BEA WebLogic Server™

WebLogic Enterprise Connectivity ユーザーズ ガイド（非推奨）

BEA WebLogic Server バージョン 7.0
マニュアルの改訂 : 2002 年 6 月 28 日

WebLogic Enterprise Connectivity ユーザーズ ガイド（非推奨）

パート番号	マニュアルの改訂	ソフトウェアのバージョン
なし	2002 年 6 月 28 日	BEA WebLogic Server バージョン 7.0

目次

このマニュアルの内容

このマニュアルでは、BEA WebLogic Server™ 製品の WebLogic Enterprise Connectivity（WLEC）コンポーネントの使い方について説明します。この製品は非推奨になり、現在はサポートされていません。

このマニュアルの内容は以下のとおりです。

- 第 1 章「WebLogic Enterprise Connectivity の概要」では、BEA Tuxedo™ 製品における CORBA 環境および WLEC コンポーネントの概要について説明します。CORBA 環境と WLEC コンポーネントが対話する方法についても説明します。
- 第 2 章「CORBA オブジェクトを呼び出す WebLogic Server クライアントの作成」では、WebLogic Server から BEA Tuxedo CORBA オブジェクトへのアクセスに必要な手順について説明します。

対象読者

このマニュアルは、WebLogic Server クライアント（サーブレット、EJB、JSP、または RMI オブジェクト）から BEA Tuxedo CORBA オブジェクトを呼び出すプログラマを対象としています。WLEC コンポーネントは、WebLogic Server で確立されたセキュリティ コンテキストを BEA Tuxedo ドメインへ伝播するメカニズムを提供します。

e-docs Web サイト

BEA WebLogic Server 製品のドキュメントは、BEA の Web サイトで入手できます。BEA のホーム ページで [製品のドキュメント] をクリックします。

このマニュアルの印刷方法

Web ブラウザの [ファイル | 印刷] オプションを使用すると、Web ブラウザからこのマニュアルのファイルを一度に 1 つずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。PDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Server ドキュメントのホーム ページを開き、[ドキュメントのダウンロード] 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 は Adobe の Web サイト（http://www.adobe.co.jp）で無料で入手できます。

サポート情報

BEA WebLogic Server のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、電子メールで **docsupport-jp@beasys.com** までお送りください。寄せられた意見については、WebLogic Server ドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、ご使用のソフトウェアの名前とバージョン、およびドキュメントのタイトルと日付をお書き添えください。

本バージョンの BEA WebLogic Server について不明な点がある場合、または BEA WebLogic Server のインストールおよび動作に問題がある場合は、BEA WebSupport（www.bea.com）を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、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 サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号、ファクス番号

- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

表記法	適用
太字のテキスト	用語集で定義されている用語を示す。
〔Ctrl〕+〔Tab〕	複数のキーを同時に押すことを示す。
斜体	強調または書籍のタイトルを示す。

等幅テキスト	コード サンプル、コマンドとそのオプション、データ構造体とそのメンバー、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。
	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	コード内の変数を示す。 例： `void **commit** ( )`

表記法	適用
斜体の等幅テキスト	コード内の変数を示す。 例： `String expr`
すべて大文字のテキスト	デバイス名、環境変数、および論理演算子を示す。 例： LPT1 SIGNON OR
{ }	構文の中で複数の選択肢を示す。実際には、この括弧は入力しない。
[ ]	構文の中で任意指定の項目を示す。実際には、この括弧は入力しない。 例： `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
|	構文の中で相互に排他的な選択肢を区切る。実際には、この記号は入力しない。
...	コマンドラインで以下のいずれかを示す。 ■ 引数を複数回繰り返すことができる。 ■ 任意指定の引数が省略されている。 ■ パラメータや値などの情報を追加入力できる。 実際には、この省略符号は入力しない。 例： `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	コード サンプルまたは構文で項目が省略されていることを示す。 実際には、この省略符号は入力しない。

1 WebLogic Enterprise Connectivity の概要

以下の節では、WebLogic Enterprise Connectivity について概説します。

- WLEC の非推奨
- BEA Tuxedo の CORBA 環境
- BEA Tuxedo 製品の CORBA オブジェクト
- ドメイン、トランザクション、およびトランザクション コンテキスト
- 環境オブジェクト
- IIOP と IIOP リスナ / ハンドラ
- BEA Tuxedo の CORBA 環境に関する詳細情報
- WebLogic Enterprise Connectivity とは
- WebLogic Enterprise Connectivity の主要な機能
- WebLogic Enterprise Connectivity システム アーキテクチャ
- WebLogic Enterprise Connectivity コンポーネントにおけるクライアントとサーバ
- WLEC 接続プール
- WebLogic Enterprise Connectivity コンポーネントがサポートする BEA Tuxedo セキュリティ機能
- セキュリティ コンテキストの伝播
- 接続障害の処理

WLEC の非推奨

WebLogic Enterprise Connectivity (WLEC) は、このリリースの WebLogic Server 7.0 で非推奨となりました。WLEC を使用している Tuxedo CORBA アプリケーションは WebLogic Tuxedo Connector に移行してください。詳細については、「WebLogic Tuxedo Connector」を参照してください。

BEA Tuxedo の CORBA 環境

BEA Tuxedo は、エンタープライズ向けコンポーネントベース ソリューションの構築、デプロイメント、および管理を可能にするソフトウェア製品です。BEA Tuxedo を使用すると、Common Object Request Broker Architecture (CORBA) の Object Request Broker (ORB) とアプリケーション トランザクション モニタ インタフェース（ATMI）で使用できるオンライン トランザクション処理（OLTP）の機能を組み合わせることができます。これにより、高度に管理された環境でスケーラブルかつセキュリティが確保されたトランザクション e- コマース アプリケーションを提供するためのプラットフォームが実現します。

以下の節では、BEA Tuxedo CORBA の専門用語をいくつか解説します。

BEA Tuxedo 製品の CORBA オブジェクト

CORBA は、分散ソフトウェア アプリケーションの構築と統合に対するオブジェクト指向アプローチを促進する、特定の言語に依存しない仕様です。ビジネス ロジックを CORBA オブジェクトとしてさまざまな言語で実装することができます。BEA Tuxedo の CORBA 環境では、C++、Automation、および Java 用の言語バインディングをサポートしています。たとえば、銀行業務アプリケーションには、顧客口座のオブジェクトが存在することがあります。これらの顧客口座オブジェクトは、預け入れ、引き出し、および残高照会を行うための処理を持つことができます。

CORBA オブジェクトを記述するには、Object Management Group (OMG) のインタフェース定義言語（IDL）を使用します。CORBA オブジェクトの OMG IDL 記述を作成してから、その OMG IDL をコンパイラにかけます。このコンパイラによって、スタブとスケルトンが生成されます。

CORBA ファクトリは、別の CORBA オブジェクトへのオブジェクト参照を作成するための操作を提供するオブジェクトです。サーバ アプリケーションは CORBA ファクトリを使用して、クライアント アプリケーションがそのサーバ アプリケーションに実装されている CORBA オブジェクトにアクセスできるようにします。クライアント アプリケーションは、サーバ アプリケーションで管理される CORBA オブジェクトへの参照を取得しなければならない場合には、通常、CORBA ファクトリからオブジェクト参照を取得します。

Object Request Broker (ORB) は、サーバ アプリケーションによって管理される分散オブジェクトとクライアント アプリケーションの通信を可能にする通信バスです。CORBA 環境では、アプリケーションは、通信のためのネットワークやオペレーティング システムに関する情報を持っている必要はありません。その代わり、異質なクライアント アプリケーションとサーバ アプリケーションが ORB と通信するのです。ORB は、クライアントの要求を適切なサーバ アプリケーションに届け、サーバの応答を要求元のクライアント アプリケーションに返します。

ドメイン、トランザクション、およびトランザクション コンテキスト

BEA Tuxedo ドメインは、管理の単位となる一群のオブジェクト、サービス、マシン、およびリソースのことです。ドメインは、アプリケーション機能、セキュリティ上のニーズ、地理的な位置といった特性に基づいて設定できます。たとえば、ある銀行が、顧客口座用のオブジェクト、サービス、マシン、およびリソースから構成されているドメインを持っているとします。この場合、その銀行はその従業員と支払給与のリソース用に別個のドメインを持つことができます。

トランザクションは、ビジネス ルールに基づく一連の操作のことです。これらの操作は、たとえ地理的に分散していたとしても 1 つの論理単位となります。1 つの単位として機能することで、そのトランザクションのすべての操作が正常に

完了する（この場合、トランザクションは正常に完了する）か、すべての操作がロールバックされる（トランザクションは失敗する）かのどちらかになります。たとえば、1 つのトランザクションで、ある顧客の口座から現金を引き出してそれを別の顧客の口座に入金するとします。このトランザクションは 2 つの操作から構成されます。トランザクションが成功するには、両方の操作が正常に完了しなければなりません。

トランザクション コンテキストは、トランザクションのスコープを定義するもので、そのトランザクションに参加しているオブジェクトによって共有されます。トランザクション コンテキストは、ローカル変数、ロック、カーソル位置、ファイル制御ブロックなどのさまざまなタイプのデータから構成されます。

環境オブジェクト

BEA Tuxedo 製品の CORBA 環境では、クライアント アプリケーションから CORBA サービスを利用するための以下の環境オブジェクトを提供します。

- Bootstrap オブジェクトは、クライアント アプリケーションと BEA Tuxedo ドメインの間の通信を確立します。このオブジェクトは、BEA Tuxedo ドメイン内の他の環境オブジェクトへのオブジェクト参照も取得します。
- FactoryFinder オブジェクトは、クライアント アプリケーションが CORBA ファクトリを見つけるためのオブジェクトです。CORBA ファクトリは、CORBA オブジェクトのオブジェクト参照を作成できます。クライアント アプリケーションから使用可能なファクトリは、起動時に FactoryFinder に登録されたものです。
- TransactionCurrent オブジェクトは、クライアント アプリケーションがトランザクションを管理できるようにするためのオブジェクトです。TransactionCurrent は、CORBA Object Transaction Service (OTS) の BEA Tuxedo における実装で、複数のトランザクション モデルをサポートします。TransactionCurrent オブジェクトが提供する操作には、`begin()`、`commit()`、`rollback()`、`suspend()`、`resume()`、`get_status()` といったものがあります。
- UserTransaction オブジェクトは、クライアント アプリケーションがトランザクションに参加できるようにするためのオブジェクトです。この環境オブジェクトは、Sun Microsystems, Inc. の Java Transaction Application (JTA) プ

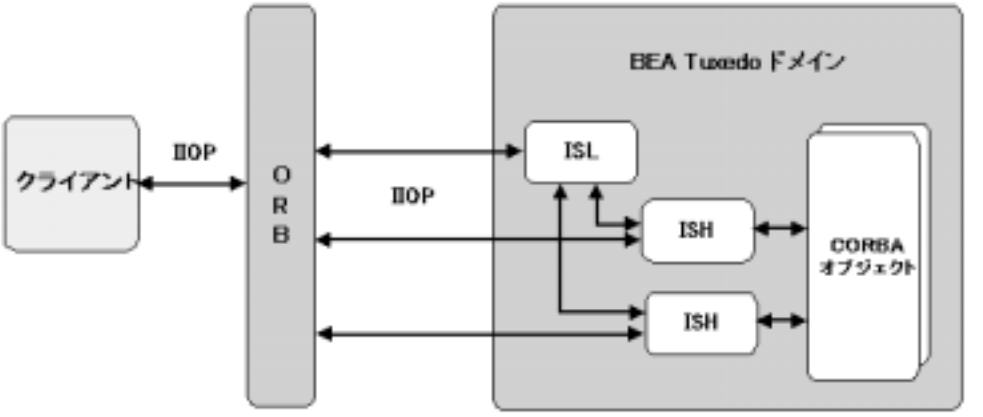
ログラミング インタフェースを実装したものです。UserTransaction オブジェクトは、CORBA Java オブジェクトで使用できます。

IIOP と IIOP リスナ / ハンドラ

Internet Inter-ORB Protocol (IIOP) は、ORB 間の相互運用に関する CORBA 仕様に定義されている標準プロトコルです。BEA Tuxedo CORBA 環境では、IIOP リスナ / ハンドラが、ORB および CORBA クライアント アプリケーション間の接続を処理します。IIOP リスナ（ISL）は、送られてくる通信データを、リモートの CORBA クライアントに代わって管理します。各 ISL には、1 つまたは複数の IIOP ハンドラ（ISH）が関連付けられています。ISL は、クライアント アプリケーションを ISH に割り当て、クライアントから生じる負荷を ISH 間で分散します。ISH は、クライアント アプリケーションと CORBA オブジェクトの間の通信リンクです。リモート クライアントをサポートする各 BEA Tuxedo ドメインには、ISL が少なくとも 1 つ存在します。

図 1-1 に、IIOP の概要を示します。

図 1-1 BEA Tuxedo システムでの IIOP の機能



BEA Tuxedo の CORBA 環境に関する詳細情報

BEA Tuxedo 製品における CORBA 環境についての詳細は、BEA Tuxedo のマニュアルを参照してください。BEA Tuxedo 製品を初めて使用する場合は、『BEA Tuxedo 製品の概要』を一読されることをお勧めします。

WebLogic Enterprise Connectivity とは

WebLogic Enterprise Connectivity は、WebLogic Server のコンポーネントの 1 つです。WLEC を使用すると、WebLogic Enterprise Connectivity 接続プール（Administration Console では WLEC 接続プールと呼ばれます）を使用して、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 BEA Tuxedo CORBA オブジェクトを呼び出すことができます。

WebLogic Enterprise Connectivity の主要な機能

WebLogic Enterprise Connectivity コンポーネントの主要な機能は以下のとおりです。

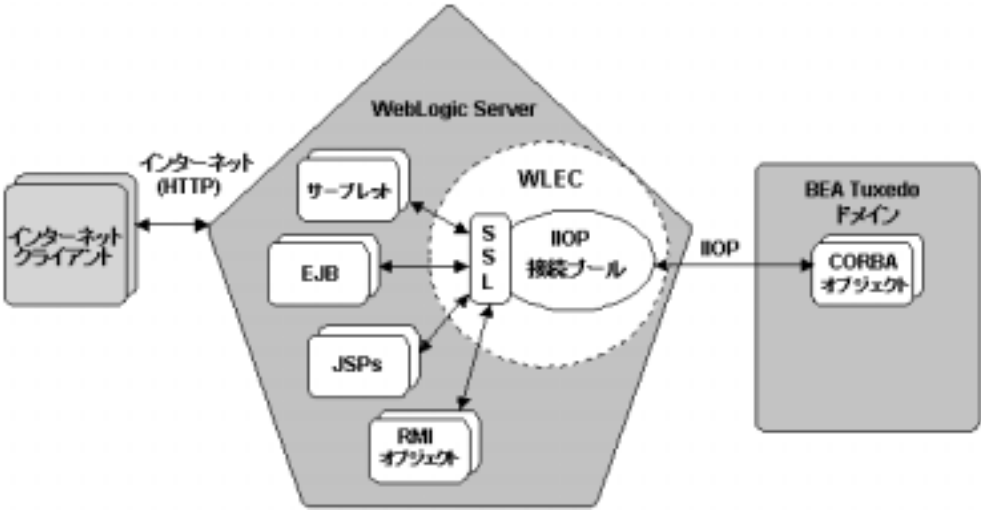
- BEA Tuxedo CORBA アプリケーションへの WebLogic Enterprise Connectivity 接続のプーリング
- 単一の WebLogic Server プロセスから開始できる複数のアクティブな CORBA クライアント トランザクション
- WebLogic Server の Administration Console を使った WLEC 接続プールのコンフィグレーション

- WebLogic Server の Administration Console を使った WLEC 接続プールのモニタ
- Secure Sockets Layer (SSL) プロトコルのサポート
- WebLogic Server から BEA Tuxedo ドメインへのセキュリティ コンテキストの伝播
- 実行時のプールの再初期化

WebLogic Enterprise Connectivity システム アーキテクチャ

図 1-2 に、WebLogic Enterprise Connectivity システム アーキテクチャと、その WebLogic Server と BEA Tuxedo システムとの関係を示します。

図 1-2 WebLogic Enterprise Connectivity システム アーキテクチャ



BEA Tuxedo ドメインごとに、WebLogic Server は WLEC 接続プールを 1 つ作成します。WLEC 接続プールは、WebLogic Server の Administration Console でコンフィグレーションされます。WebLogic Server クライアントは、その WLEC 接続プールを使用して、BEA Tuxedo ドメイン内の CORBA オブジェクトにアクセスします。それらのオブジェクトは、リモート マシン上にあっても、ファイアウォールの向こうにあっても、あるいはその両方であってもかまいません。

WebLogic Enterprise Connectivity コンポーネントにおけるクライアントとサーバ

インターネット クライアントは、WebLogic Server 上のアプリケーションからサービスを受けます。これらのアプリケーションは、WebLogic Enterprise Connectivity を介して BEA Tuxedo ドメインのクライアントとして動作します。BEA Tuxedo ドメインは要求された CORBA オブジェクトを提供し、その結果を WebLogic Server クライアントに送り返します。すると WebLogic Server クライアントは、その結果を処理し、その他の処理を実行して、その結果をインターネット クライアントに送信します。

WLEC 接続プール

WebLogic Enterprise Connectivity コンポーネントは、WLEC 接続プールを使用して WebLogic Server クライアントが BEA Tuxedo ドメインに接続できるようにします。WLEC 接続プールとは、BEA Tuxedo ドメインへの IIOP 接続の集合のことです。WebLogic Server は、起動時に WLEC 接続プールを作成し、必要に応じて WebLogic Server クライアントに接続を割り当てます。限られた数の接続で多数のユーザにサービスを提供できるため、WLEC 接続プールは効率的です。接続を作成するためのオーバーヘッドは起動時に処理されるため、WebLogic Server は CORBA オブジェクトとそれらの操作にすばやくアクセスできます。

WebLogic Enterprise Connectivity 接続プールは、以下の機能を備えています。

- IIOP を使用します。

- BEA Tuxedo ドメインごとに WLEC 接続プールを 1 つサポートします。
- WebLogic Server は、複数のアクティブな BEA Tuxedo トランザクション コンテキストを同時に持つことができます。ただし、WebLogic Server 内の 1 つのスレッドは、アクティブなクライアント トランザクション コンテキストを 1 度に 1 つしか持つことができません。WebLogic Enterprise Connectivity コンポーネントは、ネストされたトランザクションをサポートしません。
- 同一の物理的接続を用いた複数の並行要求をサポートします。各要求は、独自のセキュリティ コンテキストを備えています。
- 実行時に WLEC 接続プールを再初期化できます。

WebLogic Enterprise Connectivity コンポーネントがサポートする BEA Tuxedo セキュリティ機能

WebLogic Enterprise Connectivity コンポーネントは、以下の BEA Tuxedo セキュリティ機能を提供します。

- 認証

  認証によって、2 つの通信グループに互いに通信する権限が与えられます。BEA Tuxedo システムでは、パスワードと証明書に基づく認証をサポートしています。

- 機密性

  機密性とは、意図した受信者以外の第三者から通信上の秘密を守ることができる能力のことです。これは、すべてのデータを暗号化することで達成されます。

- 整合性

  整合性とは、転送中のデータが途中で変更されていないことを保証するものです。

- SSL プロトコル

  SSL 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でセキュリティが確保された通信を確立するプロトコルです。SSL は、CORBA オブジェクトへの IIOP 要求に対して提供されます。

注意： WebLogic Enterprise Connectivity コンポーネントは、BEA Tuxedo では CORBA セキュリティのアプリケーション プログラミング インタフェース（API）をサポートしていません。

セキュリティ コンテキストの伝播

WebLogic Enterprise Connectivity を使用すると、WebLogic Server で確立されたセキュリティ コンテキストを使用して、BEA Tuxedo ドメインのセキュリティ ID を確立できます。このようにセキュリティ資格を渡すことを、セキュリティ コンテキストの伝播と呼びます。

セキュリティ コンテキストの伝播には、WLEC 接続プールが必要です。WLEC 接続プールでは、各ネットワーク接続は、WebLogic Server のシステム管理者によって定義されたユーザ ID により認証されます。WLEC 接続プールを確立するには、パスワードまたは証明書に基づく認証を使用します。

図 1-3 に示すように、WLEC 接続プールを使用すると、クライアントから WebLogic Server を経て BEA Tuxedo 製品の CORBA 環境までセキュリティ情報を伝播できるようになります。

図 1-3 セキュリティ コンテキストの伝播

接続障害の処理

WebLogic Enterprise Connectivity コンポーネントは、WLEC 接続プールごとに 2 つの ISL アドレス リスト（プライマリ リストとフェイルオーバ リスト）を使用して、接続障害を処理します。WebLogic Enterprise Connectivity コンポーネントは、以下の場合に接続障害を処理します。

- WebLogic Server が起動したとき

  サーバの起動時にプライマリ アドレス リストのすべての ISL にアクセスできない場合、WebLogic Enterprise Connectivity はフェイルオーバ リストの ISL アドレスを使用します。

- WLEC 接続プールがアクティブな接続を失ったとき

  プールが接続を失った場合、WebLogic Server はプライマリ アドレス リストの他のアドレスを使って再接続を試みます。プライマリ リスト内のどのアドレスを使っても再接続できない場合、WebLogic Server はフェイルオーバ リストのアドレスを使って再接続を試みます。失われた接続は、必要になったときにだけ再び開かれます。WLEC 接続プールに対する現在の負荷が、失われた接続を再び開くほど重くない場合、それらの接続は切断されたままとなり、他のアクティブな接続が代わりに使われます。

2 CORBA オブジェクトを呼び出す WebLogic Server クライアントの作成

以下の節では、WebLogic Server から BEA Tuxedo CORBA オブジェクトにアクセスするための要件について説明します。

- 始める前に
- WLEC 接続プールのコンフィグレーション
- BEA Tuxedo CORBA オブジェクトへのアクセス
- トランザクションに関する詳細情報

始める前に

BEA Tuxedo CORBA オブジェクトを呼び出す WebLogic Server クライアントを実装するには、あらかじめ以下の作業を行っておく必要があります。

- WebLogic Server をコンフィグレーションして実行します。
- BEA Tuxedo CORBA サーバ アプリケーションをコンフィグレーションして実行します。
- 第 1 章「WebLogic Enterprise Connectivity の概要」で説明した WebLogic Enterprise Connectivity アーキテクチャについて理解しておきます。
- インストールされている WebLogic Server の `/samples/examples/wlec` ディレクトリに収められている WebLogic Enterprise Connectivity のサンプルを実行します。

WLEC 接続プールのコンフィグレーション

WebLogic Server クライアントからアクセスする BEA Tuxedo オブジェクトを持つ BEA Tuxedo ドメインごとに、WLEC 接続プールをコンフィグレーションします。WebLogic Server クライアント用に確立されたセキュリティ コンテキストは WLEC 接続プールを介して BEA Tuxedo ドメインに伝播されるので、WLEC 接続プールはセキュリティ コンテキスト伝播機能の基礎であると言えます。

WLEC 接続プールを使用する前に、*WL_HOME*/lib/wleorb.jar、*WL_HOME*/lib/wlepool.jar、*TUXDIR/udataobj/java/jdk/wleclient.jar* を、startAdminWebLogic.sh ファイルまたは startAdminWebLogic.cmd ファイルの CLASSPATH 変数に追加します。WLE 5.1 からサービスを受ける EJB を要求する JSP を使用している場合は、WLE 5.1 *wlej2eecl.jar* ファイルの場所を startAdminWebLogic.sh ファイルまたは startAdminWebLogic.cmd ファイルの CLASSPATH 変数に追加します。

セキュリティ コンテキストを伝播するために、新規の WLEC 接続プールを作成します。WLEC 接続プールを作成するには、以下の手順に従います。

1. Administration Console の左ペインで、[サービス | WLEC] ノードを選択します。Administration Console の右ペインで、[新しい WLEC Connection Pool のコンフィグレーション] リンクをクリックします。次の表にある属性を定義します。

表 2-1 [一般] タブの WLEC 接続プールの属性

属性	説明
[名前]	WLEC 接続プールの名前。この名前は WLEC 接続プールごとにユニークでなければならない。

表 2-1 [一般] タブの WLEC 接続プールの属性（続き）

属性	説明
[プライマリ アドレス]	WLEC 接続プールと BEA Tuxedo ドメインとの接続を確立するために使用する IIOP リスナ / ハンドラのアドレスのリスト。各アドレスのフォーマットは、*//hostname:port*。 アドレスは、UBBCONFIG ファイルに定義されている ISL アドレスと一致しなければならない。アドレスとアドレスの区切りにはカンマを使用する。たとえば、//main1.com:1024, //main2.com:1044 になる。 SSL プロトコルを使用するよう WLEC 接続プールをコンフィグレーションするには、IIOP リスナ / ハンドラのアドレスに corbalocs プレフィックスを付ける。たとえば、corbalocs:*//hostname:port* になる。
[フェイルオーバー アドレス]	[プライマリ アドレス] 属性に定義されているアドレスを使って接続を確立できない場合に使用される IIOP リスナ / ハンドラのアドレスのリスト。アドレスとアドレスの区切りにはカンマを使用する。この属性は省略可能。
[ドメイン]	WLEC 接続プールの接続先 BEA Tuxedo ドメインの名前。WLEC 接続プールは、BEA Tuxedo ドメインにつき 1 つしか定義できない。ドメイン名は、BEA Tuxedo ドメインの UBBCONFIG ファイルの RESOURCES セクションの domainid パラメータに一致しなければならない。
[最小プール サイズ]	WebLogic Server が起動したときに、WLEC 接続プールに追加する IIOP 接続の数。デフォルトは 1。
[最大プール サイズ]	WLEC 接続プールから開始できる IIOP 接続の最大数。デフォルトは 1。

2. [作成] ボタンをクリックします。

3. WebLogic Server セキュリティ レルム内のユーザのセキュリティ コンテキストを BEA Tuxedo ドメインに伝播します。伝播するには、接続プールの [コンフィグレーション] タブにある [セキュリティ] タブの属性を定義します。次の表では、これらの属性について説明します。

表 2-2 [セキュリティ] タブの WLEC 接続プールの属性

属性	説明
[ユーザ名]	BEA Tuxedo ユーザ名。この属性は、BEA Tuxedo ドメインのセキュリティ レベルが USER_AUTH、ACL、または MANDATORY_ACL の場合にのみ指定する。
[ユーザ パスワード]	[ユーザ名] 属性に定義したユーザのパスワード。この属性は、[ユーザ名] 属性を定義する場合にのみ指定する。
[ユーザ ロール]	BEA Tuxedo ユーザ ロール。この属性は、BEA Tuxedo のセキュリティ レベルが APP_PW、USER_AUTH、ACL、または MANDATORY_ACL の場合にのみ指定する。
[アプリケーション パスワード]	BEA Tuxedo CORBA アプリケーションのパスワード。この属性は、BEA Tuxedo ドメインのセキュリティ レベルが APP_PW、USER_AUTH、ACL、または MANDATORY_ACL の場合にのみ指定する。
[最小暗号化レベル]	BEA Tuxedo ドメインと WebLogic Server との間で使用される SSL の最小暗号化レベル。指定できる値は、0、40、56、128。デフォルト値は 0。ゼロ（0）は、データを署名するが暗号化しないことを示す。40、56、および 128 は暗号キーの長さ（ビット単位）を指定する。最小暗号化レベルが満たされていない場合、BEA Tuxedo ドメインと WebLogic Server との SSL 接続は失敗する。

表 2-2 [セキュリティ] タブの WLEC 接続プールの属性（続き）

属性	説明
[最大暗号化レベル]	BEA Tuxedo ドメインと WebLogic Server との間で使用される SSL の最大暗号化レベル。指定できる値は 0、40、56、および 128。デフォルト値は 0。ゼロ（0）は、データを署名するが暗号化しないことを示す。40、56、および 128 は暗号キーの長さ（ビット単位）を指定する。最小暗号化レベルが満たされていない場合、BEA Tuxedo ドメインと WebLogic Server との SSL 接続は失敗する。
[証明書を有効化]	証明書を使用できるようにするチェックボックス。 デフォルトでは、証明書は無効。
[セキュリティ コンテキストを有効化]	BEA Tuxedo ドメインに渡される WebLogic Server ユーザのセキュリティ コンテキストを渡す場合は、このチェックボックスをオンにする。 デフォルトでは、セキュリティ コンテキストは無効。

4. 変更を保存するには、[適用] ボタンをクリックします。
5. [対象] タブをクリックしてサーバを選択します。[適用] をクリックします。
6. WebLogic Server を再起動します。
7. `tpusradd` コマンドを実行して、WebLogic Server ユーザを BEA Tuxedo ドメインで認可済みのユーザとして定義します。
8. `ISL` コマンドの `-E` オプションを設定して、WebLogic Server レルムから伝播されたセキュリティ コンテキストを認識して利用するよう IIOP リスナ / ハンドラをコンフィグレーションします。`ISL` コマンドの `-E` オプションでは、プリンシパル名を指定する必要があります。プリンシパル名によって、WLEC 接続プールが BEA Tuxedo ドメインにログ インするために使用するプリンシパルが定義されます。プリンシパル名は、WLEC 接続プールを作成するときに [ユーザ名] 属性に定義した名前と一致しなければなりません。

WebLogic Server 環境と BEA Tuxedo CORBA 環境との間で証明書を使用するということは、WebLogic Server 環境から CORBA オブジェクトへの接続を確立するときに、新規の SSL ハンドシェークが開始されるということです。同じ SSL ネットワーク接続を使用して複数のクライアント リクエストをサポートするには、そうした処理を行うように証明書を次のように設定しなければなりません。

1. プリンシパルに対応するデジタル証明書を取得して、プライベート キーを BEA Tuxedo の `TUXDIR/udataobj/security/keys` ディレクトリに入れます。
2. `tpusradd` コマンドを実行して、プリンシパルを BEA Tuxedo のユーザとして定義します。
3. `-E` オプションを使用して `UBBCONFIG` ファイルの IIOP リスナ / ハンドラを定義して、プリンシパルが認証用に使用されることを示します。
4. WebLogic Server の Administration Console で WLEC 接続プールを作成するときに、[ユーザ名] 属性にプリンシパル名を定義します。
5. IIOP リスナ / ハンドラのデジタル証明書を取得します。
6. ISL コマンドの `SEC_PRINCIPAL_NAME` オプションでデジタル証明書を指定し、`-S` オプションを使用して、セキュア ポートを BEA Tuxedo ドメインと WebLogic Server セキュリティ レルムとの間で使用することを示します。

BEA Tuxedo CORBA アプリケーションでセキュリティをコンフィグレーションする方法については、『CORBA セキュリティ機能の概要』を参照してください。

BEA Tuxedo CORBA オブジェクトへのアクセス

この節では、WebLogic Server クライアントから BEA Tuxedo CORBA オブジェクトにアクセスするための手順について説明します。

手順 1. クライアント スタブを作成する

クライアント スタブは、BEA Tuxedo CORBA オブジェクトの操作に対するプログラミング インタフェースを提供します。クライアント スタブを作成するには、WebLogic Server クライアントからアクセスする BEA Tuxedo CORBA オブジェクト用の OMG IDL (Object Management Group Interface Definition Language) ファイルをコンパイルします。BEA Tuxedo CORBA オブジェクトのクライアント スタブを作成するには、BEA Tuxedo ソフトウェアに含まれている `idl` コンパイラを使います。

WebLogic Server の `CLASSPATH` 環境変数に、BEA Tuxedo CORBA オブジェクトのクライアント スタブが入っているディレクトリが含まれていることを確認します。

手順 2. Java パッケージをインポートする

以下の Java パッケージを WebLogic Server クライアントにインポートします。

- `org.omg.CORBA.*`
- `com.beasys.Tobj.*`
- `com.beasys.*`

手順 3. WebLogic Server クライアントを BEA Tuxedo ドメインに接続する

各 WLEC 接続プールは、関連付けられている BEA Tuxedo ドメインへのアクセスを可能にする `Tobj_Bootstrap` オブジェクトを持っています。WebLogic Enterprise Connectivity は、BootstrapFactory と呼ばれるオブジェクトを提供し、これが特定の BEA Tuxedo ドメインの `Tobj_Bootstrap` オブジェクトへのアクセスを提供します。BEA Tuxedo ドメインに接続するには、WebLogic Server クライアントに以下のコードを追加します。

```
Tobj_Bootstrap myBootstrap =
Tobj_BootstrapFactory.getClientContext("myPool");
```

各要素の説明は次のとおりです。

- `getClientContext()` メソッドは、*`myPool`* に関連付けられている `Tobj_Bootstrap` オブジェクトを返します。`getClientContext()` は、この名前のプールを見つけられない場合には `null` を返します。
- *`myPool`* は、目的の BEA Tuxedo ドメイン用の WLEC 接続プールの名前です。WLEC 接続プールは、Administration Console で定義する必要があります。

手順 4. BEA Tuxedo CORBA のオブジェクト参照を取得する

WebLogic Server クライアントで FactoryFinder を使用して、BEA Tuxedo CORBA オブジェクトへの参照を取得します。

1. 次のように FactoryFinder オブジェクトを取得します。

   ```
   org.omg.CORBA.Object myFFObject =
       myBootstrap.resolve_initial_references("FactoryFinder");
   FactoryFinder myFactFinder =
       FactoryFinderHelper.narrow(myFFObject);
   ```

   各要素の説明は次のとおりです。

   - `myBootstrap` は、BEA Tuxedo ドメインの `Tobj_Bootstrap` オブジェクトです。
   - `resolve_initial_references()` メソッドは、FactoryFinder オブジェクトのオブジェクト参照を返します。
   - `FactoryFinderHelper` は、`FactoryFinder` インタフェースに対する補助機能、特に `narrow()` メソッドを提供します。
   - `narrow()` メソッドは、オブジェクト参照をキャストして FactoryFinder オブジェクトを指すようにします。
2. 次のように、BEA Tuxedo CORBA オブジェクト用のファクトリを取得します。

```
org.omg.CORBA.Object myFactoryRef =
    myFactFinder.find_one_factory_by_id(myFactoryHelper.id());
myFactory =
    myFactoryHelper.narrow(myFactoryRef);
```

各要素の説明は次のとおりです。

- *myFactFinder* は、FactoryFinder オブジェクトです。
- `find_one_factory_by_id()` メソッドは、ID 番号に基づいてファクトリオブジェクト参照を検索して返します。
- *myFactoryHelper* は、*myFactory* インタフェースに対する補助機能、特に `narrow()` メソッドを提供します。
- `narrow()` メソッドは、オブジェクト参照をキャストしてオブジェクトファクトリを指すようにします。

3. BEA Tuxedo CORBA オブジェクトの `find()` メソッドを使用して、そのオブジェクトを取得します。たとえば、`Simple` というオブジェクトにアクセスする場合は、以下のコードを使用します。

   ```
   Simple mySimple = mySimpleFactory.find_simple();
   ```

   このファクトリは、`Simple` オブジェクトを検索するための `find_simple()` メソッドを提供します。

FactoryFinder オブジェクトについては、BEA Tuxedo のマニュアルの『CORBA プログラミング リファレンス』を参照してください。

手順 5. トランザクションを開始する（オプション）

BEA Tuxedo CORBA オブジェクトには、トランザクションのスコープ内でアクセスできます。以下のコード例では、TransactionCurrent オブジェクトを使用して、トランザクション スコープ内で BEA Tuxedo CORBA オブジェクトにアクセスします。BEA Tuxedo CORBA オブジェクトにアクセスするときには、UserTransaction オブジェクトも使用できます。

トランザクションを開始するには、WebLogic Server クライアントに以下のコードを追加します。

1. 次のように、TransactionCurrent オブジェクトを取得します。

```
org.omg.CORBA.Object myTCObject =
  myBootstrap.resolve_initial_references("TransactionCurrent");
CosTransactions.Current myTransaction =
    CosTransactions.CurrentHelper.narrow(myTCObject);
```

各要素の説明は次のとおりです。

- *myBootstrap* は、WLEC 接続プールに関連付けられている BEA Tuxedo ドメインの Bootstrap オブジェクトです。
- `resolve_initial_references()` メソッドは、TransactionCurrent オブジェクトのオブジェクト参照を返します。
- `CosTransactions.Current` インタフェースは、TransactionCurrent オブジェクトのインタフェースを定義します。これを使用すると、スレッドとトランザクションの関連付けを明示的に管理できるようになります。
- `CurrentHelper` インタフェースは、`Current` に対する補助機能、特に `narrow()` メソッドを提供します。
- `narrow()` メソッドは、オブジェクト参照をキャストして `CosTransactions` オブジェクトを指すようにします。

2. 次のようにトランザクションを開始します。

   ```
   myTransaction.begin();
   ```

   `begin()` メソッドは、トランザクション コンテキストを作成して、それを前の手順の *myTCObject* に関連付けます。*myTCObject* は *myBootstrap* に関連付けられており、*myBootstrap* は特定の WLEC 接続プールに関連付けられているので、*myTransaction* は特定の WLEC 接続プールに関連付けられることになります。

手順 6. BEA Tuxedo CORBA オブジェクトとその操作にアクセスする

WLEC 接続プールに関連付けられている BEA Tuxedo ドメインに属する BEA Tuxedo CORBA オブジェクトのメソッドを呼び出します。

トランザクション コンテキスト内のオブジェクトにアクセスする場合は、以下の TransactionCurrent メソッドを使用してそのトランザクション コンテキストの操作とクエリを行うことができます。

- suspend()
- resume()
- rollback_only()
- get_status()
- get_transaction_name()
- set_timeout()
- get_control()

手順 7. トランザクションを終了する（オプション）

トランザクション コンテキスト内の BEA Tuxedo CORBA オブジェクトにアクセスした場合、以下のいずれかの TransactionCurrent メソッドを使用してトランザクションを終了します。

- commit()
- rollback()

トランザクションに関する詳細情報

この節では、BEA Tuxedo 環境でのトランザクションについてさらに詳しく説明します。

マルチスレッド処理

BEA Tuxedo CORBA トランザクション サービスを使用すると、単一のアプリケーション プロセスの複数のスレッドから、別個のトランザクションを同時に開始できます。たとえば、2 つのスレッドが `CosTransactions.Current.begin()` と `UserTransaction.begin()` を同時に呼び出す場合、どちらのスレッドも、それぞれのトランザクションに対応する別個のトランザクション コンテキストを持ちます。

CORBA トランザクション サービスでは、単一のアプリケーションの複数のスレッドが、同じトランザクションを同時に扱うことはできません。CosTransactions を使用する場合、以下のようにトランザクションの中断と再開を行って、複数のスレッド内でそのトランザクションを使用します。

1. 第 1 のスレッドでは、`Current.suspend()` を呼び出して、トランザクションを中断して Control オブジェクトを取得します。
2. 第 2 のスレッドでは、その Control オブジェクトの `Current.resume()` を呼び出して、そのトランザクションを再開します。

中断されていないトランザクションをスレッドが再開しようとすると、BEA Tuxedo システムは `InvalidControl` 例外を送出します。

`UserTransaction` では、`suspend()` と `resume()` をサポートしていません。したがって、UserTransaction を使用するときには、複数のスレッドで 1 つのトランザクションを使うことはできません。

複数のアクティブな WLEC 接続プール

WebLogic Enterprise Connectivity コンポーネントでは、単一の WebLogic Server クライアントで複数の WLEC 接続プールを同時にアクティブにできます。`CosTransactions.Current.begin()` や `UserTransaction.begin()` を呼び出してトランザクション コンテキストを作成すると、BEA Tuxedo システムは、そのトランザクションを WLEC 接続プールに関連付けます。そのトランザクションのスコープ内で行われる呼び出しはすべて、そのトランザクションの WLEC 接続プールに関連付けられているドメイン内に存在するオブジェクトに対するものでなければなりません。

1 つのトランザクションが複数の BEA Tuxedo ドメインにまたがることはできません。別のドメイン内のオブジェクトを呼び出そうとすると、BEA Tuxedo システムは `INVALID_TRANSACTION` 例外を送出します。

アクティブなトランザクションと接続の関係

WLEC クライアントがトランザクションを開始または再開すると、WLEC 接続プールのインフラストラクチャは、そのトランザクションのコンテキスト内で送信される要求のための接続を確保します。WebLogic Enterprise Connectivity は、そのトランザクション コンテキスト外の要求を送信するためにこの接続を使うことはありません。WebLogic Enterprise Connectivity は、そのトランザクションがコミット、ロールバック、もしくは中断されるまでその接続を確保します。

注意： `suspend()` と `resume()` を使用できるのは CosTransactions だけです。

同時にアクティブにできるトランザクションの数は、プール内で使用できる接続の数によって決まります。スレッドがトランザクションを開始または再開するときに接続が利用できない場合、WebLogic Enterprise Connectivity は `NO_RESOURCES` 例外を送出します。

トランザクションの管理

各スレッドは、独自のトランザクション コンテキストを備えています。スレッドがトランザクションを開始または再開すると、そのトランザクションはコミット、ロールバック、または中断されるまでアクティブになります。その後に WebLogic Server クライアントを呼び出しても、それらが同じスレッド内で実行される保証はありません。したがって、スレッドでは、トランザクションをコミット、ロールバック、または中断してから、WebLogic Server クライアントの呼び出しを終了することが大切です。

注意： `suspend()` と `resume()` を使用できるのは CosTransactions だけです。

必要であれば、次の手順で、WebLogic Server クライアントへの複数の呼び出しで 1 つのトランザクションを使用できます。

1. それぞれの呼び出しの終わりに、トランザクションを中断します。
2. その次の呼び出しで、トランザクションを再開します。

WLS クライアントへの次の呼び出しを行わないうちにトランザクションがタイムアウトする可能性があるので、この方法を使用するときには注意してください。