BEA Tuxedo

BEA Tuxedo の相互運用性

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

BEA Tuxedo の相互運用性

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

目次

2. BEA Tuxedo CORBA クライアントと WebLogic Server EJB の接続

3. WebLogic Server から CORBA オブジェクトへの接続

索引

このマニュアルについて

このマニュアルでは、BEA Tuxedo とほかの BEA ソフトウェア・コンポーネントおよびサード・パーティ・ソフトウェア・プロバイダとの間の相互運用性について概説します。このマニュアルでは、相互運用性のソリューションを示し、いくつかのソリューションについて概説し、参考となる他のマニュアルおよびサンプル・アプリケーションを示します。

このマニュアルでは、以下の内容について説明します。

- 「第 1 章 はじめに」では、BEA Tuxedo システムにおける WebLogic Server J2EE と BEA Tuxedo プログラミング・モデルとの間の相互運用性および共存の機能についての概要を説明します。またこの章では、特定のコンポーネント間の相互運用性のソリューションについても説明します。
- 「第 2 章 BEA Tuxedo CORBA クライアントと WebLogic Server EJB の接続」では、Wlstrader Value Type サンプル・プログラムの詳細について説明し、このサンプル・アプリケーションの構築および実行プロセスの手順について説明します。
- 「第 3 章 WebLogic Server から CORBA オブジェクトへの接続」では、WebLogic Enterprise Connectivity (WLEC) EJB simpapp サンプル・アプリケーションについて補足説明します。

対象読者

このマニュアルは、BEA 製品スイートのコンポーネント間で相互運用できる、安全でスケーラブルなトランザクション・ベースのサーバ・アプリケーションを作成するアプリケーション開発者を対象としています。CORBA、Enterprise JavaBeans、および C++ と Java プログラミング言語に読者が精通していることを前提として書かれています。また、このマニュアルは、ほかの BEA 相互運用性のソリューションを探し出すためのリソースでもあります。

e-docs Web サイト

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

マニュアルの印刷方法

このマニュアルは、ご使用の Web ブラウザで一度に 1 ファイルずつ印刷できます。Web ブラウザの [ファイル] メニューにある [印刷] オプションを使用してください。

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

Adobe Acrobat Reader をお持ちでない場合は、Adobe 社の Web サイト (http://www.adobe.co.jp/) から無償でダウンロードできます。

関連情報

CORBA、BEA Tuxedo、分散オブジェクト・コンピューティング、トランザクション処理、C++ プログラミング、および Java プログラミングの詳細については、BEA Tuxedo オンライン・マニュアルの「Bibliography」を参照してください。

サポート情報

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

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

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

カスタマ・サポートへお問い合わせの際には、以下の情報をご用意ください。

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

表記上の規則

このマニュアルでは、以下の表記規則が使用されています。

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

規則	項目

等幅テキスト	コード・サンプル、コマンドとオプション、データ構造とメンバ、データ型、ディレクトリ、およびファイル名と拡張子を示します。また、キーボードから入力する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`*`]...`

規則	項目
|	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。
. . .	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　はじめに

ここでは、次の内容について説明します。

- WLS J2EE プログラミング・モデルと BEA Tuxedo プログラミング・モデルの相互運用性
- 相互運用性ソリューション
- 相互運用性のサンプル・アプリケーション

WLS J2EE プログラミング・モデルと BEA Tuxedo プログラミング・モデルの相互運用性

この節では、WebLogic Server J2EE プログラミング・モデルと Tuxedo プログラミング・モデルの間の BEA Tuxedo システムの相互運用性機能と共存機能について説明します。主要な相互運用性機能を、次のような分類で紹介します。

- BEA Tuxedo サーバの相互運用性
- BEA Tuxedo のクライアントとサーバの相互運用性
- BEA WebLogic Server と BEA Tuxedo の相互運用性
- BEA Tuxedo のサード・パーティ ORB との相互運用性
- BEA Tuxedo ドメイン間の相互運用性

次に、BEA クライアントおよびサーバを簡単に説明します。

以下の定義を覚えておいてください。

- **BEA クライアント**

  BEA クライアントは、次のエンティティのいずれかです。BEA ドメインの外側に位置し、ドメインへのゲートウェイとしてリスナ / ハンドラを使用します。

  - Jolt クライアント・アプリケーション (Jolt リスナ / ハンドラを使用)
  - BEA Tuxedo /WS クライアント・アプリケーション (Tuxedo /WS リスナ / ハンドラを使用)
  - BEA Tuxedo CORBA クライアント・アプリケーション (IIOP リスナ / ハンドラを使用)
  - ActiveX クライアント・アプリケーション (IIOP リスナ / ハンドラを使用)
  - RMI クライアント・アプリケーション (IIOP リスナ / ハンドラを使用)
  - WebLogic Enterprise Connectivity (WLEC) を使用した BEA WebLogic Server 上の EJB

  別の BEA Tuxedo クライアントを呼び出す BEA Tuxedo クライアントはサポートされていません。

- **BEA Tuxedo サーバ**

  BEA Tuxedo サーバとしては、BEA Tuxedo システムで動作する Tuxedo サービスと CORBA オブジェクトがあります。それらのサーバは、BEA Tuxedo ドメインの管理単位の範囲内で実行され、`UBBCONFIG` ファイルを使用してコンフィギュレーションされます。

BEA Tuxedo サーバの相互運用性

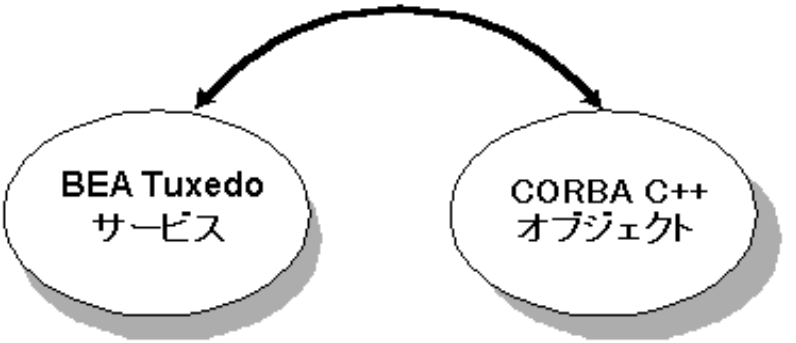
この節では、次の BEA Tuxedo サーバ・コンポーネント間の相互運用性について説明します。

- BEA Tuxedo ATMI サービス
- CORBA C++ オブジェクト

図 1-1 は、BEA Tuxedo サーバ・アプリケーション間の直接的な相互運用性のサポートを示しています。

BEA Tuxedo サービスでは、コンパイル済みの C++ クライアント・スタブ・ファイルを使用して CORBA C++ オブジェクトを呼び出すことができます。そのための 1 つの方法は、C++ オブジェクトのクライアント・スタブ・ファイルを呼び出す C-callable C++ 関数として BEA Tuxedo サービスをインプリメントすることです。この方法を利用する場合は、Tuxedo サービスをビルドするときに C++ ORB ライブラリにリンクする必要があります。

図 1-1　BEA Tuxedo サーバの相互運用性



C++ オブジェクトには、BEA Tuxedo サービスの ATMI 呼び出しを含めることができます。この機能の例については、『BEA Tuxedo CORBA University サンプル・アプリケーション』の Wrapper University サンプル・アプリケーションを参照してください。

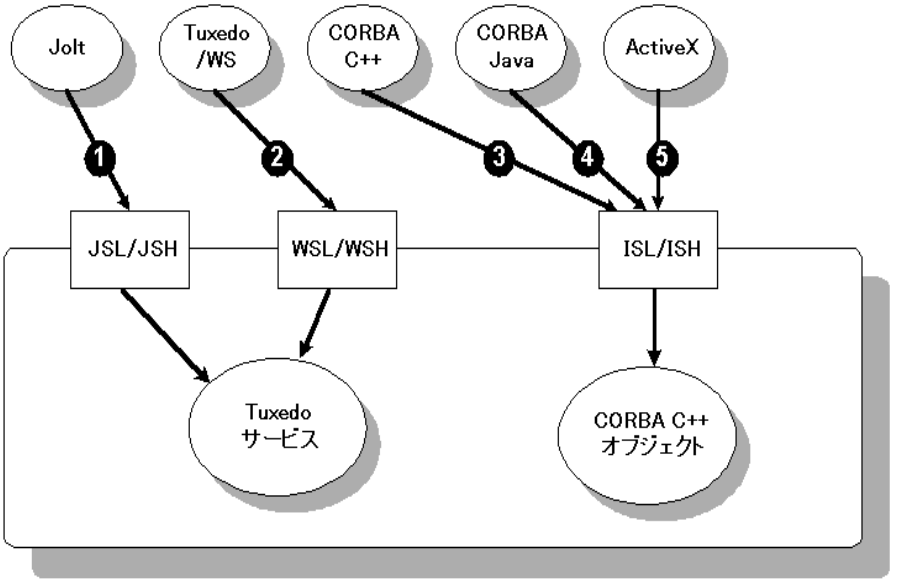
トランザクションとセキュリティ

BEA Tuxedo ドメインで動作するサーバ・アプリケーション間では、トランザクション・コンテキストおよびセキュリティ・コンテキストの伝達が完全にサポートされています。

BEA Tuxedo のクライアントとサーバの相互運用性

図 1-2 は、BEA サーバを呼び出す BEA クライアント間の相互運用性のサポートを示しています。

図 1-2　リモート・クライアントとサーバの相互運用性



上の図では、以下のことが説明されています。

1. BEA Tuxedo サービスを呼び出す Jolt クライアント・アプリケーション

   Jolt クライアントでは、BEA Tuxedo ドメインで動作している BEA Tuxedo サービスを Jolt リスナ / ハンドラを使用して呼び出すことができます。Jolt の詳細については、BEA Tuxedo オンライン・マニュアルの『BEA Jolt』および『BEA Jolt API リファレンス』を参照してください。

2. BEA Tuxedo サービスを呼び出す BEA Tuxedo /WS クライアント・アプリケーション

   BEA Tuxedo /WS クライアント・アプリケーションでは、BEA Tuxedo ドメインで動作する BEA Tuxedo サービスをワークステーション・リスナ / ハンドラを使用して呼び出すことができます。

3. CORBA オブジェクトを呼び出す BEA CORBA C++ クライアント・アプリケーション

   BEA CORBA C++ クライアント・アプリケーションでは、CORBA C++ オブジェクトを呼び出すことができます。詳細については、『BEA Tuxedo CORBA クライアント・アプリケーションの開発方法』を参照してください。

4. CORBA オブジェクトを呼び出す BEA CORBA Java クライアント・アプリケーション

   BEA CORBA Java クライアント・アプリケーションでは、BEA Tuxedo ドメインで動作する CORBA C++ オブジェクトを IIOP リスナ / ハンドラを使用して呼び出すことができます。詳細については、『BEA Tuxedo CORBA クライアント・アプリケーションの開発方法』を参照してください。

5. CORBA オブジェクトを呼び出す BEA ActiveX クライアント・アプリケーション

   BEA ActiveX クライアント・アプリケーションでは、BEA Tuxedo ドメインで動作する CORBA C++ オブジェクトを IIOP リスナ / ハンドラを使用して呼び出すことができます。詳細については、『BEA Tuxedo CORBA クライアント・アプリケーションの開発方法』を参照してください。

プロキシ・オブジェクトまたはプロキシ・サーバを使用する BEA Tuxedo 環境では、以下の呼び出しもサポートされています。

- BEA Tuxedo サービスを呼び出す BEA CORBA C++ クライアント・アプリケーション

  中間に位置する C++ サーバ側オブジェクトを使用して BEA Tuxedo サービスと 1 対 1 でマッピングされるオペレーションのセットを備える C++ クライアントを作成できます。この機能の例については、『BEA Tuxedo CORBA University サンプル・アプリケーション』の Wrapper University サンプル・アプリケーションを参照してください。

- CORBA C++ オブジェクトを呼び出す BEA Tuxedo/WS クライアント・アプリケーション

  相互運用性は、BEA Tuxedo サービス・ラッパーを使用することで実現されます。BEA Tuxedo サービス・ラッパーは、BEA Tuxedo ドメインで動作し、CORBA C++ オブジェクトの呼び出しを行う CORBA C++ オブジェクトとして作成します。

トランザクションとセキュリティ

BEA のクライアント・アプリケーションとサーバ・アプリケーションの間ではトランザクション・コンテキストとセキュリティ・コンテキストの伝達が完全にサポートされていますが、以下の制限があります。

- BEA クライアント・アプリケーションでは、トランザクションの境界を判定できますが (つまり、トランザクションを明示的に開始、一時停止、再開、およびコミットできる)、トランザクションに参加することはできません。

  たとえば、クライアントではトランザクションを開始して、ドメイン内のサービスとオブジェクトを複数回呼び出すことができ、それらのサービスとオブジェクトではさらにほかのサービスとオブジェクトを呼び出すことができます。そのクライアント・アプリケーションは、開始したトランザクションのスコープの中でローカルの操作を実行し、それらの操作をトランザクションに加えることはできません。つまり、クライアント・アプリケーションでトランザクションが開始され、ドメイン内のオブジェクトが呼び出されて、データがローカル・データベースに書き込まれる場合、そのローカル・データベース操作はトランザクションに加えることができません。

BEA Jolt について

BEA Jolt には、BEA Tuxedo ドメインに存在する BEA Tuxedo サービスの ATMI 呼び出しを Java クライアントで行えるようにするメカニズムがあります。また、BEA WebLogic Server で BEA Tuxedo サービスを呼び出せるようにするメカニズムも提供されます。後者の機能は、Jolt 接続プールを通じて実現されます (この機能は次節で説明)。

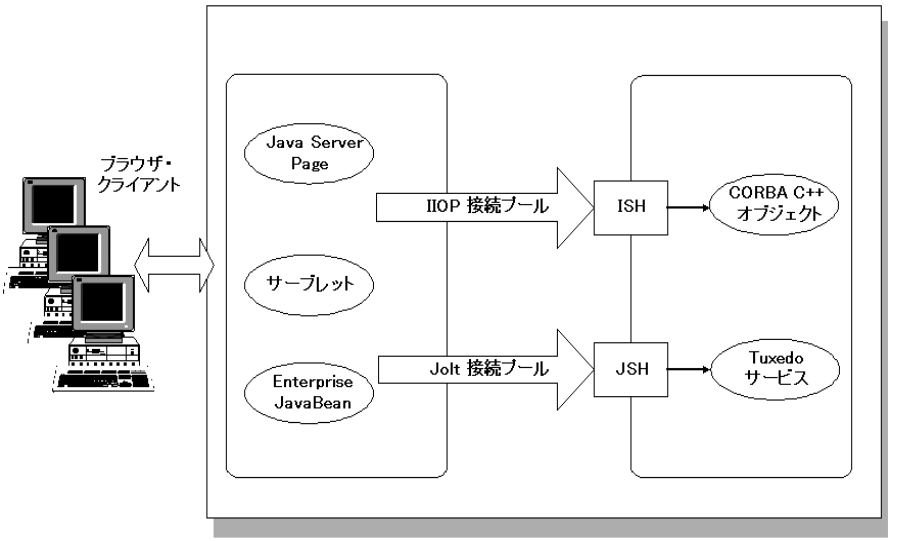
詳細については、次のマニュアルを参照してください。

- Jolt については、『BEA Jolt』を参照してください。
- WebLogic Server を BEA Tuxedo に接続するように Jolt 接続プールを設定する方法については、『BEA WebLogic Server での BEA Jolt の使用』を参照してください。

BEA WebLogic Server と BEA Tuxedo の相互運用性

WebLogic Server アプリケーションでは、BEA Tuxedo ドメイン内の CORBA オブジェクトおよび BEA Tuxedo サービスを呼び出すことができます。図 1-3 は、それらのインプリメンテーションでの接続プールの使い方を示しています。

図 1-3　WebLogic Server と BEA Tuxedo の相互運用性



接続プールについての情報を以下に示します。

- IIOP 接続プールを利用すると、BEA WebLogic Server アプリケーションで BEA Tuxedo ドメイン内の CORBA オブジェクトを呼び出すことができます。IIOP 接続プールの設定と使い方については、WebLogic Server のマニュアルの WebLogic Enterprise Connectivity (WLEC) に関するトピックを参照してください。
- Jolt 接続プールを利用すると、BEA WebLogic Server アプリケーションで BEA Tuxedo ドメイン内の BEA Tuxedo サービスを呼び出すことができます。Jolt 接続プールの設定と使い方については、『BEA WebLogic Server での BEA Jolt の使用』を参照してください。

BEA Tuxedo のサード・パーティ ORB との相互運用性

このリリースの BEA Tuxedo では、サード・パーティ ORB アプリケーションとの相互運用性を高めるために CORBA サービス・インターオペラブル・ネーミング・サービス (INS) がサポートされています。INS の追加により、INS を利用

するサード・パーティ ORB が BEA Tuxedo CORBA サーバ ORB と相互運用できます。INS を使用することで、サード・パーティ ORB は、BEA Bootstrap、SecurityCurrent、または TransactionCurrent 環境オブジェクトを使用せずに BEA Tuxedo CORBA サーバで以下の処理を実行できます。

- ブートストラップ処理
- 認証
- トランザクションの開始

注記　BEA Tuxedo CORBA クライアントの環境オブジェクトは、BEA WebLogic Enterprise 5.1 の場合と同じように BEA Tuxedo 8.0 でもサポートされます。

BEA Tuxedo ドメイン間の相互運用性

BEA Tuxedo ドメインで動作するサーバ・アプリケーションは、ドメイン・ゲートウェイを通じて別の BEA Tuxedo ドメインのサーバ・アプリケーションと相互運用できます。BEA Tuxedo ドメイン間の相互運用性の詳細については、BEA Tuxedo オンライン・マニュアルの『製品の概要』および『BEA Tuxedo アプリケーション実行時の管理』を参照してください。

相互運用性ソリューション

この節では、次のコンポーネント間の相互運用性ソリューションについて説明します。

- BEA WebLogic Server から BEA Tuxedo ATMI への接続
- BEA Tuxedo CORBA から BEA WebLogic Server への接続
- BEA WebLogic Server から BEA Tuxedo CORBA への接続

BEA WebLogic Server から BEA Tuxedo ATMI への接続

この節では、WebLogic Server と BEA Tuxedo ATMI の接続を実現するオプションについて説明します。この節の内容は次のとおりです。

- BEA Jolt の使用
- WebLogic Tuxedo コネクタ (WTC) バージョン 1.0 の使用

BEA Jolt の使用

BEA Jolt for BEA WebLogic Server を利用すると、WebLogic Server をフロント・エンドの HTTP サーバおよびアプリケーション・サーバとして使用して BEA Tuxedo サービスを Web 上で提供できます。

BEA Jolt は、Tuxedo サーバで動作する Jolt サービス・リスナ (JSL) を使用して BEA Tuxedo サービスへの要求を管理する Java ベースのクライアント API です。Jolt API は、WebLogic Server API に組み込まれ、サーブレットや JHTML などの BEA WebLogic アプリケーションからアクセスできます。

この製品は、Jolt クラス・ライブラリと Jolt リポジトリという 2 つの主なコンポーネントから構成されています。BEA Jolt を使用すると、クライアントとサーバ間に、インターネットを介した安全でスケーラブルなトランザクションを構築できます。

WebLogic Server との BEA Jolt の使い方については、BEA Tuxedo オンライン・マニュアルの『BEA WebLogic Server での BEA Jolt の使用』を参照してください。この文書では、BEA Jolt for WebLogic Server の操作について説明するとともに、BEA Jolt、BEA Tuxedo ATMI、および WebLogic Server を使用、コンフィギュレーション、および統合する方法についても説明します。この文書では、次のサンプル・プログラムが提供されます。

- 付録 B、単純なサーブレットの例

  この例では、BEA Jolt を使用して WebLogic サーブレットから BEA Tuxedo ATMI に接続する方法が示されます。

- 付録 C、Enterprise JavaBean を伴うサーブレットの例

この例では、BEA Tuxedo ATMI にアクセスするための EJB インターフェイスが示されます。

注記　現行リリースの BEA Jolt では、双方向の接続またはトランザクション・コンテキストの伝達がサポートされていません。

現行リリースの BEA Jolt は、以下の機能をサポートするように拡張されています。

- XML バッファ・タイプ

  データ依存型ルーティング (DDR) を利用して Jolt クライアントから BEA Tuxedo ATMI サービスに XML (eXtensible Markup Language) バッファを送信できます。Jolt クライアントでは、BEA Tuxedo ATMI サービスから XML 文書を受信できます。

- WebLogic Server で開始された BEA Tuxedo ATMI サービス要求の透過的なエンド・ツー・エンドのユーザ認証

  WebLogic Server で認証されたユーザ・クリデンシャルは適切なセキュリティ・インターフェイスまたはプロトコルにマッピングされ、受信した要求は BEA Tuxedo ATMI サービスを呼び出す前に再び認証する必要がありません。

- Jolt 接続プール (接続プールで障害が発生したときの接続プールのリセットをサポート)

  WebLogic Server/BEA Tuxedo 環境では、この機能によって、接続プールの再起動が必要な場合に BEA WebLogic Server を再起動する必要がなくなります。

WebLogic Tuxedo コネクタ (WTC) バージョン 1.0 の使用

WebLogic Tuxedo コネクタ バージョン 1.0 は、WebLogic Server と BEA Tuxedo ATMI の双方向の相互運用性を実現します。その結果、ATMI API を利用する BEA Tuxedo アプリケーションで JMS の高度なメッセージング機能を利用できます。

BEA Tuxedo サービスは、WebLogic Tuxedo コネクタを通じて ATMI を使用して呼び出しを行います。このコネクタを使用した Java プログラムとの通信は、BEA Tuxedo ATMI クライアントにとって透過的に行われます。WebLogic Tuxedo コネクタでは、BEA Tuxedo ATMI バッファの内容が抽出され、その内容が Java アプリケーションに提示されます。

WebLogic Tuxedo コネクタでは、ATMI サービスに変更を加えることなく WLS サーブレット、JSP、および EJB から BEA Tuxedo ATMI サービスを呼び出せるようにする Java API も提供されます。

WebLogic Tuxedo コネクタ 1.0 の機能と操作については、オンライン・マニュアルの http://e-docs.bea.com/wtc/wtc10/index.html を参照してください。このリンク先には、次の特定のトピックに関する WebLogic Tuxedo コネクタ 1.0 のマニュアルがあります。

- 管理ガイド： http://e-docs.bea.com/wtc/wtc10/admin
- ATMI ガイド : http://e-docs.bea.com/wtc/wtc10/atmi

BEA Tuxedo CORBA から BEA WebLogic Server への接続

この節では、BEA Tuxedo CORBA サーバと BEA WebLogic Server の接続を実現するオプションについて説明します。この節の内容は次のとおりです。

- RMI/IIOP および IDL インターフェイスの使用
- WebLogic Tuxedo コネクタ (WTC) バージョン 1.1 の使用

RMI/IIOP および IDL インターフェイスの使用

BEA Tuxedo CORBA C++ クライアント、およびクライアントとして機能する CORBA C++ サーバでは、オブジェクトを値で渡すという CORBA 規格がサポートされます。このサポートにより、CORBA クライアントで、WebLogic Server の EJB に対する IDL インターフェイスの RMI over IIOP 呼び出しが可能となります。

BEA Tuxedo CORBA Wlstrader Value Type クライアント・アプリケーションでは、RMI over IIOP と IDL インターフェイスを利用して WebLogic Server の Trader EJB に接続します。このサンプル・アプリケーションをビルドおよび実行する方法については、「第 2 章 BEA Tuxedo CORBA クライアントと WebLogic Server EJB の接続」を参照してください。

WebLogic Tuxedo コネクタ (WTC) バージョン 1.1 の使用

WebLogic Tuxedo コネクタ バージョン 1.1 は、BEA WebLogic Server と BEA Tuxedo コンポーネントの双方向の相互運用性を拡張します。このゲートウェイでは、BEA WebLogic Server と BEA Tuxedo CORBA および BEA Tuxedo ATMI の間の相互運用性が実現されます。この製品では、BEA Jolt および WebLogic Enterprise Connectivity と同等の機能が提供されます。ただし、この製品では Jolt とは違って透過的な移行が提供されません。

このバージョンの WebLogic Tuxedo コネクタを使用すると、WebLogic Server のプログラマは標準の CORBA API を使用して CORBA C++ オブジェクトを呼び出すことができます。WebLogic Tuxedo コネクタでは、BEA Tuxedo ドメイン・ゲートウェイを通じて適切な CORBA C++ オブジェクトに CORBA 呼び出しをルーティングできます。WebLogic Tuxedo コネクタでは、BEA WebLogic Server から BEA Tuxedo へのセキュリティとトランザクションの伝達がサポートされています。

WebLogic Tuxedo コネクタ バージョン 1.1 では、バージョン 1.0 にあったすべての機能がサポートされます。

BEA 社では、WebLogic Server と BEA Tuxedo の接続に WebLogic Tuxedo コネクタ バージョン 1.1 を使用することをお勧めします。

BEA WebLogic Server から BEA Tuxedo CORBA への接続

この節では、BEA WebLogic Server と BEA Tuxedo CORBA の接続を実現するオプションについて説明します。この節の内容は次のとおりです。

WebLogic Enterprise Connectivity (WLEC) の使用

WebLogic Enterprise Connectivity (WLEC) は、WebLogic Server のコンポーネントです。このコンポーネントを使用すると、CORBA や EJB などの RMI オブジェクトを呼び出すために、WebLogic Server クライアント (サーブレット、EJB、JSP、および RMI オブジェクト) から IIOP 接続プールを使用できます。

WebLogic Enterprise Connectivity の主な機能は以下のとおりです。

- BEA Tuxedo システムへの IIOP 接続のプール
- 1 つの WebLogic Server プロセスからの複数のアクティブな BEA Tuxedo CORBA クライアント・トランザクション
- `weblogic.properties` ファイルを使用した IIOP 接続プールのコンフィギュレーション
- WebLogic Console を使用した IIOP 接続プールの監視
- セキュア・ソケット・レイヤ (SSL) のサポート
- WebLogic Server から BEA Tuxedo CORBA へのセキュリティ・コンテキストの伝達
- 実行時のプールの再初期化

注記 WebLogic Enterprise Connector は、CORBA 2.2 に準拠しています。WebLogic Enterprise Connectivity では、値型でのオブジェクトの値渡しはサポートされていません。

WebLogic Enterprise Connectivity の詳細については、BEA WebLogic Server オンライン・マニュアルの WebLogic Enterprise Connectivity (WLEC) のトピックを参照してください。

BEA WebLogic Server のインストール先に格納されているマニュアルの例では、WebLogic Enterprise Connectivity を使用して BEA WebLogic Server 上のサーブレット、JSP、および Enterprise JavaBeans から BEA WebLogic Enterprise または BEA Tuxedo CORBA オブジェクトにアクセスする方法が紹介されています。そのファイルは、*wlserver6*\samples\examples\wlec\package-summary.html にあります。

BEA Tuxedo CORBA オブジェクトに接続する WebLogic Server WLEC EJB simpapp アプリケーションのビルドと実行の補足情報については、「WebLogic Server から CORBA オブジェクトへの接続」を参照してください。

WebLogic Tuxedo コネクタ (WTC) バージョン 1.1 の使用

WebLogic Tuxedo コネクタ バージョン 1.1 については、第 1 章の 14 ページ「WebLogic Tuxedo コネクタ (WTC) バージョン 1.1 の使用」を参照してください。

相互運用性のサンプル・アプリケーション

この節では、BEA Tuxedo および BEA WebLogic Server で提供される相互運用性のサンプル・アプリケーションを紹介します。それらのサンプル・アプリケーションでは、クライアントおよびサーバのプログラマに対して、アプリケーションで Enterprise JavaBeans (EJB) と CORBA オブジェクトを結合する基本的な概念が説明されます。

BEA Tuxedo と WebLogic Server では、表 1-1 で示されているサンプル・アプリケーションが提供されます。

表 1-1 相互運用性のサンプル・アプリケーション

アプリケーション	説明
Wlstrader Value Type サンプル・アプリケーション	RMI over IIOP を使用し、WebLogic Server の EJB にオブジェクトを値で渡す CORBA C++ クライアント・アプリケーションを例示します。
WLEC EJB simpapp サンプル・アプリケーション	WebLogic Enterprise Connectivity (WLEC) を使用して WebLogic Server 上の状態を持たない EJB から BEA Tuxedo CORBA オブジェクトにアクセスする方法を例示します。

2 BEA Tuxedo CORBA クライアントと WebLogic Server EJB の接続

ここでは、次の内容について説明します。

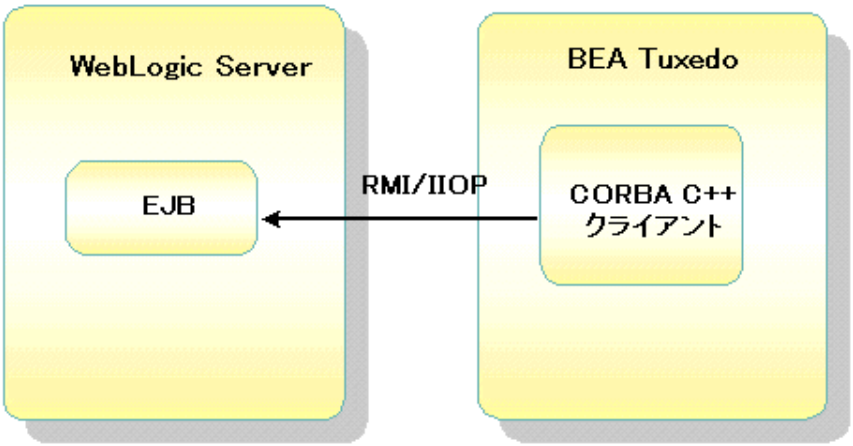
- Wlstrader Value Type サンプル・アプリケーションの概要
- Wlstrader Value Type サンプル・アプリケーションのコンポーネント
- Wlstrader Value Type サンプル・アプリケーションのビルドと実行
- リファレンス情報

Wlstrader Value Type サンプル・アプリケーションの概要

Wlstrader Value Type サンプル・アプリケーションは、RMI over IIOP クライアントと EJB の間の接続を例示します。特に、BEA Tuxedo で開発された CORBA C++ クライアント・アプリケーションがどのようにして WebLogic Server の EJB とやり取りできるのかが示されます。アプリケーションでは、WebLogic Server の例に含まれている Trader EJB が利用されます。Wlstrader Value Type のサンプルでは、Trader サンプル・アプリケーションのコンポーネントを利用すると同時に、独自のビルドと実行の手続きを行います。

図 2-1 は、Wlstrader Value Type サンプル・アプリケーションで実現される接続を説明しています。

図 2-1　CORBA C++ クライアントから WebLogic Server EJB



Wlstrader Value Type の例で、BEA Tuxedo CORBA C++ クライアントは値型を利用し、OBV (Object-by-Value) を渡します。これで、IDL インターフェイスの RMI over IIOP 呼び出しがサポートされ、WebLogic Server TraderBean Enterprise JavaBean (EJB) への接続が実現されます。

値型は、値をシステム間で移動できるクラスを表します。値型は、リモート・メソッドの引数または結果として、あるいはリモートで渡されるほかのオブジェクトのフィールドとして渡すことができます。BEA Tuxedo CORBA での値型のサポートについては、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』の「第 13 章 OMG IDL 文の C++ へのマッピング」を参照してください。

CORBA クライアントでは、WebLogic Server の JNDI インプリメンテーションで提供される COSNaming サービス層を呼び出して WebLogic Server インターオペラブル・オブジェクト・リファレンス (IOR) を取得します。CORBA ネーム・サービスおよび COSNaming データ構造体については、BEA Tuxedo オンライン・マニュアルの『BEA Tuxedo CORBA ネーム・サービス』を参照してください。

Wlstrader Value Type サンプル・アプリケーションのコンポーネント

この節では、Wlstrader Value Type サンプル・アプリケーションのコンポーネントと操作について説明します。この節の内容は次のとおりです。

- WebLogic Server Trader サンプル・アプリケーション
- WebLogic Server から CORBA クライアントへのマッピング
- BEA Tuxedo CORBA C++ クライアント

WebLogic Server Trader サンプル・アプリケーション

Wlstrader Value Type のサンプルは、WebLogic Server Trader サンプル・アプリケーションのコンポーネントに基づき、その XML デプロイメント・ファイルと Trader EJB を利用します。この例から最高の恩恵を得るためには、ソース・コード・ファイルを読み通して設計と手順を理解してください。デプロイメント・ファイルを調べると、Trader EJB の全体的な構造を理解できます。

Trader サンプルの WebLogic Server コード・サンプル・ファイルは、*wlserver6*\samples\examples\rmi_iiop\ejb\rmi_iiop ディレクトリにあります。

表 2-1 は、この例の Java インターフェイスの概要を示しています。

表 2-1 Java インターフェイスの概要

Java インターフェイス	説明
Trader	このインターフェイスのメソッドは、TraderBean のパブリック・インターフェイスです。

表 2-1 Java インターフェイスの概要 (続き)

Java インターフェイス	説明
TraderHome	このインターフェイスは、TraderBean.java のホーム・インターフェイスです。WebLogic では、コード生成のコンテナ・クラス TraderBeanC によってインプリメントされます。

表 2-2 は、この例の Java クラスの概要を示しています。

表 2-2 Java クラスの概要

Java クラス	説明
Tuxclient	この C++ クライアント・プログラムのロジックは、ステートレス・セッション Bean を呼び出し、演習 (Trader の作成、Trader を使用した株式の購入、値型で返される NumberTraded の取得、および Trader の削除) を実行します。
TraderBean	TraderBean はステートレス・セッション Bean です。
TradeResult	このクラスは、シリアライズ可能なインターフェイスをインプリメントし、値型で生成されたクラスを持ちません。TradeResult の状態はプライベートのインスタンス・メンバに維持され、値型の状態へのデフォルトのパブリック・アクセスはありません。TradeResult はシリアライズ可能なインターフェイスをインプリメントするので、ejbc ユーティリティは IDL ファイルでその値型を生成します。

WebLogic Server から CORBA クライアントへのマッピング

WebLogic Server アプリケーションから CORBA クライアントへのマッピングを派生させる場合、そのマッピング情報のソースは Java ソース・ファイルで定義されている EJB クラスです。WebLogic Server には、必要な CORBA IDL ファイルを生成するための `weblogic.ejbc` ユーティリティがあります。それらのファイルは、ターゲット EJB の状態と振る舞いの CORBA ビューを表します。

`weblogic.ejbc` ユーティリティでは、次の処理が実行されます。

- EJB クラス、インターフェイス、およびデプロイメント記述子ファイルを JAR ファイルに配置します。
- EJB の WebLogic Server コンテナ・クラスを生成します。
- RMI コンパイラを通じて各 EJB コンテナ・クラスを実行し、スタブとスケルトンを作成します。
- 上記のクラスへの CORBA インターフェイスを記述する CORBA IDL ファイルのディレクトリ・ツリーを生成します。

`weblogic.ejbc` ユーティリティでは、多くのコマンド修飾子がサポートされています。Wlstrader Value Type の例では、`ejbc` のステップでコマンド行で次の修飾子を呼び出します。

- `-idl`

  すべての適切なクラスの CORBA インターフェイス定義言語 (IDL) ファイルが作成されます。

- `-idlDirectory idlSources`

  `idlSources` という名前のディレクトリ・ツリーが作成され、IDL ファイルがこの位置に格納されます。ディレクトリ・ツリーの構造は、Java パッケージの階層に対応します。

- `-idlOverwrite`

  `idlSources` 出力ディレクトリの既存の IDL ファイルが上書きされます。

作成されたファイルは BEA Tuxedo IDL コンパイラを使用して処理され、`idlSources` ディレクトリからソース・ファイルが読み取られて、CORBA C++ スタブ・ファイルおよびスケルトン・ファイルが生成されます。値型を除くすべ

ての CORBA データ型は、生成されたそれらのファイルで満たされます。値型は、それらが定義または参照される各プラットフォームでインプリメントする必要があります。-i 修飾子を指定すると、IDL コンパイラで、*FileName*_i.h および *FileName*_i.cpp という名前のインプリメンテーション・ファイルが作成されます。たとえば、次の構文では、TradeResult_i.h および TradeResult_i.cpp というインプリメンテーション・ファイルが作成されます。

```
idl -IidlSources -i
idlSources\examples\rmi_iiop\ejb\rmi_iiop\TradeResult.idl
```

生成されたソース・ファイルは、値型のアプリケーション定義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。インプリメンテーション・ファイルは、CORBA クライアント・アプリケーションに含められます。

BEA Tuxedo CORBA C++ クライアント

CORBA C++ クライアント・プログラム tuxclient では、Trader に基づく演習が行われます。Trader トランザクションの結果は、TradeResult 値型を通じてクライアントに返されます。tuxclient アプリケーションでは、次の操作が実行されます。

- 受信する出力のソースである値型の値ファクトリを登録します。リスト 2-1 は tuxclient.cpp から抜粋したコードであり、register_value_factory オペレーションを示しています。

リスト 2-1 tuxclient.cpp— 値ファクトリの登録

```
 . . .
// 受信する可能性のあるすべての値型の値ファクトリ
// を登録する必要がある
//
examples_rmi_iiop_ejb_rmi_iiop_TradeResult_factory* TRf = new
    examples_rmi_iiop_ejb_rmi_iiop_TradeResult_factory();
orb->register_value_factory((char*const)
   examples::rmi_iiop::ejb::rmi_iiop::_tc_TradeResult->id(), TRf);
//
 . . .
```

- WebLogic Server の文字列化された IOR を格納する `ior.txt` ファイルを読み取ります。
- IOR をオブジェクトに変換します。
- COSNaming コンテキストを取得します。
- `Trader` のインスタンスを作成します。
- 取引要求を発行して株式を購入します。
- 要求された銘柄と株式数を指定するために `WStringValue` 値型を利用します。
- `TradeResult` 値型から実際の取引株式数を取得します。
- 購入した実際の株式数を報告します。

リスト 2-2 の `tuxclient.cpp` の抜粋コードには、`Trader` インスタンスを作成し、BEAS の株式を購入するステップが含まれています。

リスト 2-2 tuxclient.cpp—Trader の作成と株式の購入

```
 . . .
// Trader インスタンスの作成
::examples::rmi_iiop::ejb::rmi_iiop::Trader_ptr trader =
     home->create();

// 株式の購入
CORBA::WStringValue_ptr BEASsym = new
    CORBA::WStringValue(CORBA::wstring_dup((wchar_t *)L"BEAS"));
::examples::rmi_iiop::ejb::rmi_iiop::TradeResult_ptr result;
cout << "Buying 3000 shares of BEAS" << endl;
result = trader->buy(BEASsym, 3000);
 . . .
```

CORBA クライアントからの「購入」要求は、BEAS という銘柄の BEA 社の株式 3000 株です。実際に取り引きされる株式数は、WebLogic Server の例の `ejb-jar.xml` デプロイメント・ファイルで定義されている `tradeLimit` しきい値によって 500 になります。

リスト 2-3 ejb-jar.xml で定義された tradeLimit

```
 . . .
<env-entry-name>tradeLimit</env-entry-name>
<env-entry-type>java.lang.Integer</env-entry-type>
<env-entry-value>500</env-entry-value>
 . . .
```

Wlstrader Value Type サンプル・アプリケーションを実行して別の結果を得るには、ejb-jar.xml ファイルの定義、または tuxclient.cpp ソース・ファイルの定数を変更します。

Wlstrader Value Type サンプル・アプリケーションのビルドと実行

この節では、Wlstrader Value Type サンプル・アプリケーションをビルドおよび実行するプロセスを説明します。

例を実行する前に、host2ior ユーティリティを実行して WebLogic Server インターオペラブル・オブジェクト・リファレンス (IOR) を取得する必要があります。WebLogic Server のインストールでは、これを必ず 1 回行います。このファイルは、WebLogic Server をインストールしてからアプリケーションを実行するまでにいつでも取得できます。

wlstrader サンプル・アプリケーションをビルドおよび実行するには、次の手順に従います。各ステップは、以降の節で詳しく説明します。ここで詳しく説明する手順は、WebLogic Server の *wlserver6*\samples\examples\rmi_iiop\ejb\rmi_iiop\package-summary.html の情報を補足するものです。

- 開発環境の設定
- BEA Tuxedo Wlstrader Value Type ファイルのコピー
- 例のビルド
- サーバのコンフィギュレーション

開発環境の設定

Java で開発するときには、管理された開発環境を利用するようにしてください。

1. 「WebLogic Server Examples Guide」(製品に同梱) の説明に従って開発環境を設定します。「Setting Up Your Environment for Building and Running the Examples」を参照してください。
2. ユーザ・インターフェイス・ツールまたはコマンド・プロンプトを使用して、TUXDIR 環境変数を BEA Tuxedo のインストール・ディレクトリに設定します。

   次に例を示します。

   Windows

   ```
   > set TUXDIR=D:\TUXDIR
   ```

   UNIX

   ```
   ksh prompt> export TUXDIR=/usr/local/TUXDIR
   ```

BEA Tuxedo WIstrader Value Type ファイルのコピー

build コマンド・スクリプトの名前を変更し、BEA Tuxedo Wlstrader Value Type のファイルを BEA Tuxedo CORBA サンプル・ディレクトリから WebLogic Server の例のビルド領域にコピーします。

1. コマンド・プロンプトまたはユーザ・インターフェイス・ツールを使用して、build コマンド・スクリプトの名前を識別可能な一意の名前に変更します。ここからは、スクリプト名 tuxbuild を使用します。このファイルは、build コマンド・スクリプトが格納されている WebLogic Server の例のディレクトリにコピーします。

   次に例を示します。

   Windows

   ```
   rename %TUXDIR%\samples\corba\wlstrader\build.cmd tuxbuild.cmd
   ```

UNIX

```
mv $TUXDIR/samples/corba/wlstrader/build.sh tuxbuild.sh
```

2. コマンド・プロンプトまたはユーザ・インターフェイス・ツールを使用して、BEA Tuxedo CORBA サンプルの `wlstrader` ディレクトリから WebLogic Server の例のビルド領域にファイルをコピーします。

   次に例を示します。

   Windows

   ```
   copy %TUXDIR%\samples\corba\wlstrader\*.*
   D:\bea\wlserver6\samples\examples\rmi_iiop\ejb\rmi_iiop\*.*
   ```

   UNIX

   ```
   cp $TUXDIR/samples/corba/wlstrader/*.*
   /usr/local/wlserver6/samples/examples/rmi_iiop/ejb/rmi_iiop/*.*
   ```

表 2-3 のサンプル・ファイルは、現時点で WebLogic Server の例のビルド領域にあるはずです。

表 2-3 Wlstrader Value Type サンプル・ファイル

ファイル	説明
`TradeResult_i.cpp`	TradeResult 値型のコンストラクタを定義する C++ ソース。`TradeResult_i.cpp` と `TradeResult_i.h` は、TradeResult 値型の具象クラスの実際のインプリメンテーションを提供します。
`TradeResult_i.h`	TradeResult 値型のインプリメンテーション・クラスを定義する C++ ヘッダ・ファイル。
`tuxbuild.cmd`	名前変更した Windows システム用の build コマンド・ファイル。
`tuxbuild.sh`	名前変更した UNIX システム用の build コマンド・スクリプト。
`tuxclient.cpp`	CORBA クライアント・アプリケーションの C++ ソース・コード。

例のビルド

この例で名前変更した build スクリプトを実行します。このファイルは、BEA Tuxedo の samples\corba\wlstrader ディレクトリから WebLogic Server の samples\examples\rmi_iiop\ejb\rmi_iiop ディレクトリにコピーする前に名前変更したファイルです。このスクリプトでは、次のステップが実行されます (Windows 環境)。

1. TUXDIR 環境変数を設定します。このステップは、tuxclient.cmd スクリプトで修正できます。または、TUXDIR が既に定義されている場合はコメント・アウトすることもできます。
2. %TUXDIR%\bin ディレクトリで idl 実行可能プログラムを定義し、必要なコマンド行パラメータを設定して、IDL2CPP 環境変数を設定します。
3. build ディレクトリ構造を作成し、デプロイメント記述子をコピーし、*.gif 画像をコピーします。

```
> mkdir build build\META-INF build\images
> copy *.xml build\META-INF
> copy *.gif build\images
```

4. EJB クラスを build ディレクトリにコンパイルします (JAR の準備)。

```
> javac -d build Trader.java TraderHome.java
        TraderResult.java TraderBean.java
```

5. EJB JAR ファイル (XML デプロイメント記述子を含む) を作成します。

```
> cd build
> jar cv0f std_ejb_over_iiop.jar META-INF examples images
> cd ..
```

6. JAR ファイルに対して weblogic.ejbc ユーティリティを実行します。生成される IDL ファイルは idlSources ディレクトリに配置されるように指定します。

```
> java weblogic.ejbc -compiler javac -keepgenerated
  -idl -idlDirectory idlSources
  -iiop build\std_ejb_over_iiop.jar
  %APPLICATIONS%\ejb_over_iiop.jar
```

7. EJB インターフェイスとクライアント・アプリケーションを CLIENT_CLASSES ターゲット変数で定義されたディレクトリにコンパイルします。

```
> javac -d %CLIENT_CLASSES% Trader.java TraderHome.java
  TradeResult.java Client.java
```

8. weblogic.ejbc のステップでビルドされた IDL ファイルに対して IDL コンパイラを実行し、C++ ソース・ファイルを作成します。

```
>%IDL2CPP% idlSources\examples\rmi_iiop\ejb\rmi_iiop\Trader.idl
  ...
>%IDL2CPP% idlSources\javax\ejb\RemoveException.idl
```

9. buildobjclient コマンドを呼び出して、CORBA C++ クライアントの実行可能プログラム tuxclient.exe をビルドします。

サーバのコンフィギュレーション

WebLogic Server の例のマニュアルで定義されているステップを調べてインプリメントし、WebLogic Server を起動します。「Starting WebLogic Server with the Examples Configuration」を参照してください。すべての例のコンフィギュレーション属性を格納するコンフィギュレーション・ファイル (WebLogic Server インストール・ディレクトリの config\examples\config.xml) を調べます。サーバ起動スクリプトは、次のようにして実行します。

Windows

```
prompt> startExampleServer
```

Windows プラットフォームでは、[スタート] メニューから起動することもできます。

UNIX

```
$ sh startExamplesServer.sh
```

例の実行

環境を設定し、tuxclient.exe 実行可能プログラムを起動して、Wlstrader Value Type サンプル・プログラムを実行します。Windows 環境では次のように行います。

1. ejb_over_iiop.jar ファイルを WebLogic Server の CLASSPATH に追加します。次に例を示します。

   ```
   prompt> set CLASSPATH=%CLASSPATH%;%WL_HOME%\config\
           examples\applications\ejb_over_iiop.jar
   ```

2. host2ior ユーティリティを実行して、WebLogic Server インターオペラブル・オブジェクト・リファレンス (IOR) を取得しておきます。ior.txt ファイルは、tuxclient.exe CORBA C++ クライアント・プログラムからアクセス可能でなければなりません。このファイルは、WebLogic Server のインストールごとに 1 つ作成します。次に例を示します。

   ```
   prompt> java utils.host2ior hostname port
   ```

3. 例をビルドするときに作成した実行可能クライアント・アプリケーションを起動して、tuxclient.exe CORBA クライアント・プログラムを起動します。

   ```
   prompt> tuxclient.exe
   ```

4. 次のような CORBA クライアントの出力メッセージが表示されます。

   ```
   Creating a trader

   Buying 3000 shares of BEAS
   500 shares bought
   Buying 100 shares of PSFT
   100 shares bought

   End statelessSession.Client
   ```

リファレンス情報

WebLogic Server アプリケーションとやり取りする BEA Tuxedo CORBA クライアント・アプリケーションの開発プロセスは EJB から始まります。新しい WebLogic Server アプリケーションを作成する場合、その最初のステップはサーバを設計およびインプリメントすることです。

以降の節では、RMI over IIOP CORBA クライアントと WebLogic Server EJB の間の接続をインプリメントするために必要な手順についての情報の参照先を示します。

WebLogic Server

次のトピックは、WebLogic Server オンライン・マニュアルで参照できます。

- WebLogic RMI over IIOP (『WebLogic RMI over IIOP プログラマーズ ガイド』)
- WebLogic EJB (『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』)

■WebLogic RMI (『WebLogic RMI プログラマーズ ガイド』)

Object Management Group (OMG)

CORBA オブジェクトおよび値型 (値で渡すことができるオブジェクト) の情報は、OMG の Web サイト (http://www.omg.org) で参照できます。

BEA Tuxedo CORBA

BEA Tuxedo のオンライン・マニュアルでは、BEA Tuxedo 製品の CORBA について詳しく説明されています。ここで紹介する参照先では、CORBA クライアントの作成、COSNaming サービス、IDL コマンド、および BEA Tuxedo の値型のサポートについての情報を参照できます。

- 『BEA Tuxedo CORBA クライアント・アプリケーションの開発方法』
- 『BEA Tuxedo CORBA ネーム・サービス』
- 『BEA Tuxedo コマンド・リファレンス』
- 『BEA Tuxedo CORBA プログラミング・リファレンス』

3 WebLogic Server から CORBA オブジェクトへの接続

ここでは、次の内容について説明します。

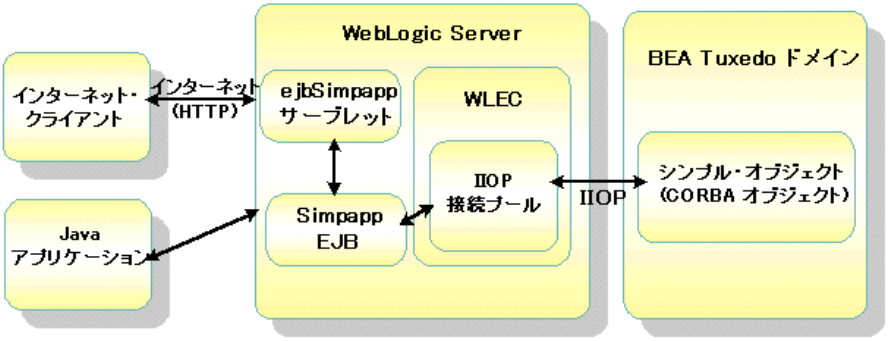
WLEC EJB simpapp サンプル・アプリケーション

WebLogic Server WebLogic Enterprise Connectivity (WLEC) EJB simpapp サンプル・アプリケーションは、WebLogic Enterprise Connectivity (WLEC) を使用して WebLogic Server 上の状態を持たない EJB から BEA Tuxedo CORBA オブジェクトにアクセスする方法を例示します。

注記 このサンプル・アプリケーションをビルドおよび実行するための情報は、WebLogic Server のオンライン・マニュアルで参照できます。この節では、補足的な情報を提供し、BEA Tuxedo 8.0 の用語を使用します。WebLogic Server 6.0 のマニュアルでは、WebLogic Enterprise 5.1 の用語が使用されています。

図 3-1 は、WLEC EJB simpapp サンプル・アプリケーションで実現される接続を説明しています。

図 3-1　WebLogic Server EJB から BEA Tuxedo CORBA オブジェクト



この例では、WebLogic Server Enterprise JavaBean と BEA Tuxedo simpapp サンプル・アプリケーションのシンプル・オブジェクトを結合します。この例は、次の 2 通りの方法で実行できます。

- Java アプリケーションから
- インターネット・クライアントおよびサーブレットから

WLEC EJB simpapp サンプル・アプリケーションのビルドと実行

WLEC EJB simpapp サンプル・アプリケーションをビルドおよび実行するための WebLogic Server コード・サンプル・ファイルと手順は、WebLogic Server の *wlserver6*\samples\examples\wlec\ejb\simpapp ディレクトリにあります。以降の節の情報は、このサンプルの WebLogic Server マニュアルの内容を補足するものです。

説明

wlserver6\samples\examples\wlec\ejb\simpapp\package-summary.html ファイルでは、サンプル・アプリケーションをビルドおよび実行する方法が説明されています。

起動時に、WLS では simpapp ドメイン (BEA Tuxedo ドメイン) の WLEC 接続プールが作成されます。WebLogic Server simpapp EJB は、実行時にリモートの BEA Tuxedo CORBA クライアントとして機能します。

前提条件

このサンプル・アプリケーションを実行するためには、次に示すソフトウェア製品が必要です。バージョン情報については、『BEA Tuxedo システムのインストール』を参照してください。次のソフトウェア製品をインストールして設定してください。

- WebLogic Server (WLS)
- Java Software Development Kit (SDK)
- BEA Tuxedo CORBA

例のビルド

BEA Tuxedo CORBA simpapp サンプルをビルドして実行するには、BEA Tuxedo オンライン・マニュアルのサンプルのページを表示して、simpapp サンプル・アプリケーションを選択します。

注記 WebLogic Server の起動スクリプトで CLASSPATH に RemoteObjectReference クラスを追加する必要はありません。BEA Tuxedo 8.0 リリースには、wlej2eecl.jar ファイルは含まれていません。

索引