# BEA WebLogic Integration™

## B2B Integration
## セキュリティの実装

リリース 7.0
マニュアルの日付 : 2002 年 6 月

## 著作権



## 限定的権利条項



## 商標または登録商標



**B2B Integration セキュリティの実装**

| パート番号 | 日付 | ソフトウェアのバージョン |
|---|---|---|
| なし | 2002 年 6 月 | 7.0 |

# 目次

## 4. セキュリティのコンフィグレーション

## 5. 否認防止性の実装



## 索引

# このマニュアルの内容

このマニュアルでは、WebLogic Integration B2B デプロイメントのためのセキュリティ方式を実装する方法について説明します。

このマニュアルの内容は以下のとおりです。

- 第 1 章「WebLogic Integration B2B セキュリティ入門」では、WebLogic Integration B2B セキュリティの概要、および基盤となる WebLogic Server セキュリティのしくみについて説明します。
- 第 2 章「トレーディング パートナの認証と認可」では、B2B ソフトウェアで使用されている認証および認可プロセスについて説明します。
- 第 3 章「キーストアのコンフィグレーション」では、SSL 認証および認可に使用されるトレーディング パートナの証明書を格納する WebLogic Server キーストアの作成とコンフィグレーションについて説明します。
- 第 4 章「セキュリティのコンフィグレーション」では、B2B トレーディング パートナおよび B2B 環境のためにセキュリティをコンフィグレーションする方法について説明します。
- 第 5 章「否認防止性の実装」では、ビジネス プロセスに否認防止性メカニズムを実装する方法について説明します。

# 対象読者

このマニュアルは主に、次のユーザを対象としています。

- WebLogic Integration デプロイメントのためのセキュリティ メカニズムを設計するビジネス アナリストおよびプログラマ
- B2B セキュリティを設定および管理するシステム管理者

WebLogic Integration B2B のアーキテクチャの概要については、『*B2B Integration 入門*』を参照してください。

# 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 ファイルずつ印刷できます。

このマニュアルの PDF 版は、WebLogic Integration のマニュアル CD にあります。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。

Adobe Acrobat Reader がない場合は、Adobe の Web サイト（`http://www.adobe.co.jp/`）で無料で入手できます。

# サポート情報

WebLogic Integration 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、WebLogic Integration B2B の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、WebLogic Integration 7.0 リリースのドキュメントをご使用の旨をお書き添えください。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号、ファクス番号
- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

# 表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

| 表記法 | 適用 |
|---|---|
| **太字** | 用語集で定義されている用語を示す。 |
| 〔Ctrl〕+〔Tab〕 | 複数のキーを同時に押すことを示す。 |
| *斜体* | 強調または書籍のタイトルを示す。 |
| 等幅テキスト | コード サンプル、コマンドとそのオプション、データ構造体とそのメンバー、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。<br>*例*<br>`#include <iostream.h> void main ( ) the pointer psz`<br>`chmod u+w *`<br>`\tux\data\ap`<br>`.doc`<br>`tux.doc`<br>`BITMAP`<br>`float` |
| 太字の等幅テキスト | コード内の重要な箇所を示す。<br>*例*<br>`void **commit** ( )` |
| *斜体の等幅テキスト* | コード内の変数を示す。<br>*例*<br>`String *expr*` |

| 表記法 | 適用 |
| --- | --- |
| 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。<br>*例*<br>LPT1<br>SIGNON<br>OR |
| { } | 構文の中で複数の選択肢を示す。実際には、この括弧は入力しない。 |
| [ ] | 構文の中で任意指定の項目を示す。実際には、この括弧は入力しない。<br>*例*<br>`buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...` |
| \| | 構文の中で相互に排他的な選択肢を区切る。実際には、この記号は入力しない。 |
| ... | コマンドラインで以下のいずれかを示す。<br>■ 引数を複数回繰り返すことができる<br>■ 任意指定の引数が省略されている<br>■ パラメータや値などの情報を追加入力できる<br>実際には、この省略記号は入力しない。<br>*例*<br>`buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...` |
| .<br>.<br>. | 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。<br>実際には、この省略記号は入力しない。 |

# 1 WebLogic Integration B2B セキュリティ入門

ここでは、以下の内容を取り上げます。

- WebLogic Integration B2B のセキュリティ モデル
- プリンシパル、ユーザ、およびグループ
- デジタル証明書
- 認証局
- SSL プロトコル
- 安全な環境を保証するためのコンフィグレーションの制限

## WebLogic Integration B2B のセキュリティ モデル

WebLogic Integration B2B のセキュリティ モデルには、以下の主要機能が組み込まれています。

- 基盤となる BEA WebLogic Server™ プラットフォームのセキュリティ機能を使用して、プリンシパルの認証および認可を実行してから、B2B リソースに対するアクセス権を付与します。
- トレーディング パートナのデジタル証明書を検証したり、重要なビジネス メッセージには欠かせない否認防止性のサポートを実装したりするための ユーザ独自のツールまたはサードパーティ ベンダのツールを組み込むことを 可能にすることで、機能拡張に対応します。

この節では、WebLogic Integration の認証および認可機能に関与する WebLogic Server および B2B のエンティティについて説明します。

WebLogic Server 6.x のセキュリティ レルムおよび互換モードの WebLogic Server Security Service を使用する必要があります。WebLogic Platform セキュリティの詳細については、次の URL にある『WebLogic Platform 7.0 セキュリティの紹介』を参照してください。

```
http://edocs.beasys.co.jp/e-docs/platform/docs70/secintro/index.html
```

WebLogic Integration B2B *認証*とは、システム エンティティが主張するアイデンティティまたはシステム エンティティに対して主張するアイデンティティを確認するプロセスのことです。認証では、関心の対象がエンティティの身元にあります。つまり、認証はアイデンティティとエンティティとの関連付けです。認可では、関心の対象がそのアイデンティティに基づいて表示可能な内容または実行可能な処理にあります。WebLogic Integration B2B では、以下の方法を使用して認証を実行します。

- ユーザ名とパスワード ー人間ユーザ（管理者）は、ユーザ名とパスワードを使用して自身のアイデンティティを証明します。
- デジタル証明書ートレーディング パートナは、デジタル証明書を使用して自身のアイデンティティを B2B エンジンに対して証明します。
- Secure Sockets Layer (SSL) ー SSL プロトコルを使用すると、プリンシパル間の接続に対してデータの整合性および機密性が提供されます。

*認可*は、システム エンティティがシステム リソースにアクセスするために付与される権利（パーミッション）です。認可プロセスは、そのような権利を与える手順です。B2B リソースにアクセスするパーミッションは、ACL (Access Control List: アクセス制御リスト ) とロールによって割り当てられます。

WebLogic Server と WebLogic Integration B2B が共同して WebLogic Integration B2B エンジンのプリンシパルを認証および認可するしくみの詳細については、第 2 章「トレーディング パートナの認証と認可」を参照してください。

次の図は、WebLogic Server と WebLogic Integration が B2B のセキュリティ モデルで提供するエンティティと機能を示しています。

**図 1-1 WebLogic Integration B2B のセキュリティ モデル**


次の表では、この B2B セキュリティ モデルで示した各機能について説明します。

**表 1-1 WebLogic Integration B2B のセキュリティ モデルのコンポーネント**

| コンポーネント | 説明 |
|---|---|
| 会話認可 | トレーディング パートナのビジネス メッセージが到着すると、B2B エンジンは、ビジネス メッセージの認可プロセスの一部としてビジネス メッセージの内容を調べ、コラボレーション アグリーメントに照らして検証する。つまり、特定のトレーディング パートナが送受信できるビジネス メッセージをコラボレーション アグリーメントが定義する。B2B エンジンは、コラボレーション アグリーメントのロールおよび会話定義によって、トレーディング パートナが送信または受信することになっているビジネス メッセージと、受け取ったビジネス メッセージの内容が矛盾していないことを確認する。<br>この認可方法により、関連するコラボレーション アグリーメントと矛盾しないビジネス メッセージだけが B2B エンジン リソースに対するアクセス権を付与される。 |
| データ暗号化サービス | データ暗号化サービスでは、暗号化を必要とするビジネス プロトコルに合わせてビジネス メッセージを暗号化する。データは、送信側の証明書、プライベート キー、および受信側の証明書を組み合わせてビジネス メッセージをエンコードすることで暗号化される。暗号化したメッセージは、受信側のプライベート キーを使用しないと解読できない。<br>データ暗号化サービスの使い方については、4-35 ページの「メッセージ暗号化のコンフィグレーション」を参照。 |

**表 1-1 WebLogic Integration B2B のセキュリティ モデルのコンポーネント**

| コンポーネント | 説明 |
|---|---|
| 転送サーブレットでの認証 | 転送サーブレットは、以下のリソースを含む B2B のリソースに対する HTTP および HTTPS アクセスのエントリ ポイントとなる WebLogic Integration 固有のサーブレットである。<br>■ WebLogic Integration リポジトリ<br>■ WebLogic Integration ワークフロー テンプレートおよび定義<br>■ JDBC 接続プール<br>■ JMS 送り先<br>転送サーブレットは、特定のコラボレーション アグリーメントにバインドされたトレーディング パートナ用として、WebLogic Server 環境で動的に登録される。 |
| SSL 経由の発信要求の認証 | B2B エンジンは、SSL ハンドシェークで取得した SSL 証明書を使用してすべての発信メッセージの受信側を認証し、バインドされている関連コラボレーション アグリーメントとメッセージが矛盾していないことを保証する。 |
| `WLCCertAuthenticator` クラス | `WLCCertAuthenticator` クラスは、トレーディング パートナの証明書を、そのトレーディング パートナに対してコンフィグレーションされている WebLogic Server ユーザにマップする。<br>`WLCCertAuthenticator` クラスは、`weblogic.security.acl.CertAuthenticator` インタフェースを実装している。<br>このクラスをコンフィグレーションすると、トレーディング パートナの証明書を検証する独自の実装または信頼性のあるサードパーティ ベンダの実装を呼び出すことができる。詳細については、第 2 章「トレーディング パートナの認証と認可」を参照。 |

**表 1-1 WebLogic Integration B2B のセキュリティ モデルのコンポーネント**

| コンポーネント | 説明 |
|---|---|
| 否認防止性フレームワーク | B2B のセキュリティ システムには、否認防止性のサポートを実装するための手段が用意されている。否認防止性とは、トレーディング パートナが別のトレーディング パートナ宛てのビジネス メッセージを送信したこと証明したり、トレーディング パートナからのビジネス メッセージを受信したことを証明できることを表す。否認防止性には、以下のサービスが必要となる。<br>■ データ暗号化<br>■ デジタル署名<br>■ セキュア タイムスタンプ<br>■ セキュア監査ログ<br>WebLogic Integration では、優れた否認防止性の実装と独自の実装または信頼性のあるサードパーティの実装を組み込むためのサービス プロバイダ インタフェース（SPI）が提供される。<br>否認防止性の詳細については、第 5 章「否認防止性の実装」を参照。 |
| プライベート キーストア | トレーディング パートナ コラボレーションで使用されるプライベート キー、パスワード、および証明書を格納するファイル。これらのキーおよびパスワードは、以下の証明書に埋め込まれる。<br>■ クライアント証明書－リモートまたはローカルのトレーディング パートナのデジタル証明書。<br>■ サーバ証明書－リモートのトレーディング パートナのデジタル証明書。<br>■ 署名証明書－デジタル署名サポートが必要な場合に各トレーディング パートナのビジネス メッセージに使用される。<br>■ 暗号化証明書－ビジネス メッセージの暗号化が必要な場合に各トレーディング パートナに使用される。 |

**表 1-1 WebLogic Integration B2B のセキュリティ モデルのコンポーネント**

| コンポーネント | 説明 |
|---|---|
| ルート CA キーストア | B2B コラボレーションで使用される各トレーディング パートナおよびサーバ証明書に関連付けられたすべての信頼性のある CA 証明書を格納するファイル。 |
| SSL プロトコル経由の着信要求の認証 | 着信トレーディング パートナのメッセージが到着すると、トレーディング パートナと WebLogic Server システムが証明書を交換して、互いのアイデンティティを確立する。SSL ハンドシェークが終了すると、トレーディング パートナと WebLogic Server システムとのネットワーク接続が確立される。<br>相互認証を行うための WebLogic Server での SSL プロトコルのコンフィグレーション方法については、4-2 ページの「SSL プロトコルと相互認証のコンフィグレーション」を参照。 |
| WebLogic リソースの ACL | ACL は、WebLogic Integration B2B リソースへのアクセスを保護する複数のエントリを持つデータ構造である。リソースまたはリソースのクラスに対するパーミッションは、ACL によって、リストに記載されているユーザおよびグループに付与される。ACL には `AclEntry` のリストが入っており、それぞれは特定のユーザまたはグループに関するパーミッションのセットを持っている。<br>パーミッションはリソースにアクセスするために必要な特権を表し、保護対象のリソースに固有のものである。指定可能なパーミッションは、ACL で保護するリソースのタイプによって異なる。たとえば、ファイルの送信と受信、ファイルの削除、ファイルの読み出しと書き込み、およびサーブレットのロードを行うパーミッションがある。<br>JDBC 接続プールの ACL をコンフィグレーションする方法については、4-7 ページの「WebLogic Integration B2B のアクセス制御リストのコンフィグレーション」を参照。 |

**表 1-1 WebLogic Integration B2B のセキュリティ モデルのコンポーネント**

| コンポーネント | 説明 |
|---|---|
| WebLogic キーストア プロバイダのサービス プロバイダ インタフェース（SPI） | プライベート キーおよび証明書をキーストアに追加、管理する手段を実装するインタフェースのセット。WebLogic キーストア プロバイダは、Sun Microsystems 社から提供されている参照キーストア実装を Java Development Kit に使用したコンポーネントである。また、各キーストアをファイルとして実装する標準「JKS」キーストア タイプを使用している。 |

B2B エンジンで使用される WebLogic Server セキュリティ機能の詳細については、次のマニュアルを参照してください。

- 『WebLogic Security の管理』の「SSL プロトコルのコンフィグレーション」

  `http://edocs.beasys.co.jp/e-docs/wls/docs70/secmanage/ssl.html`

- 『WebLogic Security の管理』の「ACL の定義」

  `http://edocs.beasys.co.jp/e-docs/wls/docs70/secmanage/security6.html`

# プリンシパル、ユーザ、およびグループ

プリンシパルとは、B2B の環境およびリソースへのアクセス権を必要とするエンティティのことです。WebLogic Integration B2B のプリンシパルは以下のとおりです。

- トレーディング パートナ
- 人間ユーザ（WebLogic Integration B2B 管理者）

プリンシパルには、認証および認可メカニズムを通して、WebLogic Integration B2B エンジンの環境およびリソースに対するアクセス権が付与されます。WebLogic Integration B2B のプリンシパルは、WebLogic Server ユーザにマップされます。

B2B エンジンが WebLogic Server ユーザのアイデンティティを証明できた場合、B2B エンジンは、そのユーザと、そのユーザの代わりにコードを実行するスレッドを関連付けます。スレッドがコードの実行を開始する前に、WebLogic Integration B2B は関係のある ACL をチェックして、その WebLogic Server ユーザが処理を続行するための適切なパーミッションを持っていることを確認します。

WebLogic Integration B2B は、以下のタイプの WebLogic Server ユーザをサポートしています。

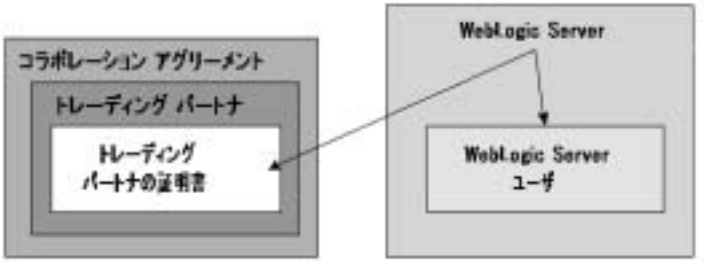
- WebLogic Integration のトレーディング パートナ
- WebLogic Integration B2B システム ユーザ
- WebLogic Integration B2B 管理者

グループとは、WebLogic Server ユーザのセットのことです。グループを使用すると、管理者がグループ全体に対するパーミッションを一度に指定できるので、多数のユーザを効率的に管理できます。

# トレーディング パートナのコンフィグレーションについて

WebLogic Integration のコラボレーション アグリーメントをコンフィグレーションする場合、そのアグリーメントにバインドされるトレーディング パートナ名も指定します。ユーザと B2B Console 内のトレーディング パートナを関連付けるには、WebLogic Server ユーザ名であるトレーディング パートナ ユーザ名を指定します。WebLogic Server では、トレーディング パートナのデジタル証明書をトレーディング パートナ ユーザに実行時にマップします。

**図 1-2 トレーディング パートナの証明書と WebLogic Server ユーザとのマッピング**




したがって、WebLogic Server は、トレーディング パートナのメッセージを受け取ったときに、トレーディング パートナの証明書を読み出してトレーディング パートナに対応する WebLogic Server ユーザを見つけることができるので、B2B エンジンの認証プロセスを開始できます。

# WebLogic Integration B2B システム ユーザのコンフィグレーションについて

B2B システム ユーザ、`wlisystem` に関しては、以下の注意事項があります。

- このユーザは、転送サーブレットを除くすべての B2B リソースにアクセスできます。この制限により、外部エンティティが B2B システム ユーザとして WebLogic Integration システムに入ることができなくなります。

- このユーザは、製品に付属しているサンプル コンフィグレーションで定義済みです。wlisystem ユーザのデフォルト パスワードは wlisystem です。ただし、BEA システム ユーザがない場合は、WebLogic Server Administration Console を使用してユーザ名 wlisystem ( パスワード wlisystem) を作成してください。
- wlisystem パスワードを変更したい場合は、B2B Console および fileRealm.properties ファイルで変更を行なってください（wlisystem パスワードが、fileRealm.properties ファイルで指定したパスワードと一致しない場合、システム ログに警告が記録され、B2B エンジンによって例外が送出されます)。

  **注意：** wlisystem パスワードは、必ず B2B Console 経由で変更してください。wlisystem パスワードの変更に WebLogic Server Administration Console を使用すると、WebLogic Integration リポジトリに格納された wlisystem パスワードは同時に更新されず、その後に B2B エンジンを起動しようとしても失敗します。

# デジタル証明書

デジタル証明書とは、インターネットなどのネットワーク上でプリンシパルとオブジェクトを一意に識別するための電子的なドキュメントのことです。デジタル証明書によって、認証局と呼ばれる信頼性のあるサードパーティによって証明されたとおりに、ユーザまたはオブジェクトのアイデンティティが特定の公開鍵に安全にバインドされます。デジタル証明書の所有者は、公開鍵とプライベート キーの組み合わせによって一意に識別されます。

デジタル証明書を使用すると、特定の公開鍵が実際に指定されたユーザまたはエンティティに属するものであることを検証できます。デジタル証明書の受信側は、その証明書およびサブジェクトの公開鍵が信頼性のある認証局（CA）が発行し、署名したことを検証できます。この検証は、信頼性のある認証局の公開鍵を使用して、デジタル署名がその認証局のプライベート キーを使って作成されたことを確認することによって行ないます。検証が成功すると、一連の論証によって、対応するプライベート キーがデジタル証明書に指定されているサブジェクトによって保持されていることと、デジタル証明書が特定の認証局によって作成されたことが保証されます。

通常、デジタル証明書には以下のようにさまざまな情報が格納されています。

- サブジェクト（保持者、所有者）の名前とサブジェクトを一意に識別するための ID 情報（URL や電子メール アドレスなど）
- サブジェクトの公開鍵
- デジタル証明書を発行した認証局の名前
- シリアル番号
- デジタル証明書の有効期間（開始日と終了日で定義）

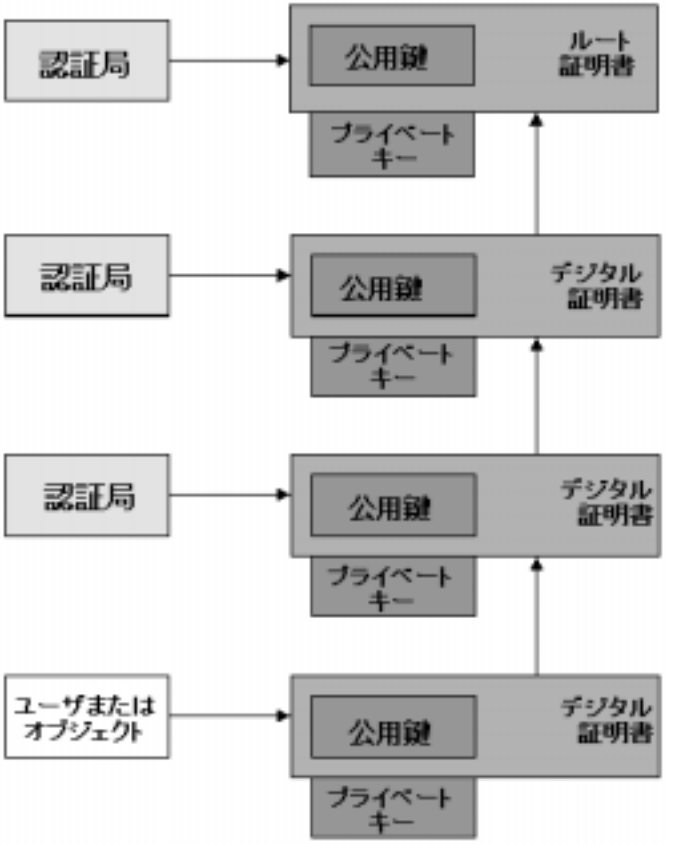
最も広く利用されているデジタル証明書のフォーマットは、ITU-T X.509 国際標準で定義されているフォーマットです。このため、X.509 標準に準拠していればどのアプリケーションを使用しても、デジタル証明書を読み出したり、書き込んだりすることができます。WebLogic Server の公開鍵インフラストラクチャ（PKI）では、X.509 バージョン 1 (X.509v3) またはバージョン 3 (X.509v3) に準拠したデジタル証明書が認識されます。

# 認証局

デジタル証明書は、認証局によって発行されます。デジタル証明書および公開鍵の発行先のアイデンティティを保証する信頼性のある第三者組織または企業はすべて、認証局になることができます。認証局は、デジタル証明書を作成する際に、証明書が改ざんされればわかるようにプライベート キーを使って署名します。認証局は、署名したデジタル証明書を要求元サブジェクトに返します。

サブジェクトは、認証局の公開鍵を使用して発行認証局の署名を検証できます。認証局の公開鍵は、下位認証局の公開鍵の妥当性を保証する上位認証局から発行されたデジタル証明書を提供することで利用できるようになります。この階層は、最終的に、ルート証明書と呼ばれる自己署名デジタル証明書にたどり着きます。次の図を参照してください。

**図 1-3　認証局の階層**




デジタル証明書を使用したり、デジタル署名を検証したり、ビジネス メッセージを解読したりする前に、そのデジタル証明書が信頼性のある認証局によって発行されていることを確認します。誰がビジネス メッセージを暗号化したかにかかわりなく、ビジネス メッセージのデジタル証明書は、認証局によって信頼性が確立されている必要があります。

# SSL プロトコル

SSL プロトコルを使用すると、ネットワーク接続を通してリンクされた 2 つのアプリケーション間で互いのアイデンティティを認証し、アプリケーション間のデータ交換を暗号化することで、安全な接続を確立できます。SSL 接続はハンドシェークで始まります。ハンドシェーク時には、アプリケーション間でのデジタル証明書の交換、使用する暗号化アルゴリズムの取り決め、そのセッションの残りで使用する暗号キーの生成が行われます。

SSL プロトコルは、以下のセキュリティ機能を提供します。

- サーバ認証ーサーバは信頼性のある認証局が発行した自身のデジタル証明書を使用して、クライアントに対して自身を認証します。
- クライアント認証ーオプションとして、クライアントが独自のデジタル証明書を提示することで、自身をサーバに対して認証することを要求できます。このタイプの認証は、相互認証とも呼ばれます。WebLogic Integration B2B の認証モデルでは相互認証を使用します。
- データ プライバシークライアントの要求とサーバの応答は暗号化され、ネットワーク経由でやり取りされるデータの機密性を維持します。
- データ整合性ークライアントとサーバ間のデータ フローを、第三者による改ざんから保護します。

SSL プロトコルは、トレーディング パートナ間のメッセージのリンクレベル暗号化を実装するために使用します。

管理者は、Web ブラウザを使用して B2B Console にアクセスします。HTTPS (Hypertext Transfer Protocol with SSL) を使用すると、このタイプのネットワーク通信をセキュリティで保護できます。

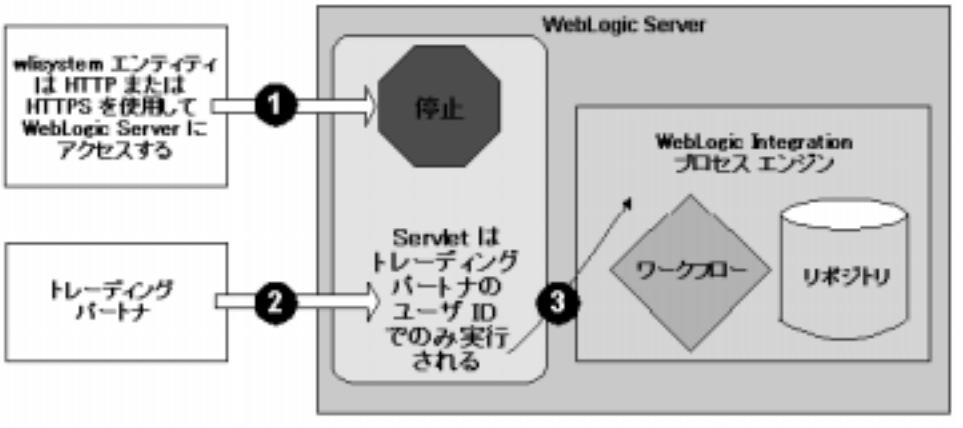
# 安全な環境を保証するためのコンフィグレーションの制限

WebLogic Integration B2B には、安全な環境を保証するために、この節で説明する制限があります。これらの制限については、必要に応じて第 4 章「セキュリティのコンフィグレーション」でも説明しています。

- B2B システム ユーザは、転送サーブレットにアクセスするための認可の対象となりません。これにより、外部エンティティが B2B システム ユーザを装うことはできなくなります。
- トレーディング パートナは、B2B リソースにアクセスするための認可の対象となりません。トレーディング パートナの証明書が認証されたら、その証明書は WebLogic Server ユーザにマップされます。トレーディング パートナのビジネス メッセージも認証されてからのみ、トレーディング パートナの証明書のマップ先となった WebLogic Server ユーザは、トレーディング パートナの代わりに B2B リソースにアクセスします。
- WebLogic Integration 環境のアーキテクチャは、トレーディング パートナのパスワード情報を明かさずに済むように設計されています。トレーディング パートナは、デジタル証明書に基づいて B2B 環境で必ずマップされるからです。

次の図は、これらのセキュリティ上の制限が WebLogic Integration B2B セキュリティ モデルで果たす役割を示しています。

**図 1-4 安全な WebLogic Integration B2B 環境**




上の図の以下の番号に注目してください。

1. B2B 転送サーブレットにアクセスしようとしている `wlisystem` というエンティティがアクセスを拒否されます。
2. トレーディング パートナの証明書およびビジネス メッセージが検証されたら、その証明書は対応する WebLogic Server ユーザにマップされます。
3. 前の手順でマップされた WebLogic Server ユーザは、トレーディング パートナのビジネス メッセージをサービスするために必要な B2B リソースにアクセスします。

# 2　トレーディング パートナの認証と認可

ここでは、以下の内容を取り上げます。

- WebLogic Integration でのトレーディング パートナの認証
- WebLogic Integration B2B でのトレーディング パートナの認可

## WebLogic Integration でのトレーディング パートナの認証

認証とは、WebLogic Integration B2B エンジンがプリンシパルのアイデンティティを確立するプロセスのことです。トレーディング パートナと WebLogic Integration 間では、相互認証（HTTPS）を行う SSL プロトコル経由でデジタル証明書を使用します。B2B エンジンは、リポジトリ内のセキュリティ情報に照らしてデジタル証明書を調べて検証します。

WebLogic Integration B2B には、2 つのレベルの認証プロセスが組み込まれています。

- 第 1 レベルでは、トレーディング パートナの証明書を検証します。
- 第 2 レベルでは、トレーディング パートナのメッセージを認証します。

トレーディング パートナのビジネス メッセージがどちらのレベルの認証にも合格すると、B2B エンジンはビジネス メッセージに対する認可プロセスを実行します。

以下の節では、2 つのレベルの B2B 認証プロセスについて説明します。

# トレーディング パートナの証明書の検証

WebLogic Integration B2B のセキュリティ モデルでは、SPI (Service Provider Interface: サービス プロバイダ インタフェース ) が提供されます。このインタフェースを使用すると、トレーディング パートナの証明書を検証するためのサードパーティのサービスを呼び出すインタフェースを実装する Java クラスを組み込むことができます。こうした CVP (Certificate Verification Provider: 証明書検証プロバイダ ) と呼ばれる実装では、以下のいずれかの証明書検証アプリケーションを呼び出せます。

- CRL (Certificate Revocation List: 証明書失効リスト ) の実装
- 証明書のステータスをリアルタイムでチェックするために認証局などの信頼性のある第三者エンティティと対話する OCSP (Online Certificate Status Protocol) の実装
- 独自の証明書検証の実装

## 証明書検証の利点

トレーディング パートナの証明書検証の目的は、トレーディング パートナのデジタル証明書の有効性を検証することです。たとえば、証明書の検証では、以下のタスクの一部またはすべてを実行します。

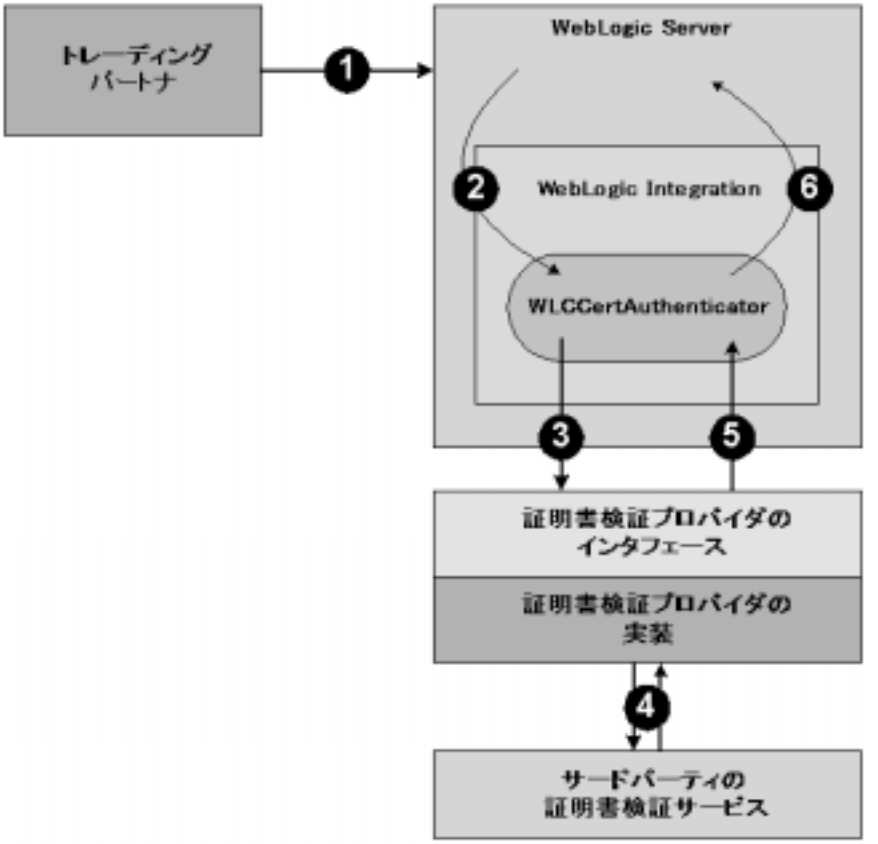
- ルート認証局に至る証明書チェーンをたどります。
- 失効されたものがないかチェーン内のすべての証明書を CRL でチェックします。
- 証明書を検証することができる信頼性のあるベンダを使用してリアルタイムに証明書チェックを実行します。
- 証明書チェーン内のすべてのデータが有効であることを確認します。
- チェーン内の各証明書の署名を検証します。

CVP の実装のコンフィグレーションおよび使用は省略可能ですが、使用すると、証明書検証プロセスでより高いレベルのセキュリティを提供できます。

# 証明書検証プロセス

次の図は、WebLogic Integration 環境の証明書検証プロセスのイベント タスク順序を示しています。

**図 2-1 WebLogic Integration でのトレーディング パートナの証明書検証**



上の図の以下の番号に注目してください。

| 番号 | 説明 |
|---|---|
| 1 | 証明書検証は、SSL を使用した場合にのみ実行される。トレーディング パートナと WebLogic Server システムは SSL ハンドシェークを実行し、その中で証明書を交換し、互いのアイデンティティを確立する。トレーディング パートナのデジタル証明書の認証局は、WebLogic Server で信頼されている必要がある。ハンドシェークでは、WebLogic Server は以下の点を検証する。<br>■ トレーディング パートナの証明書の認証局が WebLogic Server 環境で信頼されているものであること<br>■ トレーディング パートナの証明書が有効期限切れになっていないこと<br>SSL ハンドシェークが終了すると、トレーディング パートナと WebLogic Server システムとのネットワーク接続が確立される。 |
| 2 | WebLogic Server は、B2B エンジンの `WLCCertAuthenticator` クラスを呼び出す。`WLCCertAuthenticator` クラスは、トレーディング パートナの証明書をトレーディング パートナに対してコンフィグレーションされている WebLogic Server ユーザにマップするために、`weblogic.security.acl.CertAuthenticator` インタフェースを実装する。 |
| 3 | `WLCCertAuthenticator` クラスは、サードパーティの証明書検証サービスを呼び出す実装の CVP インタフェースを呼び出す。 |
| 4 | CVP の実装は、トレーディング パートナの証明書のステータスを返すサードパーティの証明書検証サービスを呼び出す。 |
| 5 | CVP の実装は、証明書のステータスを `WLCCertAuthenticator` クラスに返す。 |
| 6 | トレーディング パートナの証明書が有効な場合、B2B エンジンは証明書をリポジトリの有効なトレーディング パートナ名にマップしようとする。証明書が有効なトレーディング パートナにマップされると、WebLogic Integration は WebLogic Server ユーザを WebLogic Server に返す。 |

# 証明書検証プロバイダの実装

証明書検証プロバイダ（CVP）Java クラスは、`com.bea.b2b.security.CertificateVerificationProvider` インタフェースを実装しています。CVP クラスは、以下の 2 種類を呼び出すことができます。

- 2-5 ページの「サービス プロバイダ インタフェースの使い方」で説明されているサービス プロバイダ インタフェースに準拠した信頼性のあるサードパーティ ベンダ
- 独自の証明書検証アプリケーション

どちらを呼び出す場合でも、実際に証明書を検証するアプリケーションを呼び出す CVP SPI の Java 実装を作成する必要があります。ここからは、この CVP アプリケーションの作成、コンパイル、およびコンフィグレーションについて説明します。

## サービス プロバイダ インタフェースの使い方

WebLogic Integration B2B では、CVP のサービス プロバイダ インタフェース（SPI）を提供する `com.bea.b2b.security.CertificateVerificationProvider` インタフェースを介して CVP を実装できます。ここで説明する SPI を使用して CVP を実装または使用する場合、CVP が実行時に正しく呼び出されるように、WebLogic Integration B2B Console で CVP をコンフィグレーションする必要があります。

`com.bea.b2b.security.CertificateVerificationProvider` インタフェースには、CVP アプリケーションが実装しなければならない以下のメソッドがあります。

- `void init()`

  このメソッドは、このインタフェースを実装するクラスのカスタム初期化プロセスを呼び出すために B2B エンジンによって自動的に呼び出されます。このメソッドは、WebLogic Integration の起動時にのみ呼び出されます。

- `String verify(Certificate[] certs)`

  このメソッドは、SSL ハンドシェークで取得した証明書チェーンを検証します。このメソッドは、以下の `String` 値のいずれかを返します。

  - `good` ー トレーディング パートナの証明書が有効で、期限切れではありません。

- revoked ートレーディング パートナの証明書が証明書チェーンのいずれかの認証局によって取り消されているか、トレーディング パートナの証明書が期限切れになっています。
- unknown ー証明書チェーンのいずれの認証局もトレーディング パートナの証明書の有効性を確立できません。

インプリメンタは、このメソッドによる検証手順を選択することができます。たとえば、ファイルに格納されている証明書失効リスト（CRL）をチェックする、OCSP (Online Certificate Status Protocol) 使用してリアルタイムで証明書のステータスをチェックする、その他のメカニズムを使用するといった使い方の中から必要に応じて選択できます。

**注意：** CVP を実装する場合、引数を持たない CVP のデフォルト パブリック コンストラクタを追加する必要があります。クラス内のコンストラクタもメソッドも、例外を送出してはなりません。

CVP をコンフィグレーションしない場合、B2B エンジンは、信頼性のある認証局によって発行されたすべての証明書を有効であると見なします。

## 証明書検証プロバイダ クラスのコンパイル

CVP を実装する場合は、以下の点に注意します。

- CVP Java クラスを作成したら、コンパイルして CLASSPATH に配置する必要があります。
- B2B Console または Bulk Loader ユーティリティを使用して、CVP をコンフィグレーションする必要があります。CVP をコンフィグレーションしたら、CVP を有効にするために WebLogic Server を再起動します。CVP をコンフィグレーションしないと、B2B エンジンは、信頼性のある認証局によって発行されたすべての証明書を有効であると見なします。

## WebLogic Integration B2B での証明書検証プロバイダのコンフィグレーション

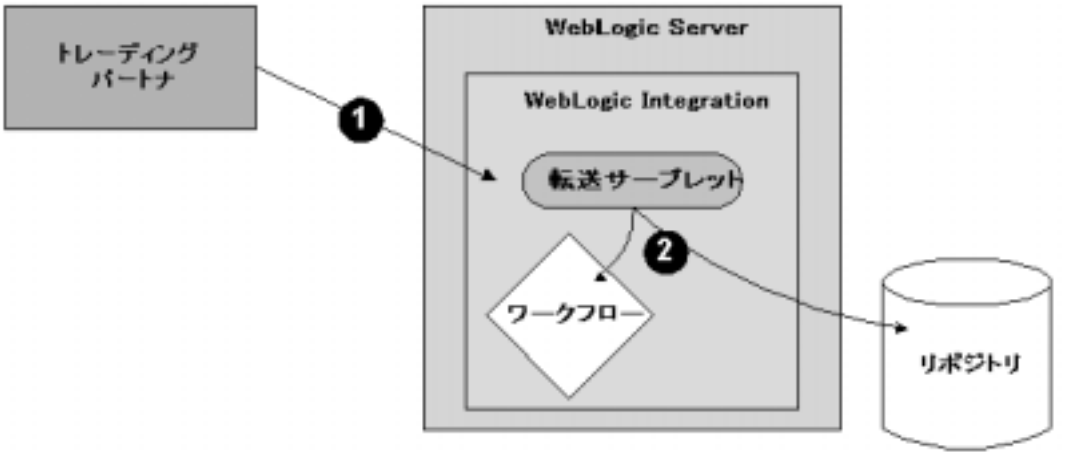
B2B Console を使用して CVP をコンフィグレーションする手順については、4-42 ページの「証明書検証プロバイダ インタフェースのコンフィグレーション」を参照してください。CVP をコンフィグレーションしたら、CVP を有効にするために WebLogic Server を再起動します。

# トレーディング パートナのメッセージの認証

トレーディング パートナの証明書が WebLogic Server によって認証されたら、B2B エンジンは、メッセージ自体を処理する前にトレーディング パートナのメッセージを認証する必要があります。トレーディング パートナのメッセージの認証では、ビジネス メッセージの送信元が WebLogic Integration リポジトリのリストに入っている有効なトレーディング パートナであることを検証します。トレーディング パートナのメッセージが認証されると、そのトレーディング パートナのアイデンティティが認められ、B2B リソースに完全にアクセスできるようになります。

次の図は、トレーディング パートナのメッセージの認証プロセスを示しています。

**図 2-2 トレーディング パートナのメッセージの認証**




上の図の以下の点に注目してください。

- 転送サーブレットは、B2B エンジンへのエントリ ポイントです。トレーディング パートナのメッセージが B2B 転送サーブレットに到着すると（番号 1）、転送サーブレットはトレーディング パートナのメッセージを検証します。トレーディング パートナの検証とは、トレーディング パートナに関連付けられている有効な証明書からトレーディング パートナの値を取り出して、トレーディング パートナ名が有効であることを確認することです。

- トレーディング パートナのメッセージが認証されると、トレーディング パートナは認可され、リポジトリや WebLogic Integration Business Process Management（BPM）テンプレートおよびワークフロー（番号 2）などの WebLogic Integration リソースにアクセスできるようになります。アクセスには、B2B システム ユーザ コンテキストが使用されます。

**注意：** トレーディング パートナだけが B2B 転送サーブレットを使用するための認証の対象となります。B2B システム ユーザが B2B リソースにアクセスするために転送サーブレットにアクセスしようとした場合、WebLogic Server によってアクセスが拒否されます。このメカニズムによって、B2B システム ユーザを装ったリモート エンティティが B2B リソースにアクセスすることはできなくなります。

# WebLogic Integration B2B でのトレーディング パートナの認可

認可とは、B2B プリンシパルに B2B リソースの特定のセットへのアクセスを許可するプロセスのことです。B2B システムの認可モデルは、ACL およびパーミッション メカニズムとロールベースの認可制御に基づいています。

B2B システムには、2 つのレベルの認可が組み込まれています。

- B2B 転送サーブレットにアクセスするためのトレーディング パートナの認可
- トレーディング パートナのビジネス メッセージに関連付けられている会話の認可

## トレーディング パートナの認可

このレベルの認可は、WebLogic Server によって実行されます。トレーディング パートナのメッセージが WebLogic Server に到着すると、トレーディング パートナと WebLogic Server は相互認証手順を実行し、トレーディング パートナは認可され、B2B 転送サーブレットにアクセスできるようになります。

転送サーブレットのパスは一定ではないので、トレーディング パートナが転送サーブレットの URL にアクセスできるように web.xml ファイルを編集する必要があります。転送サーブレットに対応する URL が B2B 環境に合わせて変化するという性質を持っているので、これを事前にコンフィグレーションすることはできません。

転送サーブレットの ACL を web.xml ファイルに指定する必要があります。次の例は、wlctransport という転送サーブレットの ACL を指定する web.xml ファイルです。

**コード リスト 2-1 転送サーブレットの ACL の例**

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 1.2//EN
" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
...
...
<!-- Authentication -->
 <security-constraint>
  <web-resource-collection>
   <web-resource-name>wlctransport</web-resource-name>
   <url-pattern>*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
   <role-name>TradingPartnerGroupA</role-name>
  </auth-constraint>
 </security-constraint>

 <login-config>
  <auth-method>CLIENT_CERT</auth-method>
 </login-config>

 <security-role>
  <role-name>TradingPartnerGroupA</role-name>
 </security-role>
</web-app>
```

上のコードについて以下で説明します。

- wlctransport は、WebLogic Integration リポジトリで定義されているエンドポイントを持つ転送サーブレットです。

- TradingPartnerGroupA は、すべてのトレーディング パートナの WebLogic Server ユーザがメンバになっている WebLogic Server ユーザ グループです。
- CLIENT_CERT は、転送サーブレットにアクセスするために必要な認証モードが相互認証を行う SSL であることを指定しています。

# 会話認可

B2B エンジンが会話認可を実行する場合、サーバは、トレーディング パートナがバインドされているコラボレーション アグリーメントに関するトレーディング パートナのビジネス メッセージの内容を調べます。つまり、コラボレーション アグリーメントで指定された特定のロールおよびパーティに対しては、トレーディング パートナは特定のビジネス メッセージのセットしか送信できません。B2B エンジンは、特定の会話に関するコラボレーション アグリーメントに指定された以下の情報と照らしてビジネス メッセージを検証します。

- パーティ情報（トレーディング パートナとロール）
- 会話定義
- ドキュメント交換 ID

受け取ったビジネス メッセージに関する会話認可が終了すると、B2B リソースに対するアクセスは ACL によって制御されます。

# 3 キーストアのコンフィグレーション

ここでは、以下の内容を取り上げます。

- キーストアとは
- ドメインの作成
- キーストアの作成とサーバ証明書の追加
- WebLogic キーストア プロバイダのコンフィグレーション
- トレーディング パートナの証明書のキーストアへの追加
- キーストアを使用するためのドメインのコンフィグレーション
- マルチノード クラスタにおけるキーストアの使用

WebLogic Integration B2B のコンフィグレーションについての一般情報は、『*B2B Integration 管理ガイド*』の「基本的なコンフィグレーション タスク」を参照してください。

## キーストアとは

キーストアは、キーおよび証明書を保管する保護されたデータベースです。キーおよび証明書を持っており、メッセージ暗号化、デジタル署名、または SSL を使用している場合、そういったキーおよび証明書の保管にキーストアを使用し、B2B アプリケーションのような、認証または署名にキーストアが必要になる可能性のあるアプリケーションに対してこのキーストアを使用可能にしておくことをお勧めします。キーストアを作成し、使用可能にするには、キーストア プロバイダが必要です。これは、WebLogic Server 7.0 セキュリティ アーキテクチャで導入されています。

WebLogic キーストア プロバイダは、Sun Microsystems 社が Java Development Kit に提供されている参照キーストア実装を使用したコンポーネントです。この WebLogic キーストア プロバイダには、次の機能があります。

- JDK がバンドルされた JKS (Java KeyStore) プロバイダを使用し、これによってキーストアをファイルとして実装する
- 各プライベート キーを個別のパスワードで保護する
- キーストア全体をパスワードで保護する

# 作成するキーストア

B2B コラボレーションに対して WebLogic Integration ドメインを設定するときに、次のキーストアを作成するように WebLogic キーストア プロバイダをコンフィグレーションします。

- プライベート キーストア

  B2B コラボレーションで通常必要とされるクライアント証明書、サーバ証明書、署名証明書、および暗号化証明書などに使用する、トレーディング パートナの証明書およびプライベート キーを格納します。B2B エンジンでは、このキーストアからプライベート キーおよび証明書を取り出し、SSL、メッセージ暗号化、およびデジタル署名に使用します。JavaSoft JDK `keytool` ユーティリティまたは WebLogic Server `ImportPrivateKey` ユーティリティを使用すれば、このキーストアを作成し、作成したキーストアにプライベート キーおよび関連証明書を追加できます。

- ルート CA キーストア

  すべての信頼性のある CA (Certificate Authority : 認証局 ) の証明書を格納します。WebLogic Keystore キーストア プロバイダは、信頼性のある CA のキーストアを作成し、WebLogic Server では、このキーストアを、クライアント、サーバ、署名、および暗号化証明書の検証に SSL が使用する、信頼性のある CA を検索するデフォルトの場所として使用します。

# キーストアの作成およびコンフィグレーション手順

B2B コラボレーションに必要なキーストアを作成およびコンフィグレーションするには、以下の手順を実行します。

1. B2B ドメインを作成します。
2. キーストアを作成し、SSL で必要なサーバ証明書とキーを追加します。
3. WebLogic キーストア プロバイダをコンフィグレーションします。
4. トレーディング パートナの証明書をキーストアに追加します。
5. 信頼性のある認証局の証明書を CA キーストアに追加します。
6. キーストアを使用するドメインをコンフィグレーションします。

この項目には、マルチノード クラスタにおけるキーストア ファイルの使い方に関する説明も含まれます。

キーストア、証明書、およびキーに関する基本情報については、以下を参照してください。

- キーストア プロバイダの詳細については、『*WebLogic Security の紹介*』の「WebLogic Security プロバイダ」を参照してください。
- 証明書およびキーの詳細については、『*WebLogic Security プログラマーズ ガイド*』の「セキュリティの基礎概念」、および『*WebLogic Security サービスの開発*』の「カスタム キーストア プロバイダの作成」を参照してください。

# ドメインの作成

セキュリティをコンフィグレーションする WebLogic Integration B2B ドメインの作成には、BEA コンフィグレーション ウィザードを使用することをお勧めします。WebLogic Integration ドメインを作成するには、以下の手順を実行します。

1. 『*Configuration Wizard の使い方*』の説明に従って、コンフィグレーション ウィザードを起動します。このマニュアルは次の URL で参照できます。

   ```
   http://edocs.beasys.co.jp/e-docs/platform/docs70/confgwiz/index.html
   ```

2. WebLogic Integration ドメインのコンフィグレーションを実行します。ドメインは以下のいずれでもかまいません。
   - WebLogic Integration BPM ドメイン
   - WebLogic Integration EAI ドメイン
   - WebLogic Integration ドメイン

   **注意：** 新規ドメインの作成には、必ず WebLogic Integration テンプレートを選択し、WebLogic Server テンプレートまたは WebLogic Portal テンプレートは使用しないでください。WebLogic Integration テンプレートを使用することにより、この手順で作成されたドメインが互換モードの WebLogic Server 6.x セキュリティ レルムに基づくことが保証されます。新たな WebLogic Server 7.0 レルムは LDAP をベースにしており、WebLogic Integration ではサポートされていません。WebLogic Server テンプレートで新規のドメインを作成すると、作成したドメインでは LDAP をベースにした、この新しい WebLogic Server 7.0 セキュリティ ドメインを使用することになります。

3. コンフィグレーション ウィザードを終了したら、新しく作成されたカスタム ドメインから、次のファイルをテキスト エディタで開きます。

   *DOMAIN_HOME*/config.xml

   上記のファイル名で、*DOMAIN_HOME* はカスタム ドメインを含むディレクトリのパスを表しています。たとえば、Windows では *DOMAIN_HOME* の値は次のようになります。

   ```
   c:\bea\user_projects\mydomain
   ```

4. カスタム ドメインに作成された WebLogic Integration アプリケーションの自動デプロイメントを無効化します。そのためには、次に示すように、config.xml ファイルの WLIapplication 要素の Deployed 属性を false に設定します。

   ```
   <Application Deployed="false" Name="WLIApplication"
   Path="<%WLI_HOME%\lib>" TwoPhase="true">
   ```

# キーストアの作成とサーバ証明書の追加

この節では、SSL を使用するときに必要なサーバ証明書およびキーを格納するためのプライベート キーストアの作成、さらに CA 証明書用の関連 CA キーストアを作成する方法について説明します。トレーディング パートナの証明書をプライベート キーストアに追加する方法については、3-11 ページの「トレーディング パートナの証明書のキーストアへの追加」を参照してください。

トレーディング パートナの認証には、SSL を使用することを強くお勧めします。ただし、SSL を使用する場合、B2B ドメインの各マシンにも SSL をコンフィグレーションする必要があります。SSL をコンフィグレーションする場合、WebLogic Server のローカル インスタンスに対して証明書とプライベート キーを提供する必要があります。この証明書は、*サーバ証明書*と呼ばれています。ローカル サーバに対するこのサーバ証明書とプライベート キーは、キーストアに格納するようにしてください。この節では、サーバ証明書とプライベート キーをキーストアに追加する方法を説明します。

トレーディング パートナの認証および認可プロセスにおいて、当該 WebLogic Server インスタンスの SSL レイヤでは、キーストアを使用して、以下の対象を入手します。

- プライベート キーストアからローカル サーバの証明書およびプライベート キーを入手する
- ルート CA キーストアから信頼性のある CA 証明書を入手する

SSL を使用する WebLogic Server のコンフィグレーションの手順については、4-2 ページの「SSL プロトコルと相互認証のコンフィグレーション」を参照してください。

WebLogic Integration のセキュリティ サービスは WebLogic Server 上に構築されるため、WebLogic Integration については、現在、使用できることが保証されているのは、JKS プロバイダ ベースのキーストアに限られます。B2B コラボレーションに必要なキーストアを作成する場合、次のユーティリティのうちいずれかを使用できます。

- JavaSoft JDK `keytool` ユーティリティ

  このユーティリティについては、Sun Microsystems 社から次の URL で公開されている『*keytool - Key and Certificate Management Tool*』を参照してください。

```
http://java.sun.com/products/jdk/1.2
```

- WebLogic Server `ImportPrivateKey` ユーティリティ

  このユーティリティについては、次の URL にある『*WebLogic Server 管理者ガイド*』の「WebLogic Java ユーティリティの使い方」を参照してください。

  ```
  http://edocs.beasys.co.jp/e-docs/wls/docs70/adminguide/utils.html
  ```

WebLogic Integration B2B ドメインに必要なキーストアを作成するには、以下の手順を実行します。

1. コマンド ウィンドウを開きます。
2. ドメインのルート ディレクトリに進みます。Windows の場合ならば、次のように入力します。

   ```
   c:\> cd bea\user_projects\b2bdomain
   ```

3. 次のファイルを取得または作成します。
   - サーバ証明書およびプライベート キー

     サーバ証明書およびプライベート キーは、SSL による認証と認可に必要です。サーバ証明書、プライベート キーは、CertGen ユーティリティを使用すれば作成できます。ただし、CertGen で作成された証明書およびキーは、テスト目的に限って使用するようにしてください。CertGen による証明書、キーは、プロダクション環境での使用は想定されていません。CertGen ユーティリティの詳細については、次の URL にある『*WebLogic Server 管理者ガイド*』の「WebLogic Java ユーティリティの使い方」を参照してください。

     ```
     http://edocs.beasys.co.jp/e-docs/wls/docs70/adminguide/utils.html
     ```

   - ルート CA 証明書およびプライベート キー

     必要に応じて、WebLogic Server にバンドルされている JDK の CA キーストアを使用することができます。このキーストア、`cacerts` は、次の場所にあります。

     ```
     JAVA_HOME/jre/lib/security
     ```

4. `keytool` または `ImportPrivateKey` ユーティリティを使用して、プライベート キーストアを作成し、サーバ証明書とプライベート キーを追加します。

**注意：** キーストアを作成するコマンドは、証明書とキーを追加するコマンドと同じです。証明書とキーを追加する時に、キーストアがなかった場合、コマンドを入力するとキーストアが作成されます。

プライベート キーストアを作成する場合の `ImportPrivateKey` コマンドの構文は、次のとおりです。

```
java utils.ImportPrivateKey keystoreName keystorepass alias
keypass certfile keyfile
```

**注意：** `ImportPrivateKey` コマンドを実行する場合、BEA WebLogic Platform がクラスパスに入っていることを確認してください。

次の表は、`ImportPrivateKey` ユーティリティで使用可能な引数を示しています。

**表 3-1 ImportPrivateKey コマンドの引数**

| コマンド引数 | 説明 |
|---|---|
| *keystoreName* | キーストア ファイルの名前を定義する。キーストアがない場合は、新しいキーストアが作成される。 |
| *keystorepass* | キーストア ファイルを開くときに必要とされるパスワードを定義する。 |
| *alias* | キーストア内の証明書およびキーのルックアップに使用される名前を定義する。**注意：**エリアスに使用される暗号化の強度（*domestic* または *export*）を記録しておいてください。 |
| *keypass* | プライベート キー ファイルのアンロック、およびキーストア内のプライベート キーの保護に使用されるパスワードを定義する。`CertGen` を使用してサーバ証明書を作成した場合、このパスワードはその証明書を作成したときのパスワードとなる。 |
| *certfile* | プライベート キーに関連付けられた証明書の名前を定義する。 |
| *keyfile* | 保護プライベート キーを保持するファイル名を定義する。**注意：**キーファイル名には、暗号化の強度を組み込むようにすることをお勧めします。<br>例：`keyDomestic.pem` または `keyExport.pem` |

プライベート キーストアに追加するサーバ証明書およびキーごとに、ImportPrivateKey または keytool コマンドを実行します。

5. ルート CA キーストアを作成します。ルート CA キーストアは、当初の CA 証明書の追加時に作成されます（プライベート キーストアの作成時に作成されるのと同様）。

   ルート CA キーストアを作成するには、keytool コマンドを以下の引数をつけて実行します。

   ```
   keytool -import -keystore keystoreName -trustcacerts -alias
   aliasName -file cert_file -storepass keystorepw -noprompt
   ```

   次の表は、keytool ユーティリティで使用可能な引数を示しています。

**表 3-2 keytool コマンドの引数**

| コマンド引数 | 説明 |
|---|---|
| -import | エリアス *aliasName* によって、ファイル *cert_file* から証明書または証明書チェーンを読み込み、これをキーストア *keystoreName*.pem に格納する。 |
| -keystore *keystoreName* | キーストアのパス名を識別する。 |
| -trustcacerts | cacerts という名前のファイル内の、信頼性のある証明書を指定する。このファイルは *JAVA_HOME*/jre/lib/security ディレクトリにある。 |
| -alias *aliasName* | 証明書のエリアスを指定する。 |
| -file *cert_file* | ルート CA の証明書を含むファイル（ここでは *cert_file* で表している）を指定する。 |
| -storepass *keystorepw* | ルート CA キーストアのパスワード（ここでは *keystorepw* で表している）を指定する。 |
| -noprompt | keytool ユーティリティが追加のコマンド引数を要求しないようにする。 |

6. ドメイン内の各マシンについて、同じファイル名および相対パスを使用して手順 4 と 5 を繰り返します。

**注意：** SSL 認証と認可が確実に正しく機能するようにするため、各マシンのキーストア、証明書、キーなどには必ず同一のファイル名およびパスを使用してください。

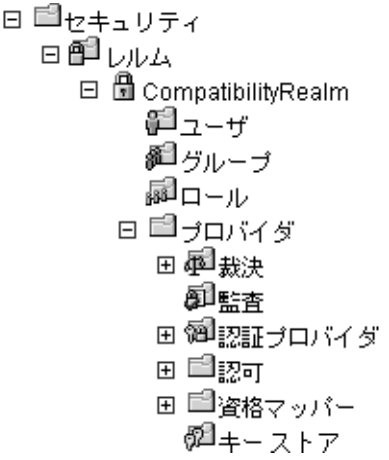
7. B2B ドメインをマルチノード クラスタにデプロイする場合、『*WebLogic Server ドメイン管理*』の「ノード マネージャによるサーバ可用性の管理」に説明されているようにノード マネージャ をコンフィグレーションしてください。

# WebLogic キーストア プロバイダのコンフィグレーション

3-5 ページの「キーストアの作成とサーバ証明書の追加」で作成したキーストアについて WebLogic キーストア プロバイダをコンフィグレーションするには、以下の手順を実行します。

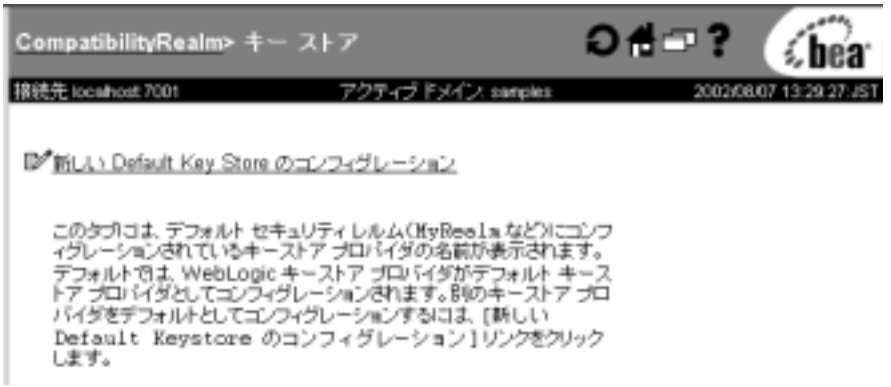
1. 新しく作成したカスタム ドメインで WebLogic Server を起動します。たとえば、Windows の場合、[ スタート ｜ BEA WebLogic Platform7.0 ｜ UserProjects ｜ *ドメイン*｜ *servername*] と選択します。
2. 『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration 管理ツールと設計ツール」、「WebLogic Server Administration Console の起動」の説明に従って、WebLogic Server Administration Console を起動します。
3. 左側のナビゲーション ペインで、[ セキュリティ｜レルム｜ CompatibilityRealm ｜プロバイダ｜キー ストア ] と選択します。

**図 3-1　ナビゲーション ペインでのキーストアの選択**




WebLogic Server Administration Console により、次の図に示すように、新規のデフォルト キーストアをコンフィグレーションするウィンドウが表示されます。

**図 3-2　新規のデフォルト キーストアのコンフィグレーション**




4. [ 新しいデフォルト キー ストアのコンフィグレーション ] をクリックします。

   次の図に示されているように、キーストアをコンフィグレーションする [ 一般 ] タブが表示されます。

**図 3-3 デフォルト キーストアのコンフィグレーションに使用する [ 一般 ] タブ**


5. [ 一般 ] タブで、次の各ファイルのパス名を指定します。
   - プライベート キーストア ファイル
   - ルート CA キーストア ファイル
6. [ 作成 ] をクリックします。
7. WebLogic Server をシャットダウンして、再起動します。

# トレーディング パートナの証明書のキーストアへの追加

キーストアにトレーディング パートナの証明書を取り込むには、この節に示されている手順を実行します。各トレーディング パートナの証明書の詳細については、4-15 ページの「トレーディング パートナの証明書のコンフィグレーション」を参照してください。

**注意：** キーストアに必要な証明書およびプライベート キーがすでに取り込まれている場合でも、以下のタスクを実行して WebLogic Integration リポジトリに必要な情報を取り込む必要があります。

WebLogic Integration では、トレーディング パートナの証明書をキーストアにロードするときに、信頼性のある認証局に照らしてその証明書を検証することはありません。

1. 次の例に示すように、config.xml ファイルの WLI アプリケーション要素の Deployed 属性を true に設定することにより、B2B ドメインで作成された WebLogic Integration アプリケーションの自動デプロイメントを有効化します。

```
<Application Deployed="true" Name="WLI" Path="<%WLI_HOME%\lib>"
TwoPhase="true">
```

2. 『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration 管理ツールと設計ツール」、「B2B Console の起動」の説明に従って、B2B Console を起動します。

この節では、B2B コラボレーションのプライベート キーストアにデータを取り込む手順を、以下に従って示します。

- ローカル トレーディング パートナの証明書およびプライベート キーの追加
- リモート トレーディング パートナの証明書の追加
- 証明書のキーストアへのバルク ロードおよびインポート
- 証明書およびプライベート キーのキーストアからの削除

# ローカル トレーディング パートナの証明書およびプライベート キーの追加

ローカル トレーディング パートナは、次の証明書およびプライベート キーを必要とします。

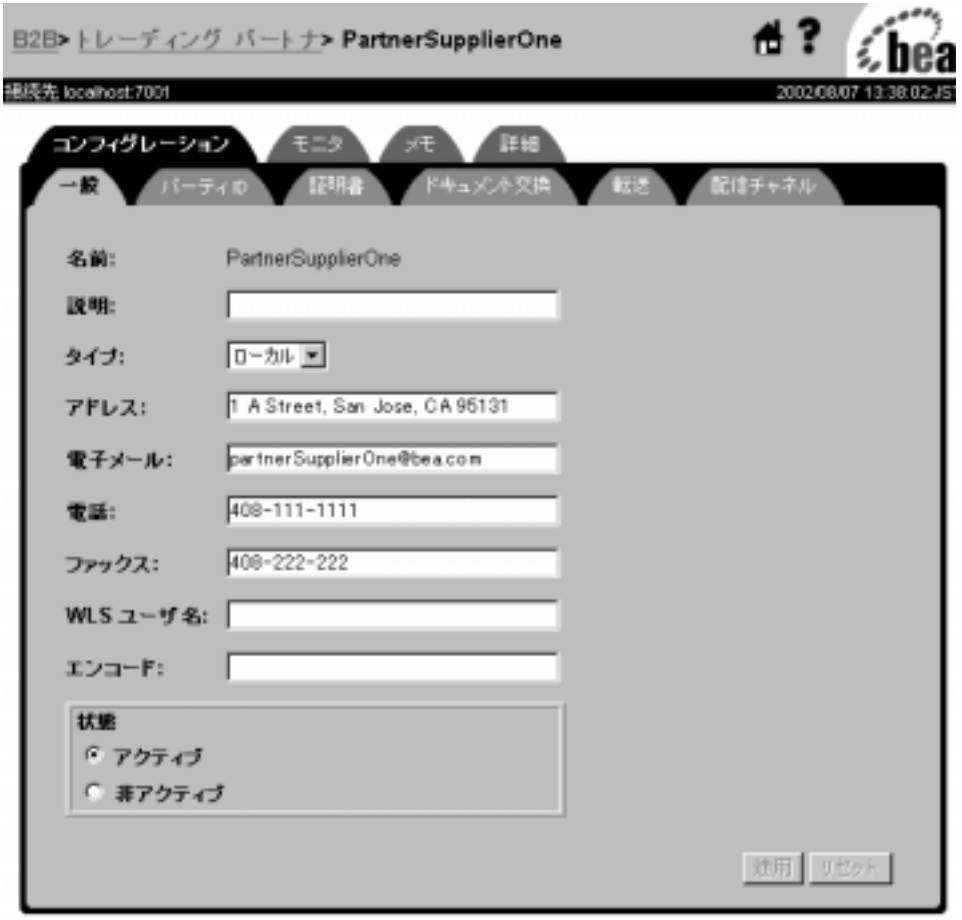
- クライアント証明書とプライベート キー
- 暗号化証明書とプライベート キー
- 署名証明書とプライベート キー

プライベート キーストアにこれらの証明書およびプライベート キーを追加するには、この節で示される手順を実行します。

**注意：** ローカル トレーディング パートナに対しては、サーバ証明書はコンフィグレーションしません。暗号化証明書および署名証明書は任意ですが、クライアント証明書は相互認証の SSL を使用する場合必須です。ローカル トレーディング パートナの証明書の詳細については、4-15 ページの「トレーディング パートナの証明書のコンフィグレーション」を参照してください。クライアント証明書を必要としないサーバ側、ないし一方向認証の使い方については、4-51 ページの「サーバ側認証のコンフィグレーション」を参照してください。

1. 左側のナビゲーション ペインで、[B2B ｜トレーディング パートナ ] と選択します。
2. 証明書を追加する対象のローカル トレーディング パートナの名前をクリックします。[ コンフィグレーション : 一般 ] タブが表示されます。

**図 3-4　ローカル トレーディング パートナの [ 一般 ] コンフィグレーション ページ**

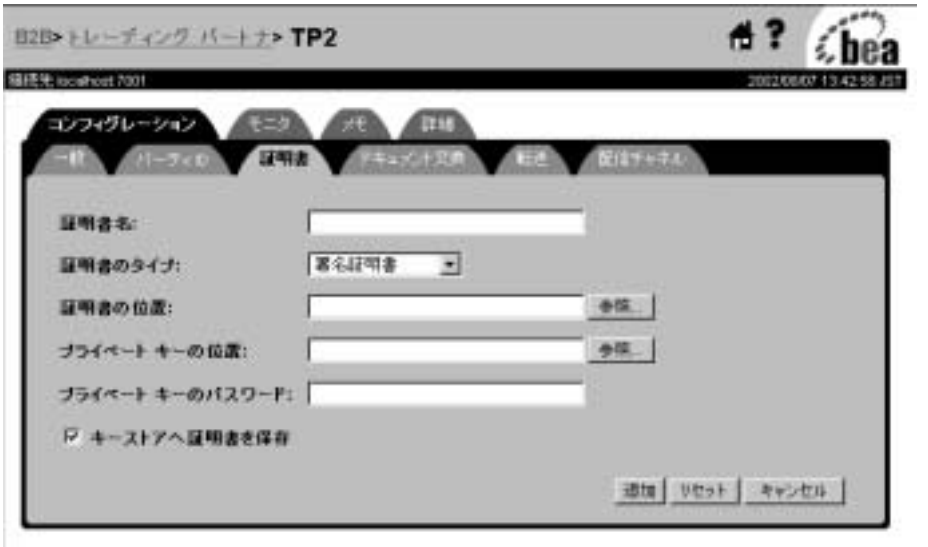



3. [ 証明書 ] タブを選択します。[ 証明書 ] コンフィグレーション ページが表示されます。

**図 3-5 ローカル トレーディング パートナの [ 証明書 ] コンフィグレーション ページ**


4. [ 証明書エントリの作成 ] をクリックします。ローカル トレーディング パートナに対して追加する証明書の詳細を指定するページが表示されます。

**図 3-6　ローカル トレーディング パートナに対する証明書エントリの作成**




5. 各トレーディング パートナの証明書に対して、次の情報を入力します。
   - 証明書の名前またはエリアス。キーストアにすでにこの証明書が取り込まれている場合、指定するエリアスは、キーストア内のこの証明書の当初エントリに使用されたエリアスに一致していなければなりません。
   - 証明書の種類。ローカル トレーディング パートナの場合、有効な証明書は、クライアント証明書、署名証明書、および暗号化証明書です。
   - 証明書の場所。クラスタに WebLogic Integration をデプロイする場合、指定する場所は、各マシンで同一にする必要があります。
   - 証明書に関連付けられたプライベート キーの場所。クラスタに WebLogic Integration をデプロイする場合、指定する場所は、各マシンで同一にする必要があります。
   - プライベート キーのパスワード。このパスワードは、サーバがプライベート キーの内容を読み込むとき、およびプライベート キーをキーストアに格納するときに、サーバによって使用されます。パスワード自体が、システム内のどこかに格納、保持されることはありません。キーストアにすでにデータが入力されている場合、指定するパスワードは、キース

トアにプライベート キーを格納する時に使用されるパスワードに一致する必要があります。

**注意：** プレーン テキスト（無保護）のプライベート キーを B2B Console を使用してインポートする場合、[ プライベート キーのパスワード ] というフィールドにプライベート キーストアのパスワードを指定します。

6. [ 追加 ] をクリックすると WebLogic Integration リポジトリに証明書およびプライベート キーが追加されます。
7. [ キーストアへ証明書を保存 ] チェック ボックスにチェックを入れると、証明書およびプライベート キーがプライベート キーストアに追加されます。

# リモート トレーディング パートナの証明書の追加

リモート トレーディング パートナは、次の証明書を持っています。

- クライアント証明書
- 暗号化証明書
- 署名証明書
- サーバ証明書

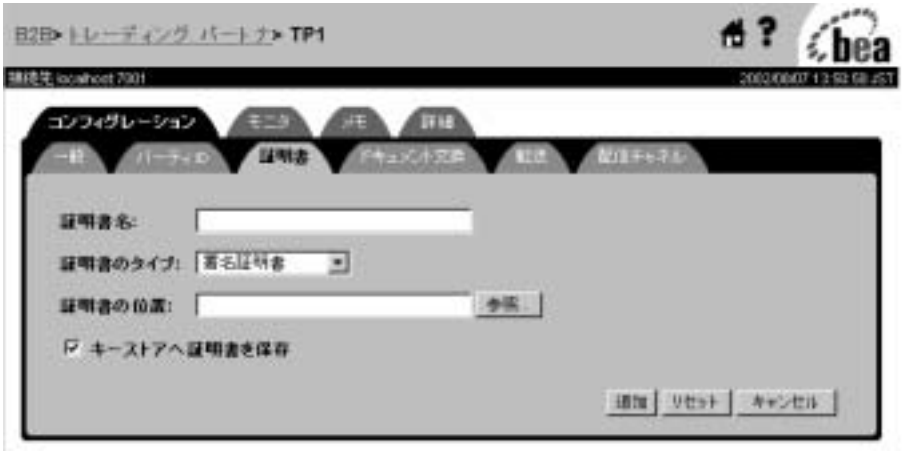
**注意：** リモート トレーディング パートナの証明書には、プライベート キーは指定しません。暗号化証明書および署名証明書は任意ですが、クライアント証明書およびサーバ証明書は相互認証の SSL を使用する場合必須です。リモート トレーディング パートナの証明書の詳細については、4-15 ページの「トレーディング パートナの証明書のコンフィグレーション」を参照してください。クライアント証明書を必要としないサーバ側、ないし一方向認証の使い方については、4-51 ページの「サーバ側認証のコンフィグレーション」を参照してください。

これらの証明書をプライベート キーストアに追加するには、以下の手順を実行します。

1. 必要に応じて B2B Console を起動します。
2. 左側のナビゲーション ペインで、[B2B ｜トレーディング パートナ ] と選択します。

3. 証明書を追加する対象のリモート トレーディング パートナの名前をクリックします。[ コンフィグレーション : 一般 ] タブが表示されます。
4. [ 証明書 ] タブを選択します。[ 証明書 ] コンフィグレーション ページが表示されます。
5. [ 証明書エントリの作成 ] リンクをクリックします。リモート トレーディング パートナに対して追加する証明書の詳細を指定するページが表示されます。

**図 3-7　リモート トレーディング パートナに対する証明書エントリの作成**




6. リモート トレーディング パートナに対して追加する各証明書の名前、場所、および種類を入力します。
7. [ 追加 ] をクリックすると WebLogic Integration リポジトリに証明書が追加されます。
8. [ キーストアへ証明書を保存 ] のチェック ボックスにチェックを入れると、証明書がプライベート キーストアに追加されます。

# 証明書のキーストアへのバルク ロードおよびインポート

Bulk Loader ユーティリティ（B2B Console またはコマンド ラインから利用）を使用して証明書を WebLogic Integration リポジトリにコンフィグレーションすると、トレーディング パートナの証明書は、キーストアには*インポートされません*。ただし、リポジトリにはその証明書のコンフィグレーション情報が格納されていますので、B2B エンジンの起動時に証明書をキーストアにインポートできます。

B2B エンジンでキーストアにトレーディング パートナの証明書をインポートするためには、事前に `startWeblogic` スクリプトで自動移行を有効化する必要があります。自動移行を有効にするには、以下の手順を実行します。

1. 3-3 ページの「ドメインの作成」で作成された B2B ドメインの WebLogic Server インスタンスをシャットダウンします。シャットダウンは、B2B ドメインのルート ディレクトリにある `stopWeblogic` スクリプトを実行することによって行ないます。たとえば

   - Windows の場合 :

     ```
     cd DOMAIN_HOME
     stopWeblogic.cmd
     ```

   - UNIX の場合 :

     ```
     cd DOMAIN_HOME
     stopWebLogic.sh
     ```

   上記の例で、*`DOMAIN_HOME`* は B2B ドメインのルート ディレクトリです。

2. Java システム プロパティ `wli.keystore.automigrate` を `startWeblogic` スクリプトの `Java` コマンド ラインに追加し、このプロパティ値を、次に示すように `true` に設定します。`wli.keystore.automigrate` プロパティは **太字**で示されています。

```
%JAVA_HOME%\bin\java %DB_JVMARGS% -Xmx256m -classpath %WLISERVERCP%
-Dbea.home=%BEA_HOME% -Dwli.bpm.server.evaluator.supportsNull=false
-Dweblogic.Domain=mydomain -Dweblogic.Name=myserver
-Dweblogic.management.username= -Dweblogic.management.password=
-Dweblogic.ProductionModeEnabled=true -Dweblogic.management.discover=false
-Djava.security.policy==%WL_HOME%\lib\weblogic.policy weblogic.Server
-Dwli.keystore.automigrate=true
```

3. 変更を保存します。

WebLogic Server がドメインで再起動されると、証明書とキーがインポートされます。

# 証明書およびプライベート キーのキーストアからの削除

B2B Console を使って、証明書、および該当する場合はプライベート キーを削除すると、その証明書およびプライベート キーに対する参照が WebLogic Integration リポジトリから削除されます。証明書と関連するキーのプライベート キーストアからの削除も同時に行なうことができます。

証明書を削除するには、以下の手順を実行します。

1. 必要に応じて B2B Console を起動します。
2. 左側のナビゲーション ペインで、[B2B ｜トレーディング パートナ ] と選択します。
3. 証明書およびプライベート キーを削除する対象のトレーディング パートナ名をクリックします。[ コンフィグレーション : 一般 ] タブが表示されます。
4. [ 証明書 ] タブを選択します。[ 証明書 ] コンフィグレーション ページが表示されます。

**図 3-8 キーストアからの証明書の削除**


5. 削除する証明書の種類を選択します。
6. リスト ボックスから証明書のエリアスを選択します。
7. [ 削除 ] をクリックすると、証明書、および該当する場合はプライベート キーが WebLogic Integration リポジトリから削除されます。
8. リポジトリからだけでなく、キーストアからも証明書とプライベート キーを削除する場合は、[ キーストアから証明書を削除 ] ボックスにチェックが入っていることを確認してから [ 削除 ] をクリックしてください。

# キーストアを使用するためのドメインのコンフィグレーション

作成したキーストアを使用するように B2B ドメインをコンフィグレーションするには、ドメインのルート ディレクトリにある `startWeblogic` スクリプトを修正する必要があります。このスクリプトを修正するには、以下の手順を実行します。

1. 次の例に示すように、ドメインのルート ディレクトリに進みます。
   - Windows の場合 :

     `c:\bea\user_projects\`*`domain`*
   - UNIX の場合 :

     `/usr/bin/bea/user_projects/`*`domain`*

     上記のパス名で、*`domain`* は B2B ドメインの名前を表しています。
2. テキスト エディタで、`startWebLogic.cmd` スクリプト（Windows の場合）または `startWebLogic.sh` スクリプト（UNIX の場合）を開きます。
3. 以下の例に示すように、WebLogic Server を起動するために `java` コマンドが発行される行に位置を合わせます。

```
%JAVA_HOME%\bin\java %DB_JVMARGS% -Xmx256m -classpath %WLISERVERCP%
-Dbea.home=%BEA_HOME% -Dwli.bpm.server.evaluator.supportsNull=false
-Dweblogic.Domain=mydomain -Dweblogic.Name=myserver
-Dweblogic.management.username= -Dweblogic.management.password=
-Dweblogic.ProductionModeEnabled=true -Dweblogic.management.discover=false
-Djava.security.policy==%WL_HOME%\lib\weblogic.policy weblogic.Server
```

4. この `java` コマンドに対して、署名およびメッセージ暗号化証明書のプライベート キー パスワードを指定するシステム プロパティを、以下の構文に従って追加します。

   `-DKey.`*`certificate-name`*`.password=`*`key_password`* `*`

   上記の構文で、
   - *`certificate-name`* は、3-11 ページの「トレーディング パートナの証明書のキーストアへの追加」のキーストアに追加される証明書の名前またはエリアスを示しています。

- *key_password* は、証明書がキーストアに追加されるときに証明書に指定されるプライベート キーのパスワードを示しています。プレーン テキストのプライベート キーにはパスワードを指定する必要はありません。B2B エンジンでは、そのようなプライベート キーの検索には、キーストアのパスワードを使用します。

5. さらに、この java コマンドに対して、プライベート キーストアおよびルート CA キーストアのパスワードを指定するシステム プロパティを、以下の構文に従って追加します。

   ```
   -Dwli.privateKeystore.password=keystore_pass
   -Dwli.caKeystore.password=caKeystore_pass
   ```

   上記の構文で、

   - *privateKeystore* は、3-5 ページの「キーストアの作成とサーバ証明書の追加」の説明に従って作成されたプライベート キーストアを示しています。
   - *keystore_pass* は、プライベート キーストアのパスワードを示しています。
   - *caKeystore* は、3-5 ページの「キーストアの作成とサーバ証明書の追加」の説明に従って作成されたルート CA キーストアを示しています。
   - *caKeystore_pass* は、ルート CA キーストアのパスワードを示しています。

**注意：** パスワードは、startWeblogic などのスクリプトにハードコーディングせずに、環境変数に設定することをお勧めします。環境変数に設定された場合、スクリプトでは、そのスクリプトが実行される環境からパスワードの値を取得します。

# マルチノード クラスタにおけるキーストアの使用

B2B ドメインをマルチノード クラスタにデプロイする場合、以下の作業を実行する必要があります。

1. プライベート キーストアおよびルート CA キーストアをクラスタ内の各マシンに複写します。複写したキーストアの相対位置は、すべてのマシンで同じになるようにします。
2. サーバ証明書がすべてのマシンで同じ相対位置に格納されるようにします。SSL サポートの要件を満たすため、サーバ証明書の場所は、ドメインの `config.xml` ファイルで識別されるようにする必要があります。
3. 3-9 ページの「WebLogic キーストア プロバイダのコンフィグレーション」に示されているように、管理サーバでプロバイダにより、プライベート キーストアおよびルート CA キーストアをコンフィグレーションします。
4. 4-2 ページの「SSL プロトコルと相互認証のコンフィグレーション」で説明されているように、サーバ証明書が管理サーバにコンフィグレーションされていることを確認します。

ドメイン内の各管理対象サーバの起動時に、管理サーバの補助により、WebLogic キーストア プロバイダのコンフィグレーションが自動的にサーバに伝播されます。

マルチノード クラスタで B2B セキュリティを管理する方法の詳細については、『*WebLogic Integration ソリューションのデプロイメント*』を参照してください。

# 4 セキュリティのコンフィグレーション

ここでは、以下の内容を取り上げます。

- SSL プロトコルと相互認証のコンフィグレーション
- WebLogic Integration B2B のアクセス制御リストのコンフィグレーション
- WebLogic Integration B2B エンジンのセキュリティのコンフィグレーション
- トレーディング パートナのセキュリティのコンフィグレーション
- メッセージ暗号化のコンフィグレーション
- 否認防止のためのデジタル署名のコンフィグレーション
- WLCCertAuthenticator クラスのカスタマイズ
- 証明書検証プロバイダ インタフェースのコンフィグレーション
- 発信 HTTP プロキシ サーバを使用するための WebLogic Integration B2B のコンフィグレーション
- Web サーバおよび WebLogic プロキシ プラグインでの WebLogic Integration のコンフィグレーション
- Business Process Management による WebLogic Integration リポジトリへのアクセスの管理
- サーバ側認証のコンフィグレーション

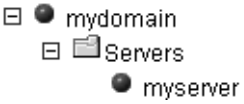
B2B セキュリティのコンフィグレーションを行なう前に、第 3 章「キーストアのコンフィグレーション」に説明されているキーストアのコンフィグレーションが完了していることを確認してください。WebLogic Integration B2B のコンフィグレーションに関する一般的な情報については、『*B2B Integration 管理ガイド*』の「基本的なコンフィグレーション タスク」を参照してください。

# SSL プロトコルと相互認証のコンフィグレーション

SSL プロトコルと相互認証を使用するように WebLogic Server をコンフィグレーションするには、以下の手順を実行します。

1. 『WebLogic Security の管理』の「SSL プロトコルのコンフィグレーション」で説明されているとおりに、WebLogic Server のデジタル証明書を取得します。
2. 『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration 管理ツールと設計ツール」、「WebLogic Server Administration Console の起動」の説明に従って、WebLogic Administration Console を起動します。
3. WebLogic Server Administration Console のナビゲーション ツリー（左ペイン）で、コンフィグレーションするドメインの [ サーバ | myserver] を選択します（次の図を参照）。

**図 4-1　ドメインの選択**

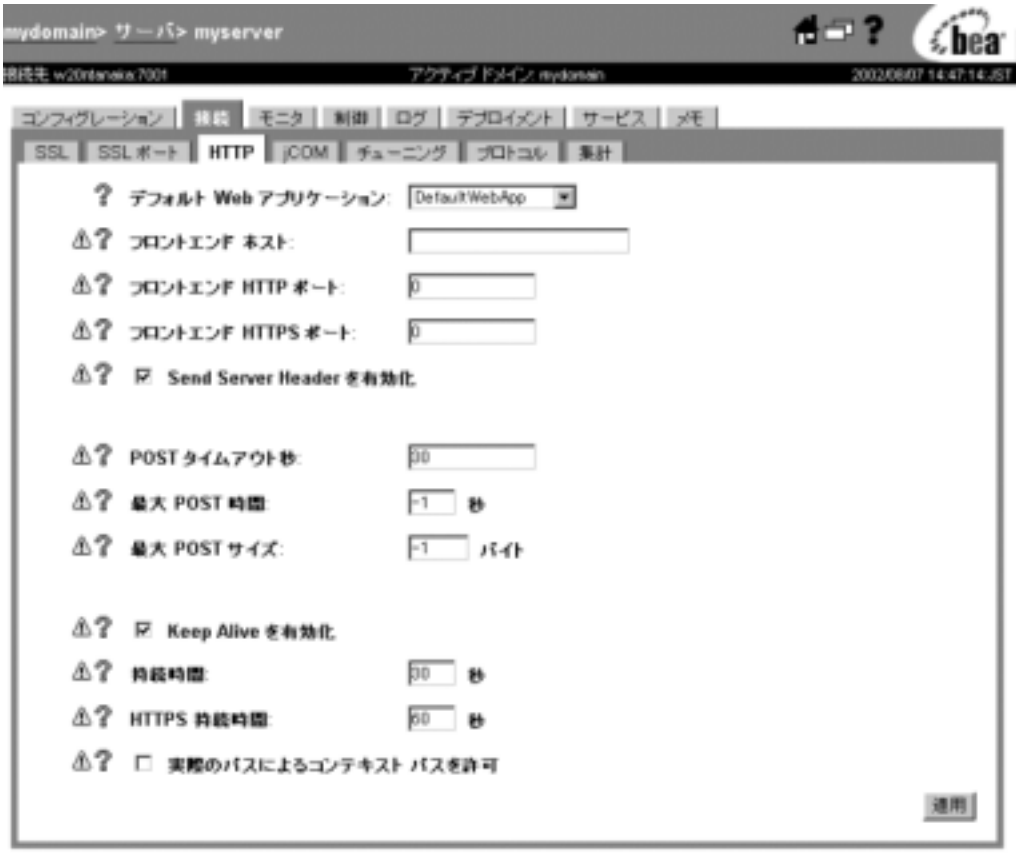



次の図に示されているように WebLogic Server の [ コンフィグレーション ] ページが表示されます。

**図 4-2 WebLogic Server Administration Console の [ コンフィグレーション ] ページ**

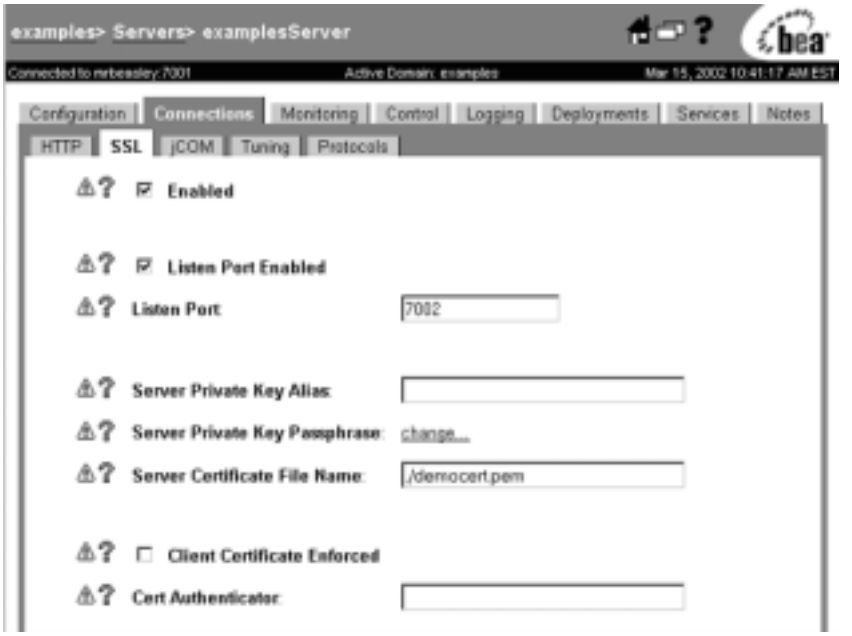

4. [ 接続 ] タブを選択します。次のページが表示されます。

**図 4-3 [ 接続 ] ページ**




5. [SSL] タブを選択し、次の図の Secure Sockets Layer (SSL) コンフィグレーション ページを表示します。

**図 4-4 [SSL] コンフィグレーション ページ**




6. 次の表では、[SSL] コンフィグレーション ページに入力する情報について説明します。

**表 4-1 [SSL] コンフィグレーション ページのフィールド**

| フィールド | 説明 |
|---|---|
| [Enabled] チェック ボックス | WebLogic Integration B2B とトレーディング パートナ間の SSL 接続を有効にする。 |
| [Listen Port Enabled] チェック ボックス | サーバのデフォルト ポート（7002）で SSL プロトコルの使用を有効にする。WebLogic Server が SSL 接続をリスンするポートは、[Listen Port] 属性を設定すれば変更できる。 |
| [Listen Port] | WebLogic Integration B2B が SSL 接続をリスンする専用ポートを指定する。 |

**表 4-1 [SSL] コンフィグレーション ページのフィールド ( 続き )**

| フィールド | 説明 |
|---|---|
| [ サーバ プライベート キー エイリアス ] | サーバ プライベート キーのキーストア入力に対するエリアス。 |
| [ サーバ プライベート キーの Pass Phrase] | キーストアのキーに対するパスフレーズを指定する（パスフレーズは、キーごとに必要。また、キーストア自体にパスフレーズを設定することもできる）。 |
| [ サーバ 証明書ファイル名 ] | WebLogic Server のデジタル証明書の絶対パスを指定する。この場所はルート認証局（ルート CA）とも呼ばれる。 |
| [ クライアント証明書を強制 ] チェック ボックス | WebLogic Integration B2B と、WebLogic Integration B2B リソースにアクセスするトレーディング パートナとの間の相互認証を有効にする。 |
| [Cert Authenticator] | トレーディング パートナのデジタル証明書の有効性を調べるための認可済み認証機関を指定する。 |
| [Client Certificate Enforced] チェック ボックス | WebLogic Integration B2B と、WebLogic Integration B2B リソースにアクセスするトレーディング パートナとの間の相互認証を有効にする。 |

# WebLogic Integration B2B のアクセス制御リストのコンフィグレーション

ユーザまたはグループが WebLogic Integration B2B のリソースにアクセスできるかどうかは、リソースの ACL (Access Control List: アクセス制御リスト ) によって決定されます。ACL を定義するには、以下の手順を実行します。

1. WebLogic Server Administration Console で、[ 新しい ACL の作成 ...] をクリックし、リソースの名前を指定します。
2. リソースのパーミッションを指定します。
3. 指定したセットのユーザおよびグループにパーミッションを付与します。

ACL の定義の詳細については、『*WebLogic Security の管理*』の「互換性レルムでの ACL の定義」を参照してください。

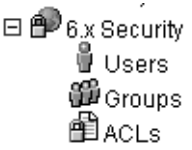
B2B リソースの場合、1 つまたは複数のパーミッションを付与できます。

WebLogic Integration B2B に付属しているサンプル コンフィグレーションに、JDBC 接続プールの設定済み ACL があります。この ACL には、このリソースの `wlcSamplesUser` ユーザに対して、`reserve`、`shrink`、および `reset` という 3 つのパーミッションが設定されています。

以下の手順では、ACL パーミッションを変更する場合に必要な手順の例を示しています。この例では、ドメイン WLIdomain の JDBC 接続プールの `reset` パーミッションを調整します。

1. WLI ドメインで WebLogic Server のインスタンスをコンフィグレーションして起動します。手順については、『*WebLogic Integration の起動、停止およびカスタマイズ*』の「はじめに」、「サンプル ドメインのコンフィグレーションと起動」を参照してください。
2. まだ起動していない場合は、WebLogic Server Administration Console を起動します（WLIdomain におけるデフォルトのシステム管理者のユーザ名は `system`、パスワードは `security`）。
3. ナビゲーション ツリーで、[6.x Security ｜ ACL] を選択します。

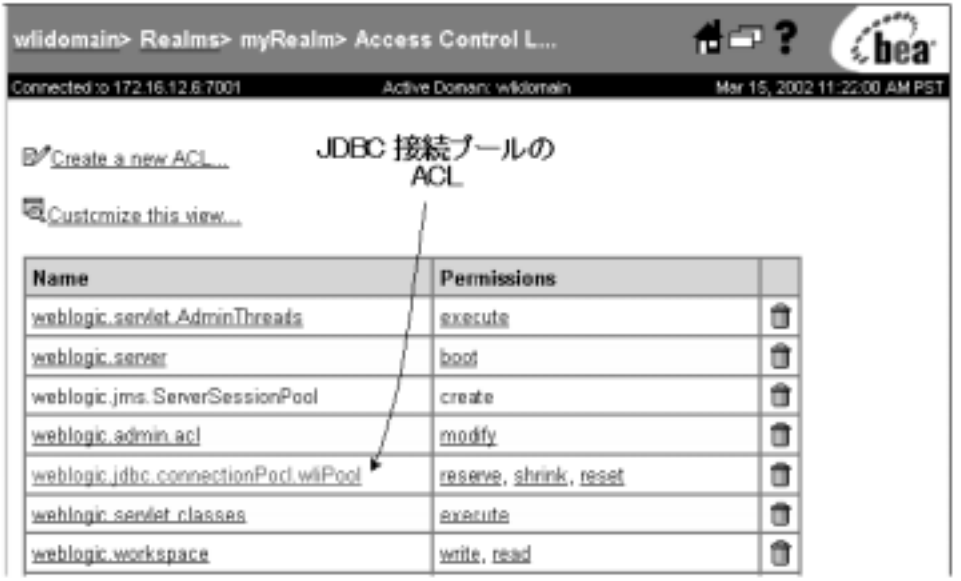
**図 4-5　ナビゲーション ツリーの ACL の選択**




[ アクセス コントロール リスト ] コンフィグレーション ページが表示されます。WebLogic Server にコンフィグレーションされた ACL が、このページにリストされます。

4. 次の図に示されているように、WebLogic Integration JDBC 接続プール ACL のエントリを含む行を見つけます。

**図 4-6　JDBC 接続プールの ACL**




5. その行の [ パーミッション ] カラムにある、`reset` リンクをクリックします。次の図のように、JDBC 接続プールの ACL の `reset` パーミッションを調整するためのダイアログ ボックスが表示されます。

**図 4-7 [ACL] ダイアログ ボックス**


6. ユーザまたはグループの reset パーミッションを指定するには、該当するフィールドにユーザまたはグループ名を入力し、[ 適用 ] をクリックします。ダイアログ ボックスに表示された [ 被許可者 ] から reset パーミッションを削除する場合は、該当するユーザまたはグループ名を選択し、[ 適用 ] をクリックします。

ACL の定義の詳細については、『*WebLogic Security の管理*』の「互換性レルムでの ACL の定義」を参照してください。

# WebLogic Integration B2B エンジンのセキュリティのコンフィグレーション

WebLogic Integration リポジトリには、WebLogic Integration B2B セキュリティ システムと B2B リソースにアクセスするトレーディング パートナに関するセキュリティ情報が格納されています。リポジトリ情報をコンフィグレーションす

るには、WebLogic Integration B2B Console を使用する方法と、データ ファイルで情報を指定してから Bulk Loader を使用してリポジトリにインポートする方法があります。

**注意：** WebLogic Integration 2.1 または WebLogic Integration 2.1 SP1 のリポジトリ データ ファイルを WebLogic Integration 7.0 のリポジトリにインポートする場合、その前にリポジトリ データ ファイルの `WLC` 要素の `system-password` 属性を変更して、`wlisystem` の現行パスワードと一致するようにする必要があります。リポジトリの移行の詳細については、『*WebLogic Integration 移行ガイド*』の「WebLogic Integration 2.1 から WebLogic Integration 7.0 への移行」、「ステップ 12. WebLogic Integration アプリケーションの開始およびテスト」を参照してください。

B2B セキュリティ システムの場合、必要に応じて以下のものをコンフィグレーションする必要があります。

- B2B システム パスワード
- 監査ログ クラス
- 証明書検証クラス
- セキュア タイムスタンプ クラス
- 認証局ディレクトリ

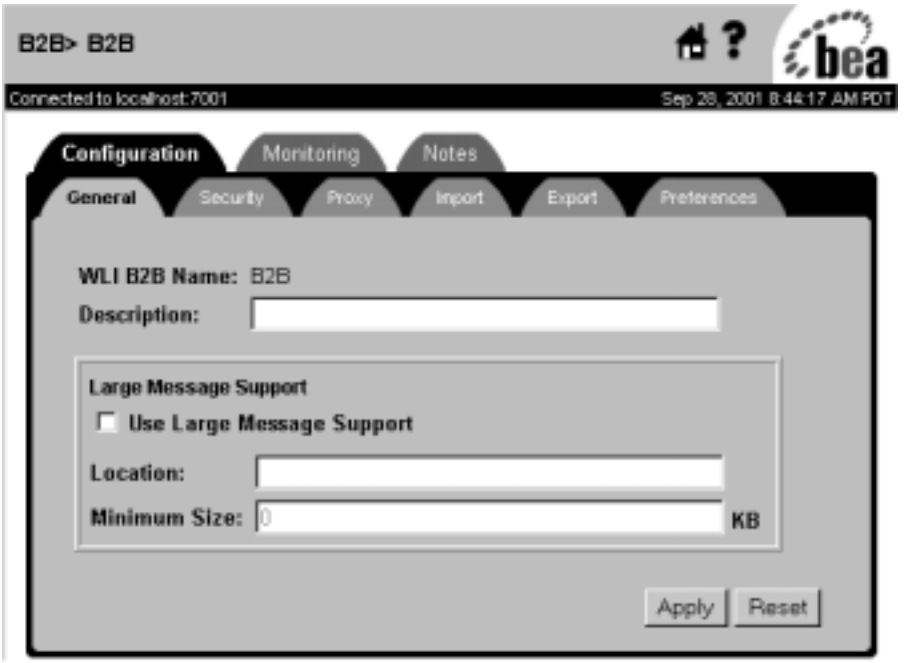
B2B セキュリティ システムのこれらのエンティティをコンフィグレーションするには、以下の手順を実行します。

1. B2B Console を起動します。
2. B2B Console のメイン ペインで、次の図に示す WebLogic Integration B2B の下のリンクをクリックします。

**図 4-8　WebLogic Integration B2B Console メイン ウィンドウ**

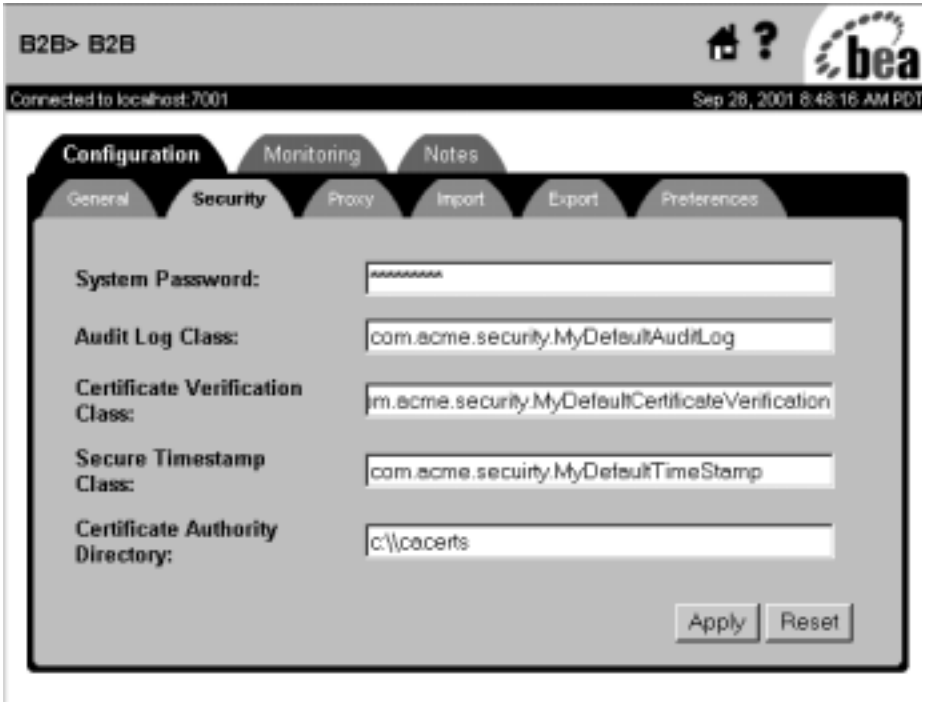

次の図に示す B2B のコンフィグレーション タブが表示されます。

**図 4-9　B2B のコンフィグレーション タブ**




3. [ セキュリティ ] タブを選択します。次の図に示す WebLogic Integration システムの [ セキュリティ ] コンフィグレーション ページが表示されます。

**図 4-10 WebLogic Integration B2B の [ セキュリティ ] コンフィグレーション ページ**




4. 次の表では、[ コンフィグレーション ] パネルの [ セキュリティ ] タブの フィールドについて説明します。新しいコンフィグレーションは、WebLogic Integration システムを再起動した後に有効になります。

**表 4-2 WebLogic Integration B2B セキュリティ システムのコンフィグレーション**

| フィールド | 説明 |
|---|---|
| [ システム パスワード ] | WebLogic Integration B2B システム ユーザのパスワード。このパスワードは、WebLogic Integration ソフトウェアのインストール時に設定され、デフォルトで `wlisystem` となる。ただし、デフォルトを変更する場合は、このフィールドに新しいパスワードを入力する。 |

**表 4-2 WebLogic Integration B2B セキュリティ システムのコンフィグレーション ( 続き )**

| フィールド | 説明 |
|---|---|
| [ 監査ログ クラス ] | 否認防止のための監査ロギングを実装する Java クラス。監査ログを使用すると、会話中にデータ交換に伴って発生したイベントのタスク順序を再現できる。監査ログのコンフィグレーションに従って、監査ログには、トレーディング パートナ間で交換される各ビジネス メッセージがデジタル署名やタイムスタンプなどのデータと一緒に格納される。監査の詳細については、5-5 ページの「セキュア監査ログ サービス」を参照。 |
| [ 証明書検証クラス ] | リモート トレーディング パートナによって提出されたデジタル証明書が有効であることを検証するソフトウェアを呼び出す Java クラス。このクラスは、WebLogic Integration によって提供される OCSP (Online Certificate Status Protocol) アプリケーション、または信頼性のあるセキュリティ プロバイダから入手した証明書検証プロバイダを呼び出すことができる。証明書検証クラスの詳細については、2-2 ページの「トレーディング パートナの証明書の検証」を参照。 |
| [ セキュア タイムスタンプ クラス ] | トレーディング パートナ間で交換されるビジネス メッセージのセキュア タイムスタンプを提供する Java クラス。タイムスタンプは否認防止を目的としている。セキュア タイムスタンプの詳細については、5-3 ページの「セキュア タイムスタンプ サービス」を参照。 |
| [ 認証局のディレクトリ ] | WebLogic Integration リポジトリでコンフィグレーションされているすべてのトレーディング パートナの証明書の認証局を格納している場所。 |

# トレーディング パートナのセキュリティのコンフィグレーション

トレーディング パートナのセキュリティのコンフィグレーションでは、トレーディング パートナごとに以下のものを設定します。

- 証明書
- 転送セキュリティのプロパティ
- ドキュメント交換セキュリティ
- 配信チャネル セキュリティ

ここからは、これらのコンポーネントごとにトレーディング パートナのセキュリティをコンフィグレーションする方法について説明します。

**注意：** Bulk Loader を使用してデータを WebLogic Integration リポジトリにインポートする場合、リポジトリでコンフィグレーションされている各トレーディング パートナを表す WebLogic Server ユーザは自動的には作成されません。これらの WebLogic Server ユーザを手動で作成する必要があります。詳細については、『*B2B Integration 管理ガイド*』の「Bulk Loader の操作」を参照してください。

## トレーディング パートナの証明書のコンフィグレーション

WebLogic Integration B2B では、トレーディング パートナの以下の証明書をコンフィグレーションすることができます。

**表 4-3 WebLogic Integration B2B でコンフィグレーションされるトレーディング パートナの証明書**

| 証明書 | 説明 |
|---|---|
| クライアント証明書 | リモートまたはローカルのトレーディング パートナのデジタル証明書。SSL プロトコルを使用する場合には、クライアント証明書をコンフィグレーションする必要がある。<br>**証明書：**<br>■ タイプ X.509 バージョン 1 または 3<br>■ PEM (Privacy Enhanced Mail) または DER (Definite Encoding Rules) でエンコードされる。ファイル名拡張子がエンコード タイプ（`.pem` または `.der`）を指定する。<br>■ 相互認証の HTTPS が使用される場合、すべてのトレーディング パートナについて必須。<br>**プライベート キー：**<br>■ PEM または DER でエンコードされる。ファイル名拡張子がエンコード タイプ（`.pem` または `.der`）を指定する。<br>■ ローカルのトレーディング パートナにのみ必要となる。<br>■ パスワード保護またはプレーン テキスト。<br>**注意：** B2B Console を使ってプレーン テキストのプライベート キーをインポートする場合、プライベート キーストアのパスワードを使用すること。 |

**表 4-3 WebLogic Integration B2B でコンフィグレーションされるトレーディング パートナの証明書**

| 証明書 | 説明 |
|---|---|
| サーバ証明書 | リモートのトレーディング パートナのデジタル証明書。SSL プロトコルを使用する場合には、サーバ証明書をコンフィグレーションする必要がある。<br>**証明書：**<br>■ タイプ X.509 バージョン 1 または 3<br>■ PEM または DER でエンコードされる。ファイル名拡張子がエンコード タイプ（`.pem` または `.der`）を指定する。<br>■ 相互認証の HTTPS が使用される場合、リモート トレーディング パートナについて必須。 |
| 署名証明書 | 否認防止性の要件であるデジタル署名のサポートが E マーケット用としてコンフィグレーションされている場合、各トレーディング パートナに必要となる証明書。デジタル署名のサポートの詳細については、5-2 ページの「デジタル署名サポート」を参照。<br>**証明書：**<br>■ タイプ X.509 バージョン 1 または 3<br>■ DER でエンコードされる。<br>■ RSA CertJ パッケージを使用して読み込まれる。<br>■ デジタル署名サービスを使用するすべてのトレーディング パートナについて必須。<br>**プライベート キー：**<br>■ 表示形式は PKCS8 フォーマットのみ<br>■ 常にパスワード保護される（パスワードは、`startWeblogic` コマンドのシステム プロパティとして指定する）。 |

**表 4-3 WebLogic Integration B2B でコンフィグレーションされるトレーディング パートナの証明書**

| 証明書 | 説明 |
|---|---|
| 暗号化証明書 | E マーケット用にビジネス メッセージの暗号化をコンフィグレーションする場合に各トレーディング パートナに必要となる証明書。暗号化のサポートは、RosettaNet プロトコルでのみ使用可能となる。メッセージの暗号化の詳細については、4-36 ページの「メッセージ暗号化のコンフィグレーション」を参照。<br>**証明書：**<br>■ タイプ X.509 バージョン 1 または 3<br>■ DER でエンコードされる。<br>■ RSA CertJ パッケージを使用して読み込まれる。<br>■ 暗号化サービスを使用するすべてのトレーディング パートナについて必須。<br>**プライベート キー：**<br>■ 表示形式は PKCS8 フォーマットのみ<br>■ 常にパスワード保護される（パスワードは、`startWeblogic` コマンドのシステム プロパティとして指定する）。 |

トレーディング パートナの証明書のコンフィグレーションに関しては、以下のような一般的な規則があります。

- 各トレーディング パートナは、クライアント証明書を 1 つと、任意の数の暗号化証明書および署名証明書を持つことができます。リモート トレーディング パートナでは、さらに、ホストとなっているシステムに対するサーバ証明書を持ちます。このサーバ証明書の名前は、そのトレーディング パートナのコンフィグレーション時に指定する必要があります。
- 証明書ごとに、トレーディング パートナの種類（ローカルまたはリモート）が指定されています。証明書のコンフィグレーション タブの内容は、トレーディング パートナの種類によって異なります。たとえば、リモートのトレーディング パートナをコンフィグレーションするためのタブには、プライベート キーに関する情報を入力するフィールドがありません。プライベート

キーに関する情報は、ローカルのトレーディング パートナの場合にのみ設定するからです。

- ローカルのトレーディング パートナの場合、サーバ証明書をコンフィグレーションしません。
- ローカルのトレーディング パートナをコンフィグレーションする場合、そのトレーディング パートナに対応する WebLogic Server ユーザ名を指定する必要はありません。唯一の例外は、ローカルのトレーディング パートナがトレーディング パートナ軽量クライアントである場合です。
- パスワードは、ローカル トレーディング パートナの署名証明書およびメッセージ暗号化証明書に対するすべてのプライベート キーに必要です。プライベート キーのパスワードは、証明書がキーストアにインポートする時に設定します。実行時、プライベート キーに対して入力するパスワードが、インポート時に指定されたこのパスワードに一致する必要があります。プライベート キーのパスワードは、`java` コマンド ラインのシステム プロパティとして指定されます。

次の例は、Hello Partner サンプル アプリケーション用の WebLogic Server を起動する `java` コマンドを示しています。

```
%JAVA_HOME%\bin\java -classic -ms64m -ms64m -classpath %START_WL_CLASSPATH%
 -Dbea.home=%BEA_HOME% -Dweblogic.home=%WL_HOME%
 -Dweblogic.system.home=%WLC_SAMPLES_HOME% -Dweblogic.Domain=samples
 -Dweblogic.management.password=%SYSTEM_PASSWORD%
 -Dweblogic.Name=myserver
 -Djava.security.policy=%WL_HOME%\lib\weblogic.policy
 -DKey.certificate-name.password=%PASSWORD% weblogic.Server
```

上の例で、**`certificate-name`** はプライベート キー パスワードが指定される証明書の名前を表し、**`%SYSTEM_PASSWORD%`** および **`%PASSWORD%`** はその 2 つの環境変数の値を表します。

**注意：** パスワードは、`startWeblogic` などのスクリプトにハードコーディングせずに、環境変数に設定することをお勧めします。環境変数が使用された場合、スクリプトでは、そのスクリプトが実行される環境からパスワードの値を取得します。

トレーディング パートナの証明書をコンフィグレーションするには、以下の手順を実行します。

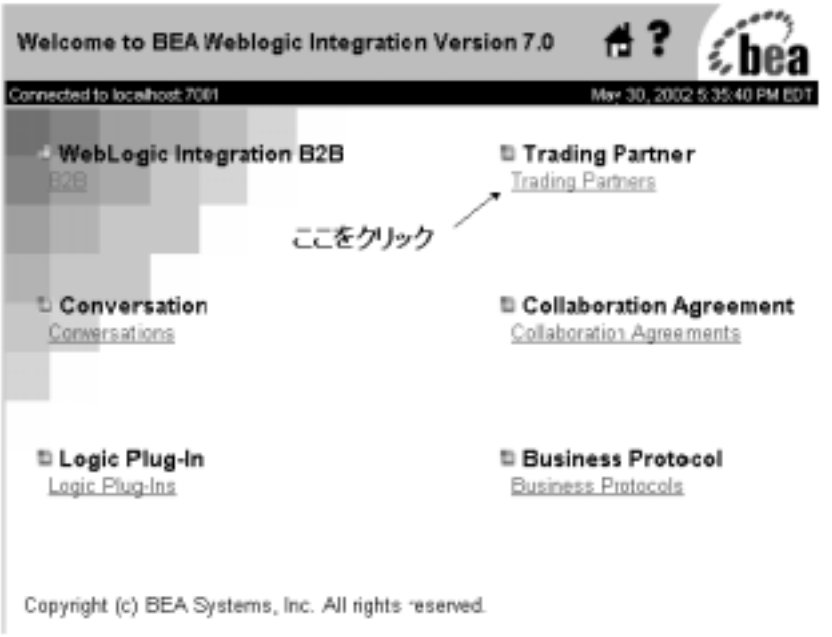
1. 次のいずれかの方法で、メイントレーディング パートナ コンフィグレーション ページを表示します。

- B2B Console のナビゲーション ツリーの [ トレーディング パートナ ] をクリックする。

**図 4-11　ナビゲーション ツリーの [ トレーディング パートナ ] エントリ**


- 右ペインの [ トレーディング パートナ ] リンクをクリックする。

**図 4-12　トレーディング パートナのコンフィグレーション ページへのアクセス方法**




次の図に示すメインの [ トレーディング パートナ ] コンフィグレーション ページが表示されます。このページで、トレーディング パートナを追加、変更、および削除することができます。

**図 4-13　メインのトレーディング パートナ コンフィグレーション ページ**


**注意：** 以下の手順では、次の事項を前提にしています。

- トレーディング パートナがすでに作成され、セキュリティ パラメータを除いてすでにコンフィグレーションされていること。トレーディング パートナのコンフィグレーションの詳細については、『*B2B Integration 管理ガイド*』の「基本的なコンフィグレーション タスク」を参照してください。
- トレーディング パートナの証明書の格納にキーストアを使用している場合、第 3 章「キーストアのコンフィグレーション」の説明に従って、必要なキーストアをローカル システムに作成し、コンフィグレーションしていること。

2. 証明書を追加する対象のトレーディング パートナの名前をクリックします。次の図に示すトレーディング パートナの [ 一般 ] コンフィグレーション ページが表示されます。

**図 4-14　トレーディング パートナの [ 一般 ] コンフィグレーション ページ**

3. [ 証明書 ] タブを選択します。次の図に示すように、トレーディング パートナの証明書をコンフィグレーションするためのページが表示されます。

**図 4-15 トレーディング パートナの [ 証明書 ] コンフィグレーション ページ**


[ 証明書 ] ページでは、トレーディング パートナにコンフィグレーションする証明書の種類ごとに選択可能な証明書を割り当てる、またはトレーディング パートナに新しい証明書を追加することができます。

4. 新しいトレーディング パートナの証明書を追加する場合は、[ 証明書エントリの作成 ] を選択します。次の図に示すように、新しいトレーディング パートナの証明書を追加するページが表示されます。

**図 4-16　新しいトレーディング パートナ証明書の追加**


5. 各トレーディング パートナの証明書をコンフィグレーションするには、次の表に示す手順を実行します。

**表 4-4 トレーディング パートナの証明書のコンフィグレーション**

| コンフィグレーションする証明書 | 実行手順 |
|---|---|
| クライアント証明書 | ローカルまたはリモートのトレーディング パートナをコンフィグレーションする場合<br>1. [ 証明書のタイプ ] 選択ボックスで [ クライアント証明書 ] を選択する。<br>2. [ 証明書名 ] フィールドにクライアント証明書の名前を入力する。<br>3. [ 証明書の位置 ] フィールドにクライアント証明書の、WebLogic Integration マシンにおけるパス名を入力する。<br>4. [ プライベート キーの位置 ] フィールドに、ローカル トレーディング パートナのプライベート キーが格納されている WebLogic Integration マシン上のパス名を入力する。この手順は、ローカルのトレーディング パートナの場合にのみ実行する。<br>5. [ 追加 / 適用 ] をクリックすると WebLogic Integration リポジトリに証明書が追加される。<br>6. [ キーストアへ証明書を保存 ] のチェック ボックスにチェックを入れる。これで、証明書がプライベート キーストアに追加される。<br>**注意：** 1 つのトレーディング パートナに対してコンフィグレーションできるクライアント証明書は 1 つに限られる。 |

**表 4-4 トレーディング パートナの証明書のコンフィグレーション ( 続き )**

| コンフィグレーションする証明書 | 実行手順 |
|---|---|
| サーバ証明書 | リモートのトレーディング パートナをコンフィグレーションする場合<br>1. [ 証明書タイプ ] 選択ボックスで [ サーバ証明書 ] を選択する。<br>2. [ 証明書名 ] フィールドに、リモートのトレーディング パートナの WebLogic Integration システムのサーバ証明書の名前を入力する。<br>3. [ 証明書の位置 ] フィールドに、トレーディング パートナのサーバ証明書の、マシン上におけるパス名を入力する。<br>4. [ 追加 / 適用 ] をクリックすると WebLogic Integration リポジトリに証明書が追加される。<br>5. [ キーストアへ証明書を保存 ] のチェック ボックスにチェックを入れると、証明書がプライベート キーストアに追加される。<br>**注意：** 1 つのリモート トレーディング パートナに対してコンフィグレーションできるサーバ証明書は 1 つに限られる。 |

**表 4-4 トレーディング パートナの証明書のコンフィグレーション ( 続き )**

| コンフィグレーションする証明書 | 実行手順 |
|---|---|
| 署名証明書 | デジタル署名サポートを使用するトレーディング パートナの場合<br>1. [ 証明書タイプ ] 選択ボックスで [ 署名証明書 ] を選択する。<br>2. [ 証明書名 ] フィールドに署名証明書の名前を入力する。<br>3. [ 証明書の位置 ] フィールドに、署名証明書の、マシン上におけるパス名を入力する。<br>4. [ プライベート キーの位置 ] フィールドに、ローカル トレーディング パートナのプライベート キーが格納されている、マシン上のパス名を入力する。この手順は、ローカルのトレーディング パートナの場合にのみ実行する。<br>5. [ 追加 / 適用 ] をクリックすると WebLogic Integration リポジトリに証明書が追加される。<br>6. [ キーストアへ証明書を保存 ] のチェック ボックスにチェックを入れると、証明書がプライベート キーストアに追加される。<br>**注意 :** 署名証明書は、1 つのトレーディング パートナに対して複数の証明書をコンフィグレーションできる。 |

**表 4-4 トレーディング パートナの証明書のコンフィグレーション ( 続き )**

| コンフィグレーションする証明書 | 実行手順 |
|---|---|
| 暗号化証明書 | RosettaNet ベースのビジネス メッセージの暗号化を使用するトレーディング パートナの場合<br>1. [ 証明書タイプ ] 選択ボックスで [ 暗号証明書 ] を選択する。<br>2. [ 証明書名 ] フィールドに暗号化証明書の名前を入力する。<br>3. [ 証明書の位置 ] フィールドに、暗号化証明書の、マシン上におけるパス名を入力する。<br>4. [ プライベート キーの位置 ] フィールドに、ローカル トレーディング パートナのプライベート キーが格納されているマシン上のパス名を入力する。この手順は、ローカルのトレーディング パートナの場合にのみ実行する。<br>5. [ 追加 / 適用 ] をクリックすると WebLogic Integration リポジトリに証明書が追加される。<br>6. [ キーストアへ証明書を保存 ] のチェック ボックスにチェックを入れると、証明書がプライベート キーストアに追加される。<br>**注意 :** 暗号化証明書は、1 つのトレーディング パートナに対して複数の証明書をコンフィグレーションできる。 |

**注意 :** `WebLogic Integration` でトレーディング パートナを作成すると、そのトレーディング パートナに対応する `WebLogic Server` ユーザが指定したユーザ名で実行時に作成されます。ただし、トレーディング パートナを `WebLogic Integration` リポジトリから削除しても、対応する `WebLogic Server` ユーザは自動的には削除されません。トレーディング パートナを削除する場合は、対応する `WebLogic Server` ユーザも手動で削除する必要があります。

リソースのリストについては、次の URL の「BEA dev2dev Online」を参照してください。WebLogic Integration B2B リソースの管理に役立つと思われます。

`http://dev2dev.bea.com/index.jsp`

ここには、デジタル証明書およびプライベート キーを処理するツールなど、便利なユーティリティが用意されているサイトのリンクもあります。

## セキュアな転送方式のコンフィグレーション

トレーディング パートナの転送方式をコンフィグレーションする場合は、トレーディング パートナの転送方式を転送セキュリティ プロトコルにバインドします。たとえば、トレーディング パートナが SSL 証明書を使用するようにコンフィグレーションされている場合、トレーディング パートナの転送方式を、SSL を使用する転送プロトコルにバインドする必要があります。セキュアな転送方式をコンフィグレーションすると、クライアント証明書が発信 SSL 用に使用されます。WebLogic Integration で使用できるクライアント証明書は 1 つなので、セキュアな転送方式をコンフィグレーションする場合にクライアント証明書を選択する必要はありません。

トレーディング パートナのセキュアな転送方式をコンフィグレーションするには、以下の手順を実行します。

1. [ 転送 ] タブを選択します。[ 転送 ] コンフィグレーション ページが表示されます。次の図は、このページのトップを示しています。

**図 4-17　トレーディング パートナの [ 転送 ] コンフィグレーション ページ**


2. 次の表で説明する情報を入力します。

**表 4-5 トレーディング パートナの転送方式のコンフィグレーション**

| フィールド | 説明 |
|---|---|
| [ 転送名 ] | トレーディング パートナの転送方式の名前。名前を入力するか、[ 利用可能な転送 ] ボックスに表示されている使用可能な転送方式の中から選択する。使用可能な転送方式にはそれぞれセキュリティ プロトコルがバインドされているので、このリストから選択すると、転送方式とセキュリティ プロトコルが自動的に設定される。転送方式名の指定の詳細については、[ 転送 ] タブの右上隅の疑問符をクリックしてオンライン ヘルプを参照。 |

**表 4-5 トレーディング パートナの転送方式のコンフィグレーション ( 続き )**

| フィールド | 説明 |
|---|---|
| [ 転送プロトコル ] | 転送方式のセキュリティプロトコル。HTTP-1.1 または HTTPS-1.1 を選択できる。HTTPS-1.1 プロトコルは SSL を使用する。HTTPS-1.1 を選択した場合、セキュリティプロトコルが [ セキュリティプロトコル ] という変更不能なフィールドに表示される。 |
| [URI エンドポイント ] | トレーディング パートナの B2B システム上の転送方式の URI。URI エンドポイントを指定するには、このフィールドに URI を入力するか、このフィールドの下のボックスに表示されている使用可能な URI のいずれかを選択する。URI エンドポイントを入力する場合、[ 設定 ] をクリックして URI を確立するか、[ 削除 ] をクリックして [URI エンドポイント ] フィールドの既存のエントリをクリアする。URI エンドポイントの指定の詳細については、[ 転送 ] タブの右上隅の疑問符をクリックしてオンライン ヘルプを参照。 |

3. [ 追加 / 適用 ] をクリックします。

# セキュアな配信チャネルのコンフィグレーション

トレーディング パートナの配信チャネルをコンフィグレーションする場合、4-29 ページの「セキュアな転送方式のコンフィグレーション」でコンフィグレーションしたセキュアな転送方式にバインドすることで配信チャネルをセキュリティで保護することができます。

セキュアなチャネルをコンフィグレーションするには、以下の手順を実行します。

1. [ 配信チャネル ] タブを選択します。次の図に示す [ 配信チャネル ] コンフィグレーション ページが表示されます。

**図 4-18　トレーディング パートナの [ 配信チャネル ] コンフィグレーション ページ**


2. 次の表で説明する情報を入力します。

**表 4-6　トレーディング パートナの配信チャネルのコンフィグレーション**

| フィールド | 説明 |
|---|---|
| [ 配信チャネル名 ] | 配信チャネル名。このフィールドに名前を入力するか、[ 有効な配信チャネル ] ボックスに示されている配信チャネルから選択する。配信チャネル名の指定の詳細については、[ 配信チャネル ] ページの右上隅の疑問符をクリックしてオンライン ヘルプを参照。 |
| [ 転送 ] | トレーディング パートナの [ 転送 ] タブでコンフィグレーションした転送方式の名前。このフィールドでは、4-29 ページの「セキュアな転送方式のコンフィグレーション」で説明したように、転送方式のプロパティをコンフィグレーションするときにセキュリティで保護した転送方式に配信チャネルをバインドできる。 |

**表 4-6 トレーディング パートナの配信チャネルのコンフィグレーション ( 続き )**

| フィールド | 説明 |
|---|---|
| [ ドキュメント交換 ] | 配信チャネルのバインド先のドキュメント交換の名前。ドキュメント交換と配信チャネルのバインドの詳細については、[ 配信チャネル ] ページの右上隅の疑問符をクリックしてオンライン ヘルプを参照。 |
| [ ルーティング プロキシ ] | トレーディング パートナの配信がルーティング プロキシ（ハブ）の役割を果たすようにする場合は、このボックスをチェックする。プロキシ サーバの詳細については、4-44 ページの「発信 HTTP プロキシ サーバを使用するための WebLogic Integration B2B のコンフィグレーション」を参照。 |

3. [ 追加 / 適用 ] をクリックします。

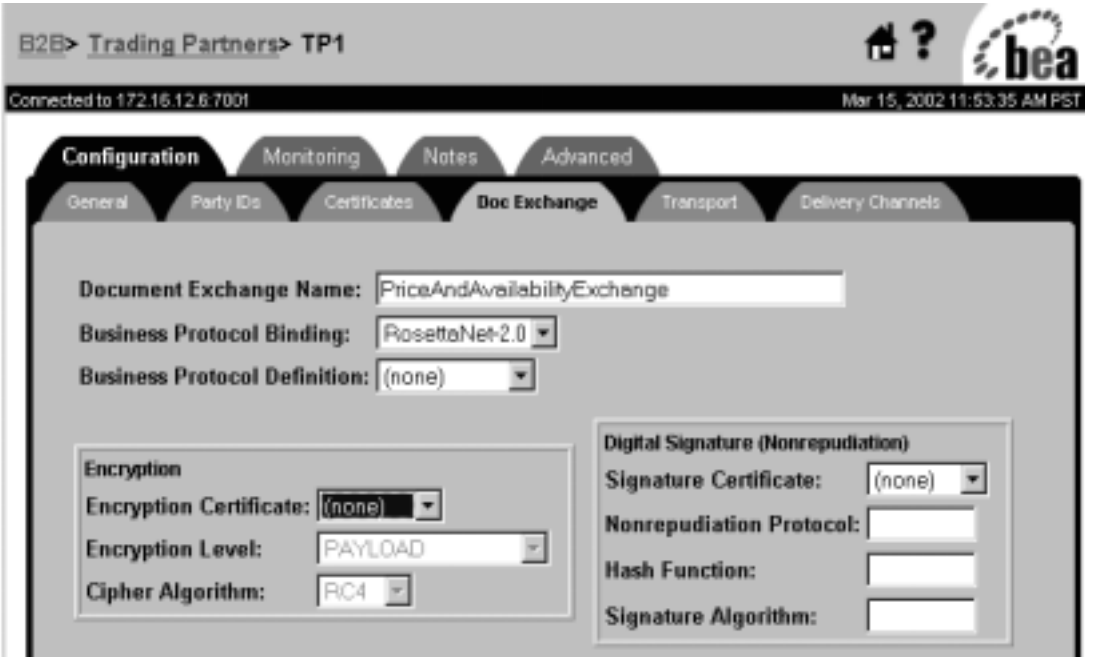
# セキュアなドキュメント交換のコンフィグレーション

トレーディング パートナのドキュメント交換をコンフィグレーションする場合、ドキュメント交換と、デジタル署名またはメッセージ暗号化を提供するビジネス プロトコル バインディングを関連付けることができます。デジタル署名サポートは、WebLogic Integration でサポートされているすべてのビジネス プロトコルで使用可能ですが、メッセージ暗号化は RosettaNet プロトコルを使用した場合にのみ使用できます。

デジタル署名またはメッセージ暗号化をサポートするには、以下の手順を実行します。

1. [ ドキュメント交換 ] タブを選択します。次の図に示す [ ドキュメント交換 ] コンフィグレーション ページが表示されます。

**図 4-19　トレーディング パートナの [ ドキュメント交換 ] コンフィグレーション ページ**




2. 次の表で説明する情報を入力します。

**表 4-7　トレーディング パートナのドキュメント交換のコンフィグレーション**

| フィールド名 | 指定する情報 |
|---|---|
| [ ビジネス プロトコル バインディング ] | デジタル署名またはメッセージ暗号化機能をサポートするビジネス プロトコルとバージョン。選択したプロトコルが、ページ上部に示されているトレーディング パートナのドキュメント交換にバインドされる。 |
| [ ビジネス プロトコル定義 ] | 前の選択ボックスで選択したビジネス プロトコル バインディングと関連付けられるビジネス プロトコル。 |

3. [ ドキュメント交換名 ]、[ エンド ポイントのタイプ ]、[ 配信の確認 ]、[ メッセージの履歴 ]、および [ 再試行 ] フィールドのデータの指定については、[ ドキュメント交換 ] ページの右上隅の疑問符をクリックしてオンライン ヘルプを参照してください。
4. デジタル署名情報のコンフィグレーションの詳細については、4-35 ページの「メッセージ暗号化のコンフィグレーション」を参照してください。
5. メッセージ暗号化情報のコンフィグレーションの詳細については、4-39 ページの「否認防止のためのデジタル署名のコンフィグレーション」を参照してください。

# メッセージ暗号化のコンフィグレーション

第 1 章「WebLogic Integration B2B セキュリティ入門」で説明したように、B2B のメッセージ暗号化サービスでは、暗号化を必要とするビジネス プロトコルに合わせてビジネス メッセージを暗号化します。現在、メッセージ暗号化は RosettaNet 2.0 プロトコルを使用する場合にのみサポートされています。

## WebLogic Integration B2B のメッセージ暗号化のしくみ

データは、送信側の証明書、プライベート キー、および受信側の証明書を組み合わせてビジネス メッセージをエンコードすることで暗号化されます。暗号化したメッセージは、受信側のプライベート キーを使用しないと解読できません。

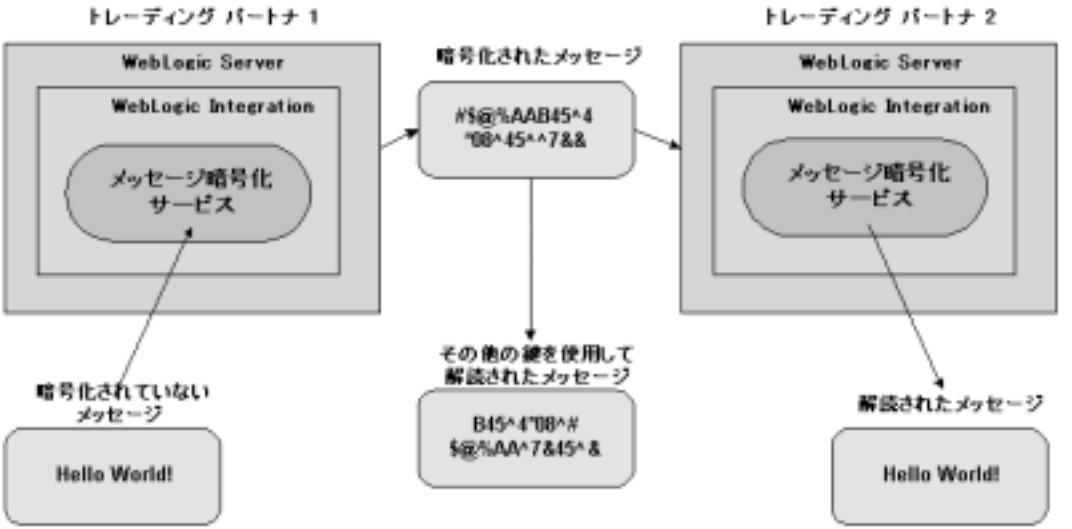
**注意：** B2B のメッセージ暗号化機能はライセンス（Encryption/Domestic または Encryption/Export）によって制御されますが、ビジネス メッセージの解読機能は制御されません。WebLogic Integration が有効な暗号化ライセンスを持っていない場合、B2B エンジンは暗号化サービスを無効にします。ただし、B2B エンジンは、受け取ったビジネス メッセージを常に解読することができます。

WebLogic Integration メッセージ暗号化サービスでは、次のアルゴリズムをサポートしています。

- RSA (Rivest-Shamir-Adleman)
- DES (Data Encryption Standard)
- 3DES (Triple Data Encryption Standard)

次の図は、公開鍵とプライベート キーを使用してデータを暗号化する方法を示しています。

**図 4-20　WebLogic Integration B2B のメッセージ暗号化サービス**




**注意：** メッセージを暗号化するには、暗号化サービスを使用するための有効なライセンスが必要です。

# メッセージ暗号化のコンフィグレーション

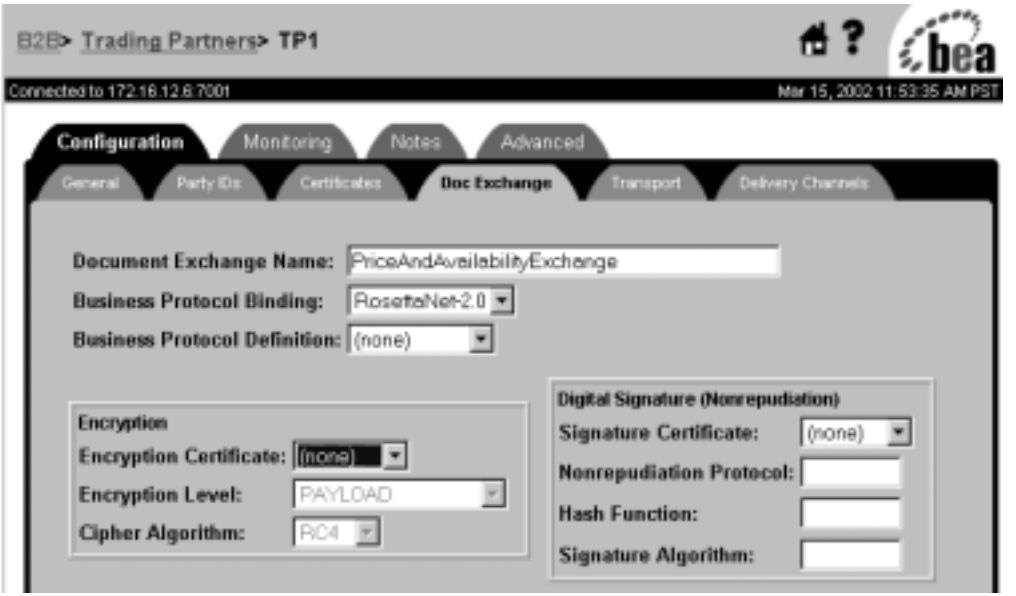
RosettaNet 2.0 ベースの会話定義のトレーディング パートナが交換するビジネス メッセージのメッセージ暗号化をコンフィグレーションするには、以下の手順を実行します。

1. 『*B2B Integration 管理ガイド*』の「基本的なコンフィグレーション タスク」に説明されているようにトレーディング パートナをコンフィグレーションします。

2. 4-31 ページの「セキュアな配信チャネルのコンフィグレーション」で説明されているように、トレーディング パートナの配信チャネルのセキュリティをコンフィグレーションします。適切な RosettaNet 2.0 プロトコル バインディングを使用する転送方式を使用する配信チャネルをコンフィグレーションします。
3. 4-33 ページの「セキュアなドキュメント交換のコンフィグレーション」で説明されているように、トレーディング パートナのドキュメント交換をコンフィグレーションします。適切な RosettaNet 2.0 ビジネス プロトコル バインディングをサポートするようにドキュメント交換をコンフィグレーションします。

   [ ドキュメント交換 ] コンフィグレーション タブで RosettaNet ビジネス プロトコル バインディングを選択すると、[ 暗号 ] ボックスがそのタブの左下隅に表示されます。次の図は、[ 暗号 ] ボックスが表示された状態の [ ドキュメント交換 ] コンフィグレーション タブを示しています。

**図 4-21 [ ドキュメント交換 ] コンフィグレーション ページのメッセージ暗号化用のコンフィグレーション ボックス**




4. [ 暗号 ] ボックスで、次の表で説明する情報を指定します。

**表 4-8 メッセージ暗号化のコンフィグレーション設定**

| フィールド | 説明 |
|---|---|
| [ 暗号証明書 ] | 4-15 ページの「トレーディング パートナの証明書のコンフィグレーション」でコンフィグレーションした暗号化証明書の名前を入力する。 |
| [ 暗号化レベル ] | 暗号化対象となるビジネス メッセージの部分を入力する。メッセージの XML ビジネス ドキュメント部分のみを暗号化する場合は、[PAYLOAD] を選択する。ビジネス ドキュメントとメッセージ内のすべての添付ファイルを暗号化する場合は、[ENTIRE_PAYLOAD] を選択する。 |
| [ 暗号アルゴリズム ] | 使用された暗号アルゴリズムの名前が表示される変更不能の情報フィールド。WebLogic Integration で使用可能な暗号アルゴリズムは次のとおり。<br>■ RSA (Rivest-Shamir-Adleman)<br>■ DES (Data Encryption Standard)<br>■ 3DES (Triple Data Encryption Standard) |

5. [ 追加 / 適用 ] をクリックします。

**注意：** リポジトリ データ ファイルで暗号強度が指定されている場合、実行時には無視されます。

# 否認防止のためのデジタル署名のコンフィグレーション

デジタル署名サポート（5-1 ページの「否認防止性の実装」を参照）は、特にトレーディング パートナ間で転送しているときに、ビジネス メッセージの内容が改ざんされることを防ぐ手段を提供します。デジタル署名サポートは否認防止の要件です。

否認防止性を実装する場合、以下の手順を実行して、B2B エンジンでデジタル署名サポートをコンフィグレーションする必要があります。

1. 『*B2B Integration 管理ガイド*』の「基本的なコンフィグレーション タスク」に説明されているようにトレーディング パートナをコンフィグレーションする。
2. 4-15 ページの「トレーディング パートナの証明書のコンフィグレーション」で説明されているように、トレーディング パートナの署名証明書をコンフィグレーションします。
3. 4-31 ページの「セキュアな配信チャネルのコンフィグレーション」で説明されているように、トレーディング パートナの配信チャネルのセキュリティをコンフィグレーションします。適切なプロトコル バインディングを使用する転送方式を使用する配信チャネルをコンフィグレーションします。
4. 4-33 ページの「セキュアなドキュメント交換のコンフィグレーション」で説明されているように、トレーディング パートナのドキュメント交換をコンフィグレーションします。適切なビジネス プロトコル バインディングをサポートするようにドキュメント交換をコンフィグレーションします。
5. [ ドキュメント交換 ] タブの右下隅にある [ デジタル署名（否認防止）] というボックスに注目します。このボックスで、4-15 ページの「トレーディング パートナの証明書のコンフィグレーション」で識別されるトレーディング パートナの署名証明書を選択します。

   署名証明書を選択すると、次の図の右下隅に示すように、署名証明書に関連付けられた変更不能なフィールドにデータが表示されます。

**図 4-22　否認防止性のコンフィグレーション**


これらの変更不能なフィールドの用途は以下のとおりです。

- [ 否認防止プロトコル ] ー署名証明書に関連付けられたビジネス プロトコルを識別します。
- [ ハッシュ関数 ] ートレーディング パートナ間で交換される暗号化パスワードに使用される関数を識別します。WebLogic Integration において RosettaNet および XOCP の両プロトコルで使用されるハッシュ関数は SHA1 です。
- [ 署名アルゴリズム ] ートレーディング パートナ間で交換される署名証明書の暗号化に使用されるアルゴリズムを識別します。WebLogic Integration において RosettaNet および XOCP の両プロトコルで使用される署名アルゴリズムは RSA です。

# WLCCertAuthenticator クラスのカスタマイズ

WLCCertAuthenticator クラスは、WebLogic Server CertAuthenticator クラスの実装です。WLCCertAuthenticator クラスのデフォルト実装は、トレーディング パートナのデジタル証明書を、WebLogic Integration リポジトリに定義されている、対応するトレーディング パートナにマップします。この機能を拡張すると、トレーディング パートナ以外のユーザに対しても相互認証を使用できます。拡張例として、Web ブラウザまたは Java クライアントを WebLogic Server ユーザにマップするようにクラスを変更する方法があります。

WLCCertAuthenticator クラスは、トレーディング パートナと WebLogic Server 間で SSL 接続が確立されてから、WebLogic Server によって呼び出されます。このクラスは、デジタル証明書からデータを抽出して、デジタル証明書に対応するトレーディング パートナ名を調べます。

次のコード例は、ユーザを取得するための WebLogic のデフォルト レルムを使用しており、WLCCertAuthenticator クラスのカスタマイズ方法を示しています。

```
public User authenticate(String userName, Certificate[] certs, boolean ssl)
{

String user = null;

// SSL が使用されていない場合は復帰する
if (ssl == false)
{
return null;
}

// 証明書が c ハブまたはトレーディング パートナのいずれかであることを検証する
// 検証後、対応する WLS ユーザを返す

if ((user = Security.isValidWLCCertificate(certs))!= null)
{
return realm.getUser(user);
}
// 証明書は有効な WLC 証明書ではない
// ここで非 WLC 証明書をチェックして対応するユーザを返す
}
```

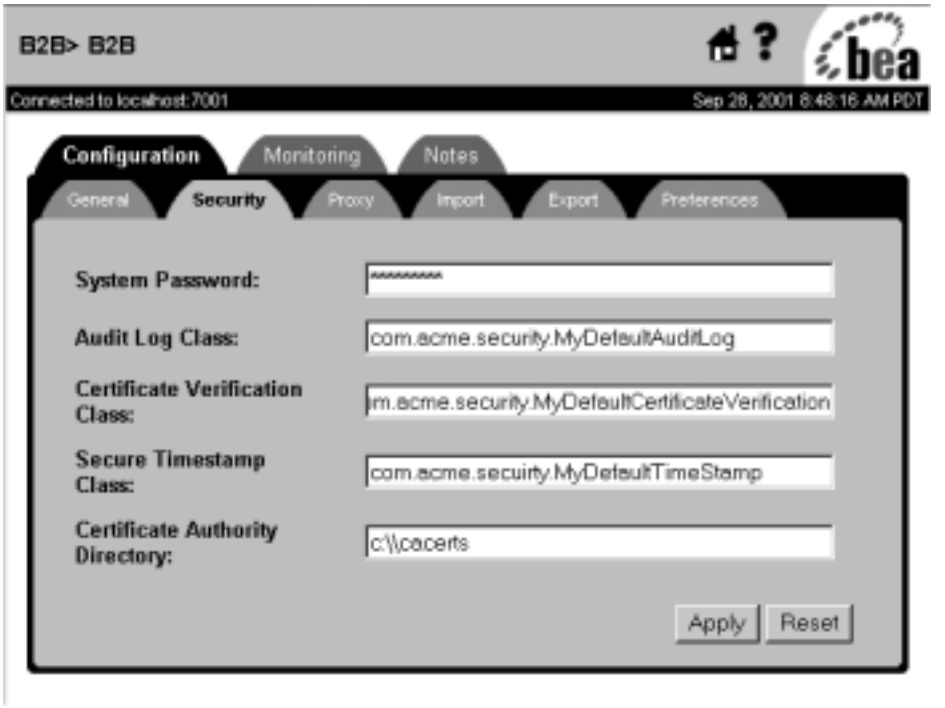
# 証明書検証プロバイダ インタフェースのコンフィグレーション

2-2 ページの「トレーディング パートナの証明書の検証」で説明したように、証明書検証プロバイダを使用して、トレーディング パートナのデジタル証明書を検証できます。証明書検証プロバイダ（CVP）を使用する場合、ここで説明する手順に従って、B2B Console でコンフィグレーションする必要があります。

CVP をコンフィグレーションするには、以下の手順を実行します。

1. B2B Console を起動します。
2. 4-9 ページの「WebLogic Integration B2B エンジンのセキュリティのコンフィグレーション」で説明したように、B2B Console のメイン ページで WebLogic Integration B2B の下のリンクをクリックします。
3. B2B の [ コンフィグレーション ] パネルで [ セキュリティ ] タブを選択します。次の図に示すページが表示されます。

**図 4-23 WebLogic Integration B2B システムのセキュリティ コンフィグレーション ページ**




4. [ 証明書検証クラス ] フィールドに、CVP を実装する Java クラスの完全修飾名を入力します。
5. [ 適用 ] をクリックします。

**注意：** 証明書検証プロバイダをロードするには、Bulk Loader を使用します。詳細については、『*B2B Integration 管理ガイド*』の「Bulk Loader の操作」を参照してください。

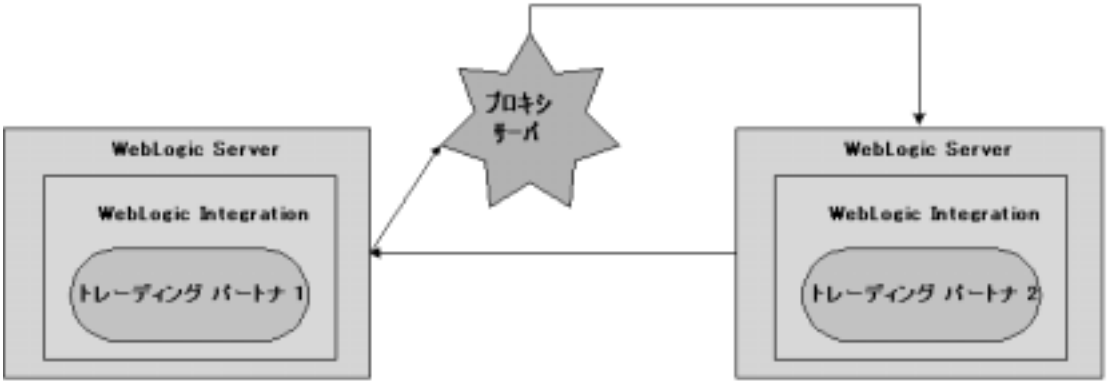
# 発信 HTTP プロキシ サーバを使用するための WebLogic Integration B2B のコンフィグレーション

高度なセキュリティで保護された環境で WebLogic Integration を使用する場合、WebLogic Integration をプロキシ サーバ越しに使用すると効果的です。プロキシ サーバを使用すると、トレーディング パートナはセキュリティを危険にさらすことなくイントラネットまたはインターネット経由で通信できます。プロキシ サーバの用途は以下のとおりです。

- WebLogic Integration のホストとなっている WebLogic Server のローカル ネットワーク アドレスの、外部ハッカーからの隠ぺい
- 外部ネットワークへのアクセスの制限
- WebLogic Integration のホストとなっている WebLogic Server への外部ネットワークからのアクセスのモニタ

プロキシ サーバをローカル ネットワーク上でコンフィグレーションすると、ネットワーク トラフィック（SSL と HTTP）は、プロキシ サーバを経由して外部ネットワークにトンネリングされます。次の図は、WebLogic Integration 環境でのプロキシ サーバの使用方法を示しています。

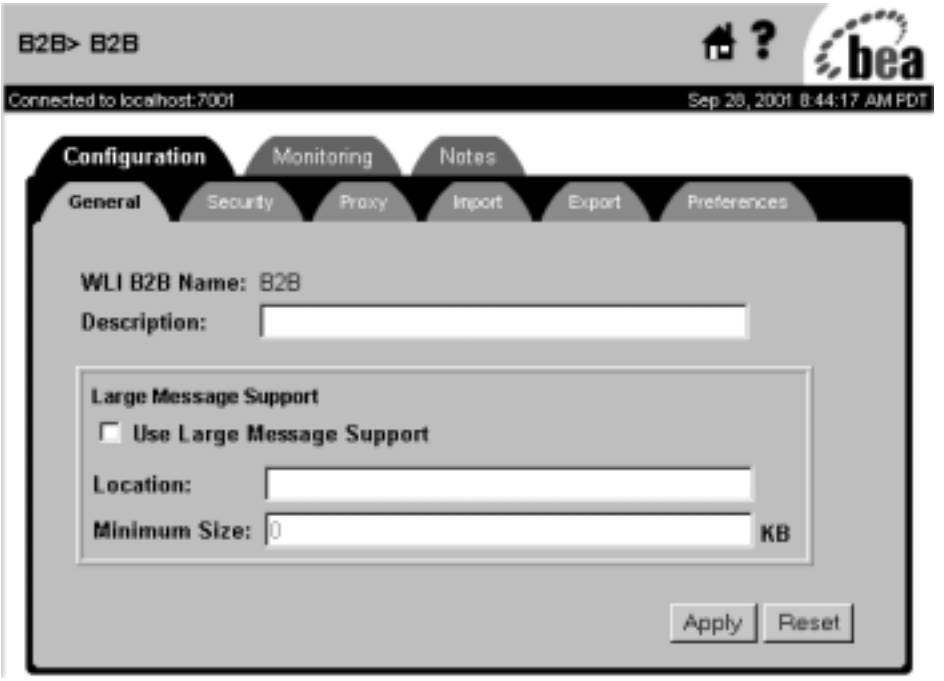
**図 4-24 プロキシ サーバ**



WebLogic Integration 用のプロキシ サーバをコンフィグレーションするには、以下の手順を実行します。

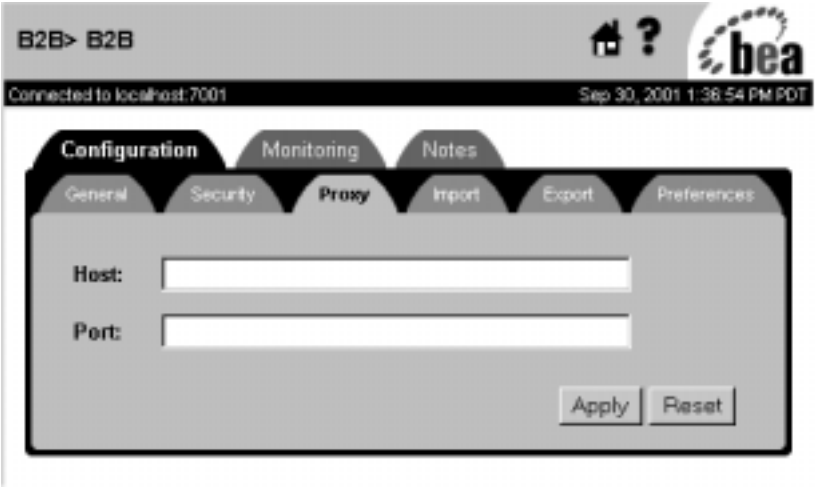
1. 次の図に示すように、B2B Console ウィンドウの右ペインにコンフィグレーション タブを表示します。

**図 4-25　WebLogic Integration B2B Console のコンフィグレーション タブ**




2. [ プロキシ ] タブを選択します。次の図に示す [ プロキシ ] コンフィグレーション ページが表示されます。

**図 4-26 WebLogic Integration のプロキシ サーバ コンフィグレーション ページ**




3. [ ホスト ] フィールドに、WebLogic Integration サーバ用のプロキシ サーバのアドレスを入力します。例 :

   `myproxy.mycompany.com.`

4. [ ポート ] フィールドにプロキシ サーバのポート番号を入力します。
5. [ 適用 ] をクリックします。
6. WebLogic Server の `ssl.proxyHost` および `ssl.proxyPort` システム プロパティを読み書きするパーミッションを追加します。これらのシステム プロパティは、WebLogic Server のインストール ディレクトリにある `weblogic.policy` ファイルに格納されています。以下の行を、`weblogic.policy` ファイルの *`grant`* セクションに追加します。

```
permission java.util.PropertyPermission "ssl.proxyHost", "read, write";
permission java.util.PropertyPermission "ssl.proxyPort", "read, write";
```

# Web サーバおよび WebLogic プロキシ プラグインでの WebLogic Integration のコンフィグレーション

リモートのトレーディング パートナからのビジネス メッセージを処理するようにプログラムされている Web サーバ（Apache サーバなど）で WebLogic Integration をコンフィグレーションすることができます。Web サーバは以下のサービスを提供できます。

- リモートのトレーディング パートナからのビジネス メッセージの受信
- トレーディング パートナのデジタル証明書の認証

Web サーバは WebLogic プロキシ プラグインを使用します。プラグインをコンフィグレーションすると、以下のサービスを提供できます。

- Web サーバが受信したビジネス メッセージを、セキュアな内部ネットワークの内側で動作している WebLogic Integration に転送します。
- Web サーバからリモートのトレーディング パートナの証明書を抽出し、認証のために WebLogic Server に転送します。WebLogic Integration は、トレーディング パートナの証明書とビジネス メッセージを認証します。

次の図は、Web サーバ、WebLogic プロキシプラグイン、および WebLogic Integration を使用する環境のトポロジを示しています。

**図 4-27 Web サーバおよび WebLogic プロキシ プラグインの使い方**


**注意：** プロキシ プラグインでは HTTP を使用しますが、プロキシ プラグインでビジネス メッセージを転送する場合は、HTTPS プロトコルを使用するように WebLogic Integration をコンフィグレーションする必要があります。

会話におけるトレーディング パートナが、プロキシ サーバに Microsoft IIS を使用する場合、その会話で使用される証明書は、すべて、Verisign、Entrust などのよく知られた認証局によって承認されていなければなりません。自己署名証明書を使用すると、IIS プロキシ サーバ経由で渡された要求がエラーになります。これは、IIS における制限で、WebLogic Integration の制限ではありません。

# Web サーバのコンフィグレーション

Web サーバをコンフィグレーションする方法については、『*WebLogic Server 管理者ガイド*』の「WebLogic Server Web コンポーネントのコンフィグレーション」を参照してください。

次のコード例は、プロキシ プラグインをコンフィグレーションするための httpd.conf (Apache サーバ用 ) のセグメントを示しています。

```
# LoadModule foo_module libexec/mod_foo.so
LoadModule weblogic_module    libexec/mod_wl_ssl.<suffix>

<Location /weblogic>
 SetHandler weblogic-handler
```

```
 PathTrim /weblogic
 WebLogicHost myhost
 WebLogicPort 80
</Location>
```

# トレーディング パートナの WebLogic Server ユーザ アイデンティティ

WebLogic Server ユーザ アイデンティティは、リモートのトレーディング パートナをコンフィグレーションするときに省略できます。特定の WebLogic Integration B2B デプロイメントが厳重なセキュリティを必要とする場合、以下のように設定することをお勧めします。

- 転送サーブレットの ACL をコンフィグレーションして、リモートのトレーディング パートナの証明書のマップ先となる WebLogic Server ユーザに対するパーミッションを有効にします。
- 不明または無効な証明書を持つユーザが WebLogic Server システムに入れないように、ゲスト ユーザを無効にします。

# Business Process Management による WebLogic Integration リポジトリへのアクセスの管理

WebLogic Integration で提供されている Business Process Management (BPM) 機能を使用する予定の場合、BPM ユーザが WebLogic Integration リポジトリへのアクセスを共有できるようにしておく必要があります。そのようなアクセスを可能にするには、リポジトリを使用するパーミッションで BPM をコンフィグレーションする必要があります。WebLogic Server グループ `wlpiUsers` を、WebLogic Integration リポジトリが使用する JDBC 接続プールの ACL に追加します。

さらに、WebLogic Integration Studio または Worklist ユーティリティのユーザが WebLogic Integration リポジトリに格納されているワークフロー テンプレートにアクセスする必要がある場合、そのユーザを WebLogic Server 管理 MBean の適切な ACL に追加する必要があります。そのためには、WebLogic Server MBean の以下の ACL をそのユーザに対して指定します。各設定で、*<user>* は BPM ユーザの名前に置き換えてください。

```
acl.access.weblogic.admin.mbean.MBeanHome=<user>
acl.lookup.weblogic.admin.mbean.MBeanHome=<user>
```

B2B リソースの ACL のコンフィグレーションについては、4-7 ページの「WebLogic Integration B2B のアクセス制御リストのコンフィグレーション」を参照してください。

# サーバ側認証のコンフィグレーション

デフォルトでは、WebLogic Integration B2B は双方向 SSL 認証を使用します。しかし、トレーディング パートナ間では証明書ベースの認証を必要としない場合、サーバ側認証を使用すると便利です。

サーバ側認証をコンフィグレーションするには、以下の手順を実行します。

1. B2B ドメインで稼働しているサーバを停止します。
2. ドメインのルート ディレクトリに進みます。
   例：

   ```
   c:\bea\user_projects\domain
   ```

3. テキスト エディタで、`config.xml` ファイルを開きます。
4. WebLogic Server SSL パラメータ、`ClientCertificateEnforced` を `false` に設定します。
5. WebLogic Server SSL パラメータ、`TwoWaySSLEnabled` を `true` に設定します。
6. `config.xml` ファイルの変更を保存します。
7. B2B ドメインのサーバ、および B2B Console を起動します。

8. 3-20 ページの「証明書およびプライベート キーのキーストアからの削除」の説明に従って、各リモート トレーディング パートナのクライアント証明書を削除します。

# 5　否認防止性の実装

ここでは、以下の内容を取り上げます。

- 否認防止性の概要
- 否認防止用の SPI の使い方

## 否認防止性の概要

否認防止性とは、トレーディング パートナが別のトレーディング パートナ宛てのビジネス メッセージを送信したこと証明したり、トレーディング パートナからのビジネス メッセージを受信したことを証明できることを表します。例を挙げて説明します。

トレーディング パートナ A は、トレーディング パートナ B から 1000 脚のエルゴノミクス チェアを購入することに同意しました。そこでトレーディング パートナ A は、チェアを定価で購入することに同意するビジネス メッセージをトレーディング パートナ B に送信しました。しかし後になって、トレーディング パートナ A は元の価格に異議を唱え、その価格を支払うことに同意することを伝えたメッセージを送信したことを否定します。

信頼性の高い否認防止システムが用意されていれば、トレーディング パートナ B は、トレーディング パートナ A が支払うことに同意した金額を明記したトレーディング パートナ A からのドキュメントを示すことで、トレーディング パートナ A の主張を退けることができます。さらに、この元のドキュメントが、信頼性のあるサードパーティ ソースによってデジタル署名され、タイムスタンプを付けられ、記録されていれば、このドキュメントは法的に完全な有効性を持ちます。

否認防止性、つまり否認を主張する当事者の関与を示す法的証拠を提示する能力は、重要なビジネス メッセージには不可欠です。WebLogic Integration B2B は、送信側の否認防止性と受信側の否認防止性の両方をサポートしています。

- 送信側の否認防止性では、ビジネス メッセージとメッセージの送信側をリンクします。これにより、ビジネス メッセージを送信したことの法的証拠が提供されます。
- 受信側の否認防止性では、ビジネス メッセージとメッセージの受信側をリンクします。これにより、ビジネス メッセージを受信したことの法的証拠が提供されます。

否認防止性をサポートするため、B2B engine には以下のサービスが組み込まれています。

- デジタル署名
- セキュア タイムスタンプ
- セキュア監査ログ

ここからは、それぞれのサービスと、B2B 環境にこれらのサービスを組み込む方法について説明します。

# デジタル署名サポート

デジタル署名サポートの目的は、特に 2 つのトレーディング パートナ間を転送しているときに、ビジネス メッセージの内容が改ざんされることを防ぐ手段を提供することです。B2B engine は、デジタル署名の Public Key Cryptography Standard 7 (PKCS7) パッケージングに準拠したデジタル署名サポートを提供します。

デジタル署名自体は、ビジネス メッセージに追加されるデータのセットです。データの内容は、特定のフォーマット（PKCS7 SignedData など）でパッケージ化されたデータを暗号化した一方向のハッシュ値です。デジタル署名の目的は以下のとおりです。

- デジタル署名されたメッセージの内容が改ざんされていないことを検証します。
- ビジネス メッセージの送信側のアイデンティティを格納します。

デジタル署名の作成に必要なデータは、リポジトリ内のトレーディング パートナのコンフィグレーション データから取り出されます。デジタル署名の作成には、以下の情報も必要となります。

- トレーディング パートナの署名証明書とプライベート キー
- トレーディング パートナの署名証明書の認証局の証明書
- ハッシュ アルゴリズム名 : SHA1
- 署名アルゴリズム名 : RSA

## デジタル署名サポートで使用可能なビジネス プロトコル

WebLogic Integration は、以下のビジネス プロトコルを使用するメッセージに対してデジタル署名サポートを提供します。

- RosettaNet 1.1
- RosettaNet 2.0
- XOCP 1.1

## デジタル署名サポートのコンフィグレーション

WebLogic Integration をコンフィグレーションする場合、デジタル署名サービスを指定するオプションを選択できます。デジタル署名サービスを使用するには、4-39 ページの「否認防止のためのデジタル署名のコンフィグレーション」で説明されているようにコンフィグレーションする必要があります。

# セキュア タイムスタンプ サービス

否認防止性を使用する場合、ビジネス メッセージがセキュア監査ログに書き込まれるときに、セキュア タイムスタンプ サービスがセキュア監査ログに UTC (Coordinated Universal Time: 協定世界時 ) タイムスタンプを付ける必要があります。たとえば、ビジネス メッセージを受け取った場合、タイムスタンプは受信側の否認防止性（NRR）メッセージとして監査ログに入力されます。ビジネス メッセージを送信した場合、タイムスタンプは送信側の否認防止性（NRO）メッセージとして監査ログに入力されます。WebLogic Integration B2B には SPI (Service Provider Interface: サービス プロバイダ インタフェース ) が含まれているので、信頼性のあるサードパーティ プロバイダのセキュア タイムスタンプ サービスを組み込むことができます。

信頼性のあるサードパーティ プロバイダのセキュア タイムスタンプ サービスを組み込む場合、com.bea.b2b.security.TimestampProvider インタフェースを実装する Java クラス ファイルを作成する必要があります。com.bea.b2b.security.TimestampProvider インタフェースを実装するクラスのクラス メソッド（getTimestamp など）で、サードパーティのタイムスタンプ プロバイダを呼び出します。このアプリケーションの作成の詳細については、5-10 ページの「セキュア タイムスタンプ サービス用の SPI の使い方」を参照してください。

B2B engine では、複数のセキュア タイムスタンプ プロバイダを WebLogic Integration に登録することは禁止されています。この制限により、WebLogic Integration で作成されるすべてのタイムスタンプは必ず日付順に並べられます。

**注意：** セキュア タイムスタンプ サービスを WebLogic Integration でコンフィグレーションしない場合、システム イベントおよび署名にはシステム時刻に基づいてタイムスタンプが付けられます。

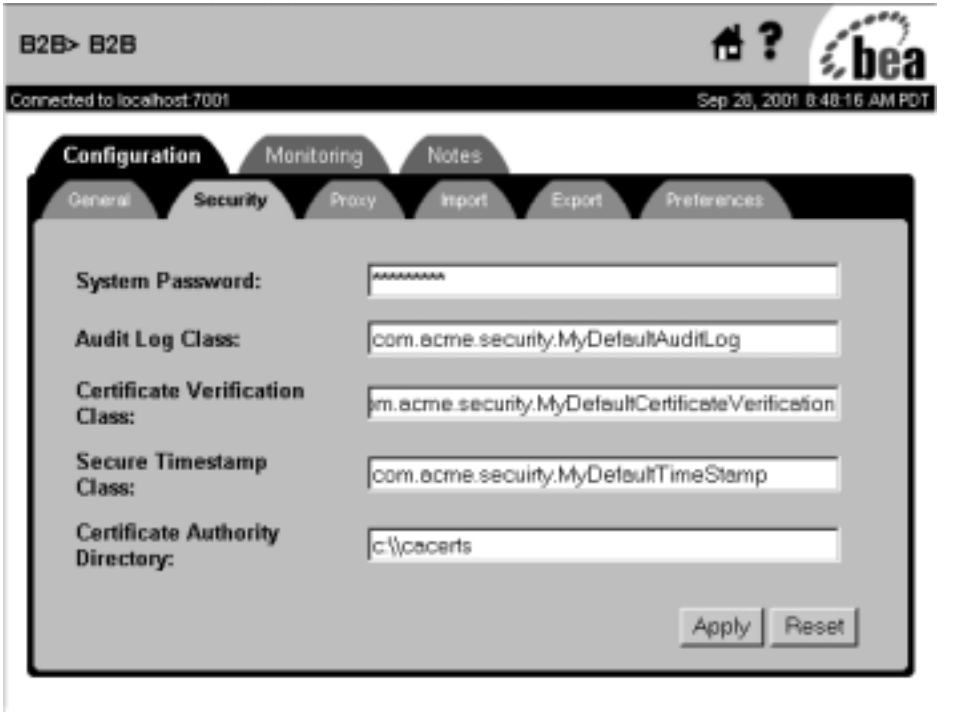
セキュア タイムスタンプ SPI の詳細については、5-10 ページの「セキュア タイムスタンプ サービス用の SPI の使い方」を参照してください。

# セキュア タイムスタンプ サービスのコンフィグレーション

セキュア タイムスタンプ サービスをコンフィグレーションするには、以下の手順を実行します。

1. 4-9 ページの「WebLogic Integration B2B エンジンのセキュリティのコンフィグレーション」で説明しているように、WebLogic Integration B2B Console を起動し、B2B コンフィグレーション ページを表示します。
2. [ セキュリティ ] タブを選択します。次の図に示す B2B の [ セキュリティ ] コンフィグレーション ページが表示されます。

**図 5-1　B2B の [ セキュリティ ] コンフィグレーション ページ**




3. [ セキュア タイムスタンプ クラス ] フィールドに、セキュア タイムスタンプ インタフェースを実装する Java クラスの完全修飾名を入力します。
4. 新しいコンフィグレーションを有効にするため WebLogic Server を再起動します。

# セキュア監査ログ サービス

セキュア監査ログも否認防止に欠かせません。通常、このログには、各ビジネス メッセージをデジタル署名およびセキュア タイムスタンプと一緒に格納します。監査ログを使用すると、トレーディング パートナ間でビジネス メッセージを交換しているときに発生したシステム イベントとメッセージのタスク順序を再現できます。

タイムスタンプ サービスと同じように、B2B engine が提供する SPI を使用して、セキュア監査ログの信頼性のあるサードパーティ プロバイダをコンフィグレーションすることができます。信頼性のあるサードパーティ プロバイダのセキュア監査ログ サービスを組み込む場合、`com.bea.b2b.security.AuditLogProvider` インタフェースを実装するクラス ファイルを作成する必要があります。`com.bea.b2b.security.AuditLogProvider` インタフェースを実装するクラスのクラス メソッド（`log` など）で、サードパーティの監査ログ プロバイダを呼び出します。この実装の作成の詳細については、5-11 ページの「セキュア監査ログ サービス用の SPI の使い方」を参照してください。

**注意：** セキュア監査ログ サービスのサードパーティ プロバイダをコンフィグレーションしない場合、B2B システムでは、`secureaudit.log` というデフォルトの監査ログが提供されます。このログを有効にするには、システム プロパティ `bea.secureaudit` を `on` に設定します。このファイルは、B2B 内のロギング サブシステムに基づいており、基盤となるオペレーティング システムのファイル パーミッション システムによってしか保護されません。このファイルは、デジタル署名されたり、暗号化されたりしません。

# 監査ログへの書き込み

監査ログに書き込むアプリケーションを呼び出す `com.bea.b2b.security.AuditLogProvider` インタフェースの Java 実装を作成する代わりに、この節のコード リストで示すように、`com.bea.b2b.security.Audit.log(byte[] data)` メソッドの呼び出しを使用して監査ログに直接書き込むアプリケーションを作成することもできます。

この例は、次のディレクトリにある `HelloPartnerServlet.java` クラスを変更したものです。`SAMPLES_HOME` は、サンプル アプリケーションがインストールされているディレクトリを表しています。

- **Windows**

```
%SAMPLES_HOME%\integration\samples\HelloPartner\src\wlcsamples\servlets
```

- **UNIX**

```
$SAMPLES_HOME/integration/samples/HelloPartner/src/wlcsamples/servlets
```

この例では、太字のコードは監査ログに書き込むことを示すために追加された文を示しています。

**コード リスト 5-1 監査ログへの書き込み例**

```
package wlcsamples.servlets;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.lang.*;
import javax.transaction.*;
import javax.naming.*;
import javax.jms.*;
import weblogic.jms.extensions.WLTopicSession;
import weblogic.jms.extensions.XMLMessage;
import org.w3c.dom.*;
import org.apache.html.dom.*;
import org.apache.xml.serialize.*;
import org.apache.xerces.dom.*;
import org.apache.xerces.parsers.DOMParser;
import org.xml.sax.*;
import com.bea.eci.logging.*;

//セキュリティ パッケージから Audit クラスをインポートする
import com.bea.b2b.security.Audit;
...
  protected void printResultHTML(PrintWriter pw, Document resultDoc)
  {
    try {
      pw.println("<P><CENTER><P><BR>   <b>Hello Partner Sample</b><BR>");
      if( resultDoc != null ) {
        Element root = resultDoc.getDocumentElement();
        NodeList productList =
          root.getElementsByTagName("integer-product");
        NodeList noteList =
          root.getElementsByTagName("note");
        Node childProduct = productList.item(0);
        Node childNote = noteList.item(0);
        if( childProduct == null || childNote == null ) {
          pw.println("<BR> The Replier Partner has responded
            with a document of unexpected structure...");
        else {
          String product = ((Text)childProduct.
            getFirstChild()).getData();
          String note = ((Text)childNote.getFirstChild()).
            getData();
          // 注（note）を Audit ログに記録する
          byte[] ba = note.getBytes();
          Audit.log(ba);
```

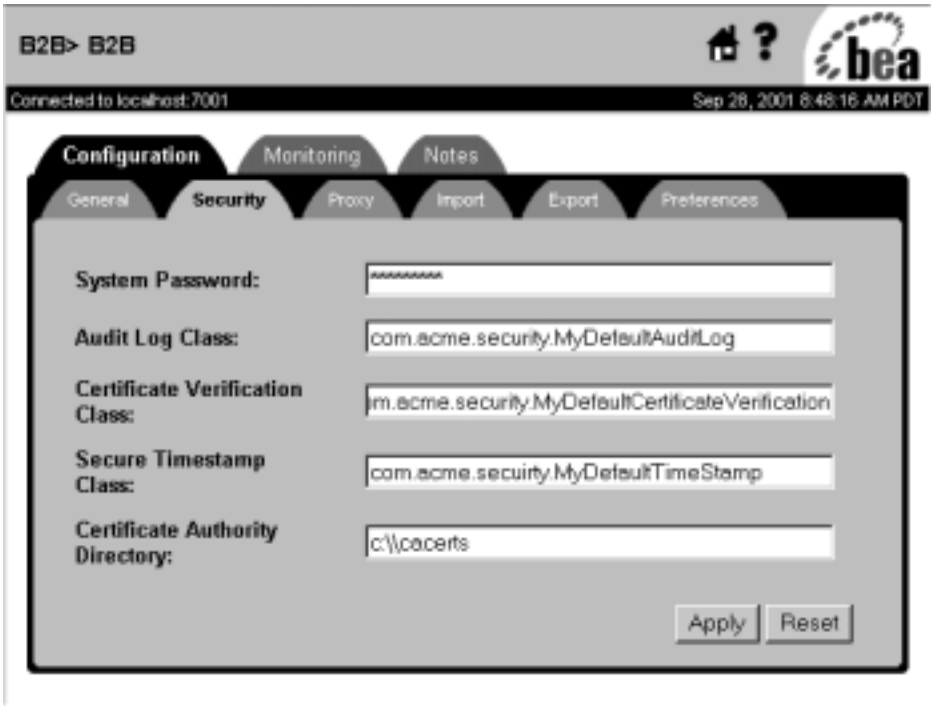
```
        //String strXMLDoc = DocSerializer.docToString
          (resultDoc);
        pw.println("<BR> The Replier Partner has responded
          with the following result... <BR>  ");
        pw.println("<BR> Product: " + product + "");
        pw.println("<BR> Note: " + note + "<P><BR><BR>
          </CENTER>");
        pw.println("<CENTER><IMG SRC=\"Hello4.gif\"
          WIDTH=650 HEIGHT=220
                  BORDER=0 NATURALSIZEFLAG=3></CENTER>");
      }
    }
    else {
      pw.println("<BR> ERROR:  ");
      pw.println("<BR> The Requestor Trading Partner's private
        workflow did not return a result.<P><BR><BR></CENTER>");
      pw.println("<CENTER><IMG SRC=\"Hello1.gif\" WIDTH=650
        HEIGHT=220 BORDER=0 NATURALSIZEFLAG=3></CENTER>");
    }
    pw.println("<P><CENTER><BR> <BR> ");
    pw.println("<P><CENTER><A HREF=\"/HelloPartnerLauncher.html\">
      <b>Click Here to Run Again</b></A></CENTER></P>");
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

# セキュア監査ログのコンフィグレーション

セキュア監査ログをコンフィグレーションするには、以下の手順を実行します。

1. 4-9 ページの「WebLogic Integration B2B エンジンのセキュリティのコンフィグレーション」で説明しているように、B2B Console を起動し、WebLogic Integration B2B コンフィグレーション ページを表示します。
2. [ セキュリティ ] タブを選択します。次の図に示す B2B の [ セキュリティ ] コンフィグレーション ページが表示されます。

**図 5-2 WebLogic Integration B2B の [ セキュリティ ] コンフィグレーション ページ**




3. [ 監査ログ クラス ] フィールドに、セキュア監査ログを実装する Java クラスの完全修飾名を入力します。
4. 新しいコンフィグレーションを有効にするため WebLogic Server を再起動します。

# 否認防止用の SPI の使い方

この節では、以下の否認防止サービス用の SPI について説明します。

- セキュア タイムスタンプ サービス
- セキュア監査ログ サービス

## セキュア タイムスタンプ サービス用の SPI の使い方

WebLogic Integration B2B では、`com.bea.security.TimeStampProvider` インタフェースを実装することで、セキュア タイムスタンプ サービスをカスタマイズできます。ここで説明する SPI を使用してタイムスタンプ サービスを実装する場合、そのサービスが実行時に正しく呼び出されるように、B2B Console でサービスをコンフィグレーションする必要があります。

`com.bea.b2b.security.TimeStampProvider` インタフェースには、タイムスタンプ アプリケーションが実装しなければならない以下のメソッドがあります。

- `String getTimestamp()`

  このメソッドは、協定世界時（UTC）形式で時刻を指定する文字列を返します。

- `long getTimestampInMillis()`

  このメソッドは、ミリ秒単位で UTC 時刻を指定する文字列を返します。

タイムスタンプ インタフェースの実装には、引数を持たないデフォルトのパブリック コンストラクタを含める必要があります。`TimeStampProvider` を実装するクラス内のコンストラクタもメソッドも、例外を送出してはなりません。

# セキュア監査ログ サービス用の SPI の使い方

WebLogic Integration B2B では、`com.bea.security.AuditLogProvider` インタフェースを実装することで、セキュア監査ログ サービスを作成できます。ここで説明する SPI を使用して監査ログ サービスを実装する場合、そのサービスが実行時に正しく呼び出されるように、B2B Console でサービスをコンフィグレーションする必要があります。

`com.bea.b2b.security.AuditLogProvider` インタフェースには、監査ログ アプリケーションが実装しなければならない以下のメソッドがあります。

- `void init()`

  このメソッドは監査ログを初期化します。

- ```
  void log (java.lang.String component,
            java.lang.String type,
            byte[] data,
            java.lang.String principal)
  ```

  このメソッドは、セキュア監査ログにメッセージを書き込むために呼び出されます。このメソッドには、以下のパラメータがあります。

  - `java.lang.String component`

    メッセージをログに書き込むコンポーネントを格納します。

  - `java.lang.String type`

    否認防止メッセージのタイプを指定します。

  - `byte[] data`

    ログに書き込むデータを格納します。

  - `java.lang.String principal`

    このメソッドをログに書き込むトレーディング パートナの名前を格納します。

セキュア監査インタフェースの実装には、引数を持たないデフォルトのパブリック コンストラクタを含める必要があります。`AuditLogProvider` を実装するクラス内のコンストラクタもメソッドも、例外を送出してはなりません。
  ```

# 監査ログ メッセージ

すべてのログ メッセージは、各メッセージ タイプの内容を定義する DTD `log-message.dtd` に対応しています。

すべての監査ログ メッセージは、以下の 3 つの識別子を持ちます。

- Location ー メッセージが格納されている WebLogic Integration 内の場所
- Type ー メッセージのタイプ
- Data ー ログに書き込まれる実際の情報

次の表では、各メッセージ タイプのデータの内容について説明します。すべてのログ メッセージには、WebLogic Integration でコンフィグレーションされているタイムスタンプ プロバイダから取得したタイムスタンプが格納されています。

| メッセージ タイプ | 説明 |
|---|---|
| NRR | 受信側の否認防止性。ビジネス メッセージとアプリケーション データを受け取るトレーディング パートナの名前を格納する。 |
| NRO | 送信側の否認防止性。送信側トレーディング パートナの名前、ビジネス メッセージ、およびアプリケーション データを格納する。 |
| APP | `Audit.log(byte[] data)` メソッドを使用してトレーディング パートナの Java クラスからログに記録される。このメッセージ タイプのデータ フォーマットは、文字列化した XML ドキュメントである。アプリケーションはメッセージをログに記録するので、データの内容はアプリケーション自体が管理する。 |

# 監査ログの DTD

次のコード例は、`log-message.dtd` ファイルを示しています。

```
<!ELEMENT LOG (non-repudiation-origin| non-repudiation-receipt | application)>
<!ATTLIST LOG  time-stamp CDATA  #REQUIRED >
<!ATTLIST LOG location CDATA  #IMPLIED >
```

```
<!ATTLIST LOG Principal CDATA  #IMPLIED >
<!ELEMENT non-repudiation-origin (#PCDATA)>
<!ELEMENT non-repudiation-receipt (#PCDATA)>
<!ELEMENT application (#PCDATA)>
```

# 索引

## ふ



## む



## め



## ゆ



## り