BEA Tuxedo®

用語集

BEA Tuxedo 8.1
2003 年 1 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

用語集

A

アボート

データベースのレコードなど、保護されているすべてのリソースに割り当てられた値をトランザクション開始時から未変更のままトランザクションを終了すること。

絶対 OID

(CORBA) OID ツリーのルートから管理対象オブジェクトまでの一意なパスを指定するオブジェクト識別子 (OID)。

「オブジェクト識別子 (OID)」を参照。

抽象構文表記法 1 (ASN.1)

データ型を定義し、データ値を符号化するための正式な表記。抽象構文を構成するデータ構造を記述する言語。ITU-T (以前の CCITT) 仕様 X.409 は、ASN.1 と同じです。ASN.1 DER (Distinguished Encoding Rules) 形式のオブジェクト識別子は、公開鍵インターフェイス (PKI) のセキュリティ・プログラミングで使用されます。

アクセス制御リスト (ACL: Access Control List)

特定のサービスを受けたり、特定のオブジェクトおよびメソッドを呼び出したりすることを許可されたクライアントをリスト化し、このリストを使用してサービス、オブジェクト、およびメソッドに対するクライアント・アクセスを制御する BEA Tuxedo のセキュリティ機能。アクセス制御リストによるセキュリティ機能が有効な場合は、クライアントがサービスを要求するたびに適切なリストが参照され、サービスへのアクセス権が付与されているかどうかが確認されます。

Access Decision **オブジェクト**

(CORBA) ターゲット・オブジェクトに対する要求を転送する前に、アクセスが許可されているかどうかを確認する CORBA アプリケーションのセキュリティ・インフラストラクチャのオブジェクト。

アクセス・マシン

クライアントの最初のアクセス先であるアプリケーションの管理ドメイン内のプロセッサ。ネイティブ・クライアントは、このプロセッサ上で実行されます。ワークステーション・クライアントは、このサイトでアプリケーションと接続します。

ACID **特性**

次に示すトランザクション処理システムの主要な特性のこと。

- 原子性 (Atomicity)。トランザクションによるデータベースへのすべての変更内容は永続的になるか、またはすべて無効になります。
- 一貫性 (Consistency)。トランザクションが正常に処理されることにより、データベースは以前の有効な状態から別の有効な状態に変換されます。
- 独立性 (Isolation)。トランザクションによるデータベースの変更は、トランザクションが完了するまで、ほかのオペレーションからは見えません。
- 持続性 (Durability)。トランザクションによるデータベースの変更は、システムが変更されたり、メディアに故障が発生しても影響を受けません。

ACL

「アクセス制御リスト (ACL: Access Control List)」を参照。

活性化

(*ATMI*) サーバを非アクティブな (使用できない) 状態からアクティブな (実行中の) 状態にすること。

(CORBA) 実行できるようにオブジェクトを準備すること。

活性化方針

(CORBA) CORBA オブジェクトがメモリ内で活性化している期間を決定する方針。

参考項目 「活性化」、「 CORBA オブジェクト」、および「 方針」

活性化されたオブジェクト

(CORBA) オブジェクト・インターフェイスの実行中のインスタンス。

参考項目 「アクティブ・オブジェクト・マップ」、「 クライアント・アプリケーション」、「 CORBA オブジェクト」、「 オブジェクト ID (OID)」、「 オブジェクト・リファレンス」、「 ポータブル・オブジェクト・アダプタ (POA: Portable Object Adapter)」、および「 サーバント」

アクティブ・オブジェクト・マップ

(CORBA) オブジェクト ID とサーバントとの関連付けをマップし、POA および TP フレームワークによって保守されるテーブル。

参考項目 「オブジェクト ID (OID)」、「 ポータブル・オブジェクト・アダプタ (POA: Portable Object Adapter)」、および「 サーバント」

アクティブなサーバ

サービス要求を処理中のサーバ、またはサービス要求の処理準備が整った BEA Tuxedo サーバ。

ActiveX

(CORBA) ActiveX コントロールは、Component Object Model (COM) 技術を使用してほかの COM サービスおよびコンポーネントとの相互運用性を実現するコンポーネントです。

「Component Object Model (COM)」を参照。

ActiveX **クライアント**

(CORBA) CORBA ドメインと ActiveX オブジェクト・システムとの相互運用性を実現するコンポーネント。ActiveX Client は、ActiveX メソッドを、CORBA ドメインにある CORBA オブジェクトのインターフェイスに変換します。

ActiveX Client は、BEA Application Builder と Object Bridge という 2 つのコンポーネントで構成されています。

参考項目 「ActiveX」、「 BEA Application Builder」、「 ドメイン」、および「 オブジェクト・ブリッジ」

ADE

「アプリケーション開発環境 (ADE: Application Development Environment)」を参照。

AdminAPI

「管理用 API」を参照。

管理用 API

管理情報ベース (MIB) の属性値を設定および変更することで BEA Tuxedo アプリケーションをコンフィギュレーションおよび制御するプログラム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。ATMI と CORBA のどちらのプログラマも使用できます。

管理ドメイン

アプリケーション全体のうち、Bulletin Board Liaison (BBL) プロセスによって実行時に能動的に管理されるアプリケーション部分。ワークステーションおよびホスト・プロセッサは含まれません。

管理者

BEA Tuxedo システムのインストール、BEA Tuxedo アプリケーションのコンフィギュレーションと管理、およびアプリケーション情報 (コンピュータの名前や位置) の更新を扱う担当者。

宣言

あるサービスのサービス・テーブル・エントリが BEA Tuxedo の掲示板に用意されている場合、そのサービスは宣言されます。Domains のゲートウェイ・サーバが起動すると、ローカル・ドメイン、つまりゲートウェイ・サーバを起動したドメインの掲示板内にあるすべてのリモート・サービス (リモート・ドメインからインポートされたサービス) が宣言されます。ドメイン・ゲートウェイ・サーバによってリモート・サービスが宣言されると、`unadvertise` コマンドが実行されるか、または MIB 要求によるサービスの削除が行われない限り、リモート・サービスは宣言されたままになります。

AEQ

「アプリケーション構成要素修飾子 (AEQ: Application Entity Qualifier)」を参照。

AET

「アプリケーション構成要素タイトル (AET: Application Entity Title)」を参照。

エージェント

- ネットワーク管理ワークステーションのマネージャと管理対象オブジェクトに関するデータをやり取りするネットワーク管理システムのコンポーネント。マネージャの要求に応じて、エージェントは管理対象リソース用のソフトウェア・インターフェイスを提供したり、そのリソースに関するデータを収集したりします。
- 2 フェーズ・コミットの同期ポイント・シーケンス (LU6.2 または MRO) では、イニシエータ (同期ポイント・アクティビティを開始するタスク) からの同期ポイント要求を受け取るタスク。

「SNMP エージェント」を参照。

エージェント / マネージャ・モデル

マネージャがシステム管理プロトコルを介して分散された数多くのエージェントと通信するモデル。

アラーム

管理対象オブジェクトが異常な状態になった、つまり、管理対象オブジェクトが定義済みのしきい値を超えたことを報告する手段。

割り当て

さまざまな種類のプログラムやレコードのカテゴリをシステムの記憶域 (主記憶域やディスク記憶域など) に割り当てること。

代替装置

分散トランザクションのプログラミングでは、ALLOCATE コマンドによってトランザクションが取得するセッション。

代替リモート・ドメイン

プライマリ・リモート・ドメインを使用できない場合に、代わりに使用されるリモート・ドメイン。

AP

「アプリケーション・プログラム (AP: Application Program)」を参照。

API

「アプリケーション・プログラミング・インターフェイス (API: Application Programming Interface)」を参照。

アプレット

Java 対応ブラウザで表示した Web ページで実行されるインタラクティブな Java プログラム。アプレットを使用すると、より高度な Web ページの表示を実現でき、ユーザはタスクを実行することができます。

アプリケーション

BEA Tuxedo システムを中心として構築されるビジネス・プログラム。1 つの BEA Tuxedo コンフィギュレーション・ファイルを使用して定義および制御し、単一つのエンティティとして管理します。このようなアプリケーションは、1 つまたは複数のクライアント (ローカルまたはリモート)、1 つまたは複数のサーバ、および 1 台または複数台のマシンで構成されます。最小構成の BEA Tuxedo アプリケーションには、1 つのクライアント、1 つのサーバ、および 1 台のマシンが含まれます。BEA Tuxedo ドメインとも呼ばれます。

複数の BEA Tuxedo アプリケーションは、ドメイン・ゲートウェイ・グループを介して互いに通信できます。

注記　このコンテキストでは、ビジネス・プログラムという用語は、協調動作する 1 つまたは複数のプログラム群という意味になります。同様に、会話ではアプリケーションという用語はあいまいに使われることが多く、スタンドアロン・プログラムを指すことも、特定のビジネス目的を実現するために協調動作するプログラム群を指すこともあります。

「UBBCONFIG ファイル」および 「TUXCONFIG ファイル」を参照。

アプリケーションの関連性

プロセスと BEA Tuxedo アプリケーション (ドメイン) の間の関連性。マルチコンテキスト化されたプロセスは、複数の BEA Tuxedo ドメインと関連する場合があります。また、同じドメインと複数の関連性を持つこともあります。

アプリケーション・コード

BEA 社が提供したシステム・コードに対し、ユーザが記述したコード。

アプリケーション・コンテキスト

(ATMI) 特定のアプリケーション関連性に対する参照。BEA Tuxedo システムでは、明示的な呼び出しによりアプリケーション・コンテキストを設定し、以降の ATMI 呼び出しで暗黙的に使用します。したがって、暗黙的なコンテキスト・インターフェイスでは、「アプリケーション・コンテキスト」と「デフォルト・コンテキスト」は、しばしば同義語として扱われます。

アプリケーション・コンテキスト名

アプリケーション・エントリ間の関連性を規定するルールのセット。

アプリケーション制御の非活性化

(CORBA) アプリケーションが `TP::deactivateEnable()` オペレーションを呼び出してオブジェクトを明示的に非活性化するまでメモリ内のオブジェクトを活性化したままにしておく、というプロセスの活性化方針で使用される機能。

アプリケーション開発環境 (ADE: Application Development Environment)

プログラマ向けのアプリケーション作成用のツールのセット。多くの場合、GUI 形式で提供されます。

アプリケーション構成要素

1 つのコンピュータ・システム上の分散トランザクション処理用アプリケーションを構成するソフトウェア・コンポーネントのセット。

アプリケーション構成要素修飾子 (AEQ: Application Entity Qualifier)

OSI TP のアプリケーション構成要素を識別するための、ローカルで一意なコンポーネント名。

アプリケーション構成要素タイトル (AET: Application Entity Title)

OSI TP アプリケーション構成要素を識別するための、グローバルで一意なコンポーネント名。

アプリケーション・フレームワーク

アプリケーション・セットどうしやその他のソフトウェア・コンポーネントどうしが相互運用するためのインフラストラクチャを提供するソフトウェア。

アプリケーション・プログラム (AP: Application Program)

1 つまたは複数の特定のタスクを実行する、ユーザ・プログラムの単一のインスタンス。アプリケーション・プログラムは、トランザクションの境界を定義し、その境界内でリソースにアクセスします。つまり、X/Open 分散トランザクション処理モデルで指定されたインターフェイスを使用して、ほかのシステム・コンポーネントと相互作用します。アプリケーション・プログラムは、最大 1 つのグローバル・トランザクションに参加できる単一の制御スレッドです。

アプリケーション・プログラミング・インターフェイス (API: Application Programming Interface)

- 特定のシステム・ソフトウェア製品をサポートするための、アプリケーション・レベルの機能および環境。
- アプリケーション内でクライアント / サーバ型の要求を発行し、処理を実行するためのコードのセット。
- サービスの呼び出し方法を定義する呼び出し規約。ソフトウェア・プログラムが別のプログラムのサービスを使用するための、明確なプログラミング・インターフェイスのセット (エントリ・ポイント、呼び出しパラメータ、戻り値) です。

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)

BEA Tuxedo システム用のアプリケーション・プログラミング・インターフェイス。トランザクション・ルーチン、メッセージ処理ルーチン、サービス・インターフェイス・ルーチン、およびバッファ管理ルーチンが組み込まれています。

アーキテクチャ

- ハードウェアおよびソフトウェアのプラットフォーム (Solaris 上の SPARC、Windows NT 上の Intel Pentium、Linux 上の Intel Pentium など)。
- システム内または環境内でのコンポーネントの構造およびコンポーネント間の関係。

ASN.1

「抽象構文表記法 1 (ASN.1)」を参照。

非対称アルゴリズム

2 つの鍵 (公開鍵と秘密鍵) を使用する暗号アルゴリズム。公開鍵は広く配布し、秘密鍵は秘密にしておきます。非対称アルゴリズムでは、符号化、デジタル署名、および鍵合意を始めとして、さまざまなオペレーションを行えます。

非対称アウトバウンド IIOP

「アウトバウンド IIOP」を参照。

非同期

ほかのイベントが発生した時期とは無関係の時期に発生するイベント。この場合、2 つのイベントは相互に非同期的であると言えます。各イベントが発生する時期の関連性は予測できません。

非同期接続

互いに独立した状態、つまり非同期的に動作するよう設定された仮想回線。障害が発生した回線への再接続が行われている間も、稼動中の回線での処理はブロックされません。BRIDGE では、複数のネットワーク・アドレスの端点を使用してデータの受信や転送を行うことにより、障害のないネットワーク・パスを使用できます。

非同期プロセス

ほかのプロセスから独立した状態で実行されるプロセス。要求が非同期的に処理される場合、クライアント・アプリケーションはその要求の完了を待つ間も、別のオペレーションを実行できます。

非同期要求

アプリケーション内での並行処理を実現する要求。クライアントは、要求の処理中にほかの処理を行うことができます。

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 (ATMI: Application-to-Transaction Monitor Interface)」を参照。

基本セット

1 つまたは複数の変数を含む SNMP セット要求を受け取った場合に、要求されたすべてのオブジェクトを設定するか、またはまったく設定しない、という SNMP エージェントの動作。この動作は、SNMP 標準の要件です。

属性

(CORBA) オブジェクトと値との間で識別可能な関連性。

OMG IDL を使用する場合は、パブリック・クラス・フィールドまたはデータ・メンバに似た OMG IDL インターフェイスの該当部分のこと。コンパイラは、OMG IDL 属性を C++ または Java プログラミング言語のアクセサおよびモディファイア・メソッドにマップします。たとえば、インターフェイス `ball` に属性 `color` が含まれているとします。`idltojava` コンパイラは、`color` を取得する C++ または Java プログラミング言語のメソッドを作成し、属性が読み取り専用でない限り、`color` を設定するメソッドを作成します。CORBA の属性は、JavaBeans のプロパティに厳密に対応しています。

「オブジェクト」および「CORBA オブジェクト」を参照。

監査

要求されたシステム・オペレーションに対する応答レコードに要求元の ID を付け、安全で不正操作ができない状態でレコードを保存するセキュリティ・メカニズム。

監査追跡

レコードの内容に影響するトランザクションを追跡すること。手動または自動のいずれかで行うことができます。

認証

エンティティ (ユーザまたはプロセス) を識別するため、サーバ側で実行するプロセス。この処理は、ユーザまたはプロセスがアプリケーションに参加する前に行われます。このプロセスでは、パスワード機能やその他のセキュリティ・メカニズムが使用されます。

認可

アクセスが許可されているエンティティ (ユーザやプロセス) を判別し、そのエンティティにサービスへのアクセス権を付与するプロセス。

autoinstall

ログオン時に、端末の定義を動的に作成し、インストールするためのコマンド。ログオフ時には定義を削除します。

自動生成

BEA Tuxedo のアプリケーション・サーバが、受信メッセージを処理するための新しいスレッドを作成すること。新しいスレッドの数は、設定可能な値の上限値により制限されています。

可用性

障害が発生しても引き続きスムーズにシステムを稼動できる、というトランザクション処理システムの機能。

B

バックアップ

リソース・マネージャ側で行われる復元処理。ログ・エントリを順番にリソースに適用し、事前に指定しておいた状態が見つかったら、その状態にリソースを復元します。

帯域幅

コンピュータまたは通信チャネルの伝送容量。

BBL

「Bulletin Board Liaison (BBL)」を参照。

BEA ActiveX **クライアント**

「ActiveX クライアント」を参照。

BEA Administration Console

ATMI または CORBA 環境で動作している BEA Tuxedo アプリケーションをリモート管理するための Web ベースのグラフィカル・ユーザ・インターフェイス。インターネット・ブラウザにダウンロード可能な Java アプレットとして配布されます。

参考項目 「BEA Tuxedo システム」

BEA Application Builder

(CORBA) CORBA インターフェイス用の ActiveX バインディングを作成する機能。

参考項目 「ActiveX」、「 バインディング」、および「 会話」

BEA **のトランザクション処理**

「トランザクション処理 (TP: Transaction Processing)」を参照。

BEA Tuxedo-ASP Connectivity

(ATMI) 以前の「JoltWAS for IIS」。

BEA Tuxedo-JSE Connectivity

(ATMI) 以前の「JoltWAS for Servlet」。

BEA Tuxedo-WebLogic Connectivity

以前の「JoltWAS for WebLogic」。BEA Tuxedo-JSE Connectivity for WebLogic のカスタマイズ・バージョン。

BEA Tuxedo **アプリケーション**

「アプリケーション」を参照。

BEA Tuxedo **掲示板**

「掲示板」を参照。

BEA Tuxedo **クライアント**

「クライアント」を参照。

BEA Tuxedo **のドメイン**

BEA Tuxedo のドメインは、実行中のビジネス・アプリケーションが 1 つ以上ある BEA Tuxedo アプリケーションで構成されます。1 つのドメインは、1 つのコンフィギュレーション・ファイルで定義され、単一のエンティティとして管理されます。Domains 機能を使用して、ほかの BEA Tuxedo のドメインと接続することもできます。

参考項目 「ドメイン」、「 TUXCONFIG ファイル」、および「 UBBCONFIG ファイル」

BEA Tuxedo Domains

BEA Tuxedo システムのクライアント / サーバ型モデルを拡張して、TP ドメイン間でのトランザクションの相互運用性を実現する BEA Tuxedo のコンポーネント。この拡張機能では、リモート・ドメインのサービスへのアクセスや、リモート・ドメインからのサービス要求は、アプリケーション・プログラマやエンド・ユーザに対して透過的に行われるため、クライアント / サーバ型モデルと ATMI インターフェイスはそのまま利用できます。

BEA Tuxedo **サーバ**

クライアント・アプリケーションから要求されたタスクを実行するプログラム。

参考項目 「サーバ」

BEA Tuxedo **システム**

ビジネス・クリティカルなクライアント / サーバ型のアプリケーションを開発し、デプロイするための、BEA 社が開発した電子商取引用の強力なプラットフォーム。BEA Tuxedo システムでは、分散トランザクション処理、アプリケーションのメッセージ処理、および全社規模のアプリケーションを構築し、実行するのに必要なすべての補完的サービスを扱います。

BEA Wrapper Callbacks API

(CORBA) CORBA 共同クライアント / サーバ・アプリケーション用のコールバック・オブジェクトのインプリメンテーションを簡略化することを目的としたアプリケーション・プログラミング・インターフェイス。API には、コールバック・オブジェクトを定義、開始、停止、および破棄するためのメソッドが用意されています。

参考項目 「アプリケーション・プログラミング・インターフェイス (API: Application Programming Interface)」、「コールバック・ラッパー・オブジェクト」、「CORBA コールバック・オブジェクト」、および「共同クライアント / サーバ・アプリケーション」

双方向アウトバウンド IIOP

「アウトバウンド IIOP」を参照。

bind

(CORBA) アプリケーション・オブジェクトまたはネーミング・コンテキスト・オブジェクトに名前を関連付けるプロセス。クライアント・アプリケーションをアプリケーション・オブジェクトに結び付けるプロセスを表す場合にも使用します。

バインディング

(CORBA) BEA ActiveX Client では、CORBA オブジェクトのインターフェイスと別のオブジェクト・システム (ActiveX オブジェクト・システムなど) との関連性。

「BEA ActiveX クライアント」、「CORBA オブジェクト」、「オブジェクト」、および 「ActiveX」を参照。

bitmap

一時記憶域で、パーティション内の一時データが使用する制御ブロック。使用中および空き状態の VSAM 制御間隔を示します。制御間隔またはトラックが割り当てられるか、または解放されるたびに更新されます。

ブロッキング

2 つ以上のレコードを 1 つのブロックに結合すること。

ブロッキング・モード

メッセージの配信を同期的に行うこと。プログラムは、アクションが完了してから次の処理を行います。ブロッキング・モードの逆は、非ブロッキング・モードです。

Bootstrap **環境オブジェクト**

(CORBA) CORBA アプリケーションを BEA Tuxedo ドメインで起動し、初期オブジェクト・リファレンスをそのアプリケーションに提供するオブジェクト。BEA Tuxedo ドメインとやり取りするすべての CORBA クライアントまたはサーバには、Bootstrap 環境オブジェクトが必要です。

「オブジェクト」、「オブジェクト・リファレンス」、「CORBA ドメイン」、および 「環境オブジェクト」を参照。

ブートストラップ処理

(CORBA) BEA Tuxedo ドメインにある CORBA オブジェクトとやり取りするようアプリケーションを設定するプロセス。

参考項目 「Bootstrap 環境オブジェクト」、「 CORBA オブジェクト」、および「 CORBA ドメイン」

ブリッジ

アプリケーションに参加するノードどうしをつなぐ仮想回線を管理する BEA Tuxedo システムのプロセス。ノード間でアプリケーション・メッセージを転送するために使用します。

ブロードキャスト

ネットワーク上のすべてのノードに同じメッセージを送信すること。

ブローカ

サブスクリプションを管理し、イベントがポストされたときにサブスクライバ側のアクションを呼び出すシステム・レベルのエンティティ。

バッファ型

メッセージの型を表す抽象名。BEA Tuxedo には、FML、VIEW、STRING、CARRAY、XML という定義済みの 5 つのバッファ型が用意されています。これらのバッファ型は、異機種間ネットワークで透過的に符号化および復号化されます。アプリケーションでは、別のバッファ型を定義することもできます。

掲示板

実行中の BEA Tuxedo アプリケーションの状態をトラッキングするための共用データ構造体の集合。掲示板には、BEA Tuxedo アプリケーションに関わるサーバ、サービス、クライアント、およびトランザクションの情報が入っています。掲示板は、アプリケーション内のすべてのネイティブな (管理ドメイン内の) 論理マシン上に複製されます。

Bulletin Board Liaison (BBL)

特定のプロセッサ上にある掲示板のコピーを管理する BEA Tuxedo の管理プロセス。システムの実行中は、1 つの BBL プロセスが、アプリケーション内の各論理マシン上で継続的に動作します。

ビジネス・オブジェクト

(CORBA) 事前に定義できない組み合わせで使用できるアプリケーション・レベルのコンポーネント。ビジネス・オブジェクトは 1 つのアプリケーションに依存せず、文書プロセッサなど、一度は利用したことがあるような一般的なエンティティを表します。ビジネス・オブジェクトは独立して配布可能で、ユーザ・インターフェイスと状態を持ち、目的のタスクを実行するために別途開発したほかのビジネス・オブジェクトと協調動作できます。

参考項目 「オブジェクト」

バイト

隣接した 8 ビットの集合。1 つの基本単位として動作します。

C

C++

1980 年代初めに AT&T ベル研究所により開発された、オブジェクト指向型のプログラミング言語。C++ は、非オブジェクト指向言語である C 言語に基づいたハイブリッド型の言語です。

キャッシュ

頻繁にアクセスされるメモリ部分のコピーを格納しておくメモリのサブセット。

コールバック・メソッド

アプリケーション・コードで実装され、特定の関数を実行する必要がある場合にシステム・コードによって呼び出されるメソッド。コールバック・メソッドは、アプリケーション・コードで直接呼び出すものではありません。

参考項目 「アプリケーション・コード」および「 メタデータ・インターフェイス」

コールバック・ラッパー・オブジェクト

(CORBA) BEA Wrapper Callbacks API を使用して CORBA 共同クライアント / サーバ・アプリケーションでコールバックするために実装されるオブジェクト。

「コールバック・メソッド」、「CORBA コールバック・オブジェクト」、「共同クライアント / サーバ・アプリケーション」、「オブジェクト」、および 「BEA Wrapper Callbacks API」を参照。

CARRAY バッファ

文字配列型のデータ構造体。ヌル文字を含めることもできます。配列の解釈は、アプリケーションごとに異なります。

カタログ

「メッセージ・カタログ」を参照。

CCR

Commitment、Concurrency、Recovery (コミットメント、同時制御、回復制御) の略。OSI 標準です。

証明書

特定の公開鍵に名前などの属性を関連付けるデジタル・ステートメント。ステートメントには、認証局 (CA) によってデジタル署名が付けられます。認証局は正当なステートメントにのみ署名するという信頼に基づいて、公開鍵がその証明書に記載された本人のものであると見なすことができます。

関連項目「認証局 (CA)」

証明書による認証

デジタル証明書を使用することで、サーバが信頼性のあるクライアント ID を提供する方法。証明書ベースの認証は、パスワード・ベースの認証よりも広く利用されています。証明書ベースの認証では、ユーザが知っている情報 (秘密鍵を保護するパスワード) だけでなく、ユーザが持っている情報 (秘密鍵) も利用するからです。

「証明書」および 「認証」を参照。

認証局 (CA: Certificate Authority)

公開鍵の証明書を発行する、有名で信頼性のある機関。認証局は、公証人のようにユーザの実際のアイデンティティを証明します。

参考項目 「証明書」

チャネル

プロセッサとローカルの入出力デバイスとの間でデータの転送処理を行う、プロセッサ管理された機能の単位。

暗号

暗号化メッセージを作成するためのコーディング・システム。

暗号スイート

通信の整合性を保護するための鍵交換アルゴリズム、対称暗号アルゴリズム、および Secure Hash Algorithm からなる SSL 暗号化方式。

参考項目 「セキュア・ソケット・レイヤ (SSL: Secure Sockets Layer)」

暗号文

暗号化されたテキスト。

クラス

(CORBA) Java では、特定の種類のオブジェクトの実装を定義する型。クラス定義では、インスタンスとクラスの変数およびメソッドを定義し、インターフェイスとクラス・インプリメンテーション、およびクラスの直接のスーパー・クラスを指定します。スーパー・クラスを明示的に指定しないと、スーパー・クラスは暗黙的に `Object` となります。

参考項目 「IDL インターフェイス」、「 インスタンス」、「 Java」、「 メタデータ・インターフェイス」、および「 オブジェクト」

クラス・ライブラリ

クライアント・プログラミング用のツールのセット。これらのツールは、Java または C++ プログラム、あるいは Web ページに埋め込まれた Java アプレットで使用できます。

クライアント

(ATMI) 次の操作を実行するプログラム。

1. ユーザ・インターフェイスを介してユーザからサービスの要求を収集します。
2. その要求をサーバに転送します。
3. サーバから応答を受け取り、ユーザに渡します。

クライアントがターゲット・サーバの所属ドメイン内のマシン上にある場合は、クライアントをネイティブ・クライアントと呼びます。一方、クライアントがドメイン外のマシン上にある場合は、リモート・クライアントまたはワークステーション・クライアントと呼びます。リモート・クライアントは、BEA Tuxedo ワークステーション・コンポーネントを使用してサーバと通信します。

(CORBA) 分散オブジェクトのオペレーションを呼び出すコード。

参考項目 「アプリケーション」、「 サーバ」、および「 ワークステーション」

クライアント・アプリケーション

BEA Tuxedo ソフトウェア用に作成され、ほかのアプリケーションのサービスを要求するプログラム。

参考項目 「サーバ・アプリケーション」

Client Data Caching **デザイン・パターン**

(CORBA) サーバ・アプリケーションのデータをクライアント・アプリケーションのあるマシン上にキャッシュすることで、データを取得するためのリモート呼び出しの繰り返しを防ぎ、クライアント・アプリケーションの性能を向上するデザイン・パターン。

参考項目 「デザイン・パターン」.

クライアント命名

クライアント・プログラムにユーザ名とクライアント名の値を指定できる BEA Tuxedo の機能。

クライアント・プログラム

「クライアント」を参照。

クライアント / サーバ

「クライアント / サーバ型コンピューティング」を参照。

クライアント / サーバ型コンピューティング

分散処理を実現するため、アプリケーション・プログラムをクライアントまたはサーバとして構成したプログラミング・モデル。クライアント・プログラムとは、実行したいサービスを要求するアプリケーション・プログラムです。サーバ・プログラムとは、クライアント・プログラムから要求された内容を処理するためのサービス・ルーチンをディス

パッチするエンティティです。サービス・ルーチンとは、クライアント・プログラムの代わりに 1 つまたは複数の特定機能を実行するアプリケーション・プログラム・モジュールです。

クライアント / サーバ型コンピューティングには、2 層構成と 3 層構成があります。2 層構成とは、クライアントとサーバのみで構成されたコンフィギュレーションのことです。3 層構成とは、クライアント、サーバ、および中間レベル (ルータまたはブローカとして動作) で構成されるコンフィギュレーションのことです。

クローズド・フレームワーク

開発者がプラグ・アンド・プレイ形式で簡単にソフトウェア・コンポーネントを削除したり、置換したりできないソフトウェア・インフラストラクチャ。

CMIP

「Common Management Interface Protocol (CMIP)」を参照。

列オブジェクト

ゼロ以上のインスタンスを持つ MIB リーフ・オブジェクト (OID ツリー内に下位のオブジェクトを持たない MIB オブジェクト)。列オブジェクトは、テーブル内の 1 つの列を表します。

COM

「Component Object Model (COM)」を参照。

COM ビュー

(CORBA) 必要なすべてのインターフェイスのインプリメンテーションを含む、Component Object Model (COM) 標準に準拠したオブジェクトの表現。

参考項目 「Component Object Model (COM)」、「 インターフェイス」、および「 オブジェクト」

コマンド行インターフェイス

システム・プロンプトでのコマンド入力より、ユーザとの対話を実現するユーザ・インターフェイス。

コミット

- データベースへの変更を記録し、安定化させてトランザクションを終了すること。保護されていたリソースは解放されます。
- トランザクションで実行された更新処理やメッセージ処理の宣言、またはそのプロセス自体を、ほかのトランザクションからも見えるようにすること。トランザクションがコミットされると、そのトランザクションが行った処理は公開され、保持されます。コミット後は、トランザクションが行った処理を自動的に元に戻すことはできません。

Common Management Interface Protocol (CMIP)

ISO 標準で定義されているネットワーク管理用のプロトコル。

Common Object Request Broker Architecture

「CORBA」を参照。

CD-ROM (Compact Disc-Read Only Memory)

読み取り専用データを格納したディスク。データはレーザーで読み取られます。特別な条件を満たさない限り、データを変更することはできません。

コンポーネント

アプリケーションの一部。

Component Object Model (COM)

(CORBA) ソフトウェア・コンポーネントをネットワーク環境で相互運用するためのサービス群。

参考項目 「COM ビュー」、「 会話」、および「 オブジェクト」

並行性

複数の関数またはプロセスを同時に実行すること。

並行の

指定された期間内に、2 つ以上のアクティビティが発生すること。並行して実行されているプロセスは、共通の共用リソースを交互に使用することができます。

コンフィギュレーション

コンピュータまたはネットワークにある、ハードウェア、ハードウェア・オプション、ソフトウェア、およびソフトウェア・セットアップのセット。

コンフィギュレーション・セット

コンフィギュレーション・パーティションの特定のコンフィギュレーションを示す名前または番号。各コンフィギュレーション・セットでは、コンフィギュレーションがアクティブな場合に使用されるサービスが記述されています。

コンフィギュレーションする

コンピュータまたはネットワーク用にハードウェアやソフトウェアをカスタマイズすること。

接続

プロセス間をつなぐ半二重通信チャネル。

接続指向型の通信

接続を介した BEA Tuxedo システムの 2 つのプロセス間での通信。

整合性の取れた状態

共用データの内容が正しく、有効な状態であること。

コンストラクタ

(CORBA) オブジェクトを作成する疑似メソッド。Java では、コンストラクタはクラスと同じ名前を持つインスタンス・メソッドです。Java コンストラクタを呼び出すには、new キーワードを使用します。

参考項目 「クラス」、「 インスタンス」、「 Java」、「 メタデータ・インターフェイス」、および「 オブジェクト」

会話

接続を介して行われる会話。

会話型

サービスの要求元とサーバ間で 1 つまたは複数のメッセージが送受信されるときに、送受信処理が終了するまでサーバがその通信にのみ対応するような通信。

会話型通信

「会話型」を参照。

会話型サーバ

サービスの要求元との接続を会話形式で行うサーバ。会話の形式は、アプリケーションで規定されたプロトコルに従います。会話型サービスは、BEA Tuxedo システムのサービスに適用される起動と終了の規則に準拠する必要があります。

会話型サービス

クライアント・プログラムからの会話型通信によって呼び出されるサービス・ルーチン。接続が確立され、サービスが呼び出されると、クライアントとサービスは、アプリケーション固有の形式でデータを交換します。サービスが返されると、接続は切断されます。

CORBA

(CORBA) Common Object Request Broker Architecture の略。Object Management Group によって発表された分散オブジェクト指向型コンピューティング用のマルチベンダ標準。

CORBA **コールバック・オブジェクト**

(CORBA) ターゲット・オブジェクトに対するクライアント・アプリケーションの呼び出しでパラメータとして提供される CORBA オブジェクト。ターゲット・オブジェクトは、ターゲット・オブジェクトの実行時、またはターゲット・オブジェクトの呼び出しが終了した後に、コールバック・オブジェクトに対する呼び出しを行うことができます。コールバック・オブジェクトは、BEA Tuxedo ドメインの内側と外側のどちらにあってもかまいません。

参考項目 「クライアント・アプリケーション」、「 CORBA オブジェクト」、および「 BEA Tuxedo のドメイン」

CORBA **クライアント・スタブ**

CORBA オブジェクトを使用する場合は、アプリケーションの OMG IDL 文をコンパイルすると IDL コンパイラによって作成されるファイル。クライアント・スタブには、クライアント・アプリケーションのビルド・プロセスで作成されたコードが格納されています。クライアント・スタブは、要求を呼び出すときに、オブジェクト型に対応する

OMG IDL オペレーションの定義を BEA Tuxedo ドメインが呼び出すサーバ・アプリケーションのメソッドにマップします。コードは、要求を CORBA サーバ・アプリケーションに送信する場合に使用されます。

OMG IDL を使用する場合は、オブジェクトの呼び出し時にコンパイラによって作成され、クライアント ORB によって透過的に使用される C++ または Java プログラミング言語のクラス。クライアントによって保持されるリモート・オブジェクト・リファレンスは、クライアント・スタブを指します。このスタブは作成元の IDL インターフェイスに固有で、クライアントが IDL に定義されている CORBA オブジェクトのメソッドを呼び出すために必要な情報を格納しています。

参考項目 「メタデータ・インターフェイス」、「 OMG IDL」、「 スケルトン」、および「 BEA Tuxedo のドメイン」

CORBA **ドメイン**

1 つの `UBBCONFIG` (ASCII バージョン) または `TUXCONFIG` (バイナリ・バージョン) コンフィギュレーションによって定義された、CORBA または ATMI サーバ、サービス、インターフェイス、および関連リソース・マネージャの集合。

参考項目 「TUXCONFIG ファイル」および「 UBBCONFIG ファイル」

CORBA **外部クライアント・アプリケーション**

(*CORBA*) ORB に実装された BEA 社製品以外のクライアント・アプリケーション。CORBA ソフトウェアの ActiveX Client コンポーネントは外部クライアント・アプリケーションではありません。このクライアントは Microsoft 製品に実装されていますが、ORB は BEA 社によって提供されます。

「 ORB」および 「BEA ActiveX クライアント」を参照。

CORBA **インターフェイス**

(CORBA) オペレーションおよび属性のセット。CORBA インターフェイスは、OMG IDL 文を使用してインターフェイス定義を作成することで定義されます。定義には、オブジェクトを操作するためのオペレーションおよび属性が格納されます。

参考項目 「属性」、「 インターフェイス」、「 オブジェクト」、「 OMG IDL」、および「 オペレーション」

CORBA ネイティブ・クライアント・アプリケーション

(CORBA) CORBA サーバ・アプリケーションと通信するために OMG IDL 文に定義されたオペレーションを呼び出すクライアント・アプリケーション。サーバ・アプリケーションが属する CORBA ドメインを基準として、クライアント・アプリケーションはネイティブ (つまりローカル) またはリモートとなります。リモート・クライアント・アプリケーションとネイティブ・クライアント・アプリケーションは同じです。アプリケーションが CORBA ドメインで実行中のマシン上にあるかどうかに応じて、要求の処理方法は異なり、透過的に処理されるかどうかも変わります。CORBA ネイティブ・クライアント・アプリケーションは、必ず CORBA ドメイン内のマシン上にあります。

参考項目 「CORBA ドメイン」、「 CORBA 外部クライアント・アプリケーション」、「 CORBA リモート・クライアント・アプリケーション」、「 CORBA サーバ・アプリケーション」、および「 OMG IDL」

CORBA オブジェクト

(CORBA) オペレーションを実行する CORBA 標準に準拠したエンティティ。オブジェクトはインターフェイスによって定義されます。

参考項目 「インターフェイス」、「 オブジェクト」、および「 オペレーション」

CORBA ORB

(CORBA) CORBA 標準に準拠したオブジェクト・リクエスト・ブローカ (ORB)。CORBA ORB は、ネットワーク全体に分散されているクライアント・アプリケーションとサーバ・アプリケーションとの通信の媒介役です。BEA Tuxedo アプリケーションで使用される ORB は CORBA ORB です。

参考項目 「会話」

CORBA リモート・クライアント・アプリケーション

(CORBA) IIOP を使用してリモート CORBA サーバ・アプリケーションと通信するために OMG IDL 文に定義されたオペレーションを呼び出すクライアント・アプリケーション。リモート・クライアント・アプリケーションとネイティブ・クライアント・アプリケーションは同じです。アプリケーションが CORBA ドメインで実行中のマシン上にあるかどうかに応じて、要求の処理方法は異なり、透過的に処理されるかどう

かも変わります。通常、CORBA リモート・クライアント・アプリケーションは、CORBA ドメインで実行中のマシン上にありません。たとえば、CORBA ソフトウェアの ActiveX Client コンポーネントはリモート・クライアント・アプリケーションです。

「IIOP」、「OMG IDL」、「CORBA ドメイン」、「CORBA 外部クライアント・アプリケーション」、「CORBA ネイティブ・クライアント・アプリケーション」、および 「BEA ActiveX クライアント」を参照。

CORBA **サーバ・アプリケーション**

(CORBA) クライアント・アプリケーションに要求されたタスクを実行し、BEA Tuxedo CORBA ソフトウェアで使用するために作成されたプログラム。

「ローカル・ファクトリ」を参照。

CORBA TP **フレームワーク**

(CORBA) CORBA サーバ・アプリケーションのビルド手順がサーバ・アプリケーションの実行可能イメージとリンクするデフォルト・インプリメンテーションのランタイム・ライブラリ。トランザクション処理 (TP) フレームワークは、以下の処理を行うプログラムを簡単に記述するための便利な関数セットで構成されています。

サーバ・アプリケーションを初期化し、起動ルーチンとシャットダウン・ルーチンを実行します。

サーバ・アプリケーションを CORBA ドメインのリソースに関連付けます。

オブジェクトを管理します。管理には、オブジェクトを必要に応じてメモリ内に呼び出す、不要になったらメモリからクリアする、永続オブジェクトのデータの読み取りおよび書き込みを管理する、という処理があります。

ハウスキーピング機能を実行します。

「CORBA サーバ・アプリケーション」および 「CORBA ドメイン」を参照。

CORBA **ファシリティ**

(CORBA) OMG が採用した共通ファシリティ。共通ファシリティは、ほとんどのアプリケーションに適用可能なエンド・ユーザ指向の水平型フレームワークを提供し、OMG IDL で定義されます。

参考項目 「OMG IDL」

CORBA サービス

(CORBA) プログラマ向けに開発されたオブジェクトのシステム・サービス群。これらのサービスは OMG の OMG IDL で定義されます。オブジェクトの作成、オブジェクトへのアクセスの制御、オブジェクトとオブジェクト・リファレンスのトラッキング、およびオブジェクトの型どうしの関係の制御に使用できます。プログラマはオブジェクト・サービス関数を呼び出すことで、独自のオブジェクト・サービス関数を記述して呼び出す必要がなくなります。

参考項目 「CORBA オブジェクト」、「 CORBA ライフサイクル・サービス」、「 CORBA ネーミング・サービス」、「 CORBA オブジェクト・トランザクション・サービス (OTS: Object Transaction Service)」、「 CORBA セキュリティ・サービス」、「 オブジェクト」、「 オブジェクト・リファレンス」、および「 OMG IDL」

CORBA ライフサイクル・サービス

(CORBA) オブジェクトの作成、削除、コピー、および移動に関する規則を定義する CORBA サービス。

参考項目 「CORBA サービス」および「 オブジェクト」

CORBA ネーミング・サービス

(CORBA) ネーミング・コンテキストに関してオブジェクトに名前を関連付ける機能を提供する CORBA サービス。

参考項目 「CORBA サービス」および「 オブジェクト」

CORBA オブジェクト・トランザクション・サービス (OTS: Object Transaction Service)

(CORBA) システム内のデータの整合性を保証するためのトランザクション・セマンティクスを提供する CORBA サービス。

参考項目 「CORBA サービス」

CORBA セキュリティ・サービス

(CORBA) プリンシパルの ID および認証、認可とアクセス制御、セキュリティ監査、オブジェクト間の通信のセキュリティ、否認防止性、およびセキュリティ情報の管理を定義する CORBA サービス。

参考項目 「認証」、「 認可」、「 コールバック・ラッパー・オブジェクト」、および「 オブジェクト」

コア・クラス

(CORBA) Java プラットフォームの標準メンバであるパブリック・クラス (インターフェイス)。最小限の Java コア・クラスを Java プラットフォームが動作するすべてのオペレーティング・システム上で利用できるようにすることが目的です。

参考項目 「クラス」、「 インターフェイス」、および「 Java」

クリデンシャル

ユーザまたはその他のプリンシパルのセキュリティ属性 (ID または特権) を記述した情報。クリデンシャルは、認証またはデレゲーションを通して要求され、アクセス制御によって使用されます。

参考項目 「認証」

Credentials **オブジェクト**

(CORBA) プリンシパルのセキュリティ属性を保持するオブジェクト。これらのセキュリティ属性には、プリンシパルの認証済みまたは認証を受けていない ID が含まれています。Credentials オブジェクトにも、セキュリティの関連付けが確立されていない情報が含まれています。Credentials オブジェクトは、そのオブジェクトが表すプリンスパネルのセキュリティ属性を取得するためのメソッドを提供します。

参考項目 「属性」、「 メタデータ・インターフェイス」、および「 オブジェクト」

暗号法

情報を「暗号文」という読み取り不可能な形式に変換 (暗号化) することにより、情報を保護する手法。メッセージを「平文」に戻す (解読する) ことができるのは、秘密鍵の所有者だけです。

参考項目 「暗号文」および「 平文」

CSI

BEA TOP END システムの API。

カレント・コンテキスト

(ATMI) クライアントは、複数のコンテキストに初期化できます。ただし、どの時点でも、どの特定のスレッドでも、カレント・コンテキストは 1 つのコンテキストだけです。

(CORBA) ユーザ・アプリケーションと特殊化された組み込みサービスとの通信用の特殊な ORB オブジェクト。

参考項目 「CORBA ORB」、「 オブジェクト」、「 SecurityCurrent」、および「 TransactionCurrent」

カスタム GUI **コンポーネント**

(ATMI) JoltBeans と通信する Java GUI クラス。通信は、JoltBeans が提供する JavaBeans のイベント、メソッド、またはプロパティを介して行われます。

D

デーモン

バックグラウンドで実行されるシステム・プロセス。

DASD

「Direct Access Storage Device (DASD)」を参照。

データベース

相互に関連するデータや独立したデータの格納場所。1 つまたは複数のアプリケーションに対するサービスをすばやく提供します。

データベース管理システム (DBMS: Database Management System)

データベース内のテーブルのデータをユーザ側で構成したり、操作できるようにしたプログラムまたはプログラムのセット。DBMS では、データのプライバシー保護、回復機能、およびマルチ・ユーザ環境での整合性が保証されています。

データ依存型ルーティング

- メッセージのデータ・フィールドの値に基づいて、特定のグループ宛てに処理対象の要求をルーティングすること。

- データ・バッファ内の指定されたフィールド値に基づいて、サービス要求を特定のサーバ・グループにマッピングする BEA Tuxedo システムのメカニズム。

Data Encryption Standard (DES)

米国政府が機密文書以外のデータに対する暗号 / 解読の標準システムとして 1976 年に採用した対称鍵アルゴリズム。DES には、DES-CBC や、2 つの鍵による Triple DES などの種類があります。

- DES-CBC は、CBC (Cipher Block Chaining) モードで実行する 64 ビットのブロック暗号です。DES-CBC の暗号化キーの長さは 56 ビット (64 ビットの暗号化キーから 8 パリティ・ビットを引いたもの) です。
- 2 つの鍵による Triple DES は、EDE (Encrypt-Decrypt-Encrypt) モードで実行する 128 ビットのブロック暗号です。これは、2 つの 56 ビットの暗号化キー (112 ビットの暗号化キー) を提供します。

データの独立性

高水準のデータ管理方法を使用して、データの格納方法や取得方法を気にせずデータを要求できること。

データ転送プロトコル

データに指定された特定のバッファ型を別の表現に変換するときに使用される規則のセット。

DB2

IBM 製のリレーショナル・データベース。

DBBL

「Distinguished Bulletin Board Liaison (DBBL)」を参照。

DBMS

「データベース管理システム (DBMS: Database Management System)」を参照。

DDE

「動的データ交換 (DDE: Dynamic Data Exchange)」を参照。

DDE 通信

クライアント・アプリケーションとサーバ・アプリケーション間で DDE メッセージを送受信すること。

デッドロック

- リソースの使用時に発生した競合が未解決の状態であること。
- プロセス内の 2 つの要素が、それぞれもう一方のアクションまたは応答を待機しているため、処理を続行できないエラー状態のこと。

復号化

符号化されたデータをネイティブな形式に戻すこと。

参考項目 「符号化」

暗号解読

暗号化されたデータを元の形式に戻すこと。

暗号解読用秘密鍵

暗号化アルゴリズムの逆の処理を行うアルゴリズム。

デフォルト

ユーザが何も指定しないときにプログラム側で設定される値。

デフォルト・コンテキスト

`tpsetctxt()` が呼び出されない場合に、以降の ATMI 呼び出しで参照される BEA Tuxedo アプリケーションの関連性。デフォルト・コンテキストは、スレッドごとに異なる場合があります。この用語は、「アプリケーション・コンテキスト」の同義語として扱われます。

参考項目 「アプリケーション・コンテキスト」

遅延同期通信

非同期通信の一種。あるソフトウェアから別のソフトウェアへメッセージを送信した後、しばらく経ってから処理を続け、メッセージに対する応答を取得すること。

デプロイメント

分散環境にアプリケーションを配置し、アプリケーションを使用可能にするプロセス。導入作業には、アプリケーションのさまざまな部分のインストール、コンフィギュレーション、および管理があります。

DES

「Data Encryption Standard (DES)」を参照。

デザイン・ドキュメント

作成するアプリケーションまたはフレームワークのデザイン全体を説明したドキュメント。システム・インテグレータが作成します。

デザイン・パターン

デザインに関する問題のソリューションを、構造化された形式に従ってカプセル化した文書。デザイン・パターンは、適切なデザインを実践するための指針となります。

参考項目 「Client Data Caching デザイン・パターン」および「Process-Entity デザイン・パターン」

デスクトップ・クライアント

(CORBA) Microsoft 製デスクトップ・プラットフォーム (Windows NT や Windows 95 など) 上で動作する CORBA クライアント・アプリケーション。デスクトップ・クライアント・アプリケーションは、Component Object Model (COM) を使用しており、COM と CORBA 間の変換に ActiveX Client を使用して BEA Tuxedo ドメインと通信します。

参考項目 「ActiveX クライアント」、「 Component Object Model (COM)」、「 会話」、および「 アプリケーション」

対話

情報の送受信を行うプロセス。

デジタル証明書

インターネットなどのネットワークを介して、個人およびリソースを識別するために使用される電子ファイル。デジタル証明書は、信頼性のある第三者機関である「認証局」によって認定された個人またはリソースの ID を特定の公開鍵に安全な方法で結び付けます。公開鍵は重複しないため、公開鍵から所有者を特定できます。

BEA Tuxedo の公開鍵によるセキュリティ機能では、X.509 バージョン 3.0 準拠の証明書が認識されます。

デジタル署名

電子的に送信されるメッセージに添付され、送信側を一意に識別するデジタル・コード。デジタル署名により、送信側の ID の認証を行うことができます。メッセージが認証されると、(1) メッセージが本物であること、(2) 内容が改ざんされていないこと、(3) 記述されている送信元から発信されたものであることを確認できます。

デジタル署名は、電子商取引で特に重要であり、多くの認証機能では主要なコンポーネントです。署名付きのデータを受信したら、その署名が確かに署名者によって生成されたものであるかどうかを第三者機関に確認します。署名者による署名であることが証明されたら、その署名には否認防止性が設定されます。つまり、署名者は、データを送信したことを後で否認することはできません。

デジタル署名アルゴリズム

任意の長さのメッセージをデジタル署名に変換するアルゴリズム。(1) 同じデジタル署名が付いた 2 つのメッセージを検索する、(2) 定義済みのデジタル署名からメッセージを作成する、または (3) 送信側の秘密鍵なしでメッセージのデジタル署名を検索する、という操作をコンピュータで行えないようにします。通常、デジタル署名アルゴリズムをインプリメントするには、メッセージのメッセージ・ダイジェストを計算し、次に送信側の秘密鍵でメッセージ・ダイジェストを暗号化します。

デジタル署名アルゴリズムの例として、DSA があります。

DII

「動的起動インターフェイス (DII: Dynamic Invocation Interface)」を参照。

Direct Access Storage Device (DASD)

ディスク、ディスク・ドライブ、ディスク・グループ、または IBM マシン上のドライブ。

Distinguished Bulletin Board Liaison (DBBL)

アプリケーションの MASTER ノード上で実行され、掲示板の更新処理を調整するために BBL と通信を行う、BEA Tuxedo の管理プロセス。

識別名 (DN: Distinguished Name)

(CORBA) 識別名 (DN) は、X.500 ディレクトリ内のオブジェクトを一意に識別するディレクトリ情報ツリー (DIT: Directory Information Tree) 内のエントリです。

参考項目 「オブジェクト」

分散アプリケーション

2 つまたはそれ以上の部分 (クライアントとサーバなど) に分割されたアプリケーション。異なるコンピュータ上に配置されたそれぞれの部分は、ネットワーク経由で通信します。

分散コンピューティング

アプリケーションを複数の単位に分割してそれぞれを異なるコンピュータ上で実行し、ネットワーク経由で通信するというアプリケーション・デザインおよびインプリメンテーション方法。たとえば、ユーザ・インターフェイス用、処理用、記憶域用の 3 つの部分にアプリケーションを分割することができます。

分散オブジェクト

(CORBA) ネットワーク上のどこにでも配置可能なオブジェクト。分散オブジェクトは、リモート・クライアントがメソッド呼び出しを介してアクセス可能な独立したコードとしてパッケージ化されています。分散オブジェクトの作成用の言語およびコンパイラは、クライアントに対して完全に透過的です。クライアント側は、分散オブジェクトがどのオペレーティング・システム上のどの場所にあるのかを知る必要がありません。

Distributed Program Interface (DPI)

SNMP エージェントを拡張した Distributed Program Interface (DPI) プロトコル。エンド・ユーザは、SNMP-DPI プロトコルを介してエージェントと通信するサブエージェントを作成することで、SNMP エージェントを再コンパイルしなくても ローカル MIB の変数を動的に追加、削除、または置換することができます。

分散トランザクション

複数のトランザクション・マネージャを含むトランザクション。分散トランザクション環境では、クライアント・アプリケーションが複数のサーバに要求を送信し、その結果、複数のリソース・マネージャでリ

ソースが更新される場合があります。トランザクションを完了するため、クライアント、サーバ、リソース・マネージャの各トランザクション・マネージャは、ドメイン内でポーリングされ、各パーティシパントのコミット処理を調整する必要があります。

分散トランザクション処理 (DTP: Distributed Transaction Processing)

複数のアプリケーション・プログラムが複数のリソース (データベースなど) を協調して更新する処理のこと。アプリケーション・プログラムおよびリソースは、ネットワーク上の 1 つまたは複数のコンピュータに常駐できます。

DLL

「ダイナミック・リンク・ライブラリ (DLL: Dynamic Link Libraries)」を参照。

ドメイン

「アプリケーション」を参照。

ドメイン・コンフィギュレーション (DMCONFIG) **ファイル**

ローカル・ドメイン (DMCONFIG ファイルがあるドメイン) とリモート・ドメイン (その他のドメイン) との間の関係を記述したファイル。DMCONFIG ファイルはドメインごとに 1 つあります。DMCONFIG ファイルには、BEA Tuxedo ドメインに関するドメイン情報が格納されています。

関連項目「ドメイン」

ドメイン・ゲートウェイ

リモート・ドメインとの間でサービス要求を処理するために BEA Tuxedo システムが提供する高度な非同期マルチタスキング・サーバ。ゲートウェイにより、リモート・ドメイン上のサービスへのアクセス、およびリモート・ドメインからのサービス要求は、アプリケーション・プログラマとユーザに対して透過的に処理されます。

ドメイン・ゲートウェイ・グループ

「ゲートウェイ・グループ」を参照。

Domains

複数のドメイン間の相互運用性を実現するフレームワークを提供する BEA Tuxedo システムのコンポーネント。このフレームワークは、次のコンポーネントで構成されています。

- さまざまなコンピュータやアプリケーション・プログラムの異種性、およびこれらのアプリケーション・プログラムの位置を透過的に扱う、拡張されたクライアント / サーバ型モデル。
- 集中管理型のサブシステム。アプリケーション管理者は、連係するすべてのマシンを単一のアプリケーションとして制御できます。

Domains レベルのフェイルバック

プライマリ・リモート・ドメインにメッセージ・トラフィックを復元する機能。「TDomain ゲートウェイ」は、プライマリ・ドメインまたはサービスに対して定義された最上位レベルの代替リモート・ドメインを常に使おうとします。これらのドメインが、回線障害やその他の理由によって使用できなくなると、ゲートウェイは優先順位の低い代替リモート・ドメインにメッセージ・トラフィックを転送し、その間、プライマリ・ドメインと最上位レベルの代替リモート・ドメインの状態を定期的にチェックします。プライマリ・ドメインまたは最上位レベルのリモート・ドメインが使用できるようになると、ゲートウェイはメッセージ・トラフィックを復元します。

「TDomain ゲートウェイ」および 「ドメイン・ゲートウェイ」を参照。

Domains レベルのフェイルオーバー

プライマリ・リモート・ドメインに障害が発生した場合に、代替リモート・ドメインにメッセージ・トラフィックを転送する機能。

「リモート・ドメイン」を参照。

ドット区切りの 10 進表記

IP アドレスの表記規則。4 組の 10 進数 (0 ～ 255) をピリオドで区切って示します (例 : 123.205.23.99)。

DPI サブエージェント

「Distributed Program Interface (DPI)」を参照。

DSA

Digital Signature Algorithm の略。デジタル署名を生成するためのアルゴリズムです。DSA は US FIPS 186 で定義されています。

DSI

「動的スケルトン・インターフェイス (DSI: Dynamic Skeleton Interface)」を参照。

DTP

「分散トランザクション処理 (DTP: Distributed Transaction Processing)」を参照。

デュアル・ペア接続のアウトバウンド IIOP

「アウトバウンド IIOP」を参照。

動的な引数

メソッドが既存の記憶域を割り当てるか、または拡張する引数。

動的データ交換 (DDE: Dynamic Data Exchange)

アプリケーションが一連のメッセージを介して情報をやり取りできるようにする Microsoft Windows プラットフォーム上で利用可能な通信形式。DDE メッセージを送受信する 2 つのアプリケーションは、「DDE 会話を行うアプリケーション」と表現します。

動的なデータ型

コードのコンパイル時には、メモリ・サイズが不明なデータ型。動的なデータ型のメモリ・サイズは、コードの実行時にのみわかります。

動的起動インターフェイス (DII: Dynamic Invocation Interface)

(CORBA) CORBA クライアントが、コンパイル時に未知の署名を持つオブジェクトの呼び出し、または遅延同期呼び出しを実行することを可能にする API。オブジェクトの署名が未知の場合、クライアントはオブジェクトを探し、インターフェイス・リポジトリを使用してオブジェクトの署名についての情報を取得し、適切なパラメータを付けて呼び出しを作成します。次に、クライアントは呼び出しを実行し、応答を受け取ります。DII は、クライアント・スタブを使用して同期呼び出しを実行する静的起動インターフェイスとは違います。DII を使用すると、クラ

イアントが要求を発行し、それが完了するまでブロックされないようにすることができます。クライアントは、応答があったかどうかを後でチェックします。

「動的スケルトン・インターフェイス (DSI: Dynamic Skeleton Interface)」を参照。

ダイナミック・リンク・ライブラリ (DLL: Dynamic Link Libraries)

ロード・モジュールにグループ化された関数の集合。Microsoft Windows のアプリケーションでは、実行時に実行形式プログラムに動的にリンクされます。

動的スケルトン・インターフェイス (DSI: Dynamic Skeleton Interface)

(CORBA) ORB からオブジェクト・インプリメンテーションに要求を送る方法を提供する API。DSI は、ORB にオブジェクト・インプリメンテーションの情報がない場合にコンパイル時に使用します。サーバ・サイドでクライアント・サイドの DII に相当する DSI を使用すると、アプリケーション・プログラマは受信した要求のパラメータを調べて、ターゲット・オブジェクトおよびメソッドを決定できます。

「動的起動インターフェイス (DII: Dynamic Invocation Interface)」を参照。

E

電子商取引

インターネット上で商品やサービスの売買を行うこと。

符号化

アーキテクチャ固有のデータを、異なるアーキテクチャ間で送信できる形式に変換すること。符号化の例として XDR があります。

暗号

アルゴリズムを使用してデータを変換し、データが不正に開示されないようにしつつ、認可されたユーザは元のデータにアクセスできるようにするプロセス。暗号化されたファイルを表示するには、解読するための秘密鍵へのアクセスまたはパスワードが必要です。暗号化されていないデータは「平文」、暗号化されたデータは「暗号文」と呼ばれます。

「平文」および 「暗号文」を参照。

暗号鍵のペア

暗号鍵のペアは、情報の暗号化用の公開鍵と解読用の鍵からなります。

環境オブジェクト

(CORBA) 基となる環境からの独立性 (たとえば、オペレーティング・システムからの独立性) を提供するサポート・オブジェクト。Bootstrap オブジェクトは環境オブジェクトの 1 つです。

参考項目 「Bootstrap 環境オブジェクト」および「オブジェクト」

環境変数

アプリケーションの特定の属性を制御するための値の文字列。環境変数は、アプリケーションの起動時に使用可能になります。

イベント

接続の切断、トランザクションの要求モード、接続要求などの状態や条件の発生を、BEA Tuxedo システムのプロセスに通知すること。

イベント・ブローカ / モニタ

定義済みのシステム・イベントやアプリケーション・イベントの発生を監視し、イベントの検出時にサブスクライバに通知する BEA Tuxedo システムのコンポーネント。

イベントのポスト

定義済みイベントが発生したことを、BEA Tuxedo システム (またはアプリケーション) からイベント・ブローカ / モニタに通知すること。

イベント・サブスクリプション

指定したイベントが検出されたことを通知するようイベント・ブローカ / モニタに要求すること。

例外

(CORBA) プログラムの実行中に発生し、通常はエラーとなってプログラムを正常に続行できないようにするイベント。C++ では、`try`、`catch`、および `throw` キーワードによって例外がサポートされています。例外には、システム例外とユーザ定義例外の 2 種類があります。

C++ では、システム例外は `CORBA::System_Exception` を継承し、ユーザ定義例外は `CORBA::User_Exception` を継承します。

拡張性

新しく発生した条件にシステムを適応できること。拡張性があると、以下のことを考慮せずに機能またはデータ (データ型、ファイル形式、データベース・スキーマ、情報モデル) を追加したり、削除できます。

- 既存の機能、データ、およびインターフェイスの変更
- パフォーマンス、信頼性、管理のしやすさ、移植性などの低下

Exterior Gateway Protocol (EGP)

自律的なシステムに到達可能なネットワークの集合を宣言するためのプロトコル。EGP を使用すると、この情報をほかの自律的なシステムと共有できます。

外部データ表現 (XDR: External Data Representation)

Sun Microsystems によって定義された標準的なデータ形式。異種ハードウェアのノード間でデータ転送を行うときに使用します。

F

ファクトリ

(CORBA)

- ほかの分散 CORBA オブジェクトへのオブジェクト・リファレンスを返す分散 CORBA オブジェクト。ファクトリは、サーバ・アプリケーション内にあります。
- CORBA オブジェクトへのオブジェクト・リファレンスを取得するためにクライアントが使用するインターフェイス。ファクトリへのオブジェクト・リファレンスは、ファクトリ・ファインダ・インターフェイスへのオブジェクト・リファレンスを使用してクライアントによって取得されます。ファクトリ・ファインダ・インターフェイスはシステムによって宣言され、クライアントのブートストラップ処理の一部としてクライアントから利用できるようになります。

参考項目 「CORBA オブジェクト」、「 オブジェクト・リファレンス」、および「 アプリケーション」

ファクトリ・ベース・ルーティング

(CORBA) オブジェクト・リファレンスがファクトリによって作成されるときに指定された条件に基づいて、CORBA オブジェクト・リファレンスのリクエストを特定のサーバ・グループにルーティングする BEA Tuxedo ソフトウェアの機能。

参考項目 「ファクトリ」および「 オブジェクト・リファレンス」

ファクトリ・ファインダ

(CORBA) アプリケーションが必要とするファクトリを探し出す CORBA オブジェクト。クライアント・アプリケーションでもサーバ・アプリケーションでもファクトリ・ファインダを使用できます。ファクトリ・ファインダ・オブジェクトは、CORBA サービスの `COSLifeCycle.FactoryFinder` インターフェイスだけでなく、BEA `Tobj.FactoryFinder` インターフェイスのインプリメンテーションも提供します。

参考項目 「ファクトリ」、「 ローカル・ファクトリ」、「 オブジェクト」、「 クライアント・アプリケーション」、および「 サーバ・アプリケーション」

`factory_finder.ini` ファイル

(CORBA) ドメインの FactoryFinder コンフィギュレーション・ファイル。このファイルは、マスタ NameManager として開始された `TMFFNAME` サービスによって解析されます。このファイルには、ほかのドメインのファクトリ・オブジェクトに対するオブジェクト・リファレンスのインポートおよびエクスポートを制御するために NameManager で使用される情報が格納されています。

参考項目 「ドメイン」、「 ファクトリ」、および「 オブジェクト・リファレンス」

フェイルバック

優先順位の高い回線にメッセージ・トラフィックを復元する機能。`BRIDGE` は、ノードに定義された最も優先順位の高い回線を常に使用しようとします。回線の異常や単に回線がふさがっているという理由により、トラフィックが優先順位の低い回線に流れているとき、`BRIDGE` は定期的に優先順位に高い回線の状況を確認します。優先順位がより高い回線が使用可能になると、データは再びこの回線に流れます。このメカニズムを「フェイルバック」と呼びます。

フェイルオーバー

(ATMI および CORBA) 優先順位の高い回線に異常が発生したときに、メッセージ・トラフィックが次に優先順位の高い回線に途切れることなく送信されること。オペレーティング・システムやハードウェアによっては、ネットワーク・カード上で発生した問題を検出し、それを別のものに置き換えるように設計されているものがあります。これが高速に処理されると、アプリケーション・レベルの TCP 仮想回線は、障害の発生を表示しません。

BEA Tuxedo システムでは、データは使用可能な回線のうち、優先順位が最も高い回線を流れます。すべてのネットワーク・グループに同じ優先順位が与えられている場合、データはすべてのネットワークに同時に送信されます。現在優先されているすべての回線で障害が発生すると、データは次に優先順位が高い回線に送信されます。これを「フェイルオーバー」と呼びます。

(Jolt) 障害防止のメカニズムのこと。接続中の Jolt リレー・アダプタ (JRAD) が接続リクエストに応答しない場合は Jolt リレー (JRLY) が有効になり、使用可能な別の JRAD に対して接続が行われます。Jolt クライアントは JRLY アドレスの一覧から、JRAD の接続先をラウンド・ロビン方式で探します。

フィールド

- レコードでは、特定の種類のデータ用として指定された領域。
- 最小単位のデータであるセグメント内の領域。
- セグメントの指定部分。
- データベース・テーブル内のデータの 1 つの項目をアドレス指定する方法。
- データが表示されるウィンドウの領域。

フィールド操作言語 (FML: Field Manipulation Language)

(ATMI) フィールド化バッファと呼ばれる記憶域構造を定義し、操作するための C 言語関数のセット。協調するプロセスでは、フィールド化バッファ内のデータを送受信できます。

(Jolt) 広義では、フィールドと値の組み合わせを持つバッファを操作するためのインターフェイス。狭義では、このようなインターフェイスの 16 ビット版。

フィールド・テーブル

(ATMI) FML フィールドの名前と識別子が格納されたファイル。フィールド・テーブルを使用すると、システム・フィールドの識別子の代わりに論理名でフィールドを参照できます。

FML

「フィールド操作言語 (FML: Field Manipulation Language)」を参照。

FML **バッファ**

(ATMI) 自己記述型のデータ項目で構成されるバッファ。フィールド操作言語の API を使用してアクセスします。

外部アクセス・パス

ネイティブな BEA Tuxedo システムのノードと外部ノードとの物理的な接続。BEA Tuxedo のノードには、少なくとも 1 つのゲートウェイ・サーバが必要です。

外部クライアント・アプリケーション

(CORBA) ORB に実装された BEA 社製品以外のクライアント・アプリケーション。BEA Tuxedo ソフトウェアの ActiveX Client コンポーネントは外部クライアント・アプリケーションではありません。このクライアントは Microsoft 製品に実装されていますが、ORB は BEA 社によって提供されます。

「ORB」および 「BEA ActiveX クライアント」を参照。

外部ノード

コンフィギュレーションの掲示板にアクセスできないネットワーク内のノード、または BEA Tuxedo システム・ソフトウェアのフル・コンポーネントを実行できないネットワーク内のノード。

データ形式の独立性

データの表示形式に関係なく、デバイスにデータを送信できること。複数のデバイス上に、同じデータが異なる形式で表示される場合があります。

フレームワーク

特定のドメインのニーズに合わせて調整されたソフトウェア環境。フレームワークには、ソフトウェア・コンポーネントの集合が含まれており、プログラマはこれらを使用して、フレームワークが提示するドメイン用のアプリケーションを構築します。フレームワークには、専用の API、サービス、およびツールが組み込まれているため、ユーザやプログラマは、特定のタスクを実行するための知識を習得する手間を省けます。

G

ゲートウェイ

異なる環境間 (ネイティブ・ノードと外部ノードなど) での BEA Tuxedo システムの通信メカニズム。または、異なるシステム間での通信や情報の交換を実現するソフトウェア・プログラム。ゲートウェイは通常、システム間の通信を担当し、必要なすべてのプロトコル変換を行って末端のアプリケーションが透過的に通信できるようにします。

ゲートウェイ・グループ

リモート・ドメインとの間で通信サービスを提供するプロセスの集合。ゲートウェイ・グループは、ゲートウェイ管理サーバ (GWADM) と、GWTDOMAIN などのゲートウェイ・プロセスが含まれます。

ゲートウェイ・サーバ

BEA Tuxedo システムのネイティブなノード上に常駐し、1 つまたは複数の外部マシンと通信するサーバ・プロセス。

一般 ORB **間プロトコル** (GIOP: General Inter-ORB Protocol)

(CORBA) 独立した CORBA オブジェクト・リクエスト・ブローカ (ORB) のインプリメンテーション間の通信標準。GIOP は、Object Management Group (OMG) によって開発されました。GIOP は、GIOP 標準を個々のトランスポート層にマップする特定のプロトコルの基準となる抽象プロトコルです。たとえば、IIOP は、GIOP 標準を TCP/IP トランスポート層にマップします。

参考項目 「CORBA ORB」および「 IIOP」

GIOP

「一般 ORB 間プロトコル (GIOP: General Inter-ORB Protocol)」を参照。

グローバル・トランザクション

複数のサーバまたは複数のリソース・マネージャ・インターフェイスを使用し、基本作業単位として調整される BEA Tuxedo のトランザクションの名前。グローバル・トランザクションは複数のローカル・トランザクションから構成され、各トランザクションは 1 つのリソース・マネージャにアクセスします。

参考項目 「リソース・マネージャ (RM: Resource Manager)」

グローバル・トランザクション識別子 (GTRID: Global Transaction Identifier)

グローバル・トランザクションを一意に識別する値のデータ構造体。

グラフィカル・ユーザ・インターフェイス (GUI: Graphical User Interface)

グラフィカルなアイコン付きの、ウィンドウやメニュー・バーを使った高水準のインターフェイス。システム・コマンドを入力する代わりに使用し、ユーザとのインタラクティブな環境を実現します。World Wide Web から利用できる BEA Administration Console を使用すると、アクセス権のあるユーザは、BEA Tuxedo アプリケーションのコンフィギュレーションや制御を行うことができます。

グループ

1 つのマシン上にあるサーバまたはサービスの集合。リソース・マネージャと関連付けられます。グループは、サーバおよびサービスの起動、シャットダウン、移行を行うときに使用される管理単位です。

参考項目 「MIB グループ」

GUI

「グラフィカル・ユーザ・インターフェイス (GUI: Graphical User Interface)」を参照。

H

ハンドラ

リモート・コンピュータで作成される要求。ハンドラは、クライアント・プログラムとして BEA Tuxedo の掲示板に登録されます。

参考項目 「ワークステーション・ハンドラ (WSH)」

階層型データベース

ツリー構造で構成されたデータベース。データベース内のデータに対するアクセス・パスを事前に設定します。階層型データベースの例として、DL/I、IMS および SQL/DS があります。

階層

最上位を root とし、そこから派生するセグメント・タイプを下位に順に配置する、ツリー形式のデータベース構造。セグメント・タイプの派生元は、1 つです。

高水準言語

プログラミング言語。

ホスト

ネットワークにアタッチされたコンピュータ。通信スイッチの役割以外のサービスも提供します。

ホスト・コンピュータ

データ通信システムでのプライマリ・コンピュータ (制御コンピュータ)。

ハイパーテキスト・マークアップ言語 (HTML: Hypertext Markup Language)

World Wide Web のページを記述するための言語。

I

ICF

「インプリメンテーション・コンフィギュレーション・ファイル (ICF: Implementation Configuration File)」を参照。

ident **文字列**

「ID 文字列」を参照。

ID **文字列**

ファイルおよび ID 情報を格納するために RCS および SNMP エージェント・ユーティリティによって拡大されるファイルの一部。コンパイルすると、これらの文字列はオブジェクト・ファイル関数に含まれ、情報が利用できるようになります。

IDL

「OMG IDL」を参照。

idl **コンパイラ**

(CORBA) OMG IDL インターフェイスを使用して、IDL インターフェイスと C++ プログラミング言語のマッピングを表す C++ プログラミング言語のインターフェイスおよびクラスを生成するツール。

参考項目 「OMG IDL」

IDL **インターフェイス**

(CORBA) CORBA オブジェクトに対する OMG IDL でのインターフェイス宣言。インターフェイス宣言には、IDL のオペレーションおよび属性が含まれます。OMG IDL インターフェイス宣言は、BEA Tuxedo CORBA オブジェクト用のスタブおよびスケルトンを作成する場合に使用します。

参考項目 「CORBA オブジェクト」、「 インターフェイス」、「 OMG IDL」、および「 スケルトン」

IDL **パラメータ**

(CORBA) クライアントが IDL オペレーションを呼び出すときにそのオペレーションに渡す 1 つまたは複数のオブジェクト。パラメータは、クライアントからサーバに渡す `in`、サーバからクライアントに渡す `out`、またはクライアントからサーバに渡し、サーバからクライアントに戻す `inout` として宣言できます。

idltojava **コンパイラ**

(CORBA) OMG IDL インターフェイスを使用して、IDL インターフェイスと Java プログラミング言語のマッピングを表す Java プログラミング言語のインターフェイスおよびクラスを生成するツール。ファイルとして `.java` ファイルが作成されます。

参考項目 「OMG IDL」

IIOP

(CORBA) Internet Inter-ORB Protocol の略。オブジェクト・リクエスト・ブローカ (ORB) 間の相互運用性を実現するために、Object Management Group (OMG) によって作成された CORBA 仕様で定義されている標準プロトコル。IIOP を使用すると、2 つ以上のオブジェクト・リクエスト・ブローカ (ORB) が協調してリクエストをオブジェクトに転送できます。

参考項目 「CORBA ORB」および「 オブジェクト」

IIOP **ハンドラ** (ISH)

(CORBA) リモート・アプリケーションとターゲット CORBA オブジェクト間のすべての IIOP 通信を処理する BEA Tuxedo システムのプロセス。

参考項目 「Jolt サーバ・ハンドラ (JSH: Jolt Server Handler)」および「 ワークステーション・ハンドラ (WSH)」

IIOP **リスナ** (ISL)

(CORBA) リモート・アプリケーションからの IIOP 接続をリッスンする BEA Tuxedo システムのプロセス。接続が確立されたら、リスナは接続を IIOP ハンドラに渡します。

参考項目 「Jolt サーバ・リスナ (JSL: Jolt Server Listener)」および「 ワークステーション・リスナ (WSL)」

IIOP **リスナ / ハンドラ**

(CORBA) クライアント・アプリケーションと BEA Tuxedo ドメイン間の通信を可能にする BEA Tuxedo ソフトウェアの機能。IIOP リスナ / ハンドラは、IIOP プロトコルを介してクライアント・アプリケーションからリクエストを受信し、それを BEA Tuxedo ドメイン内の適切なサーバ・アプリケーションに送信します。また、BEA Tuxedo ドメイン内のサーバ・アプリケーションからリクエストを受信し、それをドメイン外のサーバに送信します。

参考項目 「IIOP」、「 クライアント・アプリケーション」、「 ドメイン」、および「 サーバ・アプリケーション」

(Jolt) IIOP を使用して送信されたクライアントからのリクエストを受信し、そのリクエストを適切なサーバ・アプリケーションに転送するプロセス。

インプリメンテーション・コード

(CORBA) 特定のオブジェクトでクライアント・アプリケーションの要求を満たすために作成したメソッドのコード。インターフェイスはオペレーションを定義し、メソッドに実装されます。

参考項目 「インターフェイス」、「 メタデータ・インターフェイス」、および「 オブジェクト」

インプリメンテーション・コンフィギュレーション・ファイル (ICF: Implementation Configuration File)

(CORBA) BEA Tuxedo C++ サーバ・アプリケーションのインプリメンテーション属性を記述したファイル。ICF ファイルは、BEA Tuxedo C++ サーバ・アプリケーション用のスケルトンを作成するときに IDL への入力となります。

参考項目 「スケルトン」および「 サーバ・アプリケーション」

インプリメンテーション・ファイル

(CORBA) OMG IDL 文で定義した各オペレーションのメソッド宣言をほかのデータといっしょに格納したファイル。そのメソッドをビジネス・ロジックに実装します。サーバ・アプリケーションをビルドする場合、このインプリメンテーション・ファイルを BEA Tuxedo のビルド手順で指定します。

参考項目 「インプリメンテーション・コード」、「 メタデータ・インターフェイス」、「 OMG IDL」、「 オペレーション」、および「 サーバ・アプリケーション」

非アクティブなサーバ

現在要求を処理できないサーバ。

受信時接続

リモート・ドメインのゲートウェイ・プロセスによって開始されるローカル・ゲートウェイへの接続。

情報の非表示

コードに、ジョブに必要な情報だけを格納するソフトウェア・デザイン技術。

インフラストラクチャ

共通のコンピューティング環境基盤。システム内で上位コンポーネントをサポートするコンポーネント・セット (基礎的なサービス) のことです。上位コンポーネントは通常、システム全体に関わる特定の機能を提供します。

初期ネーミング・コンテキスト

(CORBA) CORBA を使用した場合に、メソッド `orb.resolve_initial_references(「NameService」)` を呼び出して返された `NamingContext` オブジェクト。これは、ORB に登録された CosNaming サービスへのオブジェクト・リファレンスです。初期ネーミング・コンテキストを使用すると、ほかの `NamingContext` オブジェクトを作成できます。

参考項目 「ネーミング・コンテキスト」

インスタンス

(CORBA) オブジェクトのクラスやコンピュータのプロセスなど、抽象またはテンプレートを特定の形式で現実化したもの。

インスタンス化

(CORBA) クラス内のオブジェクトを特定の定義に従って変化させ、名前を付けて物理位置に配置することでインスタンスを作成すること。

インスツルメンテーション

管理対象リソースの属性へのアクセスを提供して、属性値を取得または修正できるようにする機能。エージェントが使用する管理対象リソースにアクセスして、管理要求に応答します。

統合

サービスを要求し、要求された内容を実行することにより、情報を共有したり、プロセスを個々に処理できるようにするアプリケーションの機能。十分に統合されたシステムでは、すべての部分が意味を持ち、各部分が効果的に組み合わされてシステム全体の目的を果たします。

インタラクション・モデル

分散アプリケーションまたはアプリケーション・フレームワークでのクライアントやサーバが、どのように相互作用しているかを示すもの。

インタラクティブ

データの照会システムや航空会社の予約システムなどに見られるように、各要求にはシステムまたはプログラムからの応答が必要である、というアプリケーションに関連する性質。インタラクティブなシステムは会話型、つまり、ユーザとシステムの間で継続的に対話を行います。

インタラクティブ・インターフェイス

ユーザ・プロファイルを使用して、異なるユーザがそれぞれシステムとどのように動作しているかを制御するシステム機能。インタラクティブ・インターフェイスは、ユーザ・プロファイルで許可されているシステムの一部をサインオン時に有効にします。インタラクティブ・インターフェイスには、選択肢のセットとデータ入力用のパネルが用意されており、これらを使用してユーザはシステムと通信することができます。

インターフェイス

「IDL インターフェイス」を参照。

インターフェイス・リポジトリ

(CORBA) クライアント・アプリケーションとサーバ・アプリケーション間の CORBA の取り決めを決定するインターフェイス定義を格納するオンライン・データベース。

「IDL インターフェイス」および 「会話」を参照。

国際化

システムのテキスト・メッセージや日付形式をアプリケーション用の言語や形式にカスタマイズするメカニズム。

国際標準化団体 (ISO: International Standards Organization)

世界各国の標準化グループや調査グループで構成された国際的な組織。ISO は、コンピュータ・ネットワーク・コミュニケーションやその他の技術に関する標準を制定しています。

インターネット

TCP/IP プロトコルに基づいた、どこからでもアクセスできる世界最大のネットワーク。

インターネット ORB **間プロトコル** (IIOP: Internet Inter-ORB Protocol)

「IIOP」を参照。

インターネット・プロトコル・アドレス (IP **アドレス**)

TCP/IP ネットワークのノードを一意に識別する数値。IP アドレスは通常、ドット区切りの 10 進数で表記されます。4 組の 10 進数値 (0 ～ 255) をピリオドで区切り、「123.205.23.99」のように示します。

相互運用性

エンティティ間で要求を交換できること。

インターオペラブル・オブジェクト・リファレンス (IOR: Interoperable Object Reference)

(CORBA) タグ付きプロファイルのコレクションをオブジェクト・リファレンスに関連付けるエンティティ。ORB では、オブジェクト・リファレンスが ORB 間でやり取りされるたびに、オブジェクト・リファレンスから IOR を作成する必要があります。

参考項目 「CORBA ORB」および「 オブジェクト・リファレンス」

イントラネット

企業内または特定のグループに限定したネットワーク。ネットワークはファイアウォールでセキュリティ保護され、IP ルータで接続されます。ユーザ側では、イントラネットは単一のネットワークに見えます。

起動

(CORBA) ネットワーク上のオブジェクトの場所を認識しているかどうかに関わりなく、分散オブジェクトに対するメソッド呼び出しを実行するプロセス。起動するためにクライアント・スタブ、起動するサービスのためにサーバ・スケルトンを使用する CORBA 静的起動は、コンパイル時にオブジェクトのインターフェイスが既知の場合に使用します。コンパイル時にインターフェイスが未知の場合は、CORBA 動的起動を使用する必要があります。

参考項目 「CORBA コールバック・オブジェクト」および「 スケルトン」

呼び出しアクセス方針

(CORBA) クライアント・アプリケーションが要求で指定したターゲット・オブジェクトのメソッドを呼び出す許可を持つかどうかを制御するセキュリティ方針。

参考項目 「メタデータ・インターフェイス」および「 方針」

IP アドレス

「インターネット・プロトコル・アドレス (IP アドレス)」を参照。

ISO

「国際標準化団体 (ISO: International Standards Organization)」を参照。

J

JAR ファイル (.jar)

(CORBA) Java ARchive ファイル。数多くのファイルを 1 つのファイルに集約するためのファイル形式。

参考項目 「Java」

Java

Sun Microsystems 社が開発したオブジェクト指向のプログラミング言語。「一度書けば、どこでも実行できる」プログラミング言語です。

JavaBeans

(CORBA) Sun Microsystems 社が開発した、Java オブジェクト間の相互作用の仕組みを定義した仕様。この仕様に合致するオブジェクトは JavaBean と呼ばれます。ActiveX コントロールと似ています。JavaBeans のフォーマットを認識するアプリケーションであれば、どんなアプリケーションでも JavaBean を使用することができます。ActiveX コントロールと JavaBeans の最大の違いは、使用するプログラミング言語とプラットフォームです。ActiveX コントロールの開発はどんなプログラミング言語を使っても行うことができますが、実行できるのは Windows プラットフォームのみです。一方、JavaBeans の開発は Java 言語でのみ可能ですが、どんなプラットフォームでも実行できます。

Java Development Kit (JDK)

アプレットおよびアプリケーションを Java で作成するためのソフトウェア開発環境。

参考項目 「アプレット」および「 Java」

Javadoc

(CORBA) Java ソース・コードの注釈から HTML 形式の API マニュアルを作成する Sun Microsystems 製ツール。『Java API Reference』マニュアルは、Javadoc ツールでフォーマットされています。

参考項目 「アプリケーション・プログラミング・インターフェイス (API: Application Programming Interface)」

Java Runtime Environment (JRE)

エンド・ユーザ向けに再配布するための Java Development Kit のサブセット。JRE は、Java 仮想マシン (JVM)、Java コア・クラス、およびサポート・クラスで構成されています。

JDK

「Java Development Kit (JDK)」を参照。

共同クライアント / サーバ・アプリケーション

あるビジネス・アクションの起点となるコードを実行しつつ、オブジェクトを起動するためのメソッド・コードも実行するアプリケーション。

参考項目 「ネイティブ共同クライアント / サーバ・アプリケーション」

JoltBeans

(ATMI) Jolt クライアントを作成するための JavaBeans コンポーネント。Java 開発環境で使用します。JoltBeans は、2 つの JavaBeans のセット (JoltBeans ツールキットと Jolt 対応 AWT Beans) で構成されています。

JoltBeans **ツールキット**

(ATMI) BEA Jolt 用の JavaBeans 対応インターフェイス。JoltBeans ツールキットには、JoltServiceBean、JoltSessionBean、JoltUserEventBean が含まれます。

Jolt **クラス・ライブラリ**

(ATMI) Java クラスのセット。これを使用して Java プログラムを記述し、BEA Tuxedo サービスにアクセスすることができます。

Jolt **リレー** (JRLY: Jolt Relay)

(ATMI) Jolt リレー・アダプタ (JRAD) を介して、Jolt メッセージを Jolt クライアントから Jolt サーバ・リスナ (JSL) または Jolt サーバ・ハンドラ (JSH) にルーティングするスタンドアロンのプログラム。Jolt リレーは、BEA Tuxedo サーバや BEA Tuxedo クライアントではありません。

Jolt **リレー・アダプタ** (JRAD: Jolt Relay Adapter)

(ATMI) BEA Tuxedo サービスを持たない BEA Tuxedo アプリケーション・サーバ。これを JSL および BEA Tuxedo システムと共に動作させるには、コマンド行引数が必要です。Jolt リレー・アダプタの配置場所は、JSL サーバの接続先である BEA Tuxedo のホスト・マシンおよびサーバ・グループでなくてもかまいません。

Jolt **リポジトリ**

(ATMI) 基本的なサービスとサービスの定義内容を格納しておく記憶域。Jolt のサブシステムです。

Jolt **サーバ・ハンドラ** (JSH: Jolt Server Handler)

(ATMI) BEA Tuxedo サーバ・マシンで実行されるプログラム。リモート・クライアント用のネットワーク接続ポイントを提供します。Jolt サーバ・ハンドラは、Jolt サーバ・リスナと動作して、クライアントが BEA Tuxedo システムに接続できるようにします。

「ワークステーション・ハンドラ (WSH)」および「IIOP ハンドラ (ISH)」を参照。

Jolt **サーバ・リスナ** (JSL: Jolt Server Listener)

(ATMI) IP/ ポートの組み合わせで、クライアントをサポートするプログラム。Jolt サーバ・リスナは、Jolt サーバ・ハンドラと動作して、クライアントが Jolt システムのバックエンドに接続できるようにします。Jolt サーバ・リスナの管理には、BEA Tuxedo 環境のリソース管理用のツールを使用します。

「ワークステーション・リスナ (WSL)」および「IIOP リスナ (ISL)」を参照。

Jolt WAS for IIS

(ATMI)「BEA Tuxedo-ASP Connectivity」に変更。

Jolt WAS for Servlet

(ATMI)「BEA Tuxedo-JSE Connectivity」に変更。

Jolt WAS for WebLogic

(ATMI) 「BEA Tuxedo-WebLogic Connectivity」に変更。

ジャーナリング

後で処理が発生したときのために、情報をジャーナル (システム・ログを含む) に記録しておくこと。ジャーナリングの最大の目的は、データ・セットのフォワード・リカバリを実現することです。つまり、前のバージョンのデータ・セットに対してジャーナルに記録されたトランザクションを適用し、データ・セットを再構築します。ジャーナリングは、監査、会計、パフォーマンス分析など、ユーザ定義した内容に使用できます。

JRAD

「Jolt リレー・アダプタ (JRAD: Jolt Relay Adapter)」を参照。

JRE

「Java Runtime Environment (JRE)」を参照。

JREPSVR

Jolt リポジトリの記憶域にアクセスするためのサービスを提供する BEA Tuxedo サーバ。Jolt ランタイム環境と、必要最低限の編集やクエリの機能をサポートします。

JRLY

「Jolt リレー (JRLY: Jolt Relay)」を参照。

JSH

「Jolt サーバ・ハンドラ (JSH: Jolt Server Handler)」を参照。

JSL

「Jolt サーバ・リスナ (JSL: Jolt Server Listener)」を参照。

K

Kerberos **プロトコル**

マサチューセッツ工科大学 (MIT) の Athena プロジェクトで開発された秘密鍵の認証プロトコル。

Kerberos **セキュリティ**

認証、相互認証、およびリプレイ攻撃やシーケンス番号攻撃に対する保護機能を搭載したセキュリティ・システム。

キーワード

- パラメータを識別する記号。
- 特定の文字列で構成されるコマンド・オペランドの一部。

L

LAN

「ローカル・エリア・ネットワーク (LAN: Local Area Network)」を参照。

LAN **の分断**

アプリケーションの各マシンを接続できないという LAN の障害のこと。この結果、マシン間のメッセージ通信が切断されます。マスタ・ノードにアクセスできなくなったサイトを「分断されたサイト」と呼びます。

レイジー接続

リモート・ドメインがリモート・サービスの要求を受信すると確立される、ドメイン・ゲートウェイとリモート・ドメインとの間の接続。多数のドメインを含むコンフィギュレーションでは、レイジー接続を使用すると初期化時のオーバーヘッドを低く抑えることができます。

ドメイン・ゲートウェイ・サーバの起動時には、リモート・ドメインへの接続は確立されません。すべてのリモート・サービスは有効であると見なされ、BEA Tuxedo の掲示板で宣言されます。特定のリモート・ドメインのサービスに対して最初の要求が発行されると、ゲートウェイ・

サーバは、その要求を受け取り、接続を確立しようとします。接続が確立されると、要求はリモート・ドメインに送信されます。接続状態はアクティブなままになります。接続に失敗すると、クライアントにエラー・メッセージが返され、サービスは掲示板で宣言されたままの状態になります。

LDAP

「Lightweight Directory Access Protocol」を参照。

レガシー・アプリケーション

(ATMI) 通常は BEA Tuxedo システムの以前のリリースを基にした既存のアプリケーション。修正またはラップしてからでないと、BEA Tuxedo ドメインで使用できません。

参考項目 「ドメイン」、「 ワークステーション・ハンドラ (WSH)」、および「 ラッパー」

ライフサイクル・サービス

「CORBA ライフサイクル・サービス」を参照。

Lightweight Directory Access Protocol

情報ディレクトリにアクセスするためのプロトコルのセット。これらのディレクトリは、企業内のさまざまなアプリケーションがアクセスできるように、複数のシステムに分散させることができます。LDAP は X.500 規格に含まれる標準に基づいていますが、大幅に簡略化されています。LDAP は X.500 とは異なり、あらゆるインターネット・アクセスに必要な TCP/IP をサポートします。LDAP は、証明書に関する X.509 規格と対になっているので、証明書を発行する方法として理想的です。

参考項目 「証明書」および「 X.509」

リンク・レベルの暗号化 (LLE: Link-Level Encryption)

ネットワーク・リンク上を行き交うメッセージを暗号化すること。データは、ネットワークへの送信直前に暗号化され、受信直後に解読されます。

リンク・レベルの暗号化は、ワークステーション・クライアント、ドメイン・ゲートウェイ、ブリッジ、および管理ネットワーク・リンクで実行されます。また、対称鍵による暗号化技術である RC4 を使用します。RC4 では、暗号化と解読の際に同じ鍵を使用します。

Domains におけるリンク・レベルのフェイルオーバー

リンク・レベルのフェイルオーバーとは、プライマリ・リンクが失敗したときに、代替のネットワーク・リンクをアクティブするメカニズムです。

リスナ

「ワークステーション・リスナ (WSL)」を参照してください。

リスナ / ハンドラ

「 IIOP リスナ (ISL)」、「 Jolt サーバ・ハンドラ (JSH: Jolt Server Handler)」、「 Jolt サーバ・リスナ (JSL: Jolt Server Listener)」、「 ワークステーション・ハンドラ (WSH)」、「 ワークステーション・リスナ (WSL)」、および 「IIOP ハンドラ (ISH)」を参照。

LMID

「論理マシン ID (LMID: Logical Machine ID)」を参照。

ロード・バランシング

複数のサービス要求をドメイン内のサーバに分散し、最も効率的な方法で要求を処理する機能。具体的には、現在最も処理量が少ないサーバがシステム側で検出され、そのサーバのキューに要求が送信されます。

サービス要求がドメイン・ゲートウェイにルーティングされると、ゲートウェイは、ロード・バランシング・アルゴリズムとデータ依存型ルーティング・アルゴリズムをインプリメントし、サービス要求の適切なルーティング先を決定します。ロード・バランシングとデータ依存型ルーティング・アルゴリズムは、ゲートウェイ共用メモリに保存されているリモート・サービス・テーブル・エントリとリモート・ドメイン・テーブル・エントリの値に基づいています。

ローカル

データ通信では、直接アクセスできるデバイス、つまり通信回路を使用しないでアクセスできるデバイスのこと。

ローカル・アプリケーション名

DDE リスナは、DDE 初期化メッセージで指定されたアプリケーション名を使用して、クライアントがローカルまたはリモートの DDE アプリケーションを検索しているかどうかを判別します。ローカルの DDE ア

プリケーションの構文に含まれているのはアプリケーション名だけです。たとえば、クライアントがローカル・コンピュータの Microsoft Excel を検索している場合、アプリケーション名は EXCEL になります。

ローカル・エリア・ネットワーク (LAN: Local Area Network)

建物内または建物の一区域などの限られた範囲に設置された高速ネットワーク。ブリッジ用のデバイスを使用して、広域ネットワーク (WAN) に接続することもできます。

ローカル・ドメイン

ほかのドメインからアクセスできるアプリケーション (サービスの集合または部分集合) の一部。ローカル・ドメインは常にドメイン・ゲートウェイ・グループによって表現されるため、これらの用語は同義語として扱われます。

ローカル・ファクトリ

(CORBA) BEA Tuxedo ファクトリ・ファインダを介してリモート・ドメインからアクセス可能なローカル・ドメイン内のファクトリ・オブジェクト。

参考項目 「ファクトリ」、「 ファクトリ・ファインダ」、および「 リモート・ファクトリ」

ローカル・ゲートウェイ

ローカルの BEA Tuxedo アプリケーション内の特定のゲートウェイ・グループ (`GWADM` および `GWTDOMAIN` など)。単一の BEA Tuxedo アプリケーションで、複数のローカル・ゲートウェイが実行されている場合があります。

ローカル・ノード

ユーザのワークステーションに接続されたコンピュータ。

ローカル・サービス

ドメイン・ゲートウェイ・グループを介してリモート・ドメインからアクセスできるローカル・ドメインのサービス。

ローカル・システム

マルチシステム環境では、アプリケーション・プログラムを実行中のシステムのこと。ローカル・アプリケーションは、同一システム (ローカル) または別システム (リモート) のデータベースから取得したデータを処理できます。

ローカル・トランザクション

グローバル・トランザクションの代わりに実行される、ローカル・リソース・マネージャ・トランザクション。

参考項目 「ACID 特性」、「 CORBA ORB」、および 「グローバル・トランザクション」

位置制約付きオブジェクト

(CORBA) オブジェクトが存在するアドレス領域の外で呼び出すことができない CORBA オブジェクト。こうしたオブジェクトのアドレス領域外にリファレンスを渡そうとしたり、`CORBA::ORB::object_to_string` を使用してインターフェイスをサポートしているオブジェクトを外部化しようとしたりすると、`CORBA::MARSHAL` システム例外が発生します。

位置透過性

特定のネットワーク・アドレスや物理的な位置を具体的に示さないようにリソース名を定義できること。

ログ・ファイル

オペレーション中に発生したイベントを説明するメッセージ・ファイル。ログ・ファイルは、オペレーション中に頻繁に更新されます。システムの動作やエラーを追跡するのに便利です。

論理マシン ID (LMID: Logical Machine ID)

トランザクション・マネージャ・アプリケーションで使用される処理エレメントに対して割り当てられる論理名。コンフィギュレーション・ファイルで指定されます。

M

makefile

make コマンドによって参照されるファイル。プログラムを実行するために必要な各ファイルの作成方法を make コマンドに通知します。makefile には、ソース・ファイル、オブジェクト・ファイル、および依存関係の情報がリストされています。

管理対象オブジェクト

管理情報ベース (MIB) で定義され、管理対象リソース (プロセス、ハードウェアの一部、システム・パフォーマンス属性など) の機能を表し、管理コンソールの代わりに BEA Tuxedo TMIB などの管理インフラストラクチャを介して制御されるソフトウェア・エンティティ。

参考項目 「管理情報ベース (MIB: Management Information Base)」および「管理対象リソース」

管理対象リソース

管理情報ベース内の管理対象オブジェクトによって表される属性を持つ物理リソース。管理対象リソースは、アプリケーションやキューなどのソフトウェア・エンティティ、またはインターフェイス・カードやハブなどのハードウェア・デバイスです。

管理フレームワーク

分散システムと全社規模のネットワーク上のハードウェアおよびソフトウェア・リソースを一覧表示して、ネットワーク管理者やシステム管理者がリソースを効率的に管理できるようにするソフトウェア。

管理情報ベース (MIB: Management Information Base)

(ATMI および CORBA) BEA Tuxedo システムを構成するクラスと属性の定義を完全に備えた、BEA Tuxedo システムのコンポーネント。BEA Tuxedo システムの管理情報ベースには、総合的な MIB と、Domains や Workstation などの主要なコンポーネント用の MIB があります。ATMI を使って属性の値を設定または変更することにより、BEA Tuxedo システムのコンフィギュレーションや管理をプログラム処理することができます。

参考項目 「Tuxedo MIB」

管理ステーション

SNMP マネージャ・アプリケーションが実行されているマシン。

介在者の攻撃

第三者がマシンをネットワークに入り込ませて、ネットワーク間でやり取りされるすべてのメッセージを捕捉して修正し、再度転送する攻撃。

マッピング

(ATMI) ローカルな値やエンティティを、リモート・システムで意味のある値やエンティティに関連付けること。

(*CORBA*) CORBA では、OMG IDL 文と、OMG IDL 文をコンパイルして作成されるプログラミング言語のコードとの間の関係。たとえば、C++ IDL コンパイラは OMG IDL 文を C++ 言語のバインディングにマップします。

「OMG IDL」を参照。

マーシャリング

ネットワークを介して別のコンピュータに転送できるように、データをバイト・ストリームにパッケージ化すること。

マスク

指定したインスタンスに対してだけアラームが生成されるように、選択した SNMP トラップを非表示にする SNMP の方法。

マスタ・エージェント

管理対象ノード上の SNMP マネージャの唯一の通信ポイント。マスタ・エージェントは SNMP マネージャからのリクエストを受信し、リクエストを満たすために適切なサブエージェントと通信します。

マスタ・ノード

コンフィギュレーション・ファイルの `RESOURCES` セクションで指定されたアプリケーションの `MASTER` ノード。バイナリ形式の `TUXCONFIG` コンフィギュレーション・ファイルのマスタ・コピーが含まれています。実行中のシステムの管理は、`MASTER` ノードから行います。

MD5

Message Digest 5 の略。RFC 1321 で定義されるアルゴリズムで、任意の長さのメッセージを入力として、入力のうちの 128 ビットのメッセージ・ダイジェスト、またはハッシュ値を出力として生成します。MD5 は、サイズの大きいファイルを安全な方法で圧縮してから PKCS などの公開鍵暗号システムで秘密鍵を使って暗号化する場合に適した、デジタル署名アプリケーション用のアルゴリズムです。

メッセージ

アプリケーション間でデータおよび値を送信するための手法。メッセージの内容は、アプリケーションのプロセスに関する統計情報やステータス情報、受信者宛ての指示などです。メッセージは、メッセージ ID データを格納するヘッダと、ユーザ定義の情報を格納する本文で構成されます。

参考項目 「要求」

メッセージ・カタログ

特定の言語、地域、およびコードセット用のプログラム・メッセージ、コマンド・プロンプト、およびプロンプトへの応答が格納された国際化用のファイルまたは記憶域。

メッセージ定義ブロック

メッセージ定義を構成する本文データ全体。内容の例として、コマンド名、サブシステム名、内部用および外部用の推奨事項などがあります。

メッセージ・ダイジェスト

数字の単一文字列で表現したテキスト。one-way ハッシュ関数という式を使用して作成します。秘密鍵を使用してメッセージ・ダイジェストを暗号化すると、デジタル署名が作成されます。つまり、これが電子的な「認証」です。

メッセージ・ダイジェスト・アルゴリズム

任意の長さのメッセージを「メッセージ・ダイジェスト」または「ハッシュ値」と呼ばれる固定長の文字列に短縮する方法。メッセージ・ダイジェスト・アルゴリズムには、指定したメッセージ・ダイジェストに対応するメッセージを検索したり、同じメッセージ・ダイジェストを作成する 2 つの作成元メッセージを検索することがコンピュータ処理ではできない、という性質があります。メッセージ・ダイジェスト・アルゴリズムの例として、MD5 および SHA-1 があります。

メタデータ・インターフェイス

(CORBA) 特定のオブジェクトを説明した情報に関する情報にアクセスするインターフェイス。

method

(CORBA) オブジェクト指向のプログラミング言語では、クラスとの一部として定義され、そのクラスのオブジェクトに含まれるプログラミングされた手順。クラス (つまりオブジェクト) は複数のオブジェクトを持つことができます。オブジェクト内のメソッドはそのオブジェクトに対して既知のデータにしかアクセスできないので、アプリケーション内のオブジェクトの集合内でデータの整合性が保証されます。メソッドは、複数のオブジェクトで再利用できます。

「オペレーション」および「コールバック・ラッパー・オブジェクト」を参照。

MIB

「管理情報ベース (MIB: Management Information Base)」を参照。

MIB **グループ**

OID （または登録）ツリー内の MIB オブジェクトの親オブジェクト。MIB グループにはほかのグループが入る場合や、スカラ・オブジェクトまたは表オブジェクトが入る場合があります。

ミドルウェア

分散されたクライアント / サーバ型アプリケーションの構築に必要なサービスのセット。たとえば、ネットワーク上にほかのプログラムを配置するためのサービス、これらのプログラムとの通信を確立するためのサービス、アプリケーション間で情報を交換するためのサービスなどがあります。ミドルウェアのサービスは、異なるプラットフォームでの不均衡を解決し、複数のベンダ・ソフトウェアやオペレーティング・システムが混在するネットワーク上で同一の認証モデルを実現するためにも使用できます。

移行

ある LMID から別の LMID にサーバまたはサーバ・グループを再配置すること。移行は計画的に行い、コンフィギュレーション・ファイルで指定する必要があります。

モデル

物事を簡単に示すため、具体的な内容を抽象化して表したパラダイム。

モデル化

アーキテクチャの開発、シミュレーション、およびコンピュータ・システムで使用されるデザイン技術。

モジュール

特定のトピックおよびトピックに関連するインターフェイスの情報が格納されたコード。たとえば、銀行での現金引き出し操作を記述したコードを 1 つのモジュールに格納できます。

MOPS

Management operations per second の略。

MP **モデル**

複数のコンピュータで実行される BEA Tuxedo アプリケーションのコンフィギュレーション。MP コンフィギュレーションを構成する複数のマシンには、2 つ以上のユニプロセッサ、1 つまたは複数のマルチプロセッサ、またはユニプロセッサとマルチプロセッサの組み合わせを設定できます。

マルチコンテキスト化されたプロセス

複数のアプリケーションに接続されるか、同じアプリケーションに対して複数の接続が確立されるか、またはその両方を実現した BEA Tuxedo のプロセス。

マルチドメイン・クライアント

複数の BEA Tuxedo アプリケーションに関連付けられた BEA Tuxedo クライアント。

複数の受信アドレス

別々のネットワーク上で複数のアドレスが利用できると、1 つの仮想回線で処理が中断されても、ほかの回線で処理を続行できます。`BRIDGE` の再接続が不可能になるのは、コンフィギュレーション済みのすべてのネットワークで障害が発生したときだけです。たとえば、優先順位の高

いネットワーク上で障害が発生すると、これより優先順位の低い代替ネットワークに切り換えられます。優先順位の高いネットワークが再び使用可能になると、このネットワークに処理が戻されます。

多重仮想記憶装置 (MVS: Multiple Virtual Storage)

IBM の主要なメインフレーム・オペレーティング・システムの 1 つ。MVS/XA は、Extended Architecture の略であり、MVS/ESA は、Enterprise Systems Architecture の略です。

マルチプロセッサ

複数の処理要素で構成されるコンピュータ。各要素には専用のメモリがあります。

マルチプログラミング

複数のコンピュータ・プログラムを同時に実行すること。

マルチスレッド処理

いくつかのトランザクションで 1 つのプロセスを使用すること。

マルチスレッド CORBA **サーバ・アプリケーション**

(CORBA) 複数の独立スレッドを使用するアプリケーションの設計。一般に、アプリケーション内で並行性が実現されるので、全般的なスループットが向上します。複数のスレッドを使用すると、各スレッドが複数の独立したタスクを並列に処理する効率的なアプリケーションを構築できます。

参考項目 「CORBA サーバ・アプリケーション」、「スレッド」、および「スレッド処理」

相互認証

通信を予定した二者が互いの ID を証明し合うプロセス。通常、このプロセスはクライアントとターゲット間を安全に関連付けるための前提条件です。相互認証により、両者がセキュリティ付きのトランザクションを実行できます。

参考項目 「認証」

MVS

「多重仮想記憶装置 (MVS: Multiple Virtual Storage)」を参照。

N

ネーム・バインディング

(CORBA) オブジェクト・リファレンスに名前を関連付けること。ネーム・バインディングはネーミング・コンテキストに格納されます。

名前解決

(CORBA) 名前をオブジェクト・リファレンスに変換すること。

ネーム・サーバ

(CORBA) サービス名を物理アドレスに透過的にマッピングする BEA Tuxedo システムのソフトウェア・コンポーネント。これでユーザは、内部識別子の代わりにサービス名を使って通信できます。

名前空間

(CORBA) グループ化したネーミング・コンテキストのコレクション。

ネーミング・コンテキスト

(CORBA) 一意な名前の関連性の集合を格納したオブジェクト。

「CORBA ネーミング・サービス」を参照。

ネーミング・サービス

「CORBA ネーミング・サービス」を参照。

ネイティブ・クライアント

「クライアント」を参照。

ネイティブ・クライアント・アプリケーション

(CORBA) CORBA サーバ・アプリケーションと通信するために OMG IDL 文に定義されたオペレーションを呼び出すクライアント・アプリケーション。サーバ・アプリケーションが属する BEA Tuxedo ドメインを基準として、クライアント・アプリケーションはネイティブ (つまりローカル) またはリモートとなります。リモート・クライアント・アプリケーションとネイティブ・クライアント・アプリケーションは同じです。アプリケーションが BEA Tuxedo ドメインで実行中のマシン上にあ

るかどうかに応じて、要求の処理方法は異なり、透過的に処理されるかどうかも変わります。ネイティブ・クライアント・アプリケーションは、必ず BEA Tuxedo ドメイン内のマシン上にあります。

「外部クライアント・アプリケーション」、「リモート・クライアント・アプリケーション」、および「OMG IDL」を参照。

ネイティブ共同クライアント / サーバ・アプリケーション

(CORBA) BEA Tuxedo ドメイン内にある共同クライアント / サーバ・アプリケーション。C++ ネイティブ共同クライアント / サーバ・アプリケーションは、`buildobjclient` コマンドでビルドします。BEA Tuxedo ソフトウェアは、Java ネイティブ共同クライアントサーバ・アプリケーションをサポートしていません。

「共同クライアント / サーバ・アプリケーション」を参照。

ネイティブ・ノード

BEA Tuxedo ソフトウェアのフル・コンポーネントを装備し、コンフィギュレーション内のすべてのネイティブ・ノードの掲示板と同じ掲示板にアクセスする、BEA Tuxedo コンフィギュレーション内のマシン (アプリケーションの管理ドメインの一部)。

ネットワーク

- 互いに接続されたノードのグループ
- データ・ステーション間を接続する装置の集合。
- サーバと通信するための通信パス。

ネットワーク・エージェント (NA: Network Agent)

BEA Tuxedo の ワークステーション・ハンドラ (WSH) に相当する BEA TOP END の用語。

ネットワーク・インターフェイス (NI: Network Interface)

BEA Tuxedo の Bootstrap 環境オブジェクトに相当する BEA TOP END の用語。

ノード・マネージャ (NM: Node Manager)

BEA Tuxedo の Bulletin Board Liaison (BBL) に相当する BEA TOP END の用語。

ネットワーク・プロバイダ

ネットワーク経由でデータ通信を行うための、トランスポート・レベル以下で使用されるプロトコル。通常、トランスポート・インターフェイスでプログラムを使用してネットワーク・プロバイダにアクセスします。ネットワーク・プロバイダの例として、TCP/IP や StarLAN があります。

NLS

UNIX システムのネットワーク・リスナ・サービス。

ノード

ネットワーク上のポイント。BEA Tuxedo システムのアプリケーションに参加するコンピュータを指す場合もあります (UNIX オペレーティング・システムの単一のインスタンス)。マルチプロセッサ・システムでは、複数のノードを設定することもできます。

非ブロッキング・モード

処理の完了を待たずに次の操作に進む、という非同期的なメッセージ送信方法。

非マスタ・ノード

MASTER ノードとして指定されていない BEA Tuxedo アプリケーションのノード。

非分断

マスタ・ノード上の DBBL と引き続き通信できるネットワークのうち、分断された部分を示す用語。

n **層型のクライアント / サーバ**

アプリケーション・ロジックを 3 つまたはそれ以上の環境 (デスクトップ・コンピュータ、1 つまたはそれ以上のアプリケーション・サーバ、およびデータベース・サーバ) に分散するというアプリケーション開発の手法。n 層型のクライアント / サーバの主な利点は、クライアント / サーバのアーキテクチャの利点が企業に応用できるレベルに拡張されていることです。また、管理がしやすい、スケーラビリティがある、セキュリティ機能がある、高いパフォーマンスが実現できるという利点もあります。

O

オブジェクト

(CORBA) 状態、動作、および ID で定義されるエンティティ。プロパティとも呼ばれるこれらの属性は、オブジェクトのオブジェクト・システムによって定義されます。

「Remote Method Invocation (RMI)」および 「CORBA オブジェクト」を参照。

オブジェクトの活性化

(CORBA) CORBA オブジェクトをクライアント・アプリケーションからの呼び出しを受け付ける状態にすること。オブジェクトには、メモリ内で使用可能なメソッドおよび状態が必要です。

CORBA オブジェクトを使用する場合は、POA のアクティブ・オブジェクト・マップと TP フレームワーク内でオブジェクト ID とサーバントとを関連付けることです。CORBA オブジェクトが活性化されると、TP フレームワークは永続的な記憶域から CORBA オブジェクトの状態を再ロードし、オブジェクトを利用可能にしてクライアント・アプリケーションからのリクエストを満たします。

「メタデータ・インターフェイス」、「オブジェクトの非活性化」、「オブジェクト ID (OID)」、「オブジェクト・リファレンス」、「ポータブル・オブジェクト・アダプタ (POA: Portable Object Adapter)」、「サーバント」、および 「アクティブ・オブジェクト・マップ」を参照。

オブジェクト・ブリッジ

(CORBA) オブジェクト・システムの相互運用性を実現するフレームワークを提供する Visual Edge Software, Ltd. 製ソフトウェア。

OBV

リファレンスではなく、値によってオブジェクトを渡す方法。受信側にはオブジェクトの状態の記述が送信されます。値によるオブジェクトの受信側は、送信者とは別の ID を使用してその状態の新しいインスタンスを作成します。値によってオブジェクトを渡すと、2 つのインスタンス間には何の関係もないものと見なされます。

オブジェクトの非活性化

(CORBA) CORBA では、POA のアクティブ・オブジェクト・マップと TP フレームワーク内のオブジェクト ID とサーバントとの関連性を削除すること。オブジェクトを非活性化したら、オブジェクトを活性化してからでないと、このオブジェクト ID を持つオブジェクト・リファレンスのクライアント呼び出しは満たされません。

EJB では、EJB コンテナによってアクティブ・オブジェクト・マップ内のオブジェクト ID とインスタンスとの関連性を削除すること。オブジェクトを非活性化したら、オブジェクトを活性化してからでないと、このオブジェクト ID を持つオブジェクト・リファレンスのクライアント呼び出しは満たされません。

「オブジェクトの活性化」、「オブジェクト ID (OID)」、「オブジェクト・リファレンス」、「パッシベーション」、「ポータブル・オブジェクト・アダプタ (POA: Portable Object Adapter)」、および 「アクティブ・オブジェクト・マップ」を参照。

オブジェクト・ハンドル

(CORBA) 移植可能な方法でオブジェクトを識別します。ハンドルは連続させることができるので、オブジェクト・ハンドルを格納してから、別の Bean またはオブジェクトが別のプロセスまたはシステムで使用することができます。

オブジェクト ID (OID)

「オブジェクト識別子 (OID)」を参照。

オブジェクト識別子 (OID)

(CORBA) MIB 内の各オブジェクトに割り当てられた一意な数字。これらの OID は種類別になっており、ツリー構造をなしています。エージェントが特定のオブジェクトにアクセスする場合、MIB ファイル内の OID ツリーをたどって目的のオブジェクトを見つけます。OID は、OID ツリーのルートから目的のオブジェクトまでの一意なパスを指定することでオブジェクトを識別します。

オブジェクト・インプリメンテーション

(CORBA) インターフェイス向けに定義したオペレーションを実装するために作成したコード。

参考項目 「インターフェイス」

オブジェクト・インターフェイス

(CORBA) アプリケーションの OMG IDL 文で定義したオブジェクトのインターフェイス。オブジェクト・インターフェイスは、オブジェクトで実行可能なオペレーションのセットおよび属性を識別します。たとえば、窓口オブジェクトのインターフェイスは、引き出し、振替、預金など、そのオブジェクトで実行可能なオペレーションを識別します。`Tobj::TransactionCurrent` は、BEA Tuxedo ソフトウェアに用意されているオブジェクト・インターフェイスの一例です。

「OMG IDL」、「オペレーション」、および 「CORBA オブジェクト」を参照。

Object Management Group (OMG)

(CORBA) オブジェクト指向アプリケーション開発用の共通のフレームワークを提供するために、業界の指針とオブジェクト管理仕様を制定する国際的組織。OMG CORBA (Common Object Request Broker Architecture) では、CORBA オブジェクト・モデルを指定しています。

オブジェクト・モデル

(CORBA) アプリケーションまたはシステムの設計全体がオブジェクトとして反映されたモデル。

オブジェクト・リファレンス

(CORBA) 分散 ORB システム内のオブジェクトのインスタンスを一意に指定する識別子。

オブジェクト・リクエスト・ブローカ

「CORBA ORB」を参照。

オブジェクト・システム

(CORBA) システム固有の標準に従って、オブジェクトのコレクションを格納、操作、および使用するソフトウェア・システム。オブジェクト・システムは、オブジェクト間の情報の交換方法と、オブジェクト・モデル (CORBA COM、EJB、RMI など) に従ったオブジェクトの実装方法を指定します。

「 会話」、「 オブジェクト・モデル」、および 「Component Object Model (COM)」を参照。

オブジェクト・トランザクション・サービス

「CORBA オブジェクト・トランザクション・サービス (OTS: Object Transaction Service)」を参照。

オクテット

- 8 ビットで構成される単位。バイト。
- 8 つのバイナリ要素で構成されるバイト。

OID

「オブジェクト識別子 (OID)」を参照。

OLE

(CORBA) Object Linking and Embedding の略。OLE をサポートする ActiveX の一部。

「ActiveX」を参照。

OLE **オートメーション**

(CORBA) ActiveX オブジェクトを定義したアプリケーションの外側から ActiveX オブジェクトを操作する方法として Microsoft が提供する技術。

「 アプリケーション」および 「ActiveX」を参照。

OLTP

「オンライン・トランザクション処理 (OLTP: Online Transaction Processing)」を参照。

OMG IDL

(CORBA) Object Management Group Interface Definition Language の略。オブジェクトのインターフェイス、つまりオブジェクトで実行可能なオペレーションなど、オブジェクトの特徴や動作を記述するために OMG によって指定された定義言語。

参考項目 「オペレーション」

オンライン・トランザクション処理 (OLTP: Online Transaction Processing)

- 端末またはワークステーションのユーザからアプリケーション・プログラムにメッセージが送信されると、リアル・タイムでデータベースが更新される、というデータ処理の手法。

- 処理が迅速かつリアル・タイムで行われ、自己回復機能、記録機能、および整合性の保証機能が備えられたパフォーマンス・クリティカルな環境で作業が実行されること。

オープン・フレームワーク

開発者がプラグ・アンド・プレイ形式で簡単にソフトウェア・コンポーネントを削除したり、置換したりできるソフトウェア・インフラストラクチャ。

オープン・システム

異機種ソフトウェアに対して、指定された共通の標準をインプリメントするシステム。オープン・システムの標準をインプリメントすると、異機種コンピュータ間での通信を実現できます。

開放型システム間相互接続のコミットメント、同時制御、および回復制御 (OSI CCR: Open Systems Interconnect Commitment, Concurrency, and Recovery)

- グローバル・トランザクションのブランチをコミットまたはロールバックするために使用する、サービスおよびプロトコル用の ISO 標準プロトコル (ISOIIEC 9804)。
- ISOIIEC 9804 標準のソフトウェア・インプリメンテーション。

開放型システム間相互接続 (OSI: Open Systems Interconnection)

異機種のコンピュータ・システム間の通信を容易にするためのコンソーシアム。

開放型システム間相互接続のトランザクション処理 (OSI TP: Open Systems Interconnect Transaction Processing)

- 異機種コンピュータのクライアントおよびサービス・ルーチン間でダイアログを確立し、メッセージを渡すために使用する、サービスおよびプロトコル用の ISO の標準 (ISOIIEC 10026-2)。
- ISOIIEC 10026-2 標準のソフトウェア・インプリメンテーション。

オペレーション

(CORBA) オブジェクトによって実行可能なアクション。たとえば、ファイル・オブジェクトでは、オープン、クローズ、読み取り、印刷などのオペレーションを要求できます。

参考項目 「オブジェクト」

ORB

「CORBA ORB」を参照。

ORBMain **モジュール**

(CORBA) BEA Tuxedo サーバ・アプリケーション・プロセスのメイン手順。BEA Tuxedo ソフトウェアは ORBMain モジュールを備えています。このモジュールは変更しないでください。サーバ・アプリケーションのビルド手順では、ORBMain モジュールがサーバ・アプリケーション・プロセスに自動的に組み込まれます。ORBMain モジュールは、`buildobjserver` コマンドによって TP フレームワークを使用するサーバに提供されます。ただし、共同クライアント / サーバ・アプリケーションでは、独自のメイン手順を用意し、`buildobjclient` コマンドに `-P` スイッチを付ける必要があります。

OSI

「開放型システム間相互接続 (OSI: Open Systems Interconnection)」を参照。

OSI CCR

「開放型システム間相互接続のコミットメント、同時制御、および回復制御 (OSI CCR: Open Systems Interconnect Commitment, Concurrency, and Recovery)」を参照。

OSI TP

「開放型システム間相互接続のトランザクション処理 (OSI TP: Open Systems Interconnect Transaction Processing)」を参照。

OTS

「CORBA オブジェクト・トランザクション・サービス (OTS: Object Transaction Service)」を参照。

アウトバウンド IIOP

(CORBA) クライアントのコールバックをサポートする BEA Tuxedo ソフトウェアの機能。アウトバウンド IIOP はアウトバウンド中間ゲートウェイを ISL/ISH に追加します。

BEA Tuxedo システムは、以下の 3 種類のアウトバウンド IIOP をサポートしています。

1. 非対称アウトバウンド IIOP

別の接続を使用した、ISH に接続されていない共同クライアント / サーバ・アプリケーションとのアウトバウンド IIOP。BEA Tuxedo ソフトウェアのこの機能は、GIOP 1.0、GIOP 1.1、および GIOP 1.2 のクライアント・アプリケーション、サーバ・アプリケーション、および共同クライアント / サーバ・アプリケーションについてサポートされています。

2. 双方向アウトバウンド IIOP

ISH に接続したリモートの共同クライアント / サーバ・アプリケーションとのアウトバウンド IIOP。アウトバウンド・コールバックでは、インバウンド呼び出し用に共同クライアント / サーバ・アプリケーションが最初に使用した接続を再利用します。この機能は、BEA Tuxedo C++ GIOP 1.2 のクライアント・アプリケーション、サーバ・アプリケーション、および共同クライアントサーバ・アプリケーションについてのみサポートされています。

3. デュアル・ペア接続のアウトバウンド IIOP

ISH に接続したリモートの共同クライアント / サーバ・アプリケーションとのアウトバウンド IIOP。双方向アウトバウンド IIOP と異なり、アウトバウンド・コールバックでは、インバウンド呼び出し用に共同クライアント / サーバ・アプリケーションが最初に使用した接続とは別の接続を使用します。BEA Tuxedo ソフトウェアのこの機能は、GIOP 1.0、GIOP 1.1、および GIOP 1.2 のクライアント・アプリケーション、サーバ・アプリケーション、および共同クライアント / サーバ・アプリケーションについてサポートされています。

送信時接続

接続が自動的に再試行された場合、リモート・ドメインへ最初に要求が送信された場合、管理者によって `dmadmin(1)` `connect` コマンド・シーケンスが実行された場合、のいずれかの場合に生成されるローカル・ゲートウェイからの接続。

帯域外データ

BEA Tuxedo システムがサポートする通常のクライアント / サーバ通信チャネル外で、BEA Tuxedo システムから配信されたデータ。

P

パラレル・データ回線

パラレル・データ回線を使用すると、データを複数の回線に同時に送信することができます。データを並行処理できるように回線を設定しておくと、ネットワーク・トラフィックは最も大きいネットワーク・グループ番号 (NETGRPNO) が付いた回線へ流れるようスケジューリングされます。この回線がビジー状態の場合、トラフィックは次にネットワーク・グループ番号が大きい回線上に自動的にスケジューリングされます。すべての回線がビジー状態の場合は、回線が使用可能になるまでデータはキューに入れられます。

分断

ネットワーク・アプリケーションの 1 つまたは複数のアクティブ・ノードが、LAN の障害などが原因でほかのアクティブ・ノードと通信できない状態のこと。

パス・フレーズ

通常は ID を提供するために指定する英数字とその他の文字からなる文字列。パス・フレーズは一般にパスワードよりも長い文字列です。大文字と小文字を組み合わせた複数の語に加えて、区切り文字を含んでいなければなりません。パス・フレーズは覚えやすく、かつパスワードよりも第三者が推測しにくいものにする必要があります。

パッシベーション

Bean の状態を非活性化すること。ステートフル・セッション Bean またはエンティティ Bean の場合、通常、パッシベーションでは Bean の状態データを永続的記憶域に書き込みます。この状態は、再度活性化するときに復元できます。ステートフル・セッションおよびエンティティ Bean の場合、パッシベーションによってオブジェクトが非活性化されます。

参考項目 「オブジェクトの非活性化」

永続オブジェクト

オブジェクト・リファレンスを作成したプロセスから独立して存在するオブジェクト。

「一時オブジェクト」および 「オブジェクト・リファレンス」を参照。

PID

「プロセス ID (PID)」を参照。

PIDL (Pseudo-IDL)

(CORBA) CORBA 疑似オブジェクトを記述するためのインターフェイス定義言語。各言語のマッピング (IDL と C++ または Java プログラミング言語のマッピングなど) は、疑似オブジェクトを言語固有の構造体にマップする方法を示します。PIDL マッピングは、通常の CORBA オブジェクトのマッピングに適用される規則に従っていても従っていなくてもかまいません。

PKCS-7

「Public-Key Cryptography Standard 7 (PKCS-7)」を参照。

平文

暗号化されていないテキスト。

プラットフォーム

アプリケーションをサポートするハードウェア、オペレーティング・システム、およびウィンドウ・システム・ソフトウェアの組み合わせ。

POA

「ポータブル・オブジェクト・アダプタ (POA: Portable Object Adapter)」を参照。

方針

「 SecurityCurrent」、「 トランザクション方針」、および 「活性化方針」を参照。

ポーリング

(CORBA) 管理対象オブジェクトの値が指定されたしきい値を超えていないかどうか調べるために、マネージャがエージェントに一定の間隔で問い合わせること。エージェントは、指定された管理対象オブジェクトの値を報告します。

移植性

手間のかかる作業を行わずに、プラットフォーム間で簡単にアプリケーションを移動できること。

ポータブル・オブジェクト・アダプタ (POA: Portable Object Adapter)

(CORBA) CORBA サーバ・アプリケーションの実行可能イメージに組み込まれる関数のランタイム・ライブラリ。POA は、アプリケーションが使用するすべてのオブジェクトへのオブジェクト・リファレンスを作成および管理します。また、POA はオブジェクトの状態も管理し、永続オブジェクトをサポートするインフラストラクチャを提供して、異なる ORB 製品間でのオブジェクト・インプリメンテーションの移植性を実現します。

BEA Tuxedo サーバ・アプリケーションの手順では、POA がサーバ・アプリケーションに組み込まれます。BEA Tuxedo TP フレームワークは、BEA Tuxedo サーバ・アプリケーションと POA とのすべての対話を自動的に処理します。ただし、共同クライアント / サーバ・アプリケーションは POA と直接対話します。

「 オブジェクト・リファレンス」、「 状態」、「 WebLogic Express」、および 「CORBA オブジェクト」を参照。

ポート番号

論理通信チャネルを指定し、特定の接続をほかの接続から識別する TCP/IP ホストのエンティティ。TCP/IP サーバは、指定されたポートで接続の受信を「リッスン」します。ホストの IP アドレスと、サーバの目的のポート番号が指定されると、TCP/IP クライアントはサーバへの接続を開始します。

プラグマ

(CORBA) IDL ファイルをコンパイルするときに特定のオペレーションを実行するための、IDL コンパイラに対する指針。たとえば、プラグマ Prefix は IDL インターフェイスのインターフェイス・リポジトリに影響します。

プレコミット

安定記憶域のデータをコピーするプロセス (データを回復可能にするためリソース・マネージャで使用)。

プライマリ・リモート・ドメイン

最も優先順位の高いリモート・ドメイン。使用可能な状態では、常に使用されます。

プリンシパル

(ATMI) セキュリティ目的で認証を受けたユーザ。

(CORBA) システムのリソースを使用可能なユーザまたはプログラムのエンティティ。

Principal Authenticator **オブジェクト**

(CORBA) 指定したプリンシパルの Credentials を作成するアプリケーションから見えるオブジェクト。認証を受ける必要があって、まだ受けていないユーザまたはプリンシパルは、Principal Authenticator オブジェクトを使用します。

参考項目 「認証」および「 オブジェクト」

秘密鍵

セキュリティ付きのメッセージをやり取りする当事者にしかわからない暗号化 / 暗号解読用鍵。

プライベート MIB

プライベート MIB ディレクトリで定義された MIB。

手順

コンピュータに対する特定のタスクを実行するための指示を順番に示したもの。

プロセス ID (PID)

プロセスを識別するための一意な番号。

Process-Entity **デザイン・パターン**

クライアント・アプリケーションがサーバ・マシン上に格納されているデータベース・レコードとやり取りする必要がある場合に性能を向上するためのデザイン・パターン。

参考項目 「Client Data Caching デザイン・パターン」および「 デザイン・パターン」

プロファイル

クライアントまたはユーザに関する情報のセット。プロファイルには、サーバ側で必要な、クライアントまたはユーザを認識するための情報が格納されています。

プロトコル

- 通信リンク上で送受信するメッセージの形式や送信時期を規定した規則のセット。ネットワーク・プロトコルの例として、TCP/IP があります。
- 通信を行い、情報を交換するための 2 つのシステムが追加された「規則」のセット。

プロバイダ

OSI 通信プロトコルの第 4 層を介してネットワーク機能を提供する通信製品。

Pseudo-IDL

「PIDL (Pseudo-IDL)」を参照。

疑似オブジェクト

(CORBA) IDL で記述する CORBA オブジェクトとほぼ同様のオブジェクト。ただし、CORBA オブジェクトと異なり、オブジェクト・リファレンスを使用して渡すことができず、ナロー変換や文字列化も行えません。

DII インターフェイスは疑似オブジェクトの例ですが、DII インターフェイスはライブラリとして実装され、IDL インターフェイスの疑似オブジェクトとして OMG 仕様でより厳密に説明されています。疑似オブジェクト向けの IDL は、疑似オブジェクトが定義されていることを示す PIDL と呼ばれます。

参考項目 「PIDL (Pseudo-IDL)」

公開鍵

メッセージの暗号化および解読用として秘密鍵と組み合わせて認証局から提供される値。

公開鍵アルゴリズム

公開鍵または秘密鍵を使って、データを暗号化したり、解読するためのアルゴリズム。秘密鍵は通常、メッセージ・ダイジェストを暗号化するために使用されます。このようなアプリケーションでは、公開鍵アルゴリズムを「メッセージ・ダイジェスト暗号化アルゴリズム」と呼びます。公開鍵は通常、内容暗号化鍵、またはセッション・キーの暗号化に

使用されます。このようなアプリケーションでは、公開鍵アルゴリズムを「キー暗号化アルゴリズム」と呼びます。公開鍵アルゴリズムの例として、RSA があります。

Public-Key Cryptography Standard 7 (PKCS-7)

非公式のコンソーシアム (元の参加者は Apple、Microsoft、DEC、Lotus、Sun、および MIT) の協力のもと、RSA Laboratories が開発した公開鍵による暗号化標準の 1 つ。PKCS-7 は、デジタル署名や暗号化など、拡張された暗号化メッセージの一般的な構文を定義します。BEA Tuxedo の公開鍵のセキュリティは PKCS-7 標準に準拠しています。

公開鍵による暗号化

非対称鍵の組み合わせを使用して暗号化と解読を行う技術。鍵は、公開鍵と秘密鍵を組み合わせたものです。公開鍵は、広い範囲に配布することにより公開できます。秘密鍵が公開されることはなく、常に秘密に扱われます。

公開鍵によるセキュリティ機能

公開鍵暗号化技術を基に構築された BEA Tuxedo のセキュリティ機能。公開鍵暗号化を使用して、BEA Tuxedo アプリケーションのクライアントとサーバ間でエンド・ツー・エンドのデジタル署名とデータの秘密性を実現します。この機能は、PKCS-7 標準に準拠しています。

パブリッシュ

サブスクライバがイベントを利用できるように、構造化イベントをイベント・チャネルにパブリッシュすること。

Q

キュー

サーバに対する要求の配信を時間ごとに管理する単純なデータ構造。キューに登録された要素は、指定した優先順位で並べ替えることができます。クライアントが要求をキューに登録したら直ちにサーバが取り出す、という操作をバッチで処理したり、定期的に行うことができます。

R

RC2

Rivest's Cipher 2 の略。40 ～ 128 ビットの範囲で、暗号鍵のサイズを変更できるブロック暗号方式。DES より高速であり、40 ビットの暗号鍵は輸出できます。米国籍の企業の海外子会社および海外支店であれば、56 ビットの暗号鍵を使用することができます。BEA Tuxedo の公開鍵セキュリティ機能では暗号鍵の長さを 128 ビットに制限していますが、米国では実質的に、RC2 にはどんな長さの鍵を使用することもできます。

RC4

Rivest's Cipher 4 の略。バイトごとに暗号処理を行うストリーム型の暗号方式。暗号鍵の長さは自由に変えることができます。RC4 は、DES の約 10 倍の速さの対称鍵または秘密鍵システムであり、56 ビットの暗号鍵は輸出できます。BEA Tuxedo のリンク・レベルの暗号化では、暗号化キーの長さを 128 ビットに制限していますが、米国では実質的に、どんな長さの鍵を RC4 に使用することもできます。

ReceivedCredentials **オブジェクト**

(CORBA) アプリケーションどうしの安全な関連付けを表すオブジェクト。ReceivedCredentials オブジェクトには、その関連付けのプロパティが含まれます。

参考項目 「オブジェクト」

RECONNECT **クライアント**

アイドル状態のネットワーク接続が、指定された時間を過ぎてから切断されても、BEA Tuxedo のユーザ・コンテキストではアクティブな状態の Jolt クライアント。

レコード

BEA Tuxedo のローカルまたはリモートの領域外、または別のシステム上にある入力データまたは出力データ。

回復

指定したトランザクションを完了するために、コーディネータまたはパーティシパントから出される要求。

回復

トランザクション・システムでは、障害発生時に、直前にコミットされた状態にシステムを復元し、整合性の取れた状態にできること。分散システムの回復処理では、分散コンポーネントを再同期化する処理が含まれます。システムがいったん回復すると、処理が再開し、障害発生時にアボートされたトランザクションが再びサブミットされます。

リレーショナル・データベース

データ・アイテム間の関係に従って構成されたデータベース。データ間の関係は、テーブルによって表現されます。このテーブルの対応する属性を操作して、データ項目にアクセスします。アクセス・パスは、アクセス時に決まります。

相対 OID

(CORBA) OID ツリーのルートの下にあるいずれかのノードを起点として管理対象オブジェクトまでのパスを指定するオブジェクト識別子 (OID)。

信頼性

システム (またはシステムの一部) が、複数回の試行に対して、予期しない結果を生成することなくパフォーマンス仕様の条件に合った正しい出力を生成すること。

リモート

クライアントがネットワーク経由で使用できるサービスまたはコンピュータ。

リモート・クライアント

「クライアント」を参照。

リモート・クライアント・アプリケーション

「CORBA リモート・クライアント・アプリケーション」を参照。

リモート・ドメイン

(ATMI) ローカル・ドメイン・ゲートウェイ・グループを介してアクセスされるアプリケーションの一部。

リモート・ファクトリ

(CORBA) BEA Tuxedo ファクトリ・ファインダを介してアプリケーションからアクセス可能なリモート・ドメイン内のファクトリ・オブジェクト。

参考項目 「ファクトリ」、「 ファクトリ・ファインダ」、「 ローカル・ファクトリ」、および「 CORBA ドメイン」

リモート・ファイル共用 (RES: Remote File Sharing)

ネットワーク経由でリモート・ファイルへのアクセスを提供する UNIX システムの機能。

リモート・ゲートウェイ

リモートの BEA Tuxedo アプリケーション内の特定のゲートウェイ・グループの機能。

リモート共同クライアント / サーバ・アプリケーション

(CORBA) BEA Tuxedo ドメイン外にある CORBA 共同クライアント / サーバ・アプリケーション。リモート共同クライアント / サーバ・アプリケーションでは、BEA Tuxedo TP フレームワークを使用せず、クライアント・アプリケーションと ORB との間でより直接的に対話する必要があります。リモート共同クライアント / サーバ・アプリケーションをビルドするには、`buildobjclient` コマンド、または Java クライアント・アプリケーションのコマンドを使用します。

「 クライアント・アプリケーション」、「 WebLogic Express」、および「CORBA ORB」を参照。

Remote Method Invocation (RMI)

(CORBA) リモート・オブジェクトにアクセスするための Java 固有の API。

リモート・ノード

ネットワーク上のコンピュータのうち、ユーザのワークステーションに接続されていないコンピュータ。

リモート・プロシージャ・コール (RPC: Remote Procedure Call)

非ローカルなプログラムまたはアドレス領域で実行するローカル・プロシージャ・コール。アプリケーション・ロジックをクライアントとサーバに分割し、どちらか有効な方が利用可能なリソースを使用します。

リモート・サービス

ドメイン・ゲートウェイ・グループを介してローカル・アプリケーションからアクセスできるリモート・ドメインのサービス。

リモート・サービス・ファン・アウト

複数のリモート・ドメインからインポートされたリモート・サービスが、ローカル・ドメイン・ゲートウェイによって BEA Tuxedo 掲示板で 1 度だけ宣言されるコンフィギュレーション。リモート・サービスは、複数のリモート・ドメインから同一のサービス名をインポートするゲートウェイ・サーバによってファン・アウトされます。ファン・アウトは、ゲートウェイ共用メモリを介して実現します。

リモート・サービス名

リモート・システム (リモート・ゲートウェイからアクセス可能) によって割り当てられる 1 ～ 16 文字のサービス名。

要求

実行するオペレーションを指定した、クライアントから送信されるメッセージ。CORBA 環境では、メッセージはオブジェクト・リクエスト・ブローカにいったん送信され、そこから要求を満たす適切なサーバ・アプリケーションに転送されます。

参考項目 「クライアント」、「 サーバ」、*および*「 CORBA ORB」

要求元

- クライアントから受信したメッセージを内部形式に変換し、トランザクション要求の送信先とする 1 つまたは複数のサーバを決定し、そのサーバに対して要求を転送する、というプロセス。
- クライアントとして動作するクライアントまたはサーバを示す一般的な用語。

要求レベルのインターセプタ

(CORBA) BEA Tuxedo アプリケーションのクライアントおよびサーバ・コンポーネント間の呼び出しパスに挿入され、オブジェクトを呼び出すたびに ORB によって起動されるユーザ作成アプリケーション。要求レベルのインターセプタを使用すると、セキュリティやコンポーネントの監視などのサービスを、クライアント側またはサーバ側でオブジェクト呼び出しに追加できます。要求レベルのインターセプタによって、サードパーティ製のセキュリティ・プラグイン・ソフトウェアを簡単に使用できるようになります。

リクエスタ

クライアントから受信したメッセージを内部形式に変換し、トランザクション要求の送信先とする 1 つまたは複数のサーバを決定し、そのサーバに対して要求を転送する、というプロセス。

要求 / 応答型の通信

要求メッセージと応答メッセージが 1 対 1 の通信。この種の通信では、クライアントはタスクを要求し、タスクを実行したサーバは応答をクライアントに返します。要求 / 応答型の通信は、同期的または非同期的に実行されます。

要求 / 応答型のサーバ

要求 / 応答型のサービスを提供するサーバ。

要求 / 応答型のサービス

クライアントからの要求を受け取ると開始されるサービス。サービス・ルーチンは、1 つの要求を受け取ると、(最大) 1 つの応答を返します。要求 / 応答型のサービスは 1 つの流れに沿って処理されます。また、処理が完了するまで実行される、リクエスタとの対話は行わない、要求元に戻り値を返す、という 3 つの特徴があります。要求元は、要求 / 応答サービスを同期的または非同期的に実行できます。

Requests For Comments (RFC)

Internet Architecture Board (IAB) によって承認されたインターネット規格として公開されている文書。

リソース・マネージャ (RM: Resource Manager)

情報およびプロセスの集合に対するアクセスを提供する、インターフェイスおよび関連ソフトウェア。例として データベース管理システムがあります。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。

参考項目 「トランザクション・マネージャ (TM: Transaction Manager)」

リソース・マネージャ・インスタンス

リソース・マネージャの特定のインスタンスまたはオカレンス。グローバル・トランザクション内の同一または異なるリソース・マネージャに、多数のオカレンスまたはインスタンスが存在し、それぞれが別のデータを管理している場合があります。各リソース・マネージャ・インスタンスは、自律的で、ローカル・アクセス (ローカルおよびグローバル・トランザクションの両方) や管理などを完全に制御できると見なされます。

応答時間

照会や要求を送信してから応答を受け取るまでにかかる時間。

RFC

「Requests For Comments (RFC)」を参照。

RM

「リソース・マネージャ (RM: Resource Manager)」を参照。

ロールバックする

トランザクション内で更新されたすべてのリソースを元の、つまりトランザクションの開始前の状態に戻してトランザクションを終了すること。

ロールバック

トランザクションの途中で実行されたリソースに対するすべての変更を無効にするか、または元に戻してトランザクションを終了するイベント。

RPC

「リモート・プロシージャ・コール (RPC: Remote Procedure Call)」を参照。

RSA

Rivest 氏、Shamir 氏、Adleman 氏の 3 名によって発明された公開鍵アルゴリズム。RSA は、デジタル署名と暗号化で広く使用されています。RSA は非対称的なアルゴリズムです。つまり、秘密鍵はその所有者によって秘密にされ、対応する公開鍵は広い範囲に配布されます。

RTQ

BEA Tuxedo システムの /Q コンポーネントに相当する BEA TOP END システムのコンポーネント。

実行時のトレーシング機能

アプリケーション間のトランザクションを監視し、必要に応じて開発中または生産過程の分散アプリケーションの問題を解決する BEA Tuxedo の機能。ユーザは、この機能を使って、ハードウェア、オペレーティング・システム、ネットワーク、またはアプリケーション・コードの特定の問題を検出することもできます。

S

スケーラビリティ

開発者が、1 つの解決策を使用して異なるサイズの問題に対処できること。1 つの解決策をあらゆる問題に対して適用できるのは理想的です。ただし実際には、あまり複雑でない問題に対して、より簡単な解決策が適用されるのが普通です。

スカラ・オブジェクト

1 つのインスタンスのみを持つ MIB リーフ・オブジェクト (OID ツリー内に下位のオブジェクトを持たない MIB オブジェクト)。

SCM

「サービス・コントロール・マネージャ (SCM: Service Control Manager)」を参照。

スコープ

(ATMI) クラスを特定の目的のためにアプリケーションに対して強制的に適用すること。

セキュア・ソケット・レイヤ (SSL: Secure Sockets Layer)

公開鍵技術を基にして 2 つの通信アプリケーション間で認証およびデータの暗号化を行うために Netscape によって開発されたトランスポート・レベルの技術。

参考項目 「認証」

セキュリティ

情報が不正に変更または開示されたり、リソースが不正に使用されることを防ぐこと。

SecurityCurrent

(CORBA) システムのセキュリティ機能にアクセスできるようにするオブジェクト。

参考項目 「オブジェクト」

セキュリティ方針

(CORBA) セキュリティ管理者が定義して適用するアプリケーションに関するセキュリティの規則。セキュリティ方針では、ユーザ (またはプリンシパル) のコレクションを定義し、ユーザを認証するための手順を明確に定義します。セキュリティ規則の設定を簡略化するためにグループが使用される場合もあります。

EJB を使用する場合、セキュリティ方針では、JVM でアクセス可能なオペレーションを決定する Java のパーミッションを定義します。

セキュリティ・プリンシパル

セキュリティ・システムによって認識され、認証されるエンティティ。

セキュリティ・サービス

「CORBA セキュリティ・サービス」を参照。

セキュリティ・サービス・プロバイダ・インターフェイス (SSPI: Security

Service Provider Interface)

(CORBA) 複数のベンダから提供されるセキュリティ・コンポーネントを BEA Tuxedo のセキュリティ・サービスに統合することを可能にするインターフェイス。

サーバント

(CORBA) アプリケーションの OMG IDL 文で定義するインターフェイスを実装するクラスのインスタンス。サーバントには、1 つまたは複数の CORBA オブジェクトのオペレーションを実装するメソッドのコードが格納されます。

「インプリメンテーション・コード」、「インスタンス」、「メタデータ・インターフェイス」、「OMG IDL」、「オペレーション」、および「CORBA オブジェクト」を参照。

サーバント・ファクトリ

(CORBA) サーバントを自動的にインスタンス化するための BEA Tuxedo Java サーバ・アプリケーションの機能。BEA Tuxedo C++ サーバと異なり、Java サーバではコールバックしなくてもサーバントをインスタンス化できます。

サーバント・プール

(CORBA) サーバントと特定のオブジェクト ID との関連付けが解除されてからも、BEA Tuxedo サーバ・アプリケーションがメモリ内にサーバントを保持することを可能にする BEA Tuxedo (C++) ソフトウェアの機能。

「サーバント」および「オブジェクト識別子 (OID)」を参照。

サーバ

次の手順でクライアント / サーバ型のアーキテクチャでサービスを提供するソフトウェア・プログラム。

1. 特定のサービスに対する要求をクライアント (またはクライアントの役割を持つ機能) から受信します。

2. 要求を満たすサービス・ルーチンをディスパッチします。

3. 要求されたサービスが正常に実行されたかどうかを示し、サービスの実行結果を通知する応答を元のリクエスタに送信します。

参考項目 「クライアント」

サーバ抽象化

アプリケーションは、サーバ・プロセスの作成中に、サービス・ルーチンと BEA Tuxedo システムの `main()` を結び付けます。BEA Tuxedo システムの `main()` を実行すると、サーバの初期化および終了を実行でき、着信した要求を受け取ってそれらをサービス・ルーチンにディスパッチできます。これらの処理はすべて、アプリケーションに対して透過的に行われます。

サーバ・アプリケーション

(CORBA) BEA Tuxedo ソフトウェアで使用するために作成され、クライアント・アプリケーションから要求されたタスクを実行するプログラム。

「ローカル・ファクトリ」を参照。

サーバ記述ファイル

(CORBA) Java サーバ・アプリケーションで実装されたインターフェイスに対してデフォルトの CORBA 活性化およびトランザクション方針を割り当てるファイル。この XML には、サーバ・インプリメンテーション・クラス名およびサーバ記述子ファイル名を指定したサーバ宣言も格納されます。また、サーバ・アプリケーションの Java ARchive (`.jar`) ファイルを構成する Java クラス・ファイルも識別できます。

「JAR ファイル (.jar)」を参照。

サーバ・グループ

「グループ」を参照。

サーバ ID

1 つのサーバを識別するための識別子。同じサーバ ID が指定された 2 つのサーバを同時に操作することはできません。

Server オブジェクト

(CORBA) サーバ・アプリケーションの初期化関数を実行し、1 つまたは複数のサーバントを作成し、サーバ・アプリケーションのシャットダウンおよびクリーンナップ手順を実行するオブジェクト。

「サーバント」を参照。

サーバ間通信

(CORBA) アプリケーションが分散オブジェクトを呼び出し、分散オブジェクトからの呼び出し (コールバック) を処理することを可能にする BEA Tuxedo ソフトウェアの機能。CORBA または RMI オブジェクトは、BEA Tuxedo ドメインの内側にあっても外側にあってもかまいません。

サービス

- システム内のクライアントによる要求を処理できるアプリケーション・ルーチン。
- サービス要求を実行するアプリケーション・コード・モジュール。

サービス・コード

tlisten プロセスによってリモートで提供されるサービスに関連付けられた名前。

サービス・コントロール・マネージャ (SCM: Service Control Manager)

Windows 2000 のコントロール・パネル・アプレット。インタラクティブ・ユーザ向けの Windows 2000 サービス管理用インターフェイスです。

サービス要求

サービスの呼び出しを依頼する要求元プロセスによって開始される要求。

サービス・ルーチン

クライアントの代わりに 1 つまたは複数の特定のサービスを実行するアプリケーション・モジュール。サービス・ルーチンの構造 (サービス・ルーチンを呼び出したり、終了するメカニズム) は、XATMI インターフェイス仕様によって定義されます。

SERVICES **セクション**

コンフィギュレーション・ファイル内のセクションの 1 つ。サービスを定義します。

サーブレット

CGI を Java で置き換えたもの。サーブレットは、HTTP リクエストに応じて Web サーバによって呼び出される Java クラスです。出力として、ハイパーテキスト・マークアップ言語 (HTML) を生成します。

(Jolt) サーバ側で実行されるアプレット。通常は、Web サーバ環境で実行される Java アプレットを指します。Web ブラウザ環境で実行される Java アプレットと似ています。

セッション Bean

(*Jolt*) サーバ上で動作するビジネス・ロジックを実装する非永続オブジェクト。セッション Bean は、サーバ上で動作するクライアントを論理的に拡張したものと見なすことができます。セッション Bean は複数のクライアント間で共有できません。

セッション・キー

対称鍵アルゴリズムで使用されるキー。このアルゴリズムでは、暗号化と解読で同じキー (セッション・キー) が使用されます。セッション・キーを使って暗号化されたデータは、同じセッション・キーでしか解読できません。

SHA-1

Secure Hash Algorithm 1 の略。Secure Hash Standard で規定されたアルゴリズムであり、264 ビットを超えるメッセージを入力に取り、入力の 160 ビットのメッセージ・ダイジェスト、またはハッシュ値を出力として生成します。MD5 よりやや低速で動作しますが、サイズの大きいメッセージ・ダイジェストの場合、衝突や攻撃に対してより強力に保護されます。SHA-1 は、サイズの大きいファイルを安全な方法で圧縮してから PKCS などの公開鍵暗号システムで秘密鍵を使って暗号化する場合に適した、デジタル署名アプリケーション用のアルゴリズムです。

共有接続トランザクション

1 つのデータベースとの単一の共有接続を使用して、複数のプロセスにまたがっているように見えるトランザクション。

SHM **モデル**

単一のコンピュータ (対称型マルチプロセッサを含む) 上で完全に実行される BEA Tuxedo アプリケーション。

シンプル・イベント

BEA 社固有のイベント・インターフェイス。名前が意味するように、このインターフェイスは使いやすさを考慮して設計されています。

Simple Network Management Protocol (SNMP)

インターネット・コミュニティによって開発されたネットワーク管理プロトコルの事実上のデファクト・スタンダード。

シングル・スレッド処理

プログラムを完全に実行すること。1 つのトランザクション処理が完了してから、次のトランザクション処理が開始されます。

シングルトン・オブジェクト

(CORBA) プロセス・アドレス領域に 1 つしか存在できないオブジェクト。

スケルトン

(CORBA) メソッド呼び出しをサーバント・オブジェクトにディスパッチするために必要な情報を ORB に提供し、IDL コンパイラによって生成されるパブリック抽象クラス。サーバ・スケルトンは、クライアント・スタブと同じく、生成元の IDL インターフェイスに固有です。サーバ・スケルトンは、サーバ・サイドでクライアント・スタブに相当するものです。クライアント・スタブとスケルトンは、ORB が静的起動で使用します。

SMUX

SNMP Multiplexing の略。RFC 1227 で定義されているマスタ・エージェント / サブエージェント通信用プロトコル。

SMUX **サブエージェント**

SNMP Multiplexing (SMUX) プロトコルでは、エージェントと通信し、MIB モジュール内の特定のオブジェクトに対応する管理オペレーションを解決するサブエージェントを作成できます。

SNMP

「Simple Network Management Protocol (SNMP)」を参照。

SNMP エージェント

SNMP プロトコルを使用してシステム・マネージャとデータをやり取りするエージェント。

ソケット

名前のバインド先となる通信の端点。ソケット・インターフェイスは、BEA Tuxedo システムでサポートされるネットワーク・アクセス方式です。TCP/IP 接続の論理的な端点です。アプリケーションは、ソケットを使用して TCP/IP 接続にアクセスします。

ソケット記述子

ソケットと TCP/IP 接続を一意に識別する、TCP/IP によって割り当てられた番号。アプリケーションは、TCP/IP API 呼び出しでソケット記述子を指定し、ソケットと接続を識別する必要があります。

ソケット ID

「ソケット記述子」を参照。

ソケット番号

「ソケット記述子」を参照。

SQL

Structured Query Language の略。リレーショナル・データベースを定義したり、これにアクセスするための非手続き型言語。SQL は、データベース言語の業界標準となりました。

標準 MIB

インターネット・コミュニティによって標準として規定された MIB。例として、MIB I と MIB II があります。

StarLAN

AT&T LAN 製品。

状態

(ATMI) 参加しているトランザクションのうちの 1 つのトランザクションから見たときの会話の状況。会話の状態により、トランザクションで発行できる有効なコマンドが決まります。各トランザクションの状態は、会話の途中で動的に変わります。

(CORBA) オブジェクトの現在の状況の記述。通常、状態はメモリ内の記述です。

状態を持つアプリケーション

サービスまたはオペレーションの完了後、状態情報をメモリ内に保持するアプリケーション。

ステートフル・セッション Bean

(CORBA) クライアントとの会話の状態についての情報を保存する Bean。この会話は、会話の状態を修正する呼び出しで構成されます。

状態を持たないアプリケーション

サービスまたはオペレーションの完了後、状態情報をメモリからクリアするアプリケーション。

ステートレス・セッション Bean

(CORBA) クライアントとの会話の状態についての情報を保存しない Bean。

STRING **バッファ**

(ATMI) ヌル以外の文字配列で構成され、最後がヌル文字で終了するデータ構造体。これは自己記述型バッファです。

stroid

(CORBA) 文字列で表されるオブジェクト ID。

「オブジェクト識別子 (OID)」を参照。

構造化イベント

(CORBA) CORBA ノーティフィケーション・サービスによって定義される COS 構造化イベント。構造化イベントには、Fixed ヘッダ、Variable ヘッダ、Filterable 本文、および Remaining 本文があります。

構造化照会言語

「SQL」を参照。

サブエージェント

要求を満たし、マスタ・エージェントに応答するマスタ・エージェント・プロトコルのコンポーネント。

サブルーチン

(ATMI) 1 つまたは複数のプログラム内の 1 つまたは複数の箇所で実行できる、順序付けされた指示の集合。サブルーチンは通常、呼び出して実行します。

サブスクライブ

構造化イベントを受信するための登録を行うこと。

サブスクライバ

イベントまたはイベントのセットをサブスクライブし、イベントの発生が通知されたときに実行すべき処理を宣言しておくアプリケーション・プログラム。

対称鍵アルゴリズム

セッション・キーという同じ鍵を使って、データを暗号化したり、解読するためのアルゴリズム。乱数ジェネレータにより、通信のたびに新しいセッション・キーが作成されます。これで、以前の通信を利用した攻撃を防ぐことができます。

対称鍵アルゴリズムの例として、DES、RC2、および RC4 があります。

同期化

通信しあうトランザクション間で、コミット処理を制御する調整プロセス。回復可能なリソースに対する論理的に関連したすべての更新処理は、完了するかまたはすべて元に戻されます。

同期

- ほかのイベントと同時に発生、存在、または呼び出されるイベント。
- ほかのプロセスでの特定のイベントの発生に応じて、定期的または予測どおりに発生するオペレーション。たとえば、プログラム内で事前に指定されている時点で制御権が与えられる入力ルーチンまたは出力ルーチンを呼び出すオペレーションです。

同期型通信

タイミング・シグナルを使用してデータを転送する方法。同期型通信では、あるソフトウェアから別のソフトウェアにメッセージが送信される場合、送信元のソフトウェアは、サービス・プロバイダが要求の処理を完了してからでないと、次の処理に進めません。

同期型プロセス

常に別のプロセスと関連して実行されるプロセス。同期的に処理する要求では、クライアントは、サービス・プロバイダによる要求処理が完了してからでないと、次の処理に進めません。

システム

「BEA Tuxedo のドメイン」に相当する BEA TOP END の用語。

「Tuxedo のドメイン」を参照。

システム管理

- 特定の設定またはインストールで使用するために、システムのインスタンスを準備すること。
- インストール内容の変更に応じてシステムを変更すること。この用語は、「システム運用」という意味でも使用されます。

システム管理者

「管理者」を参照。

システム・マネージャ

エージェントからデータを受信し、そのデータを基にアクションを実行するネットワーク管理システムの一部。

システム運用

システムで定期的に実行する必要のあるタスクのこと。例として、データとログのバックアップおよび復元、システムでエラーが発生していないかどうかの監視などがあります。

T

タスク

1 回のトランザクションの実行。

TCP/IP

「Transmission Control Protocol/Internet Protocol (TCP/IP)」を参照。

TDomain ゲートウェイ

2 つの BEA Tuxedo ドメイン間の通信を処理するドメイン・ゲートウェイ。

参考項目 「ドメイン・ゲートウェイ」

端末

- コンピュータ・モニタ。
- データが入力または出力される、システムまたは通信ネットワーク上のポイント。

スレッド

プログラム実行の基本単位。プロセスでは、複数のスレッドが同時に実行される場合があります。各スレッドは、イベントを待機する、処理が完了しなくてもプログラムが続行できるような時間のかかるタスクを実行する、といった別々の処理を実行できます。一般に、タスクの実行を終えたスレッドは中断または破棄されます。

参考項目 「ワーカ・スレッド」

スレッド処理

1 つのプロセスを複数のエンティティに分割し、各エンティティが共通のアドレス領域を共有しつつ、独立して実行するためのオペレーティング・システム機能。

オブジェクトごとのスレッド同時実行モデル

(CORBA) マルチスレッド化された CORBA サーバ・アプリケーションでサポートされているスレッド処理モデル。オブジェクトごとのスレッド同時実行モデルでは、サーバ・プロセスの活性化された各オブジェクトは、常に 1 つのスレッドと関連付けられます。オブジェクトに対する要求ごとに、ディスパッチ・スレッドとオブジェクトとの関連付けが確立されます。

参考項目 「マルチスレッド CORBA サーバ・アプリケーション」

要求ごとのスレッド同時実行モデル

(*CORBA*) マルチスレッド化された CORBA サーバ・アプリケーションでサポートされているスレッド処理モデル。要求ごとのスレッド同時実行モデルでは、クライアントからの各要求は別々の制御スレッドで処理されます。

参考項目 「マルチスレッド CORBA サーバ・アプリケーション」

スレッド・プール

(*CORBA*) マルチスレッド化された CORBA サーバ・インプリメンテーションでスレッドを管理するためのコストを低減する手段。起動時に必要に応じてスレッドの作成、割り当て、および解放を行うプール。スレッドは、次の要求の処理に必要となるまでプールで待機します。スレッド・プールを使用すると、どんなスレッド処理モデルもサポートできます。

参考項目 「マルチスレッド CORBA サーバ・アプリケーション」

3 層型のクライアント / サーバ・アーキテクチャ

n 層型のクライアント / サーバ型アーキテクチャのインプリメンテーション。

参考項目 「n 層型のクライアント / サーバ」および「2 層型のクライアント / サーバ」

tie クラス

(CORBA) デレゲーション・ベースのプログラミング手法を使用した場合に、IDL コンパイラによって生成されるクラス。デレゲーション・ベースのプログラミング手法を利用するのは、継承のオーバーヘッドが大きい場合、または使用できない場合です。たとえば、必要なもの以外も継承してしまうので、グローバル・クラスの継承を必要とする場合に、既存のレガシー・コードを使用してオブジェクトを実装することができません。

デレゲーション・ベースの手法では、インプリメンテーションは POA スケルトン・クラスを継承しません。代わりに、ラッパー・クラスが POA スケルトンを継承し、必要に応じてコーディングされたインプリメンテーションに呼び出しを委譲します。tie クラスと呼ばれるこのラッパー・クラスは、継承を利用する手法で使用されるものと同じスケルトン・クラスと一緒に IDL コンパイラによって生成されます。tie クラスは、スケルトンと同じく、関連するインターフェイスの各 OMG IDL オ

ペレーションに対応するメソッドを提供します。ただし、レガシー・オブジェクトのインターフェイスに合わせて tie クラスを修正する必要があります。生成される tie クラスは、生成されるスケルトン・クラスの名前の最後に `_tie` を付けた名前になります。

TLI

「トランスポート層インターフェイス (TLI: Transport Layer Interface)」を参照。

tlisten

デーモン・プロセスとして実行され、ほかの BEA Tuxedo システム・プロセスのリモート・サービス接続を提供する、ネットワーク・プロバイダに依存しないリスナ・プロセス。

TLOG

「トランザクション・ログ (TLOG: Transaction Log)」を参照。

TM

「トランザクション・マネージャ (TM: Transaction Manager)」を参照。

TMFFNAME

(CORBA) アプリケーション提供の名前とオブジェクト・リファレンスとのマッピングを維持する FactoryFinder とサポートする NameManager サービスを実行する BEA Tuxedo ソフトウェアで提供されるサーバ・アプリケーション。

「オブジェクト・リファレンス」および「ファクトリ・ファインダ」を参照。

TMIFRSVR

(CORBA) インターフェイス・リポジトリへのアクセス用として BEA Tuxedo ソフトウェアで提供されるサーバ・アプリケーション。API は、CORBA で定義されているインターフェイス・リポジトリ API のサブセットです。インターフェイス・リポジトリ API の詳細については、C++ プログラミング リファレンスを参照してください。

参考項目 「アプリケーション・プログラミング・インターフェイス (API: Application Programming Interface)」、「会話」、および「インターフェイス・リポジトリ」

TMS

「トランザクション・マネージャ・サーバ (TMS: Transaction Manager Server)」を参照。

トークン

コマンドやサブシステム名など、メッセージ定義ブロックの個々の要素。

TP

「トランザクション処理 (TP: Transaction Processing)」を参照。

TP **フレームワーク**

「WebLogic Express」を参照。

TP **モニタ**

「トランザクション処理モニタ (TP モニタ)」を参照。

TP **プロトコル**

「トランザクション処理プロトコル (TP プロトコル)」を参照。

TPSUT

「トランザクション処理サービス・ユーザ・タイトル (TPSUT: Transaction Processing Service User Title)」を参照。

transaction

- データベースをある一貫した状態から別の状態に変換する、1 つの完全な作業単位。DTP では、トランザクションは、1 つまたは複数のシステムで実行される複数の作業単位を含むこともあります。
- アプリケーションが、データベースなどの共用リソースで作業を実行するための論理構造体。トランザクションの代わりに行われる処理は、ACID 特性 (原子性、一貫性、独立性、持続性) に準拠します。

「分散トランザクション処理 (DTP: Distributed Transaction Processing)」を参照。

トランザクション・コーディネータ

トランザクションに関係するオペレーションとデータの整合性および一貫性を保証するインフラストラクチャを備えたシステム・ソフトウェア・コンポーネント。

「トランザクション・マネージャ (TM: Transaction Manager)」を参照。

TransactionCurrent

(CORBA) トランザクションを管理するためのオブジェクト。TransactionCurrent オブジェクトは、`CosTransactions` モジュールの Current オブジェクトのメソッドをすべてサポートしています。また、リソース・マネージャをオープン / クローズするための API もサポートしています。

TransactionCurrent は、CORBA オブジェクト・トランザクション・サービス (OTS) のクライアントがスレッドとトランザクションとの関連性を明示的に管理できるようにするメソッドを定義します。このオブジェクトは、ほとんどのアプリケーションで OTS を簡単に使用できるようにするメソッドを定義します。

参考項目 「アプリケーション・プログラミング・インターフェイス (API: Application Programming Interface)」、「 CORBA オブジェクト・トランザクション・サービス (OTS: Object Transaction Service)」、「 Credentials オブジェクト」、および「 リソース・マネージャ (RM: Resource Manager)」

トランザクション・ログ (TLOG: Transaction Log)

グローバル・トランザクションをトラッキングする BEA Tuxedo システムのログ機能。

トランザクション・マネージャ (TM: Transaction Manager)

アプリケーション・プログラムの代わりに、グローバル・トランザクションを管理するシステム・ソフトウェア・コンポーネント。トランザクション・マネージャは、アプリケーション・プログラムおよび通信用リソース・マネージャから発行されるコマンドを調整し、トランザクションに参加するすべてのリソース・マネージャと通信してグローバル・トランザクションを開始したり、実行します。グローバル・トランザクションの処理中にリソース・マネージャに障害が発生した場合は、

保留中のグローバル・トランザクションをコミットするか、またはロールバックするかの判定を、リソース・マネージャの代わりにトランザクション・マネージャが行います。

参考項目 「トランザクション・コーディネータ」

トランザクション・マネージャ・サーバ (TMS: Transaction Manager Server)

グローバル・トランザクションの 2 フェーズ・コミット・プロトコルおよび回復処理を管理する BEA Tuxedo システムのサーバ・プロセス。

トランザクション方針

(CORBA) トランザクションと関連付けられるクライアント・リクエストとサーバントのトランザクション・コンテキストとの間の、TP フレームワークまたは EJB コンテナの対話を決定する方針。

トランザクション処理 (TP: Transaction Processing)

トランザクション処理は、以下の処理を行うプログラムを簡単に記述するための便利な関数セットで構成されています。

1. サーバ・アプリケーションを初期化し、起動ルーチンとシャットダウン・ルーチンを実行します。

2. サーバ・アプリケーションを BEA Tuxedo ドメインのリソースに関連付けます。

3. ハウスキーピング機能を実行します。

トランザクション処理モニタ (TP **モニタ**)

会話型のオペレーティング・システム上に設定された、トランザクションの実行環境を提供する製品クラス。

トランザクション処理プロトコル (TP **プロトコル**)

異機種システム上のトランザクション処理マネージャの相互運用で使用される標準プロトコルのセット。

トランザクション処理サービス・ユーザ・タイトル (TPSUT: Transaction Processing Service User Title)

1 つのアプリケーション・エントリ内で、OSI TP の端点を指定するために使用される値。

トランザクション・サービス

「CORBA オブジェクト・トランザクション・サービス (OTS: Object Transaction Service)」を参照。

1 秒あたりのトランザクション (TPS: Transactions Per Second)

TPC で定義された標準トランザクションを使用して計算されるスループットの値。通常、トランザクションに対する応答時間の 90% が 2 秒未満の場合の、処理可能なトランザクションの最大数を示します。

一時オブジェクト

(CORBA) 生成元のプロセスの存続期間だけ存在するオブジェクト。

「永続オブジェクト」を参照。

変換

本来のデータ型を、入力データや出力データ用の表現に変更するプロセス。データ長、バイト順序、および文字の符号化が変更されます。

Transmission Control Protocol/Internet Protocol (TCP/IP)

インターネットを定義し、トランスポート層インターフェイスによってサポートされている標準の通信プロトコル群。TCP/IP ソフトウェアは元々 UNIX オペレーティング・システム用に設計されましたが、現在ではあらゆる主要なオペレーティング・システムに利用できます。

トランスポート・インターフェイス

ネットワーク・プロバイダへのアクセスに使用されるプログラミング・インターフェイス。一般的に、トランスポート・インターフェイスは、ある程度までネットワーク・プロバイダから独立しています。

トランスポート層インターフェイス (TLI: Transport Layer Interface)

OSI 通信プロトコルの第 4 層で定義されている、データ通信機能を提供する標準的な UNIX システムのユーザ・レベル・インターフェイス。BEA Tuxedo システムでサポートされているネットワークへのアクセス手段です。

トランスポート・プロバイダ

「ネットワーク・プロバイダ」を参照。

トラップ

管理対象オブジェクトで発生したエラーについての情報を格納する SNMP データ・パケット。トラップは任意通知型イベントです。つまり、エージェントによって通知するかどうかは任意です。

TUXCONFIG **ファイル**

BEA Tuxedo アプリケ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。すべての BEA Tuxedo プロセスは、このファイルを参照してコンフィギュレーション情報を取得します。

参考項目 「アプリケーション」、「クライアント」、「サーバ」、および「UBBCONFIG ファイル」

Tuxedo

「BEA Tuxedo システム」を参照。

Tuxedo **のドメイン**

「ドメイン」を参照。

Tuxedo MIB

Tuxedo 内部のリソース用データ構造。具体的には、BEA Tuxedo または WebLogic Enterprise フレームワーク・コンポーネントです。このコンポーネントは、Tuxedo または WebLogic Enterprise フレームワークを構成するオブジェクト・クラスおよびその属性をすべて定義します。Tuxedo システムの管理情報ベース全体は、汎用の MIB と主要なコンポーネントごとに固有の MIB としてまとめられています。Tuxedo または WLE フレームワークのコンフィギュレーションや管理は、プログラム処理することができます。

Tuxedo **リモート・クライアント・アプリケーション**

「CORBA リモート・クライアント・アプリケーション」を参照。

2 **フェーズ・コミット** (2PC: Two-Phase Commit)

複数の DBMS (またはほかのリソース・マネージャ) にまたがる単一のトランザクションを調整する方法。トランザクションで実行した、関連するすべてのデータベースへの更新処理をコミットするか、または完全にロールバックしてトランザクション開始時の状態に戻すことにより、データの整合性を保証できます。

2 層型のクライアント / サーバ

アプリケーションを 2 つの部分に分割し、デスクトップ・ワークステーションとサーバ・マシンに対して処理を分けるというアプリケーション開発の手法。

TX インターフェイス

Transaction Demarcation (TX) API。トランザクション・マネージャを呼び出すためのアプリケーション・プログラム。アプリケーション・プログラムは、TX インターフェイスを使用して、グローバル・トランザクションの境界を定義し、これらのトランザクションを完了します。

バッファ型の変換

(ATMI) アプリケーション・プログラムのデータ・バッファまたはレコードを変換するプロセス。データは、目的のアプリケーション・プログラムに適した形式にフォーマットされます。

型付きバッファ

(ATMI) 特定のデータ型に関するメッセージ通信用のバッファ。

「バッファ型」を参照。

U

UBBCONFIG ファイル

BEA Tuxedo アプリケーション用の ASCII 形式のコンフィギュレーション・ファイル。BEA Tuxedo のマニュアルでは `TUXCONFIG` ファイルと呼んでいるバイナリ形式のファイルは、`UBBCONFIG` ファイルから作成します。

参考項目 「アプリケーション」、「 クライアント」、「 サーバ」、および「 TUXCONFIG ファイル」

UDP

「ユーザ・データグラム・プロトコル (UDP: User Datagram Protocol)」を参照。

宣言の取り消し

あるサービスのサービス・テーブル・エントリが BEA Tuxedo の掲示板にない場合、そのサービスの宣言は取り消されます。

ユニプロセッサ

1 つの CPU だけを搭載したコンピュータ。

参考項目 「マルチプロセッサ」

汎用デバイス・リスト (UDL)

システム全体にわたるデバイスの一覧 (raw ディスク・スライスまたは UNIX ファイル)。これらのデバイスでは、TUXCONFIG コンフィギュレーション・テーブル、BEA Tuxedo のトランザクション・ログ、およびデータベース用の領域が割り当てられます。汎用デバイス・リストは、TUXCONFIG 環境変数で指定された場所に格納されます。

URL

Uniform Resource Locator の略。Web サイトのアドレス。URL の例は www.bea.com です。

ユース・ケース

設計中のアプリケーションとユーザが対話する方法を示すテキスト。ユース・ケースは、ユーザが従うプロセスを反映します。

参考項目 「アプリケーション」

ユーザ・データグラム・プロトコル (UDP: User Datagram Protocol)

TCP/IP のデータグラム・トランスポート層プロトコル。

ユーザ・スポンサー

ユーザ認証用のセキュリティ・インターフェイスを呼び出すコード。

参考項目 「認証」

UserTransaction **環境オブジェクト**

(CORBA) クライアント・アプリケーションを BEA Tuxedo トランザクション・サブシステムに接続するオブジェクト。接続後、クライアントはトランザクションのコンテキスト内でオペレーションを実行できます。UserTransaction オブジェクトは、Java クライアント・アプリケーションの場合にのみ存在します。

UserTransaction **インターフェイス**

アプリケーションがトランザクションの境界を明示的に管理することを可能にするメソッドを定義するインターフェイス。

参考項目 「インターフェイス」、「 メタデータ・インターフェイス」、および「 TransactionCurrent」

V

ビュー

(ATMI) VIEW System Manager では、管理対象のホストを表すアイコンをクリックすると表示されるウィンドウのこと。

(CORBA) 別のオブジェクト・システム (ActiveX など) にある BEA Tuxedo ドメイン内の CORBA オブジェクトの表現。

参考項目 「ActiveX」、「 CORBA オブジェクト」、「 オブジェクト・システム」、および「 CORBA ドメイン」

VIEW **バッファ**

(ATMI) C 構造体に似たデータ構造体。このバッファ型を定義するときには、VIEW 記述ファイルが作成されます。これは自己記述型バッファです。VIEW バッファでは、常に VIEW 記述が作成されます。

VIEW **の定義**

(ATMI) BEA Tuxedo 環境で、入力データまたは出力データに対して使用されるデータ構造体の説明。

仮想マシン

端末のユーザによって制御される、コンピュータと同等の機能を備えたマシンおよび関連するデバイス。

仮想 1 **ホップ・ネットワーク**

すべてのノードが、1 回の伝送でほかのすべてのノードから到達可能なネットワーク。すべてのノードが完全に相互接続されている限り、物理的なネットワーク構成 (リング型、スター型、バス型、またはその他の有効なネットワーク構成) の種類は関係ありません。

Volume Table of Contents (VTOC)

BEA Tuxedo システムおよび場合によってはデータベース・テーブルが格納されたファイル。

VTOC

「Volume Table of Contents (VTOC)」を参照。

W

WAN

「広域ネットワーク (WAN: Wide Area Network)」を参照。

Web GUI

「BEA Administration Console」を参照。

WebLogic Express

(CORBA) Java アプレットまたは Java アプリケーションで使用する JDBC のインプリメンテーション。

WebLogic Server

分散型 Java アプリケーションのアセンブリ、デプロイ、および管理を行うピュア Java アプリケーション・サーバ。

広域ネットワーク (WAN: Wide Area Network)

通信回線経由でデータを送受信する、公共または民間のデータ通信システム。

ウィンドウ

グラフィカル・ユーザ・インターフェイスのシステムで表示されるユーザ側の画面。ユーザと会話を行うためのアプリケーションのメカニズムです。

関連付け

(CORBA) Bean が別の Bean からのイベントのリスナとして登録されていることを示します。

WLS

「WebLogic Server」を参照。

ワーカ・スレッド

(CORBA) クライアント・アプリケーションからの要求を実行する予定のスレッド。BEA Tuxedo Java ソフトウェアでは、ソフトウェアがワーカ・スレッドのプールを管理するスレッド・プール・モデルを使用します。BEA Tuxedo Java ソフトウェアがクライアント・アプリケーションから要求を受信すると、ソフトウェアはスレッド・プールのワーカ・スレッドをスケジューリングして、要求を実行します。要求が完了すると、ワーカ・スレッドはスレッド・プールに戻ります。ワーカ・スレッドは一度に 1 つの要求だけを処理します。

「スレッド」を参照。

ワークステーション

BEA Tuxedo システムのサーバ側機能がインストールされていないサイトにアプリケーション・クライアントを配置するための BEA Tuxedo システムのコンポーネント。ワークステーション・クライアントは、管理サーバ、アプリケーション・サーバ、および掲示板をサポートしません。クライアントとその他のアプリケーションの間での通信は、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 (WSC: Workstation client)

BEA Tuxedo サーバ・ソフトウェアがインストールされていないマシン上で実行するクライアント・プロセス。Windows、Windows NT、UNIX、および VMS プラットフォームでは、複数の ワークステーション・クライアントを同時に実行できます。ワークステーション・クライアントでは、ATMI を使用できます。

ワークステーション・ハンドラ (WSH)

ワークステーション・クライアントとネイティブな BEA Tuxedo サーバの間の 1 つまたは複数の接続を管理するプロセス。ワークステーション・ハンドラは、クライアントの代わりにサービスを要求し、トランザクションを管理し、応答を返します。WSH プロセスは、ワークステーション・リスナ (WSL) によって起動および停止されます。また、WSH プロセスはアプリケーションの管理ドメイン内に常駐します。ハンドラは、クライアントとして BEA Tuxedo の掲示板に登録されます。

「 ワークステーション・リスナ (WSL)」および 「IIOP ハンドラ (ISH)」を参照。

ワークステーション・リスナ (WSL)

ワークステーション・ハンドラ (WSH) をワークステーション・クライアント (WSC) に割り当てるプロセス。割り当てが行われると、指定された WSH は指定のクライアントからのすべてのサービス要求を管理します。WSL はワークステーション・ハンドラのプールも管理し、負荷の条件に応じてワークステーション・ハンドラを起動および停止します。また、WSL プロセスはアプリケーションの管理ドメイン内に常駐します。

「ワークステーション・ハンドラ (WSH)」および「IIOP リスナ (ISL)」を参照。

ラップする

アプリケーションをほかのアプリケーションでも利用できるようにソフトウェア層に入れて囲むこと。

参考項目 「アプリケーション」および「 ラッパー」

ラッパー

(CORBA) レガシー・アプリケーションをインプリメンテーションとして CORBA クライアント・アプリケーションで利用できるように、レガシー・アプリケーションをラップするためのソフトウェア層。

参考項目 「レガシー・アプリケーション」、「 クライアント・アプリケーション」、および「 ワークステーション・ハンドラ (WSH)」

ラッパー・オブジェクト

「tie クラス」を参照。

X

X.509

証明書の形式を指定する規格。証明書を使用すると、厳密な認証が行われ、名前を公開鍵に安全に関連付けることができます。

参考項目 「認証」および「 証明書」

XA

アプリケーション、トランザクション・マネージャ、およびデータベース・システム間の通信用双方向システム・レベル・インターフェイス。X/Open 分散トランザクション処理 (DTP) モデルで定義されています。

XATMI アプリケーション・サービス・エレメント

XATMI インターフェイスのプリミティブを OSI TP プロトコルにマッピングするソフトウェア。

XATMI インターフェイス

グローバル・トランザクションの処理時に要求 / 応答型の通信および会話型の通信を行うため、アプリケーション・プログラムで使用できるインターフェイス。

X_COMMON バッファ

(ATMI) short 型、long 型、char 型、のいずれかの C データ型で構成される、入れ子状ではない C 構造体。X_COMMON は、X/Open XATMI 標準で定義されている 3 つのバッファ型のうちの 1 つです。これは、BEA Tuxedo の VIEW バッファと同じですが、X_COMMON は、C 言語と COBOL 言語に共通するフィールド・タイプのサブセットのみを表します。

X_C_TYPE バッファ

(ATMI) int 型、short 型、long 型、char 型、float 型、double 型、文字列型、オクテット配列型、のいずれかの C データ型で構成される、入れ子状ではない C 構造体。X_C_TYPE は、X/Open XATMI 標準で定義されている 3 つのバッファ型のうちの 1 つです。BEA Tuxedo の VIEW バッファと同じです。

XDR

「外部データ表現 (XDR: External Data Representation)」を参照。

XML

eXtensible Markup Language の略。World Wide Web Consortium (W3C) によって開発され、Sun Microsystems, Inc. によって編成された言語。サーバ記述子ファイルおよび EJB デプロイメント記述子を指定する場合に使用します。

X_OCTET バッファ

(ATMI) バイト配列。構造体はアプリケーションで定義されます。
X_OCTET は、X/Open XATMI 標準で定義されている 3 つのバッファ型のうちの 1 つです。BEA Tuxedo の CARRAY バッファと同じです。

X/Open

オープン・システム標準の規定を目的とした、国際的な民間コンソーシアムである X/Open Company, Ltd. のこと。オープン・システムのベンダやユーザで構成されます。BEA Tuxedo 製品は、分散トランザクション処理を実現するため、X/Open 標準を実装するよう設計されています。

X/Open **分散トランザクション処理モデル**

X/Open Company, Ltd. が開発した標準で指定されている分散トランザクション処理モデル。BEA Tuxedo のアーキテクチャは、この標準に基づいています。このモデルでは、DTP システムの 4 つのコンポーネントが定義されています。

- アプリケーション・プログラム (AP: Application Program) は、トランザクションの境界を定義し、トランザクションを構成する処理を実行します (典型的な処理は更新処理)。
- データベース管理システムなどのリソース・マネージャ (RM: Resource Manager) は、共用リソースにアクセスします。
- 通信リソース・マネージャ (CRM: Communication Resource Manager) は、アプリケーション・プログラムがお互いに通信できるようにします。
- トランザクション・マネージャ (TM: Transaction Manager) は、一意な識別子 (XID) をトランザクションに割り当て、トランザクションの進行を監視し、トランザクションの完了や障害発生時の回復処理を実行します。

Y

「Y」で始まる用語はありません。

Z

「Z」で始まる用語はありません。