BEA WebLogic Integration™

Data Integration プラグイン ユーザーズ ガイド

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

Data Integration プラグイン ユーザーズ ガイド

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

索引

1 Data Integration プラグイン

このマニュアルでは、Data Integration プラグインの機能と使用方法について説明します。この章で説明するトピックは以下のとおりです。

- XML 変換について
- データ統合とは
- リポジトリの使い方

XML 変換について

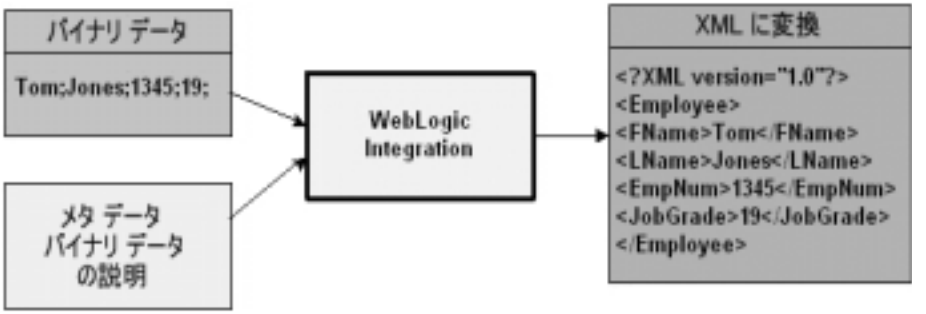
Enterprise Application Integration (EAI) のほとんどの領域で、EAI ソリューションにはデータ変換が関係してきます。XML は急速にアプリケーション間の情報交換に関する標準となりつつあり、異なるアプリケーションを統合する上でかけがえのない手段となっています。しかし、データ変換エンジンのほとんどは、バイナリ データ フォーマットと XML の変換をサポートしていません。Data Integration プラグインは、従来のシステムのバイナリ フォーマットと XML との間のデータ変換をサポートすることにより、アプリケーション間の情報交換を可能にします。

従来のアプリケーションとの間でやり取りされるデータは、情報の発信元である機器固有のバイナリ フォーマットで作成された、プラットフォーム依存情報であることが多くなっています。バイナリ データは自己記述型ではありません。このため従来のアプリケーションからのバイナリ データを使用するアプリケーションが理解できるように、データのフォーマット情報（メタデータ）を各アプリケーションの中に埋め込む必要があります。

XML は、アプリケーション間で情報を交換するための標準になりつつあります。その理由として、XML ではデータの記述がデータ ストリームの中に埋め込まれ、アプリケーション間でデータを簡単に交換できることが挙げられます。XML は解析が容易であり、複雑なデータ構造も表現できます。このため、アプリケーションを結合する場合に各アプリケーションの中にメタデータを埋め込む必要がありません。

バイナリ データと XML データの変換では、標準の XML 解析方法によってデータにアクセスできるように、構造化されたバイナリ データを XML ドキュメントに変換します。変換を行うときに使用するメタデータを作成する必要があります。変換プロセスでは、バイナリ データの各フィールドが、フィールドについて定義されたメタデータに従って XML に変換されます。メタデータには、フィールドの名前、データの型、サイズと、フィールドが省略可能かを指定します。バイナリ データは、このバイナリ データの記述を使用して XML に変換されます。図 1-1 は、XML へのバイナリ データ変換の例を示しています。

図 1-1 XML データ変換 (Tom;Jones;1345;19;)



WebLogic Server プラットフォームで開発されたアプリケーションの多くは、標準のデータ フォーマットとして XML を使用します。WebLogic Server プラットフォーム上の他のアプリケーションから従来のシステムのデータにアクセスする場合、WebLogic Integration を使用することでバイナリ データ フォーマットと XML データ フォーマットを相互に変換できます。最終的な用途として特定のデータを XML 固有言語の XML に変換する必要がある場合は、XML データ マッピング ツールを使用してデータを変換する必要があります。

データ統合とは

異なるエンタープライズ アプリケーションのデータの統合をサポートするために、WebLogic Integration では従来のシステムのバイナリ データを XML に、またはその逆に変換できます。従来のデータが XML フォーマットで使用できるようになれば、XML アプリケーションが直接に使用し、特定の XML 文法に変換

することも直接に使用して WebLogic Integration Studio のワークフローを開始することもできます。WebLogic Integration では、次の 3 つのデータ統合ツールを使用して、非 XML と XML 間またはその逆方向の変換をサポートしています。

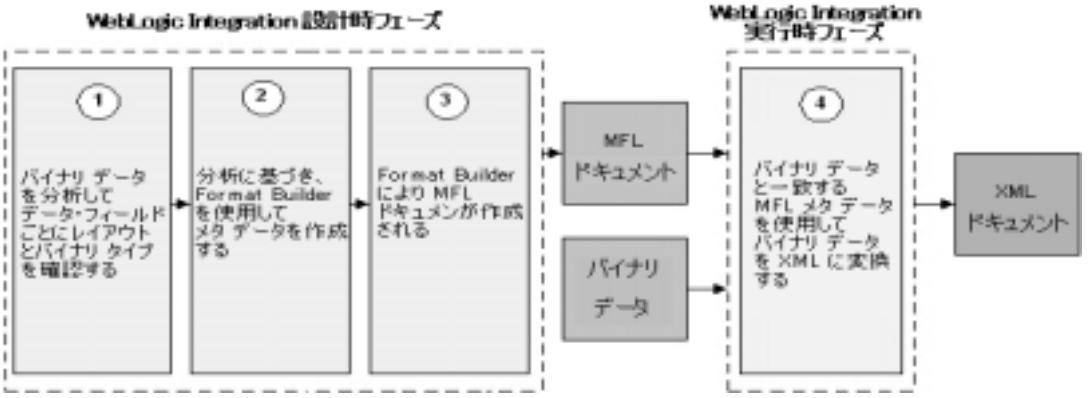
- 設計時 Data Integration コンポーネント
- 実行時 Data Integration コンポーネント
- BPM (Business Process Management) 機能のプラグイン

変換は 2 段階で実行されます。まず、設計時ツールである Format Builder を使用してバイナリ データの記述を作成します。まず、Format Builder で作成するメタデータにバイナリ データのレコード レイアウトが正確に反映されるようにバイナリ データを分析します。

次に、Format Builder でメタデータ（入力データの記述）を作成し、このメタデータをメッセージ フォーマット言語（MFL）ドキュメントとして保存します。WebLogic Integration には COBOL コピーブックなどバイナリ メタデータの共通ソースからメッセージ フォーマット定義を自動的に作成する *インポータ*が用意されています。

次に、WebLogic Integration の実行時コンポーネントを使用してバイナリ データのインスタンスを XML に変換できます。図 1-2 は非 XML から XML へのデータ変換のイベント フローを示しています。BPM 機能へのプラグインにより、変換設定タスクは簡単になっています。

図 1-2 Data Integration を使用しての非 XML から XML への変換のイベント フロー



設計時 Data Integration コンポーネント

WebLogic Integration の設計時データ統合コンポーネントは、Format Builder と呼ばれる Java アプリケーションです。Format Builder はバイナリ データ レコードの記述を作成するときに使用します。Format Builder では、XML データとバイナリ データを相互に変換できるようにバイナリ データのレイアウトと階層を記述できます。

Format Builder で作成した記述は、メッセージ フォーマット言語（MFL）と呼ばれる XML 文法で保存されます。MFL ドキュメントには Data Integration の実行時に使用するメタデータが含まれています。BPM 機能への Data Integration プラグインでもこのメタデータを使用して、バイナリ データ レコードのインスタンスを XML ドキュメントのインスタンス（またはその逆）への変換を行います。

さらに、Format Builder は、変換によって作成される XML ドキュメントを記述する DTD または XML スキーマ ドキュメントも作成します。

実行時 Data Integration コンポーネント

WebLogic Integration の実行時 Data Integration のコンポーネントは Java クラスであり、このクラスには、バイナリ フォーマットと XML フォーマットの間でデータを変換するためのさまざまなメソッドが用意されています。この Java クラスの使用方法にはいくつか種類があります。具体的には、以下のとおりです。

- WebLogic Server を使用して EJB にデプロイ
- Studio で、ワークフローからビジネス オペレーションとして呼び出される
- Java アプリケーションへ統合

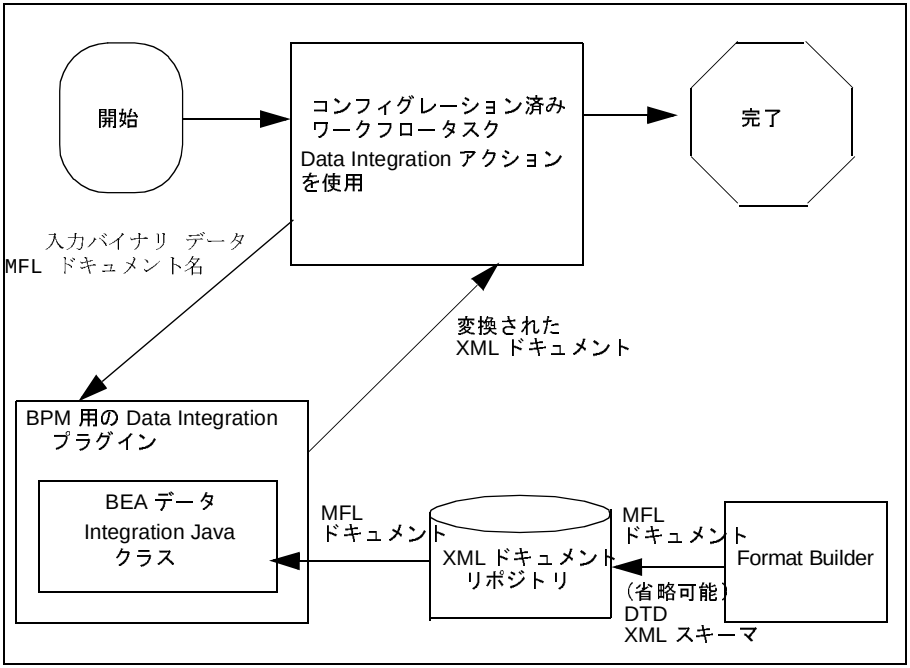
BPM (Business Process Management) 機能のプラグイン

Business Process Management (BPM) の Data Integration プラグインは、従来のシステムのバイナリ データから XML へのデータ変換をサポートすることにより、アプリケーション間の情報交換を可能にします。Data Integration プラグインは、XML からバイナリへの変換と、バイナリから XML への変換にアクセスできる BPM アクションを提供します。

このようなデータ変換機能のほかに、Data Integration プラグインには以下の機能があります。

- バイナリ フォーマットでのイベント データ処理
- パフォーマンスの向上を目的とした MFL ドキュメントのインメモリ キャッシング、変換オブジェクト プール機能
- バイナリ データの編集および表示に使用する `BinaryData` 変数型
- WebLogic Server のクラスタ環境内での実行

次の図は、Data Integration と BPM 機能の間の関係を示しています。



リポジトリの使い方

リポジトリは、ドキュメントをまとめて格納しておくための機能で、次の 4 種類のドキュメントがサポートされます。

- MFL − メッセージフォーマット言語ドキュメント
- DTD − XML 文書型定義ドキュメント
- XSD − XML スキーマ ドキュメント
- XSLT − XSLT スタイルシート

リポジトリではこれらの各ドキュメント型にアクセスでき、Data Integration、Business Process Management、B2B Integration の間でドキュメントを共有できます。リポジトリには、以前に構築された MFL、DTD、XSD、および XSLT ドキュメントをリポジトリに簡単に移行できる バッチ インポート ユーティリティも用意されています。

2 Data Integration プラグインの使用

この章の内容は以下のとおりです。

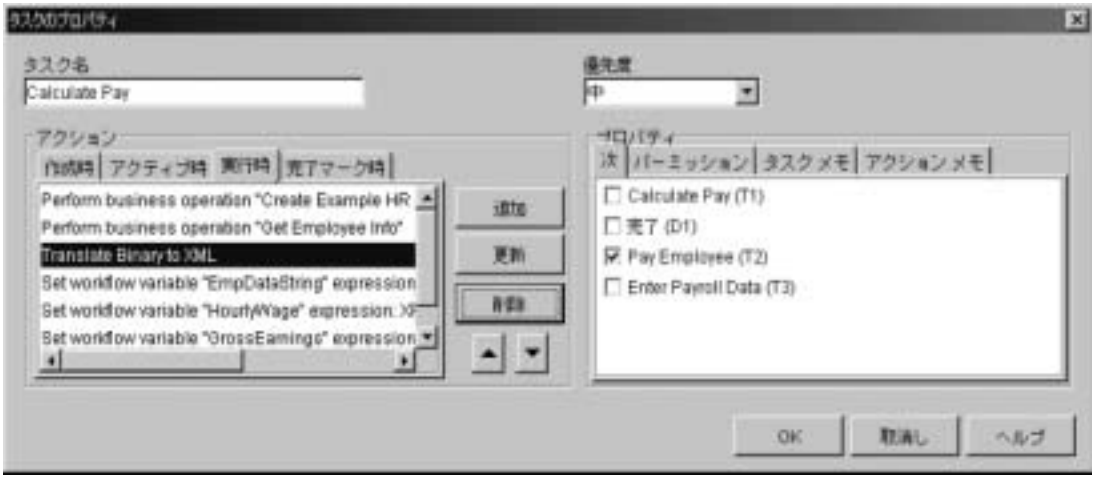
- Data Integration プラグインを使用するデータ変換
- イベント データの処理
- データ変換パフォーマンスの向上
- カスタム データ型と Data Integration プラグイン
- WebLogic Server クラスタ化 のサポート

Data Integration プラグインを使用するデータ変換

Data Integration プラグイン（Data Integration plug-in）は Business Process Management (BPM) 機能の実行、非 XML ドキュメントから XML ドキュメントあるいはその逆方向への変換機能を提供します。変換を実行するには、次の手順に従います。

1. WebLogic Integration Studio を開始する。Studio の開始及びロギングについては、『*WebLogic Integration Studio ユーザーズ ガイド*』の「Studio インタフェースの使用法」を参照してください。
2. 目的のテンプレート定義を開き、タスクをダブルクリックします。[タスクのプロパティ] ダイアログ ボックス（図 2-1）が表示されます。

図 2-1 [タスクのプロパティ] ダイアログボックス



3. 選択したタスクにデータ変換アクションが含まれている場合は、これをリストからもう一度選択して [更新] をクリックして手順 4 に進みます。含まれていない場合、[追加] をクリックして新しいアクションを追加します。[アクションを追加] ダイアログ ボックス（図 2-2）が表示されます。

図 2-2 [アクションを追加] ダイアログ ボックス

4. [統合アクション] ノードを拡張するとアクションのリストが表示されます。リストの [データ統合] 項目を拡張します。変換アクションのリストが表示されます。「XML からバイナリへの変換」および「バイナリから XML への変換」を参照。実行する変換タイプを選択します。

XML からバイナリへの変換

XML からバイナリへの変換を実行する手順は以下のとおりです。

1. [アクションを追加] ダイアログボックス（図 2-2）で、[Translate XML to Binary] を選択する。[XML to Binary] ダイアログ ボックス（図 2-3）が表示されます。

図 2-3 [Translate XML to Binary] ダイアログボックス

2. 次の表に従って、フィールドにデータを入力します。

表 2-1 [Translate XML to Binary] ダイアログボックス

ダイアログボックス領域	フィールド	説明
メッセージフォーマットパラメータ	[名前]	メッセージ フォーマットの名前。テキスト フィールドに名前を直接入力するか、[参照] をクリックしてリポジトリ ドキュメントのリストを呼び出し、その中から選択します。
	[説明]	メッセージ フォーマットの説明を表示する。 **注意：** テキストの表示にのみ使用されるフィールドです。このフィールドでテキストを編集することはできません。
	[メモ]	メッセージ フォーマットに添付されたノートを表示する。 **注意：** テキストの表示にのみ使用されるフィールドです。このフィールドでテキストを編集することはできません。
メッセージフォーマットアクション ボタン	[参照]	リポジトリ内の MFL ドキュメントを参照する。
	[表示]	Message Format 領域に項目が表示される。変換対象として正しいドキュメント タイプが選択されているかどうか確認できる。
	[デバッグ]	メッセージのデバッグを可能もしくは不可能にします。このオプションを選択すると、変換アクションが WebLogic Server ログ ファイルに書き込まれる。
変数パラメータ	[入力 XML 変数]	XML ワークフロー変数を表示する。変換処理で使用する変数を選択するか、以下の手順に従って新しい変数を作成する。 1. 変数の新しい名前を入力し、[OK] をクリックします。確認メッセージ ボックスが表示される。 2. [Yes] をクリックする。新しい変数が作成される。
	[結果の割り当て先]	Binary Data ワークフロー変数が表示される。変換された情報を格納するための変数を選択するか、以下の手順に従って新しい変数を作成する。 1. 変数の新しい名前を入力し、[OK] をクリックします。 2. [Yes] をクリックする。新しい変数が作成される。

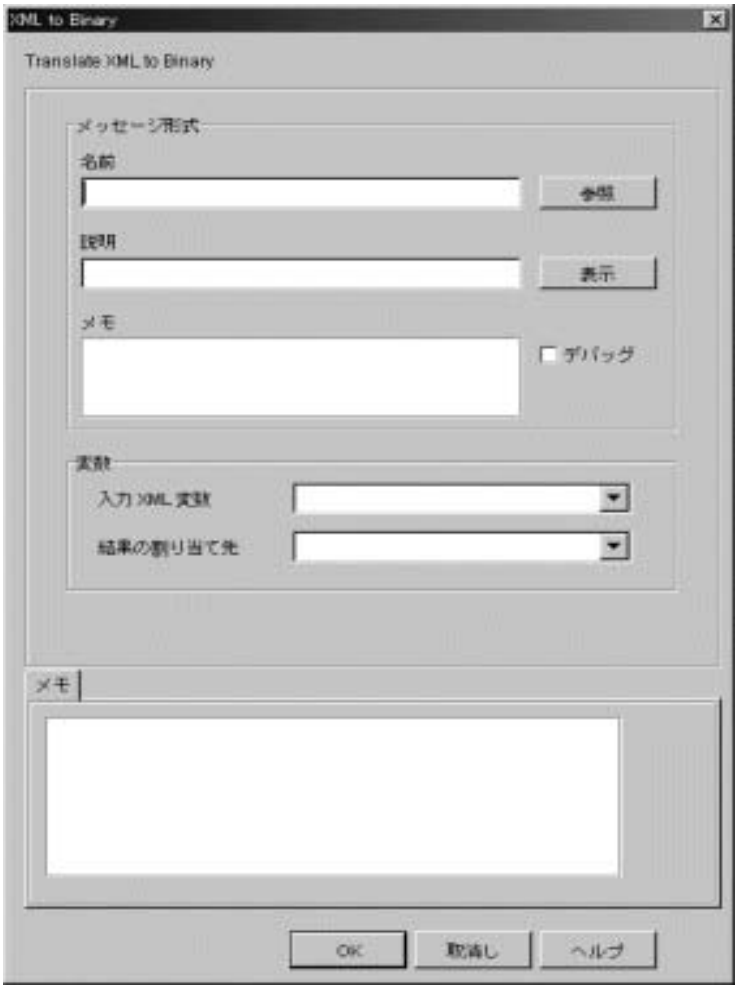
3. 変換情報をワークフローに保存するには [OK] をクリックします。

バイナリから XML への変換

バイナリから XML への変換を実行する手順は以下のとおりです。

1. [アクションを追加] ダイアログボックス（図 2-2）で、[アクション｜データ統合｜ Translate Binary to XML] を選択する。[XML to Binary] ダイアログボックス（図 2-4）が表示されます。

図 2-4 [Translate Binary to XML] ダイアログボックス



2. 次の表に従って、フィールドにデータを入力します。

表 2-2 [Binary to XML] ダイアログ ボックス フィールド

ダイアログ ボックス領域	フィールド	説明
メッセージ フォーマット パラメータ	[名前]	メッセージ フォーマットの名前。テキスト フィールドに名前を直接入力するか、[Browes] をクリックしてリポジトリ ドキュメントのリストを呼び出し、その中から選択します。
	[説明]	メッセージ フォーマットの説明を表示する。 **注意：** テキストの表示にのみ使用されるフィールドです。このフィールドでテキストを編集することはできません。
	[メモ]	メッセージ フォーマットに添付されたノートを表示する。 **注意：** テキストの表示にのみ使用されるフィールドです。このフィールドでテキストを編集することはできません。
メッセージ フォーマット アクション ボタン	[参照]	リポジトリ内の MFL ドキュメントを参照する。
	[表示]	Message Format 領域に項目が表示される。変換対象として正しいドキュメント タイプが選択されているかどうか確認できる。
	[デバッグ]	メッセージのデバッグを可能もしくは不可能にします。このオプションを選択すると、変換アクションが WebLogic Server ログ ファイルに書き込まれる。

表 2-2 [Binary to XML] ダイアログ ボックス フィールド

ダイアログ ボックス領域	フィールド	説明
変数パラメータ	[入力バイナリ変数]	バイナリ ワークフロー変数を表示する。変換処理で使用する変数を選択するか、以下の手順に従って新しい変数を作成する。 1. 変数の新しい名前を入力し、[OK] をクリックします。確認メッセージ ボックスが表示される。 2. [Yes] をクリックする。新しい変数が作成される。
	[結果の割り当て先]	XML データ ワークフロー変数が表示される。変換後の情報を格納する変数を選択するか、以下の手順に従って新しい変数を作成する。 1. 変数の新しい名前を入力し、[OK] をクリックします。 2. [Yes] をクリックする。新しい変数が作成される。

3. 変換情報をワークフローに保存するには [OK] をクリックします。

イベント データの処理

Data Integration プラグイン（Data Integration plug-in）には、バイナリ データを XML に変換する、またはイベント処理のフロント エンドで事前に処理することで、バイナリ データによってワークフローを開始させる機能が用意されています。この機能は *イベント ハンドラ* と呼ばれます。イベント ハンドラは JMS メッセージがトピックにパブリッシュされると実行されます。

Data Integration プラグイン（Data Integration plug-in）で事前処理されるメッセージには、3 つの JMS プロパティが必要です。

- `WLPIContentType: "binary/x-application/wlxt"`
- `WLPIPlugin: "com.bea.wlxt.WLXTPlugin"`
- `WLPIEventDescriptor: MFL_document_name`

最初の 2 つの JMS メッセージ プロパティは、イベント ハンドラにアドレッシングされるすべてのメッセージ用の定数です。3 番目のプロパティには、メッセージ内のバイナリ データを記述する MFL ドキュメントの名前が含まれます。

注意： WLPI EventDescriptor の中で参照される MFL ドキュメントは、リポジトリに格納されている必要があります。

コード リスト 2-1 は、イベント ハンドラに処理させるメッセージを構築するためのコード例です。

コード リスト 2-1　イベント ハンドラのコード例

```
byte[] bindata = ... the binary data ...
pub = sess.createPublisher(topic);
BytesMessage msg = sess.createBytesMessage();
msg.writeBytes(bindata);
msg.setStringProperty("WLPIPlugin", "com.bea.wlxt.WLXTPlugin");
msg.setStringProperty("WLPIContentType",
 "binary/x-application/wlxt");
msg.setStringProperty("WLPIEventDescriptor", “mymfldoc”);
pub.publish(msg);
```

サーブレット サンプル アプリケーションでは、このコードで構築されたメッセージがイベント ハンドラで処理される様子を見ることができます。サーブレット サンプルの実行方法については、「サーブレット サンプルの実行」ページ 3-1 を参照してください。

データ変換パフォーマンスの向上

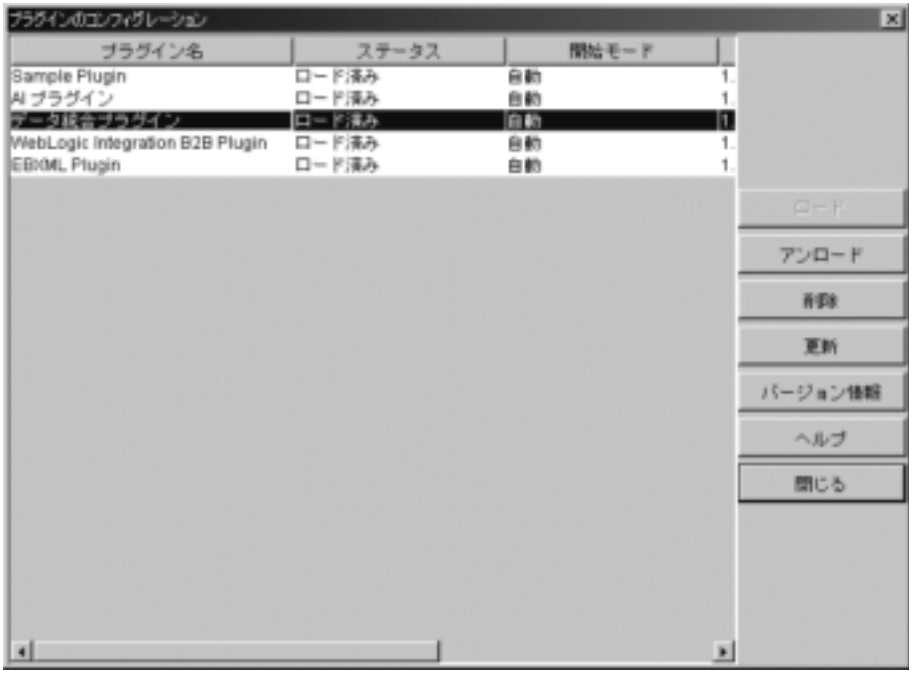
Data Integration プラグイン（Data Integration plug-in）には、MFL ドキュメント インメモリ キャッシュの管理と監視、イベント ハンドラのデバッグを有効または無効にするためのコンフィグレーション パネルが用意されています。このパネルを使用することで、インメモリ キャッシュと変換オブジェクト プールを調整して、データ変換のパフォーマンスを向上させることができます。

注意： MFL ドキュメントに行う更新を有効にするためには、MFL ドキュメント インメモリ キャッシュを消去する必要があります。

コンフィグレーション パネルにアクセスする手順は以下のとおりです。Business Process Management 固有のアクションの詳細については、Business Process Management を参照してください。

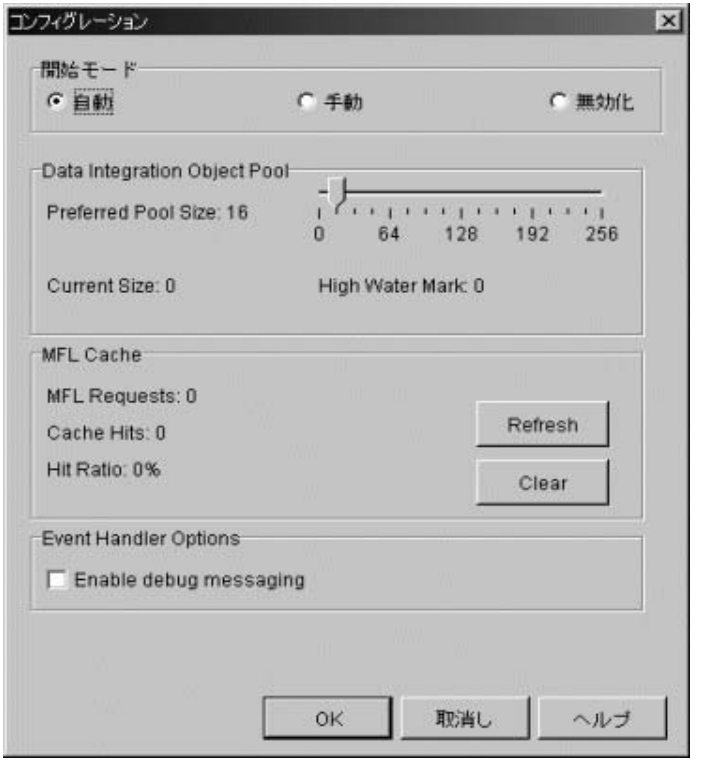
1. WebLogic Integration Studio を開始する。Studio の開始及びロギングについては、「Studio インタフェースの使用法」を参照してください。
2. [コンフィグレーション | プラグイン] を選択します。[プラグインのコンフィグレーション] ダイアログ ボックスが表示されます（図 2-5）。

図 2-5 [プラグインのコンフィグレーション] ダイアログボックス



3. [データ統合プラグイン] を選択し、[更新] をクリックします。Data Integration プラグイン（Data Integration plug-in）の [コンフィグレーション] ダイアログボックスが開きます（図 2-6）。

図 2-6 Data Integration プラグインの [コンフィグレーション] ダイアログボックス



4. 変換パフォーマンスを監視、向上させるには、次のテーブルで説明するフィールドにデータを入力します。

ダイアログボックス領域	フィールド	説明
[開始モード]	[自動]	Studio の開始時に自動的に Data Integration プラグイン（Data Integration plug-in）を立ち上げます。
	[手動]	Studio から Data Integration プラグイン（Data Integration plug-in）を使用できるようにします。
	[無効化]	Studio から Data Integration プラグイン（Data Integration plug-in）を使えないようにします。

ダイアログ ボックス領域	フィールド	説明
[Data Integration Object Pool]	[Preferred Pool Size]	プール内の永続的オブジェクトの最大数を定義する。スライダを使用して、プール サイズを該当する値に設定する。 **注意：** ユーザが設定したプリファレンス プール サイズを超過すると、変換エンジンによって一時的なプール オブジェクトが作成されます。それらのオブジェクトは、プールに返されるときに削除されます。
	[Current Size]	現在プール内にあるオブジェクト数を表示する。
	[High Water Mark]	サーバ起動時以降の、プール内の最大オブジェクト数を表示する。
MFL キャッシュ	[MFL Requests]	MFL ドキュメントの変換リクエストの合計数を表示する。
	[Cache Hits]	必要な MFL ドキュメントがすでにキャッシュ内に存在している場合の要求数が表示される。
	[Hit Ratio]	データベースからではなく、キャッシュから MFL ドキュメントを読み込んで対応されたリクエストの割合を表示する。
MFL キャッシュ アクション ボタン	[Refresh]	サーバにリクエストを送信して、MFL キャッシュの統計を更新する。
	[Clear]	MFL ドキュメント キャッシュを消去する。後続の変換要求のことを考え、MFL ドキュメントはリポジトリからロードします。
Event Handler オプション	[Enable Debug Messaging]	イベント ハンドラに対して、デバッグ メッセージ機能を有効または無効に設定する。有効にすると、変換時にデバッグ メッセージが WebLogic Server ログ ファイルに書き込まれる。

Data Integration プラグイン（Data Integration plug-in）により、標準の BPM 機能の編集や表示機能が拡張されます。これらの機能は、バイナリ データを表示、編集するための Format Tester の Hex Editor コンポーネントによって提供されます。

変数型と Data Integration プラグイン

Data Integration プラグイン（Data Integration plug-in）には、バイナリ データを編集、表示するときに使用できる `Binary Data` 変数型が用意されています。`Binary Data` 変数はバイナリ データの論理セットのコンテナとして機能し、追加の表示機能があります。変数は、Data Integration プラグイン（Data Integration plug-in）に用意されているアクションを呼び出してバイナリ データをやりとりするプログラムによって使用されます。また、Workflow Instance Monitor がバイナリ変数の内容を表示、編集するときにも使用されます。

カスタム データ型と Data Integration プラグイン

WebLogic Integration には、ユーザに独自に必要なデータ型に適したカスタム データ型を作成できるユーザ定義データ型機能が用意されています。ユーザ定義データ型機能を使用すると、Data Integration の実行時エンジンにカスタムのデータ型をプラグインできます。いったんプラグインされたユーザ定義のデータ型は、特徴や機能の面で組み込みデータ型と区別できません。

ユーザ定義データ型の設定

Data Integration プラグイン（Data Integration plug-in）で使用されるユーザ定義データ型は、`CLASS` ドキュメントとして WebLogic Integration リポジトリに格納されます。実行時に、Data Integration プラグイン（Data Integration plug-in）は必要に応じてリポジトリからユーザ定義データ型のクラスを読み込みます。さらに、Data Integration プラグイン（Data Integration plug-in）は、アクティブ テンプレートをサポートするのに必要な MFL およびクラス ファイルをエクスポートし、これによって別の Business Process Management インスタンス上でテンプ

レートをそのままインポートできます。クラスドキュメントは、次のどちらかの方法を使用してリポジトリに格納できます。以下の節では、それらのメカニズムについて説明します。

- Format Builder の使い方
- Repository Import ユーティリティの使い方

Format Builder の使い方

Format Builder を使用してユーザ定義データ型をリポジトリに公開するには、次の手順を実行します。

1. [スタート | プログラム | BEA WebLogic Platform 7.0 | WebLogic Integration 7.0 | Format Builder] を選択して、Format Builder を起動する。Format Builder メイン ウィンドウが表示されます。
2. [Repository | Log In] を選択します。[WebLogic Integration リポジトリへのログイン] ウィンドウが開きます。

3. [ユーザ名] フィールドに、サーバ ユーザに対して指定されたユーザ ID を入力します。
4. [パスワード] フィールドに、サーバ ユーザに対して指定されたパスワードを入力します。

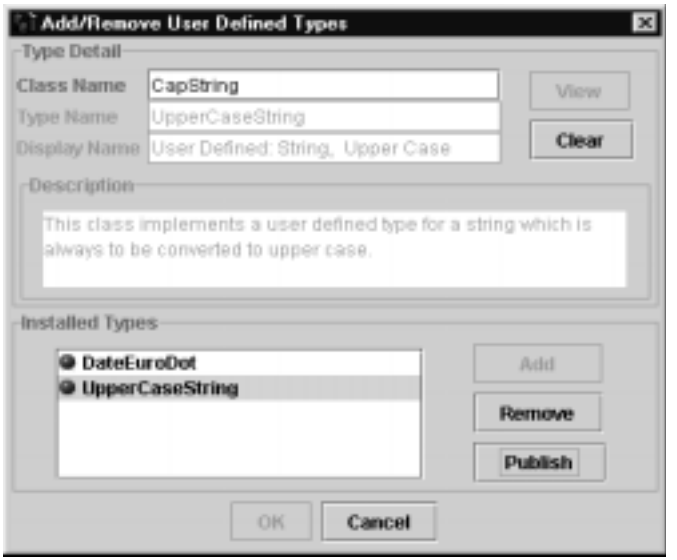
注意： デフォルト ユーザとパスワードについては、『*WebLogic Integration の起動、停止およびカスタマイズ*』の「開始する前に」の「WebLogic Integration ユーザとパスワード」を参照してください。

5. サーバの名前と該当するポートの番号を [Server[:port]] フィールドに入力します。

   注意： [WebLogic Integration リポジトリへのログイン] ウィンドウでは、ログインは 3 回までしか行うことができません。3 回失敗すると、ログイン失敗のメッセージが表示されます。ログインに 3 回失敗した場合は、[Repository ｜ Log In] を選択して、ログインの手順を繰り返します。

6. [接続] をクリックします。ログインに成功すると、[ログイン] ウィンドウが閉じ、Format Builder のタイトル バーに [WebLogic Integration リポジトリへのログイン] ウィンドウで入力したサーバ名とポート番号が表示されます。アクティブなリポジトリ項目のメニューが表示されます。アクセスしたい項目を選択します。

7. [Tools ｜ User Defined Types] を選択します。[Add/Remove User Defined Types] ダイアログ ボックスが表示されます。



リポジトリとの接続が確立されていると、[Add/Remove User Defined Types] ダイアログボックスには登録済みの各ユーザ定義データ型のステータスが表

示され、リポジトリに公開できます。ユーザ定義型のリポジトリのステータスは、ダイアログ ボックスの [Installed Types] 領域の各エントリの前に、ボール方のアイコンでで示されます。

各ユーザ定義型の名前の前にあるアイコンの色は、そのステータスを表わします。

- [緑] ーユーザ定義データ型は、リポジトリに公開されている。
- [黄色] ーユーザ定義データ型はリポジトリに公開されているが、クラスのローカル バージョンがリポジトリのバージョンと異なる。
- [赤] ーユーザ定義データ型は、リポジトリ内に存在しない。

8. [Installed Types] のリストから公開するクラスをクラスを選択して、[Publish] をクリックします。選択したエントリのアイコンの色が緑になるはずです。これは、そのクラスがリポジトリの中に正常に格納されたことを意味します。

Repository Import ユーティリティの使い方

リポジトリ インポート ユーティリティを使用して、ユーザ定義データ型を含めて Java クラス ファイルをインポートする手順は以下のとおりです。

1. [CLASSPATH] 内に `wlxt-repository.properties` ファイルを作成します。このファイルの内容は次のようになります。

   `wlxt.repository.url=`*`server_url`*

   たとえば、

   `wlxt.repository.url=t3://localhost:7001`

2. 次のコマンドを入力して、クラス ファイル名を Import コマンド ラインに渡します。

   `java com.bea.wlxt.repository.Import` *`filename`*

   たとえば、次のコマンドは現在のディレクトリ内のすべてのクラス ファイルをインポートします。

   `java com.bea.wlxt.repository.Import *.class`

注意： Repository Import ユーティリティでは、ユーザ定義データ型だけでなく任意の Java クラス ファイルをリポジトリにインポートできます。この機能は、ユーザ定義データ型が、`com.bea.wlxt.bintype.Bintype` クラスの拡張ではない追加のクラス ファイルに依存する場合に便利です。

WebLogic Server クラスタ化 のサポート

Data Integration プラグイン（Data Integration plug-in）は、WebLogic Server クラスタ化環境でも正しく機能します。クラスタ化環境では、プラグイン管理者は常にクラスタの 1 つのノードだけに接続されます。管理者が発行するコマンドは、クラスタ内の別のノードに伝播します。

クラスタ内のサービス間の通信は、JMS トピックを使用して処理されます。JMS トピックは、WebLogic Integration 環境内のクラスタ内の異なるノード上の通信に使用されます。

クラスタ化のための Data Integration プラグインの設定

クラスタ化機能を利用する場合には、Data Integration プラグイン（Data Integration plug-in）を以下のように設定する必要があります。

1. クラスタ内のサーバの 1 つで JMS トピックを作成します。このトピックの JNDI 名は次のようになります。

   `com.bea.wlxt.cluster.BroadcastTopic`

   注意： JMS トピック作成の詳細については、WebLogic Server マニュアルを参照してください。

2. テキスト エディタで、`config.xml` ファイルを開く。ファイルは *`SAMPLES_HOME`*`\integration\config\samples\` ディレクトリにあります。*`SAMPLES_HOME`* は WebLogic Platform サンプルのインストール ディレクトリです。

   注意： `config` ディレクトリには、作成した各ドメインごとに個別のサブディレクトリが含まれます。これらの各サブディレクトリには、それ

ぞれの config.xml ファイルがあります。正しいファイルを開くようにしてください。

3. Business Process Management の <Application> セクションを見つけて、このセクション内の任意の場所に以下を追加します。

```
<EJBComponent Name="wlxt-cluster"
 DeploymentOrder="99"
 Targets="[server_name]"
 URI="wlxtmb.jar"
/>
```

4. config.xml ファイルを保存します。

注意： config.xml ファイルへの変更を有効にするには、サーバを再起動する必要があります。

3 WebLogic Integration サンプル アプリケーションの実行

Data Integration ソフトウェアには、WebLogic Integration 内での Business Process Management (BPM) の統合を理解するために設計された 2 つのサンプル アプリケーションが含まれています。この節では、この 2 つのサンプルに基づきアプリケーションの実行について段階的に説明します。説明するトピックは以下のとおりです。

- 前提条件
- サーブレット サンプルの実行
- EJB サンプルの実行

前提条件

この節の説明は、WebLogic Integration について、特に Data Integration と WebLogic Integration プロセス エンジンについて理解されていることを前提として書かれています。また、サンプル アプリケーションを実行するためには、WebLogic Integration を正しくインストールし、サンプル ワークフローを実行しておく必要があります。

サーブレット サンプルの実行

このサンプル アプリケーションでは、サーブレットをインストールする Web アーカイブ ファイル (`WLPI_sample.war`) を実装します。このサーブレットは、バイナリ データから XML への変換の要求を受け付けます。アクセスはブラウザを通じて行います。またサーブレットは、生成された XML データを表示するこ

とによって応答します。さらに、アプリケーションはデータを XML 形式またはバイナリ形式のいずれかで WebLogic Integration イベント トピックにポストされます。このデータを使用してワークフローを開始できます。

サーブレット サンプルの内容

サーブレット サンプル アプリケーションは WebLogic Integration インストールの *SAMPLES_HOME*\integration\samples\di\wlpi ディレクトリに格納されます。*SAMPLES_HOME* は WebLogic Platform インストールにおけるサンプル ディレクトリです。次の表では、サーブレット サンプル アプリケーションに含まれるファイルについて説明しています。

表 3-1 サーブレット サンプル アプリケーション ファイル

ディレクトリ	ファイル	説明
\wlpi\source	WLPI_sample.java	HTML ファイルを画面に表示する、およびバイナリ データを XML に変換するためのサーブレットのソース コード。この XML は、オプションとして JMS トピック上に置くこともできる。
\wlpi	SampleData.mfl	サンプル ワークフローの開始に使用するサンプル バイナリ データ ファイルのメッセージ フォーマット言語による記述。
\wlpi	SampleData.data	サンプル ワークフローの開始時にインプットとして使用されたサンプル データ ファイル。
\wlpi	DI_ServletSample.jar	サンプルで使用される、エクスポートされるワークフロー。ワークフローはサンプルを設定すると自動的にインポートされる。説明については 3-4 ページの「ステップ 1. Sample Application Launcher の起動」を参照。
\wlpi	Makefile	サンプル ソースを .jar ファイルに構築するためのメイク ファイル。
\wlpi	build.cmd	ソースから .jar ファイルを構築する。

表 3-1 サーブレット サンプル アプリケーション ファイル

ディレクトリ	ファイル	説明
\wlpi\images	bealogo.jpg	サンプル サーブレットによって HTML ページ上に表示される BEA ロゴ イメージ。
\wlpi\WEB-INF	hello.html	ユーザからの入力データを取得するサンプル サーブレットによって使用される HTML ページ。
\wlpi\WEB-INF	web.xml	サンプル サーブレットのデプロイメント情報を定義する J2EE コンフィグレーション ファイル。
\wlpi\WEB-INF	weblogic.xml	サンプル サーブレットの WebLogic の固有情報を定義する BEA コンフィグレーション ファイル。
wlpi\WEB-INF\lib	cos.jar	サンプル コードの実行で使用されるユーティリティ ライブラリ。
wlpi\WEB-INF\lib	HtmlScreen.jar	サンプル コードの実行で使用されるユーティリティ ライブラリ。
WebLogic Platform ホーム ディレクトリ直下		
samples\integration\config\samples\applications	WLPI_sample.war	すべての実行可能なサンプル コードとコンフィグレーション ファイルを含む Web Archive ファイル。このファイルは、インストール時に WebLogic Integration アプリケーション ディレクトリに自動的にデプロイされる。

サーブレット サンプルの実行方法

サーブレット サンプルを実行するには、この節で説明する手順を実行します。WebLogic Server および BPM 機能固有のタスクの説明については、edocs.beasys.co.jp/e-docs/index.html の BEA ドキュメントを参照してください。

ステップ 1. Sample Application Launcher の起動

初めて使用する場合

Sample Application Launcher を起動するには、プラットフォームに合わせて適切な手順を実行します。

- Windows

1. [スタート | BEA WebLogic Platform 7.0 | WebLogic Integration 7.0 | Integration Samples | Start Server] を選択して [Launch Example] を選択します。すべてのサンプルが設定されていれば、サンプル起動ページが表示されます。すべてのサンプルが設定されるのに数分かかります。図 3-1 は Sample Application Launcher です。

図 3-1 Sample Application Launcher

2. [Data Integration Servlet] をサンプル アプリケーション リストから選択します。図 3-2 は Data Integration Servlet サンプル ページです。

図 3-2 Data Integration Servlet サンプル ページ

- UNIX

1. PATH 環境変数に、Netscape 実行ファイル（netscape）が格納されたディレクトリが含まれていることを確認します。含まれていない場合は、サンプル起動ページを表示できません。
2. WebLogic Integraion のホーム ディレクトリ（WebLogic Integraion のインストール先ディレクトリ）に移動します。たとえば、

   ```
   cd /home/me/bea/weblogic700/integration
   ```

3. setenv スクリプトを実行して、最上位の WebLogic Integraion 環境変数を設定します。

   ```
   . setenv.sh
   ```

4. RunSamples スクリプトを実行します。たとえば、

   ```
   cd /home/me/bea/weblogic700/samples/integration/samples/bin
   RunSamples
   ```

5. RunSamples スクリプトのコンフィグレーション セクションが実行済みであることが検知されると、次のプロンプトが表示されます。

   ```
   The WebLogic Integration repository has already been created
   and populated, possibly from a previous run of this
   RunSamples script. Do you want to destroy all the current
   ```

```
data in the repository and create and populate the
WebLogic Integration repository, again?
Y for Yes, N for No
```

この質問に N と入力すると、リポジトリの作成および格納を行う手順が省略され、WebLogic Server のサンプル インスタンスを起動する手順のみが実行されます。

この質問に Y と入力すると、リポジトリの作成および格納が改めて行われ、その後で WebLogic Server のサンプル インスタンスを起動する手順が実行されます。Y と入力した場合、その時点でリポジトリに格納されている全データが破棄され、リポジトリにサンプル データが再ロードされます。現在のサンプル データが変更または削除され、新規または未変更のサンプル データをリポジトリに格納する場合にのみ、Y を入力してください。

以上により、WebLogic Server のインスタンスがバックグラウンド プロセスとして起動され、サンプル起動ページが表示されます。

以前にサンプルを設定している場合

サーバを起動して Sample Launcher を表示するには、プラットフォームに合わせて適切な手順を実行します。

- Windows

1. [スタート | BEA WebLogic Platform 7.0 | WebLogic Integration 7.0 | Integration Examples | Start Server] を選択します。
2. [スタート | BEA WebLogic Platform 7.0 | WebLogic Integration 7.0 | Integration Examples | Launcher Examples] を選択します。
3. [Data Integration Servlet] を選択し [Data Integration Servlet サンプル] ページを開きます。

- UNIX

1. サンプル ドメインの bin ディレクトリに移動します。たとえば、製品をデフォルトのディレクトリにインストールした場合、次のように入力します。

   ```
   cd SAMPLES_HOME/integration/samples/bin
   ```

2. 次のように入力してサーバを起動します。

   ```
   . ./startServer
   ```

3. 次のメッセージが表示された場合は、startServer スクリプトは正しく終了しています。

```
StartServer execution successful
```

4. Web ブラウザを起動して次の URL を入力します。

   ```
   http://localhost:7001/index.html
   ```

   Samples Launcher Web ページが表示されます。

ステップ 2. メール セッションの設定

メール ホストをすでに設定している場合は、このステップを省いても構いません。コンフィグレーションを検証したい場合は、実行するとよいでしょう。

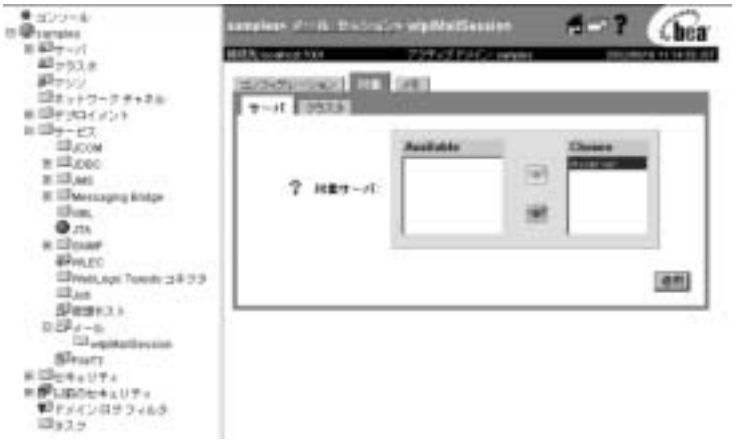
1. Samples Launcher の Administration Console から [WebLogic Server] をクリックします。WebLogic Server Administration Console が表示されます。
2. ナビゲーション ツリーから [サービス | メール | wlpiMailSession] を選択します。
3. 正しい情報を入力してメール ホストを設定します。この場合、必ず `mail.host=mailserver` とします。[メール セッション] 画面の [コンフィグレーション] タブの例を次の図に示します。

図 3-3 [メール セッション] 画面の [コンフィグレーション] タブ

4. [対象] タブを選択します。

5. 次の図に示すように、メール サーバ名を [Available] 列から [Chosen] 列に移動します。

図 3-4 [メール セッション] 画面の [サーバ] タブ



6. [適用] をクリックします。

ステップ 3. サンプル XML データの更新とメッセージの送信

1. テキスト エディタで *SAMPLES_HOME*\integration\samples\di\wlpi\SampleData.data ファイルを開きます。テキスト *user@nowhere.com* を、ワークフローが電子メール メッセージの配信に使用する有効な電子メール アドレスで置き換えます。

2. [Sample Application Launcher] の [Input File] フィールドで次のデータ ファイルに移動します。

   SAMPLES_HOME\integration\samples\di\wlpi\SampleData.data

   指定したファイルはローカル システム内に無ければなりませんが、サーバを実行しているものでなくて構いません。ファイルがローカル システムに無い場合は、エラー メッセージが表示されます。

3. [Submit] をクリックします。データ ファイルで指定したアドレスに、短い電子メール メッセージが送られます。

EJB サンプルの実行

このサンプルは、給与データの入力によって開始される、HR システムから給与システムへのデータ フローをシミュレートします。従業員のデータは、バイナリ データが使用されている従来の給与システムから取得されます。データは XML に変換されるので、従業員への支払情報を確定するための計算を実行できます。計算の結果は、再びバイナリ フォーマットに変換され、給与システムに送られます。

EJB サンプルの内容

EJB サンプル アプリケーションは WebLogic Integration インストールの *SAMPLES_HOME*\integration\samples\di\ejb ディレクトリにあります。次の表では、EJB サンプル アプリケーションに含まれるファイルについて説明しています。

表 3-2 EJB サンプル アプリケーション ファイル

ディレクトリ	ファイル	説明
\ejb	makefile	サンプル ソースを .jar ファイルに構築するためのメイク ファイル。
	WLXTExample.jar	Data Integration からエクスポートされたサンプル ワークフロー
	HR.mfl	サンプルの HR Bean から返されるバイナリ データの MFL ファイル。
	Payroll.mfl	サンプルの Payroll Bean に渡されるバイナリ データの MFL ファイル。
	Autopay.cmd	コマンド ラインからワークフローを開始するための Windows NT コマンド スクリプト。
	Autopay.sh	コマンド行からワークフローを開始するための UNIX シェル スクリプト。
	build.cmd	ソースから wlxtejb.jar を構築する。

表 3-2 EJB サンプル アプリケーション ファイル （続き）

ディレクトリ	ファイル	説明
\ejb\source	Payroll.java	従来の給与システムを表すサンプル EJB。
	PayrollHome.java	
	PayrollBean.java	
	HR.java	従来の HR システムを表すサンプル EJB。
	HRHome.java	
	HRBean.java	
	AutoPay.java	プリフォーマットされたメッセージを Event Topic 上に配置してサンプル ワークフローを開始させるプログラム。
	HexDump.java	サンプル EJB によって使用されるユーティリティ クラス。
	EmployeeRecord.java	サンプル HR EJB によって使用される従業員のデータ クラス。
WebLogic Platform ホーム ディレクトリ直下		
samples\integration\samples\di\ejb\lib	WLXTEJB.jar	サンプル アプリケーションの実行ファイル。

EJB サンプルの実行方法

EJB サンプルを実行するには、次に説明する手順に従ってください。

ステップ 1. ワークフロー定義のインポート

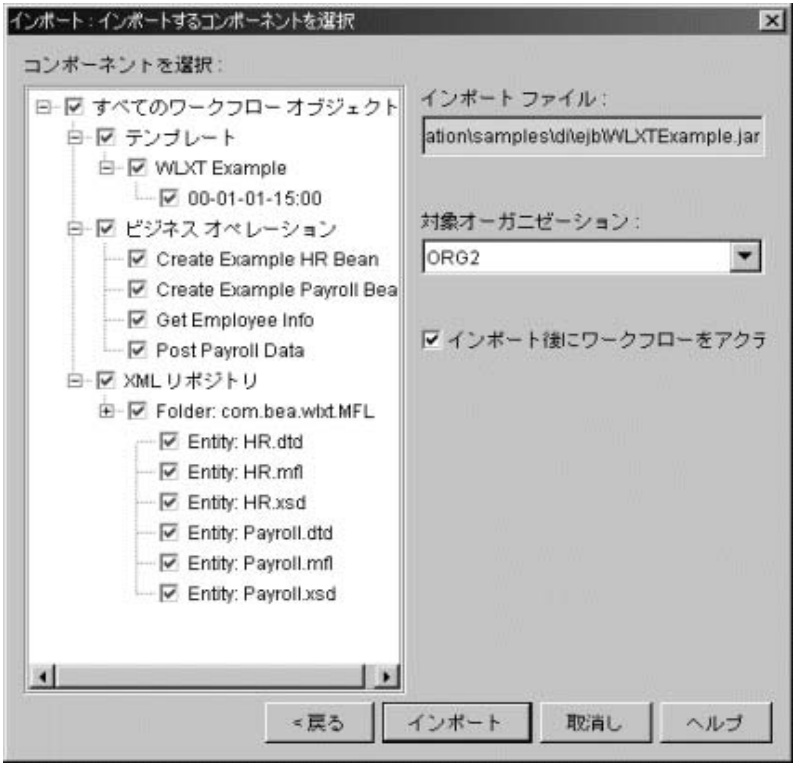
1. WebLogic Integration Studio を起動します。Studio の開始及びロギングについては、『*WebLogic Integration Studio ユーザーズ ガイド*』の「Studio インタフェースの使用法」を参照してください。

2. [ツール | パッケージをインポート] を選択します。[インポート：ファイルを選択] ダイアログ ボックスが表示されます（図 3-5）。

図 3-5 [インポート：ファイルを選択] ダイアログボックス

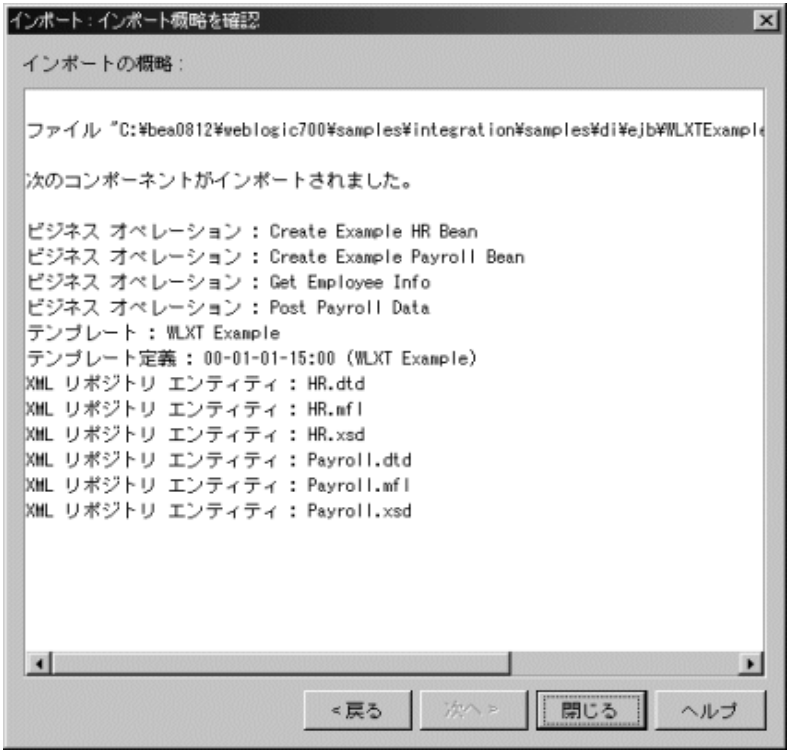
3. [参照] をクリックし、定義ファイル `WLXTExample.jar` を選択し、[開く] をクリックします。[次へ] ボタンをクリックすると、[インポートするコンポーネントを選択] ダイアログ ボックスが開きます（図 3-6）。

図 3-6 [インポート：インポートするコンポーネントを選択]



4. インポート後、実行中のワークフローのチェック ボックスが選択され、すべてのコンポーネントが選択されていることを確認してから [インポート] ボタンをクリックします。[インポート：インポート概略を確認] ダイアログ ボックスが表示されます（図 3-7）。

図 3-7 [インポート：インポート概略を確認]

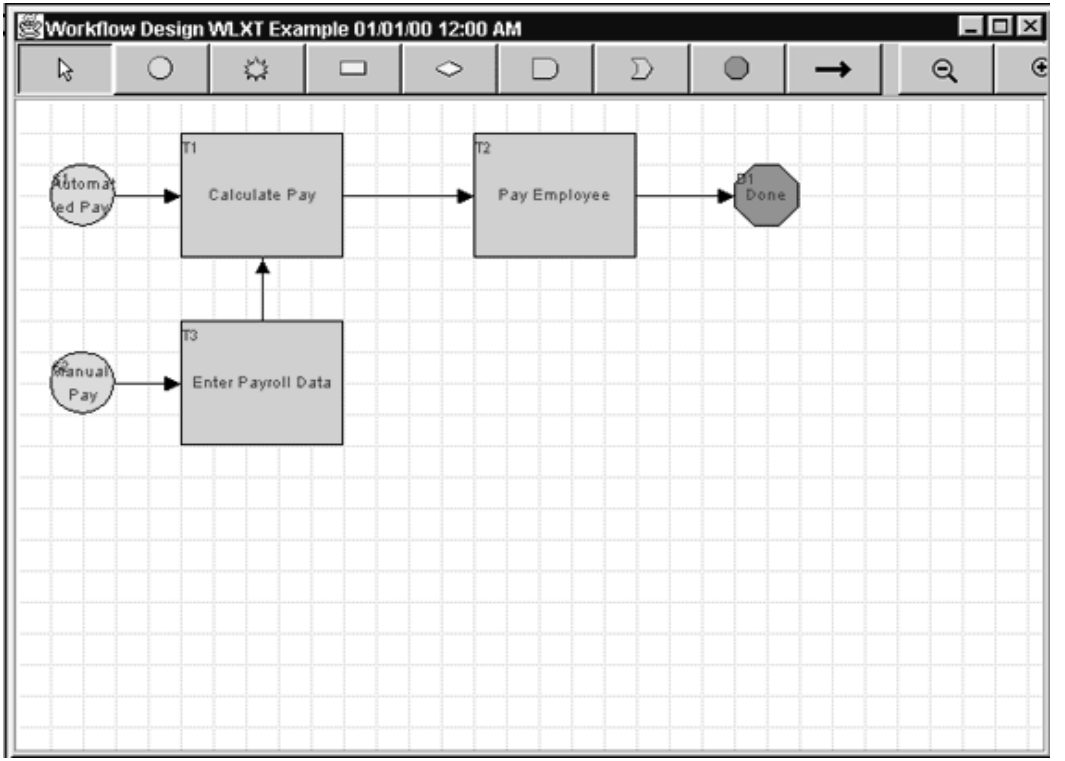


5. 正しいコンポーネントがリストされていることを確認します。正しくない場合は、[戻る] ボタンをクリックしてコンポーネントをもう一度選択します。コンポーネントのリストが正しければ、[閉じる] をクリックします。これでテンプレートを開く準備が整いました。

ステップ 2. テンプレートを開く

1. ナビゲーション ツリーで、前のステップでインポートした `WLXT Example` テンプレートを展開します。テンプレート定義 `1-1-00-12:00-AM` を右クリックします。
2. [開く] を選択します。このサンプル アプリケーションのために作成されたワークフローが表示されます（図 3-8）。

図 3-8　WebLogic Integration のサンプルのワークフロー



ステップ 3. ワークフローの開始

サンプルで作成したワークフローの開始方法には 2 つあります。

- WebLogic Integration Worklist から
- コマンド ラインから

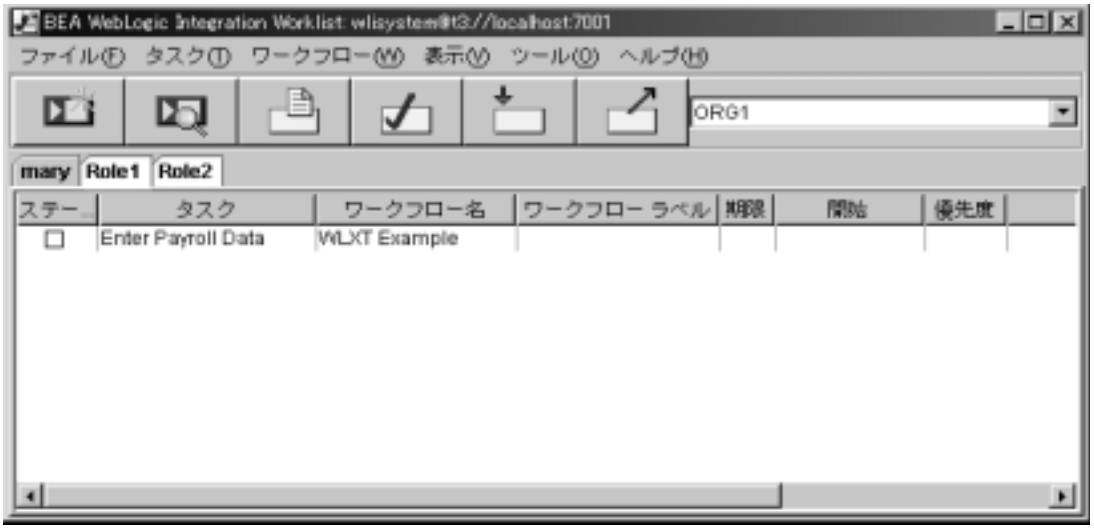
ワークフローを開始すると、HR システムと給与システム間でシミュレートしたデータ フローを Studio で監視できます。

WebLogic Integration Worklist から

サンプル ワークフローを WebLogic Integration から開始する手順を次に示します。

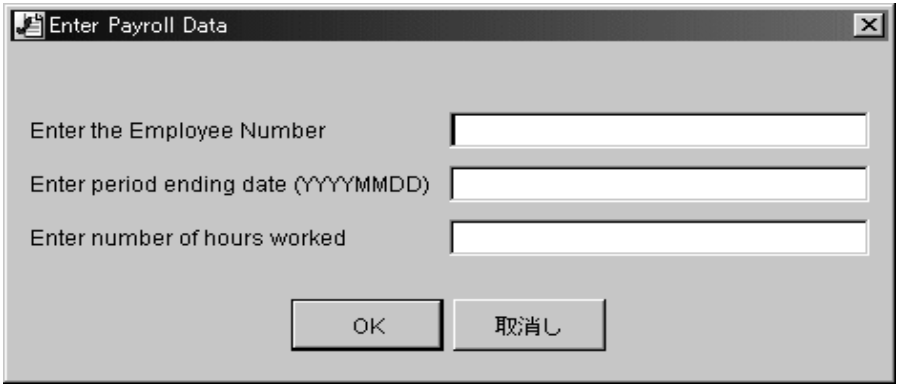
1. WebLogic Integration Worklist を開始し、[ワークフロー | ワークフローを開始] を選択します。
2. [WLXT Example] を選択します。[OK] をクリックします。

図 3-9 WLXT のサンプル ワーク リスト



3. [Enter Payroll Data] タスクを右クリックし、[実行] を選択します。[Enter Payroll Data] ダイアログ ボックスが表示されます（図 3-10）。

図 3-10 [Enter Payroll Data] ダイアログ ボックス



4. 給与データを入力し、[OK] をクリックします。タスクが開始され、ワークフローが実行されます。

   注意： この例では、従業員番号として 1 ～ 4 のみが有効です。期間の終了日付と、作業時間数として任意の数値を入力できます。

コマンド ラインから

コマンド ライン プロンプトからサンプル ワークフローを開始する手順は以下のとおりです。

1. テキスト エディタでスクリプト（Windows NT では `Autopay.cmd`、UNIX では `Autopay.sh`）を開き、`WebLogic Integration` プロセス エンジンの位置を確認します。デフォルトでは、位置は `localhost`、ポート番号は `7001` になっています。
2. お使いのシステムのホストとポートに一致するようにロケーション情報を変更します。
3. 環境変数 `WL_HOME` をディレクトリのパス名として設定します。ここには、`WebLogic Server` がインストールされています。たとえば、

   ```
   set WL_HOME=c:\bea\weblogic700
   ```

4. 環境変数 `WLI_HOME` をディレクトリのパス名として設定します。ここには、`WebLogic Integration` がインストールされています。たとえば、

   ```
   set WLI_HOME=c:\bea\weblogic700\integration
   ```

5. システム（Windows NT または UNIX）のコマンド スクリプトを実行し、図 3-10 に示されるのと同じパラメータを渡します。たとえば、

   ```
   Autopay 1 2000-11-30 60
   ```

   ワークフローが開始します。

手順 4. 変数の値を確認する

サンプル データフローを監視する手順は次のとおりです。

1. Studio で、[ワークフロー インスタンス] ダイアログ ボックスを表示します。ワークフロー監視の詳細については、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフローのモニタリング」を参照してください。
2. ポップアップ メニューで任意のワークフロー インスタンスを右クリックし、[変数] を選択します。[ワークフロー変数] ダイアログ ボックスが表示されます。
3. それぞれの変数を確認し、3-16 ページの「ステップ 3. ワークフローの開始」にある値に設定されているか検証します。

索引