BEA Tuxedo®

BEA Tuxedo システム入門

BEA Tuxedo 8.1
2003 年 1 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

目次

3. BEA Tuxedo システムの管理とサーバ・プロセス

4. BEA Tuxedo の管理ツール

このマニュアルについて

このマニュアルでは、BEA Tuxedo アプリケーション・トランザクション・モニタ・インターフェイス (ATMI) 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 システムの管理とサーバ・プロセス」― BEA Tuxedo システムに基づいて作成された ATMI アプリケーションのインフラストラクチャを共同で構成する中心的な BEA Tuxedo システム管理とサーバ・プロセス
- 第 4 章「BEA Tuxedo の管理ツール」― Tuxedo アプリケーションを管理するためにユーザが利用できる BEA Tuxedo 管理プロセス

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 システムを使用して分散 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、e-docs Web サイトの BEA Tuxedo マニュアル・ページから入手できます。また、マニュアルの CD-ROM にも収められています。この PDF を Adobe Acrobat Reader で開くと、マニュアル全体または一部をブック形式で印刷できます。PDF 形式を利用するには、BEA Tuxedo マニュアル・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

次の BEA Tuxedo ドキュメントには、『BEA Tuxedo システム入門』に関連する情報が記載されています。

- 『製品の概要』
- 『BEA Tuxedo の相互運用性』
- 『BEA Tuxedo アプリケーションの設定』
- 『BEA Tuxedo アプリケーション実行時の管理』

ATMI、トランザクション処理、および BEA Tuxedo ATMI のアーキテクチャ 環境の構成と管理の詳細については、「参考資料」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 8.1 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、`http://www.bea.com` の BEA WebSUPPORT を通して BEA カスタマ・サポートにお問い合わせください。カスタマ・サポートへの問い合わせ方法は、製品パッケージに同梱されている カスタマ・サポート・カードにも記載されています。

カスタマ・サポートへお問い合わせの際には、以下の情報をご用意ください。

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

表記上の規則

このマニュアルでは、以下の表記規則が使用されています。

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

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

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

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

|	構文の行で、相互に排他的な選択肢を分離します。記号は入力しません。

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

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

1 BEA Tuxedo システムの基本概念

以下の節では、BEA Tuxedo プログラミング環境の概要を説明します。

- 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 のクライアントとは
- BEA Tuxedo サーバとは
- BEA Tuxedo システムによって提供されるアプリケーション処理サービス
- BEA Tuxedo システムによって提供される管理サービス

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 (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 20 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。

アーキテクチャに関する機能

BEA Tuxedo システムは、ATMI アプリケーションのアーキテクチャ的側面を実現する多くの機能を提供します。

- 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。
- 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。

- スケーラビリティー－アプリケーション設計者が、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 Tuxedo 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して、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。

管理に関する機能

BEA Tuxedo システムは、ATMI アプリケーションの管理的側面を実現する多くの機能を提供します。

- 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ー－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。
- アプリケーション固有のイベントおよびシステム・イベントの通知－ BEA Tuxedo システムでは、サーバが予期せず終了したりネットワークで障害が発生したりといったアプリケーション・イベントやシステム・イベントの詳細情報が提供されます。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、Tuxedo パブリッシュ・アンド・サブスクライブ・コンポーネント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。
- 管理情報ベース (MIB) －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、および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、フィールド操作言語 (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。

プログラミングに関する機能

BEA Tuxedo システムは、ATMI アプリケーションのプログラミングの側面を実現する多くの機能を提供します。

- 通信方法－ BEA 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 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、アプリケーション設計者はデータの整合性を維持しながら、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。
- 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。BEA Tuxedo は、トランザクションの境界判定のための独自の ATMI インターフェイスも提供します。

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

クライアント / サーバ アーキテクチャには以下の特徴があります。

- 非対称のプロトコル－クライアントとサーバの間の多対 1 の関係。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 プロセスと同じマシン、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。クライアントがサーバの位置を意識する必要はありません。
- 名前空間透過性－クライアントは、ネットワーク上のどのサーバの検索にも同じ命名規則 (および名前空間) を使用できる必要があります。
- 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。

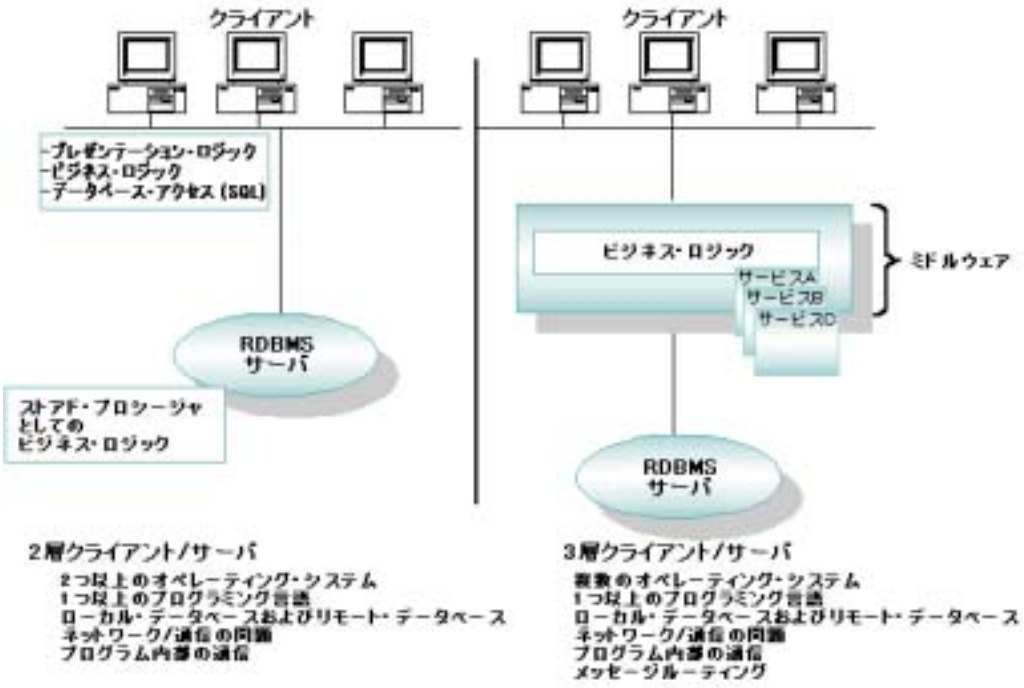
2 層および 3 層のクライアント / サーバ・アーキテクチャの違い

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 プログラムの構成要素またはアプリケーション内の 1 つまたは複数のサーバ プログラムの構成要素になります。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、次の図に示されているように、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。

図 1-12 2 層および 3 層のクライアント / サーバ・モデル



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。

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。

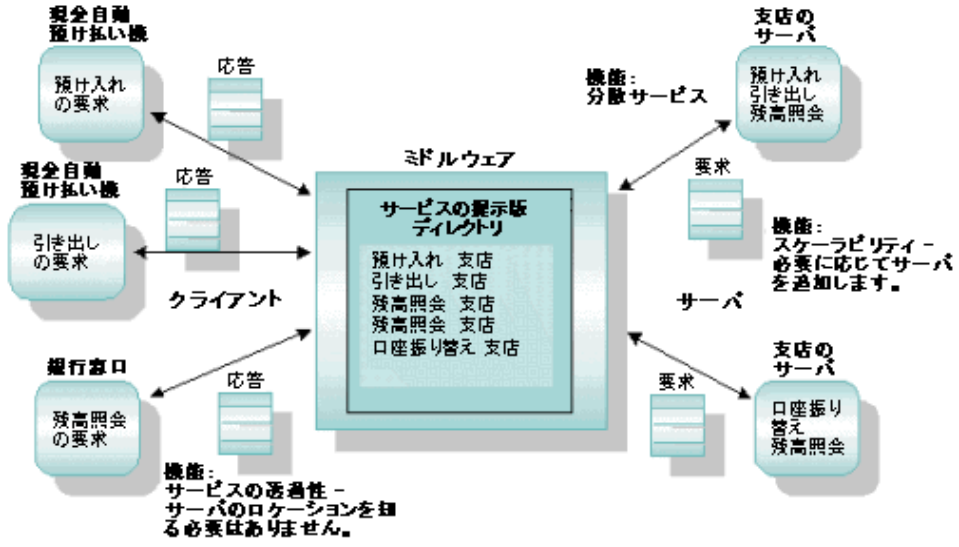
- 小規模な事業およびラップトップ－クライアント、ミドルウェア・ソフトウェア、および大部分のビジネス・サービスが同じマシン上で動作します。この方法は、歯科医院、在宅業務、ラップトップ・コンピュータを多用する出張者などの小規模な事業に使用することをお勧めします。
- 中小企業および社内の部署－ LAN ベースの単一サーバ・アプリケーションを必要とします。このタイプのアプリケーションのユーザには、複数の医師を擁する病院、複数の部署がある企業、複数の支店を持つ銀行など、中小規模のビジネスが該当します。このタイプのアプリケーションでは、複数のクライアントが 1 つのローカル・サーバと通信します。管理業務は単純です。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。社内ネットワーク、イントラネット、およびインターネット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グローバル・ディレクトリが定義されています。

たとえば、次の銀行業務サンプル・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を該当する支店で見つけます。

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



この銀行業務サンプル・アプリケーションでは、BEA Tuxedo アプリケーションを構成する以下の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・パラダイムで構成されます。ミドルウェアには、クライアント・ユーザ・インターフェイス、アプリケーション・ロジック、またはサーバによって提供されるサービスは含まれません。

BEA Tuxedo 銀行業務サンプル・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。

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 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。そのサーバは、その責任として該当するサービスに各要求をディスパッチします。

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送信します。

BEA Tuxedo システムによって提供されるアプリケーション処理サービス

BEA Tuxedo システムでは、アプリケーション開発者が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
- 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

Tuxedo アプリケーション処理サービスの説明については、2-1 ページの「BEA Tuxedo ATMI のアーキテクチャ」を参照してください。

BEA Tuxedo システムによって提供される管理サービス

BEA Tuxedo システムでは、アプリケーション管理者が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。

- アプリケーションの起動とシャットダウン
- 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
- ワークステーション管理
- セキュリティ管理
- トランザクション管理
- メッセージ・キューイング管理
- イベント管理

管理サービスを提供する Tuxedo システム管理プロセスの説明については、3-1 ページの「BEA Tuxedo システムの管理とサーバ・プロセス」および 4-1 ページの「BEA Tuxedo の管理ツール」を参照してください。管理サービスの詳しい使い方については、『BEA Tuxedo アプリケーションの設定』および『BEA Tuxedo アプリケーション実行時の管理』を参照してください。

2 BEA Tuxedo ATMI のアーキテクチャ

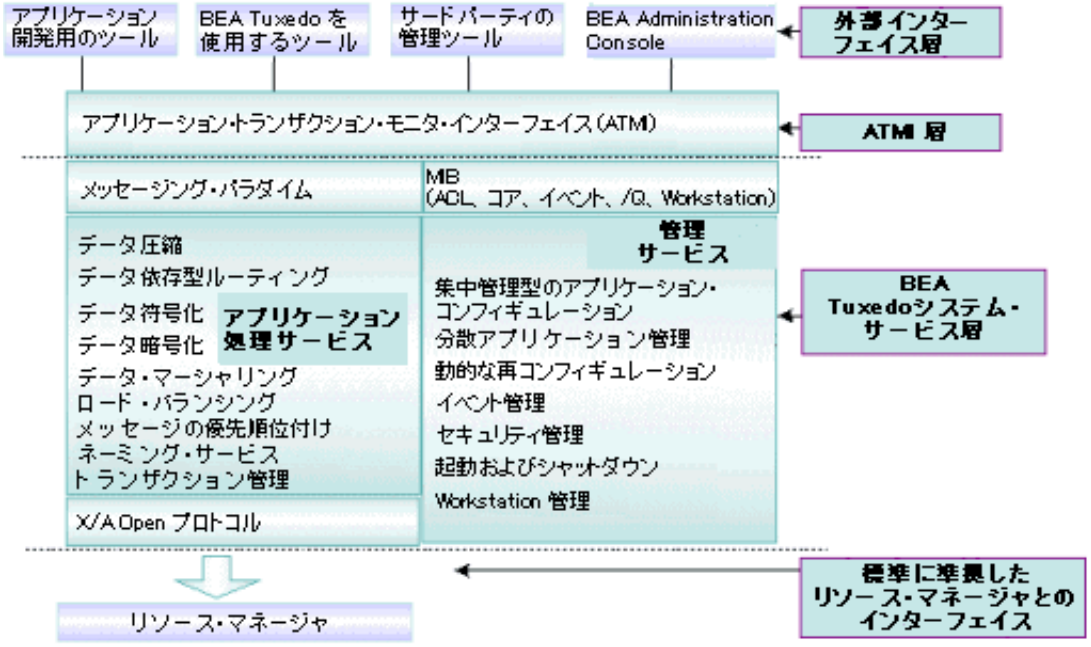
以下の節では、BEA Tuxedo ATMI 環境の基本的な構成要素について説明します。

- BEA Tuxedo ATMI 環境の基本アーキテクチャ
- ATMI の使用
- BEA Tuxedo のメッセージング・パラダイム
- 要求の入れ子と転送
- BEA Tuxedo によるメッセージの処理
- 型付きバッファ
- データ圧縮
- データ依存型ルーティング
- データの符号化と復号化
- データの暗号化
- データの暗号化
- ロード・バランシング
- メッセージの優先順位付け
- ネーミング

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 2-1BEA 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。

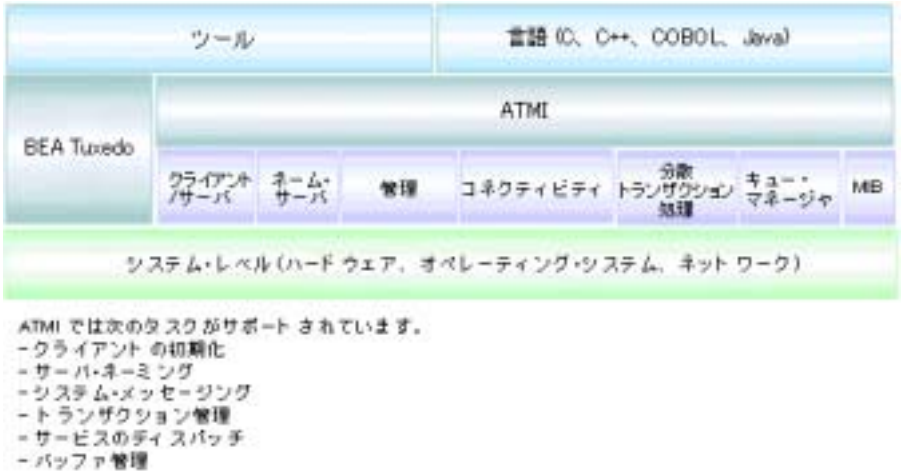
構成要素	説明
外部インターフェイス層	この層は、ユーザとシステム間のインターフェイスから構成されています。アプリケーション開発ツールと、BEA Tuxedo Administration Console などの管理ツールの両方が含まれています。Administration Console は、標準の管理コンソールとやり取りすることができます。そのため、ユーザは BEA Tuxedo ATMI 環境とネットワークのコンフィギュレーションを 1 つのコンソールから管理できます。また、アプリケーションのアーキテクチャの設計者や開発者は、独自の管理ツールまたはアプリケーション固有のツールや特定分野のツールを MIB の最上位に構築できます。
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。BEA Tuxedo ATMI メッセージング・パラダイムには、要求 / 応答、会話、キューイング、パブリッシュ・アンド・サブスクライブ、および任意通知型通知 . 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。MIB および MIB インターフェイスについては、4-1 ページの「BEA Tuxedo の管理ツール」を参照してください。

構成要素	説明 （続き）
BEA Tuxedo サービス (アプリケーション処理サービスおよび管理サービス)	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。 BEA Tuxedo 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、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、およびサービスとイベントのネーミングがあります。これらのサービスについては後述します。 BEA Tuxedo の管理者から利用できる管理サービスには、アプリケーションの起動とシャットダウン、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、動的なアプリケーションの再コンフィギュレーション、ワークステーション管理、セキュリティ管理、トランザクション管理、メッセージ・キューイング管理、およびイベント管理があります。 管理サービスを提供するシステム管理プロセスについては、3-1 ページの「BEA Tuxedo システムの管理とサーバ・プロセス」および 4-1 ページの「BEA Tuxedo の管理ツール」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 (分散) トランザクション内でアクセスして制御できます。

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 つの単純なインターフェイスです。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 2-2ATMI の使用



ATMI ライブラリには、BEA Tuxedo ATMI 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 (ルーチンなど) 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。

ATMI 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、C および COBOL のバインディング対応の ATMI 関数と、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 2-1ATMI 関数

タスク . .	C **関数** . .	COBOL **言語の関数** . .	**説明** . .
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。
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。

表 2-1ATMI 関数 （続き）

タスク	C 関数	COBOL 言語の関数	説明
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。
メッセージ・キューイング通信	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 2-1ATMI 関数 （続き）

タスク . .	C 関数 . .	COBOL 言語の関数 . .	説明 . .
パブリッシュ・アンド・サブスクライブ通信	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。

表 2-1ATMI 関数 （続き）

タスク . .	C 関数 . .	COBOL 言語の関数 . .	説明 . .
トランザクション管理 (表最後の注記を参照)	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。
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。

表 2-1ATMI 関数 （続き）

タスク . .	C 関数 . .	COBOL 言語の関数 . .	説明 . .
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　BEA Tuxedo 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。BEA Tuxedo は X/Open TX トランザクション管理関数もサポートしているので、トランザクション管理にそれらの関数を使用することもできます。

関連項目

- 『BEA Tuxedo アプリケーション実行時の管理』の 2-33 ページの「ATMI を使用してシステム・エラーとアプリケーション・エラーを処理する」

BEA Tuxedo のメッセージング・パラダイム

アプリケーションのサーバ・プロセスの管理およびトランザクションの管理に加えて、BEA Tuxedo ATMI ではクライアント / サーバ通信も管理します。つまり、クライアント (およびサーバ) が次表で示されたメッセージング パラダイムのいずれかを使用してアプリケーション・サービスを呼び出せるようにします。

BEA Tuxedo ATMI のメッセージング・パラダイム	説明
要求 / 応答型通信	クライアントから 1 つの要求が出され、呼び出された要求 / 応答型サーバから 1 つの応答が返される単純なダイアログ。要求 / 応答型トランザクションは通常は人が関与し、したがって即座の対応が必要となります。要求 / 応答型のトランザクションは高い優先順位で実行されます。
会話型通信	クライアントと呼び出された会話サーバの間の状態が保持される接続 (メッセージからメッセージへ保持されるコンテキスト)。会話型トランザクションも通常は人が関与し、したがって即座の対応が必要となります。会話型のトランザクションは高い優先順位で実行されます。
メッセージ・キューイング通信	クライアントとサーバ間の時間に依存しない通信。キューに入れられたトランザクションは、優先順位の高いまたは低いメッセージとして実行できます。BEA Tuxedo システムには、*/Q* という回復可能キューの独自バージョンがバンドルされています。
パブリッシュ・アンド・サブスクライブ通信	BEA Tuxedo ATMI アプリケーションのクライアントとサーバ間でのイベントの非同期ルーティング。パブリッシュ・アンド・サブスクライブ・トランザクションは通常、優先順位の高いメッセージとして実行されます。BEA Tuxedo システムには、イベント・ブローカというトランザクション対応パブリッシュ・アンド・サブスクライブ・システムがあります。
任意通知型通知メッセージング	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。任意通知型メッセージは、イベント・ブローカによって処理されます。

要求 / 応答型通信

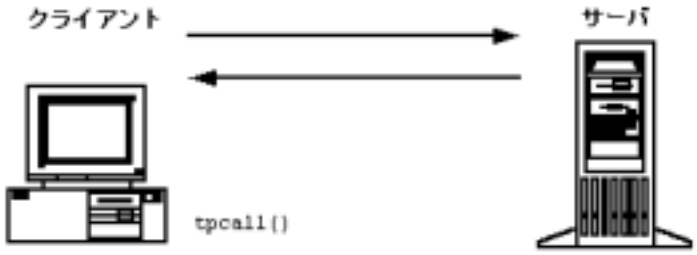
ATMI クライアントとサーバの間で要求 / 応答型通信をインプリメントするために、BEA Tuxedo システムではプロセス間通信 (IPC) メッセージ キューを使用します。キューは、コネクションレス型通信の基本要素です。各サーバには、要求キューと呼ばれる IPC メッセージ・キューが割り当てられ、各クライアントには応答キューが割り当てられます。そのため、クライアント・アプリケーションでは、サーバとの継続的な接続を確立する代わりに、要求をサーバのキューに登録して、サーバにその要求を送信できます。また、アプリケーションの応答キューからメッセージを取り出して、サーバからのメッセージを確認して取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

要求 / 応答型モデルは、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。

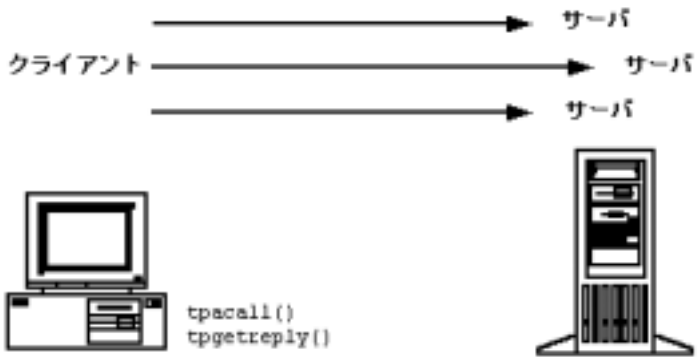
図 2-3 同期要求 / 応答型通信



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 2-4 非同期要求 / 応答型通信

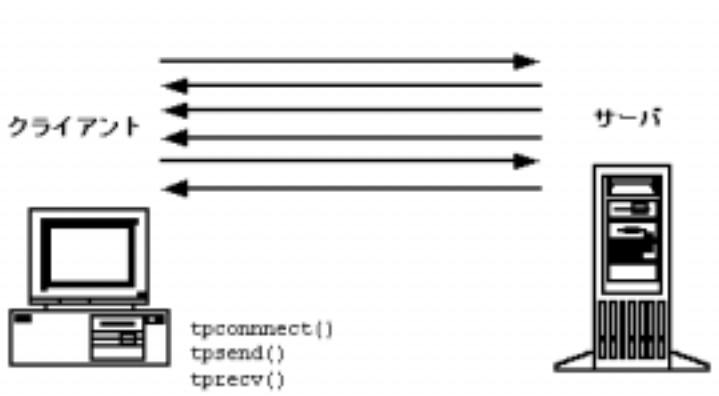


会話型通信

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 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。

図 2-5 会話型通信



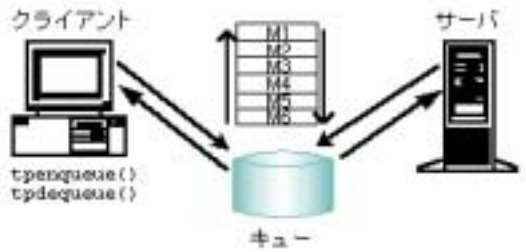
会話は入れ子にすることもできます。ただし、そのためにパフォーマンスが低下する場合があります。会話には、トランザクションまたはサービス要求のいずれかが含まれます。会話型サービスによってサービス呼び出しを行ったり、会話を確立できます。ただし、これらのサービス呼び出しと会話は転送できません。会話は、トランザクションのスコープ内に置いて、トランザクションによって制御されます。

メッセージ・キューイング通信

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、ATMI 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、FIFO 順、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。

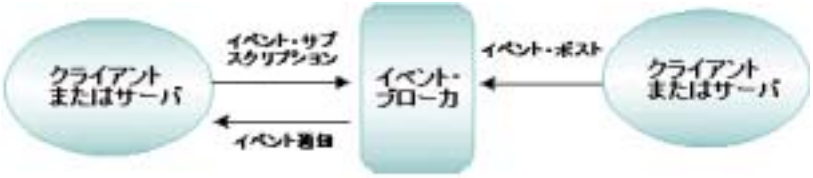
アプリケーション・キューを使用するには、アクセスするキューと、そのキューが置かれたキュー・スペースに名前を付ける必要があります。アプリケーションでは、複数のキュー・スペースを使用できます。また、各キュー・スペースには、複数のメッセージ・キューを格納できます。

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。

パブリッシュ・アンド・サブスクライブ通信

BEA Tuxedo のイベント・ブローカと呼ばれるパブリッシュ・アンド・サブスクライブ・コンポーネントは、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 ATMI クライアント・プロセスとサーバ・プロセスは、「サブスクリプション」に基づいて相互に通信します。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 2-7 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。

通知されるイベントのタイプ

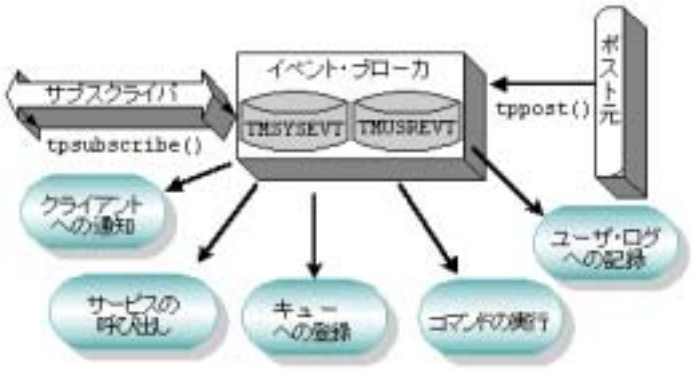
BEA Tuxedo システムでは、次のイベントが通知されます。

- アプリケーション定義のイベント－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。
- システム・イベント－サーバやネットワークの障害など、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。

イベントの通知

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 2-8 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。

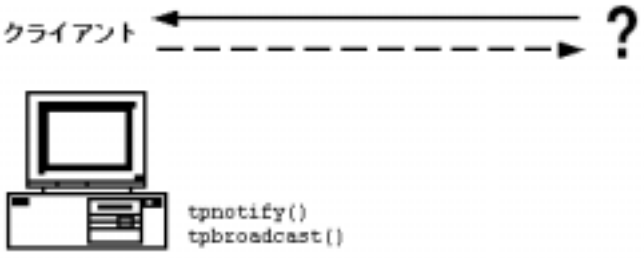
- ディスク・ベースのキューへの登録
- 非同期のサービス呼び出し
- ユーザ・ログのエントリ
- 任意通知型メッセージ
- システム・コマンド

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、ATMI 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。イベント・ブローカによって管理されるこの機能によって、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。

図 2-9 任意通知型通知のメッセージング



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。

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 1-7 ページの「要求 / 応答型モデル (同期呼び出し)」
- 『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 1-12 ページの「会話型通信」
- 3-15 ページの「BEA Tuxedo メッセージ・キュー・サーバ」
- 4-13 ページの「コマンド行ユーティリティを使用したアプリケーション・キューの管理」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 1-16 ページの「キュー・ベースの通信」
- 3-18 ページの「BEA Tuxedo パブリッシュ・アンド・サブスクライブ・サーバ」
- 4-19 ページの「イベント・ブローカを使用したイベントの管理」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 1-15 ページの「イベント・ベースの通信」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 1-13 ページの「任意通知型通知」

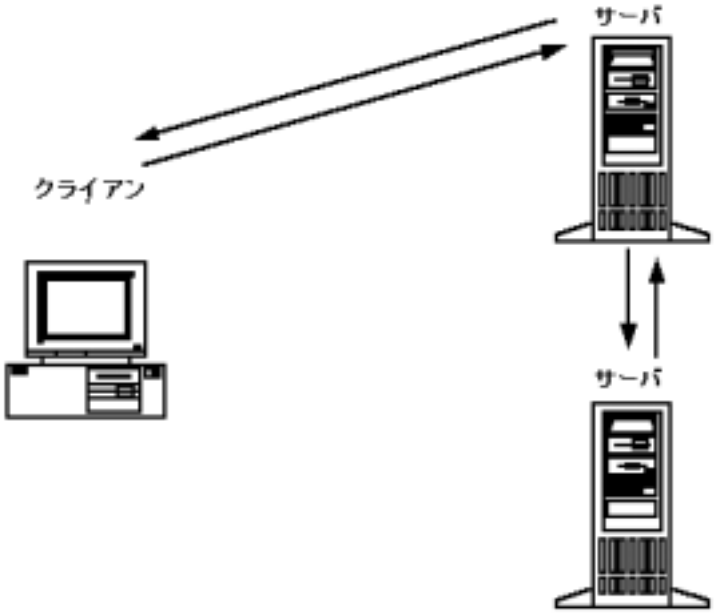
要求の入れ子と転送

サービス要求の入れ子および転送を利用すると、BEA Tuxedo サービスは ATMI クライアントとして機能して他のサービスを呼び出すことができます。

入れ子になった要求

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

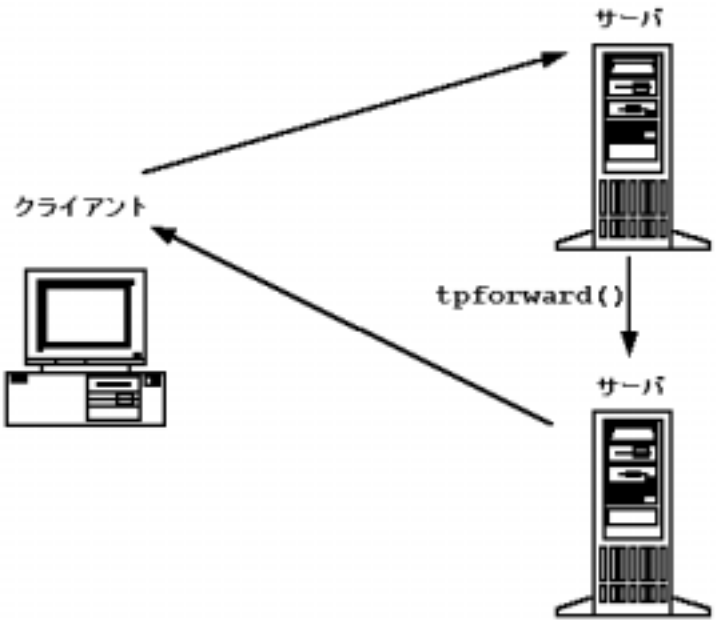
ある顧客が現金自動支払機を使って、普通口座から当座預金に預金を移すとします。振替に必要な操作は、BEA Tuxedo アプリケーションによって行われます。まず、顧客に代わって、クライアントが TRANSFER サービスへの要

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。

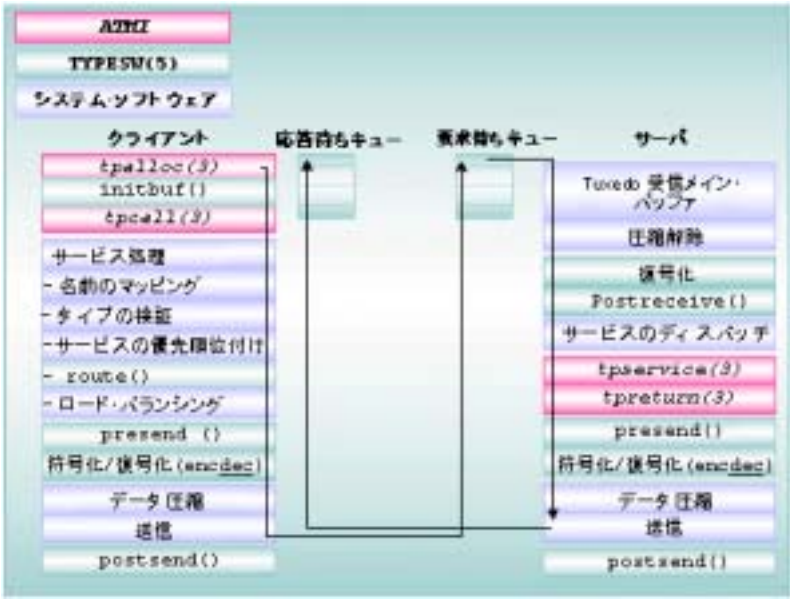
関連項目

- 『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 1-9 ページの「入れ子になった呼び出し」
- 『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 1-10 ページの「転送された呼び出し」

BEA Tuxedo によるメッセージの処理

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 システムでは、オペレーティング・システム (OS) のプロセス間通信 (IPC) メッセージ・キューを使用して、ATMI 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、OS 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、データ依存型ルーティングが使用されます。データ依存型ルーティングは、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・オプションです。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。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、フィールド操作言語 (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

サービス要求処理の利点は次のとおりです。

- 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

- 2-11 ページの「BEA Tuxedo のメッセージング・パラダイム」
- 2-27 ページの「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
- MBSTRING (BEA Tuxedo 8.1 の新機能)

バッファ・タイプは、Tuxedo (`UBBCONFIG`) 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。Tuxedo 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、『BEA Tuxedo のファイル形式とデータ記述方法』の `tuxtypes(5)` の `tm_typesw` の説明を参照してください。

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。さまざまなタイプの BEA Tuxedo バッファの説明については、『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』の「型付バッファの管理」を参照してください。

関連項目

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

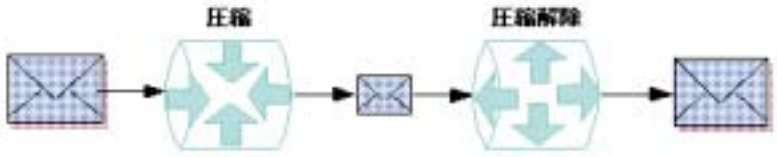
データ圧縮

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 (MIB の動的部分) 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とは、1 つ以上のローカルまたはリモートのクライアントが、ネットワーク接続された複数のマシン上の 1 つ以上のサーバと通信するアプリケーションです。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。

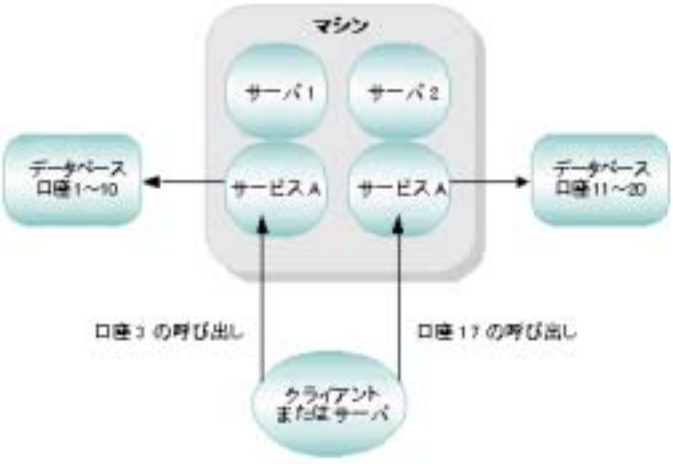
水平分離型データベースでのデータ依存型ルーティングの例

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

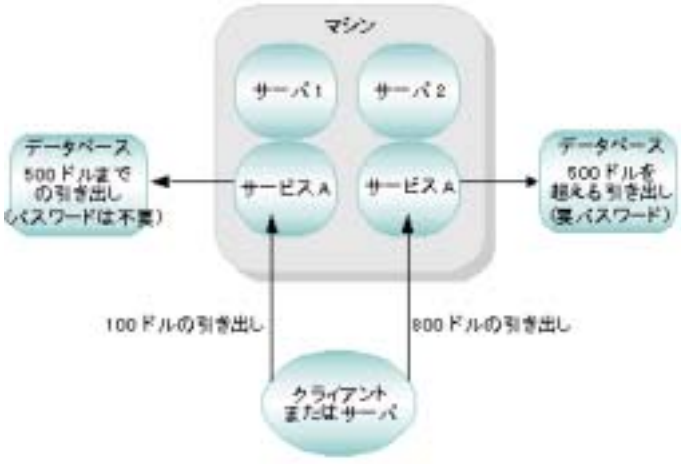
2 つのクライアントが 1 万円と 8 万円の引き出しを要求したとします。引き出しの規則でデータ依存型ルーティングが有効になっている場合、BEA Tuxedo では次のような処理が行われます。

1. 2 つのサービス要求で指定された引き出し額 (100 ドルと 800 ドル) を取得します。

2. 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、500 ドルまでのすべての引き出し要求をサーバ 1 が処理し、500 ドルを超えるすべての引き出し要求をサーバ 2 が処理しています。
3. それぞれの要求を該当するサーバに送信します。つまり、100 ドルの要求をサーバ 1 に転送し、800 ドルの要求をサーバ 2 に転送します。

次の図は、このプロセスを示しています。

図 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

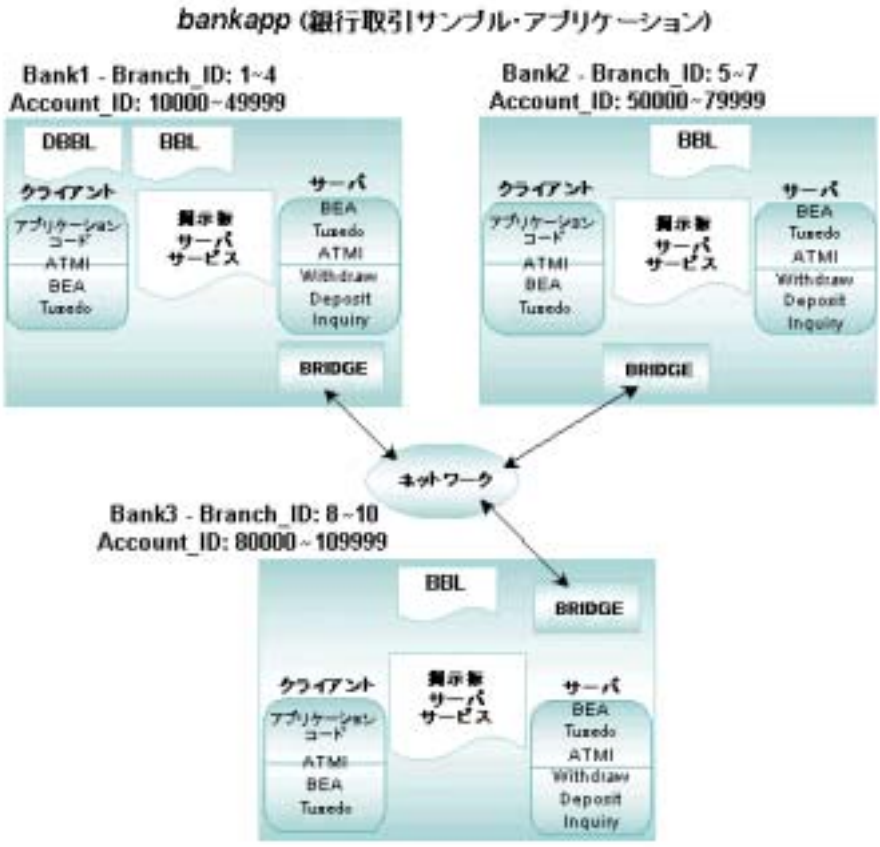


分散アプリケーションでのデータ依存型ルーティングの例

次の図は、分散アプリケーションで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、BEA Tuxedo アプリケーションに関与する他のマシンに転送できるように、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ます。

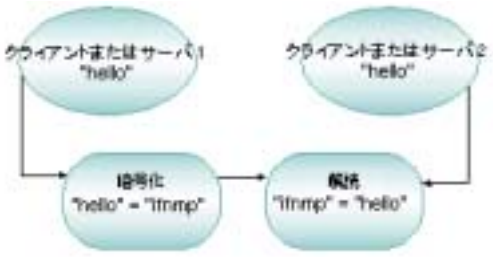
BEA Tuxedo システムではデフォルトで、外部データ表現 (XDR) アルゴリズムが採用されています。BEA Tuxedo システムの関数をユーザ定義の関数で置き換えると、このアルゴリズムをカスタマイズできます。符号化および復号化は、マシン間でのみ使用されます。また、リモート・マシンのデータ表現がローカル・マシンのものと異なる場合にのみ使用されます。符号化と復号化によって、異なるデータ構造を持つマシンが異機種の BEA Tuxedo システムで動作できるようになります。プログラマは、それぞれの環境に適した表現でデータを管理できます。

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、メッセージの受け取りユーザ以外のすべての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。

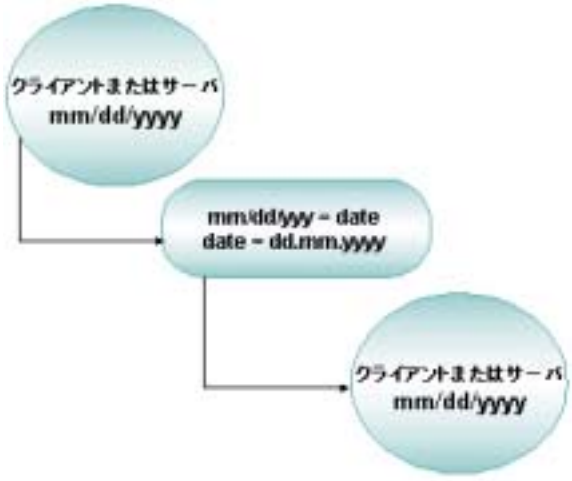
データの暗号化

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 (RPC) 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

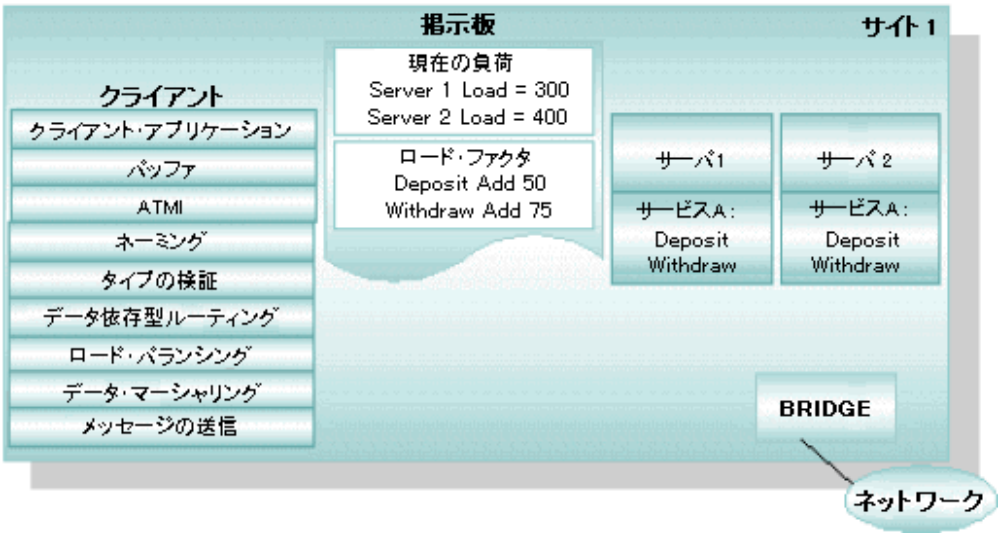
ロード・バランシングとは、同じサービスを提供するサーバ間でサービス要求を均等に分散する BEA Tuxedo システムの機能です。ロード・バランシングを使用すると、負荷の重いサーバがある一方で、アイドル状態または使用頻度の低いサーバが生じる状態を防ぐことができます。要求をサービス・ルーチンに送信する前に、その要求を処理できるすべてのサーバが BEA Tuxedo システムで識別され、コンフィギュレーション内のすべてのサーバで負荷を均衡化するために最も適したサーバが選択されます。

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、BEA Tuxedo システムでは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、または複数サーバ、単一キュー (MSSQ) 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

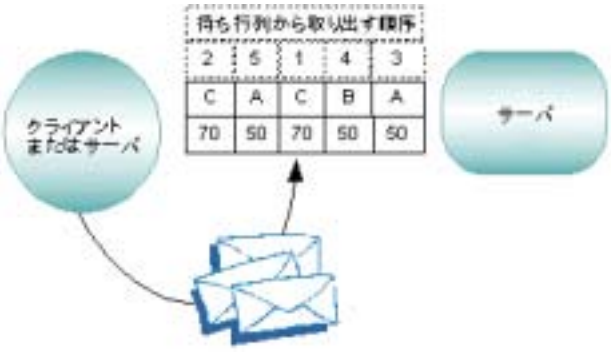


メッセージの優先順位付け

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 番目にあるものが先入れ先出し (FIFO) 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

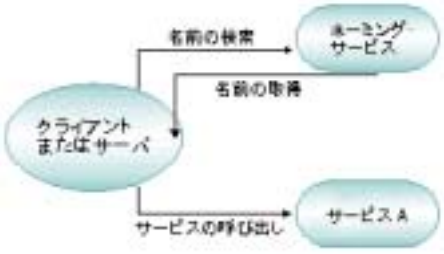
BEA Tuxedo システムでは、サービス名、メッセージ・キュー名、イベント名の 3 つのネーミング・デバイスが使用されています。名前には、任意の単語または英数字の文字列を使用できます。ただし、ピリオド「.」で始まる文字列は使用できません。管理サーバでも BEA Tuxedo システムの同じインフラストラクチャが使用されるため、システム・リソースとアプリケーション・リソースは明確に区別する必要があります。

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、掲示板によってそのサービスの名前が宣言されます。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



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

- 2-17 ページの「パブリッシュ・アンド・サブスクライブ通信」

3 BEA Tuxedo システムの管理とサーバ・プロセス

以下の節では、BEA Tuxedo システムに基づいて作成された ATMI アプリケーションのインフラストラクチャを共同で構成する中心的な BEA Tuxedo システム管理とサーバ・プロセスについて説明します。

- BEA Tuxedo ATMI のインフラストラクチャ
- BEA Tuxedo 管理プロセス
- BEA Tuxedo ワークステーション・サーバ
- BEA Tuxedo 認証サーバ
- BEA Tuxedo トランザクション管理サーバ
- BEA Tuxedo メッセージ・キュー・サーバ
- BEA Tuxedo パブリッシュ・アンド・サブスクライブ・サーバ
- BEA Tuxedo Domains (マルチ・ドメイン) サーバ
- 異なる BEA Tuxedo コンフィギュレーションから利用できるシステム・サービス

BEA Tuxedo ATMI のインフラストラクチャ

次のカテゴリの BEA Tuxedo システム・プロセスは、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、アプリケーション・キュー、および ATMI アプリケーションのイベントのポストと通知のためのインフラストラクチャを提供します。

- BEA Tuxedo 管理プロセス
- BEA Tuxedo ワークステーション・サーバ・プロセス
- BEA Tuxedo 認証サーバ・プロセス
- BEA Tuxedo トランザクション管理サーバ・プロセス
- BEA Tuxedo メッセージ・キューイング・サーバ・プロセス
- BEA Tuxedo パブリッシュ・アンド・サブスクライブ・サーバ・プロセス
- BEA Tuxedo Domains (マルチ・ドメイン) サーバ・プロセス

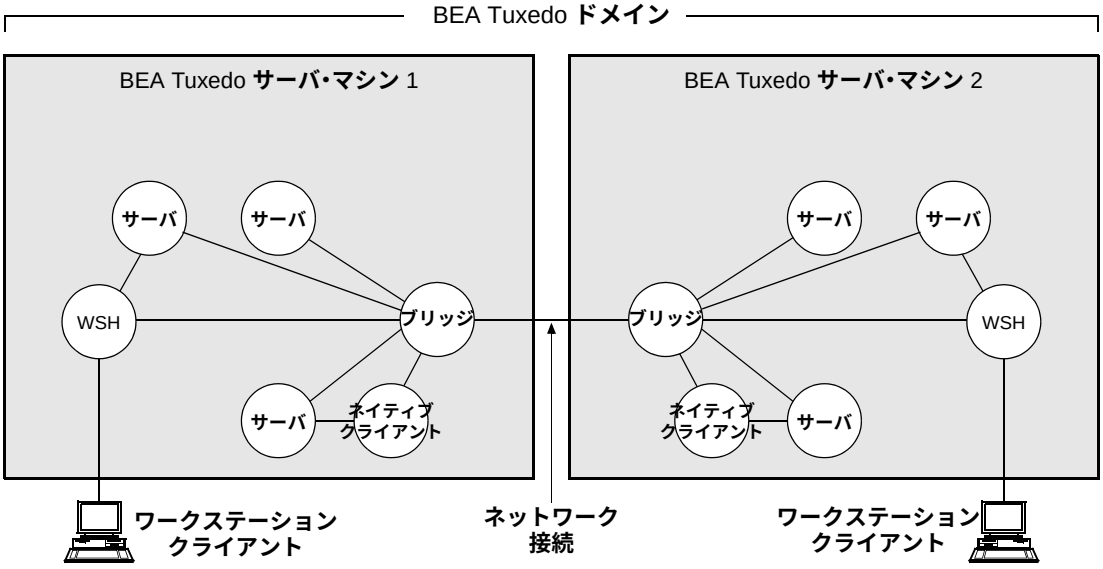
これらのカテゴリの BEA Tuxedo システム・プロセスを説明する前に、BEA Tuxedo に関する以下の重要な用語と概念を正しく理解しておく必要があります。

Tuxedo ドメイン

BEA Tuxedo ドメイン (BEA Tuxedo アプリケーション とも呼ばれる) は、1 つの Tuxedo コンフィギュレーション・ファイルから 1 つの単位として管理される Tuxedo システム・プロセス、クライアント・プロセス、およびサーバ・プロセスの集合です。Tuxedo ドメインは、多くのシステム・プロセス、1 つ以上のアプリケーション・クライアント・プロセス、1 つ以上のアプリケーション・サーバ・プロセス、およびネットワーク経由で接続された 1 つ以上のコンピュータ・マシンで構成されます。

次の図は、高いレベルから見た BEA Tuxedo ドメインです。

図 3-1 高いレベルから見た BEA Tuxedo ドメイン



BEA Tuxedo の用語で、ドメインはアプリケーション (ビジネス・アプリケーション) と同じ意味です。両方とも、BEA Tuxedo のマニュアル全体で同義で使用されます。現在 Tuxedo 上で実行されているビジネス・アプリケーションの例には、空港とホテルの予約システム、クレジット認可システム、株式売買システム、銀行業務システム、および現金自動預入支払機があります。

Tuxedo クライアント / サーバ・アプリケーションのクライアント側で動作するアプリケーション・プロセスは、通常はアプリケーション・クライアントまたは単にクライアントと呼ばれます。Tuxedo クライアント / サーバ・アプリケーションのサーバ側で動作するアプリケーション・プロセスは、通常はアプリケーション・サーバと呼ばれます。

注記　ドメインまたはアプリケーションという用語は、BEA Tuxedo クライアント / サーバ・アプリケーションのサーバ側ソフトウェアを表すのに使用される場合もあります。

Tuxedo コンフィギュレーション・ファイル

各 BEA Tuxedo ドメインは、インストールに依存するパラメータが定義されているコンフィギュレーション・ファイルによって制御されます。テキスト形式のコンフィギュレーション・ファイルは UBBCONFIG と呼ばれますが、コンフィギュレーション・ファイルには名前がある場合もあります。ただし、そのファイルの内容が『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5) で説明されている形式に準拠している場合に限ります。典型的なコンフィギュレーション・ファイル名は、文字列 ubb で始まり、その後にファイル名 ubbsimple の simple のようなニーモニック文字列が続きます。

Tuxedo ドメインの UBBCONFIG ファイルには、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格納されます。UBBCONFIG ファイルは 9 つのセクションで構成され、そのうちの 5 つ (RESOURCES、MACHINES、GROUPS、SERVERS、および SERVICES) はすべてのコンフィギュレーションで必須です。

バイナリ形式の UBBCONFIG ファイルは TUXCONFIG と呼ばれます。UBBCONFIG と同じように、TUXCONFIG ファイルにも任意の名前を付けることができます。その実際の名前は、TUXCONFIG 環境変数で指定されたデバイス・ファイル名またはシステム・ファイル名です。

Tuxedo のマスタ・マシン

BEA Tuxedo ドメインのマスタ・マシン (マスタ・ノード) はドメインの UBBCONFIG ファイルが格納されたサーバ・マシンであり、UBBCONFIG ファイルの RESOURCES セクションでマスタ・マシンとして指定されます。Tuxedo ドメインの 1 つ以上のサーバ・マシンの起動、停止、および管理は、マスタ・マシンを使用して行います。

Tuxedo ドメインのマスタ・マシンには、TUXCONFIG ファイルのマスタ・コピーも配置されます。TUXCONFIG ファイルは、Tuxedo システムがマスタ・マシンで起動されたときに Tuxedo ドメイン内の他のすべてのサーバ・マシン (非マスタ・マシン) に複製転送されます。

さまざまなリリースの BEA Tuxedo システム・ソフトウェアが動作するマルチ・マシン・ドメインでは、ドメイン内の最も新しいリリースの Tuxedo システム・ソフトウェアをマスタ・マシンで実行する必要があります。

Tuxedo TUXCONFIG 環境変数

TUXCONFIG 環境変数では、tmloadcf(1) コマンドがバイナリの TUXCONFIG ファイルをロードするマスタ・マシン上の場所を定義します。この環境変数は、TUXCONFIG がロードされるデバイス・ファイル名またはシステム・ファイル名で終わる絶対パス名に設定する必要があります。

TUXCONFIG のパス名値は、UBBCONFIG ファイルの MACHINES セクションで指定します。その値は、マスタ・マシンと Tuxedo ドメイン内の他のすべてのサーバ・マシンについて指定します。バイナリ TUXCONFIG ファイルのコピーがシステム起動時に非マスタ・マシンに転送されると、そのコピーは TUXCONFIG のパス名値に従って非マスタ・マシンに格納されます。

Tuxedo TUXDIR 環境変数

TUXDIR 環境変数では、マスタ・マシンでの BEA Tuxedo システム・ソフトウェアのインストール・ディレクトリを定義します。この環境変数は、インストール・ディレクトリの名前で終わる絶対パス名に設定する必要があります。

TUXDIR のパス名値は、UBBCONFIG ファイルの MACHINES セクションで指定します。その値は、マスタ・マシンと Tuxedo ドメイン内の他のすべてのサーバ・マシンについて指定します。

Tuxedo の掲示板

BEA Tuxedo システムでは、TUXCONFIG ファイルを使用して Tuxedo ドメインの各サーバ・マシンで掲示板 (BB) を設定します。Tuxedo サーバ・プロセスがアクティブになると、掲示板でそのサービスの名前が宣言されます。掲示

板の一部の情報はグローバルであり、Tuxedo ドメインの各サーバ・マシンで複製されます (特定のサービスを提供するすべてのサーバの名前と位置など)。他の情報はローカルであり、ローカルの掲示板でのみ表示されます (ローカルのサーバ要求キューで待機しているクライアント要求の実際の数とタイプなど)。

掲示板は、Tuxedo ドメイン内で位置と名前空間の透過性を実現します。位置の透過性とは、Tuxedo クライアント・プロセスとサーバ・プロセスが Tuxedo ドメイン内でリソースの位置を意識する必要がないことを意味します。名前空間の透過性とは、Tuxedo クライアント・プロセスとサーバ・プロセスが同じ命名規則 (および名前空間) で Tuxedo ドメイン内のリソースを検索できることを意味します。

関連項目

- 『BEA Tuxedo アプリケーションの設定』の 3-2 ページの「コンフィギュレーション・ファイルの作成方法」
- 『BEA Tuxedo アプリケーションの設定』の 8-1 ページの「分散型の ATMI アプリケーション用のコンフィギュレーション・ファイルの作成」

BEA Tuxedo 管理プロセス

BEA Tuxedo 管理プロセスは、分散アプリケーションの次のような管理タスクのほとんどを自動化します。

- 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

この説明では、BEA Tuxedo のシングル・マシンまたはマルチ・マシン・アプリケーション (ドメイン) で掲示板を設定および管理する管理プロセスにのみ焦点を当てます。

- 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・アプリケーション・クライアントが、同じサーバ・マシンに存在し、Tuxedo ドメインに属する 1 つ以上のアプリケーション・サーバと通信します。
- マルチ・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・アプリケーション・クライアントが、複数のサーバ・マシンに存在し、Tuxedo ドメインに属する 1 つ以上のアプリケーション・サーバと通信します。BEA Tuxedo ブリッジ・プロセスは、サーバ・マシン間でサービス要求を送受信し、要求をローカルのシステムまたはアプリケーション・サーバ・プロセスにルーティングします。

Tuxedo アプリケーションを起動、シャットダウン、および動的に再コンフィギュレーションするための管理プロセスについては、4-1 ページの「BEA Tuxedo の管理ツール」を参照してください。

掲示板の役割

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 BEA Tuxedo アプリケーション用に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。

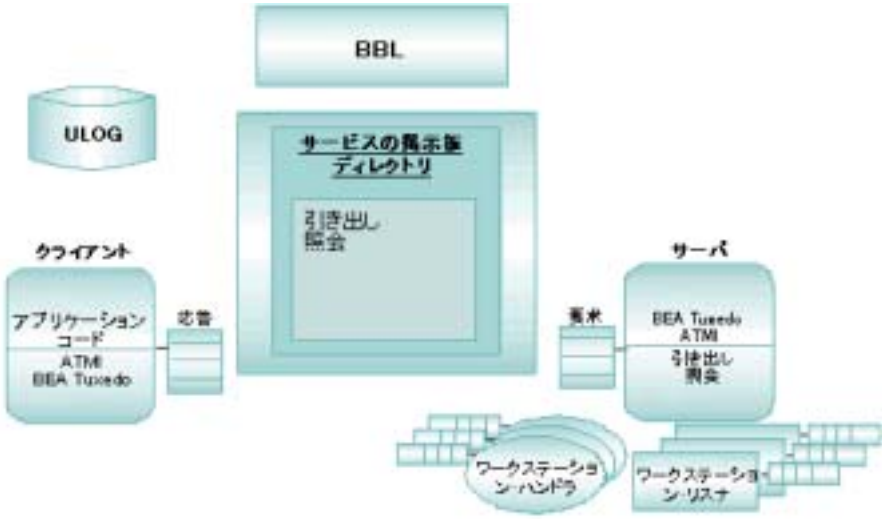
- サービス要求を特定のサーバに割り当てます。サービスが呼び出されると、掲示板は要求されたサービスを提供するサーバを検索します。この情報とデータ依存型ルーティング基準に基づいて、要求データを有効なサーバの要求キューに挿入します。
- サーバのキューで待ち状態にある要求の数や、処理済みの要求の数など、アプリケーションの状態に関する動的な情報を保持します。
- サーバの位置透過性を実現し、運用に依存しないアプリケーションの開発を可能にします。これにより、開発および運用のコストを最小限に抑えることができます。
- サーバ名のエイリアスをサポートし、同じサービスに複数の名前を割り当てることができるよう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

Tuxedo アプリケーションの各サーバ・マシンに掲示板があります。

BBL の役割

BBL は、Tuxedo アプリケーションの各サーバ・マシンで実行される BEA Tuxedo 管理プロセスであり、ローカル掲示板への変更を調整し、ローカル・マシンでアクティブになっているソフトウェア・プログラムの正常性を検証します。Tuxedo ドメインの各サーバ・マシン (マスタ・マシンを含む) では BBL が 1 つだけ実行されます。

図 3-2 掲示板と BBL



DBBL

DBBL は、複数のサーバ・マシン間でアプリケーションの分散を可能にする BEA Tuxedo 管理プロセスです。DBBL は、各サーバ・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。

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サーバ・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・マシン上のブリッジ・プロセス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) ブリッジを介してリモート・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・プロセスに送られます。

注記　Tuxedo シングル・マシン・アプリケーションでは、UBBCONFIG ファイルの RESOURCES セキュリティの MODEL パラメータの値によって DBBL プロセスが実行される場合とされない場合があります。MODEL=SHM の場合、DBBL プロセスは実行されません。MODEL=MP の場合、DBBL プロセスとブリッジ・プロセスが実行されます。DBBL の利点は、BBL の状態が定期的にチェックされ、終了している場合は再起動されることです。欠点は、DBBL とブリッジの 2 つのシステム・プロセスが追加的に実行されることです。

関連項目

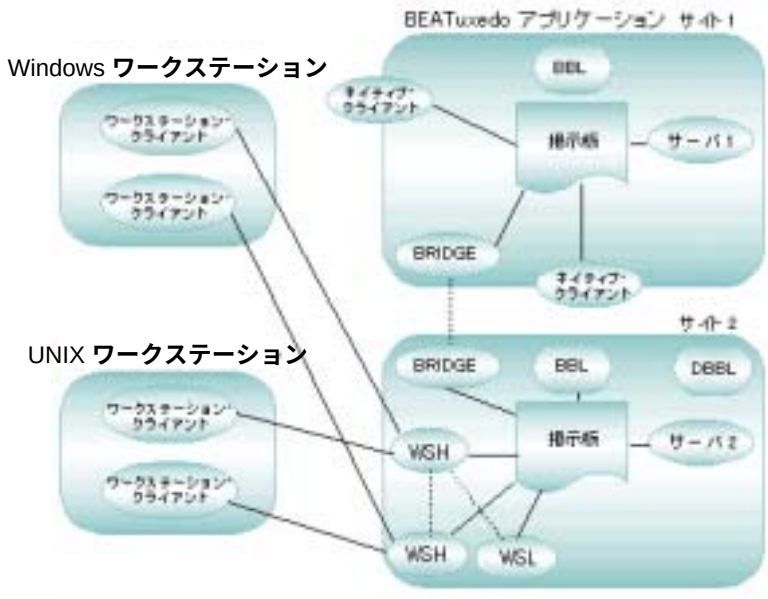
- 『BEA Tuxedo アプリケーションの設定』の 7-1 ページの「ネットワークへの ATMI アプリケーションの分散」
- 『BEA Tuxedo アプリケーションの設定』の 9-1 ページの「分散アプリケーションのネットワーク設定」
- 『BEA Tuxedo アプリケーション実行時の管理』の 4-1 ページの「分散アプリケーションでのネットワーク管理」

BEA Tuxedo ワークステーション・サーバ

BEA Tuxedo ワークステーション・サーバ・プロセスを使用すると、BEA Tuxedo サーバ側の機能がフル・インストールされていないリモート・マシン、つまり BEA Tuxedo 管理サーバも掲示板もサポートしないマシンに、ワークステーション・クライアント (リモート ATMI クライアント) を収容できます。ワークステーション・クライアントと BEA Tuxedo アプリケーション・サーバ間の通信は、すべてネットワーク経由で行われます。

ワークステーション・クライアントには、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。その情報は、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 BEA Tuxedo アプリケーションで動作するワークステーション・リスナ (WSL) およびワークステーション・ハンドラ (WSH) サーバ・プロセスのペアに送信できます。次の図は、WSL および WSH プロセスがワークステーション・クライアントをどのように BEA Tuxedo サーバ・アプリケーションに接続するのかを示しています。

図 3-3 ワークステーション・クライアントの処理



ワークステーション・リスナの役割

ワークステーション・リスナ (WSL) は BEA Tuxedo サーバ・マシンで実行される BEA Tuxedo リスニング・プロセスであり、ワークステーション・クライアントから接続要求を受け付け、同じようにサーバ・マシンで実行されるワークステーション・ハンドラに接続を割り当てます。ワークステーション・ハンドラ・プロセスのプールも管理し、要求されたロードに対応して起動と停止を行います。

管理者は、Tuxedo ドメイン内にいくつかの WSL を定義すると、複数のサーバ・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。

ワークステーション・ハンドラの役割

ワークステーション・ハンドラ (WSH) は BEA Tuxedo サーバ・マシンで実行される BEA Tuxedo ゲートウェイ・プロセスであり、ワークステーション・クライアントと BEA Tuxedo サーバ・アプリケーションの間で通信を処理します。WSH プロセスはアプリケーションの管理ドメインの中に位置し、ローカルの BEA Tuxedo 掲示板でクライアントとして登録されます。

各 WSH プロセスは、複数のワークステーション・クライアントを管理できます。WSH は、単一接続を介して特定のワークステーション・クライアントで処理されるすべての要求と応答を多重化します。

関連項目

- BEA Tuxedo Workstation コンポーネント
- 『BEA Tuxedo のセキュリティ機能』の「セキュリティ管理」
- 『BEA Tuxedo のファイル形式とデータ記述方法』の UBBCONFIG(5)、WS_MIB(5)、および WSL(5)

BEA Tuxedo 認証サーバ

BEA Tuxedo 認証サーバ (AUTHSRV) を使用すると、システム管理者はワークステーション・クライアントの認証と認可に必要な追加セキュリティを設定できます。AUTHSVR は、ユーザが適切な認証レベルを持っているかどうかを検証する単一のサービスを提供します。

管理者は、認証および認可のレベルを設定して BEA Tuxedo アプリケーションを設定できます。管理者は、AUTHSVR を除くすべてのサーバが、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。

アプリケーション設計者は、AUTHSVR の代わりに、そのアプリケーションに固有のロジックをインプリメントする認証サーバを使用できます。たとえば、広く使用されている Kerberos のメカニズムを使用して認証を行うため、認証サーバをカスタマイズすることもできます。

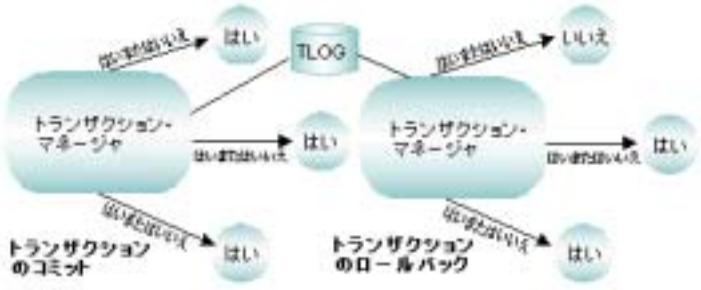
関連項目

- 『BEA Tuxedo のセキュリティ機能』の「セキュリティ管理」
- 『BEA Tuxedo のファイル形式とデータ記述方法』の AUTHSVR(5)

BEA Tuxedo トランザクション管理サーバ

BEA Tuxedo トランザクション管理サーバ (TMS) は、BEA Tuxedo ATMI アプリケーションのために、その起点 (通常はクライアント上) から 1 つ以上のサーバ・マシンをまたがり元のクライアントに戻るまでグローバル・トランザクションを調整します。TMS 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し、2 フェーズ・コミット・プロトコルを監視して、トランザクションのコミットまたはロールバックが各サイトで適切に処理されるようにします。

図 3-4 処理中のトランザクション・マネージャ・サーバ



操作の調整

実行されるすべての操作、およびトランザクションによって影響を受けるすべてのモジュールを調整するために、TMS は 1 つ以上のリソース・マネージャ (リレーショナル・データベース、階層データベース、ファイルシステム、ドキュメント・ストア、メッセージ・キュー、および他のバックエンド・サービスなど) のアクションを指示します。TMS とリソース・マネージャは共同でトランザクションの原子性を維持しますが、2 フェーズ・コミット・プロトコルとトランザクションの回復 (必要な場合) を実際に管理するのは TMS です。

トランザクション・ログによるパーティシパントのトラッキング

トランザクション・ログ (TLOG) で、TMS は 2 フェーズ・コミットの第 1 フェーズの終わりにグローバル・トランザクションのパーティシパントからすべて「Yes」応答を受け取った後にのみグローバル・トランザクションを記録します。TLOG の記録があることは、グローバル・トランザクションをコミットしなければならないことを示します。TLOG の記録がないことは、トランザクションをロールバックする必要があることを示します。Tuxedo ドメインの各サーバ・マシンにはそれ専用の TLOG が必要です。

関連項目

- 『BEA Tuxedo アプリケーションの設定』の 5-1 ページの「トランザクション対応の ATMI アプリケーションのコンフィギュレーション」
- 『サンプルを使用した BEA Tuxedo アプリケーションの開発方法』の 1-18 ページの「トランザクション」
- 『BEA Tuxedo のファイル形式とデータ記述方法』の `TM_MIB(5)`

BEA Tuxedo メッセージ・キュー・サーバ

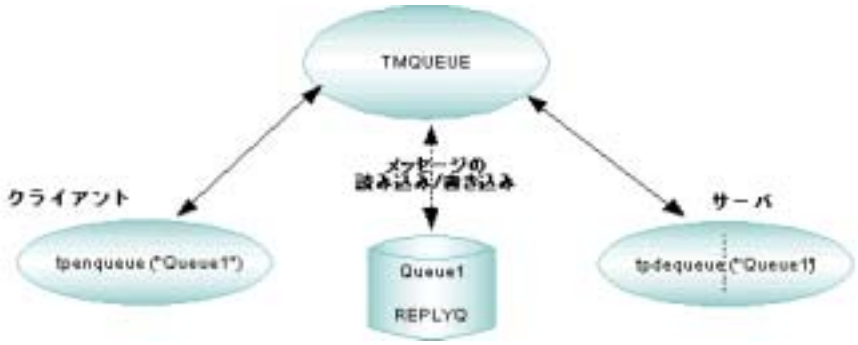
BEA Tuxedo メッセージ・キュー・サーバは、BEA Tuxedo ATMI アプリケーションのクライアントとサーバの間で時間に依存しない通信を実現します。メッセージ・キュー・サーバは、アプリケーションが、グローバル・トランザクション内で、クライアントおよびサーバが生成したメッセージを後で処理できるよう安定記憶域に格納できるようにします。メッセージ・キューイング通信に関与するクライアント・プロセスまたはサーバ・プロセスは、メッセージをいつそのキューから取得するかを決めます。

BEA Tuxedo メッセージ・キュー・サーバは、`TMQUEUE` という「メッセージ・キュー・マネージャ」サーバと `TMQFORWARD` という「メッセージ転送」サーバで構成されます。

TMQUEUE サーバの役割

`TMQUEUE` サーバは、クライアントおよびサーバの代わりにメッセージの格納 (登録) および取り出し (登録解除) を行います。次の図は、`TMQUEUE` の機能を示しています。

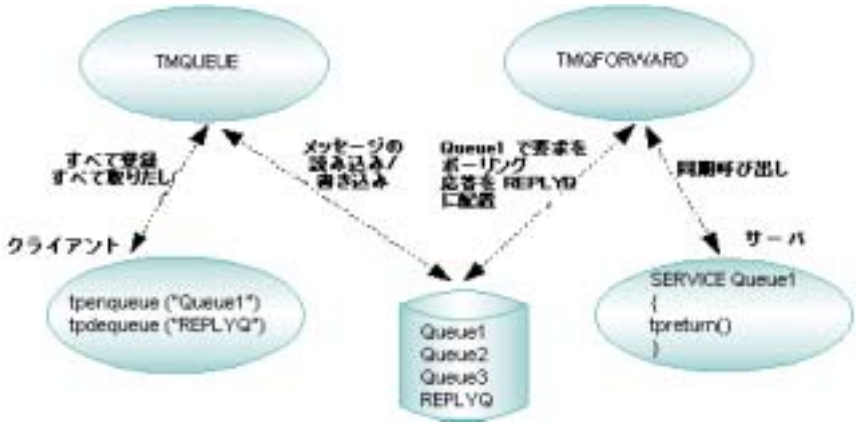
図 3-5TMQUEUE を使用したメッセージのキューへの登録



TMQFORWARD サーバの役割

TMQFORWARD サーバは、メッセージをキューから取り出し、適切なサーバで処理されるよう転送します。TMQFORWARD は、キューに入れられたメッセージがサービス呼び出しを必要とする場合のみ必要です。たとえばキューは、あるプロセスがメッセージをキューに入れて、別のプロセスがそれを削除するプロセス間通信に (BEA Tuxedo クライアントまたはサーバ上で) 使用される場合もあります。次の図は、TMQFORWARD の機能を示しています。

図 3-6TMQFORWARD を使用したメッセージの格納と転送



関連項目

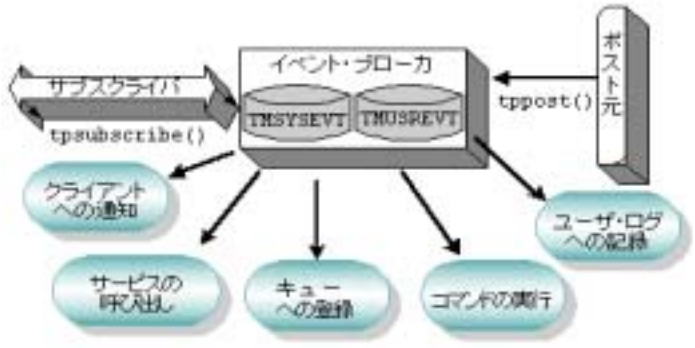
- 4-13 ページの「コマンド行ユーティリティを使用したアプリケーション・キューの管理」
- BEA Tuxedo /Q コンポーネント
- 『BEA Tuxedo C リファレンス』の `tpenqueue(3c)` および `tpdequeue(3c)`
- 『BEA Tuxedo のファイル形式とデータ記述方法』の `APPQ_MIB(5)`、`TMQUEUE(5)`、`TMQFORWARD(5)`、および `UBBCONFIG(5)`

BEA Tuxedo パブリッシュ・アンド・サブスクライブ・サーバ

BEA Tuxedo パブリッシュ・アンド・サブスクライブ・サーバは、BEA Tuxedo ATMI アプリケーションで動作するプロセス間のアプリケーション・イベントおよびシステム・イベントの非同期ルーティングを実現します。イベントとは、管理者、オペレータ、またはソフトウェアが関心の対象とする、アプリケーション・プログラムまたは BEA Tuxedo システムにおける状態の変化などのことです。イベントの例には、「株式が指値かそれ以上の値で取り引きされた」や「ネットワーク障害が発生した」などがあります。

BEA Tuxedo パブリッシュ・アンド・サブスクライブ・サーバは、TMUSREVT という「アプリケーション・イベント」サーバと TMSYSEVT という「システム・イベント」サーバで構成されます。TMUSREVT サーバはクライアントおよびサーバの代わりにアプリケーション・イベントを処理し、TMSYSEVT サーバはクライアントおよびサーバの代わりにシステム・イベントを処理します。次の図は、TMUSREVT および TMSYSEVT の機能を示しています。

図 3-7TMUSREVT および TMSYSEVT を使用したイベントの処理



関連項目

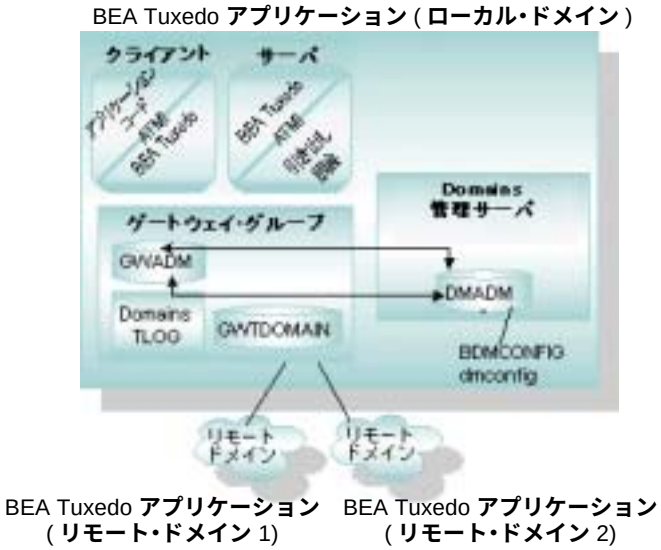
- 4-19 ページの「イベント・ブローカを使用したイベントの管理」
- 『BEA Tuxedo アプリケーション実行時の管理』の「イベント・ブローカについて」
- 『BEA Tuxedo C リファレンス』の tppost(3c)、tpsubscribe(3c)、および tpunsubscribe(3c)
- 『BEA Tuxedo のファイル形式とデータ記述方法』の EVENTS(5)、EVENT_MIB(5) に関する追加情報、TMSYSEVT(5)、TMUSREVT(5)、および UBBCONFIG(5)

BEA Tuxedo Domains (マルチ・ドメイン) サーバ

BEA Tuxedo Domains (マルチ・ドメイン) サーバ・プロセスは、BEA Tuxedo システムのクライアント / サーバ・モデルを拡張してトランザクション処理 (TP) ドメイン間にトランザクションの相互運用性を提供します。この拡張では、アプリケーション・プログラマとエンド・ユーザの両方がリモート・ドメインのサービスに透過的にアクセスできるようにすることで (またはリモート・ドメインからのサービス要求を受け付けることで) モデルおよび ATMI インターフェイスが維持されます。

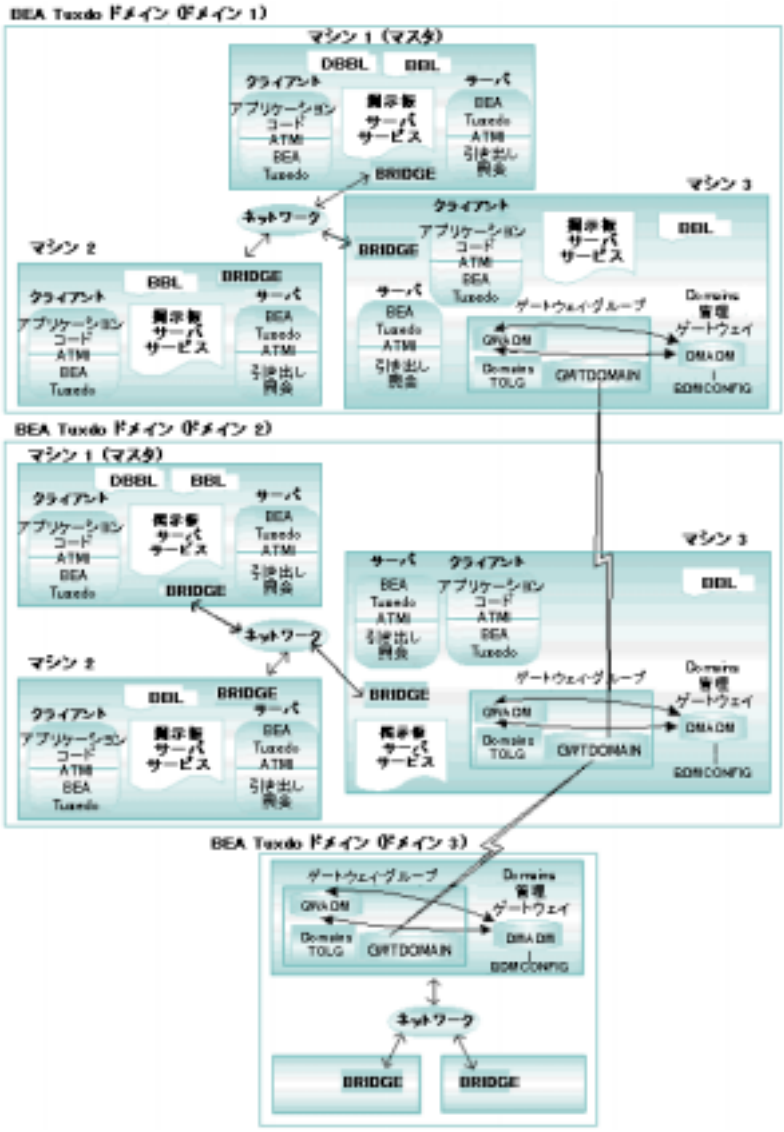
BEA Tuxedo Domains サーバ・プロセスは、DMADM という「Domains 管理」サーバ、GWADM という「ゲートウェイ管理」サーバ、およびいくつかある「ドメイン・ゲートウェイ」サーバ (GWTDOMAIN プロセスでインプリメントされる TDomain ゲートウェイ・サーバなど) のいずれかで構成されます。次の図は、DMADM、GWADM、および GWTDOMAIN の機能を示しています。

図 3-8TDomain ゲートウェイ・グループを使用したドメイン間通信



次の図は、Domains コンフィギュレーションのコネクティビティを示しています。

図 3-9Domains コンフィギュレーション



DMADM サーバの役割

DMADM サーバは、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、登録済みのゲートウェイ・グループのリストを管理し、Domains コンフィギュレーション・ファイル (BDMCONFIG) が変更されると、変更内容をリスト内のゲートウェイ・グループに伝播します。

複数のドメインがどのように接続されるか、およびどのサービスが別のドメインからアクセスできるようになるかは、Domains コンフィギュレーション・ファイルで定義します。Domains コンフィギュレーション・ファイルのテキスト形式は DMCONFIG、バイナリ形式は BDMCONFIG と呼ばれます。Domains コンフィギュレーションに関与する各 BEA Tuxedo ドメインでは、それ専用の Domains コンフィギュレーション・ファイルが必要です。

GWADM サーバの役割

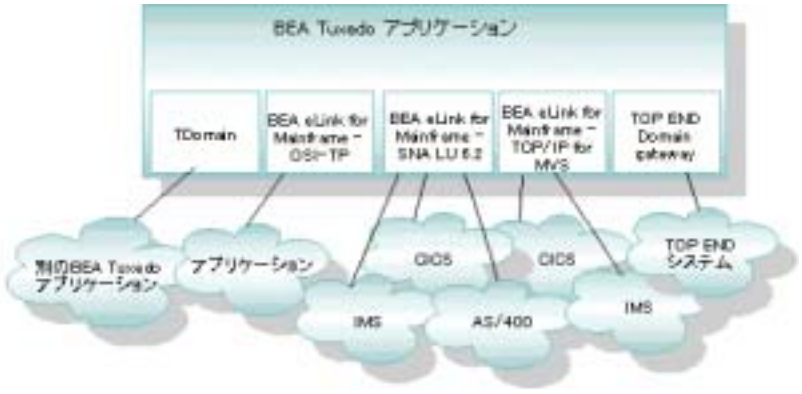
GWADM サーバは、DMADM サーバに登録して、対応するゲートウェイ・グループで使用されるコンフィギュレーション情報を取得します。GWADM は、DMADM からの要求、つまり、指定したゲートウェイ・グループの実行時オプションでの統計情報や変更に対する要求を受け付けます。

ドメイン・ゲートウェイ・サーバの役割

ドメイン・ゲートウェイは非同期性の高いマルチタスク型サーバ・プロセスであり、リモート・ドメインとの間で送受信されるサービス要求を処理します。ドメイン・ゲートウェイは、アプリケーション・プログラマとアプリケーション・ユーザが両方とも別のドメインのサービスに透過的にアクセスできるようにします。

次の図で示されているように、BEA Tuxedo システムは複数種のドメイン・ゲートウェイをサポートしています。そのため、BEA Tuxedo アプリケーションは他の BEA Tuxedo アプリケーションまたは他の TP システムで動作するアプリケーションと通信できます。

図 3-10 ドメイン・ゲートウェイの種類



関連項目

- 4-14 ページの「コマンド行ユーティリティを使用した Domains アプリケーションの管理」
- BEA Tuxedo Domains コンポーネント
- 『BEA Tuxedo のファイル形式とデータ記述方法』の `DMADM(5)`、`DMCONFIG(5)`、`GWTOPEND(5)` の DMCONFIG 、`GWADM(5)`、`GWTDOMAIN(5)`、および `UBBCONFIG(5)`
- BEA eLink Documentation

異なる BEA Tuxedo コンフィギュレーションから利用できるシステム・サービス

次の表は、BEA Tuxedo シングル・マシン・アプリケーション、マルチ・マシン (分散) アプリケーション、および Domains アプリケーションで利用可能な BEA Tuxedo システム・サービスのリストです。シングル・マシン・アプリケーションとマルチ・マシン・アプリケーションは、BEA Tuxedo ドメイン・コンフィギュレーションです。Domains アプリケーションは、TDomain (GWTDOMAIN) ゲートウェイを通じて互いに通信する複数の BEA Tuxedo ドメインで構成された BEA Tuxedo Domains コンフィギュレーションです。

表 3-1 異なるタイプの BEA Tuxedo ドメイン・コンフィギュレーションで利用できる機能

利用できる機能	シングル・マシン・アプリケーション	マルチ・マシン・アプリケーション	Domains アプリケーション
ATMI	X	X	X
メッセージング・パラダイム	X	X	X

表 3-1 異なるタイプの BEA Tuxedo ドメイン・コンフィギュレーションで利用できる機能

利用できる機能	シングル・マシン・アプリケーション	マルチ・マシン・アプリケーション	Domains アプリケーション
管理用の要素 :			
UBBCONFIG、TUXCONFIG、掲示板 (BB)、	X	X	X
BBL、DBBL、	X	X	X
ULOG、TLOG、	X	X	X
ブリッジ	表最後の注記を参照	X	X
管理プロセス :	X	X	X
tmloadcf、tmunloadcf、		X	X
tmboot、tmadmin など	X	X	X
BEA Tuxedo 管理プロセスの概要については、4-11 ページの「コマンド行ユーティリティを使用した操作の管理」を参照してください。	X	X	X
Domains の要素 :			
DMCONFIG、BDMCONFIG、			X
DMADM、GWADM、GWTDOMAIN、			X
DMTLOG			X
Domains 管理プロセス :			
dmloadcf、dmunloadcf、			X
dmadmin			X
BEA Tuxedo Domains 管理プロセスの概要については、4-14 ページの「コマンド行ユーティリティを使用した Domains アプリケーションの管理」を参照してください。			
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

表 3-1 異なるタイプの BEA Tuxedo ドメイン・コンフィギュレーションで利用できる機能

利用できる機能	シングル・マシン・アプリケーション	マルチ・マシン・アプリケーション	Domains アプリケーション
ワークステーション・クライアントの管理	X	X	X
セキュリティ管理	X	X	X
トランザクション管理	X	X	X
メッセージ・キューイング管理	X	X	X
イベント管理	X	X	

注記　Tuxedo シングル・マシン・アプリケーションでは、UBBCONFIG ファイルの RESOURCES セクションの MODEL パラメータの値によって DBBL プロセスが実行される場合とされない場合があります。MODEL=SHM の場合、DBBL プロセスは実行されません。MODEL=MP の場合、DBBL プロセスとブリッジ・プロセスが実行されます。DBBL の利点は、BBL の状態が定期的にチェックされ、終了している場合は再起動されることです。欠点は、DBBL とブリッジの 2 つのシステム・プロセスが追加的に実行されることです。

4 BEA Tuxedo の管理ツール

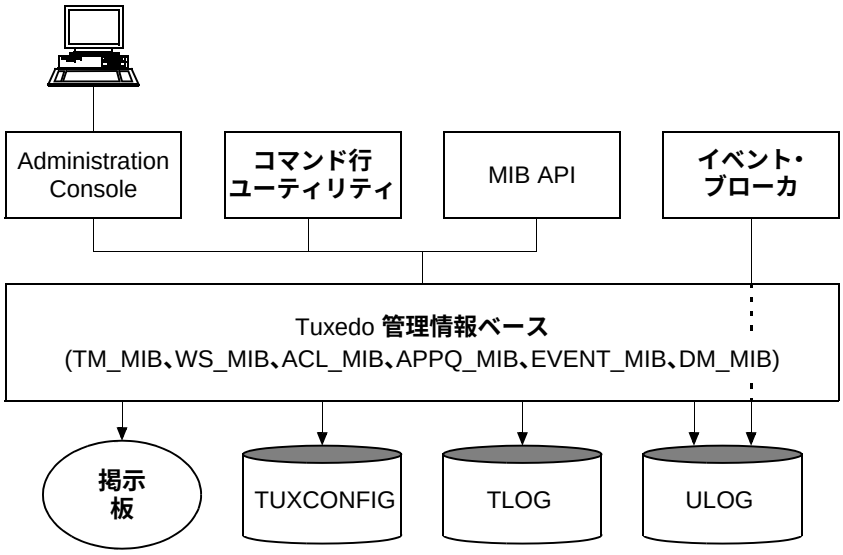
以下の節では、Tuxedo アプリケーションを管理するためにユーザが利用できる BEA Tuxedo 管理プロセスについて説明します。

- BEA Tuxedo ツールのアーキテクチャ
- BEA Tuxedo Administration Console を使用した管理操作
- BEA Tuxedo Administration Console のメイン・メニュー
- MIB を使用した操作の管理
- コマンド行ユーティリティを使用した操作の管理
- イベント・ブローカを使用したイベントの管理

BEA Tuxedo ツールのアーキテクチャ

次の図で示されているように、Tuxedo アプリケーションを管理するための BEA Tuxedo 管理プロセスでは、BEA Tuxedo 管理情報ベース (MIB) に基づいて作成されたさまざまなツールが包括的に利用されています。

図 4-1BEA Tuxedo アプリケーションを管理するためのツール



BEA Tuxedo の MIB には、Tuxedo 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。Tuxedo の MIB は、すべてのアプリケーションで共通の TM_MIB と、それぞれが BEA Tuxedo システムのサブシステムを記述する次のコンポーネント MIB で構成されます。

- 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。
- ACL_MIB －アクセス制御リスト (ACL) 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。
- 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。
- DM_MIB － Tuxedo Domains (マルチ・ドメイン) コンフィギュレーションを管理するために使用されます。

MIB リファレンス・ページ (`TM_MIB(5)`、汎用リファレンス・ページ `MIB(5)`) は、『BEA Tuxedo のファイル形式とデータ記述方法』で定義されています。

ツールの MIB とのインターフェイス

次のリストで簡単に説明されている BEA Tuxedo 管理ツールは、MIB との各種インターフェイスを提供します。

- BEA Tuxedo Administration Console ― Tuxedo 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 GUI アプリケーション。
- コマンド行ユーティリティ― Tuxedo アプリケーションの起動、終了、コンフィギュレーション、および管理に使用するコマンド。
- 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。
- イベント・ブローカ― Tuxedo アプリケーションで実行中のクライアント・プロセスとサーバ・プロセスの間でアプリケーション・イベントを非同期にルーティングし、システム・イベントを受信する必要のあるアプリケーション・プロセスにシステム・イベント (通常は障害や例外の発生) を配信する BEA Tuxedo コンポーネント。

他のシステム・コンポーネントとの MIB のインタフェース

MIB は、以下の BEA Tuxedo システム・コンポーネントにアクセスします。

- `TUXCONFIG` ファイル― Tuxedo アプリケーションのコンフィギュレーション (`UBBCONFIG`) ファイルのバイナリ版。Tuxedo アプリケーションのすべてのサーバ・マシンが、`TUXCONFIG` ファイルのコピーを格納します。MIB は `TUXCONFIG` ファイルを更新し、`TUXCONFIG` ファイルから情報を読み込みます。
- 掲示板― Tuxedo アプリケーションのすべての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。Tuxedo アプリケーションのすべてのサーバ・マシンが掲示板を持ちます。MIB は掲示板を更新し、掲示板から情報を読み込みます。

- ULOG ― Tuxedo システムおよびアプリケーションのメッセージ (エラー・メッセージ、警告メッセージ、情報メッセージ、およびデバッグ・メッセージ) が格納されるユーザ・ログ・ファイル。Tuxedo アプリケーションのすべてのサーバ・マシンで ULOG が必要です。MIB は、ULOG から情報を収集します。
- TLOG ―コミットされたグローバル・トランザクションの記録が格納されるトランザクション・ログ・ファイル。Tuxedo アプリケーションのすべてのサーバ・マシンで TLOG が必要です。MIB は、TLOG から情報を収集します。

BEA Tuxedo Administration Console を使用した管理操作

Java および Web 技術に基づく BEA Tuxedo 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 アプリケーションを操作できます。Administration Console は Java ベースのアプレットであり、Web ブラウザにダウンロードして Tuxedo 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。

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。

BEA Tuxedo Administration Console を使用する利点

- 認証― 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 Tuxedo 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 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。暗号化により、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のを防ぐことができます。
- ファイアウォールへの対応－ BEA Tuxedo 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－ 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 Tuxedo 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。Console サーバ・コンポーネントが存在する Tuxedo アプリケーション内のマシンの URL をブラウザで指定して、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 Tuxedo 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。

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 Tuxedo Administration Console でサポートされているブラウザについては、『BEA Tuxedo システムのインストール』の 7-1 ページの「BEA Tuxedo Administration Console の起動」を参照してください。

制限事項

現時点の BEA Tuxedo Administration Console では、BEA Tuxedo リリース 7.1 以後に導入された機能がサポートされていません。

関連項目

- BEA Tuxedo Administration Console オンライン・ヘルプ
- 4-6 ページの「BEA Tuxedo Administration Console のメイン・メニュー」
- 『BEA Tuxedo アプリケーション実行時の管理』の「アプリケーションの監視方法」
- 『BEA Tuxedo システムのインストール』の「BEA Tuxedo Administration Console の起動」

BEA Tuxedo Administration Console のメイン・メニュー

初めて Web にアクセスして BEA Tuxedo Administration Console を起動すると、次のメイン・ウィンドウが表示されます。

図 4-2BEA Tuxedo Administration Console のメイン・メニュー

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。Tuxedo アプリケーションに接続されていない場合、ツール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― BEA Tuxedo アプリケーショ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―タブ付きのページ。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。

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 アプリケーショ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 (Tuxedo アプリケーショ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。

アプリケーションを設定してアクティブにすると、アプリケーション内の管理クラス・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。

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、Tuxedo アプリケーションから構成されます。次のルートには、BEA Tuxedo TM_MIB 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 Tuxedo アプリケーションの一部です。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、アプリケーション内に、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ョクトなので、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	説明
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	Tuxedo アプリケーションの一部または全部をアクティブにします。

ボタン	説明
Deactivate	Tuxedo アプリケーショ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	アクティブな Tuxedo アプリケーショ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	Tuxedo アプリケーショ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	Administration Console のオンライン・ヘルプを別の Web ブラウザで開きます。

関連項目

- BEA Tuxedo Administration Console オンライン・ヘルプ
- 『BEA Tuxedo アプリケーション実行時の管理』の「アプリケーションの監視方法」

- 『BEA Tuxedo システムのインストール』の「BEA Tuxedo Administration Console の起動」

コマンド行ユーティリティを使用した操作の管理

BEA Tuxedo は、BEA Tuxedo システムに基づいて作成されたアプリケーションの各構成要素を管理するためのコマンドを提供します。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・キューの管理
- コマンド行ユーティリティを使用した Domains アプリケーションの管理

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

アプリケーションは、コマンド行ユーティリティを使用してコンフィギュレーションできます。具体的には、テキスト・エディタを使用してアプリケーションのコンフィギュレーション・ファイル (UBBCONFIG) を作成および編集し、tmloadcf というコマンド行ユーティリティを使用してテキスト・ファイル (UBBCONFIG) をバイナリ・ファイル (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。

次のリストは、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を示しています。

- tmloadcf(1)—マスタ・マシンで実行されるコマンドであり、アプリケーションの UBBCONFIG ファイルをバイナリの TUXCONFIG ファイルにコンパイルできるようにします。tmloadcf コマンドは、TUXCONFIG 環境変数で定義された場所にバイナリ・ファイルをロードします。
- tmunloadcf(1)—マスタ・マシンで実行されるコマンドであり、バイナリの TUXCONFIG ファイルを元のテキスト形式に変換して、UBBCONFIG ファイルと TUXCONFIG ファイルを同期できるようにします。tmunloadcf コマンドは、テキスト形式を標準出力に出力します。

  注記　バイナリの TUXCONFIG ファイルを動的に更新しても、テキストの UBBCONFIG ファイルは更新されません。

- tpusradd(1)、tpusrdel(1)、tpusrmod(1) —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。
- tpgrpadd(1)、tpgrpdel(1)、tpgrpmod(1) —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。
- tpacladd(1)、tpaclcvt(1)、tpacldel(1)、tpaclmod(1) —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。

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。

- tmboot(1) —マスタ・マシンで実行されるコマンドであり、アプリケーション・サーバを一元的に起動できるようにします。tmboot コマンドは TUXCONFIG 環境変数を読み込んで、アプリケーションの TUXCONFIG ファイルの位置を特定します。tmboot コマンドは、TUXCONFIG を共用メモリにロードして掲示板を確立します。変更内容は、マルチ・マシン・ドメインのリモート・サーバ・マシンに複製転送されます。

- tmadmin(1) －通常はマスタ・マシンで実行される対話型メタ・コマンドであり、アプリケーションをコンフィギュレーション、監視、およびチューニングするサブコマンドを実行できるようにします。tmadmin コマンドは、(コンフィギュレーション・モードで) アプリケーションが起動する前、またはアプリケーションの実行中に使用できます。
- tmconfig(1) －通常はマスタ・マシンで実行されるもう 1 つの対話型メタ・コマンドであり、アプリケーションをコンフィギュレーション、監視、およびチューニングするサブコマンドを実行できるようにします。tmconfig コマンドは、アプリケーションの実行中にのみ使用できます。tmconfig コマンドは tmadmin コマンドより効果的ですが、使いやすさは劣ります。
- tmshutdown(1) －マスタ・マシンで実行されるコマンドであり、アプリケーション・サーバを一元的にシャットダウンできるようにします。tmshutdown コマンドは、TUXCONFIG 環境変数を読み込んでアプリケーションの TUXCONFIG ファイルの位置を特定します。

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・キューの管理

コマンド行ユーティリティの qmadmin(1) を使用すると、アプリケーション・キューのすべての管理機能を実行できます。tmadmin コマンドや tmconfig コマンドと同じように、qmadmin は多くのサブコマンドを実行できるようにする対話型のメタ・コマンドです。

1 つの BEA Tuxedo 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、複数のサーバ・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、各サーバ・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。

コマンド行ユーティリティを使用した Domains アプリケーションの管理

BEA Tuxedo Domains (マルチ・ドメイン) アプリケーションを作成するには、既存の BEA Tuxedo アプリケーションを他のドメインと統合します。. そのためには、システム・サーバ (DMADM、GWADM、および GWTDOMAIN) のドメイン・ゲートウェイ・グループを UBBCONFIG ファイルに追加する必要があります。これらのサーバについては、3-19 ページの「BEA Tuxedo Domains (マルチ・ドメイン) サーバ」で説明します。

Domains コンフィギュレーションに関与する BEA Tuxedo アプリケーションのすべての Domains コンフィギュレーション情報は、DMCONFIG というファイルに格納されます。UBBCONFIG ファイルと同様に、DMCONFIG ファイルも任意の名前を持つことができます。ただし、そのファイルの内容が『BEA Tuxedo のファイル形式とデータ記述方法』のリファレンス・ページ DMCONFIG(5) で説明されている形式に準拠している場合に限ります。テキスト・エディタを使用して DMCONFIG ファイルを作成および編集し、dmloadcf というコマンド行ユーティリティを使用してテキスト・ファイル (DMCONFIG) をバイナリ・ファイル (BDMCONFIG) に変換します。BDMCONFIG ファイルは、DMADM サーバを実行するマシン上に配置する必要があります。

注記　DMADM サーバは、Tuxedo ドメインのどのマシン (マスタ・マシン、非マスタ・マシン) でも実行できます。

次のリストは、Domains コンフィギュレーションに関与する BEA Tuxedo アプリケーションのシステム・サーバのドメイン・ゲートウェイ・グループを設定および操作するために使用できるコマンド行ユーティリティを示しています。

- dmloadcf(1)―DMADM サーバと同じマシン上で実行されるコマンドであり、アプリケーションの DMCONFIG ファイルをバイナリの BDMCONFIG ファイルにコンパイルできるようにします。dmloadcf コマンドは、BDMCONFIG 環境変数で定義された場所にバイナリ・ファイルをロードします。
- dmunloadcf(1)―DMADM サーバと同じマシン上で実行されるコマンドであり、バイナリの BDMCONFIG ファイルを元のテキスト形式に変換して、

DMCONFIG ファイルと BDMCONFIG ファイルを同期できるようにします。dmunloadcf コマンドは、テキスト形式を標準出力に出力します。

注記　バイナリの BDMCONFIG ファイルを動的に更新しても、テキストの DMCONFIG ファイルは更新されません。

- dmadmin(1)―通常は DMADM サーバと同じマシン上で実行される対話型メタ・コマンドであり、ドメイン・ゲートウェイ・グループをコンフィギュレーション、監視、およびチューニングするサブコマンドを実行できるようにします。dmadmin コマンドは、(コンフィギュレーション・モードで) アプリケーションが起動する前、またはアプリケーションの実行中に使用できます。

関連項目

- BEA Tuxedo コマンド・リファレンス
- 『BEA Tuxedo のファイル形式とデータ記述方法』の DMADM(5)、DMCONFIG(5)、GWADM(5)、GWTDOMAIN(5)、および UBBCONFIG(5)
- 『BEA Tuxedo アプリケーション実行時の管理』の 2-13 ページの「コマンド行ユーティリティを使用してアプリケーションを監視する」
- 3-6 ページの「BEA Tuxedo 管理プロセス」
- 3-15 ページの「BEA Tuxedo メッセージ・キュー・サーバ」
- 3-19 ページの「BEA Tuxedo Domains (マルチ・ドメイン) サーバ」

MIB を使用した操作の管理

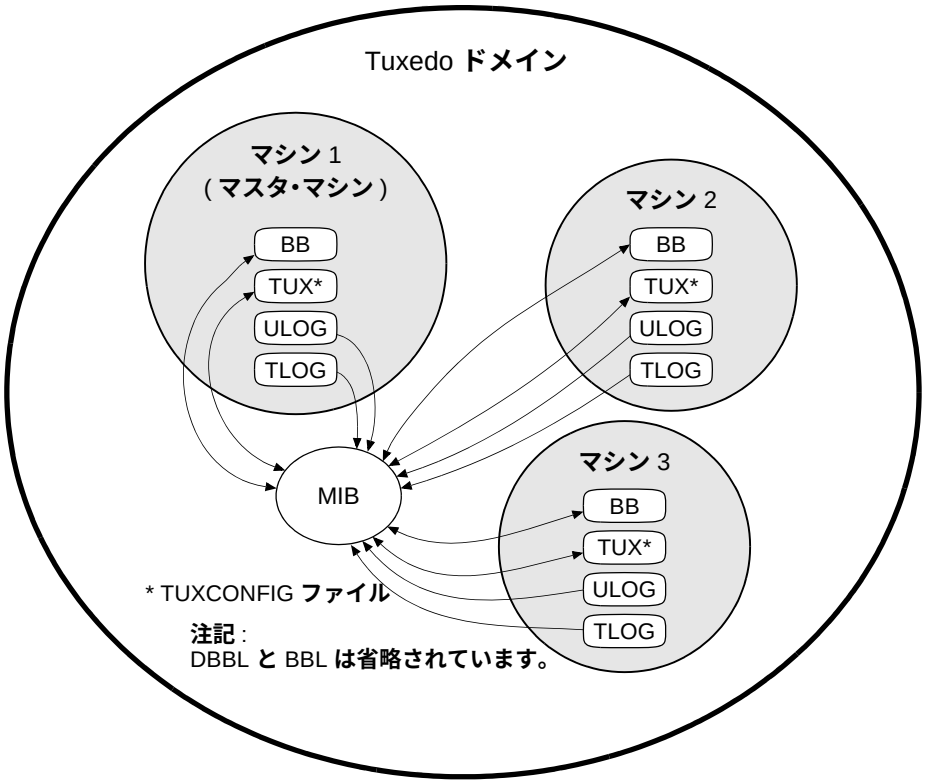
BEA Tuxedo MIB は、Tuxedo アプリケーションの管理に使用します。MIB では、すべての Tuxedo ドメインで必要とされるアプリケーションの構成要素を定義します。MIB では、Tuxedo アプリケーションがクラス (サーバ、

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。

Tuxedo サーバ・マシンがアクティブになると、そのサービスの名前が掲示板 (BB) で宣言されます。掲示板は、MIB の実行時 (動的) 表現です。掲示板は、MIB のグローバルおよびローカルの状態変化がポストされる場所です。Tuxedo システムでは、マスタ・マシン上にあるバイナリの TUXCONFIG ファイルを使用して掲示板を作成し、TUXCONFIG のコピーをアプリケーション内の非マスタ・マシンに転送してそれらのマシンで掲示板を設定します。掲示板は、Tuxedo アプリケーションの各サーバ・マシンで実行します。

次の図は、高いレベルから見た BEA Tuxedo MIB の操作です。

図 4-3 高いレベルから見た BEA Tuxedo MIB の操作



AdminAPI

AdminAPI は、BEA Tuxedo MIB のシステム設定に直接アクセスして操作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。

- フィールド操作言語 (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。

MIB ユーザのタイプ

MIB では、システム (またはアプリケーション) 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 (またはアプリケーション) 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。

ユーザのタイプ	説明
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(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。

関連項目

- 『BEA Tuxedo のファイル形式とデータ記述方法』の ACL_MIB(5)、APPQ_MIB(5)、DM_MIB(5)、EVENT_MIB(5) に関する追加情報、MIB(5)、TM_MIB(5)、および WS_MIB(5)
- FML を使用した BEA Tuxedo アプリケーションのプログラミング

イベント・ブローカを使用したイベントの管理

イベントとは、管理者、オペレータ、またはソフトウェアが関心の対象とする、アプリケーション・プログラムまたは BEA Tuxedo システムにおける状態の変化などのことです。イベントの例には、「株式が指値かそれ以上の値で取り引きされた」や「ネットワーク障害が発生した」などがあります。

BEA Tuxedo イベント・ブローカは、BEA Tuxedo ATMI アプリケーションで動作するプロセス間のアプリケーション・イベントおよびシステム・イベントの非同期ルーティングを実現します。アプリケーション・イベントはアプリケーション定義のイベントの発生、システム・イベントはシステム定義のイベントの発生を意味します。

アプリケーション定義のイベントとシステム定義のイベントの違い

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

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

アプリケーションのイベントの監視

次の表は、BEA Tuxedo アプリケーションでイベントを監視できるようにするための基本的な作業を示しています。

作業	説明
1. 監視するイベントを決める	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(3c) 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。 アプリケーション設計者は、どのイベントを監視するのかを決めます。システム・イベントの場合、アプリケーション設計者は EVENTS(5) リファレンス・ページからシステム定義のイベントを選択します。
2. イベント・リストを作成する	システム・イベントのリストが EVENTS(5) によって BEA Tuxedo システムから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。システム定義のイベントの名前はドット (.) で始まり、アプリケーション定義のイベントの名前はドット (.) で始まることはできません。 アプリケーション定義イベントのリストを用意する際、アプリケーション設計者はリファレンス・ページ EVENTS(5)、TMUSREVT(5)、TMSYSEVT(5)、および field_tables(5) を参照する必要があります。

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、アプリケーション定義イベントまたはシステム定義イベントの公開リストを使用し、tpsubscribe(3c) を呼び出してサブスクリプションを要求できます。EVENTS(5) は、システム・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(3c) 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。

システム定義イベントの各サブスクリプションは、いくつかある通知方法の 1 つを指定します。通知方法の 1 つはメッセージを ULOG に配置することです。EVENT_MIB の T_EVENT_USERLOG クラスを使用して、サブスクライバはシステム USERLOG メッセージを書き込むことができます。イベントが検出されて合致すると、これらのイベントは ULOG に書き込まれます。

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・イベントのポストには、イベントが発生したオブジェクトの現在の MIB 表現が含まれます。

関連項目

- 3-18 ページの「BEA Tuxedo パブリッシュ・アンド・サブスクライブ・サーバ」
- 『BEA Tuxedo アプリケーション実行時の管理』の「イベント・ブローカについて」
- 『BEA Tuxedo アプリケーション実行時の管理』の「イベントのサブスクライブ」
- 『BEA Tuxedo C リファレンス』の tppost(3c)、tpsubscribe(3c)、および tpunsubscribe(3c)
- 『BEA Tuxedo のファイル形式とデータ記述方法』の EVENTS(5)、EVENT_MIB(5) に関する追加情報、TMSYSEVT(5)、TMUSREVT(5)、および UBBCONFIG(5)
- 『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「イベント・ベースの通信」