BEA WebLogic Server™

WebLogic Tuxedo Connector
管理ガイド

BEA WebLogic Server バージョン 6.1
マニュアルの日付：2003 年 4 月 24 日

著作権

限定的権利条項

商標または登録商標

WebLogic Tuxedo Connector 管理ガイド

マニュアルの日付	ソフトウェアのバージョン
2002 年 6 月 24 日	BEA WebLogic Server 6.1

目次

対象読者 xii
e-docs Web サイト xii
このマニュアルの印刷方法 xiii
関連情報 xiii
サポート情報 xiv
表記規則 xv

1. WebLogic Tuxedo Connector の概要

WebLogic Tuxedo Connector の概要 1-2
主要な機能と管理機能 1-2
確認済みの制限 1-3
WebLogic Tuxedo Connector と Jolt の相違点 1-4
マニュアル 1-4
プラットフォームのサポート 1-5
ライセンス 1-5

2. WebLogic Tuxedo Connector のコンフィグレーション

環境の変更と考慮事項の概要 2-1
Tuxedo の変更 2-1
WebLogic Server の変更 2-2
WebLogic Server のスレッド 2-2
アプリケーション用の WebLogic Tuxedo Connector のコンフィグレーション 2-3
WebLogic Tuxedo Connector XML コンフィグレーション ファイルの作成 2-4
コンフィグレーション ファイル コンポーネント 2-4
wtc_config.dtd のローカル コピーの使用 2-6
XML ファイルの有効性の検証 2-7
WebLogic Server 環境の設定 2-7
WebLogic Tuxedo Connector のスタートアップ クラスの作成 2-8
WebLogic Tuxedo Connector のシャットダウン クラスの作成 2-9

アプリケーション サーバの再起動.....2-10
インストールの検証.....2-10
非 ASCII コードセットに対する WebLogic Tuxedo Connector のコンフィグレーション.....2-12

3. BDMCONFIG のコンフィグレーション

ドメイン間の通信のコンフィグレーション.....3-2
起動時に接続を要求する方法（ON_STARTUP）.....3-2
RetryInterval のコンフィグレーション方法.....3-3
MaxRetries のコンフィグレーション方法.....3-3
クライアント要求による接続のリクエスト方法（ON_DEMAND）.....3-4
受信時接続の受け付け（INCOMING_ONLY）.....3-5
LOCAL 接続ポリシーの使用方法.....3-5
ConnectionPolicy の動的ステータスへの影響.....3-6
ドメインレベルのフェイルオーバとフェイルバックのコンフィグレーション 3-6
ドメインレベルのフェイルオーバおよびフェイルバックを使用するための要件.....3-7
フェイルオーバをサポートするためのドメインのコンフィグレーション 3-7
フェイルバックをサポートするためのドメインのコンフィグレーション.....3-8
リモート ドメインの認証.....3-9
T_DM_PASSWORD 要素の設定.....3-10
解説.....3-10
例.....3-11
LocalPasswords.....3-11
RemotePasswords.....3-11
AppPasswords.....3-11
ユーザ認証.....3-11
サーバのセキュリティ要件.....3-12
クライアントのセキュリティ要件.....3-12
Tuxedo と WebLogic Server の間でセキュリティを提供するように WebLogic Tuxedo Connector をコンフィグレーションする方法.....3-13
T_DM_LOCAL_TDOMAIN のコンフィグレーション.....3-13
T_DM_LOCAL_TDOMAIN のコンフィグレーション例.....3-14

Tuxedo *DM_LOCAL_DOMAINS のコンフィグレーション例...3-14
T_DM_REMOTE_TDOMAIN のコンフィグレーション.....3-15
T_DM_LOCAL_TDOMAIN のコンフィグレーション例.....3-15
Tuxedo *DM_LOCAL_DOMAINS のコンフィグレーション例...3-16
Simpapp サンプルおよび Simpserv サンプルの ACL ポリシー例.....3-17
リンクレベルの暗号化.....3-23

4. CORBA アプリケーションの管理

CORBA サービス アプリケーション用 WebLogic Tuxedo Connector のコンフィグレーション方法.....4-1
XML コンフィグレーション ファイルの例.....4-2
着信 RMI/IIOP の WebLogic Tuxedo Connector を管理およびコンフィグレーションする方法.....4-3
XML コンフィグレーション ファイルのコンフィグレーション 4-4
Tuxedo アプリケーション環境の管理.....4-4
発信 RMI/IIOP 用 WebLogic Tuxedo Connector のコンフィグレーション方法 4-5
XML コンフィグレーション ファイルのサンプル 4-6

5. tBridge のコンフィグレーション

tBridge の概要 5-2
tBridge が JMS と Tuxedo を接続する仕組み.....5-3
tBridge が Tuxedo を JMS に接続する仕組み.....5-4
tBridge の制限 5-4
tBridge 用 WebLogic Tuxedo コンフィグレーション XML ファイル コンフィグレーション.....5-5
tBridge の起動 5-5
エラーのロギング 5-6
tBridge の接続性.....5-6
接続タイプのコンフィグレーションの例 5-6
JmsQ2TuxQ コンフィグレーションの例 5-7
TuxQ2JmsQ コンフィグレーションの例 5-8
JmsQ2TuxS コンフィグレーションの例 5-10
優先度のマッピング.....5-11
エラー キュー 5-15
wlsServerErrorDestination.....5-15

サポートされていないメッセージ タイプ 5-16
tuxErrorQueue 5-16
制限 5-16

6. WebLogic Tuxedo Connector での FML の使用

FML の概要 6-1
WebLogic Tuxedo Connector FML API 6-2
FML フィールド テーブルの管理 6-2
mkfldclass32 クラスに対する DynRdHdr プロパティの使い方 6-4
tBridge XML/FML32 変換 6-6
FLAT 変換の使い方 6-6
NO 変換の使い方 6-7
FML32 の考慮事項 6-7

7. WebLogic Process Integrator と Tuxedo アプリケーションの接続

WebLogic Process Integrator と Tuxedo の同期接続 7-2
ビジネス オペレーションの定義 7-2
eLink アダプタの呼び出し 7-2
例外ハンドラの定義 7-3
WebLogic Process Integrator と Tuxedo の同期非ブロッキング接続 7-3
WebLogic Process Integrator と Tuxedo の非同期接続 7-3
非同期 Tuxedo /Q と WebLogic Process Integrator の接続 7-4
Tuxedo と WebLogic Process Integrator の双方向非同期接続 7-5

8. WebLogic Tuxedo コネクタのトラブルシューティング

WebLogic Tuxedo コネクタのモニタ 8-1
トレース レベルの設定 8-1
コンソールの設定 8-2
よくある質問 8-3
コンフィグレーション ファイルが見つからない 8-3
EJB デプロイメント メッセージ 8-4
接続の問題 8-4

9. WebLogic Tuxedo Connector XML コンフィグレーショ

ン ファイル

XML コンフィグレーション ファイルの作成 9-1
XML コンフィグレーション ファイルの例 9-2
XML ファイルの有効性の検証 9-3
要素の階層構造図 9-4

10. wtc_config.dtd

wtc_config.dtd 10-1

11. wtc_config.dtd の要素および属性

WTC_CONFIG 11-1
BDMCONFIG 11-1
T_DM_LOCAL_TDOMAIN 11-2
WlsClusterName 11-2
AccessPointId 11-2
Type 11-3
Security 11-3
ConnectionPolicy 11-3
RetryInterval 11-4
MaxRetries 11-5
ConnPrincipalName 11-5
NWAddr 11-5
CmpLimit 11-6
MinEncryptBits 11-6
MaxEncryptBits 11-6
Interoperate 11-7
BlockTime 11-7
T_DM_REMOTE_TDOMAIN 11-8
LocalAccessPoint 11-8
AclPolicy 11-8
CredentialPolicy 11-9
FederationURL 11-9
FederationName 11-9
TpUsrFile 11-10
T_DM_EXPORT 11-10

RemoteName....11-10
EJBName....11-10
T_DM_IMPORT....11-11
TranTime....11-11
T_DM_PASSWORD....11-12
LocalPassword....11-12
RemotePassword....11-13
T_DM_RESOURCES....11-13
FieldTables....11-13
FldTblClass....11-14
AppPassword....11-14
tBridge....11-14
direction....11-15
fromto....11-15
redirect....11-15
source....11-15
target....11-15
AccessPoint....11-15
Qspace....11-15
Name....11-16
ReplyQ....11-16
metadataFile....11-16
translateFML....11-16
transactional....11-17
timeout....11-17
idleTime....11-17
retries....11-18
retryDelay....11-18
wlsErrorDestination....11-18
tuxErrorQueue....11-18
defaultRelativeBirthtime....11-19
defaultRelativeExpiration....11-19
expirationAdjustment....11-19
priorityMapping....11-20
JmstoTux....11-20

TuxtoJms....11-20
pMap....11-20
range....11-21
value....11-21
deliveryModeOverride....11-21
defaultReplyDeliveryMode....11-21
userID....11-22
allowNonStandardTypes....11-22
jndiFactory....11-22
jmsFactory....11-22
tuxFactory....11-23

このマニュアルの内容

このマニュアルでは、BEA WebLogic Tuxedo コネクタ ™ アプリケーションの開発環境を紹介します。WebLogic Server と Tuxedo との間で相互運用する WebLogic Tuxedo Connector のコンフィグレーションおよび管理方法について説明します。

このマニュアルの構成は次のとおりです。

- 第 1 章「WebLogic Tuxedo Connector の概要」では、WebLogic Tuxedo Connector の概要を説明します。
- 第 2 章「WebLogic Tuxedo Connector のコンフィグレーション」では、WebLogic Tuxedo Connector のコンフィグレーションの方法について説明します。
- 第 3 章「BDMCONFIG のコンフィグレーション」では、BDMCONFIG のコンフィグレーション情報について説明します。
- 第 4 章「CORBA アプリケーションの管理」では、CORBA アプリケーションの管理方法について説明します。
- 第 5 章「tBridge のコンフィグレーション」では、tBridge 機能とコンフィグレーションについて説明します。
- 第 6 章「WebLogic Tuxedo Connector での FML の使用」では、フィールド操作言語（FML）および WebLogic Tuxedo Connector が FML を使用する方法について説明します。
- 第 7 章「WebLogic Process Integrator と Tuxedo アプリケーションの接続」では、Tuxedo アプリケーションをビジネス ワークフローで動作させるために、WebLogic Process Integrator ユーザに必要なインフラストラクチャを説明します。
- 第 8 章「WebLogic Tuxedo コネクタのトラブルシューティング」では、WebLogic Tuxedo Connector のトラブルシューティング情報について説明します。

- 第 9 章「WebLogic Tuxedo Connector XML コンフィグレーション ファイル」では、WebLogic Tuxedo Connector XML コンフィグレーション ファイルの作成方法および要素の階層構造図について説明します。
- 第 10 章「wtc_config.dtd」では、WebLogic Tuxedo Connector XML コンフィグレーション ファイルを作成するために使用する要素と属性の構造について説明します。
- 第 11 章「wtc_config.dtd の要素および属性」では、`wtc_config.dtd` に含まれる要素および属性のリファレンス情報について説明します。

対象読者

このマニュアルは、WebLogic Server と Tuxedo 環境で相互に運用される分散 Java アプリケーションを構築するシステム管理者およびアプリケーション開発者を対象としています。WebLogic Server、Tuxedo、XML、および Java プログラミング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。

e-docs Web サイト

BEA 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を一度に 1 章ずつ印刷できます。

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）から無料で入手できます。

関連情報

BEA の Web サイトでは、WebLogic Server および Tuxedo のすべてのドキュメントを提供しています。

Java および Java CORBA アプリケーションの詳細については、以下を参照してください。

- OMG Web サイト（http://www.omg.org/）
- Sun Microsystems, Inc. の Java サイト（http://java.sun.com/）

サポート情報

BEA 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@bea.com までお送りください。寄せられた意見については、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名とバージョン名、およびマニュアルのタイトルと作成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
〔Ctrl〕+ 〔Tab〕	同時に押すキーを示す。
斜体	強調または本のタイトルを示す。

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、Java クラス、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 :
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

斜体の等幅テキスト	コード内の変数を示す。 例 : `String *CustomerName*;`
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 BEA_HOME OR
{ }	構文内の複数の選択肢を示す。

表記法	適用
[ ]	構文内の任意指定の項目を示す。 例 : `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
|	構文の中で相互に排他的な選択肢を区切る。 例 : `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`
...	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。
. . .	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。

1 WebLogic Tuxedo Connector の概要

以下の節では、サービス パック 2 を適用した WebLogic Server Release 6.1 向けの WebLogic Tuxedo Connector の概念と機能についてまとめてあります。

- WebLogic Tuxedo Connector の概要
- 主要な機能と管理機能
- 確認済みの制限
- WebLogic Tuxedo Connector と Jolt の相違点
- マニュアル
- プラットフォームのサポート
- ライセンス

WebLogic Tuxedo Connector の概要

WebLogic Tuxedo Connector は、WebLogic Server アプリケーションと Tuxedo サービス間の相互運用性を提供します。コネクタでは XML コンフィグレーション ファイルを使用します。このファイルによって、WebLogic Server クライアントが Tuxedo サービスを呼び出し、Tuxedo クライアントがサービス要求に応じて WebLogic Server エンタープライズ JavaBean（EJB）を呼び出すことができます。

主要な機能と管理機能

WebLogic-Tuxedo Connector は、Tuxedo ATMI とほぼ同じ Java Application-to-Transaction Monitor Interface（JATMI）を使用することで、WebLogic Server と Tuxedo を相互運用するアプリケーションの開発とサポートを可能にします。WebLogic-Tuxedo Connector の tBridge 機能には、高度な Tuxedo /Q および JMS メッセージ サービスが用意されています。

WebLogic-Tuxedo Connector によって、以下のような双方向の相互運用性が実現します。

- Tuxedo アプリケーションからの WebLogic Server アプリケーションの呼び出し、およびその逆、または Tuxedo アプリケーションからの EJB の呼び出し
- 既存の Tuxedo 環境への WebLogic Server アプリケーションの統合
- トランザクション サポート
- CORBA Java と CORBA C++ サーバ アプリケーション間の相互運用性を提供する機能
- WebLogic Process Integrator を使用した、eLink 1.2 アダプタなどの Tuxedo ATMI サービス全体にわたるワークフローの管理
- WebLogic Server と Tuxedo の間の複数接続の定義

WebLogic Tuxedo Connector の主要な管理機能を次に示します。

- 簡単な実装。WebLogic Tuxedo Connector では、既存の Tuxedo アプリケーション コードを修正する必要がありません。
  - 既存の Tuxedo クライアントは、WebLogic Tuxedo Connector を使用して WebLogic Server EJB を呼び出します。
  - 新規または修正された WebLogic Server クライアントは、WebLogic Tuxedo Connector を使用して Tuxedo サービスを呼び出します。
- XML コンフィグレーション ファイルを使用した管理。
- ドメインおよび ACL セキュリティを含む、双方向のセキュリティ伝播。
- ドメインレベルのフェイルオーバおよびフォールバック。
- Tuxedo /Q および JMS が提供する高度なメッセージング サービス。
- eLink を使用した、メインフレームと他の従来のアプリケーションとの相互運用。

確認済みの制限

WebLogic Tuxedo Connector には以下の制限があります。

- WebLogic-Tuxedo Connector ゲートウェイに対するコンフィグレーションの動的な変更がサポートされていません。
- バッファ表示機能がサポートされていません。
- Tuxedo アプリケーションからの着信 RMI/IIOP トランザクションをサポートしていません。
- クラスタをサポートしていません。クラスタ環境で使用できる WebLogic Tuxedo Connector のインスタンスは、1 つだけです。
- VMS、AS/400、OS/390 の各プラットフォーム上で稼働する Tuxedo 6.5 は、サポートしていません。
- IBM AIX プラットフォームで稼働する WebLogic Server からの発信 CORBA アプリケーションはサポートされていません。WebLogic Server がソケット作成用に使用している拡張機能は Sun 固有のものであり、IBM JDK との互換性がありません。

WebLogic Tuxedo Connector と Jolt の相違点

WebLogic Tuxedo Connector は、Jolt の代替機能ではありません。WebLogic Tuxedo Connector は、次の点で Jolt と異なっています。

- WebLogic Tuxedo Connector は、Jolt と似てはいるが異なる API を提供します。
- Jolt は汎用的な Java クライアントおよび他の Web サーバ アプリケーションの開発が可能ですが、WebLogic Tuxedo Connector にはできません。
- Jolt は、統合された WebLogic Server-Tuxedo トランザクションのメカニズムを提供しません。

汎用的な Java クライアントまたは他の Web サーバ アプリケーションが必要で、WebLogic Server がソリューションの一部ではない場合は、ソリューションとして WebLogic Tuxedo Connector ではなく Jolt を使用してください。

マニュアル

WebLogic Tuxedo Connector のマニュアルは、次の場所からダウンロードできます。

- BEA の Web サイト。BEA のホーム ページ（http://www.beasys.co.jp）で [製品のドキュメント] をクリックしてください。
- WebLogic Server の「e-docs」製品マニュアル ページ（http://edocs.beasys.co.jp/e-docs/）を直接参照してください。WebLogic Server 6.1 マニュアル センターのリンクに従ってください。

プラットフォームのサポート

プラットフォームのサポートに関する最も正確な最新情報については、「プラットフォーム サポート」ページを参照してください。

ライセンス

この節では、WebLogic Tuxedo Connector のライセンス情報について説明します。

- 暗号を使用しない場合は、コネクタを使用するためのライセンス要件はありません。
- 暗号を使用する場合は、適切な Tuxedo LLE ライセンスおよび適切な WebLogic Server ライセンスが必要です。

2 WebLogic Tuxedo Connector のコンフィグレーション

この章では、WebLogic Tuxedo Connector のコンフィグレーションの方法について説明します。

- 環境の変更と考慮事項の概要
- アプリケーション用の WebLogic Tuxedo Connector のコンフィグレーション
- 非 ASCII コードセットに対する WebLogic Tuxedo Connector のコンフィグレーション

環境の変更と考慮事項の概要

この節では、WebLogic Tuxedo Connector を使用する前に、Tuxedo および WebLogic Server 環境に行う必要のある変更の概要を示します。

Tuxedo の変更

注意： Tuxedo ドメインの詳細については、『BEA Tuxedo Domains コンポーネント』を参照してください。

Tuxedo ユーザは、次のような環境の変更を行う必要があります。

- 既存の Tuxedo アプリケーションがすでに Tuxedo /T DOMAINS を使用している場合、WebLogic Tuxedo Connector インスタンス化への接続ごとに、ドメイン コンフィグレーション ファイルへ新しいドメインを追加する必要があります。
- 既存の Tuxedo アプリケーションがドメインを使用していない場合、アプリケーションの *TUXCONFIG* にドメイン サーバを追加する必要があります。

WebLogic Tuxedo Connector インスタンス化に対応する Tuxedo /T Domain エントリを使用して、新しい *DMCONFIG* を作成する必要があります。

- WebLogic Tuxedo Connector は、Tuxedo ドメインで常にエンコーディングが有効になっていることを要求します。*DMCONFIG* ファイルで MTYPE を設定しないか、NULL に設定する必要があります。

WebLogic Server の変更

注意： WebLogic Tuxedo Connector クライアントまたはサーバの作成の詳細については、『WebLogic Tuxedo Connector ATMI プログラマーズ ガイド』を参照してください。

WebLogic Server ユーザは、次のような環境の変更を行う必要があります。

- Java クライアントまたはサーバを作成します。
- WebLogic Tuxedo Connector XML コンフィグレーション ファイルを作成します。
- WebLogic Server に WebLogic Tuxedo Connector をデプロイします。

WebLogic Server のスレッド

注意： WebLogic Server のパフォーマンスおよびチューニングの詳細については、『BEA WebLogic Server パフォーマンス チューニング ガイド』を参照してください。

ゲートウェイからサービスをディスパッチするときに使用できるクライアント スレッドの数によって、同時に実行できるサービスの数が制限されることがあります。このリリースの WebLogic Tuxedo Connector では、利用可能なスレッドの数を増やすための WebLogic Tuxedo Connector XML コンフィグレーション ファイル パラメータはありません。サービス EJB を呼び出すときは、適切なスレッド モデルを使用します。場合によっては、利用可能な WebLogic Server スレッドの数を大きな値に増やす必要があります。

アプリケーション用の WebLogic Tuxedo Connector のコンフィグレーション

注意： このリリースの WebLogic Tuxedo Connector では、静的なコンフィグレーションのみを提供します。WebLogic Tuxedo Connector の XML コンフィグレーション ファイルのパラメータを変更しなければならない場合は、WebLogic Server を再起動してその変更を有効にする必要があります。たとえば、ドメイン ネットワーク リンクの追加や削除、ネットワーク アドレスの変更、および新しいサービスのインポートやエクスポートは実行できません。

この節では、WebLogic Server アプリケーションと Tuxedo アプリケーションの相互運用を可能にするために WebLogic Tuxedo Connector をコンフィグレーションする方法について説明します。コネクタをコンフィグレーションするには、次の主要な手順に従います。

- WebLogic Tuxedo Connector XML コンフィグレーション ファイルの作成
- WebLogic Server 環境の設定
- WebLogic Tuxedo Connector のスタートアップ クラスの作成
- WebLogic Tuxedo Connector のシャットダウン クラスの作成
- アプリケーション サーバの再起動
- インストールの検証

WebLogic Tuxedo Connector XML コンフィグレーション ファイルの作成

WebLogic Tuxedo コネクタは XML コンフィグレーション ファイルを使用して、WebLogic Server を Tuxedo にリンクするために使用する Tuxedo /T DOMAINS 接続を記述します。

注意： WebLogic Tuxedo コネクタ XML コンフィグレーション ファイルは、JATMI インスタンスによってすべての WebLogic Server にインストールする必要があります。

- `vi` やメモ帳などのテキスト エディタを使用して、XML コンフィグレーション ファイルを作成します。
- 新しいコンフィグレーション ファイルを作成する最も効率的な方法は、`\examples\simpapp\DBMCONFIG.xml` などの WebLogic Tuxedo コネクタ配布キットのサンプル ディレクトリにあるサンプル コンフィグレーション ファイルの 1 つを修正することです。
- WebLogic Server アプリケーションでコンフィグレーション ファイルを保存します。

注意： コンフィグレーション ファイルの階層構造および要素の定義の詳細については、第 9 章「WebLogic Tuxedo Connector XML コンフィグレーション ファイル」を参照してください。

コンフィグレーション ファイル コンポーネント

WebLogic Tuxedo コネクタ コンフィグレーション ファイルは、次のセクションで構成されています。

- Version
- DOCTYPE
- WTC_CONFIG
- BDMCONFIG
- tBridge

Version

必須。使用する XML のバージョンを指定します。

例 :`< ?xml version="1.0"?>`

DOCTYPE

必須。DOCTYPE 宣言は、`wtc_config.dtd` の場所を提供します。WebLogic Server が起動されると、WebLogic Tuxedo Connector XML コンフィグレーション ファイルでドキュメント タイプ定義（DTD）のエラーがあるかどうかチェックされます。BEA では、次の場所に WebLogic Tuxedo Connector DTD を保持しています。`http://www.bea.com/servers/wls610/dtd/wtc_config.dtd`

例 :

```
<!DOCTYPE WTC_CONFIG SYSTEM
"http://www.bea.com/servers/wls610/dtd/wtc_config.dtd">
```

WTC_CONFIG

必須。`WTC_CONFIG` 要素は、コンフィグレーション ファイルのルートです。`WTC_CONFIG` には、次の 2 つの子要素があります。

- BDMCONFIG
- tBridge

BDMCONFIG

注意 : BDMCONFIG のコンフィグレーション方法の詳細については、第 3 章「BDMCONFIG のコンフィグレーション」を参照してください。

必須。BDMCONFIG は、WebLogic Server と Tuxedo 間でサービス リクエストを処理するために、WebLogic Tuxedo Connector が使用する接続情報およびセキュリティ プロトコルを記述します。これらのコンフィグレーション パラメータは、Tuxedo ドメイン間の通信に必要な相互運用属性に類似しています。
BDMCONFIG は、以下の子要素で構成されます。

- `T_DM_LOCAL_DOMAIN`: 他の（リモートの）ドメインで認識されるローカル ドメインのビューを提供します。
- `T_DM_REMOTE_TDOMAIN`: ローカル ドメインで認識されるリモート ドメインのビューを提供します。

- T_DM_EXPORT: ローカル ドメインによってエクスポートされるサービスに関する情報を提供します。
- T_DM_IMPORT: ローカル ドメインにインポートされる、リモート ドメイン上で利用可能なサービスに関する情報を提供します。
- T_DM_PASSWORD: タイプ TDOMAIN のアクセス ポイントの相互ドメイン認証に関するコンフィグレーション情報を提供します。
- T_DM_RESOURCES: ドメインのグローバル フィールド テーブル クラスとアプリケーション パスワードを指定するための情報を提供します。

tBridge

注意： tBridge のコンフィグレーション方法の詳細については、第 5 章「tBridge のコンフィグレーション」を参照してください。

省略可能。tBridge は、インポートされた Tuxedo サービスへの双方向 JMS インタフェースです。このセクションをコンフィグレーションすると、tBridge は WebLogic Server アプリケーション環境の一部として起動されます。

wtc_config.dtd のローカル コピーの使用

WebLogic Tuxedo Connector では、外部インターネット接続を使用して BEA DTD リポジトリの wtc_config.dtd ファイルを参照します。場合によっては、インストールのセキュリティ要件を満たすために wtc_config.dtd ファイルのローカル コピーを使用する必要があります。wtc_config.dtd のローカル コピーを作成する必要がある場合は、次の手順を行います。

- wtc_config.dtd のコピーをローカルのディレクトリに保存します。コピーは、必ずテキスト ファイルとして保存します。
- ローカル ディレクトリにある wtc_config.dtd の DOCTYPE 宣言を更新し、新しい絶対パスとファイル名を反映させます。必ず **http** を **file** に変更してください。

例：

```
<!DOCTYPE BDMCONFIG SYSTEM
"file:my_bea_directory\weblogic\wtc\gwt\wtc_config.dtd">
```

XML ファイルの有効性の検証

WTCValidateCF を使用して、コンフィグレーション ファイルの有効性を検証します。このユーティリティを使用すると、WebLogic Server を起動する前に XML コンフィグレーション ファイルの有効性を検証できます。

XML コンフィグレーション ファイルの有効性を検証するには、次のコマンドを入力します。

```
> java weblogic.wtc.gwt.WTCValidateCF your_XML_configuration_file
```

your_XML_configuration_file は、XML コンフィグレーション ファイルの名前です。

WebLogic Server 環境の設定

WebLogic Server アプリケーションの環境は、setEnv スクリプトを実行して設定します。

- NT/2000 ユーザの場合は、setEnv.cmd を実行します。
- UNIX ユーザの場合は、setEnv.sh を実行します。

環境を初めて設定する場合は、スクリプトの設定をチェックする必要があります。必要に応じて、次の手順に従ってアプリケーション環境の設定を修正します。

1. コマンドラインで、WebLogic Server アプリケーションの場所にディレクトリを変更します。
2. vi などのテキスト エディタを使用して、setEnv スクリプトを編集します。
   - NT/2000 ユーザの場合は、setEnv.cmd を編集します。
   - UNIX ユーザの場合は、setEnv.sh を編集します。
3. ファイルを保存します。

注意： setExamplesEnv ファイルは、配布キットで提供されている WebLogic Server サンプルの環境を設定するために使用します。

WebLogic Tuxedo Connector のスタートアップ クラスの作成

WebLogic Tuxedo Connector を使用するには、WebLogic Server スタートアップ クラスを作成し、コンフィグレーション ファイルの位置をスタートアップ クラスのプロパティとして割り当てる必要があります。ドメインのスタートアップ クラスを作成するには、次の手順に従います。

1. `startWebLogic` スクリプトを実行します。

   - NT/2000 ユーザの場合は、`startWebLogic.cmd` を実行します。

   - UNIX ユーザの場合は、`startWebLogic.sh` を実行します。

   WebLogic Server が起動されます。

2. WebLogic Server Console を起動します。
3. 必要に応じて、ドメイン ルートを右クリックし、[**他のドメインの作成または編集**] を選択します。左クリックして、リポジトリからドメインを選択します。
4. 左クリックして、[**デプロイメント**] ブランチを展開します。
5. [**起動と停止**] ブランチを右クリックします。
6. [**新しい StartupClass のコンフィグレーション**] を選択します。

   [コンフィグレーション] タブがアクティブになります。

7. [**名前**] を入力します。

   例 : MyWTCStartup Class

8. [**クラス名**] に **`weblogic.wtc.gwt.WTCStartup`** と入力します。
9. [**引数**] を入力します。2 つ以上の引数を使用する場合は、それらをカンマで区切ります。有効な引数は次のとおりです。

   BDMCONFIG: この必須引数は、WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの場所を指定します。

   例 : **`BDMCONFIG=.\config\mydomain\wtc_config.xml`**

- TraceLevel: この省略可能な引数は、使用するエラー追跡のレベルを指定します。

例 :

```
BDMCONFIG=.\config\mydomain\wtc_config.xml,TraceLevel=100000
```

- PasswordKey: この省略可能な引数は、ローカルおよびリモート ドメインのコンフィグレーション時にパスワードを暗号化するために使用するパスワード キーを指定します。

```
BDMCONFIG=.\config\mydomain\wtc_config.xml,PasswordKey=mykey
```

10. [**失敗したらサーバを起動しない**] をチェックします。
11. [**作成**] をクリックします。
12. [対象] タブを選択します。
13. [選択可] サーバ リストから、選択するサーバをクリックします。

    サーバが強調表示されます。
14. 右矢印ボタンをクリックします。

    選択したサーバが [選択済み] サーバ リストに表示されます。
15. [**適用**] をクリックします。

WebLogic Tuxedo Connector のシャットダウン クラスの作成

ドメインのシャットダウン クラスを作成するには、次の手順に従います。

1. [**起動と停止**] ブランチを右クリックします。
2. [**新しい ShutdownClass のコンフィグレーション**] を選択します。
3. [**名前**] を入力します。

   例 : MyWTCShutdown Class
4. [**クラス名**] に **weblogic.wtc.gwt.WTCShutdown** と入力します。
5. [**作成**] をクリックします。

アプリケーション サーバの再起動

コンソールへの変更をアクティブにするには、アプリケーション サーバをシャット ダウンして、再起動する必要があります。

インストールの検証

WebLogic Server config.xml ファイルをチェックします。WebLogic Tuxedo Connector コンフィグレーションは、WebLogic Server スタートアップ クラスおよびシャットダウン クラスに追加されます。

```
.
.
.
<StartupClass
     Arguments="BDMCONFIG=d:\bea\wlserver6.1\config\examples\bdmconfig.xml,Trac
eLevel=100000"
     ClassName="weblogic.wtc.gwt.WTCStartup" FailureIsFatal="true"
     Name="MyWTCStartup Class" Targets="examplesServer"/>
<ShutdownClass ClassName="weblogic.wtc.gwt.WTCShutdown" Name="MyWTCShutdown
Class"/>
.
.
.
```

TraceLevel をコンフィグレーションした場合は、Weblogic Server ログ ファイルで WebLogic Tuxedo Connector のエントリをチェックできます。**TraceLevel=100000** の場合は、以下のようなメッセージが表示されるはずです。

```
.
.
.
####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> <]/WTCStartup/crossCheck/50/true>
```

```
####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> <]/WTCStartup/extractInfo/80/true; DONE>

####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> <]/WTCStartup/loadFile/90/void; LOADED>

####<Jul 25, 2001 6:27:38 PM EDT> <Info> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180001> <Done Loading the XML config file.>

####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> <Done loading the XML Config File.>

####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> <Setting up federation points>

####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> <Federating [TDOM1] to [tgiop://TDOM1]>

####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-5> <> <> <180056> <[/WTCStartup/OatmialListener/run/>

####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> </WTCStartup/recover returned null>

####<Jul 25, 2001 6:27:38 PM EDT> <Debug> <WTC> <randyr-nt> <examplesServer>
<Thread-3> <system> <> <180056> <]/WTCStartup/startup/100/WTC started...>

####<Jul 25, 2001 6:27:38 PM EDT> <Info> <WebLogicServer> <randyr-nt>
<examplesServer> <Thread-3> <system> <> <000288> <weblogic.wtc.gwt.WTCStartup
reports: WTC started...>

.

.

.
```

インストールが正常に実行されなかった場合は、第 8 章「WebLogic Tuxedo コネクタのトラブルシューティング」を参照してください。

非 ASCII コードセットに対する WebLogic Tuxedo Connector のコンフィグレーション

注意： WebLogic Server プロパティの設定方法の詳細については、「WebLogic Server の起動と停止」を参照してください。

WebLogic Server と Tuxedo アプリケーションの間で非 ASCII（マルチバイト）文字列を転送するには、文字セットの変換を行うように WebLogic Tuxedo Connector をコンフィグレーションする必要があります。WebLogic Tuxedo Connector は、WebLogic Server のプロパティを使って、WebLogic Tuxedo Connector サービスで指定されているすべての Tuxedo リモート ドメインで使用されるエンコーディングを一致させます。複数のコードセットを同時に使用する必要がある場合は、独立した複数の WebLogic Server インスタンスで稼働する WebLogic Tuxedo Connector サービスが必要です。

文字セットの変換を有効にするには、サーバ起動スクリプトで `JAVA_OPTIONS` 変数を変更します。次に例を示します。

```
JAVA_OPTIONS=-Dweblogic.wtc.encoding=codesetname
```

codesetname は、サポートされていて、リモート Tuxedo ドメインが使用するコードセットの名前です。サポートされている基本コードセットおよび拡張コードセットのリストについては、「Supported Encodings」を参照してください。

リモート ドメインで使用されるエンコーディングを一致させるための正確なエンコーディング名を選択できない場合があります。そのような場合は、リモート ドメインに対応するエンコーディング名を選択する必要があります。

例：

- サポート対象エンコーディングのリストには、`EUC_JP` が含まれている。
- リモート ドメインは、`eucJP` を使用する Solaris オペレーティング システムによってサポートされている。

名前は厳密に一致してはいませんが、EUC_JP と eucJP は同等なエンコーディング セットであり、WebLogic Server とリモート ドメインの間の正確な文字列変換を行います。エンコーディングのプロパティを、EUC_JP に設定する必要があります。

```
JAVA_OPTIONS=-Dweblogic.wtc.encoding=EUC_JP
```

3 BDMCONFIG のコンフィグレーション

注意： WebLogic Tuxedo コネクタ XML コンフィグレーション ファイル、要素と属性、および `wtc_config.dtd` に関する詳細なリファレンス情報については、第 10 章「wtc_config.dtd」を参照してください。

WebLogic Tuxedo Connector XML コンフィグレーション ファイルの BDMCONFIG セクションは、接続の確立方法を記述し、WebLogic Server と Tuxedo 環境のドメイン間にセキュリティを提供します。XML コンフィグレーション ファイルは、Tuxedo ドメイン間の通信に必要な相互運用属性に類似したコンフィグレーション パラメータで構成されます。

WebLogic Tuxedo Connector がコンフィグレーションされている場合、それは WebLogic Server アプリケーション環境の一部として起動されます。WebLogic Tuxedo Connector が起動できないようなコンフィグレーション条件があるとエラーが発生し、WebLogic Server エラー ログに記録されます。

この章では、BDMCONFIG に関するコンフィグレーション情報について説明します。

- ドメイン間の通信のコンフィグレーション
- ConnectionPolicy の動的ステータスへの影響
- ドメインレベルのフェイルオーバとフェイルバックのコンフィグレーション
- リモート ドメインの認証
- ユーザ認証
- Tuxedo と WebLogic Server の間でセキュリティを提供するように WebLogic Tuxedo Connector をコンフィグレーションする方法
- Simpapp サンプルおよび Simpserv サンプルの ACL ポリシー例
- リンクレベルの暗号化

ドメイン間の通信のコンフィグレーション

注意： 動的ステータスの詳細については、3-6 ページの「ConnectionPolicy の動的ステータスへの影響」を参照してください。

ローカル ドメイン ゲートウェイがリモート ドメインとの接続を確立する条件を指定できるオプションがいくつかあります。これらの条件は、BDMCONFIG の T_DM_LOCAL_TDOMAIN セクションおよび T_DM_REMOTE_TDOMAIN セクションの ConnectionPolicy パラメータを使用して指定します。次の接続ポリシーのいずれかを選択できます。

- 起動時に接続を要求する方法（ON_STARTUP）
- クライアント要求による接続のリクエスト方法（ON_DEMAND）
- 受信時接続の受け付け（INCOMING_ONLY）
- LOCAL 接続ポリシーの使用方法

ON_STARTUP および INCOMING_ONLY の接続ポリシーでは、動的ステータスが呼び出されます。動的ステータスは、リモート サービスのステータスをチェックして、レポートします。

起動時に接続を要求する方法（ON_STARTUP）

ON_STARTUP のポリシーは、ゲートウェイ サーバの初期化時に、ドメイン ゲートウェイがリモート ドメイン アクセス ポイントを使用して接続を確立することを示します。この接続ポリシーは、RetryInterval パラメータと MaxRetries パラメータで指定された一定の間隔で、エラーとなった接続を再試行します。起動時に接続を要求するには、XML コンフィグレーション ファイルの BDMCONFIG セクションに次のエントリが必要です。

例：

```
<ConnectionPolicy>ON_STARTUP</ConnectionPolicy>
```

RetryInterval のコンフィグレーション方法

自動接続の試行の頻度は、接続を再試行する前にゲートウェイが待機する間隔（秒）を指定することによって制御できます。最小値は 0、デフォルト値は 60、そして最大値は 2147483647 です。次の例は、ゲートウェイが 30 秒待機してから接続を再試行することを指定します。

例 :

```
<RetryInterval>30</RetryInterval>
```

MaxRetries のコンフィグレーション方法

注意 : ConnectionPolicy が ON_STARTUP に設定されているときだけ使用してください。他の接続ポリシーの場合、再試行処理は無効になります。

`MaxRetries` パラメータに値を割り当てることによって、ドメイン ゲートウェイが終了するまでにリモート ドメイン アクセス ポイントへの接続を試行する回数を指定します。最小値は 0、デフォルトおよび最大値は 2147483647 です。

- `MaxRetries` を 0 に設定すると、自動接続の試行処理はオフになります。サーバは、リモート ゲートウェイに自動接続しません。

  例 :

  ```
  <MaxRetries>0</MaxRetries>
  ```

- `MaxRetries` に数値を設定すると、ゲートウェイは指定された回数だけ接続を再試行してから終了します。次の例では、サーバは接続の確立を 10 回試行してから終了します。

  例 :

  ```
  <MaxRetries>10</MaxRetries>
  ```

- `MaxRetries` を 2147483647 に設定すると、再試行処理は無期限でまたは接続が確立するまで繰り返されます。

  例 :

  ```
  <MaxRetries>2147483647</MaxRetries>
  ```

表 3-1 MaxRetries パラメータおよび RetryInterval パラメータの設定例

設定	処理
<ConnectionPolicy>ON_STARTUP</ConnectionPolicy> <RetryInterval>30</RetryInterval> <MaxRetries>3</MaxRetries>	ゲートウェイは、接続の確立を 30 秒間隔で 3 回試行してから終了する。
<ConnectionPolicy>ON_STARTUP</ConnectionPolicy> <MaxRetries>0</MaxRetries>	ゲートウェイは初期化時に接続の確立を試行するが、最初の試行がエラーとなった場合は再試行しない。
<ConnectionPolicy>ON_STARTUP</ConnectionPolicy> <RetryInterval>30</RetryInterval>	ゲートウェイは、接続が確立されるまで、30 秒おきに接続の確立を試行する。

クライアント要求による接続のリクエスト方法（ON_DEMAND）

注意： XML コンフィグレーション ファイルに ConnectionPolicy を指定しない場合、WebLogic Tuxedo Connector は ON_DEMAND の ConnectionPolicy を使用します。

ON_DEMAND の接続ポリシーは、リモート サービスに対するクライアント リクエスト、または管理接続コマンドのいずれかによって要求されたときのみに、接続が試行されることを示します。接続のクライアント要求によるリクエストを許可するには、XML コンフィグレーション ファイルの BDMCONFIG セクションで次のエントリを使用します。

例：

```
<ConnectionPolicy>ON_DEMAND</ConnectionPolicy>
```

受信時接続の受け付け（INCOMING_ONLY）

INCOMING_ONLY の接続ポリシーは、ドメイン ゲートウェイが起動時にリモート ドメインへの接続を確立しないことを示します。ドメイン ゲートウェイはリモート ドメイン アクセス ポイントからの受信時接続に使用可能で、リモート サービスはこのローカル ドメイン アクセス ポイントのドメイン ゲートウェイが受信時接続を受け付けたときに通知されます。接続のクライアント要求によるリクエストを許可するには、XML コンフィグレーション ファイルの BDMCONFIG セクションで次のエントリを使用します。

例：

```
<ConnectionPolicy>INCOMING_ONLY</ConnectionPolicy>
```

LOCAL 接続ポリシーの使用方法

注意： LOCAL の ConnectionPolicy は、ローカル ドメインでは無効です。

LOCAL の接続ポリシーは、リモート ドメインの接続ポリシーが明示的にローカル ドメインの ConnectionPolicy 属性値にデフォルト設定されることを示します。リモート ドメインの ConnectionPolicy が定義されていない場合、システムは関連するローカル ドメイン（LocalAccessPoint によって指定）で指定されている設定を使用します。リモート ドメインの接続ポリシーを LOCAL に設定するには、XML コンフィグレーション ファイルの T_DM_REMOTE_TDOMAIN セクションで次のエントリを使用します。

例：

```
<ConnectionPolicy>LOCAL</ConnectionPolicy>
```

ConnectionPolicy の動的ステータスへの影響

動的ステータスはゲートウェイ プロセス（GWTDOMAIN）の機能で、リモート サービスの可用性を決定します。WebLogic Tuxedo Connector コンフィグレーション ファイルで使用される接続ポリシーは、動的ステータス機能がサービスで使用可能かどうかを決定します。次の表は、`ConnectionPolicy` の動的ステータス機能に対する影響を示します。

ON_STARTUP	動的ステータスがオンになる。リモート ドメインからインポートされたサービスは、そのリモート ドメインへの接続が存在する間、通知される。
ON_DEMAND	動的ステータスがオフになる。リモート ドメインからインポートされたサービスは、常に通知される。
INCOMING_ONLY	動的ステータスがオンになる。リモート サービスは最初にサスペンドされる。ドメイン ゲートウェイは、リモート ドメインから受信時接続に対して使用可能である。リモート サービスは、ローカル ドメイン ゲートウェイが、受信時接続を受け付けたときに通知される。

ドメインレベルのフェイルオーバとフェイルバックのコンフィグレーション

注意： Tuxedo T/ Domain では、バックアップ リモート ドメインは 3 つに制限されています。WebLogic Tuxedo Connector には、サーバにコンフィグレーションできるバックアップ ドメインの数に制限はありません。

ドメインレベルのフェイルオーバは、プライマリ リモート ドメインでエラーが発生したとき、代替リモート ドメインにリクエストを転送するメカニズムです。このフェイルオーバは、ドメインが復元されると、プライマリ リモート ドメインにフェイルバックします。

このレベルのフェイルオーバおよびフェイルバックは、動的ステータスに依存します。ドメインレベルのフェイルオーバおよびフェイルバックを有効にするには、ドメインを ON_STARTUP または INCOMING_ONLY の CONNECTION_POLICY でコンフィグレーションする必要があります。

ドメインレベルのフェイルオーバおよびフェイルバックは、リモート ドメインへのネットワーク接続がある場合はリモート ドメインを使用可能として定義し、リモート ドメインへのネットワーク接続がない場合は使用不可能として定義します。

ドメインレベルのフェイルオーバおよびフェイルバックを使用するための要件

ドメインレベルのフェイルバックを使用するには、CONNECTION_POLICY パラメータの値に ON_STARTUP または INCOMING_ONLY を指定する必要があります。

ON_DEMAND の接続ポリシーは、リモート ドメインが常に使用可能であることを想定して動作するため、ドメインレベルのフェイルバックには適しません。接続ポリシーとして ON_STARTUP または INCOMING_ONLY を指定しないと、サーバは Tuxedo の RDOM パラメータを使用して指定した代替リモート ドメインにフェイルオーバできません。

注意： リモート ドメインはそれに対するネットワーク接続があれば「使用可能」であり、それに対するネットワーク接続がなければ「使用不可能」です。

フェイルオーバをサポートするためのドメインのコンフィグレーション

フェイルオーバをサポートするには、特定のサービスを実行する責任を持つリモート ドメインを指定する必要があります。具体的には、XML コンフィグレーション ファイルの T_DM_REMOTE_TDOMAIN セクションで、各リモート ドメインを指定する必要があります。

サービスが、TDOM1 および TDOM3 という 2 つのリモート ドメインで使用可能であると仮定します。この場合、XML コンフィグレーション ファイルの BDMCONFIG セクションに次のエントリを追加します。

```
<T_DM_REMOTE_TDOMAIN AccessPoint="TDOM1">
              <LocalAccessPoint>TDOM2</LocalAccessPoint>
              <AccessPointId>TDOM1</AccessPointId>
              <Type>TDOMAIN</Type>
              <NWAddr>//mydomain.acme.com:20305</NWAddr>
</T_DM_REMOTE_TDOMAIN>
<T_DM_REMOTE_TDOMAIN AccessPoint="TDOM3">
              <LocalAccessPoint>TDOM2</LocalAccessPoint>
              <AccessPointId>TDOM3</AccessPointId>
              <Type>TDOMAIN</Type>
              <NWAddr>//myotherdomain.com:50302</NWAddr>
</T_DM_REMOTE_TDOMAIN>
```

フェイルバックをサポートするためのドメインのコンフィグレーション

フェイルバックは、プライマリ リモート ドメインへのネットワーク接続が、次の理由で再確立された場合に発生します。

- 自動再試行（ON_STARTUP のみ）
- 受信時接続

リモート ドメインの認証

注意： Tuxedo 6.5 ユーザは、Interoperate パラメータを Yes に設定し、Security パラメータを NONE に設定する必要があります。セキュリティ機能が必要で、かつ WebLogic Tuxedo Connector を使用する場合は、Tuxedo 7.1 以上にアップグレードする必要があります。

ドメイン ゲートウェイは、リモート ドメインから要求された受信時接続、およびローカル ドメインから要求された送信時接続を認証するように設定できます。アプリケーション管理者は、リモート ドメインからの受信時接続のセキュリティを強化する必要のある場合を定義できます。特定のローカル ドメインによって使用されるセキュリティのレベルを指定するには、XML コンフィグレーション ファイルの T_DM_LOCAL_TDOMAIN セクションで SECURITY パラメータを設定します。パスワード セキュリティには次のような 3 つのレベルがあります。

- 「セキュリティなし」（NONE オプションを使用）― リモート ドメインからの受信時接続は認証されません。
- 「アプリケーション パスワード」（APP_PW オプションを使用）― リモート ドメインからの受信時接続は、XML コンフィグレーション ファイルの T_DM_PASSWORD 要素で定義されたアプリケーション パスワードを使用して認証されます。暗号化されたアプリケーション パスワードを作成するには、weblogic.wtc.gwt.genpasswd ユーティリティを使用します。
- 「リモート ドメイン パスワード」（DM_PW オプションを使用）― この機能は 2 つ以上のドメイン間のセキュリティを強化します。ローカル ドメインとリモート ドメイン間の接続は、XML コンフィグレーション ファイルの T_DM_PASSWORD 要素で定義されたパスワードのペアを使用して認証されます。暗号化されたリモート ドメイン パスワードを作成するには、weblogic.wtc.gwt.genpasswd ユーティリティを使用します。

XML コンフィグレーション ファイルの T_DM_LOCAL_TDOMAIN セクションの SECURITY パラメータは、Tuxedo ドメイン コンフィグレーション ファイルの *DM_LOCAL_DOMAINS セクションの SECURITY パラメータと一致していなければなりません。

- 認証が必要な場合、ローカル ドメインとリモート ドメイン間で接続が確立されるたびに認証が行われます。

- T_DM_LOCAL_TDOMAIN のセキュリティ タイプが *DM_LOCAL_DOMAINS のセキュリティ タイプと一致しない場合、またはパスワードが一致しない場合、接続は失敗します。

T_DM_PASSWORD 要素の設定

LocalPassword、RemotePassword、および AppPassword 要素の暗号パスワードを作成するには、`weblogic.wtc.gwt.genpasswd` を使用します。このユーティリティは、キーを使用して WebLogic Tuxedo Connector XML コンフィグレーション ファイルにコピーされるパスワードを暗号化します。結果は、有効な WebLogic Tuxedo Connector XML 要素です。

- XML コンフィグレーション ファイルは、クリア テキストのパスワードを格納しません。
- 起動時にキーへの WebLogic Tuxedo Connector アクセスを提供するため、StartUp クラスの `Argument` フィールドにキーが含まれていなければなりません。
  - キーを割り当てるには、`PasswordKey` パラメータを使用します。
  - `PasswordKey` 引数は、1 つのキー値にのみ割り当てられます。
  - XML コンフィグレーション ファイルに T_DM_PASSWORD エントリがない場合、`PasswordKey` 引数は無視されます。

解説

引数なしでユーティリティを呼び出し、コマンドライン オプションを表示します。

例 :

```
$ java weblogic.wtc.gwt.genpasswd

Usage: genpasswd Key <LocalPassword|RemotePassword|AppPassword>
<local|remote|application>
```

例

この節では、各パスワード要素タイプの例を紹介します。

LocalPasswords

次の例では、ローカル ドメインのパスワードとして「LocalPassword1」を暗号化するために *key1* を使用します。

```
$ java weblogic.wtc.gwt.genpasswd Key1 LocalPassword1 local
<LocalPassword IV="I#^Da0efo1">!djK*87$klbJJ</LocalPassword>
```

RemotePasswords

次の例では、リモート ドメインのパスワードとして「RemotePassword1」を暗号化するために *mykey* を使用します。

```
$ java weblogic.wtc.gwt.genpasswd mykey RemotePassword1 remote
<RemotePassword IV="Rq$45%%kK">McFrd3#f41Kl</RemotePassword>
```

AppPasswords

次の例では、アプリケーション パスワードとして「test123」を暗号化するために *key1* を使用します。

```
$ weblogic.wtc.gwt.genpasswd mykey test123 application
<AppPassword IV="gx8aSkAgLFg=">c98Y/P94HY3rCAVmkF=</AppPassword>
```

ユーザ認証

注意： Tuxedo 6.5 ユーザは、`Interoperate` パラメータを `Yes` に設定し、`Security` パラメータを `NONE` に設定する必要があります。`AclPolicy` 要素と `CredentialPolicy` 要素は、WebLogic Tuxedo Connector を使用し

て Tuxedo 6.5 と相互運用する場合は無視されます。セキュリティ機能が必要で、かつ WebLogic Tuxedo Connector を使用する場合は、Tuxedo 7.1 以上にアップグレードする必要があります。

アクセス制御リスト（ACL）は、サービスを実行できるリモート ドメインを制限することによって、ローカル ドメイン内でのローカル サービスへのアクセスを制限します。リモート ドメインからの着信ポリシーは、`AclPolicy` 要素を使用して指定します。リモート ドメインへの発信ポリシーは、`CredentialPolicy` 要素を使用して指定します。これによって、WebLogic Server と Tuxedo アプリケーションは同じセットのユーザを共有でき、ユーザはあるシステムから別のシステムへユーザの資格を伝播できます。

このパラメータの有効な値は次のとおりです。

- `LOCAL`: ドメイン ゲートウェイのセキュリティ トークンが渡されます。
- `GLOBAL`: ユーザのセキュリティ トークンが渡されます。

サーバのセキュリティ要件

- `RemoteAccessPoint` が `LOCAL` の `AclPolicy` で実行されている場合、その `RemoteAccessPoint` からのリクエストは、WebLogic Tuxedo Connector のインスタンス化の資格を持っている必要があります。
- RemoteAccessPoint が `GLOBAL` の `AclPolicy` で実行されている場合、その `RemoteAccessPoint` からのリクエストは、リクエストで受信したトークンからのユーザ ID を持っている必要があります。

クライアントのセキュリティ要件

- リモート ドメインが `LOCAL` に設定された `CredentialPolicy` で実行されている場合、Tuxedo へのリクエストはリモート ドメイン ゲートウェイの資格を持っています。
- リモート ドメインが `GLOBAL` に設定された `CredentialPolicy` で実行されている場合、WebLogic Tuxedo Connector は呼び出し側の ID を基にトークンを構築します。

Tuxedo と WebLogic Server の間でセキュリティを提供するように WebLogic Tuxedo Connector をコンフィグレーションする方法

注意： Tuxedo 6.5 は、セキュリティ マッピングをサポートするために必要なセキュリティ インフラストラクチャを持っていません。Tuxedo 6.5 ユーザは、`Interoperate` パラメータを `Yes` に設定し、`Security` パラメータを `NONE` に設定する必要があります。セキュリティ機能が必要で、かつ WebLogic Tuxedo Connector を使用する場合は、Tuxedo 7.1 以上にアップグレードする必要があります。

Tuxedo アプリケーションと WebLogic Server アプリケーションの間でセキュリティを提供するように WebLogic Tuxedo Connector をコンフィグレーションするには、以下の手順を行います。

- T_DM_LOCAL_TDOMAIN のコンフィグレーション
- T_DM_REMOTE_TDOMAIN のコンフィグレーション

T_DM_LOCAL_TDOMAIN のコンフィグレーション

XML コンフィグレーション ファイルの T_DM_LOCAL_TDOMAIN セクションの `SECURITY` パラメータを、Tuxedo ドメイン コンフィグレーション ファイルの *DM_LOCAL_DOMAINS セクションの SECURITY パラメータと一致するように設定します。

T_DM_LOCAL_TDOMAIN のコンフィグレーション例

```
.
.
.
<T_DM_LOCAL_TDOMAIN AccessPoint="Your weblogic domain">
     <Type>TDOMAIN</Type>
     <Security>DM_PW</Security>
     <ConnectionPolicy>ON_DEMAND</ConnectionPolicy>
     <ConnPrincipalName>wldom1</ConnPrincipalName>
     <NWAddr>//DKUMAR:3045</NWAddr>
     <Interoperate>Yes</Interoperate>
</T_DM_LOCAL_TDOMAIN>
.
.
.
```

Tuxedo *DM_LOCAL_DOMAINS のコンフィグレーション例

```
.
.
.
*DM_LOCAL_DOMAINS
domain1      DOMAINID = "domain1"
      GWGRP = "GWGRP"
      CONNECTION_POLICY=ON_DEMAND
      DMTLOGDEV = "/nfs/home1/dkumar/lcsol5/tmp.100/DMTLOG"
      DMTLOGNAME = "DMTLG1"
      SECURITY=DM_PW
      BLOCKTIME=10
.
.
.
```

T_DM_REMOTE_TDOMAIN のコンフィグレーション

T_DM_REMOTE_TDOMAIN をコンフィグレーションすると、着信および発信のアクセス制御リスト（ACL）ポリシーを確立できます。

WebLogic Server 環境を準備するには、次の手順を実行します。

1. XML コンフィグレーション ファイルの T_DM_REMOTE_TDOMAIN セクションに、AclPolicy 要素を追加します。

   例 :

   ```
   <AclPolicy>GLOBAL</AclPolicy>
   ```

2. XML コンフィグレーション ファイルの T_DM_REMOTE_TDOMAIN セクションに、CredentialPolicy 要素を追加します。

   例 :

   ```
   <CredentialPolicy>GLOBAL</CredentialPolicy>
   ```

3. `CredentialPolicy` が `GLOBAL` に設定されている場合は、WebLogic Server 環境で Tuxedo `tpusr` ファイルのコピーが必要になります。XML コンフィグレーション ファイルに `TpUserFile` 要素を追加して WebLogic Tuxedo Connector をコンフィグレーションするには、以下の手順を行います。

   a. TUXEDO から WebLogic Server アプリケーション環境に `tpusr` ファイルをコピーするか、独自の `tpusr` ファイルを生成します。

   b. XML コンフィグレーション ファイルの T_DM_REMOTE_TDOMAIN セクションに、TpUserFile 要素を追加します。

      例 :

      ```
      <TpUsrFile>pathname_filename_of_tpusr_file</TpUsrFile>
      ```

注意 : Tuxedo tpusr ファイル作成の詳細については、「How to Enable User-Level Authentication Security」を参照してください。

T_DM_LOCAL_TDOMAIN のコンフィグレーション例

.

```
.
.
<T_DM_REMOTE_TDOMAIN AccessPoint="Your Tuxedo domain">
     <LocalAccessPoint>wldom1</LocalAccessPoint>
     <AccessPointId>domain1</AccessPointId>
     <Type>TDOMAIN</Type>
    <AclPolicy>LOCAL</AclPolicy>
     <ConnPrincipalName>domain1</ConnPrincipalName>
    <CredentialPolicy>LOCAL</CredentialPolicy>

    <TpUsrFile>C:\runs\tmp.100\config\wldom1\tpusr</TpUsrFile>
     <NWAddr>//lcsol5:3500</NWAddr>
</T_DM_REMOTE_TDOMAIN>
.
.
.
```

Tuxedo *DM_LOCAL_DOMAINS のコンフィグレーション例

```
.
.
.
*DM_REMOTE_DOMAINS
     wldom1 DOMAINID = "wldom1"
     ACCESSPOINTID="wldom1"
    ACL_POLICY="LOCAL"
    CREDENTIAL_POLICY="LOCAL"
.
.
.
```

Simpapp サンプルおよび Simpserv サンプルの ACL ポリシー例

この節では、simpapp および simpserv の例を使用した、ACL 制御の設定方法の例を示します。

- John と Bob だけが Toupper へのアクセスを持ちます。
- Dan と John だけが Tolower へのアクセスを持ちます。
- Toupper は、リモート Tuxedo サービス TOUPPER にアクセスするために使用します。
- Tolower は、リモート Tuxedo ユーザに実際のサービスを提供します。

ACL 制御を確立するには、次の手順に従います。

1. WebLogic Server Console を使用して、ユーザ John、Bob、および Dan を WebLogic Security に追加します。
2. Tuxedo TOUPPER サービスの security-role および method_permission 要素を追加するために、ejb-jar.xml を修正します。太字の部分は、セキュリティ実装をサポートするための変更箇所を示します。

コード リスト 3-1 TOUPPER ejb-jar.xml セキュリティ サンプル コード

```
<?xml version="1.0"?>
<!--
Copyright (c) 2000 BEA Systems, Inc. All rights reserved

THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc. The copyright
notice above does not evidence any actual or intended publication of such source
code.

-->

<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
2.0//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_2_0.dtd'>

<ejb-jar>
     <enterprise-beans>
```

```
          <session>
               <ejb-name>Toupper</ejb-name>
     <home>weblogic.wtc.examples.simpapp.ToupperHome</home>
    <remote>weblogic.wtc.examples.simpapp.Toupper</remote>
     <ejb-class>weblogic.wtc.examples.simpapp.ToupperBean</ejb-class>
               <session-type>Stateful</session-type>
               <transaction-type>Container</transaction-type>
          </session>
     </enterprise-beans>
     <assembly-descriptor>
        <security-role>
              <role-name>dom2</role-name>
         </security-role>
         <method-permission>
              <role-name>dom2</role-name>
              <method>
                    <ejb-name>Toupper</ejb-name>
                    <method-name>Toupper</method-name>
              </method>
         </method-permission>
         <container-transaction>
              <method>
                   <ejb-name>Toupper</ejb-name>
                   <method-intf>Remote</method-intf>
                   <method-name>*</method-name>
              </method>
              <trans-attribute>Supports</trans-attribute>
          </container-transaction>
     </assembly-descriptor>
</ejb-jar>
```

3. Tuxedo TOUPPER サービスの security-role-assignment 要素を追加するために、Weblogic-ejb-jar.xml を修正します。太字の部分は、セキュリティ実装をサポートするための変更箇所を示します。

コード リスト 3-2 TOUPPER Weblogic-ejb-jar.xml セキュリティ サンプル コード

```
<?xml version="1.0"?>
<!--

Copyright (c) 2000 BEA Systems, Inc. All rights reserved
```

```
THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc. The copyright
notice above does not evidence any actual or intended publication of such source
code.

-->

<!DOCTYPE weblogic-ejb-jar PUBLIC '-//BEA Systems, Inc.//DTD WebLogic 6.0.0
EJB//EN' 'http://www.bea.com/servers/wls600/dtd/weblogic-ejb-jar.dtd'>

<weblogic-ejb-jar>
     <weblogic-enterprise-bean>
          <ejb-name>Toupper</ejb-name>
          <stateful-session-descriptor>
               <stateful-session-cache>
          <max-beans-in-cache>100</max-beans-in-cache>
               </stateful-session-cache>
          </stateful-session-descriptor>
          <jndi-name>tuxedo.services.ToupperHome</jndi-name>
     </weblogic-enterprise-bean>
     <security-role-assignment>
          <role-name>dom2</role-name>
          <principal-name>john</principal-name>
          <principal-name>bob</principal-name>
     </security-role-assignment>
</weblogic-ejb-jar>
```

4. Tolower サービスの security-role および method-permission 要素を追加するため、ejb-jar.xml を修正します。太字の部分は、セキュリティ実装をサポートするための変更箇所を示します。

コード リスト 3-3 Tolower ejb-jar.xml セキュリティ サンプル コード

```
<?xml version="1.0"?>
<!--

Copyright (c) 2000 BEA Systems, Inc. All rights reserved
THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc. The copyright
notice above does not evidence any actual or intended publication of such source
code.

-->
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
2.0//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_2_0.dtd'>
```

```
<ejb-jar>
     <enterprise-beans>
          <session>
               <ejb-name>Tolower</ejb-name>
            <home>weblogic.wtc.jatmi.TuxedoServiceHome</home>
           <remote>weblogic.wtc.jatmi.TuxedoService</remote>
            <ejb-class>weblogic.wtc.examples.simpserv.TolowerBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
          </session>
     </enterprise-beans>

     <assembly-descriptor>
        <security-role>
               <role-name>rdom2</role-name>
         </security-role>
         <method-permission>
               <role-name>rdom2</role-name>
               <method>
                    <ejb-name>Tolower</ejb-name>
                    <method-name>service</method-name>
               </method>
         </method-permission>
         <container-transaction>
               <method>
                    <ejb-name>Tolower</ejb-name>
                    <method-intf>Remote</method-intf>
                    <method-name>*</method-name>
               </method>
               <trans-attribute>Supports</trans-attribute>
         </container-transaction>
     </assembly-descriptor>
</ejb-jar>
```

5. Tolower サービスの security-role-assignment 要素を追加するために、Weblogic-ejb-jar.xml を修正します。太字の部分は、セキュリティ実装をサポートするための変更箇所を示します。

コード リスト 3-4 Tolower Weblogic-ejb-jar.xml セキュリティ サンプル コード

```
<?xml version="1.0"?>
<!--
```

```
Copyright (c) 2000 BEA Systems, Inc. All rights reserved

THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF BEA Systems, Inc. The copyright
notice above does not evidence any actual or intended publication of such source
code.

-->

<!DOCTYPE weblogic-ejb-jar PUBLIC '-//BEA Systems, Inc.//DTD WebLogic 6.0.0
EJB//EN' 'http://www.bea.com/servers/wls600/dtd/weblogic-ejb-jar.dtd'>

<weblogic-ejb-jar>
     <weblogic-enterprise-bean>
          <ejb-name>Tolower</ejb-name>
     <stateless-session-descriptor>
          <pool>
               <max-beans-in-free-pool>100</max-beans-in-free-pool>
          </pool>
     </stateless-session-descriptor>
     <jndi-name>tuxedo.services.TOLOWERHome</jndi-name>
     </weblogic-enterprise-bean>
    <security-role-assignment>
          <role-name>rdom2</role-name>
          <principal-name>john</principal-name>
          <principal-name>dan</principal-name>
     </security-role-assignment>
</weblogic-ejb-jar>
```

6. 発信リクエストの Tuxedo 環境を準備するには、次の手順を実行します。
   - 必要に応じて、tpgrpadd を使用してグループを追加します。
   - 必要に応じて、tpusradd を使用してユーザ John、Bob、および Dan を追加します。
   - tpacladd を使用して、Tuxedo ACL が保護する TOUPPER サービスを追加します。
   - ACL_POLICY="GLOBAL" を使用して、リモート ドメイン（WebLogic Server ドメイン）の BDMCONFIG を設定します。
7. 着信リクエストの Tuxedo 環境を準備するには、次の手順を実行します。
   - 必要に応じて、tpgrpadd を使用してグループを追加します。

- 必要に応じて、tpusradd を使用してユーザ John、Bob、および Dan を追加します。
- tpacladd を使用して、Tuxedo ACL が保護する TOUPPER サービスを追加します。
- ACL_POLICY="GLOBAL" を使用して、リモート ドメイン（WebLogic Server ドメイン）の BDMCONFIG を設定します。
  ACL_POLICY="LOCAL" を設定する場合、tpusradd を使用し、ユーザとしてリモート DOMAINID をコンフィグレーションする必要があります。

8. WebLogic Server 環境を準備するには、次の手順を実行します。
   - Tuxedo から tpusr ファイルをコピーするか、独自の tpusr ファイルを生成します。
   - XML コンフィグレーション ファイルの T_DM_REMOTE_TDOMAIN セクションに、TpUserFile 要素を追加します。

     例 :

     ```
     <TpUsrFile>full path name to tpusr</TpUsrFile>.
     ```

   - XML コンフィグレーション ファイルの T_DM_REMOTE_TDOMAIN セクションに、CredentialPolicy 要素を追加します。値は GLOBAL に設定します。

     例 :

     ```
     <CredentialPolicy>GLOBAL</CredentialPolicy>
     ```

リンクレベルの暗号化

ドメイン間で暗号化を使用すると、データのプライバシを守ることができます。これによって、ネットワークベースの傍受者は、あるドメイン ゲートウェイから別のドメイン ゲートウェイに送信されるメッセージやアプリケーション生成メッセージの内容を知ることができなくなります。このセキュリティ メカニズムは、XML コンフィグレーション ファイルの T_DM_LOCAL_TDOMAIN および T_DM_REMOTE_TDOMAIN セクションで `MINENCRYPTBITS` および `MAXENCRYPTBITS` パラメータを設定することによってコンフィグレーションします。

注意： 暗号を使用するには、適切なライセンスが必要です。ライセンス要件の詳細については、1-5 ページの「ライセンス」を参照してください。

4 CORBA アプリケーションの管理

注意： CORBA アプリケーションの詳細については、「Tuxedo CORBA」を参照してください。

この章では、Tuxedo CORBA クライアントおよびサービスをサポートする WebLogic Tuxedo Connector の管理およびコンフィグレーション方法について説明します。

- CORBA サービス アプリケーション用 WebLogic Tuxedo Connector のコンフィグレーション方法
- 着信 RMI/IIOP の WebLogic Tuxedo Connector を管理およびコンフィグレーションする方法
- 発信 RMI/IIOP 用 WebLogic Tuxedo Connector のコンフィグレーション方法

CORBA サービス アプリケーション用 WebLogic Tuxedo Connector のコンフィグレーション方法

注意： XML コンフィグレーション ファイルのコンフィグレーション方法については、9-1 ページの「WebLogic Tuxedo Connector XML コンフィグレーション ファイル」を参照してください。

この節では、WebLogic Server EJB から Tuxedo CORBA サーバへの呼び出しをサポートする XML コンフィグレーション ファイルのコンフィグレーション方法について説明します。XML コンフィグレーション ファイルの `BDMCONFIG` セクションを修正するには、次の手順に従います。

1. WebLogic Server ドメインの `T_DM_LOCAL_TDOMAIN` セクションをコンフィグレーションします。

2. Tuxedo CORBA ドメインの T_DM_REMOTE_TDOMAIN セクションをコンフィグレーションします。
3. T_DM_IMPORT セクションをコンフィグレーションします。
   - **ResourceName** を 「*//domain_id*」に設定します。*domain_id* は、Tuxedo UBBCONFIG ファイルで指定される DOMAINID です。CORBA ドメインのこのユニークな識別子の最大長は、// を含めて 15 文字です。
   - **LocalAccessPoint** を、T_DM_REMOTE_TDOMAIN の LocalAccessPoint 要素の値に設定します。
   - **RemoteAccessPointList** を T_DM_REMOTE_TDOMAIN の AccessPointId 要素の値に設定します。

WebLogic Server EJB を使用して Tuxedo CORBA サービスを呼び出すクライアント アプリケーションの開発方法については、『WebLogic Tuxedo Connector ATMI プログラマーズ ガイド』を参照してください。

XML コンフィグレーション ファイルの例

次の XML コンフィグレーション ファイルでは、Tuxedo CORBA サーバへの WebLogic Tuxedo Connector のコンフィグレーション方法の例を示します。

コード リスト 4-1 CORBA サーバ アプリケーションの XML コンフィグレーション ファイルの例

```
<?xml version="1.0"?>

<!DOCTYPE WTC_CONFIG SYSTEM
"http://www.bea.com/servers/wls610/dtd/wtc_config.dtd">

<!--Java and XML-->
     <WTC_CONFIG >
     <BDMCONFIG >
          <T_DM_LOCAL_TDOMAIN AccessPoint="examples">
               <WlsClusterName>Coolio</WlsClusterName>
               <AccessPointId>examples</AccessPointId>
               <Type>TDOMAIN</Type>
               <Security>NONE</Security>
```

```
            <ConnectionPolicy>ON_DEMAND</ConnectionPolicy>
            <BlockTime>30</BlockTime>
            <NWAddr>//localhost:20304</NWAddr>
        </T_DM_LOCAL_TDOMAIN>
        <T_DM_REMOTE_TDOMAIN AccessPoint="TUXDOM">
            <LocalAccessPoint>examples</LocalAccessPoint>
            <AccessPointId>TUXDOM</AccessPointId>
            <Type>TDOMAIN</Type>
            <NWAddr>//localhost:20305</NWAddr>
        </T_DM_REMOTE_TDOMAIN>
      < T_DM_IMPORT
          ResourceName="//simpapp"
          LocalAccessPoint="examples"
             RemoteAccessPointList="TUXDOM">
       </T_DM_IMPORT>
</BDMCONFIG>
</WTC_CONFIG>
```

着信 RMI/IIOP の WebLogic Tuxedo Connector を管理およびコンフィグレーションする方法

この節では、アプリケーション環境の管理方法および Tuxedo CORBA オブジェクトが RMI/IIOP API を使用して WebLogic Server にデプロイした EJB を呼び出すことを可能にする XML コンフィグレーション ファイルのコンフィグレーション方法について説明します。

- XML コンフィグレーション ファイルのコンフィグレーション
- Tuxedo アプリケーション環境の管理

XML コンフィグレーション ファイルのコンフィグレーション

注意： XML コンフィグレーション ファイルのコンフィグレーション方法については、9-1 ページの「WebLogic Tuxedo Connector XML コンフィグレーション ファイル」を参照してください。

この節では、Tuxedo CORBA オブジェクトが RMI/IIOP API を使用して WebLogic Server にデプロイした EJB を呼び出すことを可能にする WebLogic Tuxedo Connector のコンフィグレーション方法について説明します。XML コンフィグレーション ファイルの `BDMCONFIG` セクションを修正するには、次の手順に従います。

1. WebLogic Server ドメインの `T_DM_LOCAL_TDOMAIN` セクションをコンフィグレーションします。
2. Tuxedo CORBA ドメインの `T_DM_REMOTE_TDOMAIN` セクションをコンフィグレーションします。

Tuxedo アプリケーション環境の管理

注意： Tuxedo アプリケーション環境のコンフィグレーション方法の詳細については、「Tuxedo Administration Topics」を参照してください。

Tuxedo アプリケーション環境をコンフィグレーションする場合は、さらに次の手順を実行する必要があります。

1. 環境の TOBJADDR を設定します。

   例：`//<hostname>:2468`

2. 次のコマンドを入力して、Tuxedo ドメインの CosNaming ネームスペースに WebLogic Server（WLS）ネーミング サービスを登録します。

   ```
   cnsbind -o ior.txt WLS
   ```

`ior.txt` ファイルには、WebLogic Server のドメイン ネーミング サービスが含まれています。

コード リスト 4-2 iiop.ejb.stateless.server.tux Tuxedo クライアントの ior.txt ファイルの例

```
corbaloc:tgiop:examples/NameService
```

発信 RMI/IIOP 用 WebLogic Tuxedo Connector のコンフィグレーション方法

注意： XML コンフィグレーション ファイルをコンフィグレーションする方法の詳細については、9-1 ページの「WebLogic Tuxedo Connector XML コンフィグレーション ファイル」を参照してください。

ここでは、XML コンフィグレーション ファイルをコンフィグレーションし、RMI/IIOP API を使用して WebLogic Server EJB が Tuxedo CORBA オブジェクトを呼び出すことができるようにする方法を説明します。以下の手順に従って、XML コンフィグレーション ファイルの `BDMCONFIG` セクションを変更します。

1. WebLogic Server ドメインに対する `T_DM_LOCAL_TDOMAIN` セクションをコンフィグレーションします。
2. Tuxedo CORBA ドメインに対する `T_DM_REMOTE_TDOMAIN` セクションをコンフィグレーションします。発信 RMI/IIOP では、**`FederationURL`** と **`FederationName`** という 2 つの要素を追加する必要があります。
   - **`FederationURL`** には、JNDI に結合する 外部ネーム サービスの URL を設定します。これは、リモート Tuxedo CORBA オブジェクトにアクセスするために使用する初期コンテキストを取得するために EJB が使用する URL と同じです。
   - **`FederationName`** には、結合ポイントの識別名を設定します。

注意： 詳細については、11-9 ページの「FederationURL」および 11-9 ページの「FederationName」を参照してください。

3. `T_DM_IMPORT` セクションをコンフィグレーションします。

- **ResourceName** に「*//domain_id*」を設定します。*domain_id* は、オブジェクトがデプロイされているリモート Tuxedo ドメインの Tuxedo UBBCONFIG ファイルで指定されている DOMAINID です。CORBA ドメインに対するこのユニークな識別子の最大長は、「//」を含めて 15 文字です。
- **LocalAccessPoint** に、T_DM_REMOTE_TDOMAIN の LocalAccessPoint 要素の値を設定します。
- **RemoteAccessPointList** に、T_DM_REMOTE_TDOMAIN の AccessPointId 要素の値を設定します。
- 省略可能。トランザクション タイムアウト値を指定します。

RMI/IIOP を使って WebLogic Server EJB を使用する Tuxedo サービスを呼び出すアプリケーションの開発方法については、『WebLogic Tuxedo Connector ATMI プログラマーズ ガイド』を参照してください。

XML コンフィグレーション ファイルのサンプル

次に示す XML コンフィグレーション ファイルは、発信 RMI/IIOP 用の WebLogic Tuxedo Connector をコンフィグレーションする方法のサンプルです。

コード リスト 4-3 発信 RMI/IIOP 用 XML コンフィグレーション ファイルのサンプル

```
<?xml version="1.0"?>

<!DOCTYPE WTC_CONFIG SYSTEM
"http://www.bea.com/servers/wls610/dtd/wtc_config.dtd">

<!--Java and XML-->
<WTC_CONFIG>
<BDMCONFIG>
     <T_DM_LOCAL_TDOMAIN AccessPoint="examples">
          <WlsClusterName>Coolio</WlsClusterName>
          <AccessPointId>examples</AccessPointId>
          <Type>TDOMAIN</Type>
          <Security>NONE</Security>
          <ConnectionPolicy>ON_DEMAND</ConnectionPolicy>
          <BlockTime>30</BlockTime>
          <NWAddr>//127.0.0.1:5000</NWAddr>
```

```
     </T_DM_LOCAL_TDOMAIN>
     <T_DM_REMOTE_TDOMAIN>
          <T_DM_REMOTE_TDOMAIN AccessPoint="TDOM1">
          <LocalAccessPoint>examples</LocalAccessPoint>
          <AccessPointId>TDOM1</AccessPointId>
          <Type>TDOMAIN</Type>
          <FederationURL>corbaloc:tgiop:simpapp/NameService</FederationURL>
          <FederationName>tuxedo.corba.remote</FederationName>
          <NWAddr>//127.0.0.1:4000</NWAddr>
     </T_DM_REMOTE_TDOMAIN>
     <T_DM_IMPORT
          ResourceName="//simpapp"
          LocalAccessPoint="examples"
          RemoteAccessPointList="TDOM1">
          <TranTime>600</TranTime>
     </T_DM_IMPORT>
</BDMCONFIG>
</WTC_CONFIG>
```

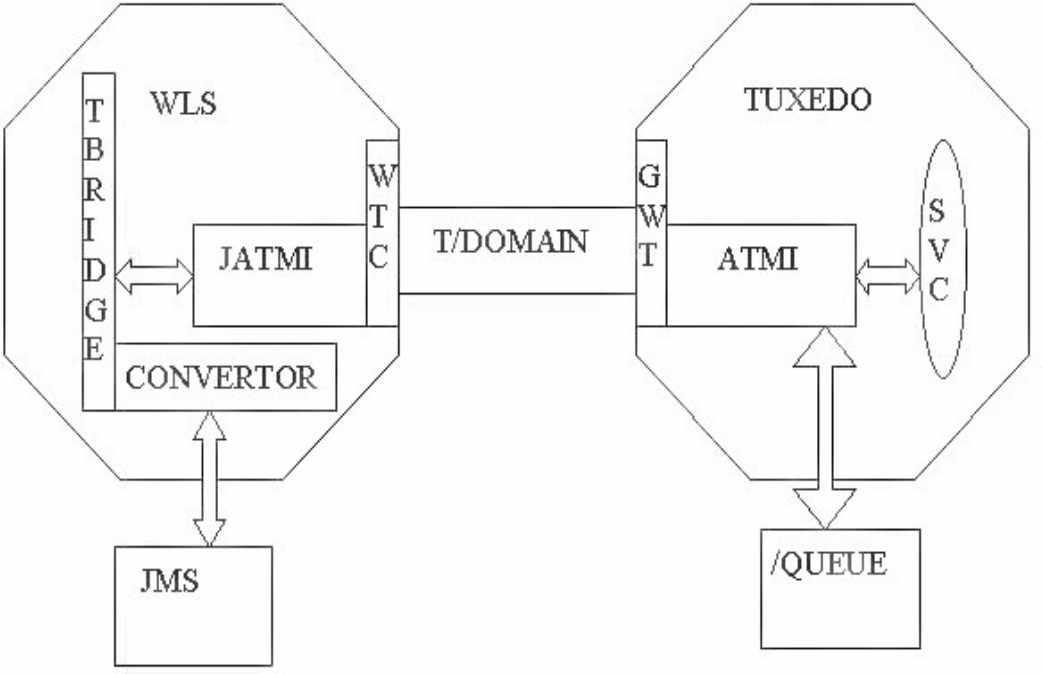
5 tBridge のコンフィグレーション

注意： WebLogic Tuxedo Connector XML コンフィグレーション ファイル、要素と属性、および `wtc_config.dtd` に関する詳細なリファレンス情報については、第 10 章「wtc_config.dtd」を参照してください。

この章では、tBridge の機能とコンフィグレーションについて説明します。

- tBridge の概要
- tBridge 用 WebLogic Tuxedo コンフィグレーション XML ファイル コンフィグレーション
- tBridge の接続性
- 接続タイプのコンフィグレーションの例
- 優先度のマッピング
- エラー キュー

tBridge の概要



tBridge は WebLogic Tuxedo Connector の一部で、Tuxedo アプリケーション環境と通信する WebLogic Server アプリケーションの双方向 JMS インタフェースを提供します。環境間でのメッセージングの転送は、クライアント アプリケーションの代わりにサービスを呼び出すのに使用するテキスト、Byte、または XML データ ストリームを含む JMS ベースのメッセージで構成されます。

次の機能は、tBridge の機能を決定します。

- 接続性は、WebLogic Tuxedo Connector XML コンフィグレーション ファイルの `tBridge` セクションにあるパラメータのコンフィグレーションによって決定されます。追加のアプリケーション プログラミングは必要ありません。
- tBridge は、Java Messaging Service（JMS）を使用して Tuxedo /Q または Tuxedo サービスへのインタフェースを提供します。

- tBridge は、既存の Tuxedo システムに接続性を提供するために、XML と FML32 間においてシンプルな変換を提供します。

tBridge が JMS と Tuxedo を接続する仕組み

注意： メッセージはすべて、確認応答されるまで JMS キューに残っています。

ここでは、JMS メッセージが tBridge を通して Tuxedo のキューとサービスまで送られる仕組みについて説明します。

1. Web 対応の WLPI アプリケーションのような JMS クライアントは、Tuxedo による処理の必要なメッセージを JMS キューに格納します。このメッセージがトランザクションの一部である場合は、トランザクションがコミットします。
2. tBridge Converter が、処理を行うために JMS キューからメッセージを取り出します。
3. tBridge Converter は、メッセージのタイプをチェックし、サポートされている JMS タイプを JATMI バッファ タイプに変換します。

- BytesMessage、TextMessage、XML は、それぞれ、TypedCArray、TypedString、TypedFML32 に変換されます。XML/FML の変換は、`TranslateFML` 属性に従って行われます。
- 変換エラーは `wlsServerErrorDestination` キューに送信されて、メッセージは JMS セッションで確認応答されます。
- 認識できない JMS メッセージを受信した場合は、適切なエラー メッセージが記録され、メッセージは肯定応答されてから破棄されます。これはコンフィグレーションのエラーと考えられ、tBridge はメッセージをエラー キューにリダイレクトしません。

4. 変換されたメッセージは、T/Domain ゲートウェイを使って Tuxedo に送信されます。

- リダイレクトが `JmsQ2TuxQ` に設定されたメッセージは、JATMI の `tpenqueue` を使って Tuxedo キューに配信されます。
- リダイレクトが `JmsQ2TuxS` に設定されたメッセージは、JATMI の `tpcall` を使って Tuxedo サービスに配信されます。

5. tpenqueue または tpcall の処理が正常に行われて、結果が replyQ に格納されます。メッセージは、JMS セッションで確認応答されます。

- tpenqueue または tpcall の処理が失敗すると、tBridge はメッセージを wlsServerErrorDestination キューに配信し、メッセージは JMS セッションで確認応答されます。wlsServerErrorDestination キューがコンフィグレーションされていない場合は、メッセージは破棄されて、tBridge は次に取り出すことのできる確認応答されていないメッセージを処理します。

tBridge が Tuxedo を JMS に接続する仕組み

注意： tBridge は、トランザクションを使用して、Tuxedo /Q から JMS キューへの転送中にメッセージが失われることを防ぎます。

ここでは、Tuxedo メッセージが TuxQ2JmsQ リダイレクトを使用して tBridge 経由で JMS キューに送られる仕組みについて説明します。

1. tBridge は、取り出すことのできるメッセージがあるかどうかを Tuxedo キューにポーリングします。
2. Tuxedo サービスは、Tuxedo キューにメッセージを格納します。
3. tBridge は、JATMI の tpdequeue を使用して Tuxedo からメッセージを転送し、JMS キューにメッセージを格納します。

- 指定されている回数だけリトライを実行しても何らかの理由でメッセージを JMS キューにリダイレクトできない場合には、メッセージは Tuxedo キューと同じキュー スペース内の tuxErrorDestination キューに格納されます。
- 何らかの理由で、tBridge がメッセージを tuxErrorDestination キューに格納できない場合は、ログにエラーが記録されて、メッセージは失われます。
- tuxErrorDestination キューが指定されていない場合は、メッセージは失われます。

tBridge の制限

tBridge には、以下の制限事項があります。

- JMS の場所からメッセージを取り出して Tuxedo キューにメッセージを格納したり、Tuxedo サービスを呼び出したりするときには、トランザクションは使用されません。
- tBridge は多くのスレッドを使用します。スレッドは、JMS キューから Tuxedo への各メッセージの転送に使用されます。コンフィグレーションされている Tuxedo キューをモニタするには、ポーリング スレッドが必要です。
- XML/FML トランスレータは、単純なメッセージ構造を作成するためのものです。XML から FML への変換の詳細については、6-7 ページの「FML32 の考慮事項」を参照してください。

tBridge 用 WebLogic Tuxedo コンフィグレーション XML ファイル コンフィグレーション

WebLogic Tuxedo Connector の tBridge 接続性は、Tuxedo への接続を確立するために必要な情報を含む XML コンフィグレーション ファイルによって決定されます。

tBridge の起動

tBridge は、XML コンフィグレーション ファイルの tBridge セクションがコンフィグレーションされている場合は、WebLogic Server アプリケーション環境の一部として起動されます。tBridge が起動できないようなコンフィグレーション条件が発生した場合は、エラーがログされます。

エラーのロギング

WebLogic Tuxedo Connector エラーは、WebLogic Server エラー ログにログされます。

tBridge の接続性

注意 : MapMessage、ObjectMessage、StreamMessage の JMS メッセージ タイプは、WebLogic Tuxedo Connector では無効です。これらのメッセージ タイプの 1 つが tBridge によって受信された場合は、サポートされていないタイプであることを示すログ エントリが作成され、メッセージが破棄されます。

tBridge は、JMS キューと Tuxedo /Q または JMS キューと Tuxedo サービスのインスタンス間で、一方向のデータ接続を確立します。この接続は、`<fromto>` 要素として、コンフィグレーション ファイルの `tBridge` セクションで指定されます。各データ接続は、識別されたポイント間に 1 対 1 の接続を提供します。3 つのタイプの接続がコンフィグレーションできます。各接続タイプは、`<direction>` 要素の値として、コンフィグレーション ファイルの tBridge セクションで指定されます。次に、各接続タイプについて説明します。

- **`JmsQ2TuxQ`**: JMS キューから読み取りを行い、指定された Tuxedo /Q にメッセージを転送します。
- **`TuxQ2JmsQ`**: Tuxedo /Q から読み取りを行い、JMS にメッセージを転送します。
- **`JmsQ2TuxS`**: JMS キューから読み取りを行い、同時に指定された Tuxedo サービスを呼び出し、指定された JMS キューに応答を返します。

接続タイプのコンフィグレーションの例

次の節では、各接続タイプのコンフィグレーションの例を紹介します。

JmsQ2TuxQ コンフィグレーションの例

次は、JMS キューから読み取りを行い、Tuxedo /Q に送信するコードの例を示します。

```
<fromto>
     <direction>JmsQ2TuxQ</direction>
          <source>
               <Name>weblogic.jms.Jms2TuxQueue</Name>
          </source>
          <target>
               <AccessPoint>TDOM2</AccessPoint>
               <Qspace>QSPACE</Qspace>
               <Name>STRING</Name>
          </target>
     <replyQ>RPLYQ</replyQ>
     <translateFML>NO</translateFML>
</fromto>
```

次に、**JmsQ2TuxQ** コンフィグレーションの各コンポーネントを説明します。

- <direction> 接続タイプは *JmsQ2TuxQ* です。
- <source> <Name> キーワードは、読み取る JMS キューの名前が *weblogic.jms.Jms2TuxQueue* であることを指定します。tBridge は、CLIENT_ACKNOWLEDGE セマンティクスを使用して、このキューに JMS クライアント セッションを確立します。
- <target> キーワードは、宛先を明示的に参照するために必要な要素を指定します。
  - <AccessPoint> キーワードは、アクセス ポイントの名前が *TDOM2* であることを指定します。
  - <Qspace> キーワードは、Qspace の名前が *Qspace* であることを指定します。
  - <Name> キーワードは、キューの名前が *STRING* であることを指定します。
- <replyQ> キーワードは、JMS 応答キューの名前が *ReplyQ* であることを指定します。このキューを使用すると、tpenqueue は TMFORWARD 機能を提供します。
- <translateFML> キーワード *NO* は、tBridge がデータ変換を提供しないことを指定します。

次の表は、JmsQtoTuxQ メッセージ マッピングの情報を示します。

マッピング元 : JMS メッセージ タイプ	マッピング先 : WebLogic Tuxedo Connector JATMI（Tuxedo）
BytesMessage	TypedCArray
TextMessage (translateFML = NONE)	TypedString
TextMessage (translateFML = FLAT)	TypedFML32

TuxQ2JmsQ コンフィグレーションの例

次に、Tuxedo /Q を WebLogic Server にインポートするためのサンプル コードを示します。

```
< T_DM_IMPORT
     ResourceName="QSPACE"
     LocalAccessPoint="LDOM2"
     RemoteAccessPointList="MYLOCAL"
<TranTime>600</TranTime>
</T_DM_IMPORT>
```

各要素の説明は次のとおりです。

- `ResourceName` は、`BDMCONFIG` ファイルの `*DM_LOCAL_SERVICES` セクションで記述される Tuxedo /Q です。
- `LocalAccessPoint` は、`BDMCONFIG` ファイルの `*DM_REMOTE_DOMAINS` セクションで記述される WebLogic Server ドメインの名前です。
- `RemoteAccessPointList` は、`BDMCONFIG` ファイルの `*DM_LOCAL_DOMAINS` セクションで記述される Tuxedo ドメインの名前です。

次は、Tuxedo /Q キューから読み取りを行い、JMS キューに送信するコードの例を示します。

```
<fromto>
     <direction>TuxQ2JmsQ</direction>
          <source>
               <AccessPoint>TDOM2</AccessPoint>
               <Qspace>QSPACE</Qspace>
               <Name>STRING</Name>
          </source>
          <target>
               <Name>weblogic.jms.Tux2JmsQueue</Name>
```

```
        </target>
    <ReplyQ>NO</ReplyQ>
    <translateFML>NO</translateFML>
</fromto>
```

次に、**TuxQ2JmsQ** コンフィグレーションの各コンポーネントを説明します。

- <direction> 接続タイプは *TuxQ2JmsQ* です。
- <source> <Name> キーワードは、読み取る JMS キューの名前が *weblogic.jms.Tux2JmsQueue* であることを指定します。
- <target> は、宛先を明示的に参照するために必要な要素を指定します。
  - <AccessPoint> キーワードは、アクセス ポイントの名前が *TDOM2* であることを指定します。
  - <Qspace> キーワードは、Qspace の名前が *Qspace* であることを指定します。
  - <Name> キーワードは、キューの名前が *STRING* であることを指定します。
- <translateFML> キーワード *NO* は、tBridge がデータ変換を提供しないことを指定します。

次の表は、TuxQ2JmsQ メッセージ マッピングの情報を示します。

マッピング元 : WebLogic Tuxedo Connector JATMI（Tuxedo）	**マッピング先 : JMS メッセージ タイプ**
TypedCArray	BytesMessage
TypedString (translateFML = NONE)	TextMessage
TypedFML32 (translateFML = FLAT)	TextMessage
TypedFML (translateFML = FLAT)	TextMessage
TypedXML	TextMessage

JmsQ2TuxS コンフィグレーションの例

注意： XML/FML 変換の詳細については、第 6 章「WebLogic Tuxedo Connector での FML の使用」を参照してください。

次は、JMS キューから読み取りを行い、Tuxedo サービスを呼び出して、結果を JMS キューに返すコードの例を示します。

```
<fromto>
     <direction>JmsQ2TuxS</direction>
          <source>
               <Name>weblogic.jms.Jms2TuxQueue</Name>
          </source>
          <target>
               <AccessPoint>TDOM2</AccessPoint>
               <Name>REVERSE_STRING</Name>
          </target>
     <replyQ>weblogic.jms.Tux2JmsQueue</replyQ>
     <translateFML>FLAT</translateFML>
</fromto>
```

次に、**JmsQ2TuxS** コンフィグレーションの各コンポーネントを説明します。

- <direction> 接続タイプは *JmsQ2TuxS* です。
- <source> <Name> キーワードは、読み取る JMS キューの名前が *weblogic.jms.Jms2TuxQueue* であることを指定します。
- <target> は、宛先を明示的に参照するために必要な要素を指定します。
  - <AccessPoint> キーワードは、アクセス ポイントの名前が *TDOM2* であることを指定します。
  - <Name> キーワードは、キューの名前が *REVERSE_STRING* であることを指定します。
- <replyQ> キーワードは、JMS 応答キューの名前が *ReplyQ* であることを指定します。
- <translateFML> キーワード *FLAT* は、JMS メッセージの受信時に、メッセージが XML フォーマットで、対応する FML32 データ バッファに変換されることを指定します。メッセージは、引数 *TDOM2* および *REVERSE_STRING* とともに tpcall に配置されます。メッセージは FML32 から XML に変換され、weblogic.jms.Tux2JmsQueue に配置されます。

次の表は、JMSQ2TuxX メッセージ マッピングの情報を示します。

JMS メッセージ タイプ	WebLogic Tuxedo Connector JATMI（Tuxedo）	JMS メッセージ タイプ
BytesMessage	TypedCArray	BytesMessage
TextMessage (translateFML = NONE)	TypedString	TextMessage
TextMessage (translateFML = FLAT)	TypedFML32	TextMessage

優先度のマッピング

WebLogic Tuxedo Connector は、複数の tBridge リダイレクト インスタンスをサポートしています。多くの環境では、複数のリダイレクト インスタンスを使用することで、アプリケーションのスケーラビリティとパフォーマンスが大幅に向上します。ただし、メッセージが処理される順序はランダムになります。優先度のマッピングを使用すると、順序は保証されませんが、割り当てられている重要度に基づいてメッセージに対応するメカニズムが提供されます。配信の順序を保証する必要がある場合は、単一の tBridge リダイレクト インスタンスを使用してください。

JMS および Tuxedo 間で優先度をマップするには、`priorityMapping` を使用します。

- JMS には 10 の優先度があります（0 ～ 9）。
- Tuxedo/Q には 100 の優先度があります（1 ～ 100）。

この節では、Tuxedo および JMS サブシステム間で優先度をマップするメカニズムについて説明します。2 つのマッピング方向があります。

- **`JmstoTux`**
- **`TuxtoJms`**

次の `value:range` の組み合わせで示すように、すべての値にデフォルトが提供されています。

- value は入力優先度を指定します。
- range は結果出力優先度のシーケンシャル グループを指定します。

JmstoTux- 0:1 | 1:12 | 2:23 | 3:34 | 4:45 | 5:56 | 6:67 | 7:78 | 8:89 | 9:100

TuxtoJms- 1-10:0 | 11-20:1 | 21-30:2 | 31-40:3 | 41-50:4| 51-60:5 | 61-70:6 | 71-80:7 | 81-90:8 | 91-100:9

例 :

次の priorityMapping は、デフォルトのマッピングを示します。

コード リスト 5-1 優先度マッピングのコンフィグレーション例

```
.
.
.
<priorityMapping>
         <TuxtoJms>
            <pMap>
               <value>1-10</value>
               <range>0</range>
            </pMap>
            <pMap>
               <value>11-20</value>
               <range>1</range>
            </pMap>
            <pMap>
               <value>21-30</value>
               <range>2</range>
            </pMap>
            <pMap>
               <value>31-40</value>
               <range>3</range>
```

```
            </pMap>
            <pMap>
               <value>41-50</value>
               <range>4</range>
            </pMap>
            <pMap>
               <value>51-60</value>
               <range>5</range>
            </pMap>
            <pMap>
               <value>61-70</value>
               <range>6</range>
            </pMap>
            <pMap>
               <value>71-80</value>
               <range>7</range>
            </pMap>
            <pMap>
               <value>81-90</value>
               <range>8</range>
            </pMap>
            <pMap>
               <value>91-100</value>
               <range>9</range>
            </pMap>
    </TuxtoJms>
        <JmstoTux>
            <pMap>
               <value>0</value>
               <range>1</range>
            </pMap>
```

```
<pMap>
   <value>1</value>
   <range>12</range>
</pMap>
<pMap>
   <value>2</value>
   <range>23</range>
</pMap>
<pMap>
   <value>3</value>
   <range>34</range>
</pMap>
<pMap>
   <value>4</value>
   <range>45</range>
</pMap>
<pMap>
   <value>5</value>
   <range>56</range>
</pMap>
<pMap>
   <value>6</value>
   <range>67</range>
</pMap>
<pMap>
   <value>7</value>
   <range>78</range>
</pMap>
<pMap>
   <value>8</value>
   <range>89</range>
```

```
            </pMap>
            <pMap>
               <value>9</value>
               <range>100</range>
            </pMap>
         </JmstoTux>
 </priorityMapping>
.
.
.
```

このコンフィグレーションでは、優先度 7 の JMS メッセージが Tuxedo /Q の優先度 78 に割り当てられます。優先度 47 の Tuxedo /Q は、優先度 4 の JMS に割り当てられます。

エラー キュー

tBridge が Tuxedo キューまたは JMS キューからメッセージを受信したときに問題を検出した場合、再試行間隔の後に次の処理が行われます。

- 情報がログされます。
- コンフィグレーションされている場合は、メッセージがエラー キューに保存されます。

wlsServerErrorDestination

`wlsErrorDestination` は、JMS メッセージが Tuxedo エラーまたは変換エラーにより適切に送信できない場合に使用されます。

サポートされていないメッセージ タイプ

認識できない JMS メッセージを受信した場合、適切なエラー メッセージがログされ、そのメッセージは破棄されます。これはコンフィグレーション エラーと解釈され、tBridge はメッセージをエラー キューにリダイレクトしません。

tuxErrorQueue

`tuxErrorQueue` は、`TuxQ2JmsQ` リダイレクト中の JATMI プリミティブ `tpdequeue` に対するエラー キューです。

制限

tBridge エラー キューには、次のような制限があります。

- `TuxErrorDestination` は一度だけ指定できます。`ErrorDestination` に関連するエラー キュー名は、すべての QSPACE が使用可能な同じエラー キュー名を持っていることを示します。
- エラーが発生した場合、メッセージはソース QSPACE に返されます。QSPACE が破損または一杯と判断されると、後続のメッセージは消失されます。
- エラー時にメッセージを破棄するよう指定する方法はありません。すべてのメッセージが受信されるか、まったく受信されません。
- エラーに関する情報は、サーバ ログでのみ提供されます。

6 WebLogic Tuxedo Connector での FML の使用

この章では、フィールド操作言語（FML）および WebLogic Tuxedo Connector が FML を使用する方法について説明します。

- FML の概要
- WebLogic Tuxedo Connector FML API
- FML フィールド テーブルの管理
- tBridge XML/FML32 変換

FML の概要

注意： FML の使い方の詳細については、『Programming a Application Using FML』を参照してください。

FML は、フィールド バッファと呼ばれる記憶構造を定義および操作する Java 言語機能のセットです。各フィールド バッファには、フィールドに属性および値の組み合わせが含まれています。各フィールドは次のように構成されます。

- 属性はフィールドの識別子です。
- 関連する値はフィールドのデータ内容を示します。
- オカレンス番号。

2 種類の FML があります。

- FML16。フィールドの長さと識別子は 16 ビットの値で表されます。ユニークなフィールドの数は 8191、個々のフィールドの長さは 64K バイト、フィールド化されるバッファの全体のサイズは 64K バイトにそれぞれ制限されます。

- FML32。フィールドの長さと識別子は 32 ビットの値で表されます。フィールドの数は約 3000 万で、フィールドとバッファ長は約 20 億バイトです。

WebLogic Tuxedo Connector FML API

注意： WebLogic Tuxedo Connector は、FML 機能のサブセットを実装しています。たとえば、views はサポートされていません。

FML アプリケーション プログラム インタフェース（API）は、WebLogic Server クラスの Javadoc に含まれている weblogic.wtc.jatmi パッケージにドキュメント化されています。

FML フィールド テーブルの管理

フィールド テーブルは、Tuxedo フィールド テーブルに似た方法で生成されます。フィールド テーブルは、2 つのシステム間で共通のフィールド名定義、フィールド タイプ、および識別番号を提供するテキスト ファイルです。FML を使用して Tuxedo システムと相互運用するには、次の手順を実行する必要があります。

1. Tuxedo システムから WebLogic Tuxedo Connector 環境に、フィールド テーブルをコピーします。

   例 : Tuxedo 配布キットには、bankapp という銀行アプリケーション サンプルが含まれています。このアプリケーションには、次の構造を持つ bankflds というファイルがあります。

```
#Copyright (c) 1990 Unix System Laboratories, Inc.
#All rights reserved
#ident  "@(#) apps/bankapp/bankflds     $Revision: 1.3 $"
# Fields for database bankdb

# name                          number  type    flags   comments
ACCOUNT_ID                      110     long    -       -
ACCT_TYPE                       112     char    -       -
ADDRESS                         109     string  -       -
.
```

.

.

2. フィールド テーブル定義を Java ソース ファイルに変換します。
   weblogic.wtc.jatmi パッケージの mkfldclass ユーティリティを使用します。このクラスは、FML32 フィールド テーブルを読み取るユーティリティ機能で、FldTbl インタフェースを実装する Java ファイルを生成します。このユーティリティには、次のような 2 つのインスタンスがあります。

   a. mkfldclass

   b. mkfldclass32

   コマンドの正しいインスタンスを使用して、bankflds フィールド テーブルを FML32 Java ソースに変換します。次は、mkfldclass を使用した例です。

   ```
   java weblogic.wtc.jatmi.mkfldclass bankflds
   ```

   作成されるファイルの名前は bankflds.java で、次のような構造になります。

```
import java.io.*;
import java.lang.*;
import java.util.*;
import weblogic.wtc.jatmi.*;

public final class bankflds
        implements weblogic.wtc.jatmi.FldTbl
{
        /** number: 110  type: long */
        public final static int ACCOUNT_ID = 33554542;
        /** number: 112  type: char */
        public final static int ACCT_TYPE = 67108976;
        /** number: 109  type: string */
        public final static int ADDRESS = 167772269;
        /** number: 117  type: float */

.

.

.
```

3. 次のコマンドを使用して、作成した bankflds.java ファイルをコンパイルします。

   ```
   javac bankflds.java
   ```

   コンパイルすると、bankflds.class ファイルとなります。ロード時、WebLogic Tuxedo Connector はクラス ファイルを使用して FML32 フィールドからフィールド エントリを追加、検索、および削除します。

4. フィールド テーブルのクラス ファイルをアプリケーション CLASSPATH に追加します。
5. WebLogic Tuxedo Connector XML コンフィグレーション ファイルを更新します。
   - フィールド テーブルのクラス ファイルの完全に修飾された場所を反映するため、T_DM_RESOURCES セクションを更新します。
   - fml16 または fml32 の FML バッファ タイプを記述するために必要なキーワードを使用します。
   - 追加のフィールド テーブルを指定するために、複数の <FldTblClass> 行を入力できます。

   例 :

```
<T_DM_RESOURCES>
        <FieldTables>
        <FldTblClass Type="fml32">com.bea.mystuff.bankflds</FldTblClass>
        </FieldTables>
</T_DM_RESOURCES>
```

6. フィールド テーブルのクラス定義をロードするために、WebLogic Server を再起動します。

mkfldclass32 クラスに対する DynRdHdr プロパティの使い方

WebLogic Tuxedo Connector には、FML テーブルをコンパイルするための代替手段を提供するプロパティが用意されています。以下の場合に、DynRdHdr ユーティリティの使用が必要になる場合があります。

- 非常に大きな FML テーブルを使用していて、mkfldclass32 クラスによって作成される .java メソッドが、単一のクラスまたはインタフェースの全体的な複雑さに対する Java 仮想マシンの内部的な限界を超える場合。
- 非常に大きな FML テーブルを使用していて、.java メソッドのコンパイル時に作成されるクラスをロードできない場合。

DynRdHdr プロパティを使って FML テーブルをコンパイルするときは、以下の手順で行います。

1. フィールド テーブルの定義を、Java のソース ファイルに変換します。

   ```
   java -DDynRdHdr=Path_to_Your_FML_Table weblogic.wtc.jatmi.mkfldclass32 userTable
   ```

このコマンドの引数は、次の表に示すとおりです。

引数	説明
`-DDynRdHdr`	FML テーブルのコンパイルに使用する WebLogic Tuxedo Connector のプロパティ。
`Path_to_Your_FML_Table`	FML テーブルの完全修飾パスとファイル名。
`weblogic.wtc.jatmi.mkfldclass32`	FML32 フィールド テーブルを読み取って `FldTbl` インタフェースを実装する Java ファイルを生成するユーティリティ関数のクラス。
`userTable`	mkfldclass32 クラスによって作成される `.java` メソッドの名前。

2. 次のコマンドを使って *`userTable`* ファイルをコンパイルします。

   ```
   javac userTable.java
   ```

3. アプリケーション CLASSPATH に *`userTable`*`.class` ファイルを追加します。
4. WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの `T_DM_RESOURCES` セクションを変更し、*`userTable`*`.class` ファイルの完全修飾された場所を指定します。
5. サーバを起動します。サーバは、起動すると、WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの `T_DM_RESOURCES` セクションで指定されている場所を使って、FML テーブルをロードします。

いったん *`userTable`*`.class` ファイルを作成した後は、FML テーブルを変更してデプロイする際に、*`userTable`*`.class` を手作業で更新する必要はありません。サーバは、起動時に、WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの `T_DM_RESOURCES` で指定されている場所を使って、更新された

FML テーブルをロードします。ただし、*Path_to_Your_FML_Table* 属性を変更する場合は、前記の手順を使って、*userTable*.java ファイルと *userTable*.class ファイルを更新する必要があります。

tBridge XML/FML32 変換

注意： 指定するデータ型は、FLAT または NO にする必要があります。他のデータ型を指定すると、リダイレクションは失敗します。

<translateFML> 要素は、FML32 変換がメッセージペイロードで実行されることを示すために使用します。FML32 変換には、FLAT および NO の 2 つのタイプがあります。

FLAT 変換の使い方

メッセージ ペイロードは、WebLogic Tuxedo Connector の内部 FML32/XML トランスレータを使用して変換されます。フィールドは、メッセージ構造（階層構造）を意識せず、グループ化を反復せずに、フィールドごとの値に変換されます。

FML32 バッファを XML に変換するため、tBridge は FML32 バッファにある各フィールドの各インスタンスを抽出して文字列に変換し、それをフィールド名で構成されるタグ内に配置します。これらすべてのフィールドは、サービス名で構成されるタグ内に配置されます。たとえば、次のフィールドで構成される FML32 バッファがあるとします。

```
NAME          JOE
ADDRESS       CENTRAL CITY
PRODUCTNAME   BOLT
PRICE         1.95
PRODUCTNAME   SCREW
PRICE         2.50
```

変換された XML バッファは次のようになります。

```
<FML32>
  <NAME>JOE</NAME>
  <ADDRESS>CENTRAL CITY</ADDRESS>
  <PRODUCTNAME>BOLT</PRODUCTNAME>
  <PRODUCTNAME>SCREW</PRODUCTNAME>
```

```
  <PRICE>1.95</PRICE>
  <PRICE>2.50</PRICE>
</FML32>
```

NO 変換の使い方

変換は使用されません。tBridge は、リダイレクションの方向に応じて、JMS TextMessage を Tuxedo TypedBuffer（TypedString）に（またはその逆に）マップします。JMS BytesMessage は、Tuxedo TypedBuffer（TypedCarray）に（またはその逆に）マップされます。

FML32 の考慮事項

FML32 を使用する場合は、以下の情報について考慮する必要があります。

- XML 入力では、ルート要素が必要ですが無視されます。
- XML 出力では、ルート要素は常に <FML32> です。
- フィールド テーブル名は、6-2 ページの「FML フィールド テーブルの管理」で説明されているようにロードする必要があります。
- tBridge トランスレータは、「flat」またはリニアなグループ化でのみ有効です。これは FML32 順を記述する情報が管理されないために、繰り返しデータを含むバッファが予期しない形式で示される場合があります。たとえば、部品とそれに関連付けられた価格のリストを含む FML32 バッファがあると仮定します。ここでは PART A、PRICE A、PART B、PRICE B などが予想されますが、tBridge 内に構造的なグループ情報がないため、変換された XML は PART A、PART B など、PRICE A、PRICE B などになります。
- XML を FML32 に変換する場合、トランスレータは空白値を無視します。たとえば、`<STRING></STRING>` は、結果的に FML32 バッファではスキップされます。
- 組み込み FML は、このリリースではサポートされません。
- FML から XML への変換では、TypedCArray はサポートされません。以下のサポートされているフィールド型から選択してください。
  - SHORT

- LONG
- CHAR
- FLOAT
- DOUBLE
- STRING
- INT (FML32)
- DECIMAL (FML32)

■ FML 内に TypedCArray が存在する場合、TypedString へのコード化を行い、XML を TypedCArray にデコードします。

■ バイナリ データを渡す必要がある場合、選択したフィールド型へのコード化を行い、受け取り側で XML をデコードします。

7 WebLogic Process Integrator と Tuxedo アプリケーションの接続

注意： アプリケーションの統合方法の詳細については、「BEA WebLogic Integration」を参照してください。

WebLogic Tuxedo Connector tBridge は、Tuxedo アプリケーションをビジネスワークフローに統合するために必要なインフラストラクチャを WebLogic Process Integrator ユーザに提供します。この章では、WebLogic Tuxedo コネクタを使用して WebLogic Process Integrator と Tuxedo を統合する方法について説明します。

- WebLogic Process Integrator と Tuxedo の同期接続
- WebLogic Process Integrator と Tuxedo の同期非ブロッキング接続
- WebLogic Process Integrator と Tuxedo の非同期接続
- 非同期 Tuxedo /Q と WebLogic Process Integrator の接続
- Tuxedo と WebLogic Process Integrator の双方向非同期接続

WebLogic Process Integrator と Tuxedo の同期接続

WebLogic Process Integrator は、JATMI EJB を使用して Tuxedo に対してブロッキング呼び出しを実行します。このプロセスは、次の 3 つの部分で構成されます。

- WebLogic Process Integrator ビジネス オペレーションの定義
- eLink アダプタの呼び出し
- WebLogic Process Integrator 例外ハンドラの定義

ビジネス オペレーションの定義

使用する JATMI メソッドの WebLogic Process Integrator ビジネス オペレーションを定義します。

- TypedFML32 バッファ操作メソッド。
- JATMI **`tpcall()`** メソッドを使用します。

  例 : out_buffer = tpcall (service_name, in_buffer, flags)

eLink アダプタの呼び出し

WebLogic Process Integrator プロセス フローから eLink アダプタを呼び出します。

- 定義されたビジネス オペレーションを使用して、TypedFML32 リクエスト バッファを構築します。
- 定義されたビジネス オペレーションを使用して、サービス名を指定する JATMI **`tpcall()`** メソッドを呼び出します。
- 定義されたビジネス オペレーションを使用して、TypedFML32 応答バッファを処理します。

例外ハンドラの定義

例外を処理するために、WebLogic Process Integrator 例外ハンドラを定義します。

WebLogic Process Integrator と Tuxedo の同期非ブロッキング接続

WebLogic Process Integrator は、Tuxedo サービスを同期的に呼び出すためにメッセージを送信します。

- JMS キューと Tuxedo サービスへの呼び出し間の 1 対 1 の関係。
- Tuxedo サービスからの応答と JMS キュー間の 1 対 1 の関係。
- WebLogic Process Integrator は、メッセージを JMS キューに書き込みます。
- メッセージが JMS キューに入ると、tBridge はターゲットの Tuxedo サービスにメッセージを移動します。
- メッセージは、XML と FML32 間で変換されます。
- 応答は、指定された JMS 応答キューに書き込まれます。
- WebLogic Process Integrator イベント ノードは、応答メッセージの応答キューを待機します。

WebLogic Process Integrator と Tuxedo の非同期接続

WebLogic Process Integrator は、Tuxedo /Q に保証された非同期メッセージを送信します。

- JMS キューと Tuxedo /Q 間の 1 対 1 の関係。
- WebLogic Process Integrator は、メッセージを JMS キューに書き込みます。

- メッセージが JMS キューに入ると、tBridge はメッセージごとにターゲットの Tuxedo /Q にメッセージを移動します。
- 次のようなエラー メッセージは、指定した JMS エラー キューに転送されます。
  - インフラストラクチャ エラー。
  - XML/FML32 変換エラー。

非同期 Tuxedo /Q と WebLogic Process Integrator の接続

Tuxedo /Q は、WebLogic Process Integrator に保証された非同期メッセージを送信します。

- JMS キューと Tuxedo /Q 間の 1 対 1 の関係。
- Tuxedo は、メッセージを Tuxedo /Q に書き込みます。
- メッセージが Tuxedo /Q でコミットされると、そのメッセージは Tuxedo /T Domain Gateway を介して Weblogic Tuxedo コネクタ tBridge およびターゲット JMS キューに転送されます。
- Tuxedo から転送できないメッセージは、Tuxedo /Q エラー キューにエンキューされます。
- 次のようなエラーのメッセージは、指定した Tuxedo /Q エラー キューに転送されます。
  - インフラストラクチャ エラー。
  - FML32/XML 変換エラー。
- JMS キューのメッセージを待機するワークフローが作成されます。これは、既存のワークフロー インスタンスの Start ワークフロー ノードまたは Event ノードで定義されます。

Tuxedo と WebLogic Process Integrator の双方向非同期接続

Tuxedo は、WebLogic Process Integrator プロセス フローのブロッキング呼び出しを実行します。JMS から Tuxedo /Q へ、および Tuxedo /Q から JMS へ接続するには、2 つの非同期インスタンスを使用します。

8 WebLogic Tuxedo コネクタのトラブルシューティング

この章では、WebLogic Tuxedo コネクタのトラブルシューティング情報について説明します。

- WebLogic Tuxedo コネクタのモニタ
- よくある質問

WebLogic Tuxedo コネクタのモニタ

WebLogic Tuxedo コネクタは、WebLogic Server ログ ファイルを使用してログ情報を記録します。

トレース レベルの設定

サーバ ノードで WebLogic Tuxedo コネクタのトレース レベルを設定するには、`TraceLevel` パラメータを使用します。トレースを有効にするには、スタートアップ クラス ウィンドウの [引数] フィールドに、キーワードの `TraceLevel` と必要なトレース値を追加します。[引数] フィールドの引数を区切るには、カンマを使用します。

例 : `Arguments=BDMCONFIG=.\mydomain\wtc_config.xml,TraceLevel=100000`

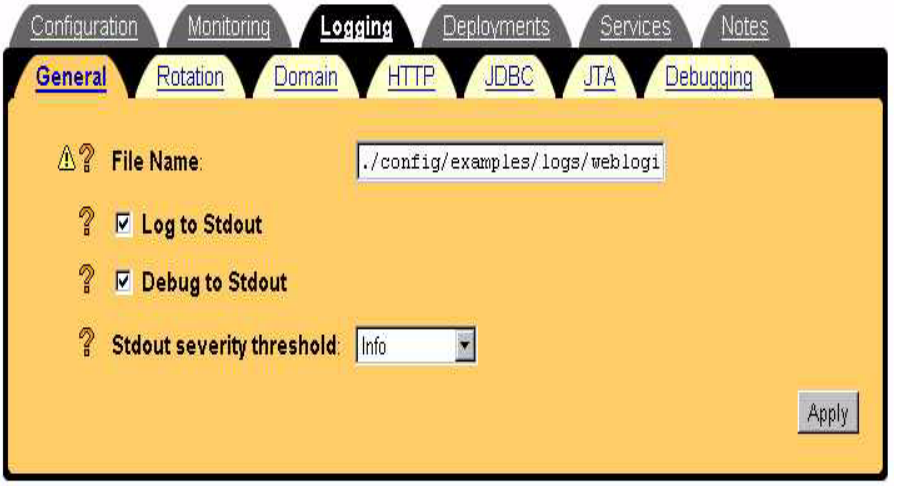
TraceLevel を設定するには、次の値を使用します。

値	トレースするコンポーネント	説明
10000	TBRIDGE_IO	tBridge の入力および出力
15000	TBRIDGE_EX	詳細な tBridge 情報
20000	GWT_IO	ATMI バーブなどのゲートウェイの入力と出力
25000	GWT_EX	詳細なゲートウェイ情報
50000	JAMTI_IO	低レベルの JAMTI 呼び出しなどの JAMTI の入力と出力
55000	JAMTI_EX	詳細な JAMTI 情報
60000	CORBA_IO	CORBA の入力および出力
65000	CORBA_EX	詳細な CORBA 情報
100000	すべてのコンポーネント	すべての WebLogic Tuxedo コネクタ コンポーネントの情報

コンソールの設定

必要なすべてのトレース情報を確実に log ファイルに書き込むには、サーバ ロギング設定によって次の内容が指定されていることを確認します。

- [Stdout へデバッグ情報出力] が選択されている。
- [Stdout 重大度しきい値] が Info に設定されている。



よくある質問

この節では、よくあるユーザからの質問に対する解決策について説明します。

コンフィグレーション ファイルが見つからない

エラー ログが、メッセージ「**Can not find the XML configuration file**」を表示します。何が原因ですか？

- スタートアップ クラスが正しくコンフィグレーションされているかどうか確認します。2-8 ページの「WebLogic Tuxedo Connector のスタートアップ クラスの作成」を参照してください。
- WebLogic Server コンフィグレーション ファイル `config.xml` をチェックします。スタートアップ クラスのエントリをチェックし、必要であれば編集します。スタートアップ クラスの例は次のとおりです。

```
<StartupClass
Arguments="BDMCONFIG=.\mydomain\dmconfig.xml,TraceLevel=100000"
ClassName="weblogic.wtc.gwt.WTCStartup" FailureIsFatal="true"
Name="MyWtcStartup Class" Targets="myserver"/>
```

EJB デプロイメント メッセージ

simpserv サンプルを構築すると、次のようなエラーが表示されます。

```
<date> <Error> <EJB> <EJB Deployment: Tolower has a class
weblogic.wtc.jatmi.tpserviceHome which is in the classpath. This class should only
be located in the ejb-jar file.>
```

このエラー メッセージは、このリリースの WebLogic Tuxedo コネクタでは無視できます。EJB は、EJB jar ファイルの EJB 呼び出しにすべてのインタフェースを要求します。しかし、WebLogic Tuxedo コネクタのインタフェースのいくつかは CLASSPATH で実装されるため、コンパイラが例外を発生します。EJB をデプロイするとき、コンパイラはクラスのいくつかが CLASSPATH で検出されるために再デプロイできないことを示します。

接続の問題

WebLogic Tuxedo コネクタと Tuxedo 間で接続の確立中に問題が発生します。どうしたらよいですか。

- Tuxedo リモート ドメインに対する WebLogic Tuxedo コネクタ コンフィグレーションをチェックします。リモート ドメインは、WebLogic Tuxedo コネクタでコンフィグレーションされたリモート ドメインの名前と一致している必要があります。

  例:Tuxedo DMCONFIG *DM_LOCAL_DOMAINS セクションで simpapp という名前がコンフィグレーションされている場合、この名前は WebLogic Tuxedo コネクタ コンフィグレーション ファイルの <AccessPointId> フィールドにある名前と一致している必要があります。

- WebLogic Tuxedo コネクタと Tuxedo のログ ファイルでエラー メッセージをチェックします。

- WebLogic Tuxedo コネクタのトレースを有効にし、接続性のテストを繰り返します。
- BEA カスタマ サポートに問い合わせます。

9 WebLogic Tuxedo Connector XML コンフィグレーション ファイル

この章では、WebLogic Tuxedo Connector XML コンフィグレーション ファイルの作成方法について説明し、要素の階層構造図を示します。

- XML コンフィグレーション ファイルの作成
- 要素の階層構造図

XML コンフィグレーション ファイルの作成

注意： XML コンフィグレーション ファイルの作成の詳細については、2-4 ページの「WebLogic Tuxedo Connector XML コンフィグレーション ファイルの作成」を参照してください。

テキスト エディタを使用して、WebLogic Tuxedo Connector XML コンフィグレーション ファイルを作成します。新しいコンフィグレーション ファイルを作成する最も効率的な方法は、サンプル `DBMCONFIG.xml` ファイルの 1 つをアプリケーションのニーズに合わせて修正します。

XML コンフィグレーション ファイルの例

この節では、simpapp サンプル アプリケーションの WebLogic Tuxedo Connector XML コンフィグレーション ファイルの例を紹介します。

コード リスト 9-1 simpapp BDMCONFIG.XML Configuration File

```
<?xml version="1.0"?>

<!DOCTYPE WTC_CONFIG SYSTEM
"http://www.bea.com/servers/wls610/dtd/wtc_config.dtd">

<!--Java and XML-->

<WTC_CONFIG >

<     BDMCONFIG >

<     T_DM_LOCAL_TDOMAIN AccessPoint="TDOM2">

           <WlsClusterName>Coolio</WlsClusterName>

           <AccessPointId>TDOM2</AccessPointId>

           <Type>TDOMAIN</Type>

           <Security>NONE</Security>

           <ConnectionPolicy>ON_DEMAND</ConnectionPolicy>

           <BlockTime>30</BlockTime>

           <NWAddr>[Network address of WTC domain]</NWAddr>

           <!-- Example address: //mydomain.acme.com:20304 -->

      </T_DM_LOCAL_TDOMAIN>

      <T_DM_REMOTE_TDOMAIN AccessPoint="TDOM1">

      <LocalAccessPoint>TDOM2</LocalAccessPoint>

           <AccessPointId>TDOM1</AccessPointId>

           <Type>TDOMAIN</Type>

           <NWAddr>[Network address of Tuxedo domain]</NWAddr>
```

```
<           !-- Example address: //mydomain.acme.com:20305 -->
      </T_DM_REMOTE_TDOMAIN>
      <T_DM_EXPORT ResourceName="TOLOWER"
      LocalAccessPoint="TDOM2">
           <EJBName>tuxedo.services.TOLOWERHome</EJBName>
      </T_DM_EXPORT>
      < T_DM_IMPORT
      ResourceName="TOUPPER"
           LocalAccessPoint="TDOM2"
             RemoteAccessPointList="TDOM1">
             <TranTime>600</TranTime>
     </T_DM_IMPORT>
</BDMCONFIG>
</WTC_CONFIG>
```

DOCTYPE 宣言は、wtc_config.dtd の場所を指定します。WebLogic Server が起動されると、WebLogic Tuxedo Connector XML コンフィグレーション ファイルでドキュメント タイプ定義（DTD）のエラーがあるかどうかチェックされます。

XML ファイルの有効性の検証

WTCValidateCF を使用して、コンフィグレーション ファイルの有効性を検証します。このユーティリティを使用すると、WebLogic Server を起動する前に WebLogic Tuxedo Connector XML コンフィグレーション ファイルの有効性を検証できます。

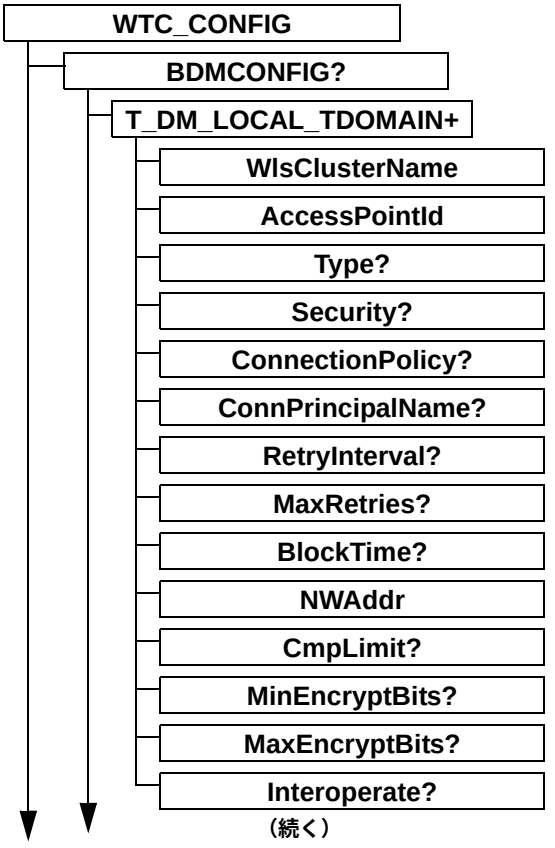
XML コンフィグレーション ファイルの有効性を検証するには、次のコマンドを入力します。

```
> java weblogic.wtc.gwt.WTCValidateCF your_XML_configuration_file
```

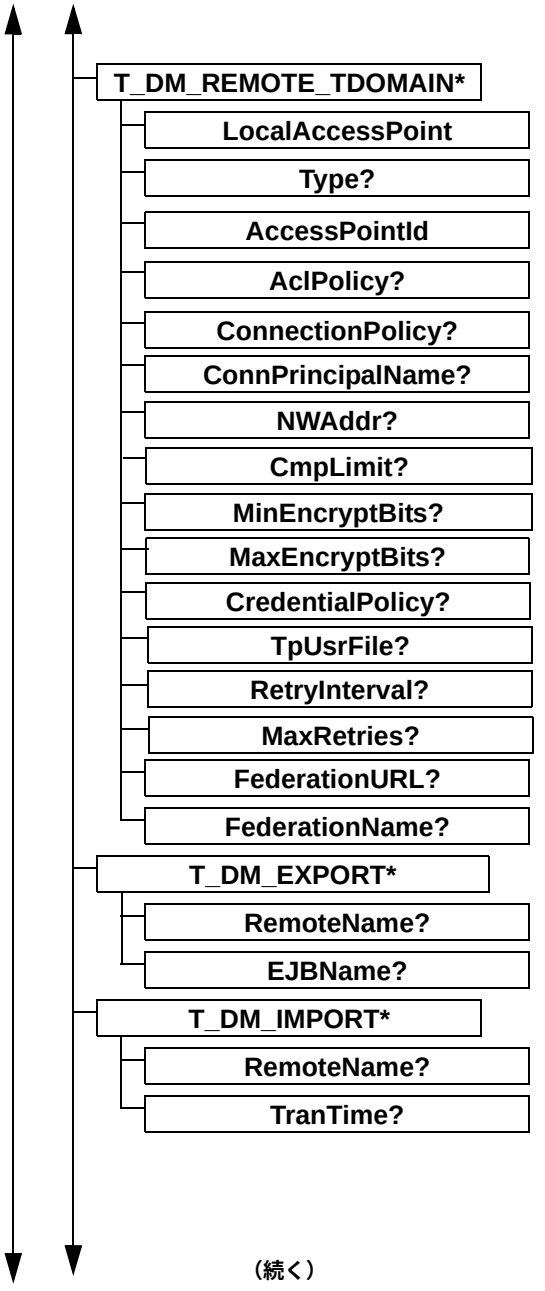
ここで、***your_XML_configuration_file*** は、XML コンフィグレーション ファイルの名前です。

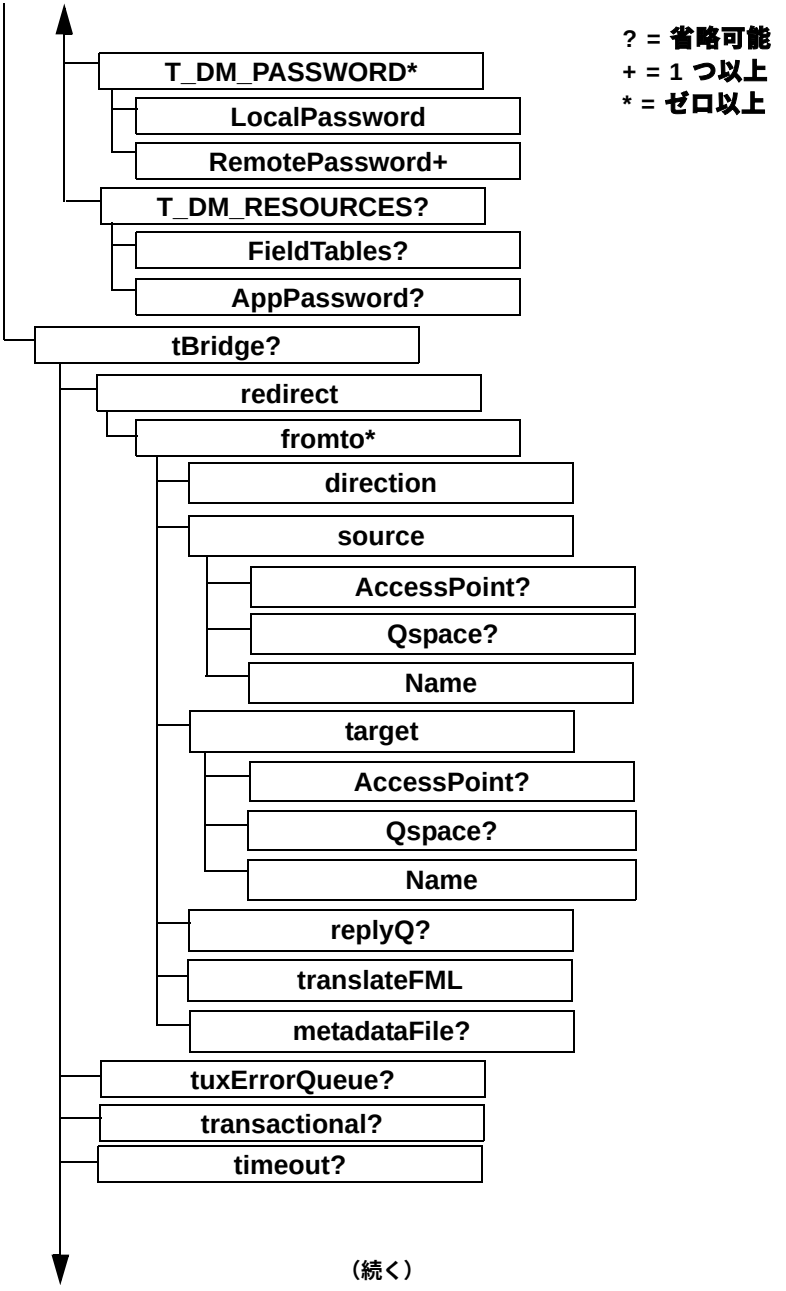
要素の階層構造図

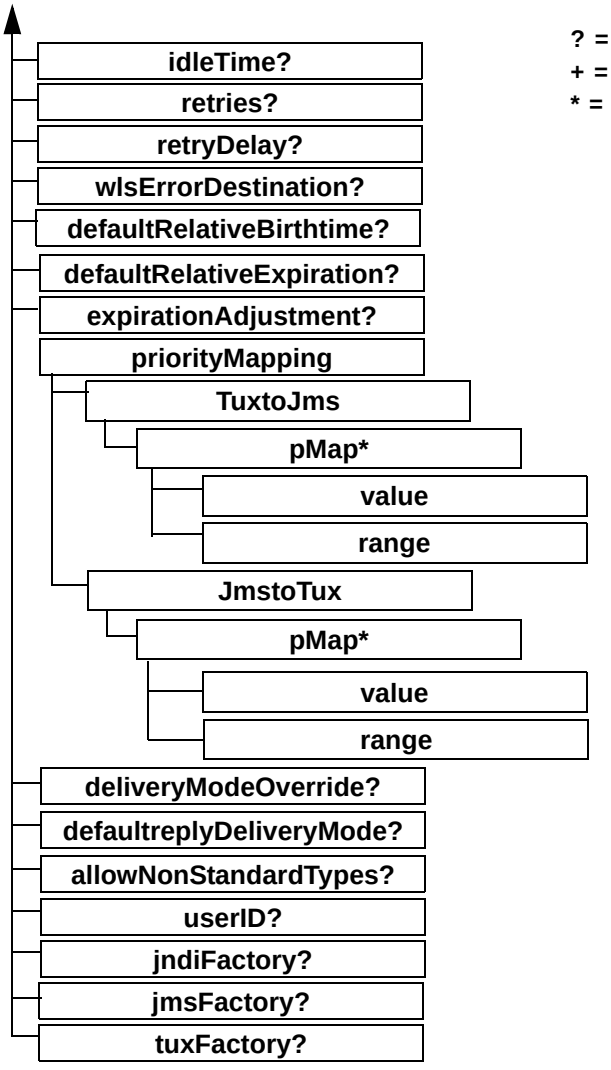
WebLogic Tuxedo コネクタコンフィグレーション ファイルは XML 要素群で構成されています。WTC_CONFIG 要素はトップレベルの要素であり、WTC_CONFIG 内のすべての要素は WTC_CONFIG 要素の子になります。子要素自身が子を持つ場合もあります。たとえば、次の図に示すように BDMCONFIG 要素には T_DM_LOCAL_TDOMAIN 子要素が含まれ、T_DM_LOCAL_TDOMAIN 要素には子要素が含まれます。



? = 省略可能
+ = 1 つ以上
* = ゼロ以上


? = 省略可能
+ = 1 つ以上
* = ゼロ以上
T_DM_REMOTE_TDOMAIN*
LocalAccessPoint
Type?
AccessPointId
AclPolicy?
ConnectionPolicy?
ConnPrincipalName?
NWAddr?
CmpLimit?
MinEncryptBits?
MaxEncryptBits?
CredentialPolicy?
TpUsrFile?
RetryInterval?
MaxRetries?
FederationURL?
FederationName?
T_DM_EXPORT*
RemoteName?
EJBName?
T_DM_IMPORT*
RemoteName?
TranTime?
（続く）


? = 省略可能
+ = 1 つ以上
* = ゼロ以上
T_DM_PASSWORD*
LocalPassword
RemotePassword+
T_DM_RESOURCES?
FieldTables?
AppPassword?
tBridge?
redirect
fromto*
direction
source
AccessPoint?
Qspace?
Name
target
AccessPoint?
Qspace?
Name
replyQ?
translateFML
metadataFile?
tuxErrorQueue?
transactional?
timeout?
（続く）


? = 省略可能
+ = 1 つ以上
* = ゼロ以上
idleTime?
retries?
retryDelay?
wlsErrorDestination?
defaultRelativeBirthtime?
defaultRelativeExpiration?
expirationAdjustment?
priorityMapping
TuxtoJms
pMap*
value
range
JmstoTux
pMap*
value
range
deliveryModeOverride?
defaultreplyDeliveryMode?
allowNonStandardTypes?
userID?
jndiFactory?
jmsFactory?
tuxFactory?

10 wtc_config.dtd

wtc_config.dtd ファイルは、WebLogic Tuxedo コネクタ XML コンフィグレーション ファイルで使う要素と属性を定義するために使用します。

wtc_config.dtd

```
<!ELEMENT WTC_CONFIG (
              BDMCONFIG?,
              tBridge?)>
<!ELEMENT BDMCONFIG (
              T_DM_LOCAL_TDOMAIN+,
              T_DM_REMOTE_TDOMAIN*,
              T_DM_EXPORT*,
              T_DM_IMPORT*,
              T_DM_PASSWORD*,
              T_DM_RESOURCES?)>
<!ELEMENT T_DM_LOCAL_TDOMAIN (
              WlsClusterName,
              AccessPointId,
              Type?,
              Security?,
              ConnectionPolicy?,
              ConnPrincipalName?,
              RetryInterval?
              MaxRetries?,
              BlockTime?,
```

```
                NWAddr,
                CmpLimit?,
                MinEncryptBits?,
                MaxEncryptBits?,
                Interoperate?)>
<!ATTLIST T_DM_LOCAL_TDOMAIN
                AccessPoint CDATA #REQUIRED>
<!ELEMENT WlsClusterName (#PCDATA)>
<!ELEMENT AccessPointId (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT Security (#PCDATA)>
<!ELEMENT ConnectionPolicy (#PCDATA)>
<!ELEMENT RetryInterval (#PCDATA)>
<!ELEMENT MaxRetries (#PCDATA)>
<!ELEMENT ConnPrincipalName (#PCDATA)>
<!ELEMENT NWAddr (#PCDATA)>
<!ELEMENT CmpLimit (#PCDATA)>
<!ELEMENT MinEncryptBits (#PCDATA)>
<!ELEMENT MaxEncryptBits (#PCDATA)>
<!ELEMENT Interoperate (#PCDATA)>
<!ELEMENT BlockTime (#PCDATA)>
<!ELEMENT T_DM_REMOTE_TDOMAIN (
                LocalAccessPoint,
                AccessPointId,
                Type?,
                AclPolicy?,
                ConnectionPolicy?,
                ConnPrincipalName?,
                NWAddr,
```

```
                FederationURL?,
                FederationName?,
                CmpLimit?,
                MinEncryptBits?,
                MaxEncryptBits?,
                CredentialPolicy?,
                TpUsrFile?
                RetryInterval?
                MaxRetries?)>
<!ATTLIST T_DM_REMOTE_TDOMAIN
                AccessPoint CDATA #REQUIRED>
<!ELEMENT LocalAccessPoint (#PCDATA)>
<!ELEMENT AclPolicy (#PCDATA)>
<!ELEMENT CredentialPolicy (#PCDATA)>
<!ELEMENT FederationURL (#PCDATA)>
<!ELEMENT FederationName (#PCDATA)>
<!ELEMENT TpUsrFile (#PCDATA)>
<!ELEMENT T_DM_EXPORT (RemoteName?, EJBName?)>
<!ATTLIST T_DM_EXPORT
                ResourceName     CDATA #REQUIRED
                LocalAccessPoint CDATA #REQUIRED>
<!ELEMENT RemoteName (#PCDATA)>
<!ELEMENT EJBName (#PCDATA)>
<!ELEMENT T_DM_IMPORT (RemoteName?, TranTime?)>
<!ATTLIST T_DM_IMPORT
                ResourceName          CDATA #REQUIRED
                LocalAccessPoint      CDATA #REQUIRED
                RemoteAccessPointList CDATA #REQUIRED>
<!ELEMENT TranTime (#PCDATA)>
```

```
<!ELEMENT T_DM_PASSWORD (LocalPassword, RemotePassword+)>
<!ATTLIST T_DM_PASSWORD
              LocalAccessPoint      CDATA #REQUIRED
              RemoteAccessPointList CDATA #REQUIRED>
<!ELEMENT LocalPassword (#PCDATA)>
<!ATTLIST LocalPassword
              IV CDATA #REQUIRED>
<!ELEMENT RemotePassword (#PCDATA)>
<!ATTLIST RemotePassword
              IV CDATA #REQUIRED>
 <!ELEMENT T_DM_RESOURCES (
              FieldTables?,
              AppPassword?)>
<!ELEMENT FieldTables (FldTblClass+)>
<!ELEMENT FldTblClass (#PCDATA)>
<!ATTLIST FldTblClass
              Type (fml16 | fml32) #REQUIRED>
<!ELEMENT AppPassword (#PCDATA)>
<!ATTLIST AppPassword
              IV CDATA #REQUIRED>
<!ELEMENT tBridge (
              redirect,
              transactional?,
              timeout?,
              idleTime?,
              retries?,
              retryDelay?,
              wlsErrorDestination?,
              tuxErrorQueue?,
```

```
                defaultRelativeBirthtime?,
                defaultRelativeExpiration?,
                expirationAdjustment?,
                priorityMapping?,
                deliveryModeOverride?,
                defaultReplyDeliveryMode?,
                userID?,
                allowNonStandardTypes?,
                jndiFactory,
                jmsFactory,
                tuxFactory)>
<!ELEMENT direction (#PCDATA)>
<!ELEMENT translateFML (#PCDATA)>
<!ELEMENT metadataFile (#PCDATA)>
<!ELEMENT AccessPoint (#PCDATA)>
<!ELEMENT Qspace (#PCDATA)>
<!ELEMENT Name  (#PCDATA)>
<!ELEMENT ReplyQ (#PCDATA)>
<!ELEMENT source (
                AccessPoint?,
                Qspace?,
                Name)>
<!ELEMENT target (
                AccessPoint?,
                Qspace?,
                Name)>
<!ELEMENT fromto (
                direction,
                source,
                target,
```

```
                ReplyQ?,
                translateFML,
                metadataFile?)>
<!ELEMENT redirect (fromto*)>
<!ELEMENT transactional (#PCDATA)>
<!ELEMENT timeout (#PCDATA)>
<!ELEMENT idleTime (#PCDATA)>
<!ELEMENT retries (#PCDATA)>
<!ELEMENT retryDelay (#PCDATA)>
<!ELEMENT wlsErrorDestination (#PCDATA)>
<!ELEMENT tuxErrorQueue (#PCDATA)>
<!ELEMENT defaultRelativeBirthtime (#PCDATA)>
<!ELEMENT defaultRelativeExpiration (#PCDATA)>
<!ELEMENT expirationAdjustment (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT range (#PCDATA)>
<!ELEMENT pMap (value,range)>
<!ELEMENT TuxtoJms (pMap*)>
<!ELEMENT JmstoTux (pMap*)>
<!ELEMENT priorityMapping (
                TuxtoJms,
                JmstoTux)>
<!ELEMENT deliveryModeOverride (#PCDATA)>
<!ELEMENT defaultReplyDeliveryMode (#PCDATA)>
<!ELEMENT userID (#PCDATA)>
<!ELEMENT allowNonStandardTypes (#PCDATA)>
<!ELEMENT jndiFactory (#PCDATA)>
<!ELEMENT jmsFactory (#PCDATA)>
```

```
<!ELEMENT tuxFactory (#PCDATA)>
```

11 wtc_config.dtd の要素および属性

次の節では、wtc_config.dtd に含まれる要素および属性のリファレンス情報について説明します。

- WTC_CONFIG
- BDMCONFIG
- tBridge

WTC_CONFIG

WTC_CONFIG は、WebLogic Tuxedo Connector デプロイメント記述子のルートです。WTC_CONFIG は、次に示す 2 つの子を持っています。

- BDMCONFIG
- tBridge

BDMCONFIG

BDMCONFIG 要素は、WebLogic Tuxedo Connector でドメインをコンフィグレーションする方法についての情報を提供します。コンフィグレーション パラメータは、Tuxedo ドメインで使用される DM_MIB 属性とクラスに類似しています。

T_DM_LOCAL_TDOMAIN

T_DM_LOCAL_DOMAIN は、他のドメインで認識されるローカル ドメインのビューを提供します。

属性	説明	タイプ	使用
AccessPoint	コンフィグレーション XML ファイルでドメインを識別するために使用するラベル。ラベルは、コンフィグレーション XML ファイルの T_DM_LOCAL_TDOMAIN および T_DM_REMOTE_TDOMAIN の AccessPoint 名のスコープ内で一意でなければならない。 例 :TDOM2	CDATA	#REQUIRED

WlsClusterName

注意 : WebLogic Tuxedo Connector はこのリリースの WebLogic Server でクラスタ化をサポートしていませんが、WlsClusterName を指定する必要があります。

必須。WlsClusterName は、このローカル アクセス ポイントがある WSL ドメインの WebLogic Server クラスタの名前を提供します。

例 : cluster20

AccessPointId

必須。別のドメインに対する接続を確立するときにドメインを識別するために使用される、接続プリンシパル名を指定します。

- T_DM_LOCAL_TDOMAIN の AccessPointId は、Tuxedo の DMCONFIG ファイルの *DM_REMOTE_DOMAINS セクションの対応する DOMAINID と一致していなければなりません。
- T_DM_REMOTE_TDOMAIN の AccessPointId は、Tuxedo の DMCONFIG ファイルの *DM_LOCAL_DOMAINS セクションの対応する DOMAINID と一致していなければなりません。

例 : TDOM2

Type

省略可能。指定する場合、値は文字列 TDOMAIN でなければなりません。

Security

注意 : Tuxedo 6.5 ユーザはセキュリティ パラメータを NONE に設定する必要があります。

省略可能。実行するアプリケーション セキュリティのタイプを指定します。このパラメータの有効な値は、NONE、APP_PW、または DM_PW です。

- NONE: セキュリティは使用されません。これがデフォルト値です。
- APP_PW: リモート ドメインからの接続が確立されたときに、パスワード セキュリティが強化されます。アプリケーション パスワードが、T_DM_PASSWORD 要素に定義されている必要があります。
- DM_PW: リモート ドメインからの接続が確立されたときに、ドメイン パスワード セキュリティが強化されます。ドメイン パスワードが、T_DM_PASSWORD 要素に定義されている必要があります。

ConnectionPolicy

省略可能。ローカル ドメインがリモート ドメインとの接続の確立を試行する場合の条件を指定します。

- ローカル ドメインに有効な値は、ON_DEMAND、ON_STARTUP、または INCOMING_ONLY です。
- リモート ドメインに有効な値は、ON_DEMAND、ON_STARTUP、INCOMING_ONLY、または LOCAL です。
- デフォルト設定は ON_DEMAND です。
  - ON_DEMAND: リモート サービスへのクライアント リクエストまたは管理接続コマンドのいずれかによってリクエストされたときのみ、接続が試行されます。

- ON_STARTUP: ドメイン ゲートウェイは、ゲートウェイ サーバの初期化時にそのリモート ドメイン アクセス ポイントで接続を確立します。リモート サービス（このローカル アクセス ポイントのドメイン ゲートウェイによって JNDI に通知されるサービス）は、接続がそのリモート ドメイン アクセス ポイントに正常に確立されたときのみ通知されます。リモート ドメイン アクセス ポイントへのアクティブな接続がない場合、リモート サービスはサスペンドされます。デフォルトでは、この接続ポリシーは 60 秒ごとに失敗した接続を再試行します。アプリケーション固有の値を指定するには、MaxRetry および RetryInterval 要素を使用します。
- INCOMING_ONLY: ドメイン ゲートウェイは、起動時にリモート ドメイン アクセス ポイントへの初期接続を行わず、リモート サービスは最初にサスペンドされます。ドメイン ゲートウェイはリモート ドメイン アクセス ポイントからの受信時接続に使用可能で、リモート サービスはこのローカル ドメイン アクセス ポイントのドメイン ゲートウェイが受信時接続を受け付けたときに通知されます。接続の再試行処理は実行できません。
- LOCAL: リモート ドメイン接続ポリシーが、明示的にローカル ドメインの ConnectionPolicy 属性値にデフォルト設定されることを示します。リモート ドメインの ConnectionPolicy が定義されていない場合、システムは関連付けられているローカル ドメイン（LocalAccessPoint によって指定）で指定された設定を使用します。

RetryInterval

省略可能。リモート ドメイン アクセス ポイントへの接続を確立するために自動的に行われる接続の時間間隔（単位 : 秒）です。ConnectionPolicy が ON_STARTUP に設定されているときだけ使用してください。

- 最小値 : 0
- 最大値 : 2147483647
- デフォルト設定 : 60

MaxRetries

省略可能。ドメイン ゲートウェイが、リモート ドメイン アクセス ポイントへの接続の確立を試行する回数です。ConnectionPolicy が ON_STARTUP に設定されているときだけ使用してください。

- 最小値 : 0
- 最大値 : 2147483647
- デフォルト値 : 2147483647

最大値は、接続が確立されるまで処理を再試行する場合に使用します。最小値は、自動再試行メカニズムを無効にする場合に使用します。

ConnPrincipalName

注意 : ConnPrincipalName は、このリリースの WebLogic Server ではサポートされていません。

省略可能。接続プリンシパル名の識別子を指定します。これは、別のドメインへの接続を確立するときに、ドメインを識別するためのプリンシパル名です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアを実行しているタイプ TDOMAIN のドメインにのみ適用されます。

- この要素を指定しない場合、接続プリンシパル名はこのドメインの AccessPoint 要素にデフォルト設定されます。

NWAddr

注意 : T_DM_LOCAL_DOMAIN に対する NWAddr をコンフィグレーションする場合、使用するポート番号は、他の WebLogic Server プロセスに割り当てられているどのポート番号とも異なっていなければなりません。たとえば、WebLogic Server のリスン ポートに `//mymachine:7001` が割り当てられている場合、NWAddr を `//mymachine:7001` に設定することはできません。

必須。ローカル ドメイン ゲートウェイのネットワーク アドレスです。次のフォーマットのうちの 1 つを使用して、TCP/IP アドレスを指定します。

- //hostname:port_number

- //#.#.#.#:port_number

hostname を使用する場合、ドメインはローカル名解決機能（通常 DNS）を使用して、ホスト名のアドレスを検索します。ドットで区切った数値のフォーマットを使用する場合、各 # は 0 ～ 255 までの数値でなければなりません。このドットで区切った数値は、ローカル マシンの IP アドレスを表します。port_number は、ドメイン プロセスが受信するリクエストをリスンする TCP ポート番号です。

CmpLimit

省略可能。リモート ドメインへのデータ送信時に使用する圧縮しきい値を指定します。このサイズより大きいアプリケーション バッファは圧縮されます。

- デフォルト値 :2,147,483,647 バイト

MinEncryptBits

注意 : 40 の MinEncryptBits 値は、BEA Tuxedo 7.1 以降のソフトウェアを実行しているタイプ TDOMAIN のドメインにのみ適用されます。

省略可能。このドメインのネットワーク リンクの確立時に使用する最小レベルの暗号キー長（単位 : ビット）を指定します。このパラメータの有効な値は、0、40、56、および 128 です。

- デフォルト値 :0 ビット
- 値 0 : 暗号は使用されません。
- この暗号の最小レベルが一致しない場合、ネットワーク リンクは失敗します。

MaxEncryptBits

注意 : 40 の MaxEncryptBits 値は、BEA Tuxedo 7.1 以降のソフトウェアを実行しているタイプ TDOMAIN のドメインにのみ適用されます。

省略可能。このドメインのネットワーク リンクの確立時に使用する最大レベルの暗号キー長（単位 : ビット）を指定します。このパラメータの有効な値は、0、40、56、および 128 です。

- デフォルト値 :128 ビット
- 値 0 : 暗号は使用されません。

Interoperate

注意 : 省略可能。ローカル ドメインが、Tuxedo リリース 6.5 に基づいたリモート ドメインと相互運用するかどうかを指定します。Tuxedo 6.5 は、セキュリティ マッピングをサポートするために必要なセキュリティ インフラストラクチャを持っていません。セキュリティ機能が必要で、かつ WebLogic Tuxedo Connector を使用する場合は、Tuxedo 7.1 以上にアップグレードする必要があります。

このパラメータの有効な値は、Yes または No です。

- Yes:Tuxedo 6.5 と相互運用します。
- No:Tuxedo 7.1 以降のドメインで機能する。
- デフォルト値 : No

BlockTime

省略可能。ブロッキング呼び出しに許可される最大待機時間（単位 : 秒）を指定します。

T_DM_REMOTE_TDOMAIN

T_DM_LOCAL_DOMAIN は、ローカル ドメインで認識されるリモート ドメインのビューを提供します。

属性	説明	タイプ	使用
AccessPoint	コンフィグレーション XML ファイルでドメインを識別するために使用するラベル。このラベルは、コンフィグレーション XML ファイルの T_DM_LOCAL_TDOMAIN および T_DM_REMOTE_TDOMAIN の AccessPoint 名のスコープ内で一意でなければならない。 例 :TDOM3	CDATA	#REQUIRED

LocalAccessPoint

必須。リモート ドメインがアクセスするローカル ドメイン名です。

例 :TDOM2

AclPolicy

注意 : Interperate パラメータが Yes に設定されている場合、AclPolicy は無視されます。詳細については、11-7 ページの「Interoperate」を参照してください。

省略可能。リモート ドメインからのリクエストに対して、着信アクセス制御リスト（ACL）ポリシーを指定します。このパラメータの有効な値は、LOCAL または GLOBAL です。

- LOCAL : ローカル ドメインは、指定のリモート ドメインから受信したサービス リクエストの ID を、指定のリモート ドメインのローカル プリンシパル名で指定されたプリンシパル名に変更します。
- GLOBAL : ローカル ドメインは、ID を変更せずにサービス リクエストを渡します。
- デフォルト値 :LOCAL

CredentialPolicy

注意： Interperate パラメータが Yes に設定されている場合、CredentialPolicy は無視されます。詳細については、11-7 ページの「Interoperate」を参照してください。

省略可能。リモート ドメインへのリクエストに対して、発信アクセス制御リスト（ACL）ポリシーを指定します。このパラメータの有効な値は、LOCAL または GLOBAL です。

- LOCAL : リモート ドメインは、ローカル ドメインから受信したサービス リクエストの ID を、このリモート ドメインのローカル プリンシパル名で指定されたプリンシパル名に設定します。
- GLOBAL : リモート ドメインは、変更せずにサービス リクエストを渡します。
- デフォルト値 :LOCAL

FederationURL

JNDI に結合される外部ネーム サービスの URL です。

省略すると、WebLogic Tuxedo Connector は外部ドメインに CosNaming サーバがあると見なします。WebLogic Tuxedo Connector は、TGIOP を使用して CosNaming サーバと結合します。CORBA 以外のサービス プロバイダと結合することができます。

FederationName

外部ネーム サービスと結合するコンテキストです。省略すると、結合ポイントは *tuxedo.domains* になります。

TpUsrFile

省略可能。uid/gid 情報を含むユーザ パスワード ファイルへの絶対パスです。これは、リモート ドメインで Tuxedo `tpusradd` ユーティリティによって生成されるファイルと同じです。正しい認可、認証、および監査を行うために、このファイルにはユーザ名、uid および gid 情報が含まれ、有効になっている必要があります。

T_DM_EXPORT

T_DM_EXPORT は、ローカル ドメインによってエクスポートされるサービスに関する情報を提供します。

- 指定しない場合、すべてのローカル ドメインは、デフォルト JNDI ルックアップ ルールに従ってすべてのサービスへのリクエストを受け付けます（EJBName を参照）。
- セクションを定義する場合は、それをリモート ドメインからリクエストされたローカル サービスのセットを制限するために使用します。

属性	説明	タイプ	使用
ResourceName	ResourceName 属性は、エクスポートされたサービス エントリを記述する。	CDATA	#REQUIRED
LocalAccessPoint	ローカル アクセス ポイント名。	CDATA	#REQUIRED

RemoteName

省略可能。サービスのリモート名です。

- 指定しない場合、ResourceName 属性が使用されます。

EJBName

省略可能。サービスの呼び出し時に使用する EJB ホーム インタフェースの完全な名前です。

- この要素を指定しない場合、使用されるデフォルト インタフェースは tuxedo.services.servicenameHome となります。

  例 : 呼び出されるサービスが TOUPPER で、EJBName 属性を指定しない場合、JNDI でルックアップされるホーム インタフェースは tuxedo.service.TOUPPERHome になります。

T_DM_IMPORT

T_DM_IMPORT は、インポートされたサービスおよびリモート ドメインで使用可能なサービスに関する情報を提供します。T_DM_IMPORT がコンフィグレーションされていない場合、リモート ドメインはすべてのリモート サービスを処理します。

属性	説明	タイプ	使用
ResourceName	ResourceName 属性は、インポートされたサービス エントリを記述する。 ResourceName、LocalAccessPoint、および RemoteAccessPointList 属性の組み合わせは、このタイプのすべてのオブジェクト間で一意でなければならない。 例 : //simpapp	CDATA	#REQUIRED
LocalAccessPoint	サービスが提供されるローカル アクセス ポイントを指定する。 例 :TDOM2	CDATA	#REQUIRED
RemoteAccessPointList	リソースがインポートされるリモート ドメイン アクセス ポイントを示す、カンマで区切られたフェイルオーバ リスト。 例 : TDOM3,TDOM4,TDOM5	CDATA	#REQUIRED

TranTime

注意 : TranTime は、このリリースの WebLogic Tuxedo Connector ではサポートされていません。

省略可能。関連付けられているサービスで開始されたトランザクションのデフォルト タイムアウト値（単位 : 秒）を指定します。これは、トランザクションの開始からロールバックまたはコミットの実行までのトータルの時間です。0 は最大タイムアウト値を表します。

- 最小値 : 0
- 最大値 : 2147483648
- デフォルト値 : 30

T_DM_PASSWORD

T_DM_PASSWORD クラスは、タイプ TDOMAIN のアクセス ポイントの相互ドメイン認証に関するコンフィグレーション情報を示します。

属性	説明	タイプ	使用
LocalAccessPoint	パスワードが適用されるローカル ドメイン アクセス ポイントの名前。 例 :TDOM2	CDATA	#REQUIRED
RemoteAccessPoint	パスワードが適用されるリモート ドメイン アクセス ポイントの名前。 例 : TDOM3	CDATA	#REQUIRED

LocalPassword

必須。genpasswd ユーティリティから返される暗号化されたローカル パスワードです。このパスワードは、LocalAccessPoint によって識別されるローカル ドメイン アクセス ポイントと、RemoteAccessPoint によって識別されるリモート ドメイン アクセス ポイント間の接続を認証するために使用します。

属性	説明	タイプ	使用
IV	ローカル パスワードを暗号化するために使用する初期化ベクトル。	CDATA	#REQUIRED

RemotePassword

必須。genpasswd ユーティリティから返される暗号化されたリモート パスワードです。このパスワードは、LocalAccessPoint によって識別されるローカル ドメイン アクセス ポイントと、RemoteAccessPoint によって識別されるリモート ドメイン アクセス ポイント間の接続を認証するために使用します。

属性	説明	タイプ	使用
IV	リモート パスワードを暗号化するために使用する初期化ベクトル。	CDATA	#REQUIRED

T_DM_RESOURCES

ドメインのグローバル フィールド テーブル クラスとアプリケーション パスワードを指定するために使用します。

FieldTables

省略可能。WebLogic Tuxedo Connector ユーザが使用可能な FldTbls テーブルのリストを示します。これらのテーブルは、mkfldclass または mkfldclass32 ユーティリティを使用して作成され、Java クラスにコンパイルされます。FML および FML32 処理に使用される配列は、getFldTbls(fml16|fml32) ユーティリティによってアクセス可能です。

FldTblClass

必須。クラス ローダーによってロードされ、FldTbl 配列に追加されるフィールド テーブル クラスの名前です。使用されるクラス名は、必要なクラスの完全修飾名です。

属性	説明	使用
タイプ	FML / FML32 処理に使用されるクラス タイプを示すフラグ。有効な値は次のとおり。 ■ fml16 ■ fml32	#REQUIRED

AppPassword

省略可能。genpasswd ユーティリティから返される暗号化されたアプリケーション パスワードです。このグローバル パスワードは、すべてのドメイン アクセス ポイント間の接続を認証するために使用します。

属性	説明	タイプ	使用
IV	グローバル パスワードを暗号化するために使用する初期化ベクトル。	CDATA	#REQUIRED

tBridge

注意： tBridge は、定義された各リダイレクションの新しいスレッドを起動することによって、1 つまたは複数のリダイレクションを処理します。最低でも 1 つのリダイレクションを指定しないと、tBridge は失敗し、エラーログが記録されます。

省略可能な tBridge をコンフィグレーションすることによって、WebLogic Server と Tuxedo 間における XML メッセージの双方向転送が可能になります。

direction

必須。データ フローの方向を指定します。有効なパラメータ値は、JmsQ2TuxQ、TuxQ2JmsQ、JmsQ2TuxS です。

- JmsQ2TuxQ :JMS から TUXEDO /Q
- TuxQ2JmsQ :TUXEDO /Q から JMS
- JmsQ2TuxS :JMS から JMS への Tuxedo サービス応答

fromto

必須。ソース、ターゲット、方向、およびメッセージの転送を指定するために使用する要素です。

redirect

必須。メッセージをリダイレクトするために使用する要素です。

source

必須。メッセージのソース位置です。

target

必須。メッセージを配置するターゲット位置です。

AccessPoint

省略可能。ドメイン コンフィグレーションの T_DM_LOCAL_TDOMAIN および T_DM_REMOTE_TDOMAIN エントリ名の範囲内で一意な識別子。

例 :TDOM2

Qspace

省略可能。ソース位置またはターゲット位置の Qspace の名前です。

Name

必須。JMS キュー名、Tuxedo キュー名、または Tuxedo サービス名の名前です。

ReplyQ

省略可能。特に Tuxedo サービスへの同期呼び出しを行う JMS キューの名前です。応答は JMS ReplyQ に返されます。

metadataFile

注意 : この要素は、このリリースの WebLogic Tuxedo Connector ではサポートされていません。

省略可能。メタデータ ファイルの URL です。

translateFML

注意 : WLXT パラメータ値は、このリリースの WebLogic Tuxedo Connector ではサポートされていません。

必須。XML/FML 変換のタイプを指定します。有効なパラメータ値は、NONE、FLAT、WLXT です。

- NO : データ変換は実行されません。転送の方向に応じて、`TextMessage` は STRING に（またはその逆に）マップされます。`BytesMessage` は、CARRAY に（またはその逆に）マップされます。他のすべてのデータ型の場合、リダイレクションは失敗します。
- FLAT : メッセージ ペイロードは、WebLogic Tuxedo Connector の組み込みトランスレータを使用して変換されます。
- WLXT : 変換は、XML-to-nonXML WL XML Translator（WLXT）によって実行されます。指定された `metadataFile` URL は、変換を実行する WLXT 外部メソッドを呼び出すために渡されます。
- デフォルト値 :NO

transactional

注意 : この要素は、WebLogic Tuxedo Connector のこのリリースではサポートされていません。

省略可能。ソース位置からのメッセージの検索時およびターゲット位置へのメッセージの配置時に、トランザクションの使用を指定します。有効なパラメータ値は、YES、NO です。

- YES : トランザクションは、両方の処理で使用されます。
- NO : トランザクションは、いずれの処理でも使用されません。

timeout

注意 : ソース位置からメッセージを検索するのに使用するトランザクションのタイムアウトは、タイムアウトよりも長くなります。このパラメータは、タイムアウトの効果的な長さをリダイレクション全体に反映します。

省略可能。ターゲット位置にメッセージを配置するために使用するトランザクション タイムアウト値（単位 : 秒）を指定します。`transactional` パラメータが YES に設定された場合は必須です。

- デフォルト値は 60 です。
- 値は 0 または正の整数でなければなりません。
- 0 は無限に待機することを示します。

idleTime

注意 : この要素は、このリリースの WebLogic Tuxedo Connector ではサポートされていません。

省略可能。新しいメッセージのソース位置をチェックするまで待機するアイドル時間（単位 : 秒）を指定します。メッセージが検出されると、ソース位置にあるすべてのメッセージが順次処理されます。

- デフォルト値は 0 です。
- 値は 0 または正の整数でなければなりません。

retries

省略可能。指定したエラー位置にメッセージを配置し、エラーをロギングするまでに、メッセージのリダイレクトを試行する回数を指定します。

- デフォルト値は 0 です。
- 値は 0 または正の整数でなければなりません。

retryDelay

注意： `retryDelay` 中、メッセージを処理するスレッドは他のメッセージをリダイレクトできません。

省略可能。メッセージをリダイレクトするまでに待機する最小時間（単位：ミリ秒）を指定します。

- デフォルト値は 10 です。
- 値は正の整数でなければなりません。

wlsErrorDestination

省略可能。メッセージがリダイレクトできない場合、WebLogic Server JMS メッセージを格納するために使用する場所の名前です。

- `wlsErrorDestination` を指定しない場合、リダイレクトできないメッセージはすべて失われます。
- 何らかの理由により `wlsErrorDestination` にメッセージを配置できない場合、エラー ログが記録され、メッセージが失われます。

tuxErrorQueue

省略可能。Tuxedo/Q ソース キューにリダイレクトできないメッセージを格納するために使用する Tuxedo キューの名前です。このキューは、ソース キューと同じキュー スペースにあります。

- `tuxErrorDestination` を指定しない場合、リダイレクトできないメッセージはすべて失われます。

- 何らかの理由により tuxErrorQueue にメッセージを配置できない場合、エラーログが記録され、メッセージが失われます。

defaultRelativeBirthtime

注意： この要素は、このリリースの WebLogic Tuxedo Connector ではサポートされていません。

省略可能。JMS_BEA_TuxGtway_Tuxedo_Birthtime プロパティが設定されていない場合、WebLogic Server から Tuxedo にリダイレクトされるメッセージの deq_time（単位 : 秒）を指定します。

defaultRelativeExpiration

注意： この要素は、このリリースの WebLogic Tuxedo Connector ではサポートされていません。

省略可能。JMSExpiration プロパティが設定されていない場合、WebLogic Server から Tuxedo にリダイレクトされるメッセージの exp_time（単位 : 秒）を指定します。

Tuxedo/Q メッセージから有効期限を取得することはできません。Tuxedo/Q にリダイレクトされるメッセージの有効期限を設定するには、defaultRelativeExpiration を使用します。

expirationAdjustment

注意： この要素は、このリリースの WebLogic Tuxedo Connector ではサポートされていません。

省略可能。ターゲット位置に配置される前に、メッセージの有効期限に追加される時間（単位 : 秒）を指定します。

- この値は、メッセージの有効期限がゼロでない場合、ソース位置から検索される前に有効期限に追加されます。
- tBridge を使用した結果として発生するネットワークまたは処理のレイテンシを処理するために使用します。

priorityMapping

必須。JMS と Tuxedo 間の優先度をマップする方法を提供します。

- JMS には 10 の優先度があります（0 ～ 9）。
- Tuxedo/Q には 100 の優先度があります（1 ～ 100）。

priorityMapping セクションは、Tuxedo および JMS サブシステム間で優先度をマップするメカニズムを提供します。2 つのマッピング方向があります。

- `JmstoTux`
- `TuxtoJms`

次の `value:range` の組み合わせで示すように、すべての値にデフォルトが提供されています。デフォルト値は、マッピング方向の `pMap` 組み合わせを記述することによってオーバーライドされます。`value` は特定の入力優先度を指定し、`range` は結果出力優先度のシーケンシャル グループを指定します。

`JmstoTux`- 0:1 | 1:12 | 2:23 | 3:34 | 4:45 | 5:56 | 6:67 | 7:78 | 8:89 | 9:100

`TuxtoJms`- 1-10:0 | 11-20:1 | 21-30:2 | 31-40:3 | 41-50:4| 51-60:5 | 61-70:6 | 71-80:7 | 81-90:8 | 91-100:9

JmstoTux

必須。JMS から Tuxedo /Q メッセージ優先度への優先度マッピング方向を指定するのに使用します。

TuxtoJms

必須。Tuxedo /Q から JMS メッセージ優先度への優先度マッピング方向を指定するのに使用します。

pMap

必須。`pMap` 要素には、値と範囲の 2 つの要素が含まれます。`value` は特定の入力優先度を指定し、`range` は結果出力優先度のシーケンシャル グループを指定します。

range

必須。range は、結果出力優先度（pMap の右側）のシーケンシャル グループを指定します。

value

必須。value は、特定の入力優先度（pMap の左側）を指定します。

deliveryModeOverride

注意： deliveryModeOverride を指定しない場合、メッセージはソース位置から指定された配信モードと同じモードを使用して、ターゲット位置に配置されます。

省略可能。ターゲット位置にメッセージを配置するときに使用する配信モードを指定します。deliveryModeOverride 値は、メッセージに関連付けらている任意の配信モードをオーバーライドします。このパラメータの有効な値は、PERSIST、NONPERSIST です。

defaultReplyDeliveryMode

注意： defaultReplyDeliveryMode が指定されていない場合、または JMS_BEA_TuxGtway_Tuxedo_ReplyDeliveryMode が設定されていない場合、Tuxedo/Q サブシステムは Tuxedo に定義されるデフォルト セマンティクスを使用します。

省略可能。ターゲット位置にメッセージを配置するときに、メッセージに関連付けられている応答配信モードを指定します。JMS_BEA_TuxGtway_Tuxedo_ReplyDeliveryMode プロパティがメッセージに設定されていない場合、JMS から Tuxedo/Q にリダイレクトされるメッセージに対して、この要素を使用します。このパラメータの有効な値は、PERSIST、NONPERSIST、DEFAULT です。

userID

省略可能。セキュリティ オプションがコンフィグレーションされている場合、ACL チェックのために tBridge によって処理されるすべてのメッセージのユーザ ID を指定します。

- セキュリティ / 認証コンテキストがサブシステム間で渡されるまで、すべてのメッセージはこの ID を想定します。セキュリティ コンテキストが渡されるまでは、ソース位置から受信したメッセージの生成者を識別する安全な方法はありません。
- 引数のユーザは、ユーザ名またはユーザ ID 番号（uid）のいずれかで指定できます。

allowNonStandardTypes

省略可能。非標準データ型が tBridge の通過を許可されている場合に指定します。このパラメータの有効な値は、NO、YES です。

- NO : 非標準データ型は拒否され、指定したエラー位置に配置されます。
- YES : 非標準データ型は、元のタイプを示すタグと共に、BLOB としてターゲット位置に配置されます。

標準データ型は次のとおりです。

- ASCII テキスト（TextMessage、STRING）
- BLOB（BytesMessage、CARRAY）

jndiFactory

必須。JNDI ルックアップ ファクトリの名前です。

例 : `weblogic.jndi.WLInitialContextFactory`

jmsFactory

必須。JMS 接続ファクトリの名前です。

例 : `weblogic.jms.ConnectionFactory`

tuxFactory

必須。Tuxedo 接続ファクトリの名前です。

例：`tuxedo.services.TuxedoConnection`