BEA Tuxedo

BEA Tuxedo CORBA ActiveX

BEA Tuxedo リリース 8.0
8.0 版
2001 年 10 月 31 日

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA Tuxedo CORBA ActiveX

Document Edition	Date	Software Version
8.0	2001 年 10 月 31 日	BEA Tuxedo 8.0

目次

このオンライン・ヘルプについて

オンライン・ヘルプの使い方 vii
ヘルプ・システムが正しく表示されない場合 viii
最新のブラウザを使用していることを確認する ix
ヘルプを読みやすくするためにフォント・サイズを
カスタマイズする x
BEA Builder と共にインストールされるブラウザの考慮事項 x
常用している Web ブラウザの使用 xi
e-docs Web サイト xii
マニュアルの印刷方法 xii
関連情報 xiii
サポート情報 xiii
表記上の規則 xiv

1. 概要

ActiveX とは 1-1
ビューとバインディング 1-2
しくみ 1-2
ActiveX ビューの命名規則 1-3
OMG IDL 1-4
インターフェイス・リポジトリ 1-5
ドメイン 1-6
環境オブジェクト 1-7
Bootstrap オブジェクト 1-8
ファクトリと FactoryFinder オブジェクト 1-9
FactoryFinder オブジェクトの命名規則と BEA Tuxedo 拡張 1-11
SecurityCurrent オブジェクト 1-12
TransactionCurrent オブジェクト 1-13
InterfaceRepository オブジェクト 1-15

2. ActiveX クライアント・アプリケーションの作成

ActiveX クライアント・アプリケーションの開発プロセスの概要 2-2

BEA Application Builder 2-4
ステップ 1: オートメーション環境オブジェクトのインターフェイス・リポジトリへのロード....2-6
ステップ 2: CORBA インターフェイスのインターフェイス・リポジトリへのロード....2-7
ステップ 3: インターフェイス・リポジトリのサーバ・アプリケーションの起動 2-8
ステップ 4: CORBA インターフェイスの ActiveX バインディングの作成 2-9
ステップ 5: ActiveX バインディングのタイプ・ライブラリのロード 2-10
ステップ 6: ActiveX クライアント・アプリケーションの記述....2-10
オートメーション環境オブジェクト、ファクトリ、および CORBA オブジェクトの ActiveX ビューの宣言のインクルード....2-11
BEA Tuxedo ドメインとの通信の確立....2-11
FactoryFinder オブジェクトへの初期リファレンスの取得....2-12
ファクトリを使用した ActiveX ビューの取得 2-13
ActiveX ビューのオペレーションの呼び出し 2-14
コールバックに対するオートメーション・サーバの作成....2-14
COM オブジェクトのインスタンスの作成 2-15
ステップ 7: ActiveX クライアント・アプリケーションのデプロイ 2-16

3. Application Builder メイン・ウィンドウ

Application Builder メイン・ウィンドウ 3-1
[Services] ウィンドウ 3-3
[Workstation Views] ウィンドウ 3-3
Application Builder のオブジェクト 3-4
メニュー・オプション 3-6
[File] メニュー・オプション 3-6
[Edit] メニュー・オプション 3-7
[View] メニュー・オプション 3-8
[Tools] メニュー・オプション 3-8
[Window] メニュー・オプション 3-9
[Help] メニュー・オプション 3-10
ツールバー・ボタン 3-11

4. タスク

CORBA インターフェイスのインターフェイス・リポジトリへのロード ..4-1
Application Builder の起動 ..4-2
CORBA インターフェイスに対する ActiveX バインディングの作成 ..4-3
CORBA インターフェイスに対する ActiveX バインディング作成の設定変更 ..4-5
デプロイメント・パッケージの作成 ..4-6
デプロイメント・パッケージのディレクトリの変更 ..4-7
デフォルト・ディレクトリの変更 ..4-8
メイン・ウィンドウに表示されるオブジェクトのフィルタ ..4-8
プロパティの表示 ..4-10

5. セキュリティの使い方

BEA Tuxedo セキュリティの概要 ..5-1
セキュリティの開発プロセスの概要 ..5-2
ステップ 1: Bootstrap オブジェクトを使用して SecurityCurrent オブジェクトを取得する ..5-2
ステップ 2: SecurityCurrent オブジェクトから PrincipalAuthenticator オブジェクトを取得する ..5-3
ステップ 3: 認証レベルを取得する ..5-3
ステップ 4: 正しい認証で BEA Tuxedo ドメインにログオンする ..5-4
ステップ 5: BEA Tuxedo ドメインからログオフする ..5-5

6. トランザクションの使い方

トランザクションの概要 ..6-1
トランザクションの開発プロセスの概要 ..6-2
ステップ 1: Bootstrap オブジェクトを使用して TransactionCurrent オブジェクトを取得する ..6-3
ステップ 2: TransactionCurrent メソッドを使用する ..6-3

7. コマンド行オプション

用語集

索引

このオンライン・ヘルプについて

このオンライン・ヘルプでは、BEA Application Builder を使用して ActiveX クライアント・アプリケーションを開発する方法について説明します。

このマニュアルでは、次の内容について説明します。

- オンライン・ヘルプの使い方
- ヘルプ・システムが正しく表示されない場合
- BEA Builder と共にインストールされるブラウザの考慮事項
- 常用している Web ブラウザの使用
- e-docs Web サイト
- マニュアルの印刷方法
- 関連情報
- サポート情報
- 表記上の規則

オンライン・ヘルプの使い方

アプリケーションの使い方についてのヘルプを表示するには、Application Builder を起動している必要があります。

主なトピックのヘルプを表示するには、Application Builder のメイン・ウィンドウにある [**Help**] を選択して、次のメニュー・トピックのいずれかを選択します。

- 「概要」― ここでは、Rose Expert の概念、および Rose Expert Application Builder GUI を使用して実行できるアプリケーション開発作業の概要について説明します。

- 「ActiveX クライアント・アプリケーションの作成」ー ここでは、Application Builder を使用して ActiveX クライアント・アプリケーションをビルドする手順を示します。
- 「Application Builder メイン・ウィンドウ」ー ここでは、Application Builder のメイン・ウィンドウのさまざまなコンポーネントについて説明します。
- 「タスク」ー ここでは、さまざまなタスク・ウィンドウの使い方について説明します。特定のウィンドウの使い方に関するヘルプ情報へは、そのウィンドウが開いている時に F1 キーを押してアクセスすることもできます。
- 「セキュリティの使い方」ー ここでは、ActiveX クライアント・アプリケーションでセキュリティを使用する方法について説明します。
- 「トランザクションの使い方」ー ここでは、ActiveX クライアント・アプリケーションでトランザクションを使用する方法について説明します。
- 「コマンド行オプション」ー ここでは、Application Builder のコマンド行バージョンについて説明します。
- 「用語集」ー ここでは、関連する BEA Tuxedo、BEA Builder、Application Builder、ActiveX、およびオブジェクト指向開発の用語について説明します。
- 「About Application Builder」ー ここでは、バージョンおよび著作権に関する情報を提供します。

これらのメニューは、現在開いているタスク・ウィンドウの [Help] ボタンをクリックして表示することもできます。

ヘルプ・システムが正しく表示されない場合

ヘルプ・システムの機能は Netscape Navigator に依存しています。そのため、表示に関する問題は、通常システム上でアクティブになっている Netscape ブラウザのバージョン、およびブラウザで選択しているフォント・サイズの設定と関連しています。また、ヘルプが表示されない、検索機能が動作しないなどの UNIX プラットフォーム上の問題は、通常ユーザの PATH および CLASSPATH 環境変数の設定が不完全であることと関連しています。

以下のトピックでは、ヘルプの起動および表示にさまざまな影響を与える可能性のある問題について、トラブル・シューティングのヒントを紹介します。

- 最新のブラウザを使用していることを確認する
- ヘルプを読みやすくするためにフォント・サイズをカスタマイズする

最新のブラウザを使用していることを確認する

コンテキスト・センシティブ・ヘルプ・システムでは、Netscape Navigator 4.0 以上がローカル・システム上に存在し、かつそれを使用する必要があります。それ以前のバージョンの Netscape ブラウザを使用すると、[Find] または [Print] ボタンをクリックしたときにエラー・メッセージが表示されます。

Navigator 4.0 がインストールされていても、以前のバージョンの Netscape ブラウザも保持していて、それが最後に使用したブラウザである場合、このエラーが表示されることがあります。この問題の対策としては、現在開いている BEA Builder アプリケーションを閉じてから、以前のバージョンの Netscape ブラウザを (開いているものがある場合は) 閉じ、Navigator 4.0 を開いてください。(Navigator 4.0 は開いた後すぐに閉じてかまいません。)

Builder アプリケーションを再起動すれば、ヘルプ・システムは正常に動作します。

注記 オンライン・ヘルプの情報を Web ブラウザで表示する場合、古いバージョンのブラウザでは HTML ヘルプ・ファイルに組み込まれている機能の一部をサポートしていない場合があります。そのため、Internet Explorer バージョン 4.0 以上、Netscape Navigator バージョン 4.0 以上、またはその他の同等の HTML サポートを備えたブラウザを使用することをお勧めします。Web ブラウザでヘルプ情報にアクセスする方法については、「常用している Web ブラウザの使用」を参照してください。

ヘルプを読みやすくするためにフォント・サイズをカスタマイズする

コンテキスト・センシティブ・ヘルプ・システムは、Netscape ブラウザのフォントの設定に依存しています。文字が小さすぎたり、大きすぎたりしてヘルプ・システムに表示される情報が読みにくい場合は、フォント・サイズを変更できます。変更するには、Netscape Navigator ブラウザのフォント設定をリセットするだけです。また、ブラウザで設定したフォント・サイズおよびスタイルは、ヘルプ・システムでも使用されます。

システム上に複数のバージョンの Netscape ブラウザがある場合は、アクティブなブラウザでフォントを設定していることを確認してください。なるべく、最新のブラウザで設定することをお勧めします。ヘルプ・システムでは、最後にアクティブになったブラウザを使用します。複数のバージョンのブラウザを使用してヘルプ・ファイルを表示する場合は、すべてのブラウザで読みやすくなるように設定してください。

最新のブラウザを使用することが大切な理由については、「最新のブラウザを使用していることを確認する」を参照してください。

BEA Builder と共にインストールされるブラウザの考慮事項

BEA Builder 製品をインストールしたときに、Netscape Navigator がシステムに入っていない場合、BEA Builder と共にこのブラウザがインストールされる可能性があります。

コンテキスト・センシティブ・ヘルプ・システムでは、Netscape Navigator 4.0 以上がローカル・システム上に存在している必要があります。そのため、BEA Builder 製品のインストール時に、対象システム上に Netscape Navigator 4.0 ブラウザが既に存在しているかどうか確認されます。Windows システムでは、適切なバージョンのブラウザが見つからない場合、インストール・スクリプトにより、オンライン・ヘルプ・システムをサポートするために BEA Builder 製品インストールの一部としてブラウザをインストールするオプションが提供されます。

BEA Builder 製品のインストール時にインストールされる Netscape Navigator 4.0 では、米国からの輸出が許可されているレベルの暗号化が使用されています。このブラウザをヘルプ・システム以外の目的で使用する場合は、これが Netscape Navigator の最も安全性の高いバージョンではないことに注意してください。

注記 UNIX 用 BEA 製品のインストールでは、正しいバージョンのブラウザは自動的にインストールされないため、この考慮事項は UNIX システムには適用されません。UNIX システムでは、この操作を手動で行う必要があります。

常用している Web ブラウザの使用

ActiveX Application Builder のグラフィカル・ユーザ・インターフェイス (GUI) は、HTML ベースのコンテキスト・センシティブ・ヘルプ・ソリューションとして Netscape NetHelp を使用するように設計および設定されています。

ただし、Microsoft Internet Explorer 4.0 ブラウザ、Netscape Navigator または Communicator 4.0、または HTML 3.0 以上をサポートするその他の Web ブラウザでも、BEA Tuxedo ActiveX クライアントのオンライン・ヘルプを表示できます。Internet Explorer またはその他のブラウザで Builder のマニュアルを表示する場合、Netscape NetHelp との主な相違点は、GUI でヘルプを表示するときにコンテキスト・センシティブ・メニューおよびダイアログ・アクセスを使用できないことです。このマニュアル・セットでは、Netscape NetHelp ビューア以外のブラウザを正式にサポートしていないため、表示上のずれが生じる場合があります。

Web ブラウザで BEA Tuxedo ActiveX Client のマニュアルを表示させるには、ブラウザで以下のファイルを開きます。

YourDrive:tuxdir\help\AppBuilderHtm\default.htm (Windows)

注意 古いバージョンのブラウザでは、HTML ヘルプ・ファイルに組み込まれている一部の機能をサポートしていない場合があります。そのため、Internet Explorer バージョン 4.0 以上、Netscape Navigator バージョン 4.0 以上、または同等の HTML サポートを備えたその他のブラウザを使用することをお勧めします。

e-docs Web サイト

BEA Tuxedo オンライン・マニュアルは BEA 社の Web サイト上で参照することができます。BEA ホーム・ページの [製品のドキュメント] をクリックするか、または http://edocs.beasys.co.jp/e-docs/index.html に直接アクセスしてください。

マニュアルの印刷方法

このマニュアルは、ご使用の Web ブラウザで一度に 1 ファイルずつ印刷できます。Web ブラウザの [ファイル] メニューにある [印刷] オプションを使用してください。(最初に、印刷する HTML コンテンツのフレーム内をどこかクリックして、フレームを選択してください)。

このオンライン・ヘルプの情報は、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo CORBA ActiveX』でも利用できます。『BEA Tuxedo CORBA ActiveX』は、PDF ファイルとして入手できます。この PDF を Adobe Acrobat Reader で開くと、マニュアル全体または一部をブック形式で印刷できます。PDF 形式を利用するには、BEA Tuxedo Documents ページの [PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

また、このオンライン・ヘルプの PDF 版は、システム上の以下の場所で入手できます。

YourDrive:tuxdir\help\app_builder_help.pdf (Windows)

Adobe Acrobat Reader は、Adobe 社の Web サイト (http://www.adobe.co.jp/) から入手できます。

関連情報

以下のマニュアルは、BEA Tuxedo システムを理解する上で役に立ちます。BEA Tuxedo マニュアルは、別のマニュアル CD の BEA Tuxedo ソフトウェアに付属しており、BEA の Web サイト (http://e-docs.beasys.co.jp/) でも入手できます。

- 『BEA Tuxedo CORBA アプリケーション入門』。このマニュアルでは、BEA Tuxedo CORBA のコンポーネントおよび API について紹介し、アプリケーション開発プロセスの基礎について説明しています。
- 『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』。このマニュアルでは、C++ プログラマが BEA Tuxedo 製品の主な機能をインプリメントして、BEA Tuxedo ドメインで実行されるスケーラブルで高パフォーマンスの CORBA C++ サーバ・アプリケーションを設計およびインプリメントする方法について説明しています。
- 『BEA Tuxedo CORBA プログラミング・リファレンス』。このマニュアルでは、OMG IDL 構文、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、および `buildobjserver` コマンドなどの CORBA プログラミング環境に関するリファレンス資料を提供しています。
- 『BEA Tuxedo CORBA University サンプル・アプリケーション』。このマニュアルでは、BEA Tuxedo に含まれている University サンプル・クライアント / サーバ・アプリケーションのビルドおよび実行方法について説明しています。

サポート情報

皆様の BEA Tuxedo マニュアルに対するフィードバックをお待ちしています。ご意見やご質問がありましたら、電子メールで docsupport-jp@bea.com までお送りください。お寄せいただきましたご意見は、BEA Tuxedo マニュアルの作成および改訂を担当する BEA 社のスタッフが直接検討いたします。

電子メール メッセージには、BEA Tuxedo 8.0 リリースのマニュアルを使用していることを明記してください。

BEA Tuxedo に関するご質問、または BEA Tuxedo のインストールや使用に際して問題が発生した場合は、http://www.bea.com の BEA WebSUPPORT を通して BEA カスタマ・サポートにお問い合わせください。カスタマ・サポートへの問い合わせ方法は、製品パッケージに同梱されているカスタマ・サポート・カードにも記載されています。

カスタマ・サポートへお問い合わせの際には、以下の情報をご用意ください。

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

表記上の規則

このマニュアルでは、以下の表記規則が使用されています。

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

規則	項目

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

太字の等幅テキスト	コード内の重要な語を示します。 例 : `void` **`commit`** `( )`
斜体の等幅テキスト	コード内の変数を示します。 例 : `String` *`expr`*
大文字のテキスト	デバイス名、環境変数、および論理演算子を示します。 例 : LPT1 SIGNON OR
{ }	構文の行で、選択肢の組み合わせを示します。かっこは入力しません。
[ ]	構文の行で、オプション項目を示します。かっこは入力しません。 例 : `buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...[-l` *`file-list`*`]...`
|	構文の行で、相互に排他的な選択肢の区切りとして使います。記号は入力しません。

規則	項目
. . .	コマンド・ラインで、以下のいずれかの場合を示します。 ■ コマンド・ラインで、同じ引数を繰り返し使用できることを示します。 ■ 文で、追加のオプション引数が省略されていることを示します。 ■ 追加のパラメータ、値、またはその他の情報を入力できることを示します。 記号は入力しません。 例 : `buildobjclient [-v] [-o name ] [-f file-list]...[-l file-list]...`
. . .	コード例または構文の行で、項目が省略されていることを示します。記号は入力しません。

1　概要

ここでは、ActiveX クライアント・アプリケーション開発プロセスの概要、および BEA Tuxedo CORBA 環境用に ActiveX クライアント・アプリケーションを開発する前に理解する必要がある概念について説明します。

ここでは、次の内容について説明します。

- ActiveX とは
- しくみ
- ActiveX ビューの命名規則
- OMG IDL
- インターフェイス・リポジトリ

ActiveX とは

ActiveX は、作成された言語に関係なく、ネットワーク環境でソフトウェア・コンポーネント同士が対話できるようにする Microsoft の技術です。ActiveX は、Component Object Model (COM) を基に構築され、Object Linking and Embedding (OLE) と統合されます。OLE は、ドキュメント埋め込み用のアーキテクチャを提供します。オートメーションは、Visual Basic、Delphi、PowerBuilder などのアプリケーションで、オートメーション・オブジェクト、ActiveX コントロール、ActiveX ドキュメントの操作を可能にする COM の一部です。

BEA ActiveX Client は、BEA Tuxedo と COM オブジェクト・システムの間の相互運用性を提供します。ActiveX Client は、BEA Tuxedo ドメインの CORBA オブジェクトのインターフェイスをオートメーション・オブジェクトのメソッドに変換します。

ビューとバインディング

ActiveX クライアント・アプリケーションは、CORBA インターフェイスのビューを使用します。ビューは、ローカルでオートメーション・オブジェクトとして BEA Tuxedo ドメインで CORBA インターフェイスを表します。CORBA オブジェクトの ActiveX ビュー (ActiveX ビューと呼ばれる) を使用するには、ActiveX のバインディングを作成する必要があります。バインディングは、CORBA オブジェクトと ActiveX のインターフェイスを定義します。CORBA オブジェクトのインターフェイスは、インターフェイス・リポジトリにロードされます。次に、BEA ActiveX Application Builder を使用してインターフェイスのオートメーション・バインディングを作成します。

ActiveX クライアント・アプリケーションと生成されたバインディングの組み合わせにより、オブジェクトの ActiveX ビューが作成されます。

しくみ

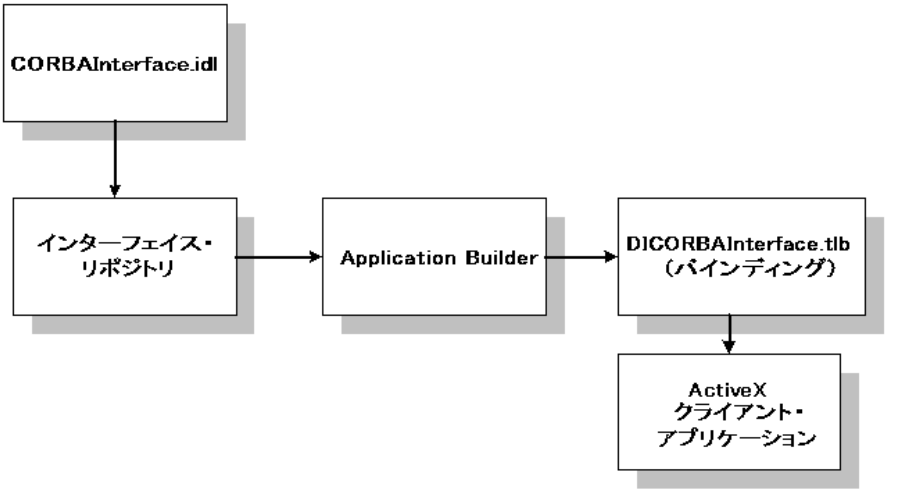
BEA ActiveX Client を使用すると、ActiveX クライアント・アプリケーションは、BEA Tuxedo ドメインの CORBA オブジェクトとやり取りできます。ActiveX クライアント・アプリケーションは、オートメーション環境オブジェクトを使用して、BEA Tuxedo ドメインの CORBA オブジェクトにアクセスします。ActiveX Client は、CORBA オブジェクトの ActiveX ビューを作成します。CORBA オブジェクトの ActiveX ビューは、ActiveX クライアント・アプリケーションから受信したすべての要求を、BEA Tuxedo ドメインの適切な CORBA オブジェクトに変換および転送します。

開発ツールの Application Builder は、ActiveX クライアント・アプリケーションの対話先となる BEA Tuxedo ドメイン内の CORBA オブジェクトを選択するためにクライアント開発ツール (Visual Basic など) と一緒に使用します。

Application Builder は、ActiveX Client への主要なユーザ・インターフェイスです。Application Builder を使用すると、デスクトップ・アプリケーションで使用可能な CORBA オブジェクトを選択し、その CORBA オブジェクトの ActiveX ビューと、その ActiveX ビューをクライアント・コンピュータにデプロイするためのパッケージを作成できます。

図 1-1 に、ActiveX Client のしくみを示します。

図 1-1 ActiveX Client のしくみ



ActiveX ビューの命名規則

命名規則では、CORBA インターフェイスを ActiveX にマッピングして型と変数名の不整合を回避するためのアルゴリズムが定義されます。また、命名規則では、特定のオブジェクトの使い方も指定されます。すべての ActiveX メソッドの名前は、`DI` で始まります。

ActiveX Client は、CORBA インターフェイスのオートメーション・バインディングを作成するときにこの命名規則を参照します。CORBA インターフェイス名が `Account` の場合、そのインターフェイスのオートメーション・バインディング名は `DIAccount` となります。

多くの場合、CORBA インターフェイス名はモジュールと呼ばれる入れ子になったレベルの中でスコープ指定されますが、ActiveX ではスコープ指定は存在しません。名前の不整合を避けるため、ActiveX Client は異なるスコープの名前をインターフェイスの先頭に追加して、CORBA インターフェイスを ActiveX にマッピングします。

たとえば、Account という名前の CORBA インターフェイスが、OMG IDL ファイルで次のように定義されているとします。

```
module University
{
   module Student
     {
       interface Account
           {//Account インターフェイスのオペレーションと属性
           };
     };
};
```

CORBA では、このインターフェイスの名前は University::Student::Account です。ActiveX Client は、この名前を ActiveX 用に DIUniversity_Student_Account に変換します。

ActiveX クライアント・アプリケーションは、OLE オートメーション環境オブジェクトを使用して BEA Tuxedo ドメインで CORBA オブジェクトにアクセスします。ActiveX クライアント・アプリケーションは、BEA ActiveX Client を使用して CORBA オブジェクトへの要求を処理します。Active X Application Builder を使用すると、ActiveX クライアント・アプリケーションで使用可能な CORBA インターフェイスを選択して、その CORBA インターフェイスの ActiveX ビューと、その ActiveX ビューをクライアント・マシンにデプロイするためのパッケージを作成できます。これらのクライアント・アプリケーションは、Visual Basic や PowerBuilder などのオートメーション開発ツールを使用してビルドします。

OMG IDL

どのような分散アプリケーションでも、クライアント / サーバ・アプリケーションは通信を行うための基本的な情報を必要とします。たとえば、クライアント・アプリケーションは、要求できるオペレーションとその引数を知る必要があります。

Object Management Group (OMG) インターフェイス定義言語 (IDL) を使用すると、クライアント・アプリケーションへの CORBA インターフェイスを定義できます。OMG IDL で記述されたインターフェイス定義では、CORBA インターフェイスが完全に定義され、各オペレーションの引数が完全に指定されます。

OMG IDL は、純粋な宣言型言語です。このため、インプリメンテーションの詳細は含まれません。OMG IDL で指定されるオペレーションは、CORBA バインディングを提供する任意の言語で記述し、呼び出すことができます。

一般に、アプリケーション設計者が使用可能な CORBA インターフェイスとオペレーション用の OMG IDL ファイルをプログラマに提供し、プログラマがクライアント・アプリケーションを開発します。

インターフェイス・リポジトリ

インターフェイス・リポジトリには、CORBA オブジェクトのインターフェイスとオペレーションの定義が含まれています。インターフェイス・リポジトリに格納される情報は OMG IDL ファイルに定義される情報と同じですが、この情報には実行時にプログラマティックにアクセス可能です。

ActiveX クライアント・アプリケーションは、インターフェイス・リポジトリを使用していることを認識しません。BEA ActiveX Client は、CORBA オペレーションを使用して、インターフェイス・リポジトリから CORBA オブジェクトに関する情報を取得します。

インターフェイス・リポジトリを管理するには、以下の BEA Tuxedo 開発コマンドを使用します。

- `idl2ir` コマンドを使用すると、インターフェイス・リポジトリに CORBA インターフェイスを追加できます。インターフェイス・リポジトリが存在しない場合は、このコマンドでインターフェイス・リポジトリを作成できます。また、インターフェイス・リポジトリ内の CORBA インターフェイスを更新できます。
- `ir2idl` コマンドを使用すると、インターフェイス・リポジトリの内容から OMG IDL ファイルを作成できます。
- `irdel` コマンドを使用すると、インターフェイス・リポジトリから CORBA インターフェイスを削除できます。

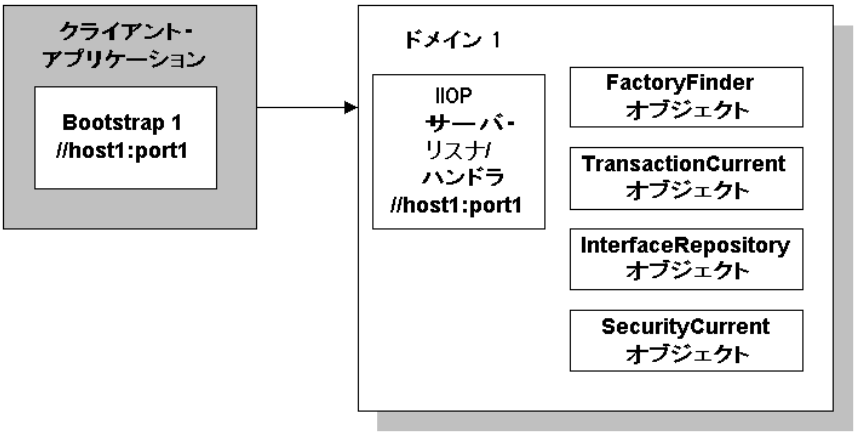
ドメイン

ドメインは、オブジェクトとサービスを管理エンティティとしてグループ化するための方法です。BEA Tuxedo ドメインは、最低 1 つの IIOP サーバ・リスナ / ハンドラ (ISL/ISH) を持ち、名前で識別されます。クライアント・アプリケーションは、複数の Bootstrap オブジェクトを使用して複数の BEA Tuxedo ドメインに接続できます。BEA Tuxedo ドメインごとに、クライアント・アプリケーションは、FactoryFinder オブジェクト、InterfaceRepository オブジェクト、SecurityCurrent オブジェクト、および TransactionCurrent オブジェクトを取得できます。各オブジェクトは、BEA Tuxedo ドメイン内で提供されるサービスに対応しています。Bootstrap オブジェクト、FactoryFinder オブジェクト、InterfaceRepository オブジェクト、SecuritiyCurrent オブジェクト、TransactionCurrent オブジェクトの説明については、この章の「環境オブジェクト」を参照してください。

注記　TransactionCurrent オブジェクトと SecurityCurrent オブジェクトが同時に存在できるのは、それぞれ 1 つだけです。また、ともに同じ Bootstrap オブジェクトに関連付けられている必要があります。

図 1-2 に、BEA Tuxedo ドメインのしくみを示します。

図 1-2　BEA Tuxedo ドメインのしくみ



環境オブジェクト

BEA Tuxedo ソフトウェアは、特定の BEA Tuxedo ドメインでクライアント・アプリケーションとサーバ・アプリケーションの間の通信を設定する環境オブジェクトのセットを提供します。BEA Tuxedo ソフトウェアには、以下の環境オブジェクトが用意されています。

- **Bootstrap**

  このオブジェクトは、クライアント・アプリケーションと BEA Tuxedo ドメインとの通信を確立します。また、BEA Tuxedo ドメインで他の環境オブジェクトのオブジェクト・リファレンスを取得します。

- **FactoryFinder**

  この CORBA オブジェクトは、CORBA オブジェクトのオブジェクト・リファレンスを代わりに作成できるファクトリを特定します。

- **SecurityCurrent**

  このオブジェクトを使用すると、クライアント・アプリケーションのログを適切なセキュリティで BEA Tuxedo ドメインに記録できます。BEA Tuxedo ソフトウェアは、CORBA サービス・セキュリティ・サービスのインプリメンテーションを提供します。

- **TransactionCurrent**

  このオブジェクトを使用すると、クライアント・アプリケーションはトランザクションに参加できるようになります。BEA Tuxedo ソフトウェアは、CORBA のオブジェクト・トランザクション・サービス (OTS) のインプリメンテーションを提供します。

- **UserTransaction**

  このオブジェクトを使用すると、クライアント・アプリケーションはトランザクションに参加できるようになります。BEA Tuxedo ソフトウェアは、Sun Microsystems, Inc. の Java Transaction Application Programming Interface (JTA API) のインプリメンテーションを提供します。このオブジェクトは、Java クライアントおよびサーバ・アプリケーションのみでサポートされます。

- **InterfaceRepository**

  この CORBA オブジェクトには、使用可能なすべての CORBA インターフェイスのインターフェイス定義と、CORBA インターフェイスへのオブジェクト・リファレンスを作成するためのファクトリが含まれています。

BEA Tuxedo ソフトウェアは、オートメーション・プログラミング環境用の環境オブジェクトを提供します。

Bootstrap オブジェクト

クライアント・アプリケーションは、Bootstrap オブジェクトを作成します。ISL/ISH のリストは、パラメータの形で、または環境変数 TOBJADDR や Java プロパティを介して提供できます。単一の ISL/ISH は、次の形式で指定します。

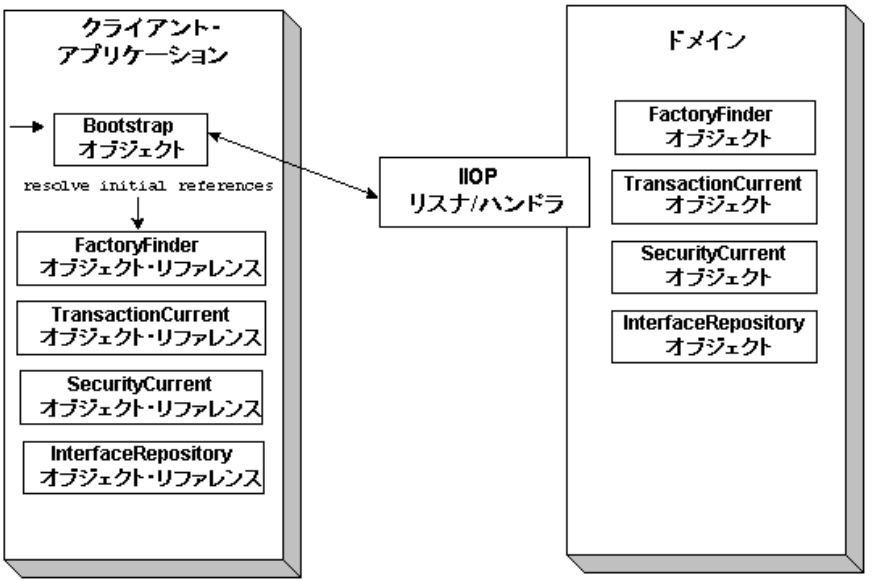
//host:port

例 : `//myserver:4000`

Bootstrap オブジェクトがインスタンス化されると、`resolve_initial_references` メソッドが呼び出され、文字列 ID が受け渡されて使用可能なオブジェクトへのリファレンスが取得されます。文字列 ID で有効な値は、FactoryFinder、TransactionCurrent、SecurityCurrent、および InterfaceRepository です。

図 1-3 に、BEA Tuxedo ドメインでの Bootstrap オブジェクトの機能を示します。

図 1-3 Bootstrap オブジェクトの機能



ファクトリと FactoryFinder オブジェクト

クライアント・アプリケーションは、CORBA オブジェクトへのリファレンスをファクトリから取得します。ファクトリは、別の CORBA オブジェクトへのリファレンスを返し、自身を FactoryFinder オブジェクトに登録する任意の CORBA オブジェクトです。

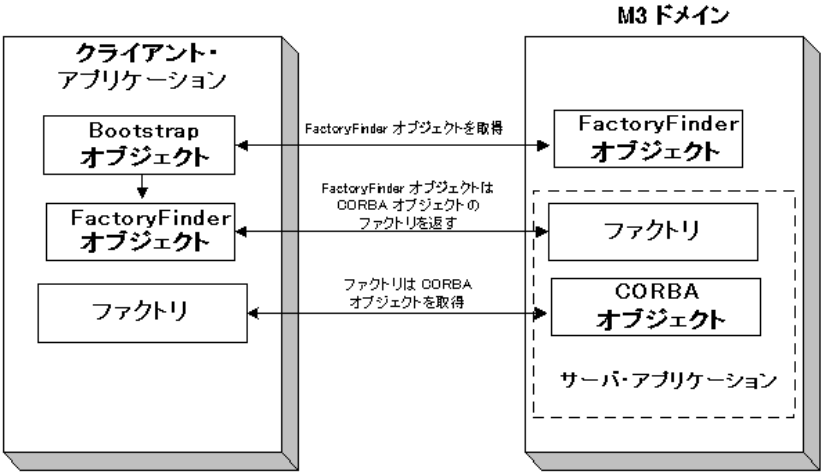
CORBA オブジェクトを使用するには、クライアント・アプリケーションは、CORBA オブジェクトのオブジェクト・リファレンスを作成するファクトリを見つけられるようにする必要があります。BEA Tuxedo ソフトウェアには、そのために FactoryFinder オブジェクトが用意されています。クライアント・アプリケーションで使用可能なファクトリは、BEA Tuxedo サーバによって起動時に FactoryFinder オブジェクトで登録されるファクトリです。

クライアント・アプリケーションは、以下の一連のステップを使用して CORBA オブジェクトへのリファレンスを取得します。

1. Bootstrap オブジェクトが作成されると、`resolve_initial_references` メソッドが呼び出され、FactoryFinder オブジェクトへのリファレンスが取得されます。
2. クライアント・アプリケーションは、FactoryFinder オブジェクトに目的のファクトリへのオブジェクト・リファレンスを問い合わせます。
3. クライアント・アプリケーションは、ファクトリを呼び出して CORBA オブジェクトへのオブジェクト・リファレンスを取得します。

図 1-4 は、クライアント・アプリケーションと FactoryFinder オブジェクトとの対話を示しています。

図 1-4　クライアント・アプリケーションによる FactoryFinder オブジェクトの使用



FactoryFinder オブジェクトの命名規則と BEA Tuxedo 拡張

クライアント・アプリケーションで使用可能なファクトリは、BEA Tuxedo サーバによって起動時に FactoryFinder オブジェクトで登録されるファクトリです。ファクトリは、以下のフィールドで構成されるキーを使用して登録されます。

- ファクトリのインターフェイスのインターフェイス・リポジトリ ID
- ファクトリへのオブジェクト・リファレンス

BEA Tuxedo ソフトウェアによって使用される FactoryFinder オブジェクトは、CORBA サービス・ライフサイクル・サービスで定義されます。BEA Tuxedo ソフトウェアは、`COS::LifeCycle::FactoryFinder` の拡張を実装します。この拡張により、クライアント・アプリケーションは FactoryFinder オブジェクトを使用してより簡単にファクトリを検索できるようになります。

CORBA サービス・ライフサイクル・サービスは、CORBA サービス・ネーミング・サービスに定義された名前を使用して、`COS::LifeCycle::FactoryFinder` インターフェイスを介してファクトリを検索するよう指定しています。これらの名前は一連の *`NameComponent`* 構造で構成され、この構造は `ID` フィールドと `kind` フィールドで構成されています。

CORBA 名を使用したファクトリの検索は、クライアント・アプリケーションにとっては面倒です。数多くの呼び出しを行って適切な名前構造を構築し、ネーミング・サービス名を構築して `COS::LifeCycle::FactoryFinder` インターフェイスの `find_factories` メソッドに渡す必要があるからです。また、メソッドは複数のファクトリを返す場合があるため、クライアント・アプリケーションは適切なファクトリの選択と不要なオブジェクト・リファレンスの破棄を行う必要があります。

FactoryFinder オブジェクトは、単純なメソッド呼び出しによってインターフェイスを拡張することで、クライアント・アプリケーションがファクトリをより簡単に検索できるよう設計されています。

FactoryFinder 拡張の目的は、クライアント・アプリケーションに対して以下の簡素化を提供することです。

- クライアント・アプリケーションは ID フィールド用の単純な文字列パラメータを使用して、ID によってファクトリを検索できます。これにより、ク

ライアント・アプリケーションが名前構造を構築するために必要な作業が削減されます。

- FactoryFinder オブジェクトは、使用可能なファクトリのプールから選択することによってロード・バランシング機構を実装できます。
- 一連のオブジェクト・リファレンスの代わりに 1 つのオブジェクト・リファレンスを返すメソッドを提供します。これにより、クライアント・アプリケーションはシーケンスから 1 つのファクトリを選択し、不要なリファレンスを破棄するためのコードを提供する必要がなくなります。

最も単純なアプリケーション設計は、クライアント・アプリケーションで `Tobj::FactoryFinder::find_one_factory_by_id` メソッドを使用することで達成できます。このメソッドは、入力としてファクトリ ID 用の単純な文字列を受け付け、1 つのファクトリを クライアント・アプリケーションに返します。クライアント・アプリケーションは、名前コンポーネントを操作し、多くのファクトリから選択する必要がなくなります。

`Tobj::FactoryFinder::find_one_factory_by_id` メソッドを使用するには、アプリケーション設計者はクライアント・アプリケーションが特定の CORBA オブジェクト・インターフェイス用のファクトリを簡単に検索するために使用できるファクトリの命名規則を定義する必要があります。この命名規則では、特定のタイプの CORBA オブジェクト・インターフェイスのオブジェクト・リファレンスを提供するファクトリの複数のニーモニック型が定義されるのが理想的です。ファクトリは、これらの規則を使用して登録されます。たとえば、Student オブジェクトのオブジェクト・リファレンスを返すファクトリであれば、StudentFactory と呼ばれます。

OMG IDL ファイルでファクトリの実際のインターフェイス ID を使用するか、OMG IDL ファイルでファクトリ ID を定数として指定することをお勧めします。このテクニックを使用することにより、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命名の一貫性が保証されます。

SecurityCurrent オブジェクト

SecurityCurrent オブジェクトは、CORBA サービス・セキュリティ・サービスの BEA Tuxedo インプリメンテーションです。BEA Tuxedo セキュリティ・モデルは、認証に基づいています。SecurityCurrent オブジェクトを使用すると、適切なセキュリティのレベルを指定できます。利用できる認証レベルは以下のとおりです。

- TOBJ_NOAUTH

  認証は不要です。ただし、クライアント・アプリケーションは自身を認証できます。また、ユーザ名とクライアント名を指定する必要がありますが、パスワードを指定する必要はありません。

- TOBJ_SYSAUTH

  クライアント・アプリケーションは、BEA Tuxedo ドメインに対して自身を認証する必要があります。ユーザ名、クライアント名、およびクライアント・アプリケーション・パスワードを指定する必要があります。

- TOBJ_APPAUTH

  クライアント・アプリケーションは、TOBJ_SYSAUTH で必要な情報の他にも情報を提供する必要があります。デフォルトの BEA Tuxedo CORBA 認証サービスを BEA Tuxedo ドメイン構成で使用した場合、クライアント・アプリケーションは、ユーザ・パスワードを指定する必要があります。パスワードを指定しなかった場合、クライアント・アプリケーションは、BEA Tuxedo ドメインでカスタム認証サービスによって解釈される認証データを提供します。

注記 クライアント・アプリケーションが認証されておらず、セキュリティ・レベルが TOBJ_NOAUTH の場合、BEA Tuxedo ドメインの ISL/ISH は、ISL/ISH に送信されるユーザ名とクライアント・アプリケーション名でクライアント・アプリケーションを登録します。

BEA Tuxedo CORBA の SecurityCurrent オブジェクトでは、PrincipalAuthenticator プロパティと Credentials プロパティのみがサポートされています。クライアント・アプリケーションでの SecurityCurrent オブジェクトの使い方については、「第 5 章 セキュリティの使い方」を参照してください。

TransactionCurrent オブジェクト

TransactionCurrent オブジェクトは、CORBA のオブジェクト・トランザクション・サービスの BEA Tuxedo インプリメンテーションです。TransactionCurrent オブジェクトは、クライアント・アプリケーションとサーバ・アプリケーションの間の現在のセッションに対するトランザクション・コンテキストを保持します。TransactionCurrent オブジェクトを使用すると、クライアント・アプリケーションは、トランザクションの開始および終了、トランザクションのステータスの取得などのトランザクション・オペレーションを実行できます。

トランザクションは、インターフェイス単位で使用されます。アプリケーション設計者は、設計時に BEA Tuxedo アプリケーション内のどのインターフェイスでトランザクションを処理するかを決定します。次に、各インターフェイスのトランザクション方針を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) に定義します。トランザクション方針には、以下の種類があります。

- Never

  このインターフェイスは、トランザクションには関与しません。このインターフェイス用に作成されたオブジェクトをトランザクションに関与させることはできません。BEA Tuxedo ソフトウェアは、この方針内のインプリメンテーションがトランザクションに関与した場合に例外 (INVALID_TRANSACTION) を生成します。インターフェイスの UBBCONFIG ファイルで指定した AUTOTRAN 方針は無視されます。

- Optional (デフォルトの transaction_policy)

  このインターフェイスは、トランザクションに関与させることができます。要求がトランザクション型の場合、オブジェクトをトランザクションに関与させることができます。AUTOTRAN パラメータがインターフェイスの UBBCONFIG ファイルで指定されている場合、AUTOTRAN は有効になります。

- Always

  このインターフェイスは、常にトランザクションの一部でなくてはなりません。インターフェイスがトランザクションの一部ではない場合、トランザクションは TP フレームワークによって自動的に開始されます。トランザクションは、メソッド終了時にコミットされます (これは、optional トランザクション方針でオブジェクトに対して AUTOTRAN を指定した場合とほぼ同じ動作ですが、この動作を有効にするには管理コンフィギュレーションが必要な点が異なります。また、管理コンフィギュレーションによって上書きすることはできません)。

- Ignore

  このインターフェイスは、トランザクションには関与しません。このインターフェイスをトランザクションに含めることはできますが、UBBCONFIG ファイルのこのインプリメンテーションで指定した AUTOTRAN 方針は無視されます。

クライアント・アプリケーションでの TransactionCurrent オブジェクトの使い方については、「第 6 章 トランザクションの使い方」を参照してください。

InterfaceRepository オブジェクト

InterfaceRepository オブジェクトは、特定の BEA Tuxedo ドメインの**インターフェイス・リポジトリ**に関する情報を返します。InterfaceRepository オブジェクトは、インターフェイス・リポジトリの CORBA 定義に基づいています。「Common Request Broker Architecture and Specification, Version 2.2」によって定義された適切な CORBA インターフェイスのセットを提供します。

ActiveX クライアント・アプリケーションは、InterfaceRepository オブジェクトを使用していることを認識しません。ActiveX クライアント・アプリケーションは、Bootstrap オブジェクトを使用して InterfaceRepository オブジェクトへのオブジェクト・リファレンスを取得します。

2 ActiveX クライアント・アプリケーションの作成

ここでは、次の内容について説明します。

- ActiveX クライアント・アプリケーションの開発プロセスの概要
- BEA Application Builder
- ステップ 1: オートメーション環境オブジェクトのインターフェイス・リポジトリへのロード
- ステップ 2: CORBA インターフェイスのインターフェイス・リポジトリへのロード
- ステップ 3: インターフェイス・リポジトリのサーバ・アプリケーションの起動
- ステップ 4: CORBA インターフェイスの ActiveX バインディングの作成
- ステップ 5: ActiveX バインディングのタイプ・ライブラリのロード
- ステップ 6: ActiveX クライアント・アプリケーションの記述
- ステップ 7: ActiveX クライアント・アプリケーションのデプロイ

ActiveX クライアント・アプリケーションを開発する前に理解しておく必要がある概念については、「第 1 章 概要」を参照してください。

ActiveX クライアント・アプリケーションの開発プロセスの概要

ActiveX クライアント・アプリケーションの作成手順は次のとおりです。

手順	説明
1	オートメーション環境オブジェクトをインターフェイス・リポジトリにロードします。
2	ActiveX クライアント・アプリケーションからアクセスする CORBA インターフェイスがインターフェイス・リポジトリにロードされていることを確認します。必要に応じて、CORBA インターフェイス用の Object Management Group (OMG) インターフェイス定義言語 (IDL) をインターフェイス・リポジトリにロードします。
3	インターフェイス・リポジトリに対してサーバ・アプリケーションのプロセスを開始します。
4	BEA Application Builder を使用して、CORBA オブジェクトのインターフェイス用に ActiveX バインディングを作成します。
5	ActiveX バインディング用のタイプ・ライブラリを開発ツールにロードします。

手順	説明
6	ActiveX クライアント・アプリケーションを記述します。ここでは、基本的なクライアント・アプリケーションの作成方法を説明します。ActiveX クライアント・アプリケーションでセキュリティおよびトランザクションをインプリメントすることもできます。 ■ ActiveX クライアント・アプリケーションでセキュリティをインプリメントする際の詳細については、「第 5 章 セキュリティの使い方」を参照してください。 ■ ActiveX クライアント・アプリケーションでトランザクションを使用する際の詳細については、「第 6 章 トランザクションの使い方」を参照してください。
7	ActiveX クライアント・アプリケーションのデプロイメント・パッケージを作成します。

このプロセスの各手順については、以降の節で詳しく説明します。

ActiveX クライアント・アプリケーションの BEA Tuxedo 開発環境には、以下のものが含まれます。

- `idl2ir` コマンド。OMG IDL に定義されているインターフェイス定義をインターフェイス・リポジトリにロードします。
- Application Builder。CORBA オブジェクトのインターフェイス用の ActiveX バインディングを作成し、そのインターフェイス用のデプロイメント・パッケージを作成します。
- オートメーション環境オブジェクト。BEA Tuxedo ドメインの CORBA オブジェクトの ActiveX ビュー (ActiveX ビューと呼ばれる)、およびその ActiveX ビューによって提供されるサービスへのアクセスを提供します。

BEA Application Builder

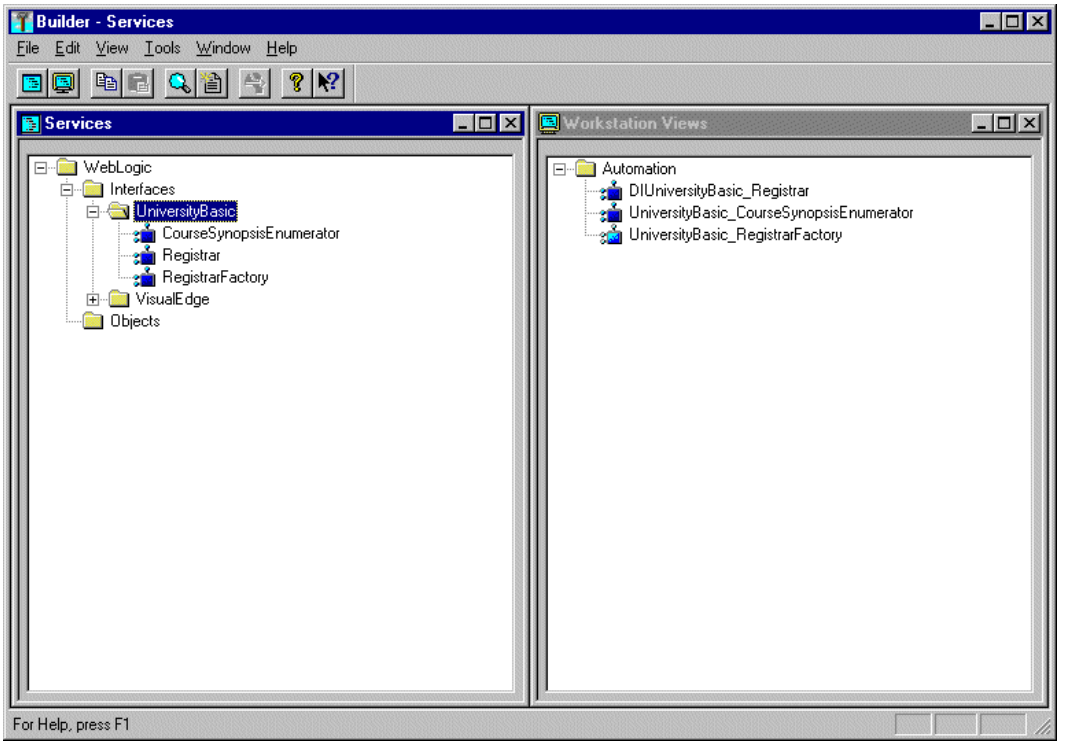
Application Builder は、CORBA オブジェクトの ActiveX ビューを作成する開発ツールです。Application Builder は、BEA ActiveX Client への主要なユーザ・インターフェイスです。Application Builder を使用すると、デスクトップ・アプリケーションで使用可能な CORBA オブジェクトを選択し、その CORBA オブジェクトの ActiveX ビューと、その ActiveX ビューをクライアント・マシンにデプロイするためのパッケージを作成できます。

ActiveX ビューを使用するには、CORBA オブジェクトのインターフェイスをインターフェイス・リポジトリにロードします。次に、CORBA インターフェイスの ActiveX バインディングを作成します。バインディングは、CORBA オブジェクトと ActiveX のインターフェイスを定義します。ActiveX クライアント・アプリケーションと生成されたバインディングの組み合わせにより、オブジェクトの ActiveX ビューが作成されます。

Application Builder の起動方法については、「第 4 章 タスク」の「Application Builder の起動」を参照してください。

図 2-1 で示すように、Application Builder のメイン・ウィンドウは、[Services] ウィンドウと [Workstation Views] ウィンドウの 2 つに分かれています。

図 2-1　Application Builder メイン・ウィンドウ



[Services] ウィンドウは、ローカル BEA Tuxedo ドメイン (BEA Tuxedo ソフトウェア・キットの一部としてインストールされている BEA Application Builder ソフトウェアの M3 ドメイン) でインターフェイス・リポジトリに含まれている CORBA モジュール、インターフェイス、およびオペレーションをすべて表示します。インターフェイス・リポジトリのすべてのインターフェイスに対してバインディングを作成できます。

[Services] ウィンドウの上部には、BEA Tuxedo ドメインで使用可能な各オブジェクト・システム用のエントリがあります。ActiveX Client は、BEA Tuxedo オブジェクト・システムのみをサポートします。オブジェクトは、インターフェイス・リポジトリで使用する同じ階層の形式で (つまり、モジュール、インターフェイス、オペレーション、およびオペレーションに含まれているパラメータとして) 表示されます。[+] 記号は、そのオブジェクトを展開すると他のオブジェクトが表示されることを示します。

[Workstation Views] ウィンドウは、CORBA インターフェイス用に作成されているすべての ActiveX バインディングを表します。CORBA インターフェイスに対するバインディングを作成するには、[Services] ウィンドウのエントリを [Workstation Views] ウィンドウにドラッグします。

Application Builder メイン・ウィンドウの詳細については、「第 3 章 Application Builder メイン・ウィンドウ」を参照してください。

以下の手順は、BEA Tuxedo に付属の University サンプル・アプリケーションを用いたものです。サンプル・アプリケーションの詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo CORBA University サンプル・アプリケーション』を参照してください。

ステップ 1: オートメーション環境オブジェクトのインターフェイス・リポジトリへのロード

オートメーション環境オブジェクトをインターフェイス・リポジトリにロードして、ActiveX クライアント・アプリケーションがそれらのオブジェクトのインターフェイス定義を使用できるようにします。MS-DOS プロンプトで、次のコマンドを入力して OMG IDL ファイル (`TOBJIN.idl`) をインターフェイス・リポジトリにロードします。

```
prompt> idl2ir -D _TOBJ -I drive:\tuxdir\include drive:\tuxdir\include\tobjin.idl
```

ステップ 2: CORBA インターフェイスのインターフェイス・リポジトリへのロード

CORBA オブジェクトの ActiveX ビューを作成する前に、CORBA オブジェクトのインターフェイスをインターフェイス・リポジトリにロードしておく必要があります。CORBA オブジェクトのインターフェイスがインターフェイス・リポジトリにロードされていない場合、Application Builder の [Services] ウィンドウにそれらが表示されません。目的の CORBA インターフェイスが [Services] ウィンドウに表示されない場合は、`idl2ir` コマンドを使用して、CORBA を定義する OMG IDL をインターフェイス・リポジトリにロードします。`idl2ir` コマンドの構文は次のとおりです。

```
idl2ir [repositoryfile.idl] file.idl
```

オプション	説明
repositoryfile	CORBA インターフェイスの OMG IDL ファイルを指定されたインターフェイス・リポジトリにロードします。ActiveX クライアント・アプリケーションがアクセスする BEA Tuxedo ドメインのインターフェイス・リポジトリの名前を指定します。
file.idl	CORBA インターフェイスの定義が記述されている OMG IDL ファイルを指定します。

`idl2ir` コマンドの詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo コマンド・リファレンス』を参照してください。

たとえば、University サンプル・アプリケーションの OMG IDL ファイルがインターフェイス・リポジトリにロードされている場合、以下の CORBA インターフェイスが Application Builder ウィンドウに表示されます。

- `RegistrarFactory`
- `Registrar`
- `CourseSynopsisEnumerator`

ステップ 3: インターフェイス・リポジトリのサーバ・アプリケーションの起動

ActiveX クライアント・アプリケーションは、実行時にインターフェイス・リポジトリから CORBA オブジェクトのインターフェイス定義を読み取り、それらをオートメーション・オブジェクトに変換します。このため、インターフェイス・リポジトリのサーバ・アプリケーションを起動して、インターフェイス定義を使用可能にする必要があります。UBBCONFIG ファイルを使用すると、インターフェイス・リポジトリに対してサーバ・アプリケーションのプロセスを開始できます。

注記　システム管理者が既にこのステップを実行している場合もあります。

BEA Tuxedo ドメインの UBBCONFIG ファイルで、インターフェイス・リポジトリのサーバ・アプリケーション、TMIFRSVR が起動しているかどうかをチェックします。UBBCONFIG ファイルには、以下のエントリが存在します。

```
TMIFRSVR
       SRVGRP = SYS_GRP
       SRVID = 6
       RESTART = Y
       MAXGEN = 5
       GRACE = 3600
```

さらに、ISL/ISH を開始する ISL パラメータが指定されていることを確認します。UBBCONFIG ファイルには、以下のエントリが存在します。

```
ISL
       SRVGRP = SYS_GRP
       SRVID  = 5
       CLOPT  = "-A -- -n //TRIXIE:2500"
```

ここで、TRIXIE はホスト (サーバ) システムの名前で、2500 はポート番号です。

サーバ・アプリケーションの起動、および ISL パラメータの指定の詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo アプリケーションの設定』を参照してください。

ステップ 4: CORBA インターフェイスの ActiveX バインディングの作成

ActiveX クライアント・アプリケーションが CORBA オブジェクトにアクセスするには、CORBA オブジェクトのインターフェイスの ActiveX バインディングを生成する必要があります。CORBA インターフェイス用の ActiveX バインディングを作成するには、Application Builder を使用します。

CORBA インターフェイス用の ActiveX バインディングを作成するには、次の手順に従います。

1. [BEA Tuxedo (C++)] プログラム・グループの [BEA Application Builder] アイコンをクリックします。

   [Domain] ログオン・ウィンドウが表示されます。

2. ログオン・ウィンドウで、UBBCONFIG ファイルの ISL パラメータで指定したホスト名とポート番号を入力します。UBBCONFIG ファイルで使用されている大文字、小文字をそのまま入力する必要があります。

   Application Builder のログオン・ウィンドウが表示されます。

3. 対象の CORBA インターフェイスを [Services] ウィンドウで選択して [Workstation Views] ウィンドウにドラッグするか、またはその CORBA インターフェイスを [Services] ウィンドウから切り取って [Workstation Views] ウィンドウに貼り付けます。

   Application Builder は、以下の処理を行います。

   - タイプ・ライブラリを作成します。デフォルトでは、タイプ・ライブラリは \tuxdir\TypeLibraries に格納されます。

     タイプ・ライブラリ・ファイル名は、DI*modulename_interfacename*.tlb です。

   - CORBA インターフェイス用の Windows システム・レジストリ・エントリ (各オブジェクト型の一意のプログラム ID を含む) を作成します。

これで、ActiveX クライアント・アプリケーションから ActiveX ビューを使用できます。

ステップ 5: ActiveX バインディングのタイプ・ライブラリのロード

ActiveX クライアント・アプリケーションを記述する前に、CORBA インターフェイスの ActiveX バインディングが定義されているタイプ・ライブラリを開発ツールにロードしておく必要があります。タイプ・ライブラリをロードするには、使用する開発ツールの説明に従います。

たとえば Visual Basic の場合、使用可能なタイプ・ライブラリのリストを取得するには、[プロジェクト] メニューの [参照設定] オプションを選択します。次に、そのリストから目的のタイプ・ライブラリを選択します。

デフォルトでは、Application Builder は生成されたすべてのタイプ・ライブラリを `\tuxdir\TypeLibraries` に格納します。CORBA インターフェイスの ActiveX バインディング用のタイプ・ライブラリは、次の形式を取ります。

```
DImodulename_interfacename.tlb
```

ステップ 6: ActiveX クライアント・アプリケーションの記述

ActiveX クライアント・アプリケーションは、以下の処理を行う必要があります。

1. オートメーション環境オブジェクト、ActiveX ビューのファクトリ、および ActiveX ビューの宣言のインクルード
2. BEA Tuxedo ドメインとの通信の確立
3. Bootstrap オブジェクトの使用による FactoryFinder オブジェクトへのリファレンスの取得
4. ファクトリの使用による ActiveX ビューへのオブジェクト・リファレンスの取得
5. ActiveX ビューのオペレーションの呼び出し

6. コールバックに対するオートメーション・サーバの作成
7. ActiveX クライアント・アプリケーションのデプロイ

以下の節では、Basic University サンプル・アプリケーションの ActiveX クライアント・アプリケーションの一部を使用して、これらのステップについて説明します。

オートメーション環境オブジェクト、ファクトリ、および CORBA オブジェクトの ActiveX ビューの宣言のインクルード

実行時のエラーを防ぐため、以下のオブジェクト型を宣言する必要があります。

- オートメーション環境オブジェクト
- CORBA オブジェクトの ActiveX ビューを作成するファクトリ
- ActiveX ビュー

次の例は、Bootstrap オブジェクトと FactoryFinder オブジェクト、Registrar オブジェクトの ActiveX ビュー用のファクトリ、および Registrar オブジェクトの ActiveX ビューを宣言する Visual Basic コードです。

```
\\Declare Bootstrap object\\
    Public objBootstrap As DITobj_Bootstrap
\\Declare FactoryFinder object\\
    Public objFactoryFinder As DITobj_FactoryFinder
\\Declare factory object for Registrar Object\\
    Public objRegistrarFactory As DIUniversityB_RegistrarFactory
\\Declare the ActiveX view of the Registrar object\\
    Public objRegistrar As DIUniversityB_Registrar
```

BEA Tuxedo ドメインとの通信の確立

ActiveX クライアント・アプリケーションを記述する場合、以下の 2 つのステップで BEA Tuxedo ドメインとの通信を確立します。

1. Bootstrap オブジェクトを作成します。

2. Bootstrap オブジェクトを初期化します。

次の Visual Basic サンプルは、CreateObject オペレーションを使用して Bootstrap オブジェクトを作成する方法を示したものです。

```
Set objBootstrap = CreateObject("Tobj.Bootstrap")
```

次に、Bootstrap オブジェクトを初期化します。Bootstrap オブジェクトを初期化する場合、次のように、対象の BEA Tuxedo ドメインの ISL/ISH のホストとポートを指定します。

```
objBootstrap.Initialize "//host:port"
```

ISL/ISH のホストとポートの組み合わせは、UBBCONFIG ファイルの ISL パラメータで定義されます。Bootstrap オブジェクトに対して指定するホストとポートの組み合わせは、この ISL パラメータと完全に一致する必要があります。ホストとポートの組み合わせの形式のほかに、大文字、小文字も一致する必要があります。アドレスが一致しない場合、Bootstrap オブジェクトの呼び出しは失敗し、次のメッセージがログ・ファイルに記録されます。

```
Error: Unofficial connection from client at <tcp/ip address>/<portnumber>
```

たとえば、UBBCONFIG ファイルの ISL パラメータでネットワーク・アドレスが //TRIXIE::3500 として指定されている場合、Bootstrap オブジェクトで //192.12.4.6.:3500 または //trixie:3500 と指定すると接続が失敗します。

BEA Tuxedo ドメインは、複数の ISL/ISH を持つことができます。複数の ISL/ISH を使用して BEA Tuxedo ドメインにアクセスする場合、*host:port* の組み合わせのリストを Bootstrap オブジェクトに提供します。Bootstrap オブジェクトは、BEA Tuxedo ドメインに接続するまでこのリストを参照します。また、ISL/ISH のリストは、TOBJADDR 環境変数に指定することもできます。

複数の BEA Tuxedo ドメインにアクセスする場合、アクセスする BEA Tuxedo ドメインごとに Bootstrap オブジェクトを作成する必要があります。

FactoryFinder オブジェクトへの初期リファレンスの取得

クライアント・アプリケーションは、自身にサービスを提供するオブジェクトへの初期リファレンスを取得する必要があります。Bootstrap オブジェクトは、FactoryFinder オブジェクト、SecurityCurrent オブジェクト、および

TransactionCurrent オブジェクトへのリファレンスを取得するのに使用します。このオペレーションに受け渡す引数は、目的のオブジェクトの progid を含む文字列です。ActiveX クライアント・アプリケーションで使用するオブジェクトの初期リファレンスだけを取得する必要があります。

次の Visual Basic サンプルは、Bootstrap オブジェクトを使用して FactoryFinder オブジェクトへのリファレンスを取得する方法を示したものです。

```
Set objFactoryFinder = objBootstrap.CreateObject("Tobj.FactoryFinder")
```

ファクトリを使用した ActiveX ビューの取得

ActiveX クライアント・アプリケーションは、CORBA オブジェクトの ActiveX ビューへのインターフェイス・ポインタをファクトリから取得します。ファクトリは、別の CORBA オブジェクトへのオブジェクト・リファレンスを返す任意の CORBA オブジェクトです。ActiveX クライアント・アプリケーションは、ファクトリのオペレーションを呼び出して特定のタイプの CORBA オブジェクトへのオブジェクト・リファレンスを取得します。ファクトリを使用するには、ActiveX クライアント・アプリケーションは必要なファクトリを検索できなければなりません。FactoryFinder オブジェクトは、そのために役立ちます。

CreateObject 関数を使用して FactoryFinder オブジェクトを作成し、次に FactoryFinder オブジェクトのメソッドの 1 つを使用してファクトリを検索します。FactoryFinder オブジェクトには以下のメソッドがあります。

- `find_factories()`

  入力キーに完全に一致するファクトリのシーケンスを返します。

- `find_one_factory()`

  入力キーに一致する 1 つのファクトリを返します。

- `find_factories_by_id()`

  名前コンポーネントの ID フィールドが入力引数に一致するファクトリのシーケンスを返します。

- `find_one_factory_by_id()`

  ファクトリの CORBA 名前コンポーネントの ID フィールドが入力引数に一致する 1 つのファクトリを返します。

- `list_factories()`

  現在 FactoryFinder で登録されているファクトリ・オブジェクトをリストします。

次の Visual Basic 例は、BEA Tuxedo University サンプル・アプリケーションで FactoryFinder の `find_one_factory_by_id()` メソッドを使用して、クライアント・アプリケーションで使用する Registrar オブジェクトのファクトリを取得する方法を示します。

```
Set objRegistrarFactory =
    objBsFactoryFinder.find_one_factory_by_id ("RegistrarFactory")
Set objRegistrar = RegistrarFactory.find_registrar
```

ActiveX ビューのオペレーションの呼び出し

ActiveX ビューのオペレーションを呼び出すには、**ファクトリ**へのポインタと**オペレーション**で必要な引数を受け渡します。

次の Visual Basic のサンプルは、ActiveX ビューのオペレーションを呼び出す方法を示したものです。

```
'Registrar オブジェクトからコースの詳細を取得'
aryCourseDetails =
           objRegistrar.get_course_details(aryCourseNumbers)
```

コールバックに対するオートメーション・サーバの作成

一部のアプリケーション開発シナリオでは、ActiveX クライアント・アプリケーションが CORBA サーバ・アプリケーションからの要求に応答できるようにすることが望ましい場合があります。CORBA サーバからのコールバックの原理には、一部のイベントが発生したときにクライアント・アプリケーションに通知すること、セキュリティを検証すること、クライアントからの追加情報を取得することなどがあります。たとえば、株価をトラッキングするクライアント・アプリケーションは、指定した株価が変動したときに通知するよう CORBA サーバに要求できます。クライアントは、通知オブジェクト・リファレンスを CORBA サーバに渡してこれを実行します。サーバは、このリファレンスを基に、株価が変動したときにクライアントに通知するためにコールバックします。次に説明す

る、COM サーバとして機能する ActiveX クライアント・アプリケーションの開発手順は、Visual Basic で ActiveX クライアントを開発することを前提にしています。

CORBA アプリケーションに関連して COM サーバとして機能する ActiveX アプリケーションを開発するには、上記の 6 つの手順を実行します。そのうえで、Visual Basic で 適切な Visual Basic クラスを作成し、CORBA インターフェイスに対する COM サーバ機能をインプリメントします。

この方法には、Visual Basic の [**プロジェクト**] メニューの [**クラス モジュールの追加**] オプションを選択して開始することなどがあります。Application Builder で作成したタイプ・ライブラリに表示された CORBA インターフェイスのオートメーション・ビューを指定するクラスに `Implements` 句を追加します (「ステップ 4: CORBA インターフェイスの ActiveX バインディングの作成」を参照)。次に例を示します。

```
Implements ChatClient_Listener
```

この例は、BEA Tuxedo に付属のチャットルーム Visual Basic クライアントのサンプルから抜粋したものです。チャットルーム・サンプルは、デフォルトでは次の場所にあります。

```
tuxdir\samples\corba\chatroom
```

この例では、`ChatClient_Listener` はインターフェイス名です。次に、プライベートな Visual Basic サブルーチンを記述し、インターフェイスに含まれる各メソッドをインプリメントします。次に例を示します。

```
Private Sub ChatClient_Listener_post(ByVal from As String,
ByVal output_line As String, Optional exceptionInfo As Variant)
MsgBox "User " + from + ": " + output_line
End Sub
```

COM オブジェクトのインスタンスの作成

COM オブジェクトをインプリメントしたので、ActiveX クライアント・アプリケーションでそのインスタンスを作成し、CORBA サービスに渡すことができます。通常の COM オブジェクトのインスタンスを作成するのと同じ方法で、これらの COM オブジェクトのインスタンスを作成します。次に例を示します。

```
Dim aListener as ChatClient_Listener
Set aListener = New MyListener
```

New を呼び出すと、インスタンスが作成されます。上の例では、ChatClient_Listener はインターフェイス名、MyListener は、インプリメント用に作成したクラス名です。インスタンスを作成したら、CORBA メソッドへのパラメータとして指定できます。次に例を示します。

```
aModerator.signon "Hansel", aListener
```

ここでは、aModerator は CORBA オブジェクト、aListener は、CORBA オブジェクトが必要に応じてコールバックする COM オブジェクトです。

ステップ 7: ActiveX クライアント・アプリケーションのデプロイ

ActiveX クライアント・アプリケーションをほかのクライアント・マシンに配布するには、デプロイメント・パッケージを作成する必要があります。デプロイメント・パッケージには、クライアント・アプリケーションが CORBA オブジェクトの ActiveX ビューを使用するために必要なすべてのデータ (バインディング、タイプ・ライブラリ、登録情報など) が含まれています。デプロイメント・パッケージは自動登録を行う ActiveX コントロールで、ファイル拡張子は .ocx です。

ActiveX ビューのデプロイメント・パッケージを作成するには、次の手順に従います。

1. [Workstation Views] ウィンドウから、ActiveX ビューを選択します。
2. [Tools] の [Deploy Modules] をクリックするか、または目的のビューを右クリックしてメニューから [Deploy Modules] を選択します。確認ウィンドウが表示されます。
3. [Create] をクリックすると、デプロイメント・パッケージが作成されます。

   デフォルトでは、デプロイメント・パッケージは \tuxdir\Packages に格納されます。

3 Application Builder メイン・ウィンドウ

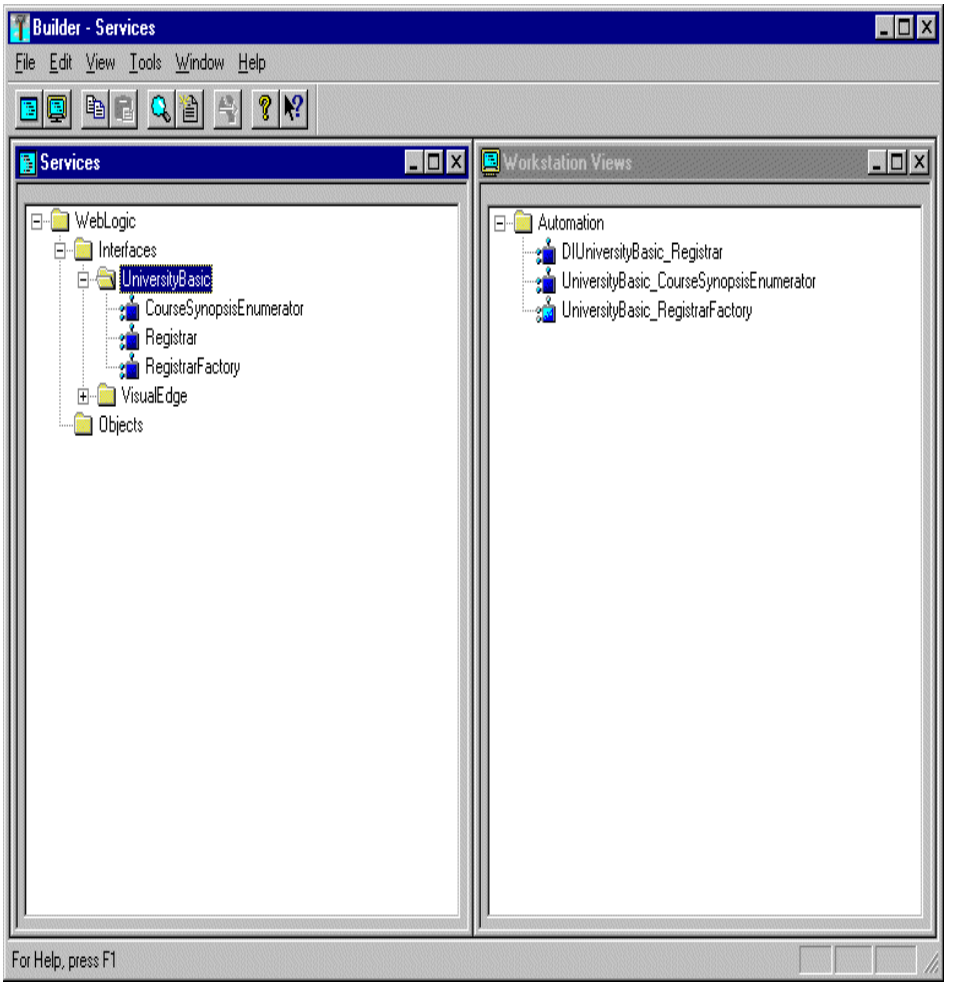
ここでは、次の内容について説明します。

- Application Builder メイン・ウィンドウ
- [Services] ウィンドウ
- [Workstation Views] ウィンドウ
- Application Builder のオブジェクト
- メニュー・オプション
- ツールバー・ボタン

Application Builder メイン・ウィンドウ

図 3-1 で示すように、Application Builder のメイン・ウィンドウは、[Services] ウィンドウと [Workstation Views] ウィンドウの 2 つに分かれています。

図 3-1 Application Builder メイン・ウィンドウ



Application Builder を起動すると、メイン・ウィンドウは、[Services] ウィンドウと [Workstation Views] ウィンドウを 1 つずつ表示します。[File] メニューの [New] オプションを使用すると、[Services] ウィンドウおよび [Workstation Views] ウィンドウを追加できます。[Window Menu] オプションを使用すると、[Services] ウィンドウおよび [Workstation Views] ウィンドウの配置を変更できます。

[Services] ウィンドウ

[Services] ウィンドウは、ローカル BEA Tuxedo ドメインのインターフェイス・リポジトリに含まれている CORBA モジュール、インターフェイス、オペレーションをすべて表示します。インターフェイス・リポジトリのすべてのアイテムに対してバインディングを作成できます。

[Services] ウィンドウの上部には、BEA Tuxedo ドメインで使用可能な各オブジェクト・システム用のエントリがあります。現在のリリースの ActiveX Client は、BEA Tuxedo オブジェクト・システムのみをサポートします。オブジェクトは、インターフェイス・リポジトリで使用する同じ階層の形式で (つまり、モジュール、インターフェイス、メソッド、およびメソッドに含まれているパラメータとして) 表示されます。[+] 記号は、そのオブジェクトを展開すると他のオブジェクトが表示されることを示します。

[Services] ウィンドウを使用して、属性、メソッド、データ型などの他のインターフェイス・リポジトリ定義を表示することもできます。[Options] ウィンドウの [Display] タブ・ページのオプションを使用すると、[Services] ウィンドウに表示するインターフェイス・リポジトリ定義の種類を選択できます ([Display] タブ・ページの詳細については、「メイン・ウィンドウに表示されるオブジェクトのフィルタ」を参照してください)。

Application Builder 内で追加の [Services] ウィンドウを開くには、[File] メニューの [New] から [Services Window] を選択します。

[Workstation Views] ウィンドウ

[Workstation Views] ウィンドウは、CORBA インターフェイス用に作成されているすべての ActiveX バインディングを表します。CORBA インターフェイスに対するバインディングを作成するには、[Services] ウィンドウのエントリをドラッグして [Workstation Views] ウィンドウにドロップします。

ActiveX オブジェクト・システムは、階層モジュール構造をサポートしていないので、[Workstation Views] ウィンドウの ActiveX バインディングのツリー構造が [Services] ウィンドウのツリー構造に一致しているとは限りません。Application Builder は、一意性を保証し、ActiveX オブジェクト・モデルの命名規則に準拠するようにバインディング名を変更します。

[Workstation Views] ウィンドウを使用して、属性、メソッド、データ型などの他のインターフェイス・リポジトリ定義を表示することもできます。[Options] ウィンドウの [Display] タブ・ページのオプションを使用すると、[Workstation Views] ウィンドウに表示するインターフェイス・リポジトリ定義の種類を選択できます ([Display] タブ・ページの詳細については、「メイン・ウィンドウに表示されるオブジェクトのフィルタ」を参照してください)。

Application Builder 内で追加の [Workstation Views] ウィンドウを開くには、[File] メニューの [New] から [Workstation Views Window] を選択します。

Application Builder のオブジェクト

表 3-1 では、Application Builder メイン・ウィンドウに表示されるオブジェクトについて説明します。

表 3-1 Application Builder のオブジェクトの説明

アイコン	説明
	BEA Tuxedo および ActiveX などの使用可能なオブジェクト・システムを示します。現在のリリースの ActiveX Client は、BEA Tuxedo システムのみをサポートします。
	パラメータとしてメソッドに渡される引数です。引数は、単一のデータ型 (整数、浮動小数点、文字など) または構造体 (浮動小数点、文字列、列挙など) に基づいています。

表 3-1 Application Builder のオブジェクトの説明 (続き)

アイコン	説明
	入力引数です。
	出力引数です。
	入出力引数です。
	データ構造体 (浮動小数点、文字列、列挙など) です。
	例外です。
	インターフェイス (メソッドおよびプロパティのセット) です。
	メソッド (オブジェクトで呼び出し可能なオペレーション) です。
	モジュール (1 つまたは複数のインターフェイスのグループ) です。
	プロパティ (オブジェクトに関連付けられたデータ属性) です。

表 3-1 Application Builder のオブジェクトの説明 (続き)

アイコン	説明
	現在の ActiveX ビューが、オブジェクトのソースとなるサーバ・アプリケーションであることを示します。

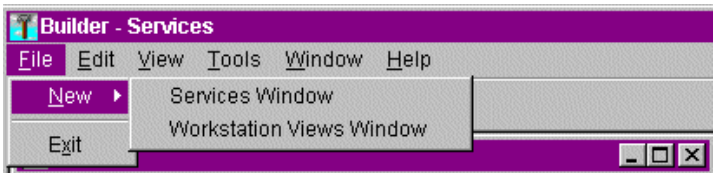
メニュー・オプション

ここでは、Application Builder のメニュー・オプションについて説明します。

[File] メニュー・オプション

図 3-2 に、[File] メニュー・オプションを示します。

図 3-2 展開された [File] メニュー



[File] メニューでは、以下のオプションを使用できます。

- [New] の [Services Window] オプションを選択すると、追加の [Services] ウィンドウが開きます。
- [New] の [Workstation Views] オプションを選択すると、追加の [Workstation Views] ウィンドウが開きます。
- [Exit] オプションを選択すると、Application Builder セッションが終了します。

[Edit] メニュー・オプション

図 3-3 に、[Edit] メニュー・オプションを示します。

図 3-3　展開された [Edit] メニュー

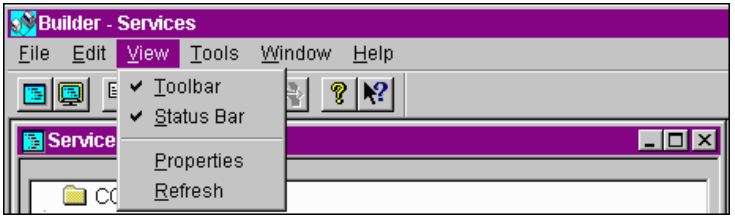
[Edit] メニューでは、以下のオプションを使用できます。

- [Copy] オプションを使用すると、[Services] ウィンドウの CORBA オブジェクトをクリップボードにコピーできます。その CORBA オブジェクトを [Workstation Views] ウィンドウに貼り付けると、CORBA オブジェクトのビューを作成できます。また、キーボード・ショートカット Ctrl+C で、このコピー・アクションを実行することもできます。[Copy] オプションは、[Workstation Views] ウィンドウでは使用できません。
- [Paste] オプションを使用すると、[Services] ウィンドウの CORBA オブジェクトを [Workstation Views] ウィンドウにコピーできます。また、キーボード・ショートカット Ctrl+V で、この貼り付けアクションを実行することもできます。その CORBA オブジェクトを [Workstation Views] ウィンドウに貼り付けると、CORBA オブジェクトの ActiveX ビューを作成できます。

[View] メニュー・オプション

図 3-4 に、[View] メニュー・オプションを示します。

図 3-4 展開された [View] メニュー



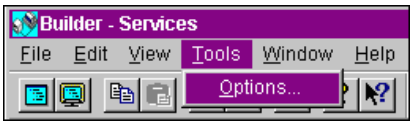
[View] メニューでは、以下のオプションを使用できます。

- [Toolbar] オプションを選択すると、ショートカットのツールバーの表示と非表示を切り替えることができます。
- [Status Bar] オプションを選択すると、Application Builder メイン・ウィンドウの下部にあるステータス・ウィンドウの表示と非表示を切り替えることができます。
- [Properties] オプションを選択すると、CORBA オブジェクトの特性または CORBA オブジェクトの ActiveX ビューを参照できます。
- [Refresh] オプションを選択すると、インターフェイス・リポジトリの新しいデータを基に全ウィンドウを更新できます。

[Tools] メニュー・オプション

図 3-5 に、[Tools] メニュー・オプションを示します。

図 3-5 展開された [Tools] メニュー



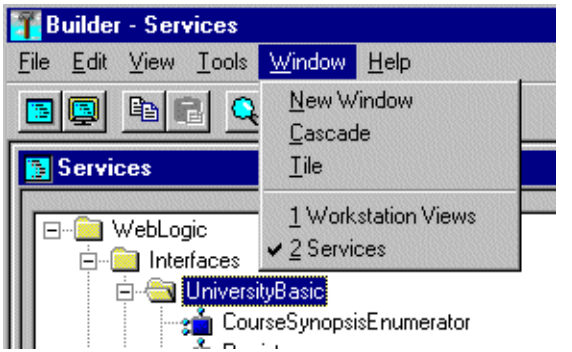
[Tools] メニューの [Options] オプションを選択すると、以下のダイアログ・ウィンドウがある [Options] ウィンドウが開きます。

- [Workstation Bindings] ー このウィンドウでは、バインディング作成時に使用するデフォルト設定を制御できます。
- [Deployment Packages] ー このウィンドウでは、デプロイメント・パッケージのデフォルト・ディレクトリを変更できます。
- [Display] ー このウィンドウでは、[Services] ウィンドウおよび [Workstation Views] ウィンドウに表示されるオブジェクトの種類を指定できます。

[Window] メニュー・オプション

図 3-6 に、[Window] メニュー・オプションを示します。

図 3-6 展開された [Window] メニュー



[Window] メニューでは、以下のオプションを使用できます。

- [New] オプションを選択すると、新しい [Services] ウィンドウか [Workstation Views] ウィンドウのいずれかを開くことができます。Application Builder は、アクティブなウィンドウと同じ種類のウィンドウを新規作成します。
- [Cascade] オプションを選択すると、開いている [Services] ウィンドウおよび [Workstation Views] ウィンドウを、タイトルが見えるように重ね合わせて配置できます。

- [Tile] オプションを選択すると、開いている [Services] ウィンドウおよび [Workstation Views] ウィンドウを並べて配置できます。

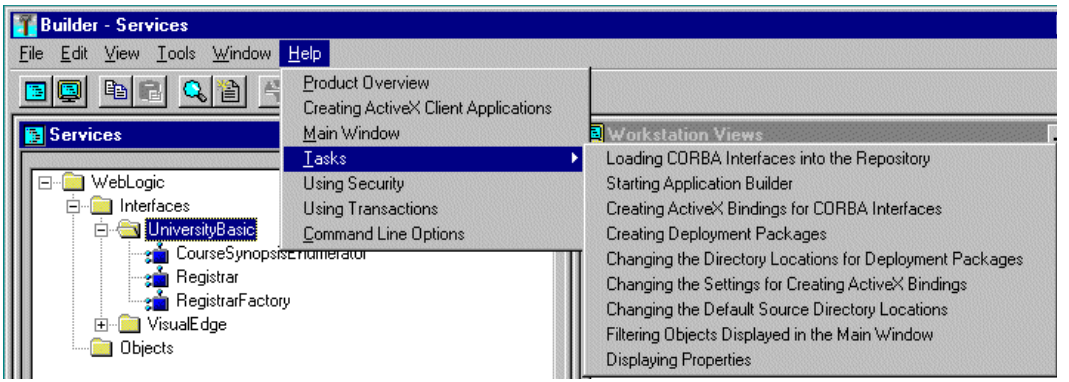
メニューの下半分に、開いている [Services] ウィンドウおよび [Workstation Views] ウィンドウがリストされています。チェック・マークは、現在アクティブなウィンドウを示します。

[Help] メニュー・オプション

[Help] メニュー・オプションを使用すると、直接 Application Builder コンポーネントの説明を参照できます。

図 3-7 に、[Help] メニュー・オプションを示します。

図 3-7 展開された [Help] メニュー



[Help] メニューでは、Application Builder のウィンドウおよび機能に関する説明を参照できます。

ツールバー・ボタン

図 3-8 に、Application Builder ツールバーを示します。

図 3-8　Application Builder ツールバー

ツールバーは、メイン・ウィンドウのメニュー・バーの下にあります。ツールバー・ボタンの機能を左から説明します。

- 新しい [Services] ウィンドウを開きます。
- 新しい [Workstation Views] ウィンドウを開きます。
- 選択したインターフェイスをクリップボードにコピーします。
- クリップボードの中身を、指定したウィンドウに貼り付けます。
- 選択されたインターフェイスまたはビューのプロパティを表示します。
- アクティブなウィンドウを最新の状態に更新します。
- 選択したインターフェイスのデプロイメント・パッケージを作成します。
- 製品、バージョン番号、著作権に関する情報を表示します。
- コンテキスト・センシティブ・ヘルプを表示します。

4　タスク

ここでは、次の内容について説明します。

- CORBA インターフェイスのインターフェイス・リポジトリへのロード
- Application Builder の起動
- CORBA インターフェイスに対する ActiveX バインディングの作成
- CORBA インターフェイスに対する ActiveX バインディング作成の設定変更
- デプロイメント・パッケージの作成
- デプロイメント・パッケージのディレクトリの変更
- CORBA インターフェイスに対する ActiveX バインディング作成の設定変更
- デフォルト・ディレクトリの変更
- メイン・ウィンドウに表示されるオブジェクトのフィルタ
- プロパティの表示

CORBA インターフェイスのインターフェイス・リポジトリへのロード

CORBA オブジェクトに対する ActiveX ビューを作成する前に、CORBA オブジェクトのインターフェイスをインターフェイス・リポジトリにロードする必要があります。CORBA オブジェクトのインターフェイスをインターフェイス・リポジトリにロードしない場合、[Services] ウィンドウにインターフェイスは表示されません。目的の CORBA インターフェイスが [Services] ウィンドウに表示されない場合は、`idl2ir` コマンドを使用し、CORBA インターフェイスの Object Management Group (OMG) インターフェイス定義言語 (IDL) をインターフェイス・リポジトリにロードします。`idl2ir` コマンドの構文は次のとおりです。

```
idl2ir -f repository-name file.idl
```

次の表は、idl2ir コマンドのオプションを示しています。

オプション	説明
-f repository-name	CORBA インターフェイスの OMG IDL ファイルをインターフェイス・リポジトリにロードします。ActiveX クライアント・アプリケーションと同じ BEA Tuxedo ドメインのインターフェイス・リポジトリを指定します。
file.idl	CORBA インターフェイスの定義を含む OMG IDL ファイルを指定します。

idl2ir コマンドの詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo CORBA プログラミング・リファレンス』を参照してください。

Application Builder の起動

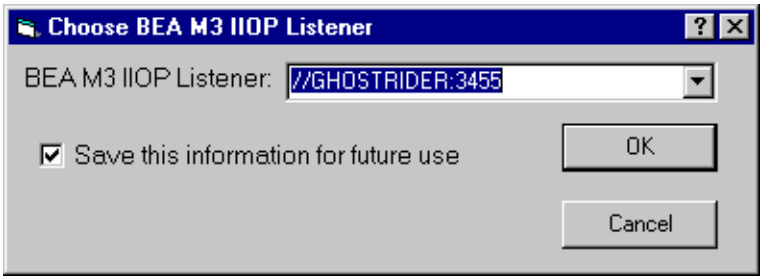
Application Builder を起動するには、以下の手順を実行します。

1. [BEA BEA Tuxedo System] プログラム・グループの [BEA Application Builder] アイコンをクリックします。

   ログオン・ウィンドウが表示されます。

2. ログオン・ウィンドウで、UBBCONFIG ファイルの ISL パラメータで指定したホスト名とポート番号を入力します。UBBCONFIG ファイルの大文字小文字を正しく区別して指定する必要があります。図 4-1 を参照してください。

図 4-1 IIOP リスナへの接続



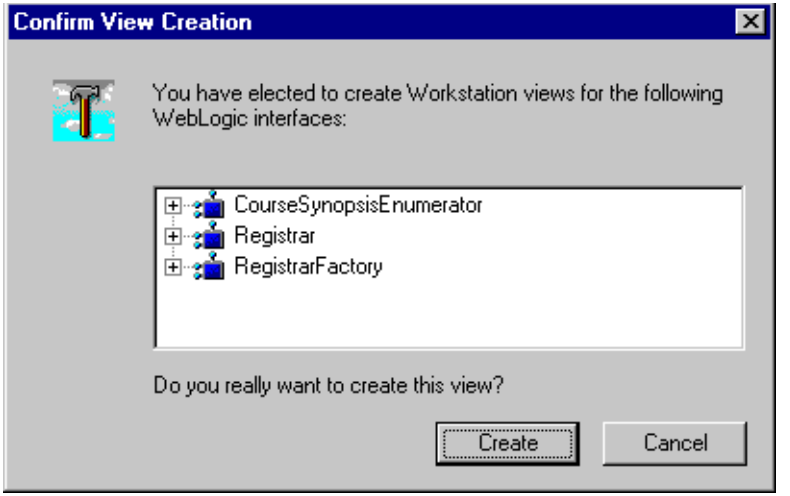
Application Builder ウィンドウが表示されます。インターフェイス・リポジトリにロードされたすべての CORBA インターフェイスが、Application Builder の [Services] ウィンドウに表示されます。

CORBA インターフェイスに対する ActiveX バインディングの作成

CORBA インターフェイスに対して ActiveX バインディングを作成するには、以下の手順を実行します。

1. Application Builder ウィンドウの [Services] ウィンドウで、目的の CORBA インターフェイスを選択します。
2. その CORBA インターフェイスを [Workstation Views] ウィンドウにドラッグするか、[Services] ウィンドウから CORBA インターフェイスを切り取って [Workstation Views] ウィンドウに貼り付けます。

   [Confirm View Creation] ウィンドウが表示されます。



3. CORBA インターフェイスに対して ActiveX バインディングを作成するには、[Create] をクリックします。

   Application Builder は、以下のものを作成します。

   - タイプ・ライブラリ。デフォルトでは、タイプ・ライブラリは `\tuxdir\TypeLibraries` に作成されます。

     タイプ・ライブラリ・ファイル名は、`DI`*`modulename_interfacename`*`.tlb` です。

   - オブジェクトのタイプごとの固有のプログラム ID を含む、CORBA インターフェイスの Windows システム・レジストリ・エントリ。

これで、AcrtiveX クライアント・アプリケーションから CORBA オブジェクトの ActiveX ビューを使用できます。

CORBA インターフェイスに対する ActiveX バインディング作成の設定変更

[Options] ウィンドウの [Workstation Bindings] タブ・ページで、CORBA オブジェクトのインターフェイスに対する ActiveX バインディングの作成に使用する設定を変更します。[Workstation Bindings] タブ・ページにアクセスするには、[Tools] の [Options] をクリックします。

表 4-1 は、[Workstation Bindings] タブ・ページのオプションを示しています。

表 4-1 [Workstation Bindings] タブ・ページのオプション

オプション	説明
[Workstation Bindings Options]	CORBA オブジェクトのインターフェイスに対して作成可能なバインディングのタイプをリストします。作成されるバインディングのタイプの横に、チェック・マークが表示されます。
[Generate COM Views on Workstation Drop]	CORBA オブジェクトのインターフェイスに対する COM バインディングを作成します。今回のリリースの ActiveX Client は、BEA Tuxedo ドメインでの CORBA オブジェクトの COM ビューをサポートしていません。
[Generate OLE Automation Views on Workstation Drop]	CORBA オブジェクトのインターフェイスに対する ActiveX バインディングを作成します。
[Create ActiveX Controls for OLE Automation Views]	CORBA オブジェクトを ActiveX コントロールとして使用できるようにするために、必要なインターフェイスを CORBA オブジェクトに追加します。また、CORBA オブジェクトを ActiveX コントロールとして登録します。CORBA オブジェクトは、ActiveX コントロール・コンテナ・アプリケーションで使用できるようになります。
[Output Folders]	CORBA オブジェクトのインターフェイスに対して作成されるバインディングのディレクトリを指定します。

表 4-1 [Workstation Bindings] タブ・ページのオプション (続き)

オプション	説明
[C++ Headers]	C++ ヘッダ・ファイルは、正しくコンパイルできるように、定義されたパスに配置する必要があります。デフォルトでは、ファイルは次の場所に配置されます。 `\tuxdir\Include` [Browse] ボタンをクリックすると、ディレクトリの場所を検索できます。
[MIDL/ODL Files]	Microsoft Definition Language (MIDL) ファイルおよびオブジェクト定義言語 (ODL) ファイルは、参照用にのみ提供されており、任意の場所に配置できます。デフォルトでは、ファイルは次の場所に配置されます。 `\tuxdir\TypeLibraries` [Browse] ボタンをクリックすると、ディレクトリの場所を検索できます。
[Type Libraries]	タイプ・ライブラリは、絶対パスで登録され、クライアント・コンピュータに対して常に使用可能なディレクトリに配置されます。デフォルトでは、ファイルは次の場所に配置されます。 `\tuxdir\TypeLibraries` [Browse] ボタンをクリックすると、ディレクトリの場所を検索できます。

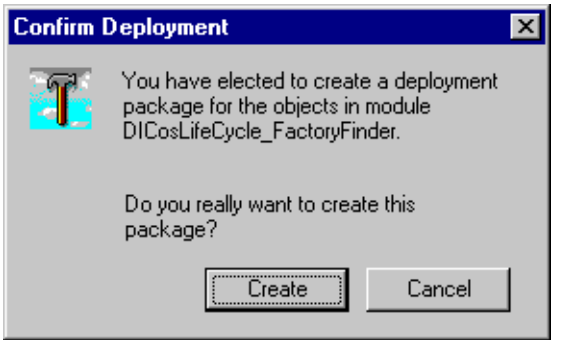
デプロイメント・パッケージの作成

クライアント・アプリケーションをほかのクライアント・コンピュータに配布するには、デプロイメント・パッケージを作成する必要があります。デプロイメント・パッケージには、バインディング、タイプ・ライブラリ、登録情報など、クライアント・アプリケーションが CORBA オブジェクトの ActiveX ビューを使用するのに必要な全データが含まれています。デプロイメント・パッケージは、拡張子 `.ocx` で自己登録する ActiveX コントロールです。

CORBA オブジェクトの ActiveX クライアント・アプリケーションのデプロイメント・パッケージを作成するには、以下の手順を実行します。

1. [Workstation Views] ウィンドウから ActiveX ビューを選択します。
2. [Tools] の [Deploy Modules] をクリックするか、目的のビューで右マウス・ボタンをクリックし、メニューから [Deploy Modules] を選択します。

   [Confirm Deployment] ウィンドウが表示されます。



3. [Create] をクリックして、デプロイメント・パッケージを作成します。

   デフォルトでは、デプロイメント・パッケージは `\tuxdir\Packages` にあります。

デプロイメント・パッケージのディレクトリの変更

[Options] ウィンドウの [Deployment Packages] タブ・ページで、CORBA オブジェクトの ActiveX ビューに対するデプロイメント・パッケージのディレクトリを変更します。[Deployment Packages] タブ・ページにアクセスするには、[Tools] の [Options] をクリックします。デプロイメント・パッケージの現在のディレクトリが表示されます。デフォルトのディレクトリは、`\tuxdir\Packages` です。

デフォルト・ディレクトリの変更

Application Builder は、C++ ヘッダ・ファイル、MIDL および ODL ファイル、タイプ・ライブラリのデフォルト・ディレクトリを提供します。ディレクトリの場所を変更できます。

ディレクトリの変更を変更するには、次の手順を実行します。

1. [Tools] メニューの [Options] を選択します。

   [Options] ウィンドウが表示されます。

2. [Options] ウィンドウの [Workstation Bindings] タブを選択します。

   [C++ Headers]、[MIDL/ODL Files]、[Type Libraries] フィールドに、それぞれのデフォルト・ディレクトリが表示されます。

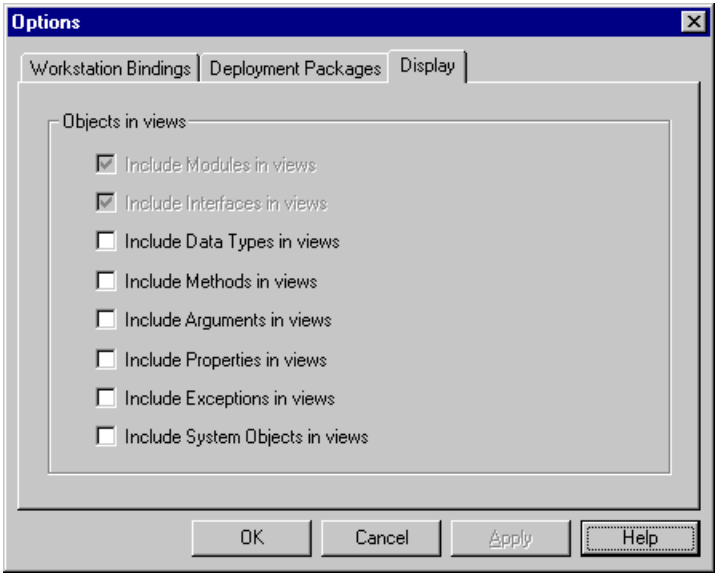
3. 目的の出力ディレクトリを選択して削除します。

4. 新しいディレクトリを入力するか、[Browse] ボタンをクリックして新しいディレクトリを検索できます。

5. [OK] をクリックして変更内容を保存します。

メイン・ウィンドウに表示されるオブジェクトのフィルタ

[Options] ウィンドウの [Display] タブ・ページを使用すると、Application Builder メイン・ウィンドウに表示されるオブジェクトの種類をフィルタできます。デフォルトでは、CORBA インターフェイスおよびモジュールが表示されます。



以下の種類の情報を表示することもできます。

- データ型
- メソッド
- 引数
- プロパティ
- 例外

[Include System Objects] オプションを選択すると、CosTransactions など、インターフェイス内の特定の定義の表示を有効にすることができます。

Application Builder メイン・ウィンドウに追加情報を表示するには、目的のオプションをチェックして [OK] をクリックします。

プロパティの表示

[Properties] ウィンドウを使用すると、選択したアダプタ、モジュール、インターフェイスのプロパティを示した 1 つまたは複数のページを表示できます。[Properties] ウィンドウの内容は、オブジェクトによって異なります。

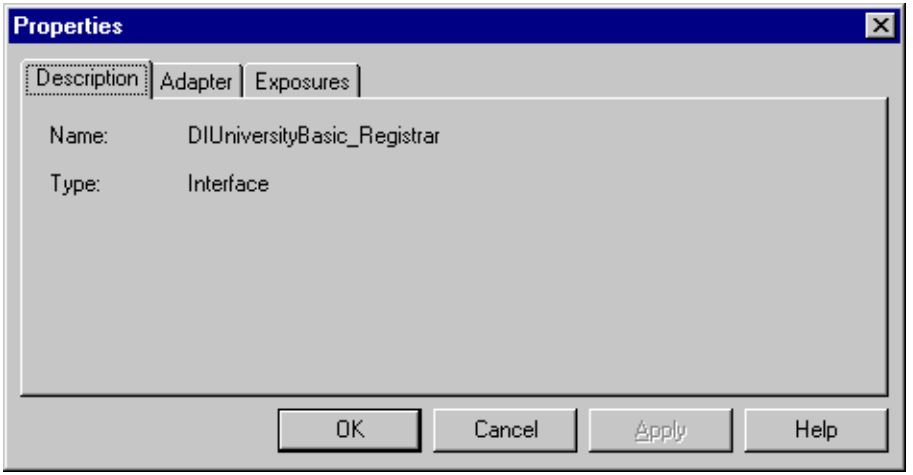


表 4-2 は、使用可能なプロパティを示しています。

表 4-2　プロパティの説明

プロパティ	説明
[Interface] の [Name]	選択した CORBA インターフェイス名。
[Interface] の [Type]	インターフェイス、モジュール、例外など、オブジェクトのタイプ。
[Adapter] の [Name]	オブジェクト・システム名。このリリースでは、このオプションは BEA Tuxedo バージョン 8.0 と表示されます。
[Adapter] の [Vendor]	オブジェクト・システムのベンダ名。このリリースでは、このオプションは BEA システムと表示されます。

表 4-2 プロパティの説明

プロパティ	説明
[Adapter] の [Platform]	オブジェクト・システムのバージョン。このオプションは、バージョン 8.0 と表示されます。
[Exposure]	BEA Tuxedo など、オブジェクトのソース・オブジェクト・システムを示します。

5　セキュリティの使い方

ここでは、BEA Tuxedo ソフトウェアの ActiveX クライアント・アプリケーションでセキュリティを使用する方法を説明します。

SecurityCurrent オブジェクトの概要については、「第 1 章 概要」を参照してください。

BEA Tuxedo セキュリティの概要

ActiveX クライアント・アプリケーションは、セキュリティを使用して BEA Tuxedo ドメインに対して自身を認証します。認証とは、クライアント・アプリケーションの ID を検証するプロセスです。正しいログオン情報を入力すると、クライアント・アプリケーションは、BEA Tuxedo ドメインに対して自身を認証します。BEA Tuxedo ソフトウェアは、CORBA サービス・セキュリティ・サービス で定義された認証を使用し、簡単に使用できるようにするための拡張を提供します。

クライアント・アプリケーションは、目的の BEA Tuxedo ドメインで定義されているセキュリティ・レベルに従ってセキュリティ情報を提供する必要があります。これらの情報は、BEA Tuxedo システム管理者が UBBCONFIG ファイルで BEA Tuxedo ドメインに対して定義します。クライアント・アプリケーションを作成するときに、BEA Tuxedo システム管理者と共同で作業し、クライアント・アプリケーションからアクセスする BEA Tuxedo ドメインに対して正しいセキュリティ情報 (ユーザ名およびユーザ・パスワードなど) を取得する必要があります。

セキュリティの開発プロセスの概要

セキュリティは、以下の手順でクライアント・アプリケーションに追加します。

手順	説明
1	Bootstrap オブジェクトを使用し、指定した BEA Tuxedo ドメインで SecurityCurrent オブジェクトへのリファレンスを取得します。
2	SecurityCurrent オブジェクトから PrincipalAuthenticator オブジェクトを取得します。
3	PrincipalAuthenticator オブジェクトの `get_auth_type` オペレーションを使用し、BEA Tuxedo ドメインで想定されている認証のタイプを返します。
4	セキュリティ情報を使用して BEA Tuxedo ドメインにログオンします。
5	BEA Tuxedo ドメインからログオフします。

以下の節では、Security University サンプル・アプリケーションにあるクライアント・アプリケーションの一部を使用して、この手順を説明します。

ステップ 1: Bootstrap オブジェクトを使用して SecurityCurrent オブジェクトを取得する

Bootstrap オブジェクトを使用し、指定した BEA Tuxedo ドメインで SecurityCurrent オブジェクトへのオブジェクト・リファレンスを取得します。SecurityCurrent オブジェクトは、CORBA サービス・セキュリティ・サービスで定義された `SecurityLevel2::Current` オブジェクトです。

次の Visual Basic の例は、Bootstrap オブジェクトを使用して SecurityCurrent オブジェクトを返す方法を示します。

```
Set objSecurityCurrent =
      objBootstrap.CreateObject("Tobj.SecurityCurrent")
```

ステップ 2: SecurityCurrent オブジェクトから PrincipalAuthenticator オブジェクトを取得する

SecurityCurrent オブジェクトは、BEA Tuxedo ドメインの PrincipalAuthenticator のリファレンスを返します。PrincipalAuthenticator オブジェクトは、BEA Tuxedo ドメインで要求されている認証レベルの取得に使用します。

次の Visual Basic の例では、BEA Tuxedo ドメインに対する PrincipalAuthenticator の取得方法を示します。

```
Set objPrincAuth = objSecurityCurrent.principal_authenticator
```

ステップ 3: 認証レベルを取得する

`Tobj::PrincipalAuthenticator::get_auth_type()` メソッドを使用し、BEA Tuxedo ドメインで要求されている認証のレベルを取得します。

次の Visual Basic の例では、BEA Tuxedo ドメインに対する PrincipalAuthenticator の取得方法を示します。

```
AuthorityType = objPrinAuth.get_auth_type
```

ステップ 4: 正しい認証で BEA Tuxedo ドメインにログオンする

Tobj::PrincipalAuthenticator::logon() メソッドを使用し、クライアント・アプリケーションから目的の BEA Tuxedo ドメインにログインします。このメソッドでは、以下の引数が必要です。

- *user_name*

  BEA Tuxedo ユーザ名です。この情報は、TOBJ_SYSAUTH および TOBJ_APPAUTH 認証レベルで必須です。TOBJ_NOAUTH 認証レベルで提供することもできますが、必須ではありません。システム設計者は、設計時にこの名前を決定します。

- *client_name*

  BEA Tuxedo クライアント・アプリケーション名です。この情報は、TOBJ_SYSAUTH および TOBJ_APPAUTH 認証レベルで必須です。TOBJ_NOAUTH 認証レベルで提供することもできますが、必須ではありません。この情報は、システム管理者から取得します。

- *system_password*

  BEA Tuxedo パスワードです。この情報は、TOBJ_SYSAUTH および TOBJ_APPAUTH 認証レベルで必須です。この情報は、システム管理者から取得します。

- *user_password*

  BEA Tuxedo 認証サービスのユーザ・パスワードです。この情報は、TOBJ_APPAUTH 認証レベルで必須です。

- *user_data*

  認証に関するアプリケーション固有のデータです。この情報は、クライアント・アプリケーションがアクセスしている BEA Tuxedo ドメインが、BEA Tuxedo ソフトウェアで提供されている認証サービスを使用していない場合に必須です。

user_password 引数と *user_data* 引数は相互に排他的であり、BEA Tuxedo ソフトウェアのコンフィギュレーションで使用する認証サービスによって異なります。BEA Tuxedo ソフトウェアで提供されている認証サービス以外の認証サービスを使用する場合は、*user_data* 引数の情報をログオンで提供する必要があり

ます。*user_password* と *user_data* の両方が設定されている場合、Tobj::PrincipalAuthenticator::logon() メソッドで CORBA::BAD_PARAM 例外が発生します。

BEA Tuxedo ドメインに TOBJ_NOAUTH 認証レベルがある場合、BEA Tuxedo ドメインへのログオン時に、*user_name* または *client_name* を提供する必要はありません。クライアント・アプリケーションが *user_name* および *client_name* でログオンしなかった場合、BEA Tuxedo ドメインの IIOP サーバ・リスナ / ハンドラ (ISL/ISH) は、UBBCONFIG ファイルの ISL/ISH に対して設定した *user_name* と *client_name* でクライアント・アプリケーションを登録します。ただし、クライアント・アプリケーションは、任意の *user_name* および *client_name* でログオンできます。

logon() メソッドは、以下のいずれかを返します。

- Security::AuthenticationStatus::SecAuthSuccess (認証に成功した場合)
- Security::AuthenticationStatus::SecAuthFailure (認証に失敗した場合、またはクライアント・アプリケーションが既に認証済みで、BEA Tuxedo ドメインからログオフしなかった場合)

次の Visual Basic の例では、Tobj::PrincipalAuthenticator::logon() メソッドの使用方法を示します。

```
If AuthorityType = TOBJ_APPAUTH Then logonStatus =
       oPrincAuth.Logon(
           UserName,ClientName,SystemPassword,_
           UserPassword,UserData)
End If
```

ステップ 5: BEA Tuxedo ドメインからログオフする

クライアント・アプリケーションは、同じ BEA Tuxedo ドメインに別のユーザとしてログオンする前に、現在の BEA Tuxedo ドメインからログオフする必要があります。Tobj::PrincipalAuthenticator::logoff() メソッドを使用し、BEA Tuxedo の現在の認証コンテキストおよびクリデンシャルを破棄します。このメ

ソッドは、BEA Tuxedo ドメインへのネットワーク接続をクローズします。認証タイプが TP_NOAUTH でない場合、BEA Tuxedo ドメインからログオフした後の既存の認証を使用した呼び出しは失敗します。

6　トランザクションの使い方

ここでは、BEA Tuxedo CORBA の ActiveX クライアント・アプリケーションでトランザクションを使用する方法を説明します。

TransactionCurrent オブジェクトの概要については、「第 1 章 概要」を参照してください。

トランザクションの概要

クライアント・アプリケーションは、トランザクション処理を使用してデータの有効性、一貫性、永続性を保証します。BEA Tuxedo CORBA のトランザクションを使用すると、クライアント・アプリケーションはトランザクションを開始および終了し、トランザクションのステータスを取得できます。BEA Tuxedo ソフトウェアでは、CORBA のオブジェクト・トランザクション・サービス で定義されたトランザクションを、使いやすいように拡張して使用します。

トランザクションは、インターフェイスで定義されます。アプリケーション設計者は、BEA Tuxedo クライアント / サーバ・アプリケーション内のどのインターフェイスでトランザクションを処理するかを指定します。C++ サーバ・アプリケーションの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) で、各インターフェイスのトランザクション方針が定義されます。通常、使用可能なインターフェイスに対する ICF ファイルまたはサーバ記述ファイルが、アプリケーション設計者からクライアント・プログラマに提供されます。

トランザクションの開発プロセスの概要

トランザクションは、以下の手順でクライアント・アプリケーションに追加します。

手順	説明
1	Bootstrap オブジェクトを使用し、指定した BEA Tuxedo ドメインで TransactionCurrent オブジェクトへのリファレンスを取得します。
2	TransactionCurrent オブジェクトのメソッドを使用して、トランザクション・オペレーションに CORBA オブジェクトのインターフェイスを含めます。

以下の節では、Transactions University サンプル・アプリケーションにあるクライアント・アプリケーションの一部を使用して、この手順を説明します。Transactions University サンプル・アプリケーションの詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo CORBA University サンプル・アプリケーション』を参照してください。Transactions University サンプル・アプリケーションは、BEA Tuxedo ソフトウェア・キットの次のディレクトリにあります。

```
drive:\tuxdir\samples\corba\university\transactions
```

ステップ 1: Bootstrap オブジェクトを使用して TransactionCurrent オブジェクトを取得する

Bootstrap オブジェクトを使用し、指定した BEA Tuxedo ドメインで TransactionCurrent オブジェクトへのオブジェクト・リファレンスを取得します。TransactionCurrent オブジェクトの詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo CORBA プログラミング・リファレンス』参照してください。

次の Visual Basic の例は、Bootstrap オブジェクトを使用して TransactionCurrent オブジェクトを返す方法を示します。

```
Set objTransactionCurrent =
             objBootstrap.CreateObject("Tobj.TransactionCurrent")
```

ステップ 2: TransactionCurrent メソッドを使用する

TransactionCurrent オブジェクトには、クライアント・アプリケーションでトランザクションを管理できるようにするメソッドがあります。これらのメソッドを使用すると、トランザクションを開始および終了して、現在のトランザクションに関する情報を取得できます。TransactionCurrent オブジェクトは、以下のメソッドを提供します。

- `begin()`

  新しいトランザクションを開始します。以降のオペレーションは、このトランザクションのスコープ内で発生します。クライアント・アプリケーションがトランザクションを開始したとき、デフォルトのトランザクション・タイムアウトは 300 秒です。`set_timeout` メソッドで、デフォルト値を変更できます。

- commit()

  トランザクションを正常に終了します。このクライアント・アプリケーションですべてのオペレーションが正常に終了したことを示します。

- rollback()

  トランザクションを強制的にロールバックします。

- rollback_only()

  可能なアクションのみがロールバックされるようにトランザクションをマークします。通常、このメソッドは、サーバ・アプリケーションでのみ使用されます。

- suspend()

  現在のトランザクションの参加を一時停止します。このメソッドは、トランザクションを示すオブジェクトを返し、クライアント・アプリケーションが後でトランザクションを再開できるようにします。

- resume()

  指定したトランザクションの参加を再開します。

- get_status()

  クライアント・アプリケーションでトランザクションのステータスを返します。

- get_transaction_name()

  トランザクションを示す出力可能な文字列を返します。

- set_timeout()

  トランザクションに関連付けられたタイムアウトを修正します。デフォルトのトランザクション・タイムアウト値は 300 秒です。begin() メソッドで明示的に開始するのではなく、トランザクションが自動的に開始した場合、タイムアウト値は、UBBCONFIG ファイルの TRANTIME パラメータで指定した値です。TRANTIME パラメータの設定に関する詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo アプリケーション実行時の管理』参照してください。

- get_control()

  トランザクションを表すコントロール・オブジェクトを返します。

基本的なトランザクションは、以下のように動作します。

1. クライアント・アプリケーションは、
   Tobj::TransactionCurrent::begin() メソッドを使用してトランザクションを開始します。このメソッドは値を返しません。
2. CORBA インターフェイスのオペレーションは、トランザクションのスコープ内で実行されます。これらのオペレーションの一部を呼び出して例外が発生した場合 (明示的に、または通信の失敗の結果として)、その例外がキャッチされ、トランザクションはロールバックされます。
3. Tobj::TransactionCurrent:commit() メソッドを使用すると、現在のトランザクションがコミットされます。このメソッドは、トランザクションを終了して、オペレーションの処理を開始します。トランザクションは、トランザクションのすべてのパーティシパントがコミットに同意した場合にのみコミットされます。

   トランザクションとクライアント・アプリケーションの関連付けは、アプリケーションが Tobj::TransactionCurrent:commit() メソッドまたは Tobj::TransactionCurrent:rollback() メソッドを呼び出したときに解除されます。次の Visual Basic の例は、クラスに登録している生徒のオペレーションをカプセル化するためにトランザクションを使用する場合を示しています。

```
' トランザクションを開始
'
objTransactionCurrent.begin
'
' コースの登録を試行
'
NotRegisteredList = objRegistrar.register_for_courses(mStudentID,
      CourseList, exception)
'
If exception.EX_majorCode = NO_EXCEPTION then
      ' 要求が成功したら、トランザクションをコミット
      '
      Dim report_heuristics As Boolean
      report_heuristics = True
      objTransactionCurrent.commit report_heuristics
Else
      ' 要求が失敗したら、トランザクションをロールバック
      '
      objTransactionCurrent.rollback
              MsgBox "Transaction Rolled Back"
End If
```

7　コマンド行オプション

ここでは、Application Builder のコマンド行バージョンについて説明します。

BEAAppBuilder コマンドは、Application Builder のコマンド行バージョンです。コマンドは、makefile、バッチ・コマンド・ファイル、または対話的にコマンド行で使用されます。このコマンドを使用する前に、UBBCONFIG の ISL パラメータが、サーバ・コンピュータのホストとポートに設定されていることを確認してください。

書式

BEAAppBuilder -v toAdpaterPath, sourcePath [,sourcePath...], -i directorypath, -t directorypath, -o directorypath

パラメータ

-v

CORBA インターフェイスに対する ActiveX バインディングを作成します。

toAdapterPath

バインディングの作成に使用するアダプタを指定します。現在のリリースの Application Builder では、toAdapterPath パスは OLEAutomation です。

sourcePath

バインディングを作成する 1 つまたは複数の CORBA インターフェイスを指定します。モジュールを指定することもできます。

-i *directorypath*

このコマンドで生成される C++ ヘッダ・ファイルのディレクトリを指定します。デフォルトのディレクトリは、\tuxdir\Include です。このオプションを指定しなかった場合、Application Builder は、最後に定義された値を使用します。

-t *directorypath*

このコマンドで生成されるタイプ・ライブラリのディレクトリを指定します。デフォルトのディレクトリは、\tuxdir\TypeLibraries です。このオプションを指定しなかった場合、Application Builder は、最後に定義された値を使用します。

`-o directorypath`

このコマンドで生成される MIDL/ODL ファイルのディレクトリを指定します。デフォルトのディレクトリは、`\tuxdir\TypeLibraries` です。このオプションを指定しなかった場合、Application Builder は、最後に定義された値を使用します。

例

次のコマンドは、`Registrar` インターフェイスおよび `RegistrarFactory` インターフェイスに対する ActiveX バインディングを作成します。

```
BEAAppBuilder -v OLEAutomation, Registrar, RegistarFactory, -i
c:\tuxdir\Include, -t c:\tuxdir\TypeLibraries
```

用語集

活性化

オブジェクト実行の準備をするプロセス。

活性化方針

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。

ActiveX

作成された言語に関係なく、ネットワーク環境でソフトウェア・コンポーネントどうしが対話できるようにする Microsoft の技術。ActiveX は、Component Object Model (COM) を基に構築されており、OLE Automation などの OLE 機能を含んでいます。

ActiveX ビュー

すべてのインターフェイスのインプリメンテーション、ActiveX でサポートされているデータ型へのデータ型のマッピングを含む、ActiveX 標準に準拠した CORBA オブジェクトの表現。

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)」を参照してください。

アプリケーション

BEA Tuxedo CORBA システムで、特定の作業を実行するために設計された単独のコンピュータ・プログラム。

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・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・レベルの関数および環境。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 アプリケーション・プログラムが、グローバル・トランザクションの開始とコミット、メッセージの送受信、修正の管理、型付きバッファの管理、および同様のタスクの実行に使用する UNIX 国際標準インターフェイス。ATMI インターフェイスは、すべての BEA Tuxedo ベースのシステムでサポートされており、X/Open TX および XATMI インターフェイスの基礎となります。

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」を参照してください。

属性

オブジェクトと値との間で識別可能な関連性。

認証する

ユーザまたはプロセッサの ID を信頼性の高い方法で判別すること。多くの場合、1 つまたは複数のパスワードを使用します。認証されると、サービスおよびオブジェクトの認証テーブルに ID をマッピングできます。このマッピングは通常、アクセス制御リストで行われます。

認証

アプリケーション・パスワードおよびセキュリティ・サービスで構成され、ユーザを検証し、アプリケーションに参加できるようにする方法。

BEA ActiveX Client

BEA Tuxedo CORBA のコンポーネント。BEA Tuxedo ドメインと ActiveX オブジェクト・システムの間の相互運用性を提供します。ActiveX Client は、ActiveX メソッドを、BEA Tuxedo 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。

BEA Tuxedo アプリケーション

単一のビジネス関数をサポートするための 1 つまたは複数の Tuxedo ドメイン。

BEA Tuxedo クライアント・アプリケーション

BEA Tuxedo CORBA で使用するために記述されたプログラム。他のアプリケーションからのサービスを要求します。

BEA Tuxedo CORBA サーバ・アプリケーション

BEA Tuxedo CORBA で使用するために記述されたプログラム。クライアント・アプリケーションによって要求されたタスクを実行します。

BEA Tuxedo CORBA TP フレームワーク

BEA Tuxedo 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。TP (トランザクション処理) 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。

a. 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。

b. 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。

c. 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。

d. ハウスキーピング機能を実行します。

BEA Tuxedo のドメイン

CORBA アプリケーションでは、1 つの UBBCONFIG ファイルで定義されたサーバ、サービス、およびする関連リソース・マネージャ。
ATMI アプリケーションでは、BEA Tuxedo システムの特定のインスタンス、顧客のサーバ・アプリケーション、および BEA Tuxedo ドメインをコンフィギュレーションするための単一の UBBCONFIG ファイル。

BEA Tuxedo 外部クライアント・アプリケーション

Netscape Navigator など、ORB にインプリメントされた BEA 社製品以外のクライアント・アプリケーション。BEA Tuxedo CORBA 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。ORB は Microsoft 製品でインプリメントされていますが、ORB 自体は BEA 社で提供されているためです。

BEA Tuxedo ネイティブ・クライアント・アプリケーション

BEA Tuxedo 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 BEA Tuxedo ドメインで実行中のマシン上にあるかどうかによって、要求がどのように処理されるか、および処理が透過的かどうかは異なります。BEA Tuxedo リモート・クライアント・アプリケーションは通常、BEA Tuxedo ドメインで実行中のマシン上にはありません。BEA Tuxedo CORBA 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。

BEA Tuxedo リモート・クライアント・アプリケーション

IIOP を使用してリモートの BEA Tuxedo 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 BEA Tuxedo ドメインで実行中のマシン上にあるかどうかによって、要求がどのように処理されるか、および処理が透過的かどうかは異なります。BEA Tuxedo CORBA 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。

BEA Tuxedo ソフトウェア

BEA 社から提供される BEA Tuxedo 製品。

BEA Tuxedo システム

BEA Tuxedo ソフトウェアが実行される BEA Tuxedo ソフトウェアおよびハードウェア。

バインディング

ActiveX オブジェクト・システムなど、CORBA オブジェクトのインターフェイスとほかのオブジェクト・システムの関連付け。

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。

ビジネス・オブジェクト

予測できない組み合わせに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、ドキュメント・プロセッサなど、単一のアプリケーションに依存しない、認識可能な通常のエンティティを表します。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。

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。

呼び出し

アプリケーション・プログラムがサービスの要求に使用する命令。

クラス

振る舞いと属性を表す変数とメソッドを格納したオブジェクトのテンプレート。クラスは、ほかのクラスからパブリックおよびプロテクト変数とメソッドを継承できます。

クライアント

(1) サーバにタスクの実行を要求するソフトウェア。クライアント / サーバという用語では、クライアント・アプリケーションは通常、ユーザ・インターフェイスを含んでおり、サーバ・アプリケーションは通常、

データを格納して操作します。クライアント / サーバ型のアーキテクチャでサービスを要求する側のソフトウェア・プログラムです。(2) BEA Tuxedo ソフトウェアで処理されるサービス要求を生成し、BEA Tuxedo ソフトウェアからの要求に対する応答を受け取るプロセス。

クライアント / サーバ

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・スタブ

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。このコードを使用して、要求をサーバ・アプリケーションに送信します。

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文字列の入力より、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

(1) 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。(2) 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、他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。

Component Object Model (COM)

Microsoft プラットフォームで使用するオブジェクト・モデル。COM は、多くの点で CORBA とは異なります。たとえば、オブジェクトの参照メカニズム、オブジェクトの作成プロセスなどで相違点があります。

COM ビュー

必要なインターフェイスのインプリメンテーションなど、Component Object Model (COM) 標準に準拠するオブジェクトの表現。

コンストラクタ

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。

会話型サーバ

サービスが要求を出した側との対話を実行するサーバ。

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

Common Object Request Broker Architecture の略。Object Management Group によって公開されている分散オブジェクト指向コンピューティング用のマルチベンダ標準です。

CORBA ファシリティ

OMG が採用した共通ファシリティ。共通ファシリティは、ほとんどのドメイ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。

CORBA インターフェイス

オペレーションと属性のセット。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。

CORBA オブジェクト

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。

CORBA ORB

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、通常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 ORB は CORBA ORB です。

コア・クラス

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 (またはインターフェイス)。Java コア・クラスは、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・システムで使用できるようにするためのものです。

デーモン

バックグラウンドで処理および実行されるシステム・プロセス。

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。

DBMS

「データベース管理システム (DBMS: Database Management System)」を参照してください。

デプロイメント・パッケージ

ActiveX Client では、タイプ・ライブラリ、Windows レジストリ・エントリ、およびクライアント・アプリケーションで CORBA オブジェクトの ActiveX ビューを使用するのに必要なアプリケーションを含む、自己登録型の OLE カスタム・コントロール実行可能ファイル。

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、オブジェクト指向アプリケーションの設計で役立つ法則と形式のことです。

デスクトップ・クライアント

Windows 2000 または Windows 98 などの Microsoft デスクトップ・プラットフォームで動作するクライアント・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。

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。

分散アプリケーション・フレームワーク

クライアント / サーバ・アプリケーションの構築と管理を行うミドルウェア。このフレームワークには、複数の運用環境にわたる接続を実現し、開発サービスと管理を提供する製品も含まれます。

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。

ドメイン

「BEA Tuxedo のドメイン」を参照してください。

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。Windows または OS/2 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。

環境オブジェクト

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。

イベント

1 つまたは複数のモジュールが関係する条件の発生、状態の変更、または一部の情報の有効化。

例外

データ・セットやファイルの処理、またはリソースの使用時に発生した入出力エラーなどの異常な事態。

ファクトリ

ほかの CORBA オブジェクトへのオブジェクト・リファレンスを返す CORBA オブジェクト。ファクトリは、サーバ・アプリケーションにあります。

ファクトリ・ファインダ

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。

フレームワーク

特定のアプリケーション・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。

ガベージ・コレクション

不要なメモリの自動検出および解放。Java ランタイム・システムは、プログラマが明示的にオブジェクトを解放しないようにガベージ・コレクションを実行します。

グローバル・トランザクション

(1) ローカル・トランザクションを含む 1 つまたは複数のリソース・マネージャにまたがるトランザクション。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トランザクション用のトランザクション・マネージャの名前。(2) 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。

GUI

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

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。

識別子

Java プログラムの項目名。

IDL

「OMG IDL」を参照してください。

IDL インターフェイス

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。

「Java インターフェイス」を参照してください。

IIOP

インターネット ORB 間プロトコル (Internet Inter-ORB Protocol)。Object Management Group (OMG) で指定するプロトコルです。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適切なオブジェクトに転送できます。

「CORBA ORB」を参照してください。

IIOP リスナ / ハンドラ

クライアント・アプリケーションと BEA Tuxedo ドメインとのやり取りを可能にする BEA Tuxedo CORBA 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。

インプリメンテーション・コード

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。

インプリメンテーション・ファイル

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・ロジックにインプリメントします。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。

継承

あるオブジェクトの機能および振る舞いを別のオブジェクトに渡すこと。オブジェクトが単一のインターフェイスから振る舞いを継承する場合は、単一継承と呼ばれます。オブジェクトが複数のインターフェイスから振る舞いを継承する場合は、多重継承と呼ばれます。

インスタンス

C++ のオブジェクト・インスタンス。オブジェクト・インスタンスは、BEA Tuxedo CORBA で CORBA オブジェクトのサーバントとして使用されます。

インターフェイス・リポジトリ

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。

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)

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。

Java

C++ をモデルとし、小さなサイズ、単純性、プラットフォームおよびオペレーティング・システム間の移植性を実現するように設計されたオブジェクト指向プログラミング言語。

Java Development Kit (JDK)

Java インタープリタ、Java クラス、Java 開発ツール (コンパイラ、デバッガ、逆アセンブラ、アプレットビューア、スタブ・ファイル・ジェネレータ、ドキュメンテーション・ジェネレータ) を含む Java 開発者用ソフトウェア・パッケージ。

Java インターフェイス

Java 言語で抽象インターフェイスを定義するための宣言。Java には多重継承がないため、Java クラスは、1 つまたは複数のインターフェイスをインプリメントして複合機能を提供できます。

「IDL インターフェイス」を参照してください。

Java Runtime Environment (JRE)

JRE を再配布するエンド・ユーザおよびプログラマ向けの Java Development Kit のサブセット。JRE は、Java 仮想マシン、Java コア・クラス、サポート用ファイルで構成されます。

Java 仮想マシン (Java Virtual Machine: JVM)

Java バイトコードを変換する Java Runtime Environment の一部。

JDK

「Java Developer's Kit」を参照してください。

レガシー・アプリケーション

BEA Tuxedo ドメインにアクセスするために、変更またはラッピングを必要とする既存のアプリケーション。

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、コンフィギュレーション・ファイルで論理名が指定された処理エレメント。

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

MIB で定義され、管理デバイスによって制御されるエンティティ (プロセス、ハードウェアの一部、システム・パフォーマンスなど)。

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

(1) BEA Tuxedo システムを構成するオブジェクト・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。(2) ASN.1 表記を使用した仮想ストレージ・データベース。MIB には、システム・マネージャ・ソフトウェアが監視および制御する各属性を表すオブジェク

トがあります。これらのオブジェクトは、ASN.1 表記で定義されます。各属性には、標準的な登録階層内での一意性を保証するオブジェクト識別子 (OID) があります。

マッピング

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。

メソッド

C++ または Java クラスのメソッド。ユーザが記述した C++ または Java クラスのメソッドは、BEA Tuxedo CORBA 分散オブジェクトの IDL オペレーションのインプリメンテーションを提供します。

MIB

「管理情報ベース (MIB: Management Information Base)」を参照してください。

MIB グループ

OID ツリーのオブジェクトの名前またはオブジェクト識別子で表すオブジェクトのグループ。管理対象オブジェクトのコレクションを含んでいます。

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。

モデル

何かを単純化した表現。表現は、詳細の一部が抽象化される形で単純化されます。

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。

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。

ネーミング・コンテキスト

一意な名前の関連性の集合を格納したオブジェクト。

オブジェクト

状態、振る舞い、および ID によって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。

「CORBA オブジェクト」を参照してください。

オブジェクト ID (OID)

指定したインターフェイスの分散オブジェクトを一意に識別する値。

オブジェクト・インプリメンテーション

インターフェイス向けに定義したオペレーションをインプリメントするために作成したコード。

オブジェクト・インターフェイス

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、引き出し、預け入れ、転送など、オブジェクトに対して実行可能なオペレーションのセットを識別します。

オブジェクト・モデル

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表現されたモデル。

オブジェクト・リファレンス

従業員の ID 番号のように、オブジェクト定義をオブジェクトのインスタンスに関連付ける識別子。

オブジェクト・システム

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 や COM など) に従ったオブジェクトの実装方法を指定します。

オクテット

8 ビットで構成される単位。バイト。

OLE

Object Linking and Embedding の略。あるアプリケーションから別のアプリケーションへのドキュメントの埋め込みから、もっと複雑な問題まで、ソフトウェア開発の問題に対処する Microsoft の技術。OLE を使用すると、ユーザに透過的な方法でクライアントおよびサーバのリンクが可能になります。

OLE オートメーション

ソフトウェア・パッケージが、固有の機能をスクリプト作成ツールおよびその他のアプリケーションにエクスポーズできるようにする技術。OLE オートメーションは、OLE Component Object Model (COM) を使用しますが、他の OLE 機能とは独立してインプリメントできます。

OMG IDL

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。

オペレーション

オブジェクトによって実行可能なアクション。

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

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 CORBA TP フレームワークは、すべてのサーバ・アプリケーションと POA の対話を自動的に処理します。

要求

実行するオペレーションを指定した、クライアント・アプリケーションから送信されるメッセージ。メッセージは、オブジェクト・リクエスト・ブローカ (ORB) に送信され、要求を満たす適切なサーバ・アプリケーションにリレーされます。

リソース・マネージャ

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。

ロールバック

(1) 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。(2) 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、またはやり直してトランザクションを終了するイベント。

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。

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

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。

サーバント

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。

サーバ

「BEA Tuxedo CORBA サーバ・アプリケーション」を参照してください。

サーバ・グループ

1 つのマシン上にあるサーバの集合。リソース・マネージャと関連付けられます。サーバ・グループは、サーバの起動、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。

サーバ・オブジェクト

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。

スケルトン

オブジェクト・インターフェイスに固有の BEA Tuxedo CORBA オブジェクト・リクエスト・ブローカ (ORB) コンポーネント。要求を特定のメソッドを渡す際にオブジェクト・アダプタをサポートします。スケルトンは、IDL コンパイラで作成されます。BEA Tuxedo ORB は、このスケルトンを実行時に使用し、要求を満たす特定のメソッドを呼び出します。

状態

オブジェクトの現在の状態に関する (通常はメモリ内の) 説明。

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。

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。

スレッド

実行の単位または実行コンテキスト。命令の実行シーケンス、および各命令が扱うメモリ。

3 層型のクライアント / サーバ

n 層型のクライアント / サーバのインプリメンテーション。

TM

「トランザクション・マネージャ (TM: Transaction Manager)」を参照してください。

トランザクション

(1) 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。(2) 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。

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。

TransactionCurrent

トランザクションを管理するためのオブジェクト。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。

トランザクション・マネージャ

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。

「トランザクション・コーディネータ」を参照してください。

トランザクション方針

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 フレームワークの対話を決定する方針。

TUXCONFIG

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。

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。

タイプ・ライブラリ

1 つのファイルで表される共有コードのリポジトリ。データ型とインターフェイス型を格納します。

UBBCONFIG

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。このファイルは、TUXCONFIG ファイルの ASCII 表現です。

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。

UserTransaction 環境オブジェクト

クライアント・アプリケーションを BEA Tuxedo CORBA 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。クライアント・アプリケーションは、その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。

ビュー

ActiveX など、別のオブジェクト・システムに存在する BEA Tuxedo ドメインの CORBA オブジェクトの表現。

「CORBA オブジェクト」と「BEA Tuxedo ドメイン」を参照してください。

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。

ラッパー

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 BEA Tuxedo 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。

XML

Extensible Markup Language の略。Sun Microsystems, Inc. によって設立された World Wide Web Consortium (W3C) が開発した言語で、SGML を World Wide Web に適用するためのものです。

索引

A

ActiveX 1-1
  概念
    バインディング 1-2
    ビュー 1-2
  命名規則 1-3
ActiveX Client
  概要 1-2
ActiveX クライアント・アプリケーション
  ISL パラメータ 2-8
  インターフェイスのインターフェイス・リポジトリへのロード 2-7
  インターフェイス・リポジトリのサーバ・アプリケーションの起動 2-8
  インターフェイス・リポジトリの使用 1-5
  オブジェクトに対する初期リファレンスの解決 2-12
  オブジェクトのオペレーションの呼び出し 2-13
  開発プロセス 2-2
  環境オブジェクトのインターフェイス・リポジトリへのロード 2-6
  記述 2-10
  作成
    バインディング 2-9
    ビュー 2-9
  セキュリティの使用 5-2
  セキュリティの定義 5-2
  ドメインとの通信の確立 2-11
  トランザクションを使用 6-2
  ビューのデプロイ 2-16
  ビューを使用 1-2
  ファクトリを使用 2-13
Application Builder
  ISL パラメータ 2-9
  ウィンドウ 2-4
  主なユーザー・タスク 1-2
  概要 1-1, 1-2
  作成
    タイプ・ライブラリ 2-9
    デプロイメント・パッケージ 2-16
    バインディング 2-9
    ビュー 2-9
  しくみ 1-2
  説明 1-2
  メイン・ウィンドウ 3-1
Application Builder の起動 4-2

B

Bootstrap オブジェクト
  SecurityCurrent オブジェクトの取得 5-2
  TransactionCurrent オブジェクトの取得 6-2
  初期リファレンスの解決
    Visual Basic 2-11
  説明 1-8
  宣言
    Visual Basic 2-11

C

C++ 5-5
  コード例
    PrincipalAuthenticator オブジェクト
      C++ 5-3
    SecurityCurrent オブジェクト 5-3

TransactionCurrent オブジェクト 6-3
ドメインへのログオン 5-5
トランザクション 6-5
C++ ヘッダ・ファイル
ディレクトリの場所 4-5
COM オブジェクト、インスタンスの作成 2-15
CORBA C++ クライアント・アプリケーション
インターフェイス・リポジトリの使用 1-5
セキュリティの使用 5-2
セキュリティの定義 5-2
トランザクションを使用 6-2
CORBA Java クライアント・アプリケーション
インターフェイス・リポジトリの使用 1-5
セキュリティの使用 5-2
セキュリティの定義 5-2
トランザクションを使用 6-2
CORBA インターフェイス
インターフェイス・リポジトリへのロード 2-7
バインディングの作成 2-9
CORBA インターフェイスのインターフェイス・リポジトリへのロード 4-1
CORBA サービス・セキュリティ・サービス 5-1
CORBA のオブジェクト・トランザクション・サービス 6-1

D

[Deployment Packages] ウィンドウ 3-8
[Display] ウィンドウ 3-8

E

[Edit] メニュー 3-7

F

FactoryFinder オブジェクト
コード例
Visual Basic 2-12
図 1-9
説明 1-9
宣言
Visual Basic 2-11
[File] メニュー 3-6

H

[Help] メニュー 3-10

I

ICF ファイル
トランザクション方針の定義 6-1
idl2ir コマンド
ActiveX クライアント・アプリケーションでの使用 2-3
インターフェイスのインターフェイス・リポジトリへのロード 2-7
インターフェイス・リポジトリへのデータの追加 1-5
オートメーション環境オブジェクトのインターフェイス・リポジトリへのロード 2-6
構文 2-7
説明 1-5
InterfaceRepository オブジェクト
説明 1-15
ir2idl コマンド
OMG IDL ファイルの作成 1-5
説明 1-5
irdel コマンド
インターフェイス・リポジトリからの CORBA インターフェイスの削除 1-5
説明 1-5
ISL パラメータ 2-8

ActiveX クライアント・アプリケーションでの使用 2-12
Application Builder での使用 2-9

J

Java
コード例
PrincipalAuthenticator オブジェクト
Java 5-3
SecurityCurrent オブジェクト 5-3

M

MIDL ファイル
ディレクトリの場所 4-5

O

ODL ファイル
ディレクトリの場所 4-5
OMG IDL
説明 1-4

P

PDF の場所
オンライン・ヘルプ 1-xii
PrincipalAuthenticator オブジェクト
クライアント・アプリケーションでの使用 5-3
コード例
C++ 5-3
Java 5-3
Visual Basic 5-3
ドメインへのログオン 5-4
認証レベルの取得 5-3
引数 5-4

S

SecurityCurrent オブジェクト
クライアント・アプリケーションでの使用 5-3
コード例
C++ 5-3
Java 5-3
Visual Basic 5-3
説明 1-12
プロパティ
Credentials 1-12
PrincipalAuthenticator 1-12
[Services] ウィンドウ 3-3
説明 3-1

T

TOBJ_APPAUTH
説明 1-12
必要な引数 5-4
TOBJ_NOAUTH
説明 1-12
必要な引数 5-4
TOBJ_SYSAUTH
説明 1-12
必要な引数 5-4
[Tools] メニュー 3-8
TransactionCurrent オブジェクト
トランザクション方針 1-13
メソッド 6-3

U

UBBCONFIG ファイル
インターフェイス・リポジトリのサーバ・アプリケーションの起動 2-8
定義
セキュリティ 5-1

V

[View] メニュー
オプション 3-8
Visual Basic 5-5

コード例
Bootstrap オブジェクト 2-11
FactoryFinder オブジェクト 2-12
PrincipalAuthenticator オブジェクト 5-3
SecurityCurrent オブジェクト 5-3
TransactionCurrent オブジェクト 6-3
オペレーションの呼び出し 2-13, 2-14
ドメインへのログオン 5-5
トランザクション 6-5
ファクトリ 2-13
宣言 2-11
Bootstrap オブジェクト 2-11
FactoryFinder オブジェクト 2-11
バインディングのタイプ・ライブラリのロード 2-10
Visual Basic サンプル
チャットルーム・サンプル 2-15

W

[Windows] メニュー 3-9
[Workstation Bindings] ウィンドウ 3-8, 4-5
[Workstation Views] ウィンドウ 3-3
説明 3-1

あ

アクセス
CORBA オブジェクト 1-2
アプリケーションのデプロイ 4-6

い

インターフェイス・リポジトリ
CORBA インターフェイスのロード 4-1
格納された情報 1-5
コマンド
idl2ir 1-5
ir2idl 1-5
irdel 1-5
サーバ・アプリケーションの起動 2-8
説明 1-5
ロード
オートメーション環境オブジェクト 2-6

う

ウィンドウ
[Services] 3-3
[Workstation Bindings] 4-5
[Workstation Views] 3-3

お

オートメーション環境オブジェクト
TOBJIN.IDL 2-6
インターフェイス・リポジトリへのロード 2-6
宣言の記述 2-11
オートメーション・サーバ、作成 2-14
オブジェクト
Application Builder GUI 上の 3-4
オプション
[Edit] メニュー 3-7
[File] メニュー 3-6
[Help] メニュー 3-10
[View] メニュー 3-8
[Window] メニュー 3-9
オンライン・ヘルプ
印刷 1-xii
ウィンドウ 1-viii
使い方 1-vii

か

開発コマンド
idl2ir 1-5
ir2idl 1-5
irdel 1-5
開発プロセス
ActiveX クライアント・アプリケー

ション 2-2
セキュリティ 5-2
トランザクション 6-2
カスタマ・サポートへのお問い合わせ情報 1-xiii
環境オブジェクト 1-7
Bootstrap 1-7
C++ 1-7
FactoryFinder 1-7
Java 1-7
SecurityCurrent 1-7
TransactionCurrent 1-7
インターフェイス・リポジトリ 1-7
オートメーション 1-7, 2-3
説明 1-7
関連情報 1-xiii

く

クライアント・アプリケーション
セキュリティの使用 5-1
トランザクションを使用 6-5

こ

コード例
BEA Tuxedo ドメインへのログオン 5-5
Visual Basic 2-11
Bootstrap オブジェクト
Visual Basic 2-11
FactoryFinder オブジェクト
Visual Basic 2-12
PrincipalAuthenticator オブジェクト
C++ 5-3
Java 5-3
Visual Basic 5-3
SecurityCurrent オブジェクト
C++ 5-3
Java 5-3
Visual Basic 5-3
TransactionCurrent オブジェクト
C++ 6-3
Visual Basic 6-3
Tuxedo ドメインへのログオン
C++ 5-4
Java 5-4
オペレーションの呼び出し
Visual Basic 2-13, 2-14
宣言
Visual Basic 2-11
ドメインへのログオン 5-5
トランザクション
C++ 6-5
Visual Basic 6-5
ファクトリ
Visual Basic 2-13

さ

作成
デプロイメント・パッケージ 4-6
サポート
テクニカル 1-xiv
サンプル・アプリケーション
Security 5-2
トランザクション 6-2

せ

製品マニュアルを印刷する 1-xii
セキュリティ
PrincipalAuthenticator オブジェクトの取得 5-3
SecurityCurrent オブジェクトの取得 5-2
概要 5-1
サポートされる認証レベル 1-12
設定する 5-1
ドメインからのログオフ 5-5
ドメインへのログオン 5-4
認証レベルの取得 5-3
説明 1-1

た

タイプ・ライブラリ
- Application Builder での作成 2-9
- 開発ツールへのバインディングのロード 2-10
- ディレクトリの場所 2-9, 2-10, 4-5
- 命名規則 2-9

タスク
- Application Builder の起動 4-2
- CORBA インターフェイスのインターフェイス・リポジトリへのロード 4-1
- 作成
  - CORBA オブジェクトの ActiveX ビュー 4-3
- デフォルト・ディレクトリの変更 4-8
- デプロイメント・パッケージの作成 4-6

つ

ツールバー 3-11

て

ディレクトリの場所
- タイプ・ライブラリ 2-9
- デプロイメント・パッケージ 2-16

デフォルト・ディレクトリの変更 4-8

デプロイメント・パッケージ
- 説明 2-16
- ディレクトリの場所 2-16

と

ドメイン
- PrincipalAuthenticator オブジェクトを使用したログオン 5-4
- 図 1-6
- セキュリティの定義 5-1
- 説明 1-6
- ツウシンノカクリツ
  - ActiveX クライアント・アプリケーション 2-11
- 認証レベル 5-3
- ログオフ 5-5

ドメインとの関連 1-7

トランザクション
- TransactionCurrent オブジェクトの取得 6-2
- 概要 6-1
- クライアント・アプリケーションでの 6-5

トランザクションの方針
- ICF ファイルでの定義 6-1

トランザクション方針
- 説明 1-13

に

認証レベル
- BEA Tuxedo ソフトウェアでのサポート 1-12
- TOBJ_APPAUTH 1-12
- TOBJ_NOAUTH 1-12
- TOBJ_SYSAUTH 1-12
- クライアント・アプリケーションでの 5-3
- 取得
  - C++ 5-3
  - Java 5-3
  - Visual Basic 5-3

は

バインディング
- 作成 2-9
- 説明 1-2
- デプロイする 2-16

ひ

ビュー
- オペレーションの呼び出し 2-13, 2-14
- 作成 2-9

説明 1-2
宣言の記述 2-11
定義 1-2
デプロイする 2-16

ふ

ファクトリ
CORBA オブジェクトの作成 1-9
コード例
Visual Basic 2-13
説明 1-9
宣言
Visual Basic 2-11
宣言の記述 2-11
命名規則 1-11

ほ

ボタン
ツールバー 3-11

ま

マニュアルの場所 1-xii

め

命名規則
ActiveX 1-3
ファクトリ 1-11
メイン・ウィンドウ
[Services] ウィンドウ 3-1
[Workstation Views] ウィンドウ 3-1
メソッド
TransactionCurrent オブジェクト 6-3
メニュー・オプションの説明 3-6