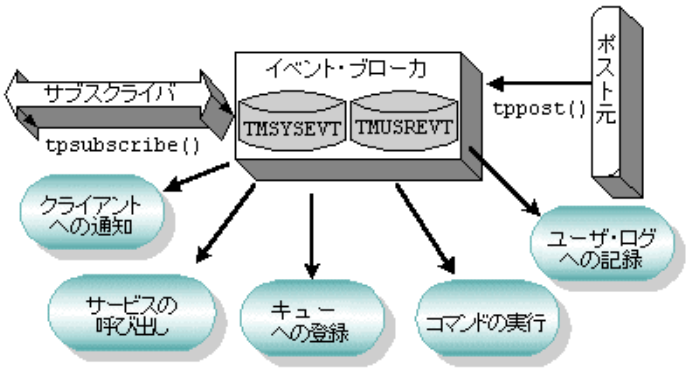
BEA Tuxedo システムでは、次のイベントが通知されます。

- システム・イベント―サーバの異常終了やネットワーク障害など、BEA Tuxedo システム・イベントの詳細が通知されます。クライアントまたはサーバによってイベントがポストされると、イベント・ブローカは、ポストされたイベントの名前をそのイベントのサブスクライバと照合し、合致すると各サブスクリプションに対して指定された処理を行います。
- ユーザ・イベントまたはアプリケーション定義のイベント―特定の条件が満たされた場合に、アプリケーション・プログラムによってイベントがポストされます。たとえば、銀行取引アプリケーションの場合、一定額を超える引き出しが行われるとイベントがポストされます。

イベントの通知

イベント・ブローカでは、パブリッシュ・アンド・サブスクライブ型の通信が行われます。プロセスは、イベント・ブローカにサブスクリプションを登録し、特定のイベントに関心があることを示します。以降、指定されたイベントが発生したことが別のプロセスから通知されると、イベント・ブローカはそのイベントをサブスクライブしているすべてのプロセスに通知します。

図 2-5　イベント・ベースのメッセージング



イベント・ブローカでは、次のメカニズムを使用して、イベントのパブリッシュ (通知の発行) が行われます。

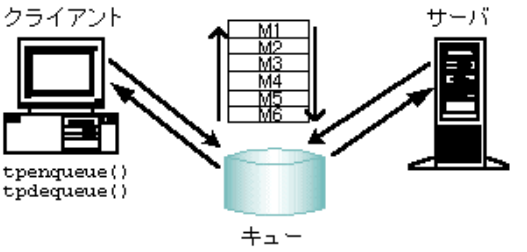
- ディスク・ベースのキューへの登録
- 非同期のサービス呼び出し
- ユーザ・ログのエントリ
- 任意通知型メッセージ
- システム・コマンド

キュー・ベースの通信

BEA Tuxedo システムでは、/Q と呼ばれるキュー・ベースのアーキテクチャが提供されています。これは、アプリケーションでデータを継続的に格納する必要がある場合に使用します。/Q コンポーネントでは、すべてのクライアントまたはサーバがメッセージまたはサービスの要求をキューに格納できます。格納された要求は、必ずトランザクション・プロトコルを使用して送信されるので、安全が保証されています。

BEA Tuxedo システムのキューは、後入れ先出し (LIFO) または先入れ先出し (FIFO)、または時刻や優先順位に基づいて順序付けすることができます。キューの集まりは、キュー・スペースと呼ばれる 1 つのエンティティとして管理され、参照されます。

図 2-6　キュー・ベースのメッセージング



アプリケーション・キュー

アプリケーション・キューは、時間に依存しない通信を行う場合に使用します。時間に依存しない通信とは、プログラムが互いに独立して動作し、互いの通信を同期する必要がない通信方法です。時間に依存しないプログラムでは、互いにアプリケーション・キューにメッセージを残すことによって同期が行われます。メッセージをキューから取り出す場合は、先入れ先出し (FIFO)、優先順位、時刻順など、任意の順序付けスキーマを使用できます。BEA Tuxedo のクライアント・プログラムとサーバ・プログラムでは、メッセージをキューに登録したり、キューから取り出すことができます。同じキューに複数のクライアントやサーバがアクセスできます。

アプリケーション・キューを使用するには、アクセスするキューと、そのキューが置かれたキュー・スペースに名前を付ける必要があります。アプリケーションでは、複数のキュー・スペースを使用できます。また、各キュー・スペースには、複数のメッセージ・キューを格納できます。

アプリケーション・キューはディスク上に存在するため、格納したメッセージはマシンに障害が発生した場合でも利用できます。どのような場合にアプリケーション・キューを使用するかは、業務 (たとえば、注文書の入力) で時間に依存しない同期をいつ行うかによって決定されます。注文書は、ディスク上のキューに登録し、品目や出荷場所など特定の注文条件によって、異なるキュー・スペースに置くことができます。各キュー・スペースに、コストや状態などの条件を追加することもできます。

関連項目

- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 15 ページ「キュー・ベースの通信」

要求 / 応答型通信

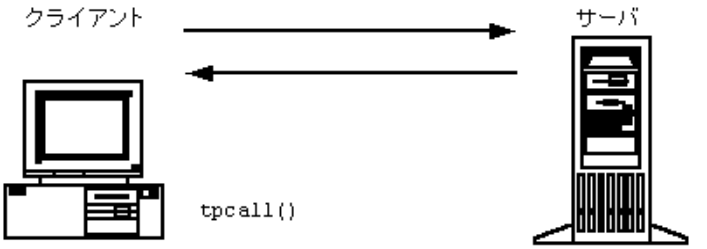
BEA Tuxedo システムでは、要求 / 応答型通信のインプリメントに IPC メッセージ・キューが使用されます。キューは、コネクションレス型通信の基本要素です。各サーバには、要求キューと呼ばれるプロセス間通信 (IPC) メッセージ・キューが割り当てられ、各クライアントには応答キューが割り当てられます。そのため、クライアント・アプリケーションでは、サーバとの継続的な接続を確立する代わりに、要求をサーバのキューに登録して、サーバにその要求を送信できます。また、アプリケーションの応答キューからメッセージを取り出して、サーバからのメッセージを確認して取得できます。

要求 / 応答型モデルは、以下に説明するように、同期と非同期の両方のサービス要求に使用できます。

同期メッセージング

同期呼び出しでは、クライアントがサーバに要求を送信し、クライアントが待機している間にサーバが要求された処理を行います。その後、サーバがクライアントに応答を送信し、クライアントが応答を受信します。

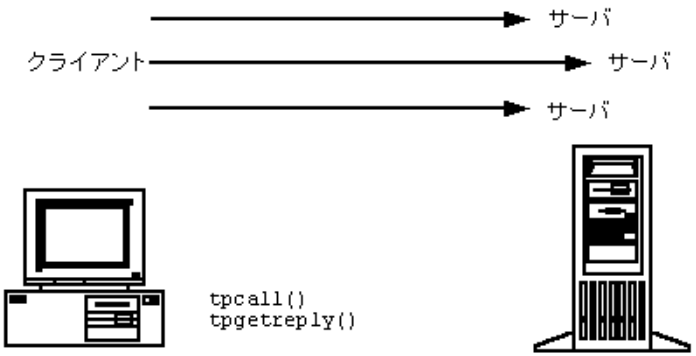
図 2-7　同期要求 / 応答型通信



非同期メッセージング

非同期呼び出しでは、BEA Tuxedo クライアントは、送信したサービス要求の処理が完了するまで待機しません。要求を発行した後も、別のタスク (ほかの要求の発行など) を実行します。最初の要求への応答が返されると、クライアントはその応答を取得します。

図 2-8 非同期要求 / 応答型通信



関連項目

- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 7 ページ「要求 / 応答型モデル (同期呼び出し)」

任意通知型通信

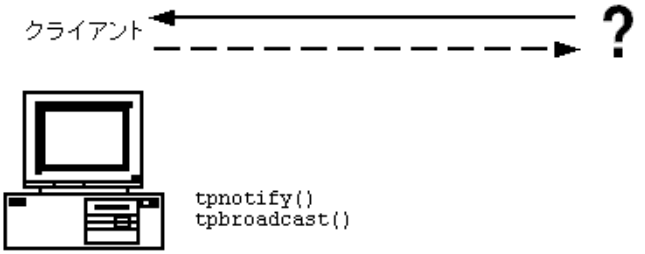
BEA Tuxedo システムでは、任意通知型通知と呼ばれる通信パラダイムが提供されています。任意通知型通知が発生すると、BEA Tuxedo クライアントは要求していないメッセージを受け取ります。この機能によって、アプリケーション・クライアントは、アプリケーション固有のイベントが発生したときに、リアルタイムで明示的に通知を要求しなくても、そのイベントの通知を受け取ることができます。

任意通知型メッセージは、名前 (`tpbroadcast`) または以前に処理されたメッセージと共に受け取った識別子 (`tpnotify`) としてクライアント・プロセスに送信されます。`tpbroadcast` 関数を使用すると、サービスまたは別のクライアントにメッセージを送信できます。メッセージは、狭い範囲または広い範囲のターゲットに対して送信できます。配達保証付きまたは配達保証なしのメッセージをポイント・ツー・ポイントの通知によって個々のクライアントに送信したり (`tpnotify`)、クライアントのグループに対して情報を送信する (`tpbroadcast`) ことができます。たとえば、クライアントから照会された口座が解約されていることをサーバからそのクライアントだけに通知できます。または、あるマシンがメンテナンスのために特定の時刻にシャットダウンされることをそのマシン上のすべてのクライアントに通知することもできます。

プロセスが特定のイベント (メンテナンスのためにシャットダウンされるマシンなど) について通知が必要な場合は、自動的に通知されるように要求をシステムに登録できます。登録されたイベントは、発生するたびにクライアントまたはサーバに通知されます。イベントのこのような自動通信は、任意通知型通知と呼ばれます。

イベントを生成したり、イベントについて任意通知型通知を受信するクライアントおよびサーバの数に制限はありません。そのため、このカテゴリの通信は、管理業務が複雑になる場合があります。BEA Tuxedo システムには、任意通知型通知を管理するイベント・ブローカと呼ばれるツールが用意されています。

図 2-9 任意通知型通知のメッセージング



関連項目

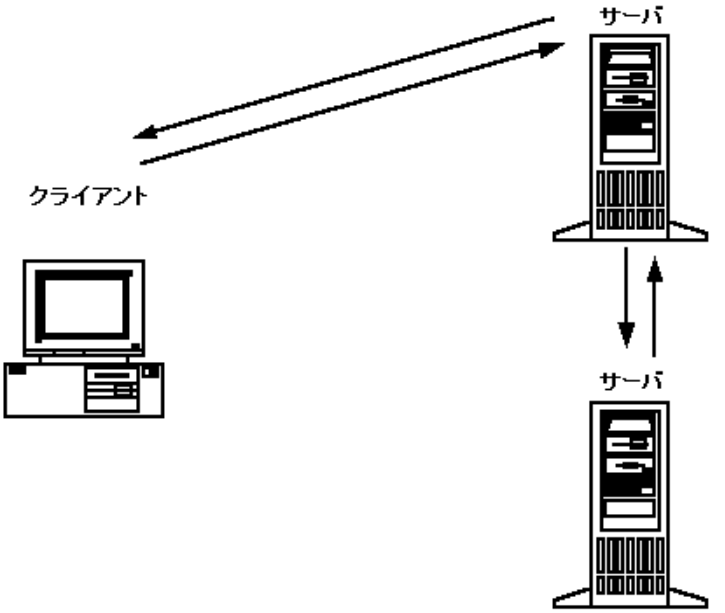
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 13 ページ「任意通知型通知」

サービス要求の入れ子と転送

入れ子になった要求

BEA Tuxedo システムでは、サービスがクライアントとして動作して、別のサービスを呼び出すことができます。入れ子のレベルは、2 レベルに制限されています。これは、3 層クライアント / サーバ・アーキテクチャ、つまりプレゼンテーション・ロジック層、ビジネス・ロジック層、およびデータベース層から構成されるシステムで特に有用です。このようなシステムでは、プレゼンテーション層で特定のビジネス関数が要求され、データベースに 1 つ以上の照会が行われます。入れ子は 2 レベルに制限されているため、パフォーマンスが低下することはありません。

図 2-10　入れ子になったサービス要求



入れ子になった要求の利点

入れ子になった要求を使用すると、各コード部分が限定された処理だけを行うので、コードが少なくて済み、再利用も可能になります。ただし、システムのサービスが複数のサーバに分散している場合は、要求を入れ子にするとパフォーマンスが低下する場合があります。入れ子になった要求が処理されている間、元のサービス (入れ子になった要求を発行したサービス) は、応答を待機する必要があります。つまり、応答を受信するまで、別の要求を処理することはできません。そのため、このサービスを持つサーバの要求キューに、メッセージがバックアップされます。

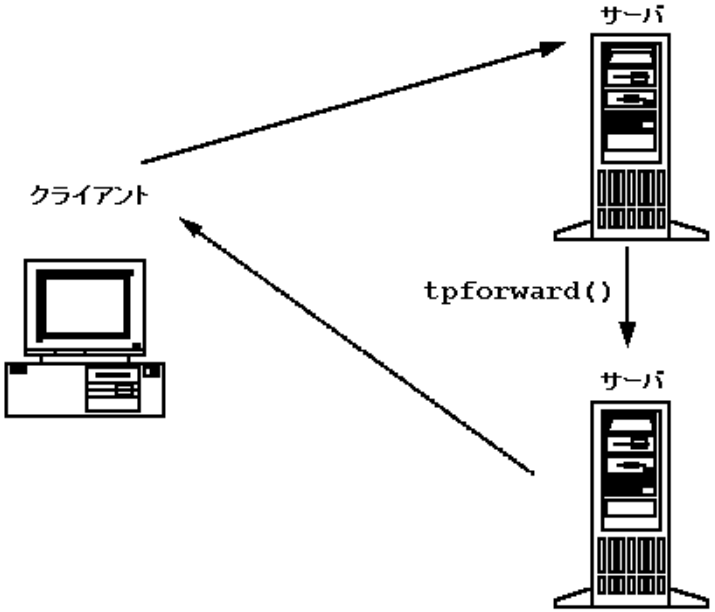
入れ子になったサービス要求の例

ある顧客が現金自動支払機を使って、普通口座から当座預金に預金を移すとします。振替に必要な操作は、BEA Tuxedo アプリケーションによって行われます。まず、顧客に代わって、クライアントが TRANSFER サービスへの要求を発行します。この要求は、そのサービスを提供するサーバのキューに置かれます。次に、TRANSFER サービスが WITHDRAW および DEPOSIT という 2 つのサービスを要求します。この 2 つのサービスは、別のサーバによって行われます。WITHDRAW および DEPOSIT の各サービスが TRANSFER サービスに応答を返します。最後に、TRANSFER サービスがクライアントの応答キューに対して応答を送ります。クライアントがキューから応答を取得すると、現金自動支払機の画面に振替が完了したことを通知するメッセージが表示されます。

要求の転送

入れ子になったサービス要求に代わる方法として、要求の転送があります。クライアントの要求を処理する代わりに、サービスはその要求を別のサービスに渡します。要求を受け取ったサービスは、要求を処理するか、または別のサービスに渡します。

図 2-11 サービス要求の転送



要求を転送できる回数に制限はありません。要求を転送したサービスは、その要求を受け取った別のサービスからの応答を待つ必要がありません。そのため、入れ子になった要求とは異なり、転送ではサーバがブロックされることはありません。ただし、転送は X/OPEN のプロトコル X/ATMI でサポートされていないため、アプリケーションによっては支障が出る場合があります。

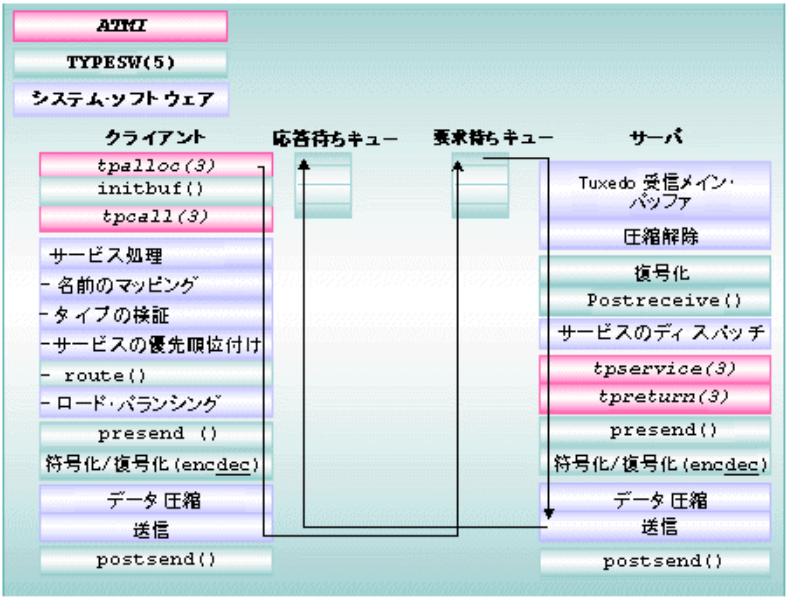
関連項目

- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の第 1 章の 10 ページ「転送された呼び出し」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の第 1 章の 9 ページ「入れ子になった呼び出し」

BEA Tuxedo ATMI のメッセージ処理

BEA Tuxedo ATMI 環境内のすべての通信は、メッセージを転送することによって行われます。BEA Tuxedo ATMI 環境では、オペレーティング・システムのプロセス間通信 (IPC) メッセージ・キューを使用して、クライアントとサーバ間でサービス要求のメッセージをやり取りします。システムのメッセージとデータは、バッファに格納され、クライアントとサーバのキュー間でやり取りされます。これは、オペレーティング・システムでサポートされたメモリ・ベースのキューです。BEA Tuxedo ATMI 環境では、メッセージは型付きバッファにパッケージ化されます。このバッファには、メッセージ・データと、送信されたメッセージ・データのタイプを識別するデータが格納されます。

図 2-12 要求の処理



クライアントは、ATMI 関数を使ってサービスを名前で要求します。ネーミング機能を使って MIB が照会され、指定されたサービスが現在利用可能かどうかが確認されます。BEA Tuxedo システムでは、特定の条件 (メッセージの値) を満たすメッセージを特定のサーバにマップする自動ルーティング・オプションが使用されています。これは、データ依存型ルーティングと呼ばれます。メッセージにデータ依存型ルーティングが使用される場合、ルーティング・アルゴリズムではバッファに格納されたデータが使用されます。このアルゴリズムによって、サービス要求を処理できるサーバのグループが選択されます。一部のサーバに要求が集中し、同じサービスを宣言するほかのサーバがアイドル状態になることを避けるために、BEA Tuxedo システムではサービス要求をすべてのサーバ間で均等に分散するために MIB を使用しています。これはロード・バランシングと呼ばれます。

選択されたサーバに対してローカルのサービス要求が用意され、そのサーバのキューに定義済みの優先順位で登録されます。これはサービスの優先順位付けと呼ばれます。サービス要求がサーバ上に置かれると、ランタイム・システムによってメッセージが優先順位に従って取得されます。メッセージは、適切なサービスにディスパッチされて、処理されます。その後、クライアントのキューに結果が返されます。

BEA Tuxedo システムで提供されるソフトウェアには、メッセージ処理の際にアプリケーションが自動的に、そして定期的に使用できる機能があります。これらの機能には、たとえば、データの符号化と復号化、データの圧縮と圧縮解除、トランザクション・コンテキストの設定、セキュリティに関する処理などがあります。また、BEA Tuxedo システムのソフトウェアは、サービス関数をディスパッチし、それを適切に前処理されたバッファに渡すことによって、アプリケーションのビジネス・ロジックを呼び出します。

サービス・ルーチンが実行されると、応答 (型付きバッファ) が返されます。ランタイム・システムは、メッセージを自動的に符号化することによって、クライアントへの応答を用意します。つまり、バイトの並び順が異なるマシン間で転送できるようにデータをパッケージ化し、データがネットワークやプラットフォームの境界を越えて転送できるようにします。その後、メッセージをクライアントに送信します。このプロセスはデータの符号化と呼ばれます。クライアント上のランタイム・システムは、応答メッセージを取得し、必要に応じて復号化した後、FML バッファ (またはほかのメッセージ・バッファ・タイプのバッファ) を送信してアプリケーション・データをパッケージ化します。必要に応じて、タイプの検証、符号化、ルーティング、およびロード・バランシングが行われます。サービス要求は、同期でも非同期でも実行されます。

リモート要求は、ローカル・ブリッジを通ってリモート・マシンに到達します。リモート・マシンでは、リモート・ブリッジがクライアントとして動作し、クライアントとサーバが同じマシン上にあるかのように要求が処理されます。ブリッジは、標準のデータ符号化 / 復号化を行い、標準のネットワーク転送を使用して通信します。クライアントとサーバからは、ブリッジは通常のローカル・サーバのように見えます。

サービス要求処理の利点

- コネクションレス型の処理―この処理をクライアント/サーバの直接通信と組み合わせると、接続の確立に関連するオーバーヘッドを減らすことができます。
- ネットワーク・トラフィックの削減―サービス要求はリモート・マシン上の複雑なサービスを呼び出し、必要最小限のデータだけを送信し、最小限の結果を受信します。

関連項目

- 第 2 章の 9 ページ「BEA Tuxedo のメッセージング・パラダイム」
- 第 2 章の 24 ページ「型付きバッファ」

型付きバッファ

すべての ATMI 関数は、型付きバッファを使ってデータを送受信します。BEA Tuxedo システムでは、異種のマシン間での翻訳とデータ変換が可能です。BEA Tuxedo プログラムではバッファが使用されているので、異なるデータ表現の異種プラットフォーム間でやり取りされるデータを変換する必要はありません。

バッファとは、データの論理的な入れ物として機能するメモリ領域です。バッファにメタデータ (バッファ自体に関する情報) が含まれていない場合、そのバッファは型なしのバッファと呼ばれます。バッファ内に格納できる情報として、タイプとサブタイプ、またはバッファを特徴付ける文字列名などのメタデータが含まれている場合、そのバッファは型付きバッファと呼ばれます。

型付きバッファは、BEA Tuxedo システムでサポートされる任意のプロトコルを使用して、どの種類のネットワークからもどの種類のオペレーティング・システムにも送ることができできます。異なるデータ表現のプラットフォーム上で使用することもできます。そのため、型付きバッファを使用すると、異種マシン間での翻訳とデータ変換のためのタスクが簡略化されます。

BEA Tuxedo システムでは、次の 5 種類の型付きバッファがサポートされています。

- STRING
- VIEW
- CARRAY
- FML
- XML

バッファ・タイプは、コンフィギュレーション・ファイルの MACHINES セクションに定義されている ENVFILE パラメータで指定します。コンフィギュレーション・ファイルの SERVERS セクションの ENVFILE パラメータでバッファ・タイプを指定したり上書きすると、バッファ・タイプが必要なプロセスでそれらのバッファ・タイプを使用できなくなる場合があります。

メッセージ・バッファの各種のタイプの定義については、『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の tuxtypes(5) の tm_typesw の説明を参照してください。tm_typesw を変更して、特定のサーバで特に必要なバッファ・タイプだけを含むようにしておくと便利です。

バッファ・タイプの特徴

ATMI 通信関数を使用する場合、まずアプリケーションで tpalloc を使用して、バッファのサイズ、タイプ、およびサブタイプ (省略可能) を指定し、システムからバッファを取得する必要があります。BEA Tuxedo システムによってバッファ・タイプが認識されて処理されるため、データは BEA Tuxedo システムによってサポートされているどの種類のネットワーク、プロトコル、およびオペレーティング・システム上でも転送できます。次の表は、BEA Tuxedo の環境で使用できる各バッファ・タイプを示しています。

表 2-3 各バッファ・タイプの特徴

型付きバッファ	説明	目的
CARRAY	文字配列型は、曖昧に処理される文字の集まりです。 ■ 文字にはどのような解釈も行われません。 ■ サブタイプは指定されません。 ■ ATMI 関数への入力として使用される CARRAY メッセージには、アプリケーションでバッファ長を指定する必要があります。	BEA Tuxedo システムによって解釈されず、データ依存型ルーティング、符号化、または復号化を必要としないデータ
STRING	NULL 文字で終了する非 NULL 文字の集まりです。データ型は文字であり、データ長はバッファ内の文字数を NULL 文字まで数えた値です。サブタイプを指定する必要はありません。	C プログラム

表 2-3 各バッファ・タイプの特徴（続き）

型付きバッファ	説明	目的
FML	フィールド操作言語 (FML) は、タグ付き値を格納するデータ構造です。値には型が付いていて、複数指定したり長さを変えることができます。 FML バッファは、フィールドの作成、変更、削除、またはアクセスの操作に使用される抽象的なデータ型です。プログラムでは、識別子を参照することによって、フィールド化バッファ内のフィールドにアクセスしたり更新します。FML 関数によって、フィールドのロケーションとデータ型が実行時に変換され、操作が実行されます。 FML へのインターフェイスには、フィールド識別子およびフィールド長として 16 ビットを使用するもの (FML16) と、32 ビットを使用するもの (FML32) があります。 ■ 16 ビットのインターフェイスでは、一意なフィールドを約 8000 個まで、文字列や配列を 64,000 バイトまで使用できます。 ■ 32 ビットのインターフェイスでは、一意なフィールドを数百万個、バッファ長として 20 億バイトまで使用できます。 この 2 つのインターフェイスの機能は同じです。FML の威力は、その柔軟性にあります。各メッセージに対するアプリケーションのニーズに応じて、バッファのサイズを変えることができます。文字フィールドも長さを変えられるため、不要な領域がなくなります。 フィールド化バッファによって、アプリケーションはデータから独立できます。アプリケーションの作成時に、フィールド化バッファのどこに、どのようにデータが格納されるのかを知る必要はありません。FML によって対応するフィールドにアクセスできるため、単にフィールド名を指定すれば値が返されます。また、FML には変換関数もあるため、記憶領域の種類に関係なく、特定のデータ形式としてフィールドを格納したり、取り出すことができます。 FML バッファは、1 つのフィールドに複数の値を格納することもできます。フィールド化バッファの可変長の型により、フィールドの複数オカレンスを格納したり、取り出すことができます。 フィールド化バッファは、メッセージごとに異なるフィールドの集まりをクライアントとサーバ間でやり取りしたり、アプリケーション・キューにフィールドを格納する場合に使用すると便利です。特にクライアントとサーバ間のインタフェースが変わる可能性がある場合は、FML を使用することをお勧めします。	■ 通信 ■ フィールドの作成、変更、削除、またはアクセス

表 2-3 各バッファ・タイプの特徴（続き）

型付きバッファ	説明	目的
VIEW	VIEW は単に C 構造体または COBOL レコードで、それに対応してレコード内でどのフィールドと型がどの順序で現れるかが定義されています。このバッファは、データ要素の固定的な集まり、または構造体やレコードに使用します。サブタイプを使用して、レコード形式名を指定します。 VIEW レコードはフラットなデータ構造です。ほかの構造体に存在する構造体、または構造体やポインタの配列はサポートされていません。長精度型、文字型、10 進数型などのデータ型がサポートされています。 VIEW は、BEA Tuxedo システムで C 構造体および COBOL レコードを使用するために提供されています。BEA Tuxedo のランタイム・システムでは、実行時に読み取られる VIEW 記述に基づいて、レコード形式が認識されます。VIEW を割り当てる場合、VIEW のバッファ・タイプおよび名前と合致するサブタイプがアプリケーションで指定されます。ランタイム・システムでは、次の操作が行われます。 ■ 構造体のサイズに基づいて、必要な領域が決定されます。アプリケーションでバッファ長を指定する必要はありません。 ■ 要求または応答で送信するデータ量が計算され、異機種間でメッセージをやり取りする場合に符号化と復号化が行われます。	BEA Tuxedo アプリケーションで使用される C 構造体および COBOL レコード
XML (Extensible Markup Language)	XML バッファを使うと、XML 技術を使用してアプリケーション内やアプリケーション間でデータを送受信できます。BEA Tuxedo アプリケーションでは、単純な XML 型バッファの送受信や、それらのバッファを適切なサーバにルーティングできます。解析など、XML 文書を扱うためのすべてのロジックはアプリケーション内に定義されています。XML 文書は、次の要素から構成されます。文書のテキストを符号化した一連の文字、文書の論理構造、および構造に関連するメタ情報。 BEA Tuxedo システムの XML パーサは、文字符号化の自動検出、文字コード変換、要素内容および属性値の検出、およびデータ型の変換を行います。 XML 型バッファでは、データ依存型ルーティングがサポートされています。	■ XML 文書およびデータグラム ■ Web サーバからユーザのブラウザへのデータ転送など、人とマシンのデータ交換 ■ アプリケーション間、またはマシンからマシンへのデータ交換

関連項目

- 『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 第 3 章の 28 ページ「バッファのカスタマイズ」

MIB

MIB プログラミング・インターフェイスを使用すると、BEA Tuxedo システムの操作を簡単に管理することができます。特に、独自のプログラムによってアプリケーションの監視、コンフィギュレーション、チューニングを行うことができます。MIB は、次のように定義されます。

- FML 属性として定義され、インプリメンテーションに依存しない管理データベース。
- ATMI 関数を使用して、BEA Tuxedo システムへの照会 (get を使用してシステムから情報を取得すること) や、BEA Tuxedo システムの更新 (set を使用してシステムの情報を変更すること) をいつでも行うことができるプログラミング・インターフェイス。これらの関数には、tpalloc、tprealloc、tpgetrply、tpcall、tpacall、tpenqueue、tpdequeue などがあります。

関連項目

- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の MIB(5)
- 第 2 章の 29 ページ「MIB ユーザのタイプ」
- 第 2 章の 29 ページ「MIB のクラス、属性、状態」

MIB ユーザのタイプ

MIB では、システム管理者、システム・オペレータ、その他という 3 つのタイプのユーザが定義されています。次の表は、各タイプについて示しています。

ユーザのタイプ	説明
システム管理者	アプリケーションの正常な実行を維持する責任者。管理者には、すべての管理ツールと MIB のすべての管理機能を使用する権限が与えられます。管理者は、実行中のアプリケーションのコンフィギュレーション、管理、変更を行うことができます。
システム・オペレータ	稼動するアプリケーションの日々の操作を監視し、問題に対応します。オペレータは、実行中のアプリケーションに関する統計を監視します。イベントと警告に対して、サーバの起動やマシンのシャットダウンなどの対処を行う場合もあります。オペレータは、アプリケーションの再コンフィギュレーション、サーバまたはマシンの追加、マシンの削除は行いません。
その他	MIB を読み取る必要はあるが、アプリケーションを変更する権限を持たない人またはプロセス (カスタム・プログラムなど)。

MIB のクラス、属性、状態

クラスとは、BEA Tuxedo アプリケーションを構成するサーバやマシンなどのエンティティの種類です。属性とは、クラス内のオブジェクトの特徴を表すもので、ID、状態、コンフィギュレーション・パラメータ、実行時の統計などがあります。属性には、MIB の操作と応答に共通するもの、および個々のクラスに共通するものがあります。すべてのクラスには、オブジェクトの状態を示す状態属性があります。MIB 上でオブジェクトの状態を変更する操作を呼び出している場合、オブジェクトの状態はユーザ変更の状態、または新しく変更された状態のいずれかになります。

MIB(5) リファレンス・ページに定義されている共通の属性は、クラスに依存しません。これらの属性は、入力操作を制御したり、ユーザが何をしようとしているかを MIB に伝達したり、特定のクラスに依存しない出力バッファの特徴をプログラマに示します。

BEA Tuxedo ATMI アプリケーション処理サービス

BEA Tuxedo ATMI 環境では、次のアプリケーション処理サービスが提供されています。

- データ圧縮
- データ依存型ルーティング
- データ符号化
- データ暗号化
- データ・マーシャリング
- ロード・バランシング
- メッセージの優先順位付け
- サービスおよびイベントのネーミング

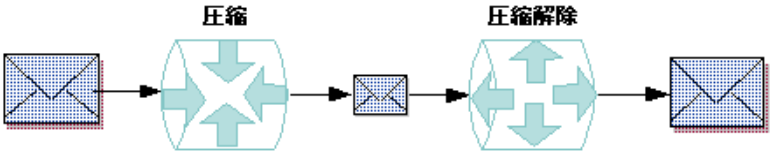
データ圧縮

データ圧縮とは、ネットワーク上またはリモート・ドメインへの転送を高速化するために、アプリケーション・バッファを縮小するプロセスです。アプリケーション・バッファの最大サイズを指定し、そのサイズ以上のアプリケーション・バッファが自動的に圧縮されるように設定できます。バッファが送信先に到着すると、データは圧縮解除されて元のサイズに戻ります。

マシン間でファイルを送付する前にデータ圧縮を行うと、ネットワークのパフォーマンスが向上します。圧縮の過程ではデータのスクランブリングも行われるので、セキュリティも若干強化されます。

注記 データ圧縮は、暗号化でも頻繁に行われます。

図 2-13　データ圧縮



データ依存型ルーティング

BEA Tuxedo システムでは、データ依存型ルーティングと呼ばれる操作を使用して、クライアントが同じサービスへの要求をそのサービスの複数のコピーに対して送信できるようにします。サービスのどのコピーが最終的に要求を受け入れて処理するのかは、要求メッセージのデータによって決まります。管理者がアプリケーションにデータ依存型ルーティングを設定すると、クライアントの要求は要求内のデータに基づいて自動的にサーバにルーティングされます。

百科事典の第 1 巻に「A」で始まる項目が複数含まれているように、アプリケーションに同じサービスのコピーが複数含まれている場合、各コピーに対して一意な目的が割り当てられます。そのサービスのすべてのコピーのリストと、各コピーの目的に関する情報の識別文字列は、BEA Tuxedo の掲示板のルーティング・テーブルに保持されています。システムがクライアントの要求を受信すると、要求メッセージ内の識別文字列を読み取り、その文字列を掲示板のルーティング・テーブルで検索します。文字列が合致すると、クライアントの要求を転送する適切なサーバが識別されます。

注記　掲示板のルーティング・テーブルは、必要に応じて変更できます。

データ依存型ルーティング

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

- 水平分離型データベース
- ルール・ベース・サーバ
- 分散アプリケーション

水平分離型データベースとは、セグメントに分割された情報リポジトリです。各セグメントには、異なるカテゴリの情報が格納されます。これは、各本棚に異なるカテゴリ (伝記、フィクションなど) の本が収納されている図書館に似ています。

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

分散アプリケーションは、ネットワークを通して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 1 つ以上のローカルまたはリモートのクライアントから構成されます。クライアント (またはクライアントとして動作するサーバ) は、特定のサービスに対する要求を発行します。要求のアドレスは、その要求を実行できるサーバを識別するデータ (要求と同じバッファで転送されるデータ) によって決定されます。要求を実行できるサーバが複数存在する場合もあります。BEA Tuxedo システムでは、掲示板のルーティング条件とデータが照合されて、要求を受け取るサーバが決定されます。

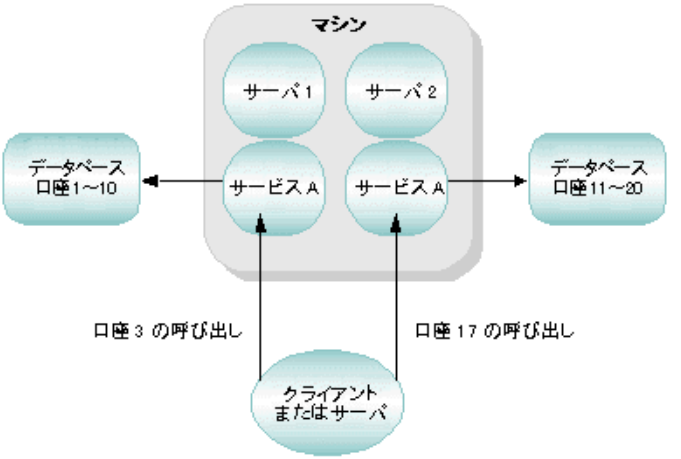
水平分離型データベースでのデータ依存型ルーティングの例

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

1. 2 つのサービス要求で指定された口座番号 (3 と 17) を取得します。
2. どのサーバがどのデータ範囲の処理を行うかを示す、BEA Tuxedo の掲示板のルーティング・テーブルを調べます。この例では、口座 1 ～ 10 に対するすべての要求をサーバ 1 が処理し、口座 11 ～ 20 に対するすべての要求をサーバ 2 が処理しています。
3. それぞれの要求を該当するサーバに送信します。つまり、口座 3 への要求をサーバ 1 に転送し、口座 17 への要求をサーバ 2 に転送します。

次の図は、このプロセスを示しています。

図 2-14　水平分離型データベースでのデータ依存型ルーティング



ルール・ベース・サーバでのデータ依存型ルーティングの例

次の規則を持つ銀行取引アプリケーションがあるとします。

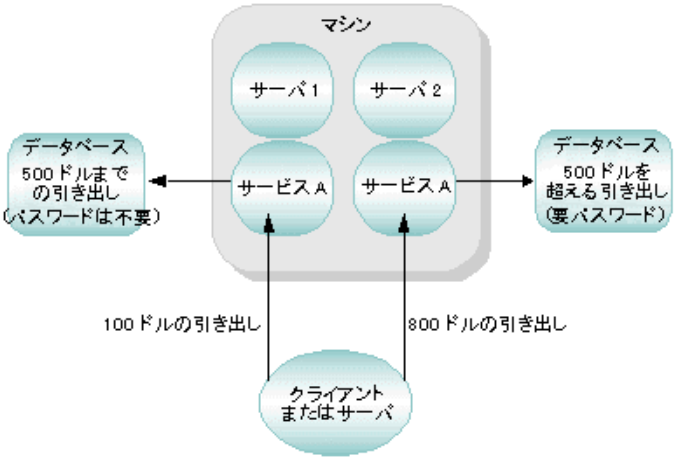
- 顧客は、特別なパスワードを入力しなくても、5 万円まで引き出すことができます。
- 5 万円を超える額を引き出すには、特別なパスワードを入力する必要があります。

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

1. 2 つのサービス要求で指定された引き出し額 (1 万円と 8 万円) を取得します。
2. BEA Tuxedo の掲示板のルーティング・テーブルで、要求された額をどのサーバが処理するのかを確認します。この例では、5 万円までのすべての引き出し要求をサーバ 1 が処理し、5 万円を超えるすべての引き出し要求をサーバ 2 が処理しています。
3. それぞれの要求を該当するサーバに送信します。つまり、1 万円の要求をサーバ 1 に転送し、8 万円の要求をサーバ 2 に転送します。

次の図は、このプロセスを示しています。

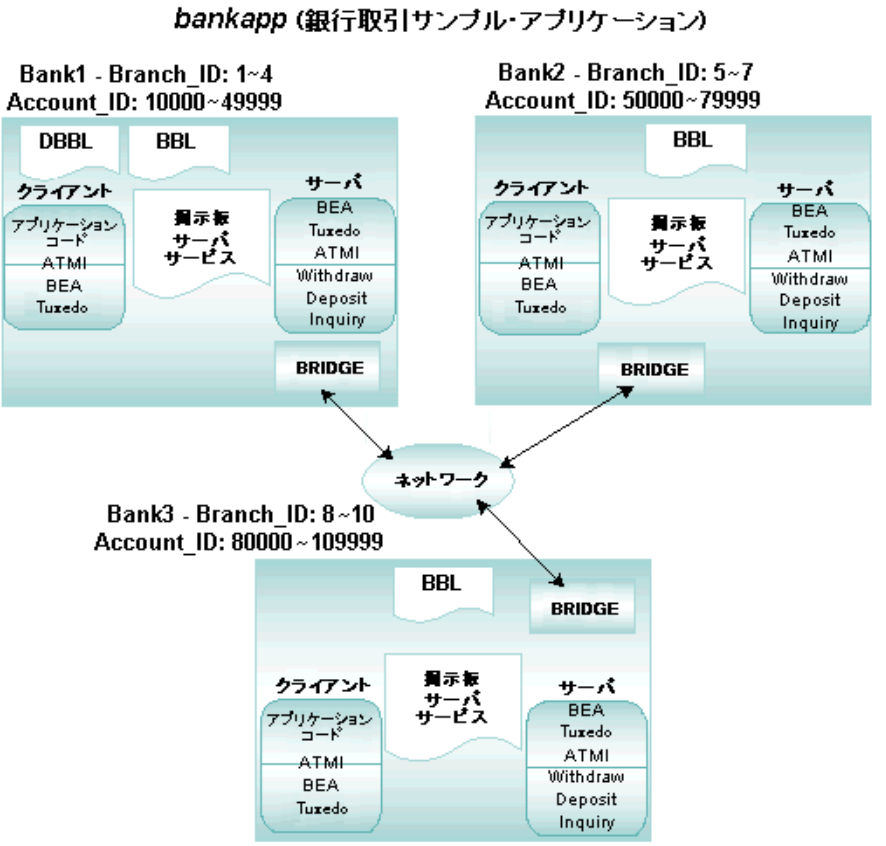
図 2-15　ルール・ベース・サーバでのデータ依存型ルーティング



分散アプリケーションでのデータ依存型ルーティングの例

次の図は、クライアントの要求がサーバにルーティングされる方法を示しています。この例では、bankapp という銀行取引アプリケーションでデータ依存型ルーティングが使用されています。bankapp には、3 つのサーバ・グループ (BANK1、BANK2、BANK3) と 2 つのルーティング条件 (Account ID と Branch ID) があります。WITHDRAW、DEPOSIT、および INQUIRY の 3 つのサービスは、Account_ID フィールドを使用してルーティングされます。サービス OPEN および CLOSE は、Branch_ID フィールドを使用してルーティングされます。

図 2-16　ルーティング条件を使用した銀行取引のサンプル・アプリケーション



前の図では、要求は次の表に示すようにルーティングされます。

引き出し、預け入れ、照会、開設 / 閉鎖を行う口座	ルーティング先
支店番号 1 ～ 4 の口座番号 10000 ～ 49999	銀行 1
支店番号 5 ～ 7 の口座番号 50000 ～ 79999	銀行 2
支店番号 8 ～ 10 の口座番号 80000 ～ 109999	銀行 3

データの符号化と復号化

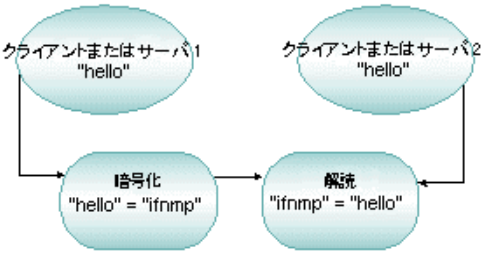
符号化と復号化を行うと、データ表現 (バイトの順序や文字セットなど) が異なるメッセージをマシン間で転送できるようになります。BEA Tuxedo システムでは、マシンに依存しない表現にデータを符号化および復号化して、マシン間でのデータの転送を可能にしています。デフォルトでは、XDR アルゴリズムが採用されています。BEA Tuxedo システムの関数をユーザ定義の関数で置き換えると、このアルゴリズムをカスタマイズできます。符号化および復号化は、マシン間でのみ使用されます。また、リモート・マシンのデータ表現がローカル・マシンのものと異なる場合にのみ使用されます。符号化と復号化によって、異なるデータ構造を持つマシンが異機種の BEA Tuxedo システムで動作できるようになります。プログラマは、それぞれの環境に適した表現でデータを管理できます。

BEA Tuxedo システムでは、バッファ・タイプを使用して、メッセージに含まれるフィールドのタイプが判別されたり、コードディングに必要なマッピングが行われます。このマッピングは、X_OCTET や CARRAY などの構造化されていないバッファ・タイプによって行われるものではありません。そのため、異機種が混在している環境でも、開発者は X_OCTET および CARRAY 型バッファを自由に運用できます。

データの暗号化

暗号化とは、メッセージをコード化された形式に変換して、ユーザが解読できないようにすることです。暗号化されたメッセージは、送信先に到着すると解読されます。つまり、元の形式に変換されます。

図 2-17 データの暗号化



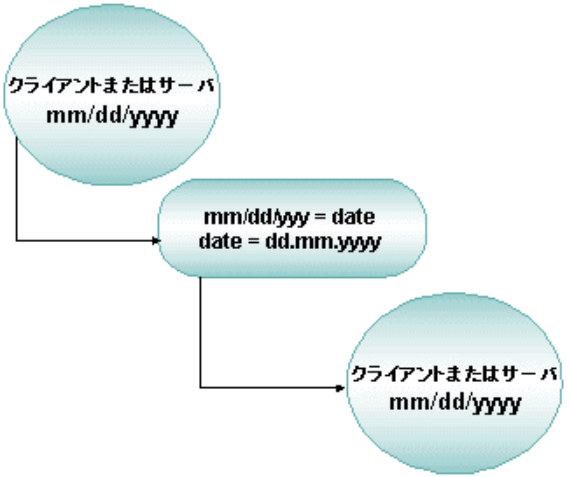
暗号化によってデータ内のビット数が増えることはありませんが、メッセージ送信での処理時間は増えます。ただし、暗号化でデータが圧縮されるため、ネットワーク上に送信されるデータ量が減少して、処理時間の増加が相殺される場合もあります。また、データが圧縮されるときに多少のスクランブリングが行われるので、セキュリティも若干強化されます。

データ・マーシャリング

データ・マーシャリングとは、BEA Tuxedo システムによって提供される言語ベースの TxRPC (X/Open-TxRPC) を使用して情報を扱う方法です。TxRPC はリモート・プロシージャ・コール用のプロトコルで、グローバル・トランザクションがサポートされます。TxRPC 呼び出しは、ローカル・プロシージャ・コールのように見えます。しかし、C 言語の関数が呼び出されると、関数に渡される引数がパッケージ化されてサーバに送られ、そこで呼び出された関数の処理が行われます。このような引数のパッケージ化をマーシャリングと呼びます。関数の引数は、ネットワークやプラットフォームの境界を越えることができるようにマーシャリング、つまりパッケージ化されます。そして、呼び出されたリモート・プロシージャに渡される前に、送信先でマーシャリングが解除されて、プロシージャで使用できる状態になります。

このプロセスは、クライアント (呼び出し元のプログラム) およびサーバ (リモート・プロシージャ) に透過的です。マーシャリングとマーシャリング解除のルーチンは、BEA Tuxedo のインターフェイス定義言語 (IDL) コンパイラによって自動的に生成されます。IDL コンパイラは RPC の記述を受け取り、クライアント・プログラムとサーバ・プログラムに対してスタブと呼ばれるルーチンを生成します。これらのスタブには、クライアントとサーバがマーシャリングされたデータを交換するための通信ロジック、およびマーシャリングとマーシャリング解除のロジックが含まれています。

図 2-18 データ・マーシャリング



ロード・バランシング

ロード・バランシングとは、同じサービスを提供するサーバ間でサービス要求を均等に分散する BEA Tuxedo システムの機能です。この機能を使用すると、負荷の重いサーバがある一方で、アイドル状態または使用頻度の低いサーバが生じる状態を防ぐことができます。要求をサービス・ルーチンに送信する前に、その要求を処理できるすべてのサーバがシステムで識別され、コンフィギュレーション内のすべてのサーバで負荷を均衡化するために最も適したサーバが選択されます。

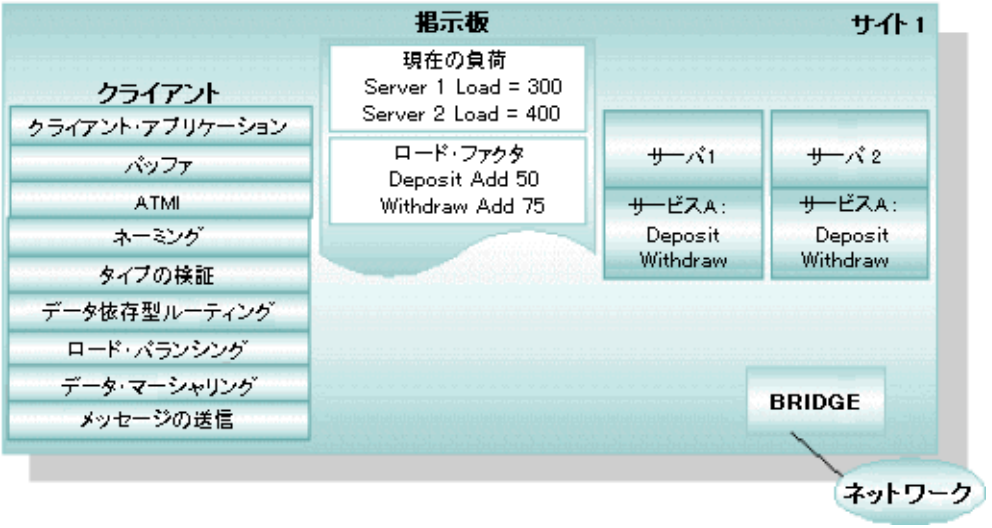
ロード・ファクタの割り当て

ロードとは、サービスの実行に必要な時間に基づいて、サービス要求に割り当てられた数値のことです。ロードはサービスに割り当てられるので、BEA Tuxedo システムで要求間の関係を識別できるようになります。コンフィギュレーション内の各サーバによって実行されている処理量、つまり負荷の合計をトラッキングするために、管理者はすべてのサービスとサービス要求にロード・ファクタを割り当てます。ロード・ファクタとは、あるサービスまたは要求を実行するために必要な時間を示す数値です。これらの数値に基づいて、各サーバに対して統計が生成され、各マシンの掲示板で維持されます。各掲示板では、各サーバの負荷が累積されます。そのため、すべてのサーバがビジー状態になった場合に、最も負荷の低いサーバを選択できます。

システム全体に対して、ロード・バランシング・アルゴリズムを使用するかどうかを制御できます。たとえば、必要な場合のみ、つまり複数のキューを使用する複数のサーバによってサービスが提供される場合のみ、このアルゴリズムを使用するように設定できます。1 つのサーバだけで提供されるサービス、または MSSQ (複数サーバ、単一キュー) にある複数のサーバによって提供されるサービスは、ロード・バランシングを必要としません。これらのサービスの LDBAL パラメータは N に設定します。それ以外の場合は LDBAL に Y を設定します。

ロード・ファクタの割り当て (UBBCONFIG の SERVICES セクション) を決定する場合は、アプリケーションを長時間実行し、各サービスの実行に要する平均時間を記録します。平均値に近い時間を要するサービスには、LOAD 値に 50 (LOAD=50) を指定します。平均値より長い時間がかかるサービスには LOAD>50、短い時間のサービスには LOAD<50 を指定します。

図 2-19 ロード・バランシング

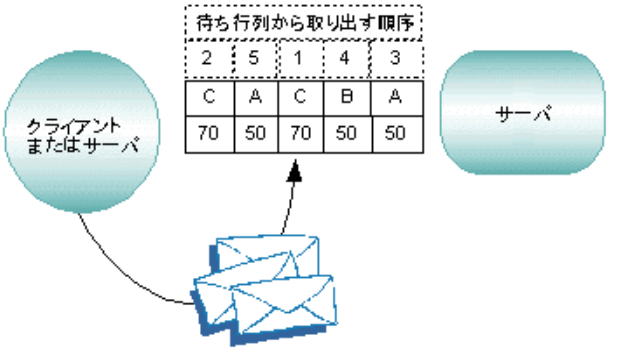


メッセージの優先順位付け

優先順位によって、サーバがキューからサービス要求を取り出す順序が決定されます。優先順位は、クライアントによって個々のサービスに割り当てられます。1 ～ 100 までの値が使用され、100 が最も高い優先順位です。

すべてのサービスには、優先順位の初期値として 50 が割り当てられます。サーバの優先順位の初期値は、アプリケーションのコンフィギュレーション時に変更できます。サービスを定義した後、これらのサービスに適切な優先順位を割り当てることができます。たとえば、ビジネス上の優先順位が比較的高いサービスに優先順位 70 を設定すると、このサービスはこれよりも低い優先順位 50 のサービスよりも先にキューから取り出されます。次の例では、サーバが A (優先順位 50)、B (優先順位 50)、および C (優先順位 70) というサービスを提供しています。

図 2-20　メッセージの優先順位付け



C の優先順位が高いため、サービス C に対する要求は、常に A または B に対する要求よりも先にキューから取り出されます。A および B に対する要求は、同じ優先順位になります。この機能は、アプリケーションで要求の緊急度や重要度が異なる場合に使用すると有用です。

「枯渇防止策」は、優先順位の低いメッセージがキューでいつまでも待ち続けるのを防ぐためのメカニズムです。この方法では、優先順位に関係なく、10 個単位のメッセージの 10 番目にあるものが FIFO (先入れ先出し) の順序でキューから取り出されます。先頭から 9 番目までのメッセージは、優先順位に従って取り出されます。

ネーミング

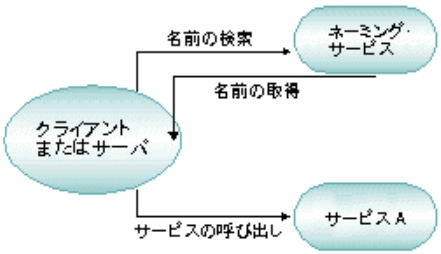
BEA Tuxedo システムでは、サービス名、メッセージ・キュー名、イベント名の 3 つのネーミング・デバイスが使用されています。名前には、任意の単語または英数字の文字列を使用できます。ただし、ピリオド「.」で始まる文字列は使用できません。管理サーバでも BEA Tuxedo システムの同じインフラストラクチャが使用されるため、システム・リソースとアプリケーション・リソースは明確に区別する必要があります。

ネーミング・サービス

サービスに名前を付けると、あるアプリケーション・コンポーネントが別のコンポーネントをその名前から見つけられるようになります。名前には、単純な単語 (「deposit」など) や英数字の文字列 (「deposit2」など」) を使用できます。名前は、アプリケーションの規模、およびアプリケーション・コンポーネント間の全体的な関係のマッピイングに基づいて選択します。これらのマッピングまたはサービスは、アプリケーション・コンポーネントを記載した電話帳のページに似ています。

BEA Tuxedo システムのサーバをアクティブにすると、掲示板 (MIB の動的な部分) によって、そのサーバのサービス名が宣言されます。サービス名はサーバの物理アドレスと対応付けられているので、要求をサーバにルーティングできます。プログラマがアプリケーションで使用する名前は、完全に位置透過的です。クライアント・プログラムがサービスを名前で要求すると、BEA Tuxedo システムは掲示板でその名前のレジストリを調べます。名前のレジストリには、文字列の名前 (`TICKET` など) をマシン名に変換するために必要な情報と、そのサービスを宣言しているサーバの物理アドレスが定義されています。BEA Tuxedo システムでは、これらを使用して要求を該当するサーバに送信します。

図 2-21 名前によるサービスの検索



サービスの宣言

BEA Tuxedo システムでは、2 種類の管理サーバを使って、掲示板上の情報をアプリケーション内のすべてのアクティブなマシンに配布します。

- DBBL —特別掲示板連絡 (DBBL) サーバは、MIB に対するグローバルな変更を複製転送し、MIB の静的な部分を保ちます。DBBL は、アプリケーションにかかわる各種のマシンの状態を調整します。DBBL は、アプリケーション全体に対して 1 つだけ存在します。フォールトトレランスを高めるために、ほかのマシンに移行することができます。

- BBL—掲示板連絡 (BBL) サーバは、掲示板を管理します。BBL はアプリケーションのアクティブなマシンごとに存在します。BBL は、ローカルの MIB への変更を調整し、自分と同じマシン上にあるアクティブなアプリケーション・プログラムが正常であるかどうかを検証します。

イベントのネーミング

BEA Tuxedo システムでは、パブリッシュ・アンド・サブスクライブのメカニズムが提供されています。クライアントおよびサーバは、特定のイベントが発生したときに警告 (またはメッセージ) を受信するための要求を動的に登録したり削除できます。ほかのクライアントおよびサーバは、アプリケーションで発生したユーザ定義のイベントまたはシステム・イベントをポストします。クライアントまたはサーバが特定のイベントに関する通知が必要なくなった場合、該当するサブスクリプションを取り消すことができます。

関連項目

- 第 2 章の 11 ページ「イベント・ブローカ」

BEA Tuxedo ATMI の管理サービス

システム・サーバによって、BEA Tuxedo ATMI 環境に必要な以下の管理サービスが提供されます。

- アプリケーション・キュー管理
- 中央集中型のアプリケーション・コンフィギュレーション
- 分散アプリケーション管理
- アプリケーションの動的な再コンフィギュレーション
- イベント管理
- セキュリティ管理
- アプリケーションの起動とシャットダウン
- トランザクション管理
- ワークステーション管理

注記 管理サービスについては、第 3 章の 1 ページ「BEA Tuxedo ATMI インフラストラクチャに対する 3 つの視点」を参照してください。

3 BEA Tuxedo ATMI インフラストラクチャに対する 3 つの視点

ここでは、次の内容について説明します。

- BEA Tuxedo ATMI の基本インフラストラクチャ
- 管理に対する視点 : 管理ツール
- BEA Tuxedo ATMI の管理サービス
- 開発の視点 :ATMI の使用
- ランタイム・システムの視点 : 各種コンフィギュレーションにおけるツールの使用

BEA Tuxedo ATMI の基本インフラストラクチャ

BEA Tuxedo ATMI 環境は、アプリケーション・サービス要求の効率的なルーティング、ディスパッチ、および管理、イベントのポストと通知、そしてアプリケーション・キューのためのインフラストラクチャを提供します。このインフラストラクチャは、次の 3 つの視点から分析することができます。

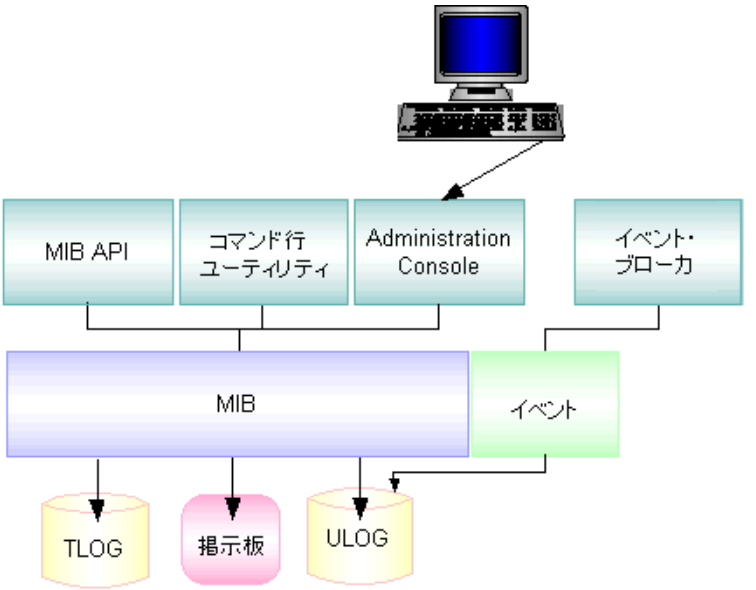
- 管理に対する視点—アプリケーションを管理するための各種ツール。
- 開発に対する (ATMI を使用した) 視点—ATMI を使用して実行できるタスク。クライアントは、ATMI を通してサービスを要求します。サーバ・プログラムはサービスをグループ化し、それらのサービスは ATMI によって定義された規則に従って呼び出されます。アプリケーション設計者は、BEA Tuxedo ランタイム・システムをアプリケーション・コードにリンクすることによって、クライアント・プログラムおよびサーバ・プログラムを構築します。
- BEA Tuxedo ランタイム・システムに対する視点—シングル・ドメイン、分散ドメイン、およびマルチ・ドメインの各コンフィギュレーション。

管理に対する視点：管理ツール

BEA Tuxedo の MIB には、アプリケーションの操作に必要なすべての情報が格納されています。MIB はプログラム可能な設計になっており、カスタムの管理プログラムを作成できます。管理ツールは MIB の周囲に構築され、MIB に対するさまざまなインターフェイスを提供します。このようなツールには、次のものがあります。

- BEA Administration Console—アプリケーションの監視や動的なコンフィギュレーションを行うための Web ベースのツール。
- BEA Tuxedo 管理サーバ—分散アプリケーションの管理タスクの大部分を自動化するサーバ。このようなタスクには、サービスおよびイベントのネーミング、アプリケーションの起動とシャットダウン、アプリケーションの動的な再コンフィギュレーションなどがあります。
- 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—MIB の情報にアクセスしたり、変更を行うための関数。
- コマンド行ユーティリティ—アプリケーションの起動 (`tmboot`—)、終了 (`tmshutdown`)、コンフィギュレーション (`tmconfig`)、管理 (`tmadmin`) に使用するコマンド。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。
- イベント・ブローカ—管理者に障害や例外の発生を通知するメカニズム。

図 3-1　アプリケーションを管理するためのツール



使用可能な BEA Tuxedo システムの MIB

管理情報ベースは、すべてのアプリケーションに共通するコア MIB と、いくつかのコンポーネント MIB (オプション) から構成されています。コア MIB は TM_MIB と呼ばれ、すべての BEA Tuxedo アプリケーションに必要なアプリケーションの構成要素が定義されています。また、これらのアプリケーションの構成要素の管理に使用することもできます。TM_MIB では、BEA Tuxedo システムのアプリケーションがクラス (サーバ、グループ、マシン、ドメインなど) として定義されています。各クラスは、各種の属性 (ID、状態など) によって特徴付けられるオブジェクトから構成されます。

各コンポーネント MIB では、BEA Tuxedo システムのサブシステムが定義されています。現在、次のコンポーネントを使用できます。

- ACL_MIB—アクセス制御リストの管理に使用されます。
- APPQ_MIB—アプリケーションの安定記憶域内のキューを管理するために使用されます。
- DM_MIB—Tuxedo ドメインの管理に使用されます。
- EVENT_MIB—イベント通知とサブスクリプション要求データベースの制御に使用されます。

- WS_MIB―ワークステーション グループとそれらに対応付けられたプロセスを管理するために使用されます。

BEA Administration Console

Java および Web 技術に基づいて、BEA Administration Console では実質的にどこからでも、セキュリティ認可を与えられていれば自宅からでも、BEA Tuxedo ドメインを操作できます。BEA Administration Console は Java ベースのアプレットであり、Internet ブラウザにダウンロードして BEA Tuxedo システムのアプリケーションをリモートで管理することができます。

BEA Administration Console は、複数層型システムの管理に必要な多くのタスクを簡略化します。システム・イベントの監視、システム・リソースの管理、管理オブジェクトの作成と設定、およびシステムの統計情報の表示ができます。

ブラウザの条件

BEA Tuxedo システムの各リリースでは、リリースの時点で利用可能なブラウザがサポートされています。BEA Administration Console で現在サポートされているブラウザについては、次の BEA 社 Web サイトを参照してください。

関連項目

- 第 3 章の 5 ページ「BEA Administration Console の利点」
- 第 3 章の 6 ページ「BEA Administration Console のメイン・メニュー」
- 『BEA Administration Console Online Help』
- 『BEA Tuxedo アプリケーション実行時の管理』の 第 2 章の 1 ページ「アプリケーションの監視方法」

BEA Administration Console の利点

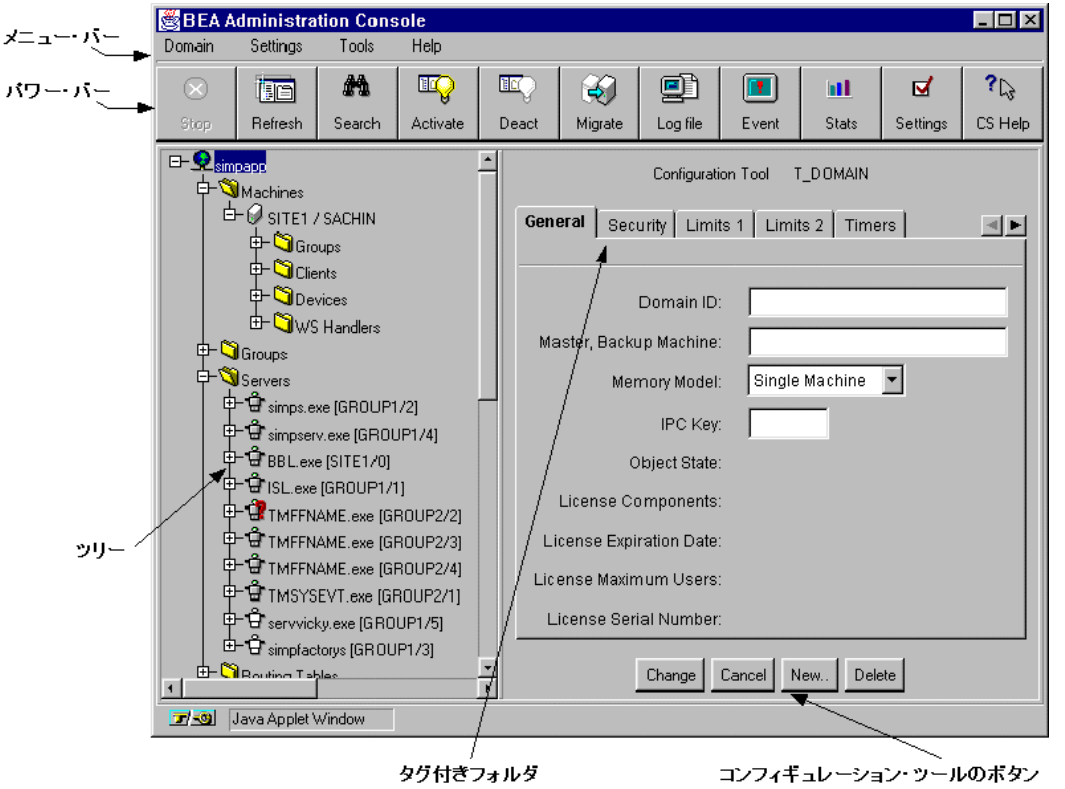
- 認証—BEA Administration Console では、ユーザの認証が必ず行われます。管理者は、ユーザ名とパスワードを入力するように求められます。入力した情報は暗号化形式でブラウザとサーバ間でやり取りされ、サーバでユーザの認証が行われます。サーバ・セットアップの大部分はインストール時、つまり BEA Administration Console のサーバ・コンポーネントがインストールされ、Web サーバからアクセスできるようになったときに設定されます。
- コンテキスト・センシティブ・ヘルプ—すべての BEA Administration Console 画面およびツールで、コンテキスト・センシティブ・ヘルプを利用できます。画面上の任意のフィールドや領域に疑問符のアイコンをドラッグしてクリックすると、そのフィールドや領域に関する情報が表示されます。
- 暗号化—サーバ側とブラウザ間で転送されるデータは、誰も読むことができないように圧縮 (56 ビットまたは 128 ビットの暗号化) されます。この機能により、ストリームの中に不正な管理プロトコル・メッセージが挿入されることを防ぐことができます。
- ファイアウォールへの対応—BEA Administration Console サーバがブラウザからの要求を受け取り、データのやり取りを行うポートは、明確に定義され、設定可能なポートです。つまり、ファイアウォールを通して許可されるポートとして、このポートを設定できます。この機能により、必要に応じて、ファイアウォールを通して Console ベースの管理を行うことができます。
- アイコン—BEA Administration Console で使用されるアイコンは、状態 (非アクティブなど)、またはアプリケーション内の特定のオブジェクト (マシンやサーバなど) を示します。
- Java 対応ブラウザ—Java ブラウザでは、アプレットを実行して通信を可能にする Java 仮想マシンがサポートされています。
- クライアント側のインストールの不要—クライアントのマシンへのインストールは必要ありません。ブラウザで、ドメイン内の Console サーバ・コンポーネントが存在するマシンの URL を指定します。続いて、Java アプレットのダウンロードを開始します。アプレットによって BEA Administration Console がインプリメントされ、サーバとの通信が確立されます。
- 世界中からの安全なアクセス—Java 対応のブラウザがあれば、セキュリティ・メカニズムが確立されたものとして、世界中のどこからでもシステムにアクセスできます。

BEA Administration Console のメイン・メニュー

初めて Web にアクセスし、BEA Administration Console を起動すると、メイン・ウィンドウが表示されます。メイン・ウィンドウは、主に次の 4 つの要素から構成されます。

- メニュー・バー―すべてのアクションにアクセスできるメニュー。
- ツールバー―よく使用されるアクションまたは管理ツールへのショートカットであるボタン。
- ツリー―BEA Tuxedo のドメイン内に存在する管理クラス・オブジェクト (サーバやクライアントなど) の階層構造。
- コンフィギュレーション・ツール―タブ付きフォルダ。オブジェクトの属性 (マシンの名前など) の表示、定義、変更を行うことができます。

図 3-2　Administration Console **のメイン・メニュー**



注記　ドメインに接続されていない場合、パワー・バーのボタンとメニュー項目の一部が表示されません。

ツリー

[Tree View] ペインはメイン GUI ウィンドウの左側に表示されます。ツリーは、1 つの BEA Tuxedo システム・ドメインを構成する管理オブジェクトの階層構造です。オブジェクトの入れ子のレベルと親オブジェクトを示すことによって、オブジェクト間の関係がグラフィカルに表されます。ツリー全体 (ドメイン内の設定可能なあらゆる種類のすべてのオブジェクト) を表示するか、またはオブジェクトのサブセットを表示することができます。

ドメインを設定してアクティブにすると、ドメイン内の管理クラス・オブジェクトを表すラベル付きのアイコンがツリーに表示されます。

管理オブジェクト

BEA Administration Console ツリーには、管理オブジェクトそれぞれに対して 1 つずつ、複数のルートが含まれます。最初のルートは、アプリケーション・ドメインから構成されます。次のルートには、BEA Tuxedo TMIB で定義されるオブジェクト・クラスが表示されます。オブジェクト・クラスの各セットはアプリケーション・ドメインの一部です。3 番目のレベルは、オブジェクト・クラスに属すオブジェクトのインスタンスを表します。

たとえば、ドメイン内に、SITE1 に存在する romeo と juliet という 2 つのマシンが含まれているとします。両方のマシンがオブジェクトなので、それらが属しているオブジェクト・クラスの名前 `Machines` の下に表示されます。つまり、次のように表示されます。

```
Machines
        SITE1/romeo
        SITE1/juliet
```

ツリー内の各オブジェクトの名前の前にはアイコンが付いています。たとえば、各マシンはコンピュータのアイコンで表され、各クライアントは人の形で表されます。

コンフィギュレーション・ツール

コンフィギュレーション・ツールは、選択したクラスの BEA Tuxedo システム・オブジェクトの属性を設定したり変更するためのユーティリティです。ツリーのオブジェクトを選択すると、そのオブジェクトの [Configuration Tool] ペインがメイン・ウィンドウの右側に表示されます。

コンフィギュレーション・ツール領域のタブ付きフォルダは、管理オブジェクトの属性に関する情報を表示したり入力するための電子フォームです。オブジェクトの管理クラス (マシンやサーバなど) ごとに、フォルダがあります。クラスに対応付けられている属性の数は、クラスによって大きく異なります。そのため、オブジェクトを選択してコンフィギュレーション・ツールを開くと、どこからでも 1 ～ 8 個のフォルダが表示されます。

コンフィギュレーション・ツール領域にデータが入力されると、タブ付きページの下に 4 つのボタンが表示されます。これらのボタンを使用すると、フォルダで行うコンフィギュレーション作業を制御できます。

ツールバー

ツールバーには、頻繁に行う管理操作のためのツールを呼び出す 12 個のボタンが並んでいます。ボタンには、アイコンと名前が付いています。次の表は、各ボタンについて示しています。

ボタン	処理内容
Stop	現在の操作を中断し、管理者に制御を戻します。この後で、管理者は新しい操作を要求できます。
Refresh	ツリーとコンフィギュレーション・ツールの各ウィンドウを最新の内容で更新します。
Search	ツリーで、特定の管理オブジェクト・クラスまたはオブジェクトを検索します。
Activate	BEA Tuxedo ドメインの一部または全部をアクティブにします。
Deactivate	BEA Tuxedo ドメインの一部または全部を非アクティブにします。
Migrate	サーバ・グループまたはサーバ・マシンを別の場所に移行します。または、マスタ・マシンとバックアップ・マシンを入れ替えます。
Log file	アクティブ・ドメイン内の特定のマシンから ULOG ファイルを表示します。
Event	システム生成イベントを監視するウィンドウを表示します。
Stats	BEA Tuxedo ドメインのアクティビティをグラフィカルに表示できるタブ付きページを表示します。
Settings	Administration Console セッションの次のデフォルト設定を行えます。 ■ BEA Tuxedo オンライン・マニュアルの場所 ■ データの順序付けの方法 (状態または名前) ■ デフォルトの作業モード (表示のみまたは編集モード)
CS Help	コンテキスト・センシティブ・ヘルプを表示します。フィールドまたは特定の画面領域をクリックすると、選択した項目についての情報が表示されます。
Help	BEA Administration Console のオンライン・ヘルプを別の Web ブラウザで開きます。

MIB を使用した操作の管理

AdminAPI は、BEA Tuxedo 管理情報ベース (MIB) の中のシステム設定に直接アクセスし、変更するためのアプリケーション・プログラミング・インターフェイス (API) です。AdminAPI を使用すると、ログ・ファイルの監視やアプリケーションの動的な再コンフィギュレーションなどの管理タスクを自動化し、人が行う作業を減らすことができます。このような利点は、ミッション・クリティカルなリアルタイム・アプリケーションで重要です。MIB プログラミング・インターフェイスを使用すると、BEA Tuxedo システムにおける操作を簡単に管理することができます。特に、独自のプログラムを使用してアプリケーションの監視、コンフィギュレーション、チューニングを行うことができます。MIB は、次のように定義されます。

- FML 属性として定義され、インプリメンテーションに依存しない管理データベース。
- ATMI 関数を使用して、BEA Tuxedo システムへの照会 (get を使用してシステムから情報を取得すること) や、BEA Tuxedo システムの更新 (set を使用してシステムの情報を変更すること) をいつでも行うことができるプログラミング・インターフェイス。これらの関数には、tpalloc、tprealloc、tpgetrply、tpcall、tpacall、tpenqueue、tpdequeue などがあります。

関連項目

- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の MIB(5)
- 第 3 章の 11 ページ「MIB ユーザのタイプ」
- 第 3 章の 11 ページ「MIB のクラス、属性、状態」

MIB ユーザのタイプ

MIB では、システム管理者、システム・オペレータ、その他という 3 つのタイプのユーザが定義されています。次の表は、各タイプについて示しています。

ユーザのタイプ	説明
システム管理者	アプリケーションの正常な実行を維持する責任者。管理者には、すべての管理ツールと MIB のすべての管理機能を使用する権限が与えられます。管理者は、実行中のアプリケーションのコンフィギュレーション、管理、変更を行うことができます。
システム・オペレータ	実行中のアプリケーションの日々の操作を監視し、問題に対応する担当者。オペレータは、実行中のアプリケーションに関する統計を監視します。イベントおよび警告に対応して、サーバの起動やマシンのシャットダウンなどを行う場合もあります。オペレータは、アプリケーションの再コンフィギュレーション、サーバまたはマシンの追加、マシンの削除は行いません。
その他	MIB を読み取る必要はあるが、アプリケーションを変更する権限を持たない人またはプロセス (カスタム・プログラムなど)。

MIB のクラス、属性、状態

クラスとは、BEA Tuxedo アプリケーションを構成するエンティティ (サーバ、マシンなど) のタイプです。属性とは、クラス内のオブジェクトの特徴を表すもので、ID、状態、コンフィギュレーション・パラメータ、実行時の統計などがあります。属性には、MIB の操作および応答に共通するものや、個々のクラスに共通するものがあります。すべてのクラスには、オブジェクトの状態を示す状態属性があります。MIB 上でオブジェクトの状態を変更する操作を呼び出している場合、オブジェクトの状態はユーザ変更の状態、または新しく変更された状態のいずれかになります。

MIB(5) リファレンス・ページに定義されている共通の属性は、クラスに依存しません。これらの属性は、入力操作を制御したり、ユーザが何をしようとしているかを MIB に伝達したり、特定のクラスに依存しない出力バッファの特徴をプログラマに示します。

コマンド行ユーティリティ

BEA Tuxedo システムには、システムの各構成要素を管理するためのコマンドが提供されています。これらのコマンドを使用すると、共通の管理ユーティリティにアクセスできます。管理ユーティリティでは、次のタスクを行うことができます。

- コマンド行ユーティリティを使用したアプリケーションのコンフィギュレーション
- コマンド行ユーティリティを使用したアプリケーションの操作
- コマンド行ユーティリティを使用したアプリケーションの監視

コマンド行ユーティリティを使用したアプリケーションのコンフィギュレーション

vi テキスト・エディタなどのコマンド行ユーティリティを使用して、アプリケーションをコンフィギュレーションできます。特に、コマンド行ユーティリティを使用すると、コンフィギュレーション・ファイル UBBCONFIG を記述し、tmloadcf コマンドを実行してこのファイルをテキスト形式 (UBBCONFIG) からバイナリ形式 (TUXCONFIG) に変換できます。この後、アプリケーションを起動します。

サーバやマシンを追加したり、マシンを削除して、コンフィギュレーションを動的に管理することもできます。ただし、TUXCONFIG (バイナリ版) を更新しても、UBBCONFIG (テキスト版) は更新されません。この 2 つのファイルを同期させるには、これらをバックアップする必要があります。その場合、tmunloadcf コマンドを実行して、バイナリ・ファイルをテキスト・ファイルに変換します。

注記 UBBCONFIG は、アプリケーション管理者によって、アプリケーション・ディレクトリ (APPDIR) に生成されて格納されます。

以下は、アプリケーションのコンフィギュレーションに使用できるコマンド行ユーティリティです。

- tmconfig —BEA Tuxedo アプリケーションの実行中に、コンフィギュレーション・ファイルのパラメータまたは MIB 属性を更新したり、TUXCONFIG セクションにレコードを追加するためのコマンド。
- tmloadcf—バイナリ形式の TUXCONFIG コンフィギュレーション・ファイルをロードするためのコマンド。
- tmunloadcf—UBBCONFIG と TUXCONFIG を同期させるために、バイナリ形式のコンフィギュレーション・ファイルをテキスト形式に変換するコマンド。
- tpacladd、tpaclcvt、tpacldel、tpaclmod—アプリケーションのアクセス制御リストを作成して管理するためのコマンド。これらのコマンドによって、セキュリティ関連の認証機能を使用できるようになります。
- tpgrpadd、tpgrpdel、tpgrpmod—アクセス制御リストを使用して、サービス、キュー、イベントへのアクセスを認めることにより、ユーザ・グループを作成して管理するためのコマンド。
- tpusradd、tpusrdel、tpusrmod—認証のためのユーザ・データベースを作成および管理するコマンド。

関連項目

- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の UBBCONFIG(5)
- 第 3 章の 20 ページ「コンフィギュレーション・ファイルの作成」
- 第 3 章の 22 ページ「コンフィギュレーションの永続的な変更」

コマンド行ユーティリティを使用したアプリケーションの操作

アプリケーションを一度コンフィギュレーションすると、次のコマンド行ユーティリティを使用してアプリケーションを操作できるようになります。

- tmadmin—分散アプリケーションのコンフィギュレーション、監視、チューニングを行うためのコマンド。
- tmboot—分散アプリケーション用のアプリケーション・サーバを中央で起動するためのコマンド。

- tmshutdown—分散アプリケーションのアプリケーション・プログラムを中央でシャットダウンするためのコマンド。

イベント・ブローカを使用したシステム・イベントの管理

BEA Tuxedo イベント・ブローカは、次のタスクを実行します。

- イベントを監視し、tppost(3c) を通じてポストされたイベントをサブスクライバに通知します。
- イベントをトラッキングし、管理者にアプリケーション内の変化を継続的に通知します。
- システム全体にわたるイベントのサマリを提供して、イベントの監視を強化します。
- イベントを使用して、さまざまな通知処理を開始するメカニズムを提供します。

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

- マシン・クラス・オブジェクトの名前 (T_MACHINE) と、そのマシンのすべての属性
- イベントをマシンの分断として識別するいくつかのイベント属性

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

関連項目

- 第 3 章の 15 ページ「イベント」
- 第 3 章の 15 ページ「イベントのサブスクライブ」
- 第 3 章の 16 ページ「イベントのタイプ」

- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 14 ページ「イベント・ベースの通信」

イベント

イベントとは、実行中のアプリケーションにおける状態の変化などで、オペレータ、管理者、またはソフトウェアによる操作を必要とするものです。イベント・ブローカでは、イベントに次の 3 つの重要度レベルのいずれかが割り当てられます。

- エラー―サーバが停止したり、ネットワーク接続が切断された場合など。
- 情報―コンフィギュレーションの変更が検出されたり、プロセスの結果として、状態の変化が発生した場合など。
- 警告―認証に失敗したために、クライアントがアプリケーションへの参加を許可されない場合など。

イベントのサブスクライブ

BEA Tuxedo アプリケーションの管理者は、クライアントまたはサーバ・プロセスの代わりに EVENT_MIB(5) を呼び出してサブスクリプションを要求できます。イベント・ブローカを使用してイベントをサブスクライブするには、tpsubscribe を使用します。イベント A、B、C をサブスクライブして、これらのイベントが発生したときに通知されるようにします。

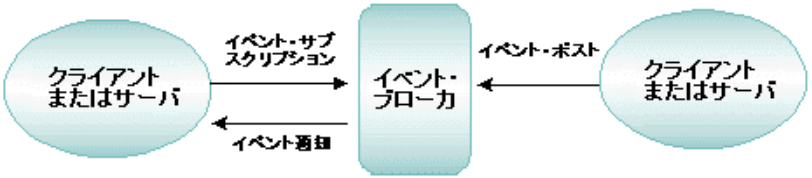
各サブスクリプションには、次のいずれかの通知方法が指定されています。

- クライアント通知―イベント・ブローカによってこれらのイベントに対するクライアントの関心がトラッキングされます。クライアントには任意通知型通知として通知されます。匿名でポストされるイベントもあります。クライアントは、そのクライアント以外がサブスクライブしている場合でも、アプリケーションに参加してイベント・ブローカにイベントをポストできます。イベント・ブローカは、これらのイベントをサブスクリプションのデータベースと照合し、該当するクライアントに任意通知型通知を送信します。
- サービス呼び出し―サブスクライバがイベント通知をサービス呼び出しに送る場合は、ctl パラメータが有効な TPEVCTL 構造体を指している必要があります。

- 安定記憶域のキューへのメッセージの登録―安定記憶域のキューへのサブスクリプションでは、eventexpr と filter のほかに、キュー・スペース、キュー名、および相関識別子を使用して合致しているかどうかが判定されます。相関識別子を利用することにより、同じ送信先の、同じイベント式やフィルタ・ルールに対する複数のサブスクリプションを区別することができます。
- ULOG へのメッセージの挿入―サブスクライバは、EVENT_MIB の T_EVENT_USERLOG― クラスを使用して、システムの USERLOG メッセージを記述できます。イベントが検出されて合致すると、これらのイベントは USERLOG に書き込まれます。
- コマンド行ユーティリティー―イベント・ブローカは、EVENT_MIB の T_EVENT_COMMAND クラスを使用して、イベントのトラッキングと照合を行います。合致するイベントが見つかると、そのイベントのサブスクライブに使用されるコマンドに渡します。

  注記 通知方法は、サブスクライバのプロセスのタイプと tpsubscribe に渡された引数によって決まります。

図 3-3 イベントのサブスクライブ



関連項目

- 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の EVENT_MIB(5)
- 『BEA Tuxedo C リファレンス』の tpsubscribe(3c)

イベントのタイプ

BEA Tuxedo システムでは、次の 2 つのイベント・タイプがサポートされています。

- システム・イベント―サーバ停止およびネットワーク障害など、BEA Tuxedo システム・イベントについての詳細を提供します。クライアントまたはサーバによってイベントがポストされると、イベント・ブローカはポストされたイベントの名前をそのイベントのサブスクライバのリストと照合し、各サブスクリプションに指定された処理を行います。
- ユーザ・イベントまたはアプリケーション固有のイベント―特定の基準が満たされた場合に、アプリケーション・プログラムがイベントをポストできるようにします。たとえば、銀行業務アプリケーションが一定額を超える引き出しに対してポストするイベントなどです。

システム・イベントとアプリケーション固有のイベントの相違点

次の表は、システム・イベントとアプリケーション固有のイベントの相違点を示しています。

表 3-1 システム・イベントとアプリケーション固有のイベントの相違点

分野	相違点
イベント	システム・イベントは、BEA Tuxedo システムのコードによって事前に定義されています。アプリケーションでは、設計者が監視すべきアプリケーション・イベントを決定します。アプリケーション・プログラムは、a) 対象となるイベントの発生時にそのイベントを検出し、b) 検出されたイベントが `tppost` を介してイベント・ブローカにポストされるように記述します。
イベント・リスト	システム・イベントのリストが EVENTS(5) によってユーザに提供されるように、アプリケーション・イベント・サブスクリプションのリストが関心のあるユーザに提供されます。システム・イベント名はドット (.) で開始されます。ただし、アプリケーション固有のイベント名をドット (.) で開始することはできません。

表 3-1 システム・イベントとアプリケーション固有のイベントの相違点

分野	相違点
サブスクリプション	アプリケーション固有のイベント・ブローカでイベントをサブスクライブすることは、BEA Tuxedo システムのイベント・ブローカをサブスクライブすることと似ています。イベントをサブスクライブするには、そのアプリケーションの公開イベント・リストを使用して `tpsubscribe` を呼び出します。EVENTS(5) は、イベントによって生成された通知メッセージとイベント名 (`tppost` が呼び出されたときに引数として使用される名前) をリストします。サブスクライバは、正規表現のワイルド・カード機能を使用して、`tpsubscribe` を 1 回呼び出すだけでイベントのカテゴリ全体に対応することができます。

BEA Tuxedo ATMI の管理サービス

システム・サーバによって、BEA Tuxedo システムに必要な以下の管理サービスが提供されます。

- アプリケーション・キュー管理
- 中央集中型のアプリケーション・コンフィギュレーション
- 分散アプリケーション管理
- アプリケーションの動的な再コンフィギュレーション
- イベント管理
- セキュリティ管理
- アプリケーションの起動とシャットダウン
- トランザクション管理
- ワークステーション管理

アプリケーション・キュー管理

キュー処理を使用すると、1 つ以上のキューにアクセスして通信を行うアプリケーションをプログラミングできます。キューは位置透過的であり、プログラミングに関する変更を行わずに、管理者はあるマシンから別のマシンにキューを移動できます。

MIB は、キュー・デバイス、キュー・スペース、(アプリケーションによって必要とされる) キュー、およびキュー・スペースへのメッセージの登録と取り出しを行う BEA Tuxedo システムのサーバから構成されます。管理者は、BEA Administration Console またはコマンド行ユーティリティを使用して、MIB にキュー・スペース、キュー、および管理サーバを定義することができます。

qmadmin を使用したアプリケーション・キューの管理

コマンド行ユーティリティ qmadmin を使用すると、コンフィギュレーション内のアプリケーション・キューに関するあらゆる管理機能を実行できます。つまり、キューを格納する汎用デバイス・リスト (UDL) およびボリューム一覧 (VTOC) の設定、キュー・デバイスでのキュー・スペースの定義などを行うことができるようになります。また、qmadmin ではファイル・システムの操作が可能です。いくつかのランタイム監視機能を使用して、キュー内のメッセージ数、メッセージ中のヘッダ数を確認できます。また、キューまたはキュー内のメッセージの属性を変更したり、キュー内のメッセージを削除したり、デバイスのサイズを変更することもできます。1 つのアプリケーションで、複数のアプリケーション・キュー・デバイスを設定し、複数のマシン上でアプリケーション・キューを実行できます。各マシンにはそれぞれ専用のキュー・デバイスがあり、qmadmin を実行して、各マシン上の特定のアプリケーション・キュー・デバイスを監視したり管理することができます。

ユーティリティ	機能説明
qmadmin	メッセージ・キューの作成、検査、および変更を行います。キュー・スペースのための汎用デバイス・リストが存在する (または存在することになる) デバイス (ファイル) 名は、コマンド行引数として指定することも、環境変数 QMCONFIG を介して指定することもできます。この両方で指定された場合には、コマンド・オプションが使用されます。

tmconfig を使用したコンフィギュレーションの変更

tmconfig コマンドを使用すると、アプリケーションの実行中に、TUXCONFIG ファイルとそれに対応付けられたエンティティの参照や変更を行ったり、新しいコンポーネント (マシンやサーバなど) を追加することができます。

コンフィギュレーション・ファイル (MASTER マシン上の TUXCONFIG) を変更すると、tmconfig コマンドは次の処理を行います。

- アプリケーションで現在起動しているすべてのマシンの TUXCONFIG ファイルを更新します。
- 新しく起動するマシンに、TUXCONFIG ファイルを自動的に複製転送します。
- BEA Tuxedo システムのクライアントとして起動します。

注記 コンフィギュレーション・パラメータの指定方法、指定可能な値の範囲、および検証については、『BEA Tuxedo コマンド・リファレンス』、および 『ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス』の tmconfig、wtmconfig(1)、TM_MIB(5) を参照してください。

コンフィギュレーションの管理

アプリケーションのコンフィギュレーションは、主にコンフィギュレーション・ファイル、つまり UBBCONFIG ファイルの作成と維持によって制御されます。コンフィギュレーションの管理には、次のタスクがあります。

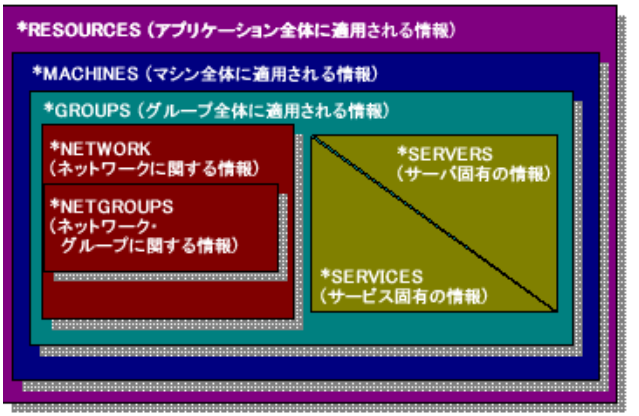
- アプリケーションのニーズに適合したコンフィギュレーション・ファイルの作成
- UBBCONFIG ファイルの更新によるコンフィギュレーションへの永続的な変更
- アプリケーション実行中のコンフィギュレーションの変更

コンフィギュレーション・ファイルの作成

アプリケーションのコンフィギュレーション・データは、UBBCONFIG で保持されます。これは、MASTER マシン上にある通常のテキスト・ファイルです。コンフィギュレーション・ファイル (UBBCONFIG) は、アプリケーションのリソース、マシン、グループ、サーバ、利用可能なサービスのリストなど、アプリケーションを起動するために必要なあらゆる情報が含まれているリポジトリです。UBBCONFIG ファイルは、作成後 TUXCONFIG バイナリ・ファイルにコンパイルされます。マルチドメインのアプリケーションを開発している場合は、アプリケーション内のドメインごとに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。アプリケーションは、コンフィギュレーション・ファイルがないと実行できません。

UBBCONFIG ファイルは、8 つのセクションから構成されています。そのうち、RESOURCES、MACHINES、GROUPS、SERVERS、SERVICES の 5 つは、すべてのコンフィギュレーションに必須です。RESOURCES と MACHINES は、それぞれ最初と 2 番目のセクションであることが必要です (次の図を参照)。GROUPS は、SERVERS および SERVICES よりも前に置く必要があります。

図 3-4　UBBCONFIG ファイル



- RESOURCES(必須)―システム全体に関するパラメータが記述され、アプリケーション全体のリソースが定義されています。
- MACHINES(必須)―物理マシンの論理名とタイプが記述されています。
- GROUPS(必須)―サーバをリソース・マネージャおよびマシンと対応付けします。
- SERVERS―アプリケーション内の各サーバを識別します。
- SERVICES―各サービスを識別し、優先順位やロードなどを指定します。
- NETWORK―LAN 環境に関するコンフィギュレーション・データが記述されています。
- ROUTING―データ依存型ルーティング・テーブルが記述されています。
- NETGROUPS―1 つのマシンに対して複数の BRIDGE を指定できるようにします。

UBBCONFIG ファイルに必要なセクションは、各コンフィギュレーションによって異なります。作成した UBBCONFIG ファイルは、TUXCONFIG と呼ばれるバイナリ・ファイルにコンパイルする必要があります。TUXCONFIG ファイルを生成するには、tmloadcf(1) コマンドを実行するか、または BEA Administration Console を使用します。

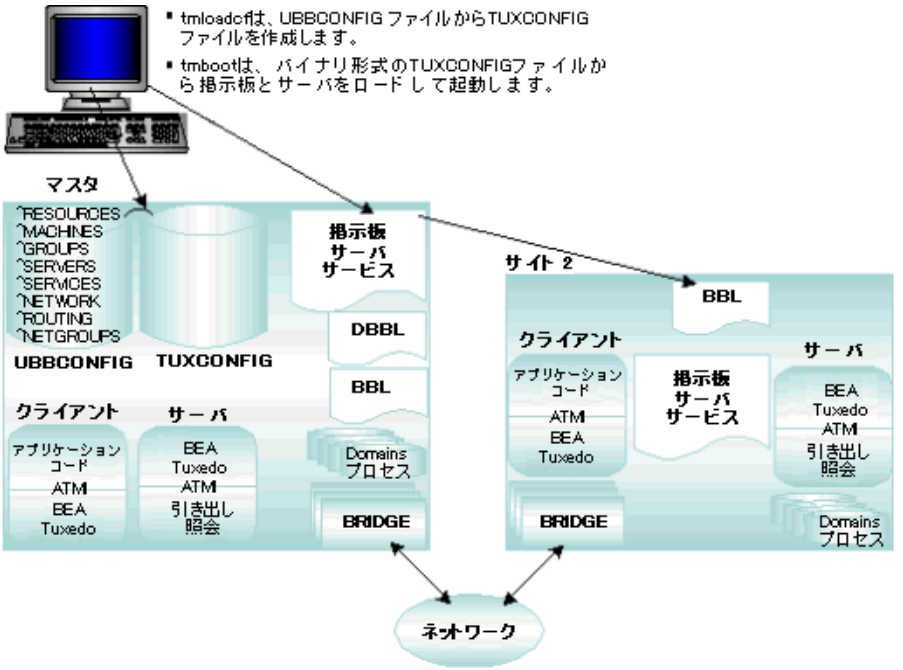
関連項目

- 『BEA Tuxedo アプリケーションの設定』の 第 3 章の 2 ページ「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の永続的な変更

コンフィギュレーションに永続的な変更を行うには、テキスト・エディタで UBBCONFIG ファイルのコンフィギュレーション・パラメータを更新した後、tmloadcf ユーティリティを使用して、このテキスト・ファイルを BEA Tuxedo システムで使用されるバイナリ形式の TUXCONFIG ファイルにロードします。アプリケーションが起動すると、tmboot によって TUXCONFIG が共用メモリにロードされて掲示板が構築されます。変更内容は、必要に応じてリモート・マシンに複製転送されます。

図 3-5　コンフィギュレーションの管理



コンフィギュレーションの動的な管理

管理者は、BEA Administration Console または BEA Tuxedo システムのコマンド行ユーティリティを使用して、システムの実行中に変化するシステム・ロードに応じてパラメータを調整して、アプリケーションの動的な再コンフィギュレーションを行うことができます。変更された TUXCONFIG ファイルは、システムのすべてのマシンに自動的に複製転送されます。ただし、RESOURCES パラメータの多くは、システムの実行中には変更できません。

動的に実行できるタスクには、サーバまたはマシンの追加やマシンの削除などがあります。コンフィギュレーション・ファイルのテキスト形式 (UBBCONFIG) とバイナリ形式 TUXCONFIG) を常に一致させるには、tmunloadcf を使用してこの 2 つのファイルをバックアップし、同期させる必要があります。tmunloadcf は、バイナリ・ファイルをテキスト形式に変換するコマンドです。

システムの大部分の要素は、動的に変更できます。たとえば、新しいサーバを作成したり、新しいマシンを追加したり、タイム・アウトのパラメータを変更することができます。ただし、システムの実行中には変更できないものもいくつかあります。

- コンフィギュレーション・ファイルのパラメータのうち、掲示板のサイズと形式に影響するものは変更できません。このようなパラメータの多くには、「MAX」という接頭辞が付いています。たとえば、BEA Tuxedo システムで同時に処理できるトランザクションの最大数を指定する MAXGTT パラメータなどがあります。
- 特定のアプリケーション内でマシンのプロセッサ名を変更することはできません。異なる名前で新しいマシンを追加することはできますが、既存のマシンの名前を変更することはできません。
- MASTER マシンおよび BACKUP マシン上で実行するために割り当てられたサーバの実行可能プログラムの値は変更できません。

関連項目

- 第 3 章の 24 ページ「tmadmin(1) を使用した動的操作の実行」

tmadmin(1) を使用した動的操作の実行

tmadmin(1) コマンドを使用すると、実行中のアプリケーションに次の操作を行うことができます。

- パフォーマンスの監視 (bbstats、bbparms)。これは、グループ、サーバ、サービスに関する統計値を調べて行います。
- ロード値の変更 (changeload)、サービスの中断と再開 (suspend および resume)、サービスの宣言と宣言の取消し (advertise および unadvertise)、AUTOTRAN のタイム・アウト値の変更 (changetrantime) などを実行するサーバ・パラメータおよびサービス・パラメータの変更。
- 起動 (boot)、クリーンアップ (pclean)、および移行 (migratemach、migrategroup)。

よく使用される tmadmin コマンド

tmadmin には、ランタイム・システムの監視、アプリケーションのチューニング、アプリケーションの動的なコンフィギュレーションなどを行うサブコマンドがあります。次は、最もよく使われる tmadmin のサブコマンドです。tmadmin コマンドの全リストについては、『BEA Tuxedo コマンド・リファレンス』の tmadmin(1) を参照してください。

- help—サブコマンドのリスト、その省略形、引数、および説明を出力します。
- printserver (psr)—アプリケーションおよび管理サーバに関する情報を出力します。
- printservice (psc)—アプリケーションおよび管理サービスに関する情報を出力します。
- printclient (pclt)—指定されたクライアント・プロセスに関する情報を出力します。引数が指定されてなく、デフォルトも定義されていない場合は、すべてのクライアントに関する情報が出力されます。

tmadmin コマンドの出力例

以下は、tmadmin printserver (psr) コマンドの出力の例です。アプリケーションおよび管理サーバに関する情報が示されます。

図 3-6 tmadmin コマンドの出力例

```
> psr

Prog Name      Queue Name  Grp Name      ID RqDone Load Done Current Service
---------      ----------  --------      -- ------ --------- ---------------
BBL             83108       SITE1         1     50         0 ( IDLE )
AUDITC         auditc      BANKB1         1      0         0 (  IDLE )
XFER           00001.00101 BANKB1       101      1        30 (  TRANSFER )
TMS_SQL        BANKB1_TMS  BANKB1     30001      0         0 (  IDLE )
ACCT           00001.00102 BANKB1       102      0         0 (  IDLE )
TMS_SQL        BANKB1_TMS  BANKB1     30002      0         0 (  IDLE )
BAL            00001.00103 BANKB1       103      6         7 (  IDLE )
BTADD          00001.00104 BANKB1       104      0         0 (  IDLE )
BALC           00001.00105 BANKB1       105      0         0 (  IDLE )
TLR            tlr1        BANKB1       111      0         0 (  IDLE )
TLR            tlr1        BANKB1       112      3       110 (  WITHDRAWAL )
TLR            tlr1        BANKB1       113      0         0 (  IDLE )
TLR            tlr1        BANKB1       114      0         0 (  IDLE )
TLR            tlr1        BANKB1       115      0         0 (  IDLE )
TLR            tlr1        BANKB1       116      9       100 (  IDLE )
TLR            tlr1        BANKB1       117     20      2048 (  IDLE )
TLR            tlr1        BANKB1       118     30       600 (  IDLE )
TLR            tlr1        BANKB1       119      0         0 (  IDLE )
TLR            tlr1        BANKB1       120      0         0 (  IDLE )
>
```

関連項目

- 『BEA Tuxedo アプリケーション実行時の管理』の 第 2 章の 12 ページ「tmadmin セッションのしくみ」
- 『BEA Tuxedo アプリケーション実行時の管理』の 第 2 章の 9 ページ「コマンド行ユーティリティを使用してアプリケーションを監視する」

分散アプリケーションの集中管理

大型で複雑な BEA Tuxedo アプリケーションでも、すべてのランタイム管理機能を 1 つの MASTER マシンから実行できます。その場合、BEA Tuxedo システムによって提供されるコマンド行ユーティリティ、BEA Administration Console、または BEA Manager 製品と併用するサード・パーティ製の管理ツールを使用します。

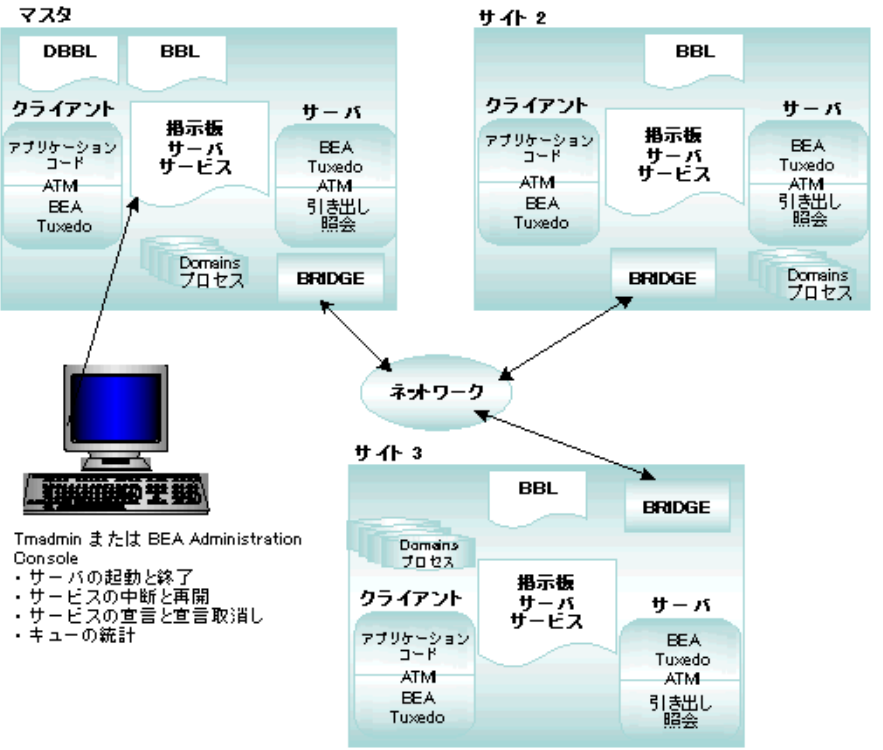
MASTER マシンから、アプリケーションのコンフィギュレーション、起動およびシャットダウン、ランタイムの管理タスクなどを実行できます。ほかのすべてのマシンは、MASTER マシンを照会できます。コンフィギュレーション、障害管理、セキュリティ、監視、パフォーマンスについても、MASTER マシンから制御できます。

実行中のシステムを変更するには、次の 2 つの方法を使用します。

- BEA Administration Console—管理タスク (システムの動的な変更など) 用のコマンドを実行するためのグラフィカル・ユーザ・インターフェイス (GUI)。
- tmadmin コマンド—50 個のサブコマンドを実行できるシェル・レベルのメタ・コマンド。システムの動的な変更など各種の管理タスクを実行します。

BEA Administration Console はグラフィカル・ユーザ・インターフェイス (GUI) を備えているため、tmadmin コマンド・インタプリタより簡単に使用できます。GUI を使用する場合は、管理タスクを開始する準備ができたら、画面上で BEA Administration Console を起動します。BEA Administration Console のグラフィックスとオンライン・ヘルプを参照すると、どの管理タスクでも実行できるようになります。次の図は、tmadmin コマンドまたは BEA Administration Console を使用して、ランタイム・アプリケーションを制御する方法を示しています。すべての操作は MASTER マシンから実行できます。ユーティリティによる操作は、MASTER マシンの掲示板に直接作用します。更新内容は、ほかの掲示板に自動的に伝達されます。

図 3-7　分散アプリケーションの集中制御



関連項目

- 第 3 章の 4 ページ「BEA Administration Console」
- 第 3 章の 24 ページ「tmadmin(1) を使用した動的操作の実行」

セキュリティの管理

アプリケーションには、BEA Tuxedo システムによって提供される適切なレベルのセキュリティを設定することができます。認証および権限のレベルを設定して、アプリケーションへのアクセスを制限できます。セキュリティには、さまざまなレベルがあります。安全性が高く、認証が行われない環境もあれば、パスワードやアクセス制御リスト (ACL) によって、サービスの使用、イベントのポスト、キューへのメッセージの登録や取り出しなどを実行できるユーザを限定する環境もあります。

ACL を使用すると、アプリケーションへの参加時にユーザの認証が行われるだけではなく、アプリケーションのエンティティ (サービスなど) へのアクセスが試みられた場合にパーミッションが確認されます。あるリソースに対して ACL が作成されると、そのリストに含まれていないユーザはそのリソースへのアクセスを拒否されます。ACL によって保護されていないリソースは、アプリケーションへの参加に成功したすべてのクライアントがアクセスできます。ACL で MANDATORY_ACL セキュリティ・オプションが指定されていると、アプリケーションに参加したすべてのクライアントに対してアクセスが拒否されます。

すべてのサーバ (BEA Tuxedo 管理サーバの AUTHSVR を除く) が、共用メモリやメッセージ・キューなどの共用リソースに対して制限付きのアクセス権を持つように、アプリケーションを設定できます。クライアントがアプリケーションに参加する場合、AUTHSVR によって認証サービスが提供され、そのユーザの MIB における認証レベルが確認されます。このサービスは、プログラマに透過的に行われます。

関連項目

- 第 3 章の 29 ページ「セキュリティ・オプションの選択」
- 第 3 章の 30 ページ「セキュリティの設定」
- 『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』の第 2 章の 1 ページ「セキュリティの管理」
- 『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』の第 3 章の 1 ページ「セキュリティのプログラミング」

セキュリティ・オプションの選択

以下は、BEA Tuxedo システムによって提供されるセキュリティ・オプションです。

- 認証なし—クライアントは、アプリケーションに参加する前に確認されません。
- アプリケーション・パスワード—アプリケーション全体に対して 1 つのスワードが定義されます。クライアントがアプリケーションに参加する場合は、このパスワードを指定する必要があります。
- ユーザ・レベル認証—アプリケーションに参加するには、アプリケーション・パスワードのほかに、各クライアントが有効なユーザ名とアプリケーション固有のデータ (パスワードなど) を指定する必要があります。
- 任意アクセス制御リスト (ACL)—クライアントがアプリケーション・パスワード、ユーザ名、およびユーザ・パスワードを指定する必要があります。ユーザ名に対応付けられた ACL がない場合、パーミッションが与えられます。このオプションによって、管理者はより高いセキュリティを必要とするリソースに対してだけアクセスを設定することができます。誰でもアクセスできるサービス、キュー、イベントには、ACL を設定する必要はありません。
- 必須アクセス制御リスト (ACL)—クライアントがアプリケーション・パスワード、ユーザ名、およびユーザ・パスワードを指定する必要があります。このレベルは任意 ACL のオプションに似ていますが、ACL をユーザがアクセスできるすべてのエンティティ (サービス、キュー、イベントなど) に対して設定する必要があります。必須 ACL のオプションが使用されている場合に、ある特定のエンティティの ACL がないと、そのエンティティに対するパーミッションは与えられません。
- リンク・レベルの暗号化—BEA Tuxedo システム・セキュリティのユーザは、BEA Tuxedo アプリケーションのマシン間を接続するネットワーク・リンク上を移動するメッセージに対して、データをプライバシ扱いできます。データは、ネットワーク・リンクに送信される前に暗号化され、リンクから取り出されるときに解読されます。セキュリティのレベルには、0 ビット (暗号化なし)、56 ビット (国際版)、128 ビット (米国およびカナダ) の 3 種類があります。
- 公開鍵暗号方式—メッセージ・ベースの暗号化とメッセージ・ベースのデジタル署名から構成されます。メッセージ・ベースの暗号化は、指定された受信者だけにデータを公開します。メッセージ・ベースのデジタル署名を使用した場合、送信プロセスは自身の ID を証明し、その証明書を特定のメッセージ・バッファにバインドしなければなりません。署名の確認は、任意のサード・パーティが行います。デジタル署名には、バッファの内容全体に対して計算され、暗号化によって安全なチェックサムが含まれているので、検出されずに改ざんを行うことは不可能です。デジタル署名には、送信元マシンのローカル・クロックに基づいた、改ざんに対するスタンプも含まれています。
- 監査—操作の要求とその結果に関する情報の収集、保存、配布を行います。

セキュリティの設定

アプリケーションのセキュリティを設定するために必要な管理作業やプログラミングは、選択したセキュリティ・オプションによって異なります。管理に関しては、BEA Administration Console またはコマンド行ユーティリティのいずれかを使用して、MIB のコンフィギュレーションを行う必要があります。

また、独自のセキュリティ・メカニズムを構築することもできます。その場合、アプリケーションのセキュリティ・レベルをユーザ・レベルの認証に設定し、BEA Tuxedo MIB で認証を実行するアプリケーション・サービスを指定します。

認証と権限付与を行うには、MIB で次のコンフィギュレーションを行う必要があります。

- AUTHSVR サーバ
- 権限を持つユーザの ID とパスワード
- サービス、キュー、イベントに対して使用されるアクセス制御リスト

アプリケーションの起動とシャットダウン

アプリケーションを起動するには、『BEA Tuxedo アプリケーション実行時の管理』に説明されているように、次の作業を行います。

1. 第 1 章の 2 ページ「環境変数の設定」の説明に従って、環境変数を設定します。
2. 第 1 章の 4 ページ「TUXCONFIG ファイルの作成」の説明に従って、TUXCONFIG ファイルを作成します。
3. 第 1 章の 5 ページ「アプリケーション固有のディレクトリとファイルを手動で複製転送する」の説明に従って、BEA Tuxedo ソフトウェアを複製転送します。
4. 必要に応じて、第 1 章の 6 ページ「TLOG デバイスの作成」の説明に従って、TLOG デバイスを作成します。
5. 第 1 章の 7 ページ「全サイトでの tlisten の起動」の説明に従って、すべてのサイト (MP 環境) で tlisten を起動します。
6. 第 1 章の 9 ページ「アプリケーションの起動」の説明に従って、アプリケーションを起動します。

アプリケーションをシャットダウンするには、次の作業を行います。

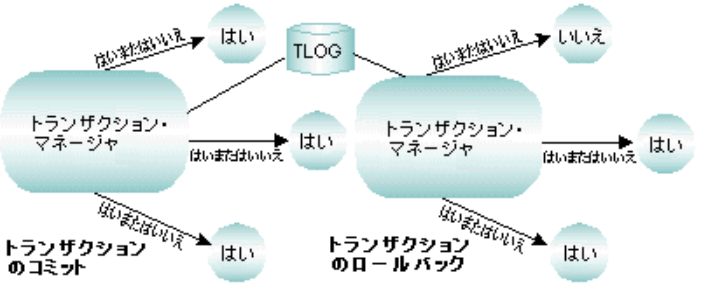
第 1 章の 12 ページ「アプリケーションのシャットダウン」の説明に従って、MASTER マシン上で tmshutdown を実行します。

トランザクションの管理

BEA Tuxedo システムの強力な機能の 1 つに、XA インターフェイスをサポートするデータベース・アプリケーションのトランザクションを管理する機能があります。トランザクションを使用すると、分散アプリケーションを簡単に記述できるようになります。トランザクションにより、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、分散環境における問題への対応が一層容易になります。

分散アーキテクチャでは、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・マシンと通信できます。このリモート・マシンが、さらに別のリモート・マシンと通信する場合もあります。リモート・マシンによって行われる通信および作業は、トランザクションの一部であり、整合性を維持する必要があります。分散トランザクション処理 (DTP) のトラッキングは、複雑な作業になる場合があります。これは、任意の時点でトランザクションをロールバックする (取り消す) のに十分な情報を保持する必要があるからです。

図 3-8　トランザクション管理



BEA Tuxedo システムでは、トランザクションのパーティシパントをトラッキングするために、トランザクション・ログが作成されます。アプリケーションの状態をコンピュータのメモリの内容どおりに保つために、1 つ以上のリソース・マネージャが使用されます。リソース・マネージャとは、情報およびその情報にアクセスするためのプロセスの集まりで、たとえばデータベース管理システムなどがあります。トランザクションによって実行されるすべての操作、および影響を受けるすべてのモジュールを調整するために、BEA Tuxedo システムではトランザクション・マネージャ (TM) が使用されます。トランザクション・マネージャは、リソース・マネージャのアクションを指示します。トランザクション・マネージャとリソース・マネージャの共同作業により、分散トランザクションの原子性が維持されます。

トランザクション・マネージャ・サーバ (TMS) による操作の調整

BEA Tuxedo トランザクション・マネージャ (TM) は、システム全体のリソースに関連するグローバル・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) 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。このサーバでは、RM への埋め込み SQL インターフェイスを使用して、サーバ・グループにアクセスされるデータベースの読み取りや更新を行います。TMS および RM は、XA インターフェイスを使用して、グローバル・トランザクションにおけるすべてのリソース操作を実行するか、またはまったく行いません。

トランザクション・ログ (TLOG) によるパーティシパントのトラッキング

グローバル・トランザクションは、コミット・プロセスにある場合のみ、トランザクション・ログ (TLOG) に記録されます。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 TLOG に記録されます。TLOG の記録は、コミットすべきグローバル・トランザクションを示します。ロールバックすべきトランザクションは、TLOG には記録されません。第 1 フェーズ、つまりプリコミットでは、各リソース・マネージャがトランザクション要求をコミットします。すべてのパーティシパントがコミットすると、そのトランザクションは、トランザクション管理によってコミットされて完了します。アプリケーションまたはシステムの障害によってこのいずれかのタスクが失敗すると、両方のタスクが失敗し、実行された処理が取り消されます。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。

注記 Domains 機能を使用している場合は、Domains ゲートウェイが Domains グループにおける TMS の機能を果たすことに注意してください。ただし、Domains では、Domains 固有の情報のほかに、TLOG と同じような情報が記録された独自のトランザクション・ログが使用されます。

関連項目

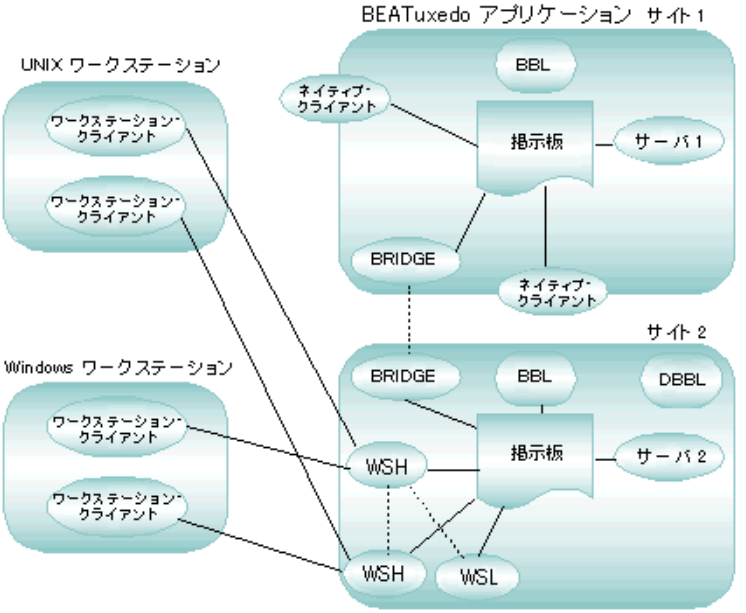
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 17 ページ「トランザクション」
- 『BEA Tuxedo アプリケーションの設定』の 第 5 章の 1 ページ「トランザクション対応の ATMI アプリケーションのコンフィギュレーション」

ワークステーションの管理

ワークステーション・クライアントには、要求に関する情報をパッケージ化するために十分な BEA Tuxedo システム・ソフトウェアが必要です。このようなソフトウェアがあると、ATMI 関数やネットワーク・ソフトウェアを含め、すべての BEA Tuxedo システム・ソフトウェアがサポートされたシステムに、その情報を送信することができます。

管理者は、1 つ以上のワークステーション・リスナ (WSL) を設定して、ワークステーション・クライアントからの接続要求を受け取ることができるようにします。各 WSL では、1 つ以上のワークステーション・ハンドラ (WSH) を使用して、クライアントの要求が処理されます。各 WSH では、複数のワークステーションが管理され、特定のワークステーションとのすべての通信が単一の接続に多重化されます。

図 3-9 ワークステーション・クライアントの処理



この図に示されているように、1 つのマシンで何千ものワークステーション・クライアントを処理できます。管理者は、ドメイン内にいくつかの WSL を定義すると、複数のマシン間でワークステーションの通信負荷を均一的に分散できます。プログラミングの観点では、ワークステーション・クライアントの開発では、すべてのクライアント ATMI プログラミング・インターフェイスがサポートされます。

開発の視点 :ATMI の使用

BEA Tuxedo の API であるアプリケーション・トランザクション・モニタ・インターフェイス (ATMI) は、通信、トランザクション、およびデータ・バッファ管理のためのインターフェイスで、BEA Tuxedo システムによってサポートされるすべての環境で動作します。この ATMI によって、アプリケーション・プログラムと BEA Tuxedo システムとのインターフェイスが定義されます。ATMI は、広範な機能に対応する 1 つの単純なインタフェースです。ATMI には、トランザクション処理の X/Open DTP モデルがインプリメントされています。

図 3-10　ATMI の使用

ATMI では次のタスクがサポートされています。
- クライアントの初期化
- サーバ・ネーミング
- システム・メッセージング
- トランザクション管理
- サービスのディスパッチ
- バッファ管理

ATMI ライブラリには、BEA Tuxedo アプリケーションでグローバル・トランザクションを定義し、制御するための各種の関数が含まれています。グローバル・トランザクションを使用すると、分散アプリケーションで、複数のプログラムおよびリソース・マネージャにわたる排他的な作業単位を管理できます。1 つのトランザクション内のすべての作業は、1 つの論理単位として処理されます。そのため、タスクを正常に完了できないプログラムが 1 つでもあると、そのトランザクションではどのプログラムでも何も処理は行われません。ほとんどの ATMI 関数では、異なる通信方法がサポートされています。これらの関数は、プログラム間でデータを交換できるようにして、分散プログラムを互いに結び付けます。すべての ATMI 関数は、型付きバッファでデータを送受信します。

実行されるタスクごとに分類した ATMI 関数 (C および COBOL のバインディング対応) については、第 2 章の 1 ページ「BEA Tuxedo ATMI のアーキテクチャ」の 表 2-1「ATMI 関数」を参照してください。

注記 ATMI トランザクション管理関数を使用することは必須ではありません。

関連項目

- 『BEA Tuxedo アプリケーション実行時の管理』の 第 2 章の 27 ページ「ATMI を使用してシステム・エラーとアプリケーション・エラーを処理する」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 2 ページ「BEA Tuxedo ATMI クライアントの作成」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 第 1 章の 4 ページ「BEA Tuxedo ATMI サーバの作成」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の第 1 章の 6 ページ「アプリケーションの型付きバッファ」
- 第 2 章の 9 ページ「BEA Tuxedo のメッセージング・パラダイム」
- 第 2 章の 41 ページ「ネーミング」

ランタイム・システムの視点 : 各種コンフィギュレーションにおけるツールの使用

BEA Tuxedo システムには、アプリケーション内のプロセスおよびプロセス間に発生する通信の作成、監視、管理を行うためのツールが提供されています。さまざまなコンフィギュレーションで、プロセスとメッセージングの基本パラダイムを使用できます。各コンフィギュレーションは、次のランタイム・カテゴリのいずれかに分類されます。

- シングル・マシン・アプリケーション—1 つ以上のローカルまたはリモートのクライアントが、同じマシン上にある 1 つ以上のサーバと通信します。
- 複数のマシンにまたがる分散アプリケーション—1 つ以上のローカルまたはリモートのクライアントが、同じドメイン内のいくつかのマシン上にある 1 つ以上のサーバと通信します。
- マルチ・ドメイン・アプリケーション—複数のドメインが互いに通信します。

ランタイム・システムの機能

次の表は、シングル・マシン・アプリケーション、分散アプリケーション、およびマルチ・ドメイン・アプリケーションで使用できる BEA Tuxedo システムの機能を示しています。

表 3-2 各タイプのコンフィギュレーションで使用できる機能

使用できる機能	シングル・マシン・コンフィギュレーション	マルチ・マシン (分散) コンフィギュレーション	マルチ・ドメイン・コンフィギュレーション
ATMI	X	X	X
メッセージング・パラダイム	X	X	X
管理用の要素 掲示板 (BB)、BBL、TLOG、UBBCONFIG、ULOG、TUXCONFIG DBBL BRIDGE	X	X X X	X X X
Domains プロセス : DMADM、GWADM、GWTDOMAIN (TDomains 用)、dmloadcf、dmunloadcf および DMCONFIG、DMTLOG、BDMCONFIG			X
アプリケーション・プロセス : クライアント、サーバ、およびサービス	X	X	X
キュー処理	X	X	X
トランザクション管理	X	X	X
イベント管理	X	X	
セキュリティ管理	X	X	X

シングル・マシン・コンフィギュレーション

シングル・マシン・コンフィギュレーションは、単一のマシン上にある 1 つ以上のサーバと通信する 1 つ以上のローカルまたはリモートのクライアントから構成されます。このマシン上では、1 つ以上のビジネス・アプリケーションが実行されています。このタイプのコンフィギュレーションは、複数のアプリケーションを含む場合でも、単一のエンティティとして管理されるので、シングル・ドメインと見なされます。

このコンフィギュレーション内のすべてのアプリケーションのすべての管理対象要素 (サービス、サーバなど) は、1 つの BEA Tuxedo コンフィギュレーション・ファイルで定義されて制御されます。次の図は、単一のマシン上にインストールされて実行されているシングル・マシン・コンフィギュレーションの基本要素を示しています。

図 3-11　BEA Tuxedo のシングル・マシン・コンフィギュレーション

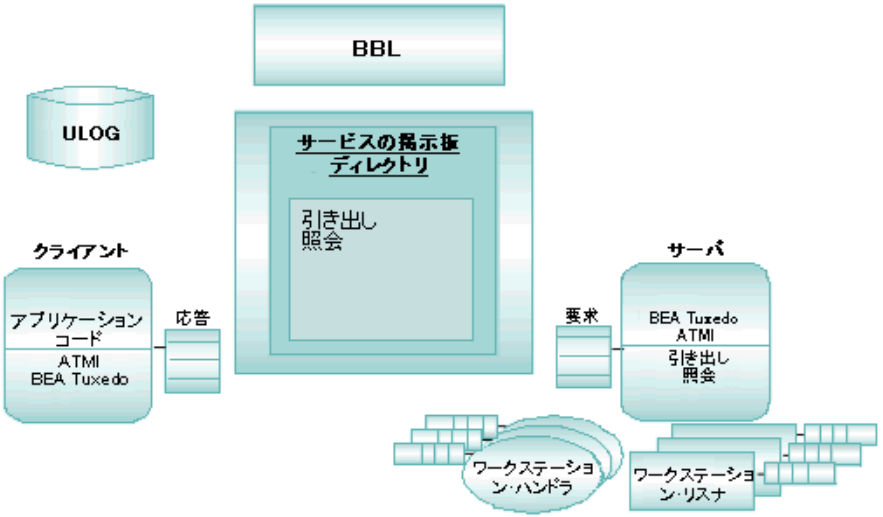


表 3-3　シングル・マシン・コンフィギュレーションの構成要素

シングル・マシンの構成要素	説明
掲示板 (BB)	システムのコンフィギュレーションと動的な情報を保持する共用メモリ・セグメント。すべての BEA Tuxedo プロセスから利用できます。

表 3-3 シングル・マシン・コンフィギュレーションの構成要素

シングル・マシンの構成要素	説明
BBL	掲示板に格納されているデータ (掲示板に対する変更を含む)、およびすべてのアプリケーション・プログラムを監視する BEA Tuxedo の管理プロセス。
クライアント	BEA Tuxedo システムを介して定期的にサービスを要求する実行可能プログラム。通常、クライアント・プログラムはユーザが作成します。
メッセージ・キュー	クライアントとサーバ間の通信は、オペレーティング・システムでサポートされ、メモリ・ベースであるメッセージ・キューを介して行われます。
メッセージング・パラダイム	クライアントとサーバ間のメッセージ転送に関する各種のモデル。たとえば、要求 / 応答モード、会話モード、イベント、任意通知型通信などがあります。
サーバ	BEA Tuxedo システムを介して指定されたサービスを提供する実行可能プログラム。通常、サーバ・プログラムはユーザが作成します。
ワークステーション・ハンドラ (WSH)	サーバ上のマルチ・コンテキストのゲートウェイ・プロセス。ワークステーション・クライアント (リモート・サイト上で実行されているクライアント・プロセス) からのサービス要求を管理します。
ワークステーション・リスナ (WSL)	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。ワークステーション・クライアント (リモート・サイト上で実行されているクライアント・プロセス) からの接続を受け付け、配信します。
ULOG (ユーザ・ログ)	エラー・メッセージが格納されるファイル。

関連項目

- 『BEA Tuxedo アプリケーションの設定』の 第 3 章の 2 ページ「コンフィギュレーション・ファイルの作成方法」

マルチ・マシン (分散) コンフィギュレーション

分散ドメイン (またはマルチ・マシン) のコンフィギュレーションは、複数のマシン上で実行される 1 つ以上のビジネス・アプリケーションから構成されます。このタイプのコンフィギュレーションは、複数のアプリケーションを含む場合でも、単一のエンティティとして管理されるので、シングル・ドメインと見なされます。つまり、このコンフィギュレーション内のすべてのマシンのすべてのアプリケーションのすべての管理対象要素 (サービス、サーバ、マシンなど) は、1 つの BEA Tuxedo コンフィギュレーション・ファイルで定義されて制御されます。

業務の拡大に伴い、個々の管理とサービスやデータの共有が可能な機能性をベースに、アプリケーション開発者は新しい部門を組織化する必要があります。機能性に基づいて定義されるアプリケーションは、複数のマシンにわたる場合もあり、またほかのアプリケーションとは独立して管理されます。このように機能的に独立したアプリケーションをドメインと呼びます。

ドメイン名は、提供される機能を表しています。ドメインに「マーケティング」や「研究開発」などの名前が付けられていると、必要なアプリケーションを簡単に見つけることができます。

次の図は、複数のマシンに分散したコンフィギュレーションの基本要素を示しています。

図 3-12 分散アプリケーション

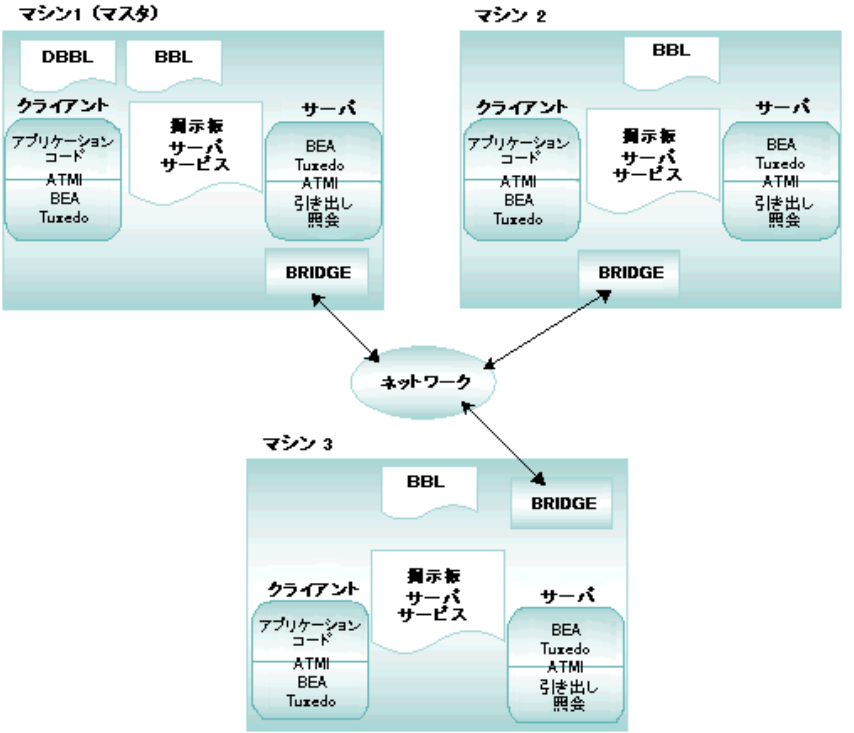


表 3-4 分散コンフィギュレーションを構成する基本要素

マルチ・マシンの構成要素	説明
BRIDGE	BEA Tuxedo システムによって提供されるドメイン内のサーバ。マシン間でサービス要求を送受信し、要求をローカル・サーバ (正確にはローカル・サーバのキュー) にルーティングします。
掲示板 (BB)	システムのコンフィギュレーションと動的な情報を保持する共用メモリ・セグメント。すべての BEA Tuxedo プロセスから利用できます。
BBL	掲示板に格納されているデータ (掲示板に対する変更を含む)、およびすべてのアプリケーション・プログラムを監視する BEA Tuxedo の管理プロセス。

表 3-4 分散コンフィギュレーションを構成する基本要素 (続き)

マルチ・マシンの構成要素	説明
クライアント	BEA Tuxedo システムを介して定期的にサービスを要求する実行可能プログラム。通常、クライアント・プログラムはユーザが作成します。
DBBL	各マシン上の BBL サーバが実行されていて、正常に機能していることを確認するための専用プロセス。このサーバは、ドメインの マスタ・マシン上で実行され、すべての管理機能と直接通信します。
メッセージ・キュー	クライアントとサーバ間の通信は、オペレーティング・システムでサポートされ、メモリ・ベースであるメッセージ・キューを介して行われます。
メッセージング・パラダイム	クライアントとサーバ間のメッセージ転送に関する各種のモデル。たとえば、要求 / 応答モード、会話モード、イベント、任意通知型通信などがあります。
サーバ	BEA Tuxedo システムを介して指定されたサービスを提供する実行可能プログラム。通常、サーバ・プログラムはユーザが作成します。
ワークステーション・ハンドラ (WSH)	サーバ上のマルチ・コンテキストのゲートウェイ・プロセス。ワークステーション・クライアント (リモート・サイト上で実行されているクライアント・プロセス) からのサービス要求を管理します。
ワークステーション・リスナ (WSL)	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。ワークステーション・クライアント (リモート・サイト上で実行されているクライアント・プロセス) からの接続を受け付け、配信します。
ULOG (ユーザ・ログ)	エラー・メッセージが格納されるファイル。

複数のマシン上で実行されるコンフィギュレーションでは、プラットフォームの相互運用性とサーバの透過性が必要です。

- プラットフォームの相互運用性とは、異なるマシン上で異なるオペレーティング・システムが実行されている場合でも、コードのカスタマイズを行わずに、アプリケーションがマシン間で通信できることを意味します。

- サーバの透過性とは、クライアントがサーバのロケーションを指定しなくてもサーバにアクセスできることを意味します。サーバのロケーションは掲示板に記録され、必要に応じてアクセスできます。そのため、アプリケーション自体を変更することなく、動的にサーバの移動、削除、追加を行うことができます。

DBBL サーバおよび BRIDGE サーバは、このような分散ドメイン・コンフィギュレーションの要件を満たしています。

関連項目

- 『BEA Tuxedo アプリケーションの設定』の 第 3 章の 3 ページ「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 アプリケーションの設定』の 第 7 章の 1 ページ「ネットワークへの ATMI アプリケーションの分散」
- 『BEA Tuxedo アプリケーションの設定』の 第 8 章の 1 ページ「分散型の ATMI アプリケーション用のコンフィギュレーション・ファイルの作成」
- 『BEA Tuxedo アプリケーションの設定』の 第 9 章の 1 ページ「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 第 4 章の 1 ページ「分散アプリケーションでのネットワーク管理」

マルチ・ドメイン・コンフィギュレーション

マルチ・ドメイン・コンフィギュレーションは、互いに通信する複数のドメインから構成されます。各ドメインは、シングル・マシン・コンフィギュレーションまたはマルチ・マシン・コンフィギュレーションのいずれかです。ドメイン間の通信は、すべてのドメインに送信されるサービス要求と、すべてのドメインから応答されるサービス要求を処理する高度な非同期マルチタスク・ゲートウェイによって実現されます。複数の BEA Tuxedo ドメインと接続することにより、あるドメインのクライアントがリモート・ドメインに物理的に存在するサービスに透過的にアクセスできます。各ドメインはサービスとデータを共有していますが、別々に管理されます。

BEA Tuxedo システムでは、さまざまなネットワーク転送プロトコルに対応した各種のゲートウェイが用意されています。以下に、各タイプの Domains ゲートウェイについて説明します。

- BEA Tuxedo Domains (TDomains) ゲートウェイは、複数の BEA Tuxedo アプリケーション間の相互運用を実現します。これは、TCP/IP などのネットワーク転送プロトコル上を流れる特別に設計された TP プロトコルを介して行われます。
- BEA eLink OSI TP ゲートウェイは、BEA Tuxedo アプリケーションと OSI TP 標準を使用するほかのトランザクション処理アプリケーションの相互運用を実現します。OSI TP は、国際標準化機構 (ISO) によって定義された分散トランザクション処理のためのプロトコルです。
- BEA eLink Adapter for Mainframe SNA ゲートウェイは、BEA Tuxedo ドメイン内のクライアントとサーバ、およびリモート SNA ドメイン内の MVS/CICS または MVS/IMS 環境のクライアントとサーバの相互運用を実現します。また、ローカルの BEA Tuxedo ドメインを複数の SNA ネットワークに接続します。
- BEA eLink Adapter for Mainframe TCP for CICS は、BEA Tuxedo 領域内のトランザクションに関与しないタスクと CICS アプリケーション・プログラム間で、一方が提供するサービスにもう一方がアクセスできるようにするゲートウェイ接続機能です。このプロトコルによって、BEA Tuxedo ドメインが TCP/IP ネットワーク転送プロトコルを介して CICS 環境と通信できるようになります。
- BEA eLink Adapter for Mainframe TCP for IMS は、IMS システム内のクライアント / サーバ・トランザクションと BEA Tuxedo ドメイン、CICS システム、またはほかの IMS システムとの間に、透過的な通信を実現するゲートウェイ接続機能です。
- TOP END Domain Gateway (TEDG) は、BEA TOP END システムと BEA Tuxedo ドメインの相互運用を実現します。

次の図は、マルチ・ドメイン・コンフィギュレーションの基本要素を示しています。

図 3-13　マルチ・ドメイン・コンフィギュレーション

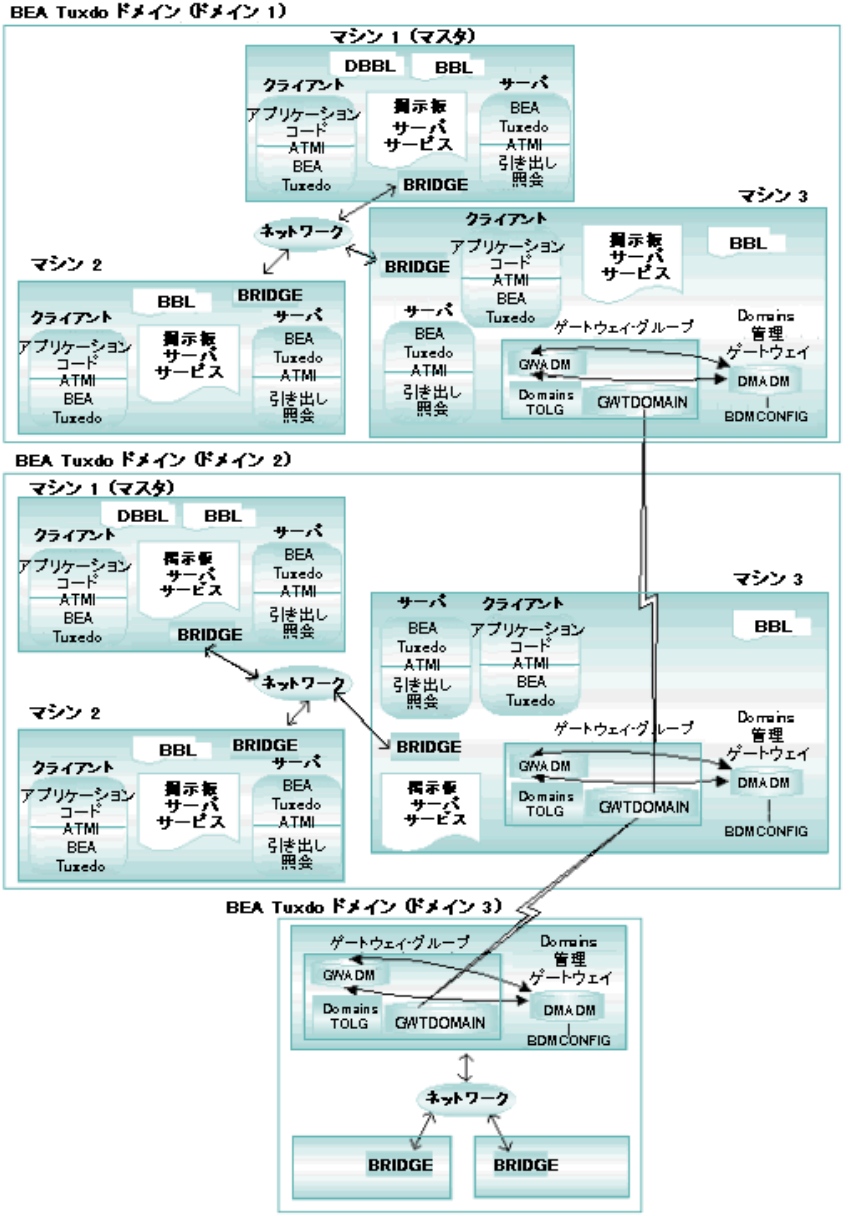


表 3-5 マルチ・ドメイン・コンフィギュレーションの構成要素

マルチ・ドメインの構成要素	説明
BRIDGE	BEA Tuxedo システムによって提供されるドメイン内のサーバ。マシン間でサービス要求を送受信し、要求をローカル・サーバ (正確にはローカル・サーバのキュー) にルーティングします。
掲示板 (BB)	システムのコンフィギュレーションと動的な情報を保持する共用メモリ・セグメント。すべての BEA Tuxedo プロセスから利用できます。
BBL	掲示板に格納されているデータ (掲示板に対する変更を含む)、およびすべてのアプリケーション・プログラムを監視する BEA Tuxedo の管理プロセス。
クライアント	BEA Tuxedo システムを介して定期的にサービスを要求する実行可能プログラム。通常、クライアント・プログラムはユーザが作成します。
DBBL	各マシン上の BBL サーバが実行されていて、正常に機能していることを確認するための専用プロセス。このサーバは、アプリケーションのマスタ・マシン上で実行され、すべての管理機能と直接通信します。
Domains ツール : `DMADM`、`GWADM`、`GWTDOMAIN`、`dmloadcf`、`dmunloadcf`、および `DMCONFIG`	■ `DMADM`—Domains 管理サーバ。 ■ `GWADM`—ゲートウェイ・グループの管理サーバ。`DMADM` サーバに登録され、ゲートウェイ・グループに使用されるコンフィギュレーション情報を取得します。 ■ `GWTDOMAIN`—リモート・ゲートウェイ・プロセスへの接続を実現するゲートウェイ・プロセス (TDomains 用)。 ■ `dmloadcf`—`DMCONFIG` ファイルをバイナリ形式の `BDMCONFIG` コンフィギュレーション・ファイルに変換します。 ■ `dmunloadcf`—`BDMCONFIG` コンフィギュレーション・ファイルをバイナリ表現から ASCII 表現に変換します。 ■ `DMCONFIG`—Domains コンフィギュレーション・ファイル。
メッセージ・キュー	クライアントとサーバ間の通信は、オペレーティング・システムでサポートされ、メモリ・ベースであるメッセージ・キューを介して行われます。

表 3-5 マルチ・ドメイン・コンフィギュレーションの構成要素 (続き)

マルチ・ドメインの構成要素	説明
メッセージング・パラダイム	クライアントとサーバ間のメッセージ転送に関する各種のモデル。たとえば、要求 / 応答モード、会話モード、イベント、任意通知型通信などがあります。
サーバ	BEA Tuxedo システムを介して指定されたサービスを提供する実行可能プログラム。通常、サーバ・プログラムはユーザが作成します。
ワークステーション・ハンドラ (WSH)	サーバ上のマルチ・コンテキストのゲートウェイ・プロセス。ワークステーション・クライアント (リモート・サイト上で実行されているクライアント・プロセス) からのサービス要求を管理します。
ワークステーション・リスナ (WSL)	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。ワークステーション・クライアント (リモート・サイト上で実行されているクライアント・プロセス) からの接続を受け付け、配信します。
ULOG (ユーザ・ログ)	エラー・メッセージが格納されるファイル。

関連項目

- 第 3 章の 38 ページ「シングル・マシン・コンフィギュレーション」
- 第 3 章の 52 ページ「Domains 管理ツール」
- 『BEA Tuxedo アプリケーションの設定』の 第 3 章の 4 ページ「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」

マルチ・ドメイン・コンフィギュレーションの特徴

複数のドメインを含むコンフィギュレーションでは、プラットフォームの相互運用性とサーバの透過性が必要です。

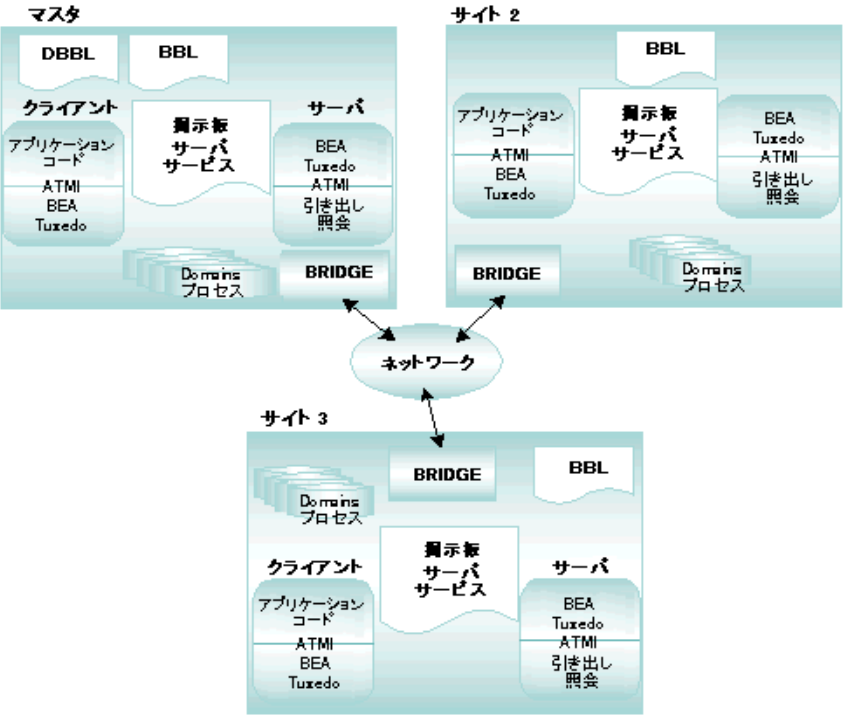
- プラットフォームの相互運用性とは、異なるマシン上で異なるオペレーティング・システムが実行されている場合でも、コードのカスタマイズを行わずに、アプリケーションがマシン間で通信できることを意味します。
- サーバの透過性とは、クライアントがサーバのロケーションを指定しなくてもサーバにアクセスできることを意味します。サーバのロケーションは掲示板に記録され、必要に応じてアクセスできます。そのため、アプリケーション自体を変更することなく、動的にサーバの移動、削除、追加を行うことができます。

BEA Tuxedo BRIDGE

BEA Tuxedo BRIDGE は、BEA Tuxedo システムによって提供されるサーバの 1 つで、マシン間でサービス要求を送受信したり、ローカル・サーバのキューに要求をルーティングします。

各 BRIDGE は、システム内のほかのすべてのブリッジとのネットワーク接続を可能にします。ネットワーク接続は必要に応じて確立され、無期限に維持されます。BRIDGE は、隠れたサーバです。つまり、明示的なコンフィギュレーション・エントリがなくても、必要に応じて自動的に起動し、停止します。メッセージは、これらの永続的なネットワーク接続にわたって非同期に送信されます。個々のメッセージに対して、ネットワーク接続のオーバーヘッドが発生することはありません。

図 3-14　マルチ・マシン (分散) アプリケーションにおける BRIDGE



関連項目

- 『BEA Tuxedo アプリケーションの設定』の 第 9 章の 1 ページ「分散アプリケーションのネットワーク設定」
- 『BEA Tuxedo アプリケーションの設定』の 第 8 章の 1 ページ「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」

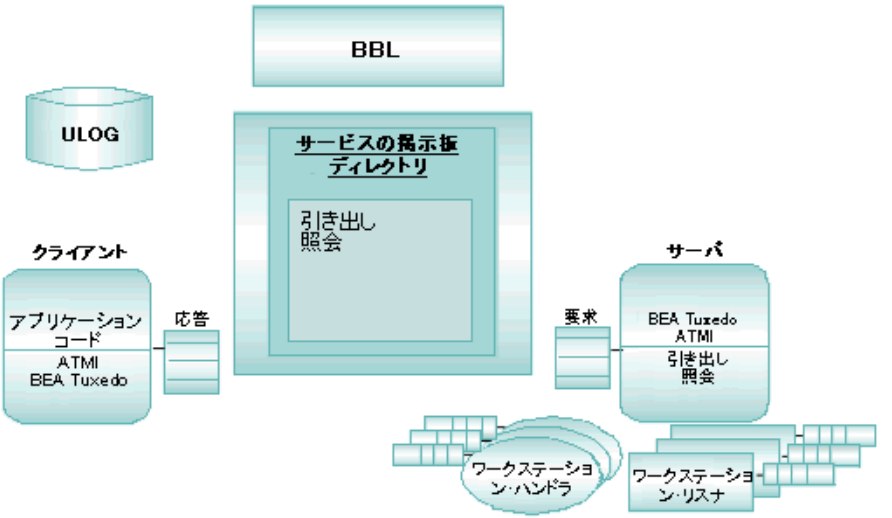
掲示板と BBL の役割

掲示板 (BB) は、すべてのアプリケーション・コンフィギュレーションと動的な処理に関する情報が実行時に保持されるメモリ・セグメントです。BB では、次の機能が提供されます。

- サービス要求を特定のサーバに割り当てます。サービスが呼び出されると、掲示板は要求されたサービスを提供するサーバを検索します。この情報とデータ依存型ルーティング基準に基づいて、要求データを有効なサーバの要求キューに挿入します。
- サーバのキューで待ち状態にある要求の数や、処理済みの要求の数など、アプリケーションの状態に関する動的な情報を保持します。
- サーバの位置透過性を実現し、運用に依存しないアプリケーションの開発を可能にします。これにより、開発および運用のコストを最小限に抑えることができます。
- サーバ名のエイリアスをサポートし、同じサービスに複数の名前を割り当てることができるようにします。この機能は、ゲートウェイなどのインタプリタを構築する場合に役立ちます。

BBL は BEA Tuxedo サーバで、掲示板の状態を定期的に調べて、システムのすべての構成要素の機能を調節します。

図 3-15 掲示板と BBL



クライアントとサーバ

- クライアントとは、ユーザからの要求を収集し、その要求を処理できるサーバに渡すプログラムです。ユーザからの入力を収集するアプリケーションのフロント・エンドの一部として、PC またはワークステーション上に置くことができます。また、ATM マシンなどの通信装置を読み取るソフトウェアの中に埋め込むこともできます。このような通信装置からデータが収集され、BEA Tuxedo サーバによって処理される前にフォーマット処理されます。
- サーバとは、一連のサービスを監視し、それらを要求したクライアントに対して自動的にディスパッチするプロセスです。それに対して、サービスとは、ビジネスに必要な特定のタスクを実行するサーバ・プログラム内の関数です。たとえば、銀行には、預け入れを受け取るサービスと、口座残高を報告するサービスが用意されています。あるサーバが、この両方のサービスをクライアントから受け取ったとします。それぞれの要求を該当するサービスにディスパッチするのは、サーバの仕事です。

DBBL

DBBL は、複数のマシン間でアプリケーションの分散を可能にするサーバです。各マシン上の BBL サーバが実行されていて、正常に機能していることを確認します。このサーバは、アプリケーションの MASTER マシン上で実行され、すべての管理機能と直接通信します。

DBBL を使用すると、コンフィギュレーションとサービスのアドレス指定情報が、コンフィギュレーション内の各マシン上の掲示板に必ず複製されるようになります。リモート・マシン上のサーバは、ローカル・マシン上の BRIDGE を介してアクセスできます。ローカル・マシン上のサーバは、直接アクセスできます。すべてのローカル通信は、オペレーティング・システムの高パフォーマンスのメッセージ・キューを介して行われます。リモート通信は、2 段階で行われます。まず、サービス要求が (ローカルの) BRIDGE を介してリモート・マシンに転送されます。次に、リモート・マシンに到達した要求が、オペレーティング・システムのメッセージを使用して適切なサーバに送られます。

Domains 管理ツール

マルチ・ドメイン・コンフィギュレーションを構築するには、既存の BEA Tuxedo アプリケーションをほかのドメインと統合する必要があります。ドメイン間の相互運用性を確認し、すべてのドメイン上のサービスへのアクセスを維持し、すべてのドメインからのサービス要求を受け付ける必要があります。これらの機能は、すべてのドメインに送信されるサービス要求と、すべてのドメインから応答されるサービス要求を処理する高度な非同期マルチタスク・ゲートウェイによって実現されます。このゲートウェイを使用するには、ドメイン・ゲートウェイ・グループとゲートウェイ・サーバ用のエントリを `TUXCONFIG` ファイルに追加する必要があります。次の図は、BEA Tuxedo システムによって提供されるマルチ・ドメイン・コンフィギュレーションの設定と管理を行うツールを示しています。

図 3-16 Domains 管理ツール

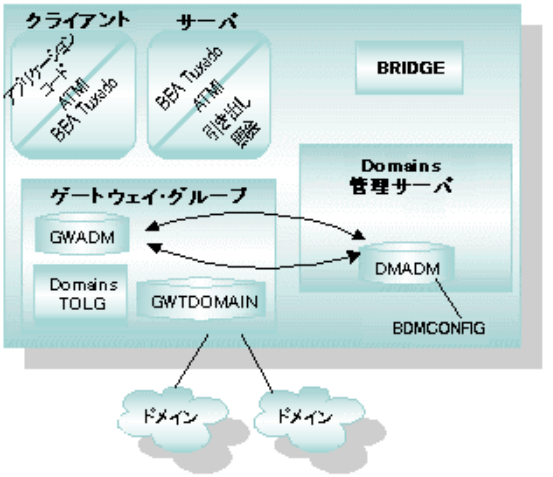


表 3-6 Domains 管理ツール

Domains ツール	機能説明
dmadmin(1)	ドメイン・ゲートウェイ・グループのコンフィギュレーション、監視、およびチューニングを動的に行うコマンド。このコマンドを使用すると、アプリケーションの実行中に BDMCONFIG ファイルを更新できます。このコマンドは、管理コマンドを DMADMIN サービスへのサービス要求に変換するフロントエンド・プロセッサとして機能します。DMADMIN サービスは、DMADM サーバにより宣言される汎用管理サービスです。DMADMIN サービスは、BDMCONFIG ファイルの管理するために、DMADM サーバによって提供される妥当性検査、取得、または更新処理を行います。
DMCONFIG(5)、BDMCONFIG	Domains のコンフィギュレーション情報は、すべて BDMCONFIG ファイルと呼ばれるバイナリ・ファイルに格納されています。テキスト形式の Domains ゲートウェイ・コンフィギュレーション・ファイル (DMCONFIG) は、テキスト・エディタを使用して作成したり編集できます。コンパイルされた BDMCONFIG ファイルは、システムの実行中に更新できます。

表 3-6 Domains 管理ツール (続き)

Domains ツール	機能説明
dmloadcf と dmunloadcf	dmloadcf は、DMCONFIG ファイルを読み取り、構文をチェックし、オプションでバイナリ形式の BDMCONFIG コンフィギュレーション・ファイルをロードします。dmunloadcf は、BDMCONFIG コンフィギュレーション・ファイルをバイナリ形式からテキスト形式に変換します。
DMADM(5)	実行時の Domains コンフィギュレーションの管理を可能にする Domains 管理サーバ。DMADM は、ゲートウェイ・グループの登録サービスを提供します。このサービスは、GWADM サーバの初期化プロシージャの一部として GWADM サーバによって要求されます。登録サービスは、要求元のゲートウェイ・グループが要求するコンフィギュレーション情報をダウンロードします。DMADM サーバは、登録したゲートウェイ・グループのリストを維持し、コンフィギュレーションに行った変更をこれらのゲートウェイ・グループに伝達します。
GWADM(5)	特定のゲートウェイ・グループの実行時の管理をサポートするゲートウェイ管理サーバ。このサーバは、DMADM サーバに登録され、対応するゲートウェイ・グループで使用されるコンフィギュレーション情報を取得します。GWADM は DMADMIN からの要求を受け付け、実行時統計情報を取得したり、指定されたゲートウェイ・グループの実行時オプションを変更します。GWADM は、定期的に "I-am-alive" メッセージを DMADM サーバに送信します。DMADM から応答がなければ、GWADM は再度登録を行います。このプロセスによって、常にグループの Domains コンフィギュレーションの最新コピーが GWADM サーバに確保されます。
GWTDOMAIN(5)	接続されているすべてのドメイン内のクライアントおよびサーバからのメッセージを受信し、転送するゲートウェイ・プロセス (TDomains 用)。
BDMCONFIG	マルチ・ドメイン・コンフィギュレーション用のバイナリ形式のコンフィギュレーション・ファイル。

IPC メッセージ・キュー

BEA Tuxedo システムでは、IPC メッセージ・キューを使用して、特定のマシン上で実行されるプロセス間の通信がサポートされています。IPC メッセージ・キューは一時メモリ領域で、通常はオペレーティング・システムによって提供され、クライアントとサーバ間の通信に使用されます。デフォルトでは、各サーバが専用の IPC メッセージ・キューを持ち、要求と応答を受け取ります。この方法は「単一サーバ、単一キュー (SSSQ)」と呼ばれます。必要に応じて、デフォルトを変更して、複数のサーバが同じキューから読み取るように設定することもできます。この方法は「複数サーバ、単一キュー (MSSQ)」と呼ばれます。同じアプリケーションで、SSSQ と MSSQ の両方を使用できます。サーバは、いずれかのタイプのキューに割り当てることができます。

単一サーバ、単一キュー (SSSQ) の使用

SSSQ を理解するために、スーパーマーケットを例に考えてみます。スーパーマーケットには、複数のレジの列があります。レジの列はそれぞれ別個のキューに相当します。各レジでは、顧客が 1 人のレジ係の作業を待ちます。その列の作業の速さは、レジ係によって異なります。ある顧客に対する作業に遅れが生じると、その列の後続の顧客にもそれぞれ遅れが発生します。ただし、ほかの列には何の影響もありません。この方式は、異なる種類のサービスを提供する複数のサーバ間でロード・バランシングを行い、作業量を調整するために使用します。要求が比較的小さい顧客は、通常の顧客とは別のキューを持つサーバによって処理することができます。小さい要求のためのサイクルまたはレジスタを確保することにより、スループットが向上します。

複数サーバ、単一キュー (MSSQ) のセットの使用

MSSQ では、オペレーティング・システムによって提供される IPC メッセージングを使用して、さらにロード・バランシングが進められます。1 つのキューには、常に同じサービスを提供する複数のサーバが対応しています。ある要求がサーバのキューに送信され、そのキューが MSSQ セットの一部である場合、メッセージがキューから取り出されて、使用可能な最初のサーバに送られます。このようにして、個々のキューのレベルでロード・バランシングが行われます。

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

大半のアプリケーションでは、「複数サーバ、単一キュー (MSSQ)」は重要な役割を果たします。この方式は、サービスの待ち時間の合計を最小限に抑える場合に理想的です。MSSQ は、要求を処理できるサーバがアイドル状態のときに、サービス要求を待機させないようにする場合に使用します。

次の場合には、MSSQ の使用をお勧めします。

- サービスの処理時間が最重視される場合
- サーバの数が適度 (2 〜 12) である場合
- 各サーバが同じサービスを提供する場合
- 適度なサイズ (キューのサイズの 75% 未満) のメッセージが含まれている場合
- キューのロードに基づいて自動的にサーバが増減するように、MSSQ を動的にコンフィギュレーションできる場合

注記 障害耐久性から、MSSQ は常に複数のサーバで使用してください。

MSSQ は、長いメッセージがサービスに渡される場合には適していません。長いメッセージによって、キューがいっぱいになることがあるからです。キューがいっぱいになると、非ブロッキング送信が失敗するか、またはブロッキング送信がブロックされます。

次の場合には、MSSQ の使用をお勧めしません。

- バッファ・サイズが 1 つのキューでいっぱいになる場合。
- サーバ数が多すぎる場合。ただし、MSSQ を複数使用して対処することもできます。
- サーバごとにサービスが異なる場合。

使用例

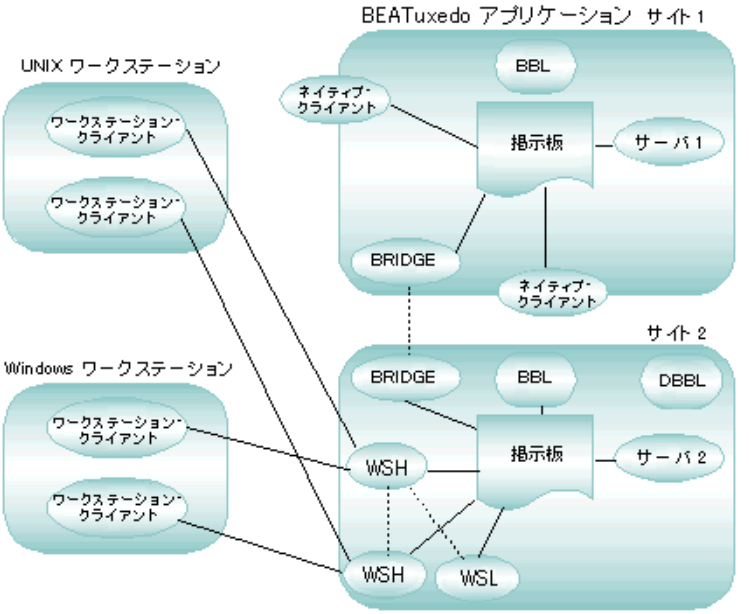
MSSQ を理解するために、銀行を例に考えてみます。銀行では、同じサービスを行う複数の窓口が 1 列に並んだ顧客に応対します。次に空く窓口が、常に列の次の顧客に応対します。この例では、各窓口がすべての顧客サービスを行うことができなければなりません。BEA Tuxedo 環境の場合、単一のキューを共有するように設定されたすべてのサーバは、常に同じサービスを提供する必要があります。MSSQ の利点は、個々のキューのレベルで、ロード・バランシングのもう 1 つの形式を提供していることです。

ワークステーション・ハンドラとワークステーション・リスナ

ワークステーション・コンポーネントは、ワークステーション上のクライアントでもネイティブの BEA Tuxedo アプリケーションを利用できるようにします。このコンポーネントを使用した場合、ワークステーションがアプリケーションの管理ドメイン内になくてもかまいません。

次の図は、2 つのワークステーション・クライアント (WSC) が接続されたアプリケーションを示しています。1 つのクライアントは、UNIX システム・ワークステーション上で実行され、もう 1 つのクライアントは Windows 2000 ワークステーション上で実行されています。どちらの WSC も ワークステーション・ハンドラ (WSH) プロセスを通じてアプリケーションと通信しています。最初に、両者ともワークステーション・リスナ (WSL) と通信することによって、アプリケーションに参加します。ワークステーションは、クライアントが代理のハンドラ・プロセスを通してアプリケーションのサービスにアクセスできる環境を定義します。

図 3-17　BEA Tuxedo アプリケーションとワークステーション・コンポーネント

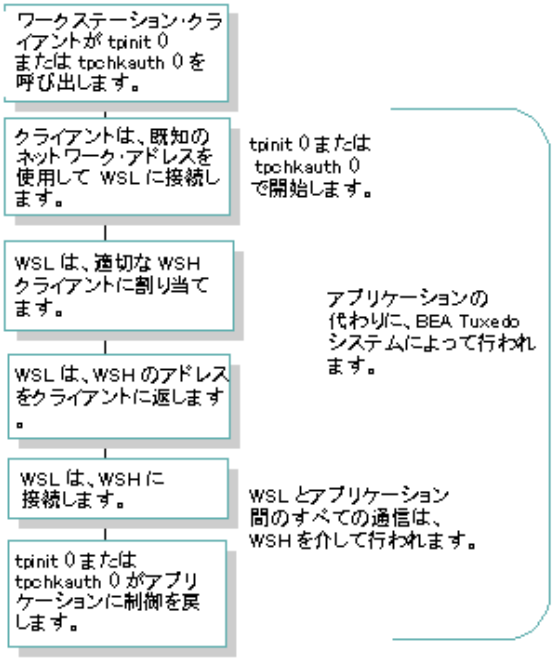


ワークステーション上のプログラミング環境は、マシンのオペレーティング・システムによって決まります。アプリケーションの管理ドメインへの接続は、ローカル・エリア・ネットワーク (LAN) によって行われます。そのため、アプリケーション・サービスを配布するハードウェアおよびソフトウェアのプラットフォームの選択で柔軟性がさらに高まります。

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

ワークステーション・クライアントは、次の手順に従ってアプリケーションに接続します。

図 3-18 アプリケーションに接続中の WSC



ユーザ・ログ (ULOG)

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

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

ULOG の作成

ULOG は、次のいずれかの処理が行われるたびに、BEA Tuxedo システムによって生成されます。

- 新しいコンフィギュレーション・ファイルのロード
- アプリケーションの起動

ULOG メッセージの例

次は、ULOG メッセージの例です。

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

ULOG メッセージは、タグとテキストから構成されています。

タグは、次の要素から構成されます。

- 時間、分、秒で時刻を表す 6 桁の文字列 (hhmmss)。
- マシンの名前 (UNIX システムの uname -n コマンドの戻り値)。

- メッセージを記録しているプロセスの名前と識別子。このプロセス ID には、オプションでトランザクション ID を含めることもできます。また、スレッド ID (1) およびコンテキスト ID (0) も含まれています。

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

テキストは、次の要素から構成されます。

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

タグが示す内容	テキストが示す内容
■ このメッセージは、午後 12 時 15 分頃にログに書き込まれました。 ■ エラーが発生したマシンは、gumby です。 ■ このメッセージは、プロセス ID が 27190 である simpserv プロセスによって記録されました。 ■ スレッド ID は 1 です。 ■ コンテキスト ID は 0 です。	■ このメッセージは、LIBTUX カタログから送信されました。 ■ メッセージの番号は 262 です。 ■ メッセージ自体の内容は、std main starting です。

注記 メッセージの詳細を確認するには、カタログの名前と番号を記録します。この情報を使用して、該当するカタログでメッセージを参照します。

ULOG の存在する場所

デフォルトでは、ユーザ・ログは ULOG.mmddyy (mmddyy は、月、日、年で表された日付) という名前で $APPDIR ディレクトリに生成されます。UBBCONFIG ファイルの MACHINES セクションで ULOGPFX パラメータを設定すると、このファイルを任意の場所に置くことができます。