Fiorano コレオグラフィによる
ビジネス プロセス構築の概要 (基礎編)

(製品評価ガイド)

対象バージョン : 2007 SP7 および 9.0.0

はじめに

この文書は、Fiorano SOA プラットフォームのオーケストレーション ツール (Fiorano Studio) を用いてビジネス プロセスを構築する基本的な方法について、例題を用いて説明するものです。

このガイド ブックの他に、以下のガイド ブックも用意されています。併せてご参照ください。

- Fiorano SOA プラットフォームのダウンロード (製品インストーラと評価用ライセンスの取得方法)
- Fiorano SOA プラットフォームのインストール
- Fiorano SOA プラットフォームの起動方法
- Fiorano SOA プラットフォームのアーキテクチャ概要

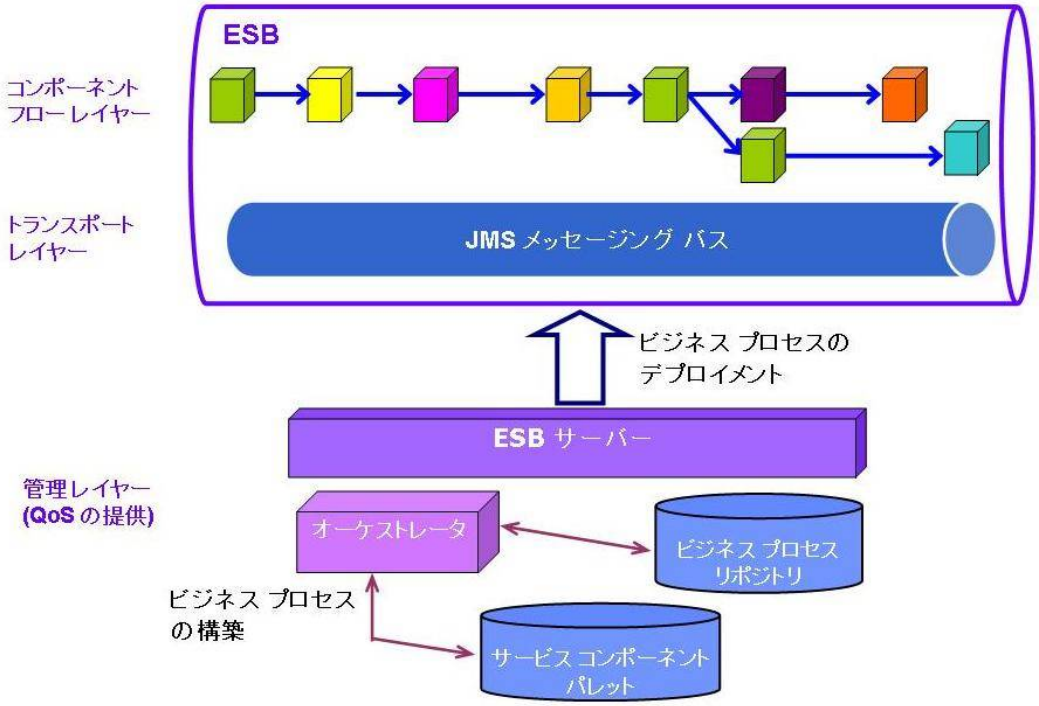
目次

1. SOA 実装フレームワーク (SOAIF)

この章では、Fiorano SOA プラットフォーム上で稼動するビジネス プロセスのフレームワークである、SOAIF について説明します。

1.1 SOAIF のレイヤー構造

Fiorano SOA プラットフォームは、SOA 実装フレームワーク (SOAIF : SOA Implementation Framework) に基づいたビジネス プロセスを構築、実行します。SOAIF とは Fiorano 社の SOA コンセプトを実現するためにデザインされたフレームワークで、ビジネス プロセスを実装するためのものです。SOIF によって実装されるビジネス プロセスは、下図に示すレイヤー構造を採っています。



本ガイドブックは、コンポーネント フロー レイヤーであるプロセス フローのオーケストレーションの方法を説明するものです。

SOAIF および Fiorano SOA プラットフォームのアーキテクチャの詳細については、ガイドブック『Fiorano SOA プラットフォームのアーキテクチャ概要』を参照してください。

2 Fiorano コレオグラフィ ツールの起動と基本操作

2.1 Fiorano Studio の起動

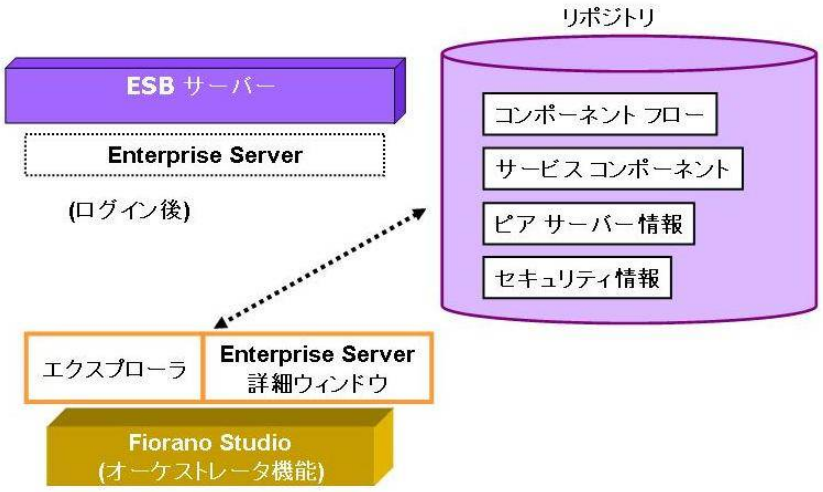
Fiorano コレオグラフィは、Fiorano Studio 内から使用します。

コレオグラフィを使用するためには、次の手順が必要です。

1. ピア サーバーと ESB サーバーの起動
2. Fiorano Studio の起動
3. ESB サーバーへのログイン

詳細は、ガイドブック『Fiorano SOA プラットフォーム の起動方法』を参照してください。

ESB サーバーへのログインが完了すると、次の図のように、リポジトリにアクセスできるようになります。

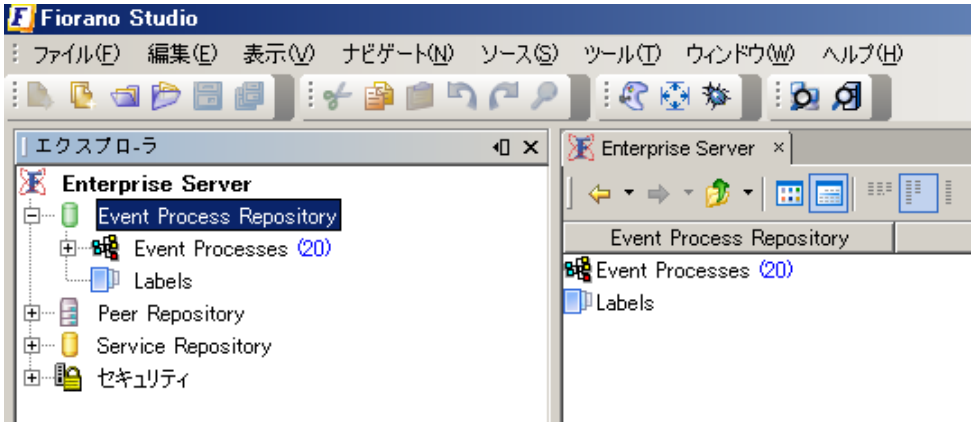


2.2 コンポーネント フローのリポジトリ

このセクションでは、コンポーネント フローのリポジトリについて説明します。

[ログイン ブラウザ] はしばらく使用しませんので、閉じてしまいます。[ログイン ブラウザ] のタイトルバーの右端にある [X] (閉じる) ボタンをクリックすることで、ウィンドウを閉じることができます。

[エクスプローラ] ウィンドウ



[エクスプローラ] ウィンドウには、Enterprise Server (ESB サーバー) のリソースがツリー表示されています。
ツリー表示の最上段にある **[Event Process Repository]** が、構築済みもしくは構築中のコンポーネント フローを格納しているリポジトリです。
Event Process (イベント プロセス) という名称を使用しているのは、コンポーネント フローが JMS メッセージングに基づくイベント ドリブン (イベント駆動) のプロセスとして構成されるためです。

[Event Process Repository] には、次の 2 つが格納されています。

- **Event Processes** – コンポーネント フローをサブ フォルダーに分けて格納
- **Labels** – コンポーネント フローには、ライフサイクルのフェーズに応じたラベルをつけることができます。**[Labels]** の下に、ラベル付け可能なフェーズが定義されています。

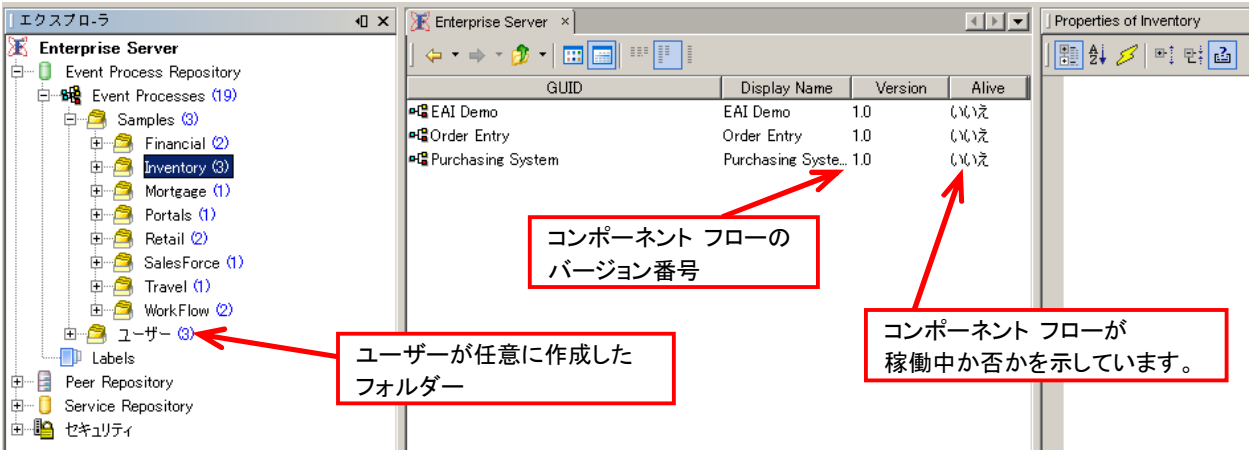
ラベル付け可能なフェーズの種類

Development (開発中) – **デフォルトで付けられるラベル**
QA (Quality Assurance テスト中)
Staging (本番への移行待ち)
Production (本番)

ラベルの詳細については、「2.3.2 コレオグラフィの画面構成」を参照してください。

[Event Processes] の展開

[Event Processes] のツリーを、下の画面のように展開してください。



インストールした状態では、**[Event Prepossess]** の下には、**[Samples]** とそのサブ フォルダーのみが作成されています。ユーザーがコンポーネント フローを作成する際に、任意のフォルダーを新規に追加していくことができます。
[Samples] の下に格納されているコンポーネント フローは、Fiorano 社が作成したサンプル フローで、カテゴリーに分けて格納されています。各フォルダーの後ろに表示されている青色の数字は、格納されているコンポーネント フローの数を表しています。

上の画面のように、**[エクスプローラ]** 上で **[Inventory]** を選択してください。**Inventory** フォルダーには、その名前が示すように受発注関連のフローが格納されています。
右側の **[Enterprise Server]** ウィンドウに、**Inventory** に格納されているサンプル フローの一覧が表示されます。
次のセクションにおいて、**Inventory** フォルダーにある “EAI_DEMO” を例にとり、ツールの操作方法、フローの実行方法

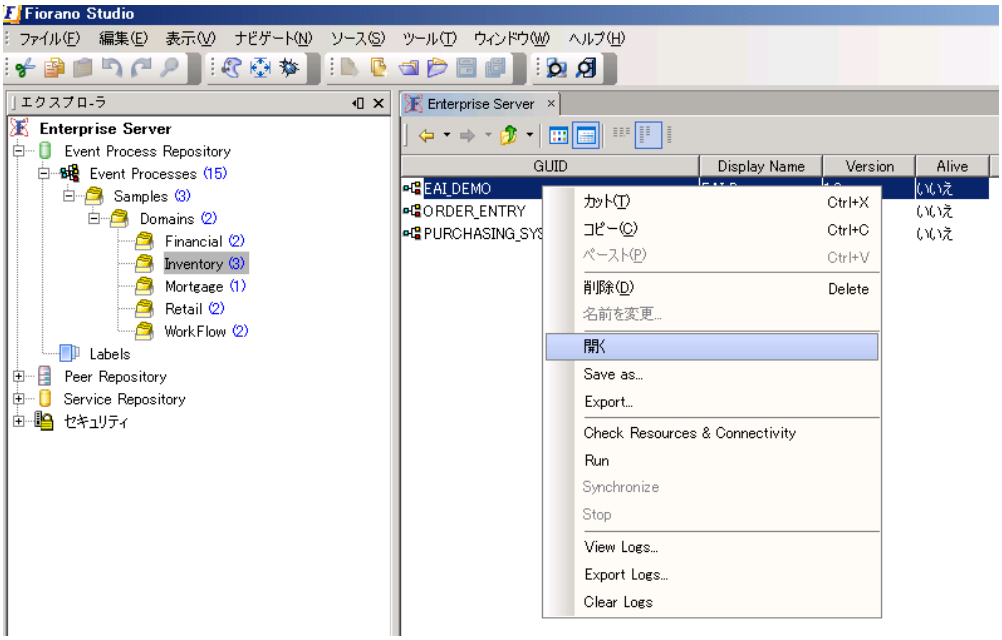
を説明します。

2.3 コレオグラフィの操作

サンプルとして用意されているコンポーネント フロー (EAI_DEMO) を例に、ツールの操作方法、コンポーネント フローのデプロイメント、実行などの方法について説明します。

2.3.1 コンポーネント フローのオープン

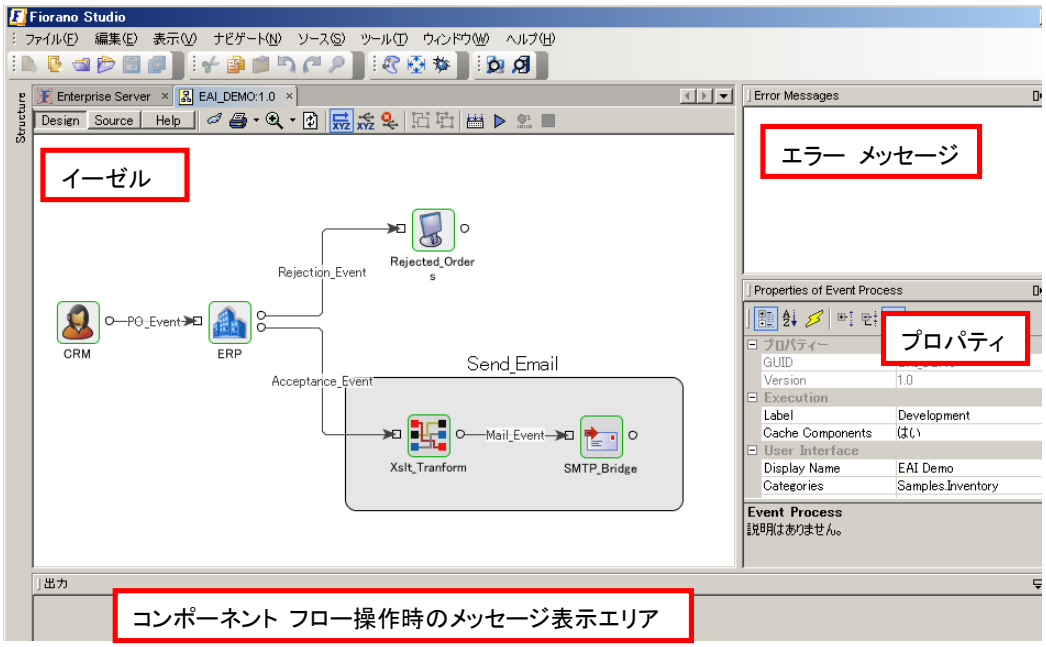
下の画面のように、**[Enterprise Server]** ウィンドウに表示されているリストから **EAI_DEMO** を右クリックしてください。表示されたメニューから **[開く]** を選択します。**EAI_DEMO** をダブルクリックしても、フローを開くことができます。



2.3.2 コレオグラフィの画面構成

コレオグラフィの画面

EAI_DEMO を開くと、コレオグラフィが起動され、下の画面が表示されます。



コンポーネント フローを構築するためのイーゼル、**[Error Message]** ウィンドウ、操作時のメッセージ表示エリアが新たにオープンされるウィンドウです。ログイン時に表示されたプロパティ ウィンドウは、コンポーネント フローのプロパティ情報を表示するようになります (ウィンドウのタイトル バーが **[Properties of Event Process]** に変更されます)。
プロパティ ウィンドウが閉じている場合には、メニュー バーの **[ウィンドウ]** メニューから **[Properties]** を選択してウィンドウを表示してください。

イーゼル

イーゼルは、コンポーネント フローを視覚的に定義、構築するためのエリアです。イーゼルでは、コンポーネント フローの構築だけでなく、ESB 上で稼動しているコンポーネント フローの監視も行えます。

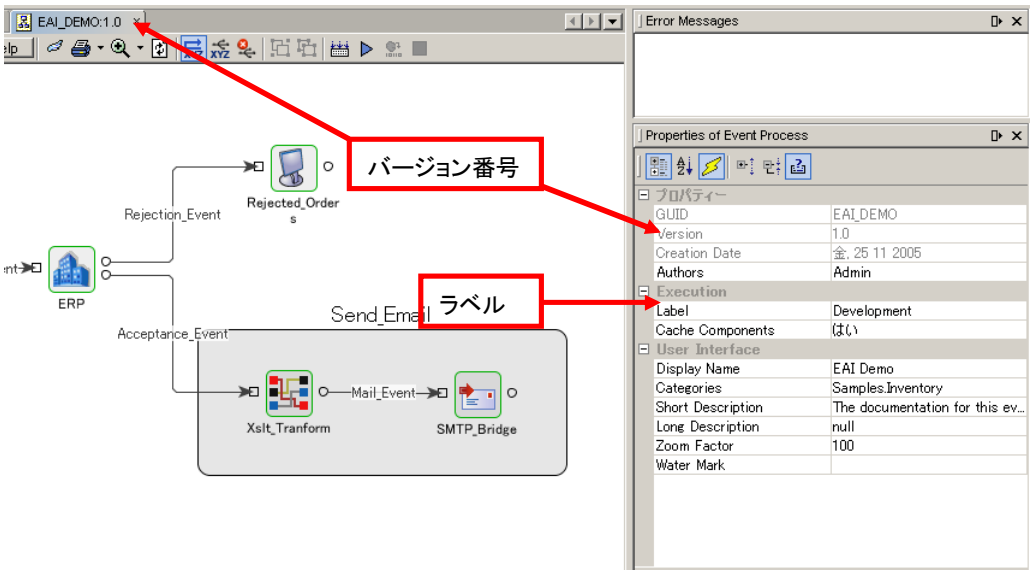
[Error Message] ウィンドウ

Error Message ウィンドウには、コンポーネント フロー構築に関わるシステム メッセージが表示されます。

コンポーネント フローのバージョン番号とラベル

イーゼル ウィンドウのタブには、フローの名前が “EAI_DEMO 1.0” と表示されます。この “1.0” は、フローのバージョン番号を表しています。バージョン番号は、プロパティ ウィンドウ (**[Property of Event Process]**) にも表示されます。
プロパティ ウィンドウには、コンポーネント フローに付けられたラベルも表示されます。下図を参照してください。

バージョン番号やラベルは、コンポーネント フローのバージョン管理やデプロイメント管理に利用します。フローの一部を変更したものを別のバージョンとすることもでき、フローの様々なバリエーションをバージョンによって区分けすることにも利用できます。バージョン番号のナンバリングや意味づけは、ユーザーが任意に行えます。デフォルトのバージョン番号は “1.0”、ラベルは “Development” となっています。

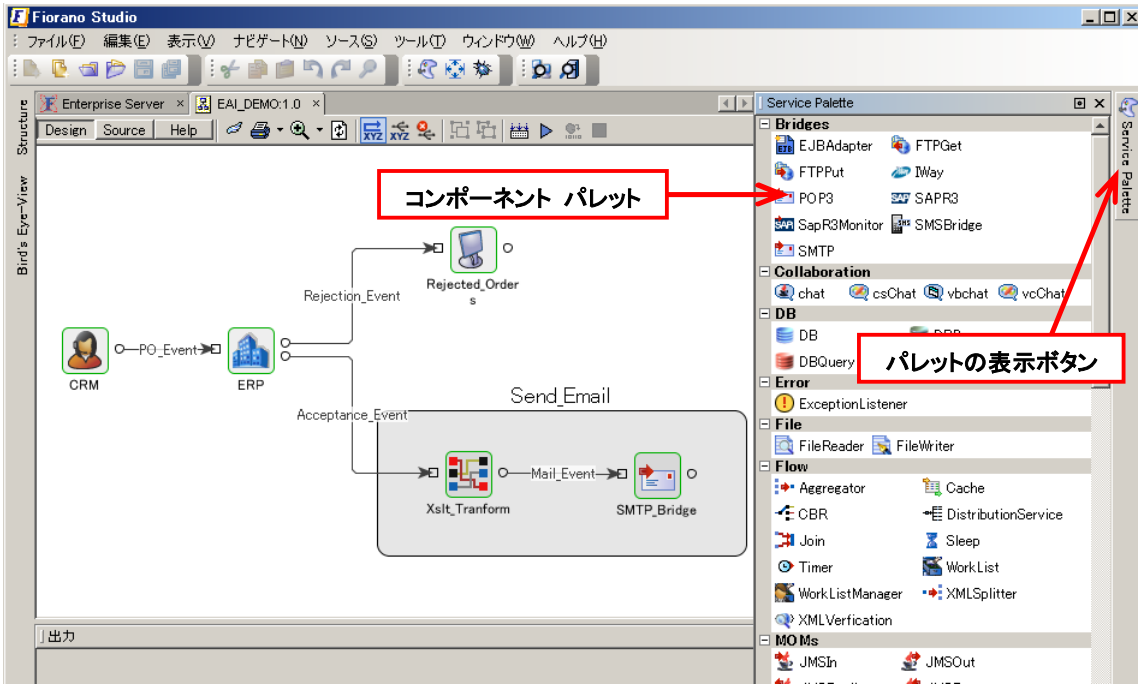


バージョン番号、ラベルの変更方法については、製品マニュアルを参照してください。

サービス コンポーネント パレット

コンポーネント フローで利用できるサービス コンポーネントは、パレットに表示されます。パレットには、Fiorano Software が製品にバンドルしているサービス コンポーネントの他に、ユーザーが独自に開発したサービス コンポーネントも表示されます。

コンポーネント パレットが表示されてない場合は、右側のウィンドウ枠にある、**[Service Pallets]** ボタンをクリックしてください。**[ウィンドウ] -> [Enterprise Application] -> [Service Pallet]** メニューによって表示させることもできます。



イーゼルのツール バー

イーゼル ウィンドウには、ツール バーがあります。ツール (コマンド) ボタンの種類は以下の図のようになっています。このセクションでは、主要なツール (コマンド) ボタンの機能について説明します。

フロー図を操作するためのツール (コマンド) ボタン

フロー図の表示

コンポーネント フローは、実際には XML 形式のソースとしてリポジトリに格納され、実行時には XML ソースが関係するピア サーバーにデプロイされます。ただし、フロー構築など人間による操作を行う場合には、理解しやすい図の形式で扱います。通常のフロー構築、システム管理においては、XML ソースを直接操作する必要はまったくありません。このため、デフォルトの設定では XML ソースを表示できないようになっています。XML ソースを表示させるためには、オプション設定を変更する必要があります。

フローの説明の表示

`[Help]` をクリックすると、コンポーネント フローの説明が表示されます。Fiorano 社が作成したサンプル フローの場合、フローの背景を説明するシナリオとフローの各実行ステップについて説明が表示されます。ユーザーがフローを作成する場合には、ユーザー独自のコメントを記述することができます。

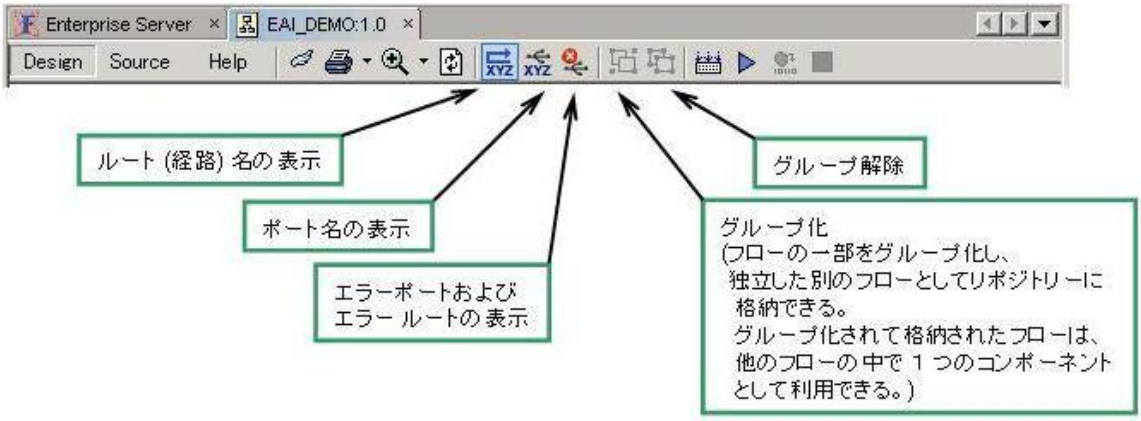
鳥瞰図

複雑なステップのフローの場合、その図は大きなものとなります。フロー図のすべてを見るためには、イーゼルをスクロールしながら参照することになります。このコマンドは、図全体を鳥瞰図として表示させるためのもので、大きなフローの全体像を把握するのに有用です。

フロー図のリフレッシュ

フロー図は、フローの実行や稼動状況の監視にも使用します。実行中にフローの一部を変更したりコンポーネントを置き換えたりした場合や、何らかの障害によって実行が停止していたピア サーバーやコンポーネントが再起動された場合に、フロー図にその状況がリアルタイムに反映されない場合があります。更新ボタンによって、フロー図 (稼動状況の表示) を最新のものにします。

コンポーネント フロー構築時に使用するコマンド



ルート名、ポート名の表示

ルート名、ポート名の表示、非表示を切り替えられます。詳細は、『2.3.3 フロー図内の操作対処オブジェクト』を参照してください。

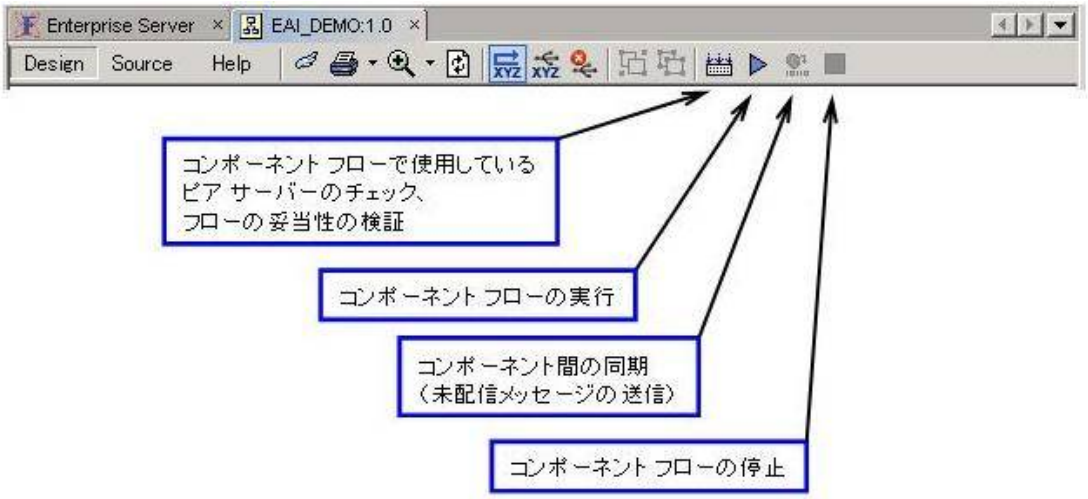
エラー ポートおよびエラー ルートの表示

デフォルトでは、エラー ポート、エラー ルートは非表示になっています。エラー ポートを用いて例外処理のフローを構築する場合や例外処理のフローを表示する場合にはこのボタンによってエラー ポートおよびエラー ルートを表示状態にします。詳細は、後のセクションで説明します。

グループ化

フローの一部をグループ化することができます。詳細は、『2.3.3 フロー図内の操作対処オブジェクト』を参照してください。

コンポーネント フローの実行に使用するコマンド



コンポーネント フローの実行に関係するコマンドの詳細については、『4. コンポーネント フローの実行』の章を参照してください。

2.3.3 フロー図内の操作対象オブジェクト

イーゼル上のコンポーネント フロー図は、次のオブジェクトからなっており、各オブジェクトに対して個別の操作が行えます。

- サービス コンポーネント
- ポート (サービス コンポーネントに付随するアウトプット ポート、インプット ポート、エラー ポート)
- ルート (ポート間を結んだイベント メッセージの転送経路)
- グループ

サービス コンポーネント

サービス コンポーネントは、コンポーネントの処理ロジックを実行します。

ポート

ポートは、サービス コンポーネントへのメッセージ (データ) 入力、サービス コンポーネントからのメッセージ (データ) 出力を担います。エラー ポートは、エラー発生時のエラー メッセージの出力を行います。

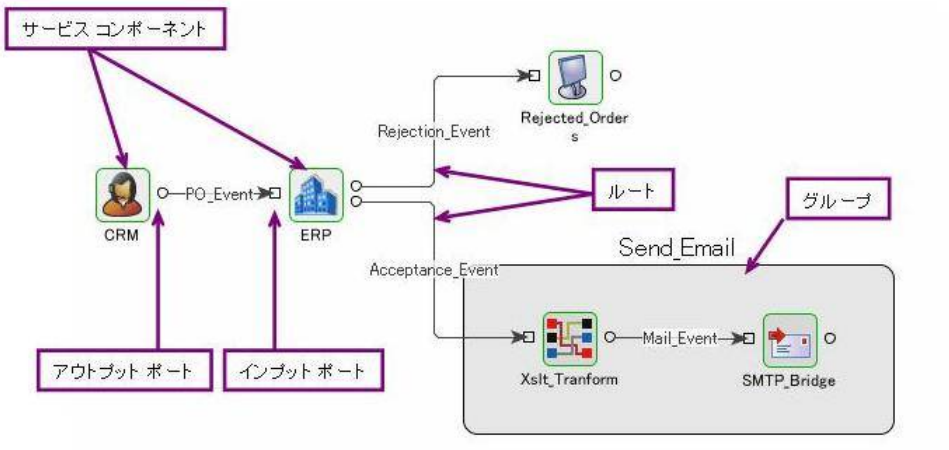
ルート

ルートは、サービス コンポーネントのアウトプット ポートと他のサービス コンポーネントのインプット ポートとの間を結びつけるもので、メッセージ (データ) の経路としての役割を担います。

グループ

フローの一部をグループ化し、その内容を表す名前を付けることで、イーゼル上に表示された複雑なコンポーネント フローが理解しやすくなるという効果が得られます。

グループ化の最大の効果は、グループを別のコンポーネント フローにおいて再利用することにあります。グループは、リポジトリに単独のプロセス フローとして格納することができます。格納されたフローは、他のフローにおいて 1 つのコンポーネントとして利用できるようになります。これを、コンポジット コンポーネントと呼びます。



各オブジェクトにはユーザー任意の名前を付けることができます。イーゼルにデフォルトで表示される名前は、サービス コンポーネントとルートの名前です。ポートの名前を表示させるためには、前述したコマンド ボタンを使用します。

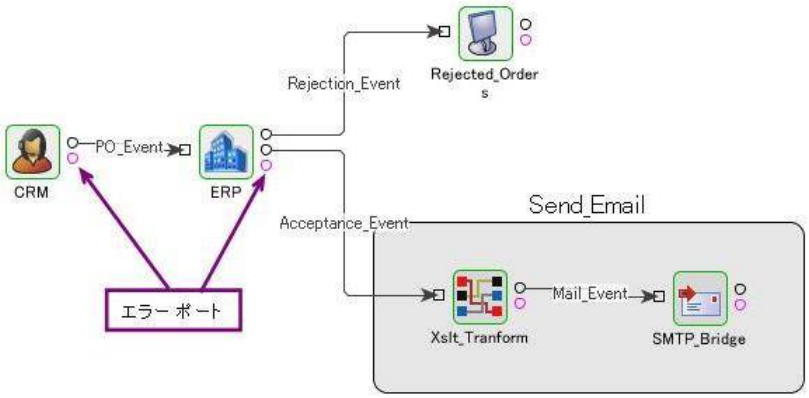
各操作オブジェクトに対して行える操作は、オブジェクトのカテゴリーやオブジェクトの状況によって異なります。オブジェクトにマウス ポインターを合わせ、マウスの右ボタンをクリックすると、オブジェクトに対して操作できるコマンドの一覧がプルダウン

メニューの形で表示されます。メニュー中でグレーアウト（灰色）されているコマンドは、その時点のオブジェクトの状況では実行不可であることを示しています。(試しに、様々なオブジェクトに対してメニュー表示を行ってみてください。)

2.3.4 エラー ポートとエラー フローの表示

デフォルトでは、エラー ポートとそこからのルートは表示されません。エラー ポートおよびそのルートは、前述したエラー ポートの表示コマンド ボタンで表示させることができます。

下図は、サンプルの EAI-DEMO に対してエラー ポートを表示させた結果です。エラー ポートが表示されていますが、そこからエラー処理を行うためのルートがでていません。これは、エラー発生時の処理フローが構築されていないことを示しています。



3 コンポーネント フローの構築手順 (基礎編)

この章では、コンポーネント フローの構築手順を、例題を用いて説明します。

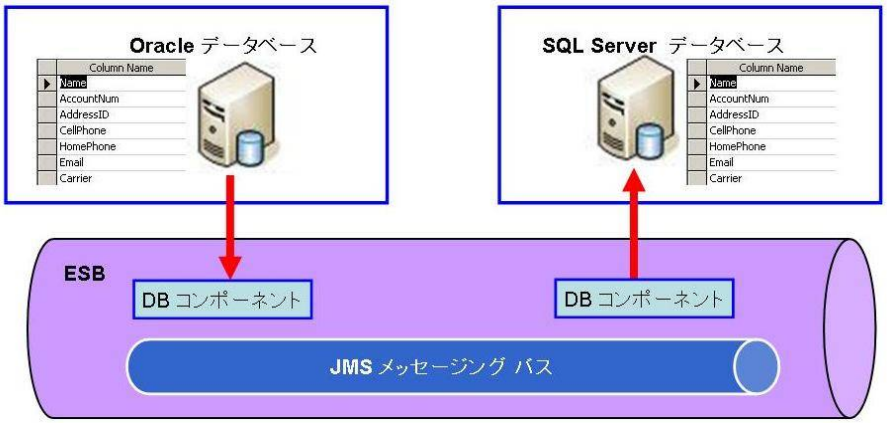
例題の構築手順は、概ね次のステップとなります。章立てもこのステップの順となっています。

1. 例題の説明
2. 例題で使用する DBMS の準備
3. サービス パレットからサービス コンポーネントをドラッグ&ドロップ
4. サービス コンポーネントのプロパティ設定
5. サービス コンポーネント間のルート設定
6. サービス コンポーネント間のデータ変換の設定

ここで示す構築手順は一例であり、必ずしもこの手順に従う必要ありません。

3.1 例題

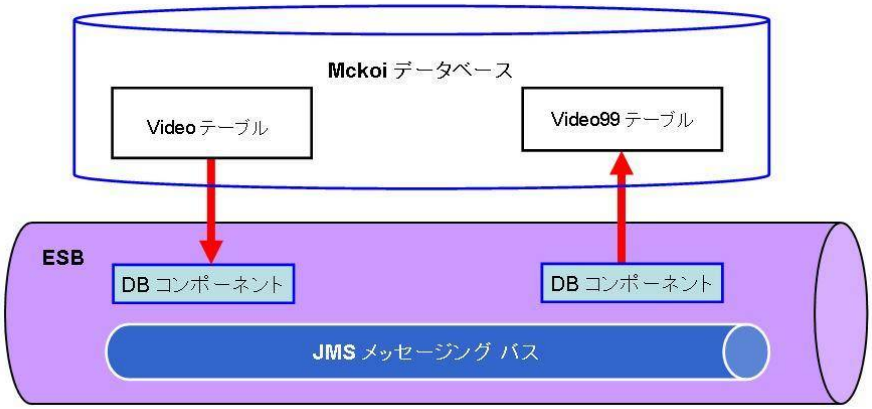
異なる 2つのアプリケーションが同じビジネス データをそれぞれのデータベースで管理していることは、よくあることです。このような場合、一方のアプリケーションが参照しているデータの値と他方のアプリケーションで参照しているデータの値が異なると、業務上の支障をきたすというようなことも発生します。このような状況では、データベース間でビジネス データの同期をとり、アプリケーション間のデータ整合性を維持することがたいへん重要になります。また業務によっては、夜中に一度実施するようなバッチ処理で整合性を取るのでは間に合わない、リアルタイムで最新のデータに更新されていることが求められることもあります。下の図は、リアルタイムでデータベース間の整合性をとるための最もシンプルなソリューション例を示しています。



Oracle データベースを用いている (図の左側) のアプリケーションによってデータが更新されます。ESB 上の左側の DB コンポーネントは、一定時間 (例えば 10 秒) 毎に Oracle データベースにログインし、あらかじめ設定されている SQL クエリによって変更のあったデータを検出します。検出したデータを次の(右側の) DB コンポーネントに JMS プロトコルを介して渡します。図の右側の DB コンポーネントは、受け取ったデータを SQL Server データベースに SQL クエリ (INSERT) によって書き込みます。この処理によって、最大 10 秒の遅れで 2 つのデータ間の整合性を維持できるようになります。

実際にコンポーネント フローを構築していただくのが、製品を効果的に評価していただける最良の方法だと考えております。例題を下記のように簡素化し、余分なシステム設定をできるだけ排除してより簡単にコンポーネント フローが構築できるようにしました。実際にコンポーネント フローを構築しながら製品の評価を行ってくださるようお願いいたします。

アクセスするデータベースを、製品にバンドルされているフリーソフトウェアの 1つである GNU プロジェクトの Mckoi データベースに変更します。Mckoi を使用するために必要な設定はあらかじめ為されており、Fiorano SOA プラットフォームをインストールするだけで使用できるようになっています。また、データベース実行などの .bat ファイルも用意されています。



例題の変更ポイント ：

1 使用するデータベースは Mckoi の一つだけとし、

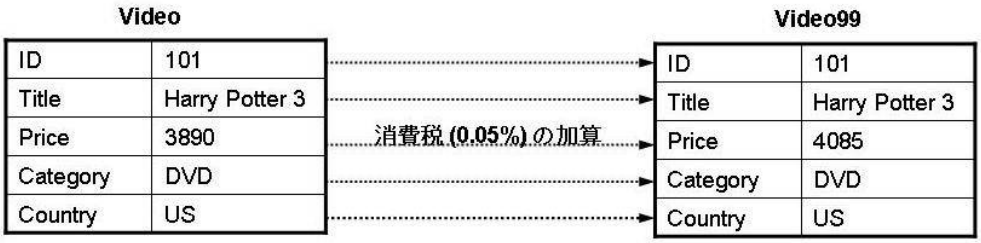
2 `[Video]` テーブルを一定時間毎にクエリ (SELECT) し、

3 SELECT の結果を、次の DB コンポーネントに転送し、

4 受け取ったデータを `[Video99]` テーブルに書き込む (INSERT)

5 `Video99` に書き込むデータには簡単な変更を加える (価格を消費税込みに変更)

テーブルの構造は次のようにします。



このビジネス プロセスをコンポーネント フローとして構築する手順を、順をおって以下に示します。

3.2 Mckoi データベースの準備

コンポーネント フローの作成の前に、データベースの準備をしておきます。

本ガイド ブックの『付録 Mckoi データベースの使用方法と設定』を参考に、

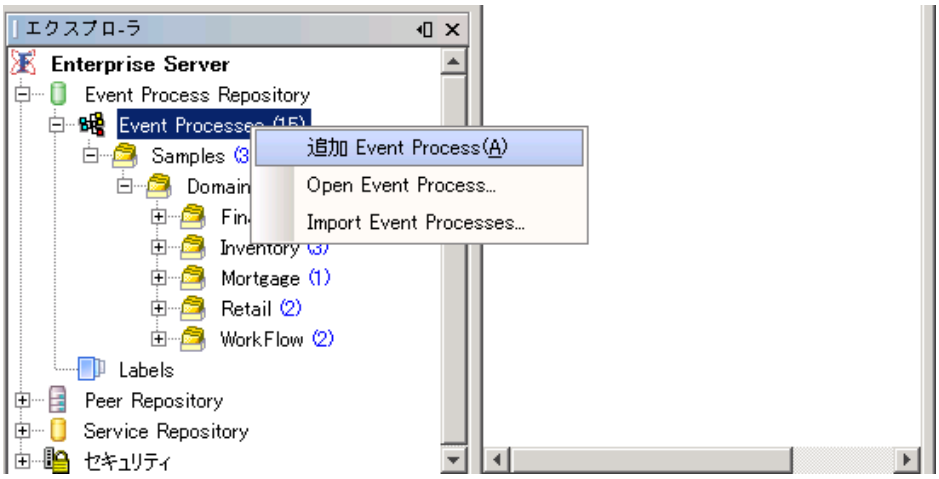
1. Mckoi データベースの起動
2. Video テーブルと Video99 テーブルの生成
3. Video テーブルへのデータ入力

を実行してください。

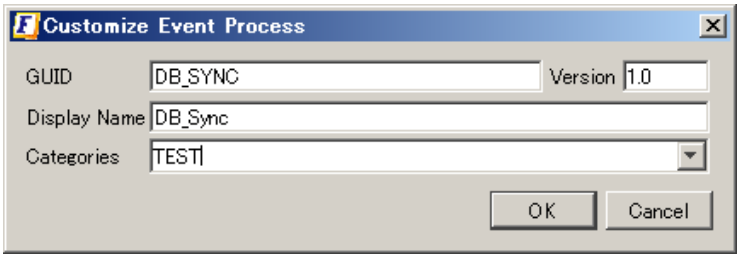
3.3 新規コンポーネント フローの登録

新規のコンポーネント フローをリポジトリに登録します。

[エクスプローラ] に表示されているツリーから **[Event Processes]** を右クリックします。表示されたメニューから **[追加 Event Processes]** を選択します。



ポップアップ ウィンドウが表示されますので、コンポーネント フローの名前（GUID、Display Name）、リポジトリ上のカテゴリーを指定します。それぞれの意味は、以下の通りです。



GUID：システム内で使用する名前

Display Name：ツール（Fiorano Studio など）上に表示される名前

Category：リポジトリ内の保存フォルダー（日本語名とすることができます）

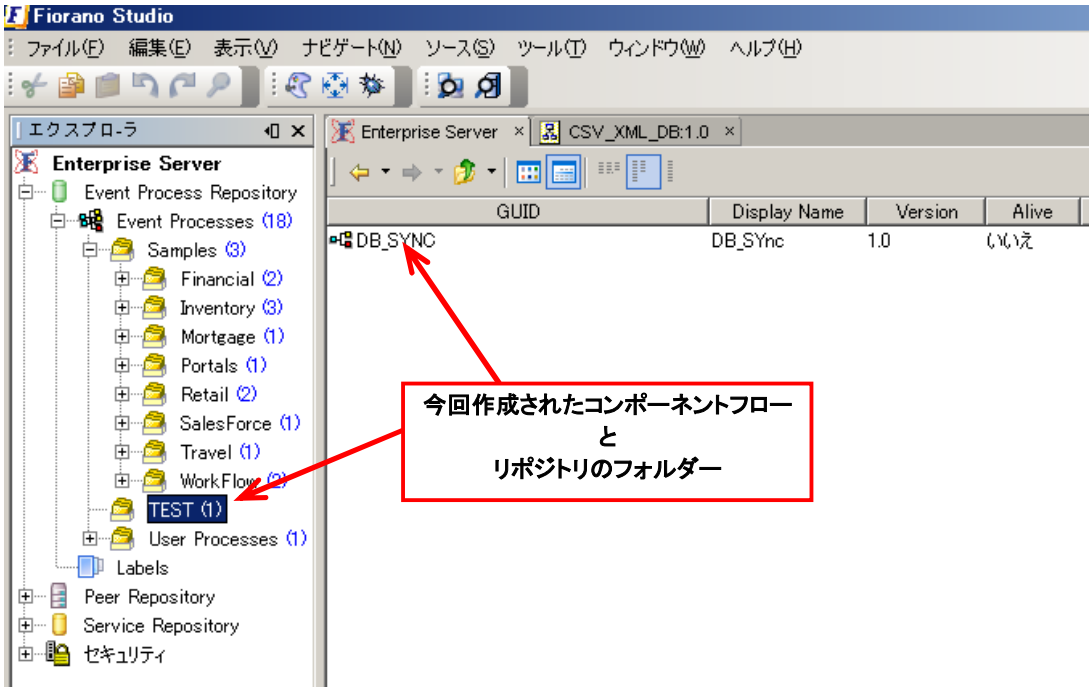
Version：任意のバージョン番号を付けることができます。デフォルト値は、1.0 です。

GUID と **Display Name** の入力フィールドは連動しており、どちらか一方に入力すれば、他方も自動的に入力されます。

図のように入力し、**[OK]** ボタンをクリックすると、次のダイアログ ボックスが表示されます。

このダイアログは、コンポーネント フローのイーゼルを開くか否かを問い合わせています。[はい] を選択するとイーゼルが自動的に開き便利ですが、今回は [いいえ] を選択してみてください。

コンポーネント フローの登録が完了し、リポジトリは次のように変更されます。



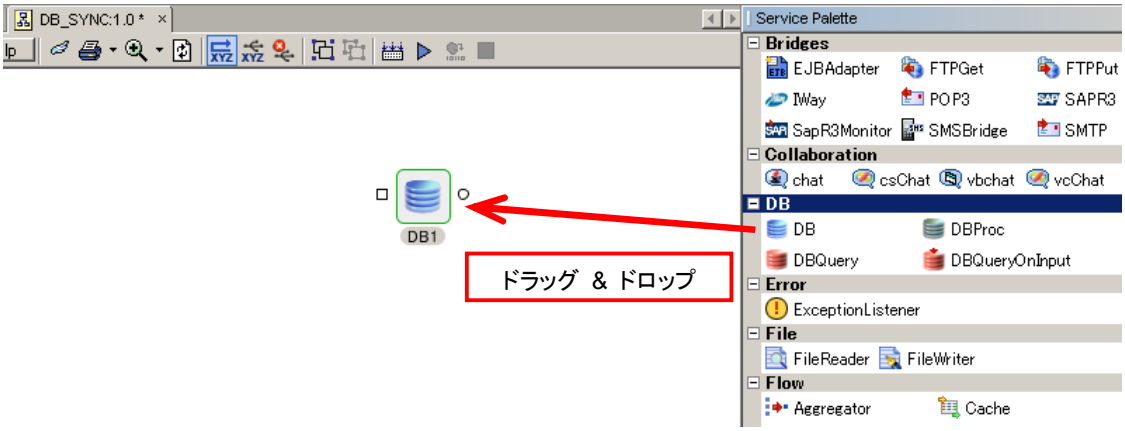
[エクスプローラ] ウィンドウの各カテゴリーの行に表示されている青色の数字は、そのカテゴリーに格納されているコンポーネント フローの数を表しています。

[Enterprise Server] ウィンドウに表示されている DB_SYNC の行をダブルクリックして、イーゼルを開きます。DB_SYNC の行を右クリックして、表示されるメニューから [開く] を選択しても開くことができます。

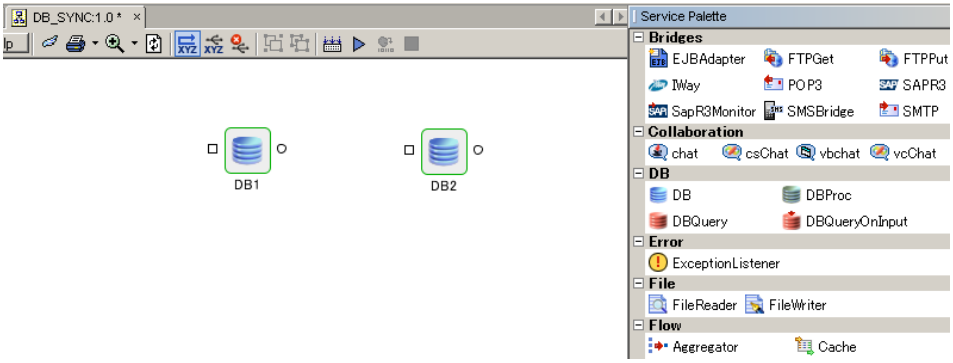
3.4 サービス コンポーネントのドラッグ&ドロップ

コンポーネント フローに必要なサービス コンポーネントを、パレットからイーゼルにドラッグ&ドロップします。パレットが表示されていない場合は、右のウィンドウ枠にある **[Service Pallets]** ボタンをクリックしてください。

下の画面は、パレットから DB コンポーネントをドラッグ&ドロップした状態です。DB コンポーネントは、 JDBC ドライバーを介して RDBMS にアクセスしてクエリ (SQL) の実行を依頼し、その結果を RDBMS から受け取るコンポーネントです。

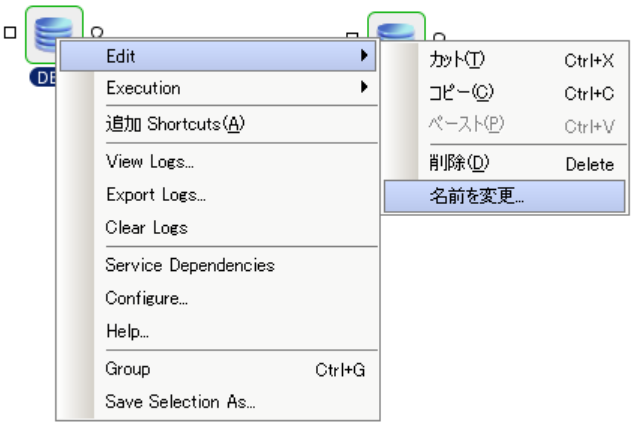


次の画面は、さらにもう一つの DB コンポーネントをドラッグ&ドロップした状態を示しています。



ドラッグしたサービス コンポーネントには、コンポーネント名が付けられます。ツールが付与した名前は、"DB1" と "DB2" ですが、ユーザー任意の名前に変更することもできます (日本語名も可です)。ここでは、アクセスするテーブル名をコンポーネントの名前とすることにします。DB1 は Video に、DB2 は Video99 に変更します。

コンポーネント名を変更するには、コンポーネントを右クリックし、メニューから **[Edit]** → **[名前を変更...]** を選択します。



表示されたポップアップ ウィンドウに任意の名前を入力します。この例では、「ビデオ」とします。

同じ操作によって、DB2 コンポーネントにも「ビデオ 99」という名前を付けます。

3.5 サービス コンポーネントのプロパティ設定 ([ビデオ] コンポーネント)

ドラッグしてきたサービス コンポーネントは、汎用的に使えるように開発されたものです。ユーザーの環境に合わせてプロパティ値をカスタマイズする必要があります。サービス コンポーネントをダブル クリックするとサービス コンポーネントの種類に応じた専用のプロパティ設定ウィザードが表示されます。このウィザードを CPS (Custom Property Sheet: カスタム プロパティ シート) と呼んでいます。

DB コンポーネントの CPS は、次のステップによってプロパティ設定を行います。ステップの途中では、クエリ ビルダーなどの専用ツール (画面) を CPS から呼び出します。

(CPS)

1. JDBC ドライバーの指定 (テスト機能による設定の確認)
2. インターアクションのコンフィグレーション (クエリ設定画面を呼び出して、データベースに対するアクションを設定します)
3. **(クエリ設定画面)**
   - ① SQL ステートメントの選択とクエリ ビルダー画面の表示

     (クエリ ビルダー)

     i SQL ステートメントの作成
   - ② クエリ ビルダーで作成した SQL ステートメントのテスト実行
   - ③ スケジューラを使用するか否かの指定
   - ④ アドバンスト プロパティの設定
4. DB コンポーネントのテスト実行
5. スケジューラの設定
6. トランスポート レイヤーにおける JMS タイプの指定 (スケジューラを使用する場合のみ)
7. エラー発生時の処理方法の設定

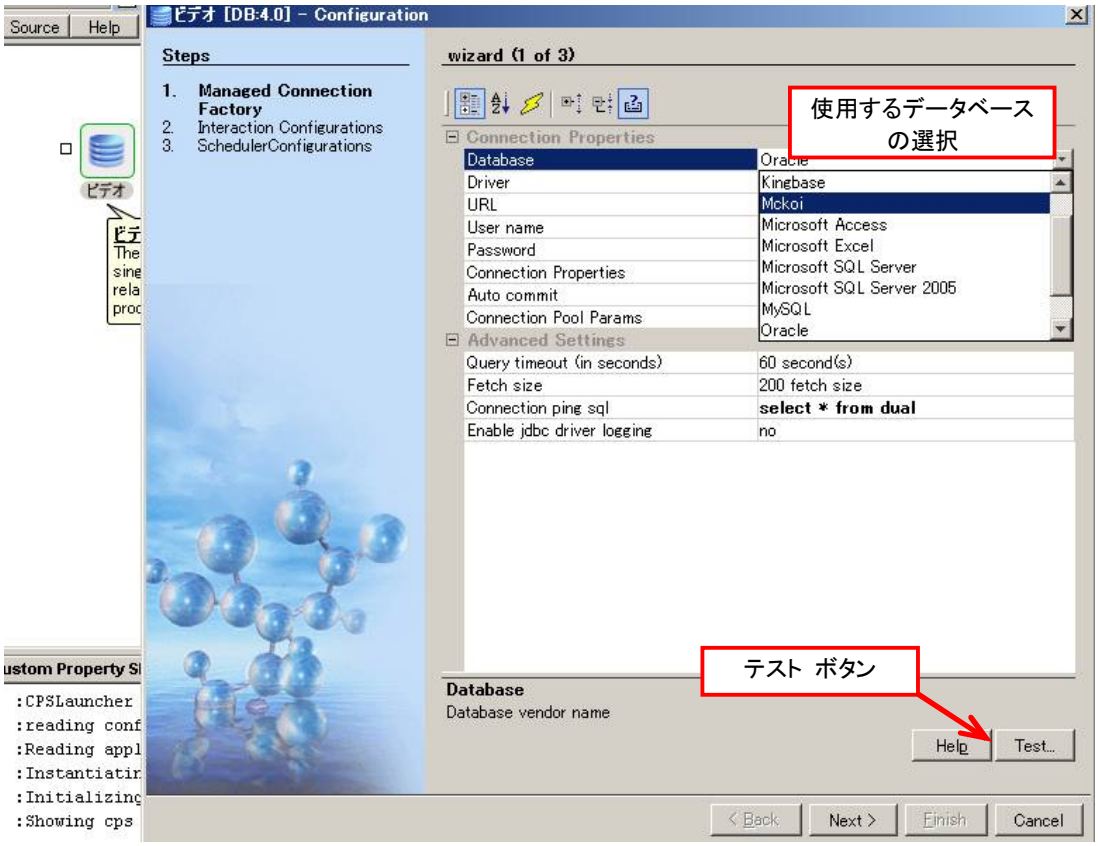
以降、[ビデオ] コンポーネントのプロパティ設定を上記ステップの順に説明します。

1 JDBC ドライバーの指定

次のキャプチャ画面は、DB コンポーネントの CPS の最初の画面です。

この最初の画面では、JDBC ドライバーの設定を行います

1) パラメータ **[Database]** に、使用するデータベースをプルダウンメニューから選択して指定します。この例題では、Mckoi を選択します。



2) その他のパラメータを、次のように指定します。

Driver : com.mckoi.JDBCDriver

URL : jdbc:mckoi: （最後に : が必要なことに注意してください）

User : mckoiuser

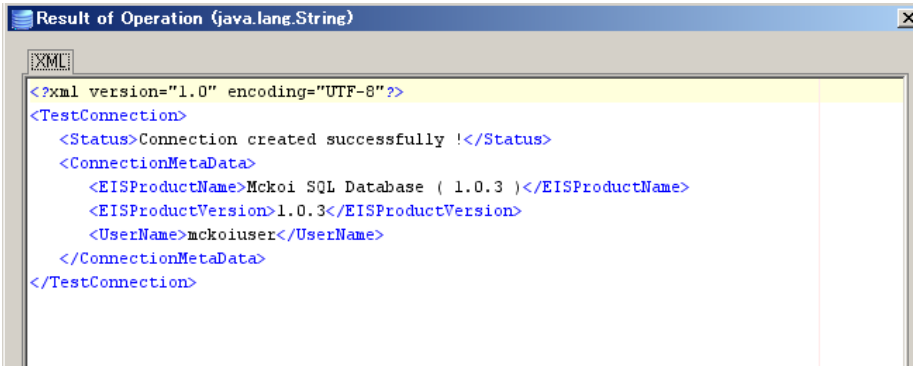
password : passwd

上記以外のパラメータは、この例題ではデフォルト値のままでかまいません。

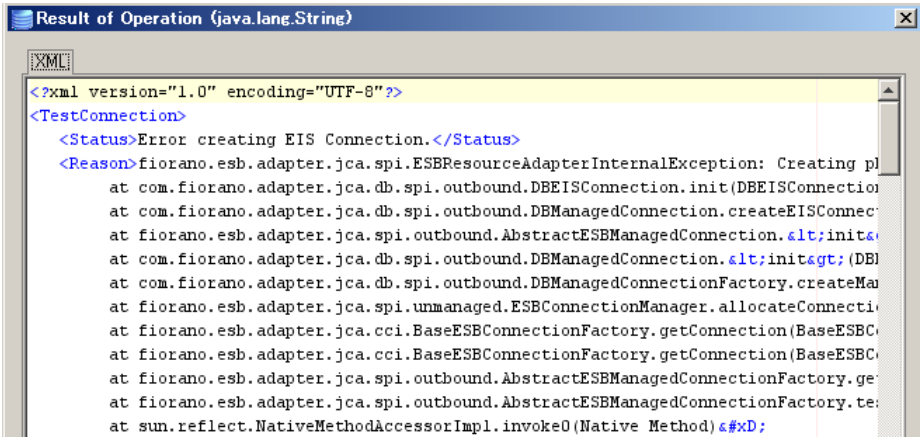
3) JBBC ドライバーの設定が完了しましたので、正しく設定されているかテストします。

右下にある **[Test]** ボタンをクリックします。

Mckoi データベースの JDBC ドライバーの設定が正しければ、次の画面が表示されます。



次の画面ようにエラー表示される場合には、JDBC の設定値が間違っているか、Mckoi データベースが起動していない可能性があります。設定値の見直し、Mckoi データベースの起動などを行って、エラーを解消します。



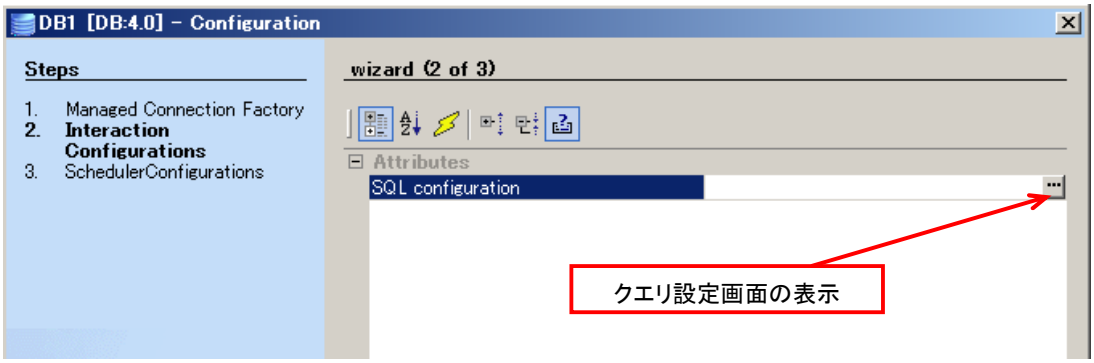
テストが正常に完了したら、テスト結果のウィンドウを閉じ、次のステップに進みます。

CPS の右下にある **[Next]** ボタンで次の設定画面（インターアクションのコンフィグ）に進みます。

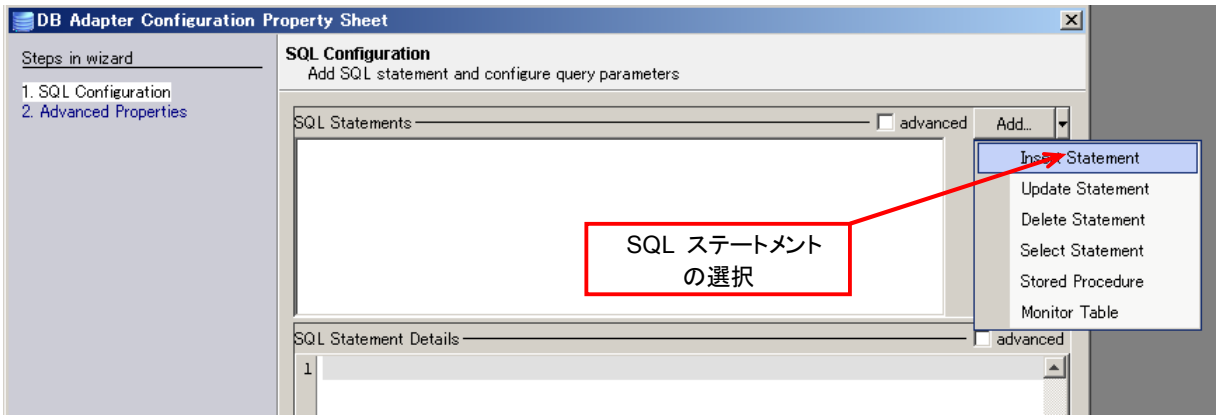
2. インターアクアションのコンフィグレーション

次のキャプチャ画面は、CPS の 2 番目のステップであるインターアクションの設定です。DB コンポーネントにおけるインターアクションとは、データベースに対するアクションを設定することであり、具体的には SQL クエリを作成することです。

クエリの設定は、クエリの設定画面で行います。CPS 画面の **[SQL Configuration]** パラメータの右側のボタンをクリックすると、クエリの設定画面が表示されます。



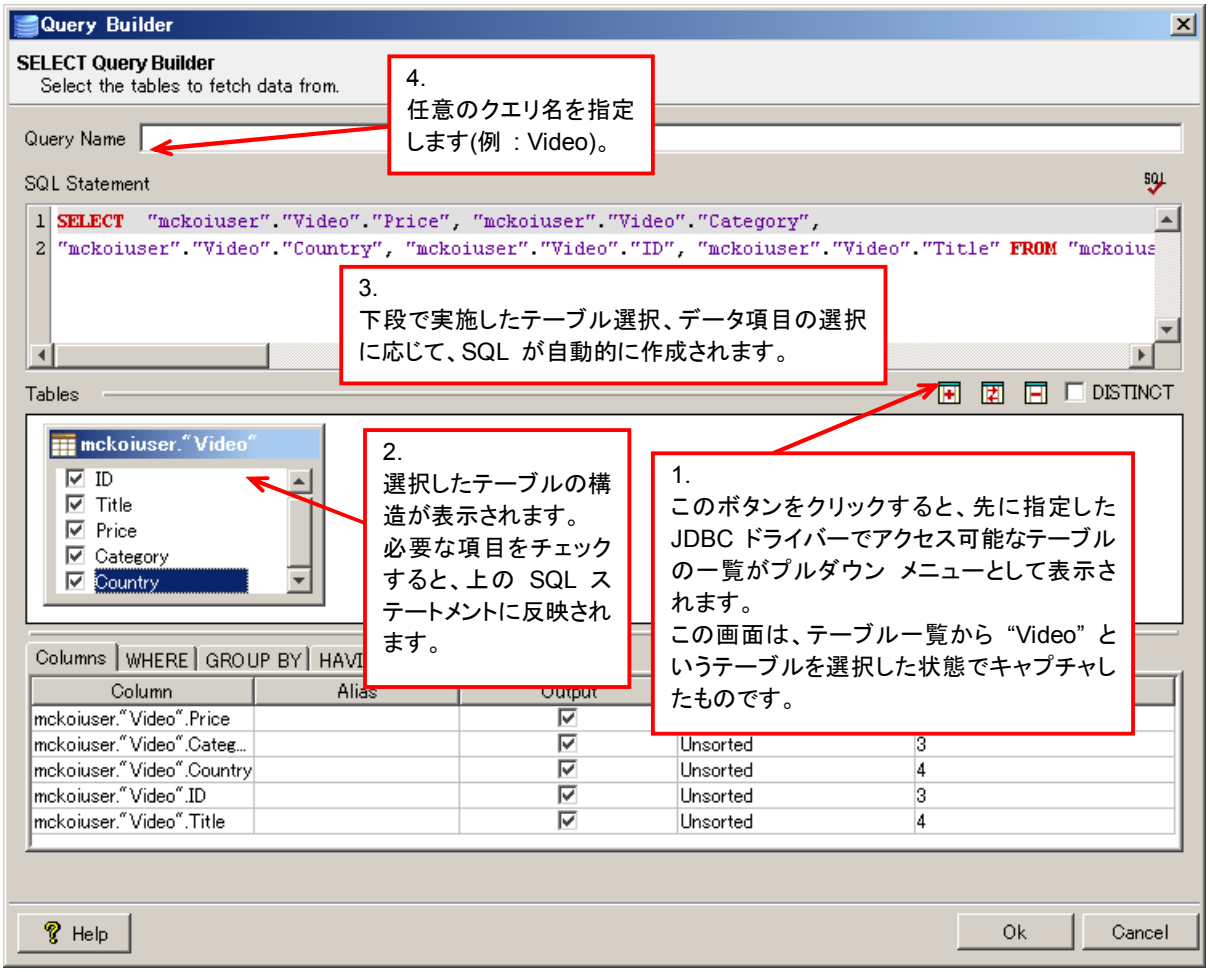
3 – ① クエリの設定画面 (DB Adapter Configuration Property Sheet)



[Add...] と表示されているボタンの右側をクリックすると、SQL ステートメントの一覧がプルダウンメニューとして表示されます。実行するステートメントを選択します。ステートメントを選択すると、選択したステートメントに応じた Query Builder (クエリ ビルダー) 画面が表示されます。今回の例題では、[Select Statement] を選択します。SELECT ステートメントの Query Builder が立ち上がります。

3 – ① – i クエリ ビルダー

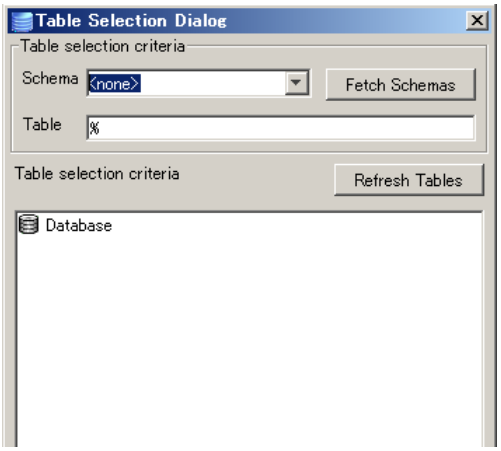
下図は、SELECT ステートメント用の Query Builder 画面において、クエリ対象のテーブルを指定した状態をキャプチャしたものです。SQL ステートメントをマニュアルで記述することもできますが、このツールでは実際のデータベース内のテーブルを参照しながらマウス操作のみで SQL ステートメントを作成することができます。



以下に示す SQL ステートメント作成ステップは、上図内の番号と対応しています。番号順に実際に操作してください。

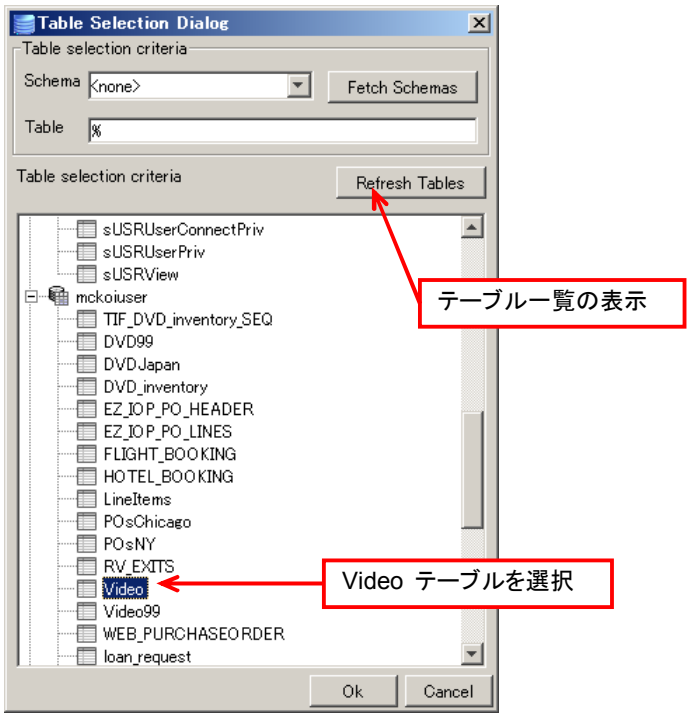
1.  ボタンをクリックし、テーブルの一覧を表示。表示された一覧の中からアクセスするテーブルを選択

次のウィンドウが表示されます。



[Refresh Tables] ボタンをクリックすると、最初に指定したユーザー ID でアクセス可能なテーブルが表示されます。

ユーザー `ID mckoiuser` 下にツリー表示されている **Video** を選択します。



2. 選択したテーブルの構造が表示されるので、SELECT するデータ項目を選択
3. ステップ 1.およびステップ 2. の選択に対応した SELECT ステートメントが自動的に作成される
4. クエリ名の指定 – クエリ名は、SELECT の結果をアウトプット メッセージ (XML 形式) として出力する際に、親エレメントの名前として使用されます。詳細は、『3.7 ポートにおけるデータ スキーマの確認』を参照してください。クエリ名には、日本語を用いることもできます。

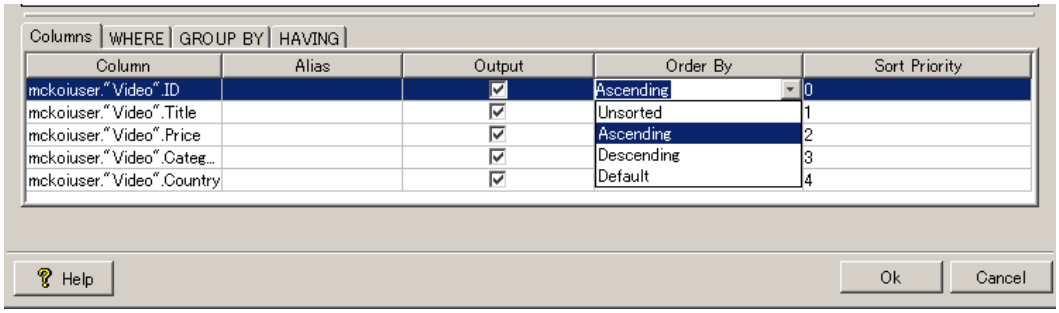
[OK] ボタンをクリックし、**Query Builder** を終了します。

補足説明

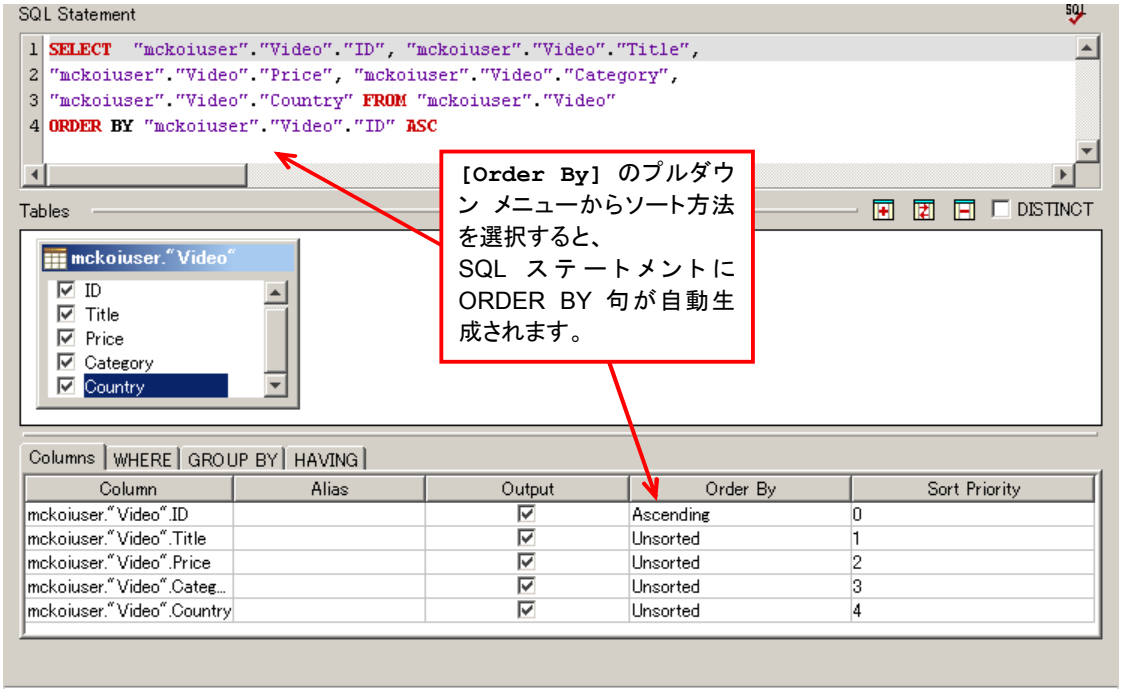
SELECT ステートメントに ORDER BY や WHERE 句を付加する場合には、クエリ ビルダーの下段にあるタブを使用します。(これらのタブは、SELECT ステートメントのクエリ ビルダー特有のものです。)

ORDER BY 句

[Columns] タブの **[Order By]** のプルダウン メニューからソートの種類を選択します。

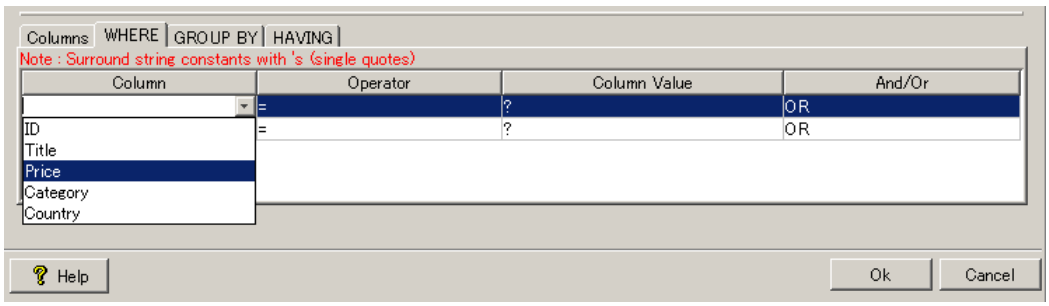


次の画面で示すように、ORDER BY 句が生成されます。

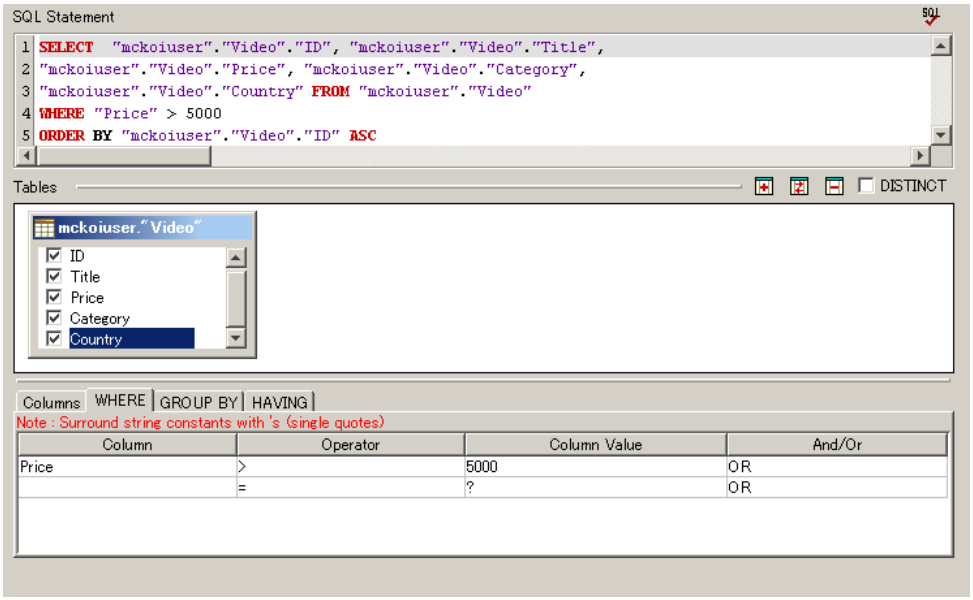


WHERE 句

WHERE 句を作成する場合には、**[WHERE]** タブをクリックします。**[Column]**、**[Operator]** のエリアにプルダウン メニューから必要な値を選択して入力します。**[Column Value]** には、値を入力します。



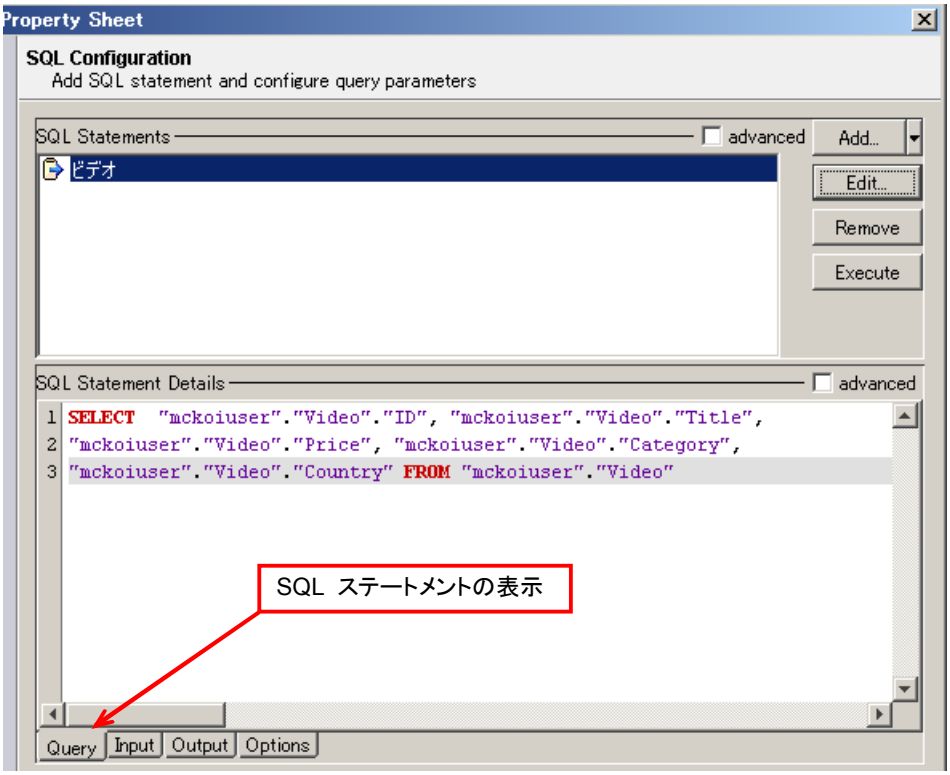
例えば、WHERE “Price” > 5000 としたい場合には、下の画面のように入力します。



(補足説明 – ここまで)

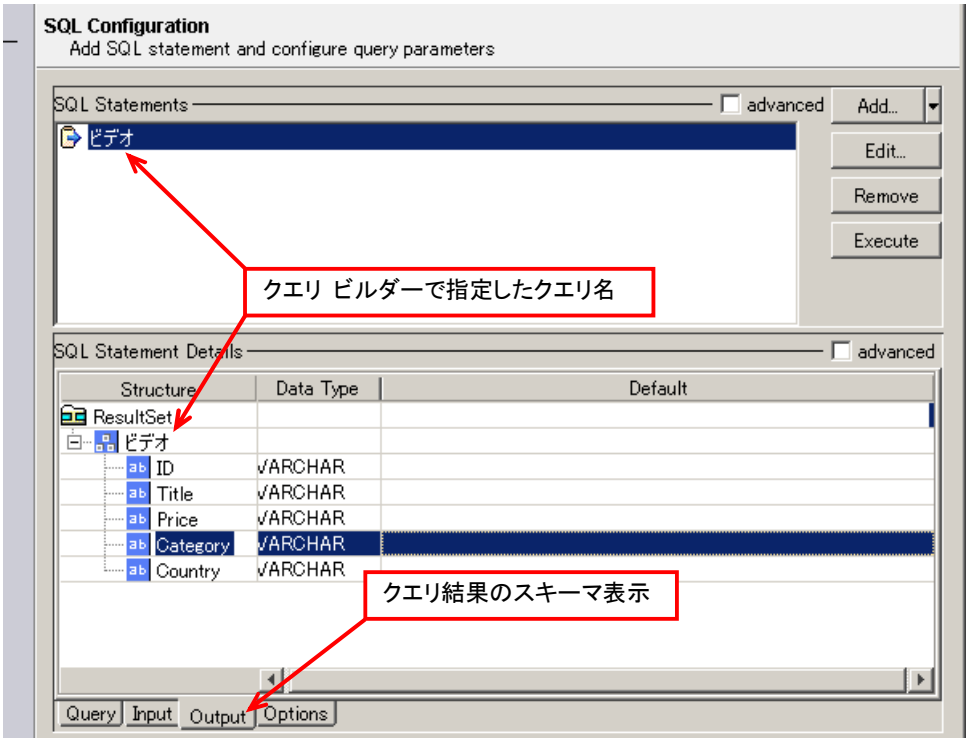
3 – ② 設定の確認とクエリのテスト

Query Builder（クエリ ビルダー）画面 で [OK] ボタンをクリックすると、クエリ設定画面に自動的に戻ります。Query Builder で設定した内容が確認できます。



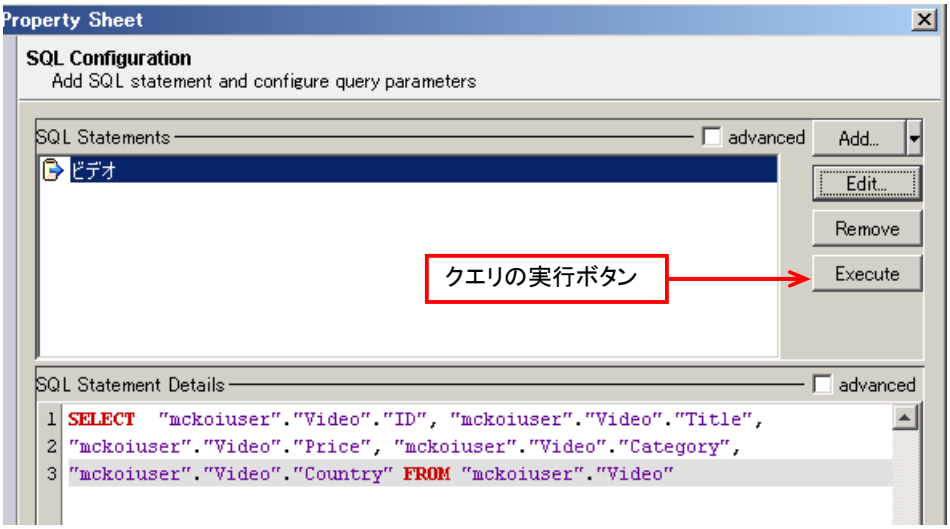
クエリ結果のデータ スキーマ

タブ [Output] をクリックすると、クエリ結果のスキーマ構造を表示することができます。



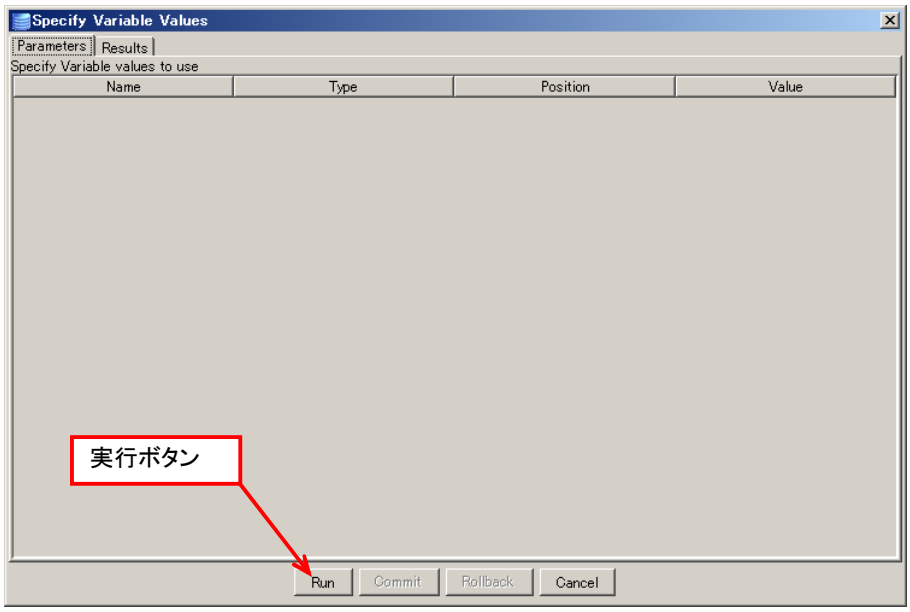
クエリのテスト実行

この段階で、作成したクエリをテスト実行することができます。**[Execute]** ボタンをクリックします。

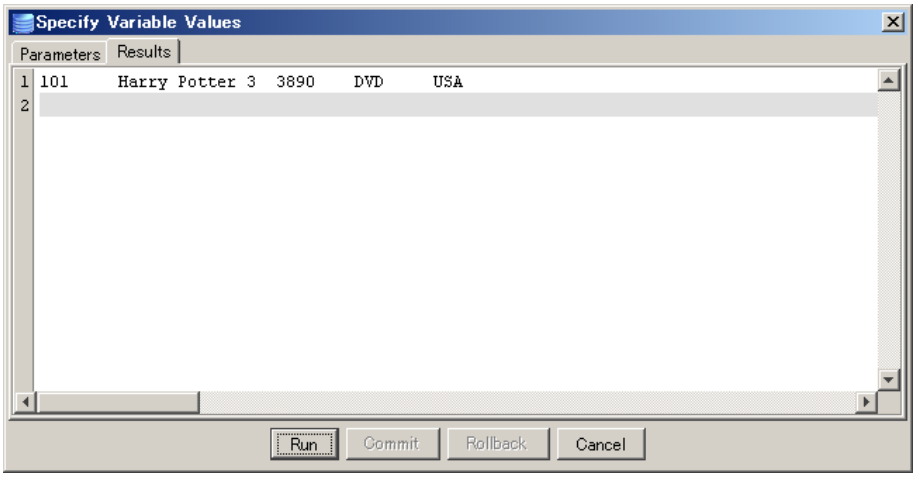


次の画面が表示されます。

[Run] ボタンをクリックして、作成した SQL ステートメントをテスト実行します。

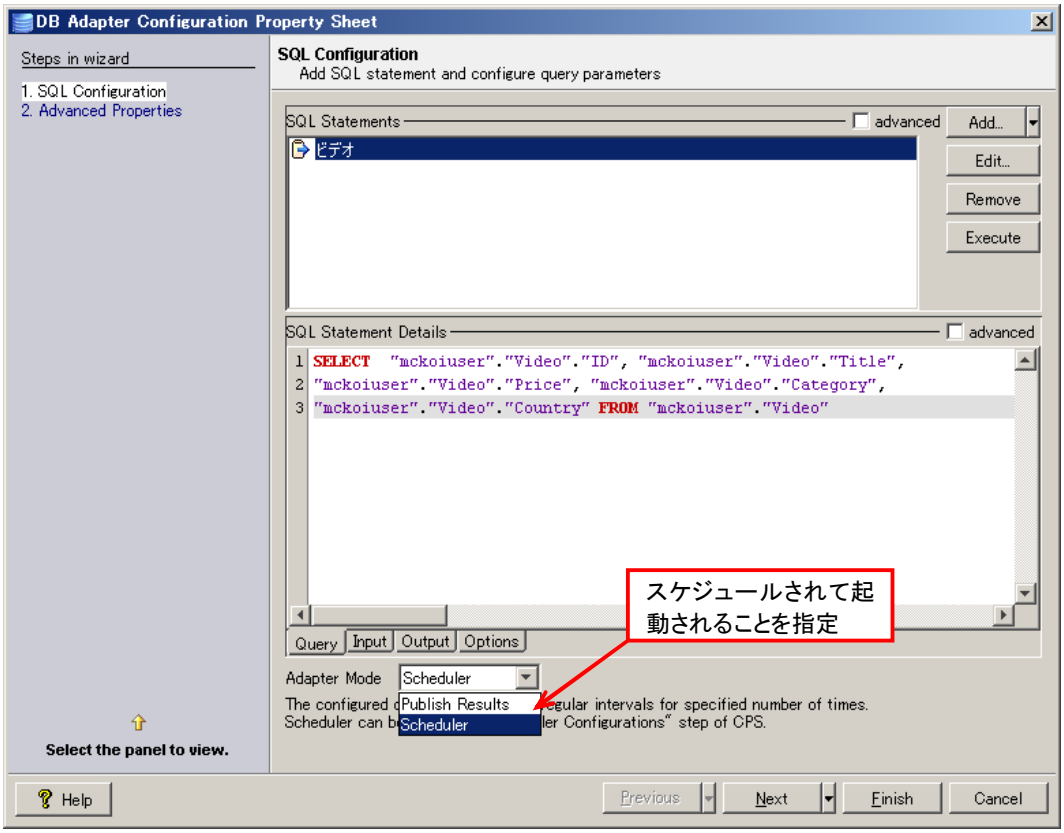


次の画面のように **[Result]** タグに実行結果が表示され、作成したクエリが正しいか否か確認できます。



3 – ③ スケジューラ使用の指定

クエリ設定画面の下段にある、**[Adapter Mode]** を **Scheduler** に設定します。この指定によって、このコンポーネントは自動的にポーリングを行うようになります。また、**Scheduler** を指定すると、インプット ポートは使えなくなります。イーゼル上のインプット ポートも消失します。



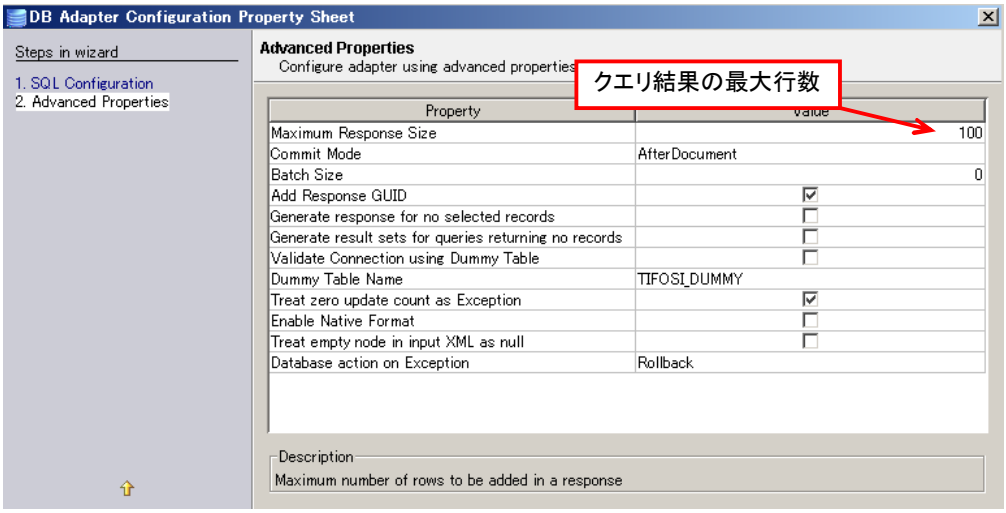
[Next] ボタンをクリックして次の設定に進みます。

3 – ④ アドバンスト プロパティの設定

データベースとのインタークションに関する詳細プロパティの設定です。

[Maximum Response Size] を除き、通常、デフォルト設定のままとします。**[Maximum Response Size]** は、クエリ結果の最大レコード数を指定します。SELECT の結果が 100 レコードを超える可能性がある場合には、この値を増やしておきます。**[Value]** 欄をクリックすれば、値を入力できます。

アドバンス プロパティ設定の他のパラメータの詳細については、製品マニュアルを参照してください。

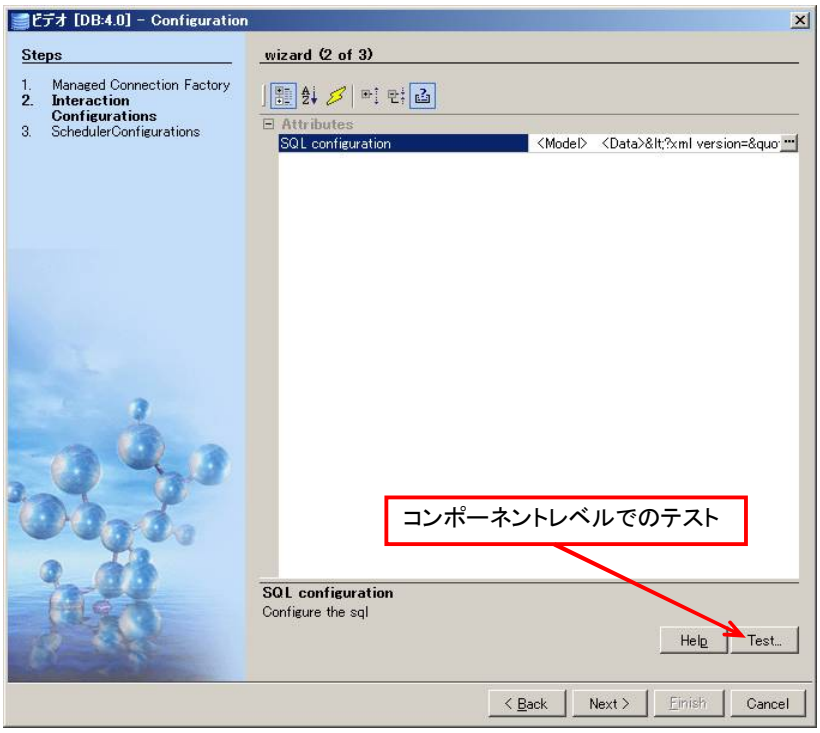


クエリ設定はこれで完了ですので、**[Finish]** ボタンをクリックしてクエリ設定を終了します。

4. コンポーネント レベルでのテスト

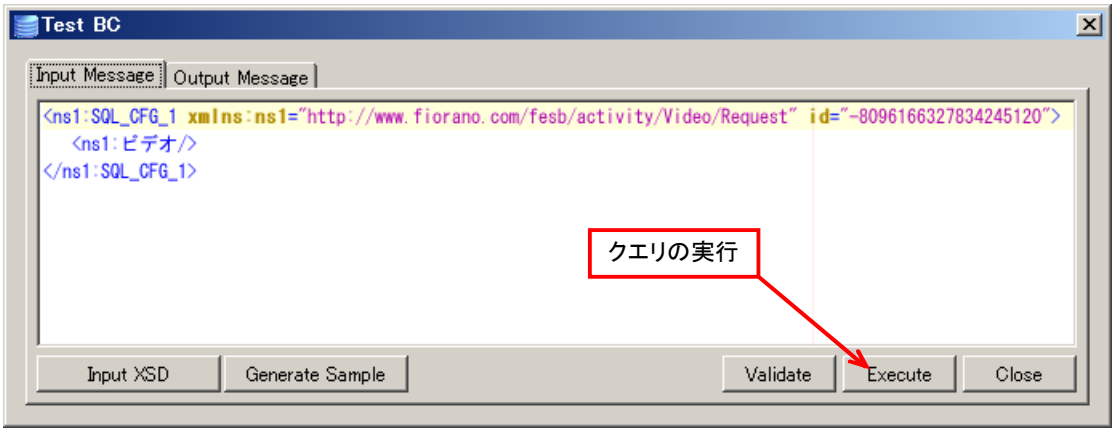
クエリ設定を終了すると、CPS の 2 番目のステップの画面（クエリ コンフィグレーション）に戻ってきます。クエリが設定されていることが確認できます。

この段階でも、作成したクエリをテストすることが可能です。**[Test]** ボタンをクリックしてください。

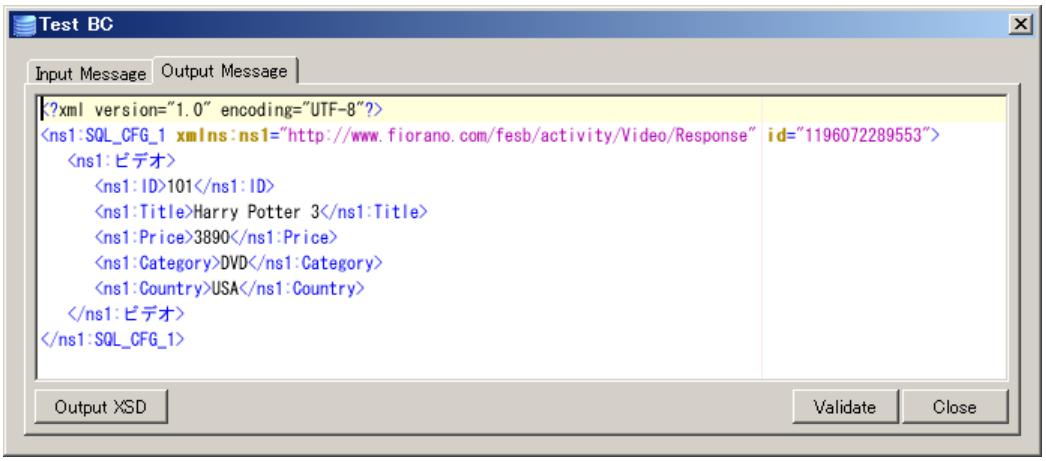


[Test] ボタンをクリックすると次の画面が表示されます。

右下の **[Execute]** ボタンをクリックすると、クエリを実行します。



実行結果が **[Output Message]** タブに表示されますので、意図した通りの結果が得られるか否か確認します。



前述のクエリ設定画面におけるテスト実行との違いは、次のようになります。

クエリ設定画面でのテスト（**[Execute]** ボタン）：クエリの実行結果（JDBC ドライバーから返される結果）を表示

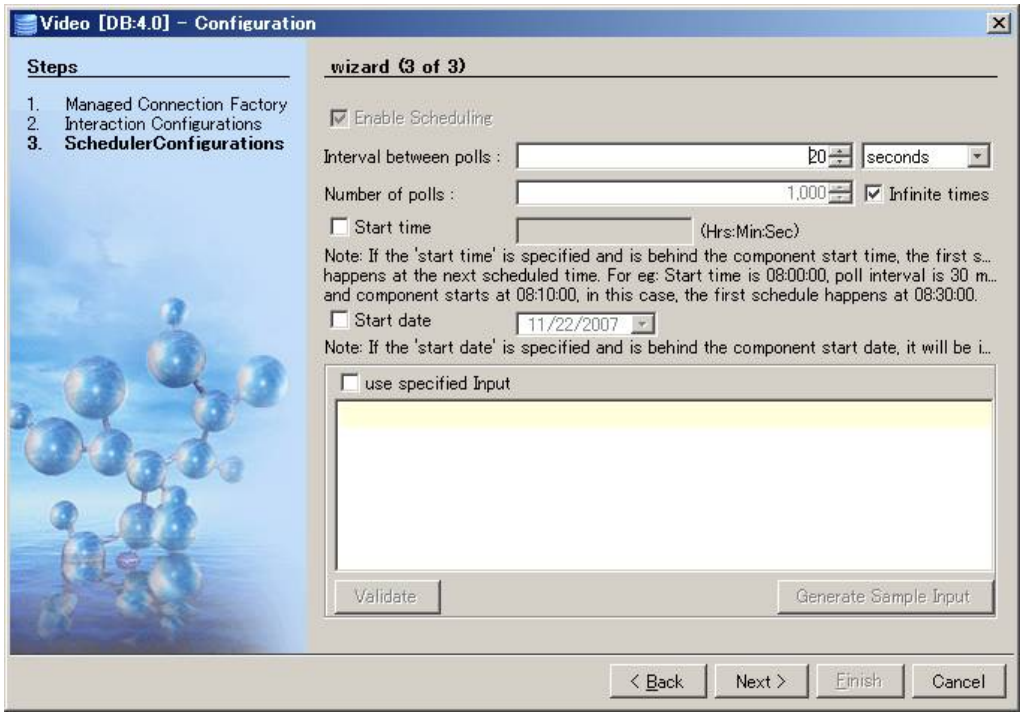
CPS 画面におけるテスト（**[Test]** ボタン）：DB コンポーネントのアウトプット ポートに送信されるデータを表示

なお、[ビデオ 99] コンポーネントのテスト方法のセクションでは、インプット データの扱いについて説明します。

[Close] ボタンをクリックして、クエリのテスト実行を終了します。

5. ポーリング間隔の設定

[Next] ボタンをクリックし、3 番目の設定項目であるスケジューラの設定画面に進みます。



この例題では、各パラメータ値を次のように指定します。

Interval between polls ：20 seconds (データベースにアクセスし、クエリを実行する間隔)

Infinite times： チェック オン

Start time： 指定しない (データベース アクセス (ポーリング) の開始時刻)

Start date： 指定しない (データベース アクセス (ポーリング) の開始日時)

上の設定値は、コンポーネント フローの実行直後から 20 秒間隔でポーリングすることを意味しています。

例えば、1日 1 回 9:00 にデータベースにアクセスするように設定する場合は、各パラメータ値は次のように指定します。

Interval between polls : 1 days

Infinite times： チェック オン

Start time : 9:00

Start date : 指定しない

[Next] ボタンをクリックして次の設定に進みます。

[注意] パラメータ Number of Polls は、Infinite times のチェックがオフ時に有効なパラメータで、全体で何回ポーリングするかを指定します。この回数に達すると、コンポーネントは実行を停止します。

例えば、20 秒間隔でポーリングするように設定し、Infinite times を 10 と設定すると、20 秒間隔でポーリングを 10 回おこなった後 (DB に 10 回アクセスした後) にこのコンポーネントは実行を停止します。

6. トランスポート コンフィグレーションの設定

4 番目の設定項目は、トランスポート コンフィグレーションの設定です。トランスポート コンフィグレーションの設定ページは、スケジューラを使用する場合にのみ表示されます。上述のステップ 2 (クエリの指定) でスケジューリングするよう指定しているため、トランスポート コンフィグレーションの設定ページが CPS に追加され、表示されます。

サービス コンポーネント間のトランスポート プロトコルは、JMS (Java メッセージ サービス) を使用しています。ここで指定するのは、JMS セッションのタイプ (トランザクション モードか acknowledge モードか)、トランザクションのメッセージ数などです。通常、これらの指定は、インプット ポートのプロパティ設定で行います。スケジューラ指定を行うとインプット ポートが消失しますので、代わりに CPS で設定するように変更されるのです。

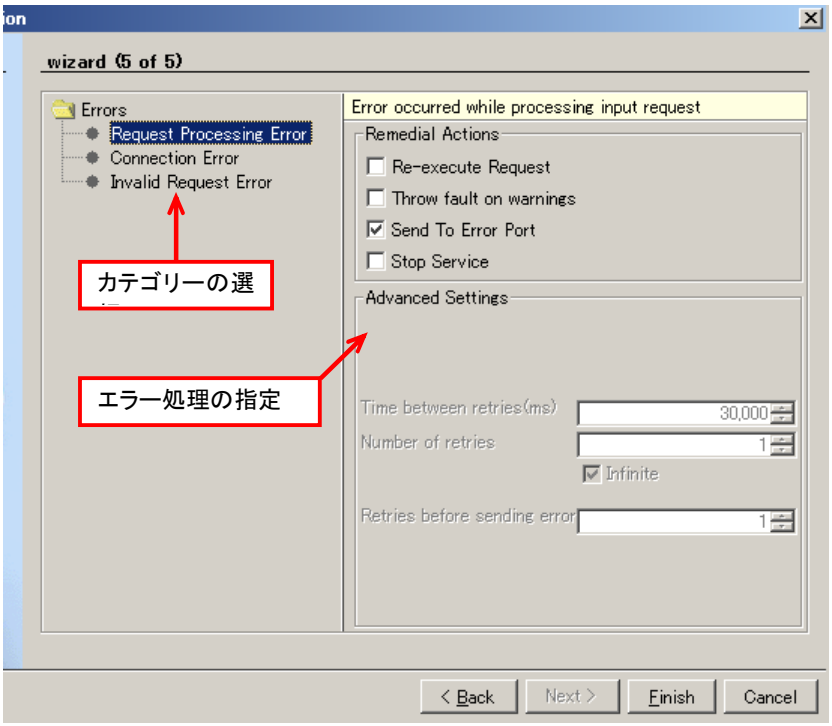
この例では、デフォルト設定のままとします。何も変更しないで **[Next]** ボタンをクリックし、次のエラー処理の設定に進んでください。

トランスポート コンフィグレーションの詳細については、製品マニュアルを参照してください。

7. エラー発生時の処理の設定

最後のプロパティは、エラー発生時の処理の設定です。

エラー処理設定の画面



エラー発生時の処理の設定項目は、どのコンポーネントにも共通しており、次の 3 つのカテゴリー毎に設定します。

- 受信データの処理中に発生したエラー (Request Processing Error)
- コネクション エラー (Connection Error)
- 入力データの形式エラー (Invalid Request Error)

(上の画面は、Request Processing Error の設定画面です)

各カテゴリー別に指定できるエラー処理およびデフォルト設定を、下の表にまとめました。エラー処理は重複して指定できます。例えば、受信データの処理中に発生したエラーに対して、エラー ポートからエラー メッセージを送信 (デフォルトの処理) とコンポーネントの停止を、同時に指定できます。

エラー処理設定項目の一覧

エラーのカテゴリー	選択できる処理	デフォルト設定
Request Processing Error **受信データの処理エラー** (受信データの処理中に例外が発生)	リトライ	
	ワーニングを throw	
	エラー ポートからエラー メッセージを送信	**デフォルト**
	コンポーネントを停止	
Connection Error **コネクション エラー** (処理中に各リソース (ライブラリ、ピア サーバー、データベースなど) への接続がロスト	再接続の試行	
	エラー ポートからエラー メッセージを送信	**デフォルト**
	コンポーネントを停止	
Invalid Request Error **入力データの形式エラー** (誤った形式のデータを入力)	エラー ポートからエラー メッセージを送信	**デフォルト**
	コンポーネントを停止しない	**デフォルト**

なお、この例題では、エラー処理を各カテゴリーともデフォルト設定のままとします。

8. CPS の終了

[ビデオ] コンポーネントのプロパティ設定が完了しましたので、**[Finish]** ボタンをクリックし、CPS を終了します。

3.7 [ビデオ 99] コンポーネントのプロパティ設定

[ビデオ 99] コンポーネントのプロパティ設定を、下記のように行います。

(1) JDBC ドライバー

[ビデオ] コンポーネントと全く同じ値を指定します

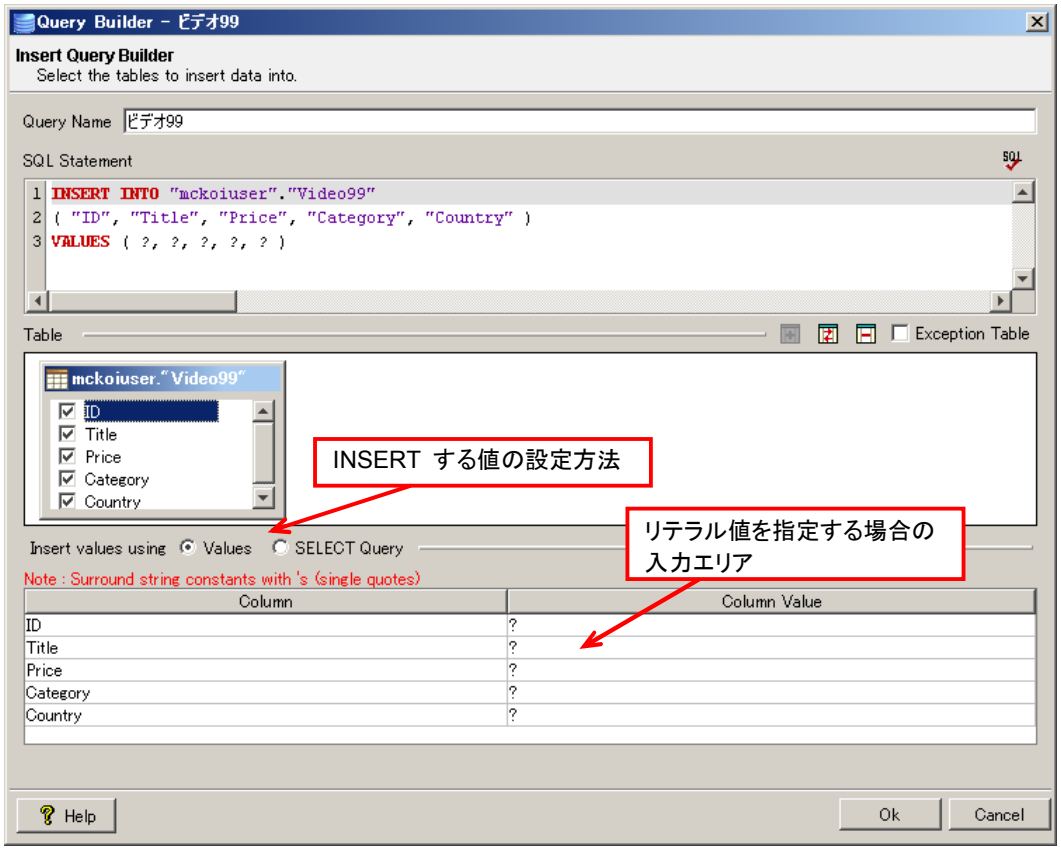
(2) クエリ設定

入力したデータを Video99 テーブルに INSERT するように設定します。

[ビデオ] コンポーネント同様に、CPS からクエリの設定画面 → INSERT を選択 → Query Builder と進んでください。

テーブルは、mckoiuser.Video99 を選択します。(設定ステップを画面図の後に記します。)

次の画面は、INSERT ステートメントの **Query Builder** (クエリ ビルダー) での設定結果を示しています。



INSERT ステートメントの **VALUES (?, ?, ?, ?, ?)** に挿入されるデータは、次の 3 つの方法から選択できます。

- インプット ポートから入力したデータ (この例題では、この挿入方法を用います)
- 別途作成した SELECT ステートメントによるリトリーブ データ
- このクエリ ビルダー画面で指定した値

以下の設定手順の説明中に詳細を記します。

クエリ ビルダーにおける設定手順

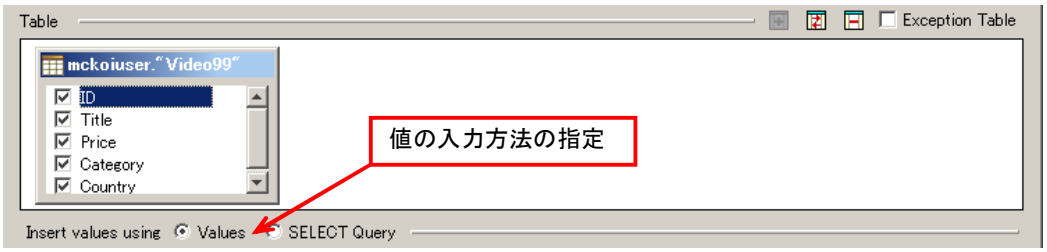
1.  ボタンをクリックして、テーブルの一覧を表示。表示された一覧の中からアクセスするテーブルを選択

2. INSERT するデータ項目を指定 (デフォルトではすべての項目が選択されています。この例題では、すべての項目に INSERT しますので、ツールが作成した INSERT ステートメントをそのまま使用します。)

3. INSERT する値の設定

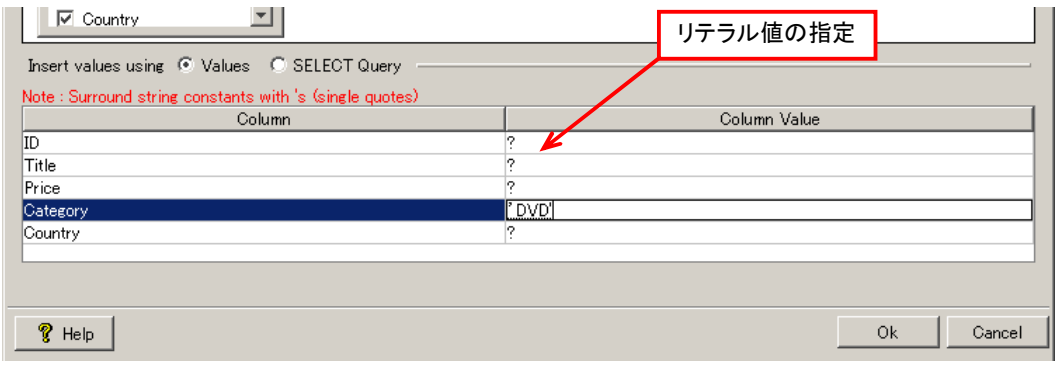
   次の画面で示すように、**[Insert values using]** の設定が **[Values]** (デフォルト値) になっていることを確認。**[Values]** を選択すると、コンポーネントのインプット ポートから入力された値を INSERT します。

   [SELECT Query] は、別途作成した SELECT ステートメントによってリトリーブした値を INSERT します。**[SELECT Query]** を選択すると、SELECT ステートメント作成用のクエリ ビルダーが自動的に立ち上がり、SELECT ステートメントの作成がうながされます。



4. リテラル値を INSERT する場合には、次の画面に示すように、値を指定します。**[Column Value]** 欄をダブル クリックすると値を入力できるようになります。値は、シングル クォートで囲ってください。

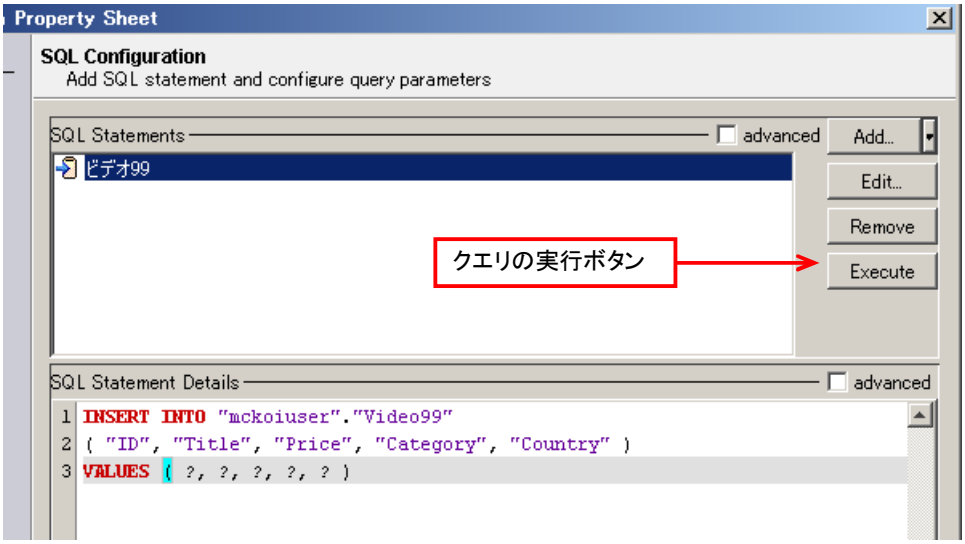
   この例題では、INSERT するデータ値はすべてインプット ポートから入力されますので、リテラル値の設定は行いません。



5. **[OK]** ボタンでクエリ ビルダーの設定を完了します。

(3) クエリ設定画面におけるテスト実行

次の画面は、クエリ ビルダーでの設定を完了して、クエリ設定画面に戻ってきた状態を示しています。この段階で、**[Execute]** ボタンをクリックして、作成したクエリをテストします。

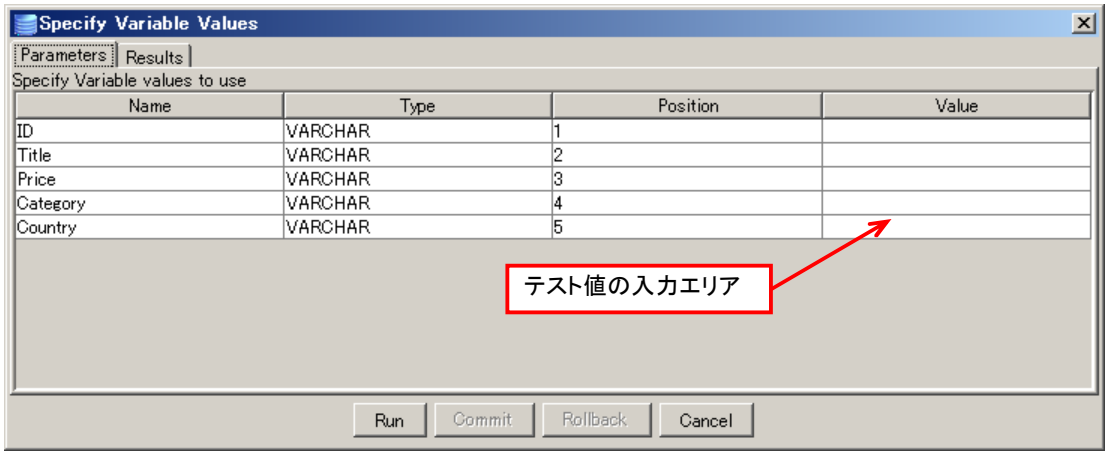


[Execute] ボタンをクリックすると、以下に示す **[Specify Variable Values]** というタイトルのウィンドウが表示されます。このウィンドウの **[Parameters]** タブは、クエリの実行に必要なインプット データを指定するための画面です。

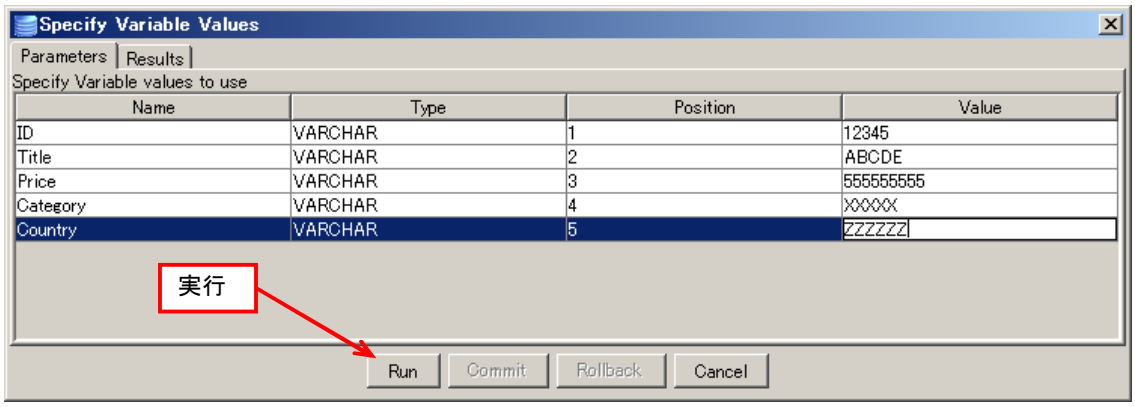
クエリ ビルダーで作成した INSERT ステートメントは、次のようになっています。

```
INSERT INTO "mckoiuser"."Video99"  ( "ID", "Title", "Price", "Category", "Country" )
VALUES (?, ?, ?, ?, ?)
```

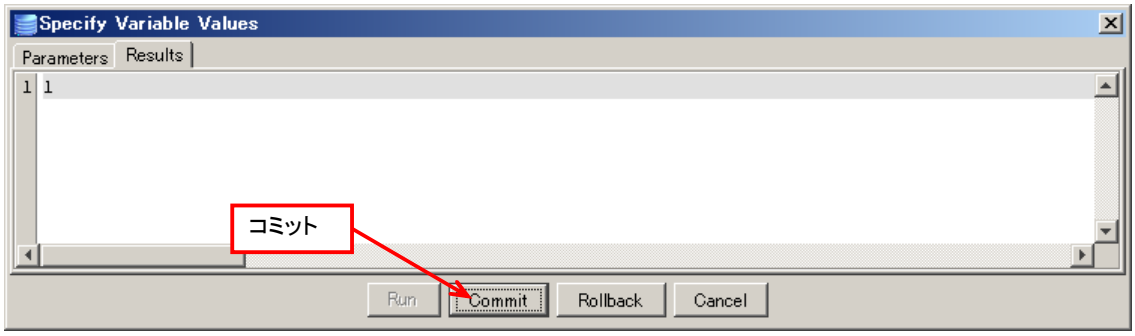
VALUES (?, ?, ?, ?, ?) への入力データ（テスト データ）を指定します。各行の **[Value]** カラムをクリックすることで、テスト値を入力できます。



テスト値を入力して、**[Run]** ボタンをクリックします。



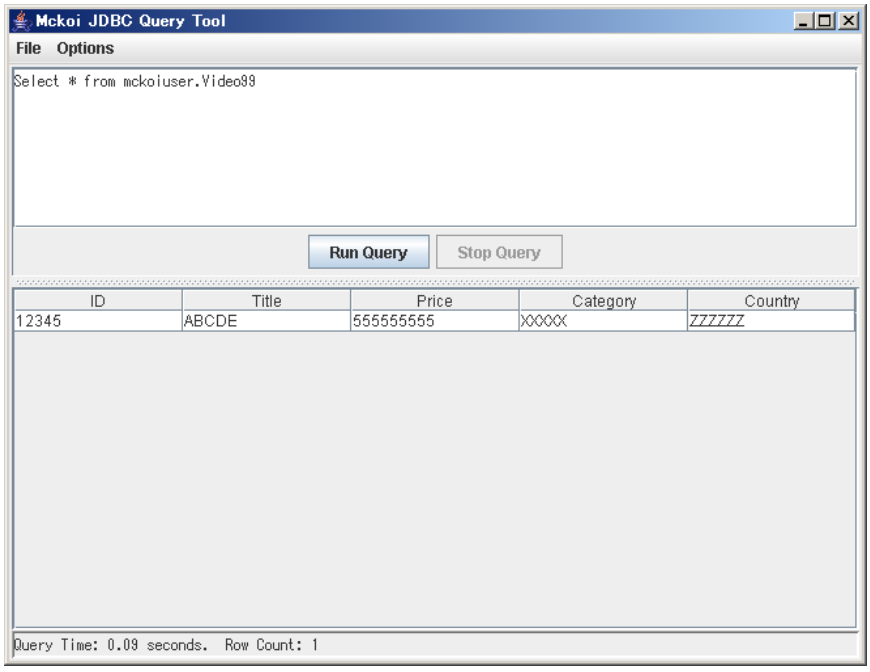
実行結果が表示されます。



クエリの実行は完了しましたが、これに対するコミット (Commit) もしくはロールバック (Rollback) をデータベースに対して送信する必要があります。**[Commit]** ボタンをクリックして DBMS にコミット処理をリクエストします。

データベースに実際に書き込まれた否かを確認するためには、QueryMckoi を使用します。QueryMckoi の起動方法および使用方法は、付録を参照してください。

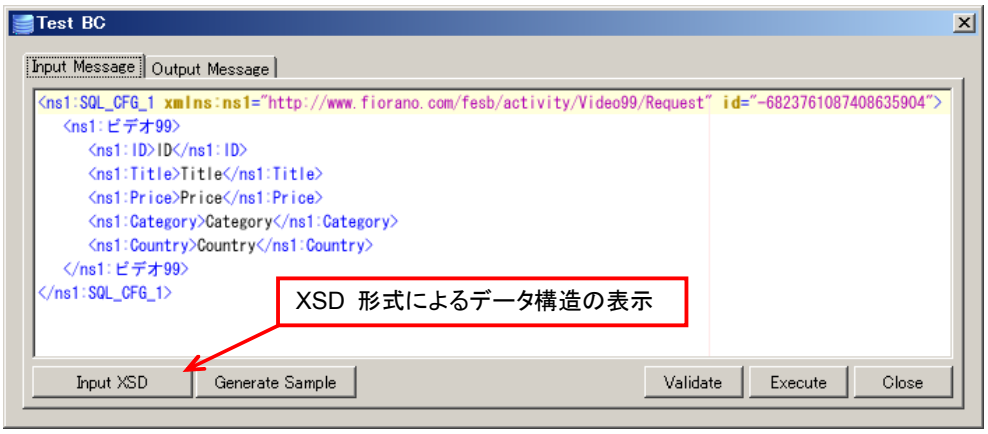
QueryMckoi で **`Select * from mckoiuser.Video99`** を実行した結果は次のようになります。テスト データが書き込まれていることが確認できます。



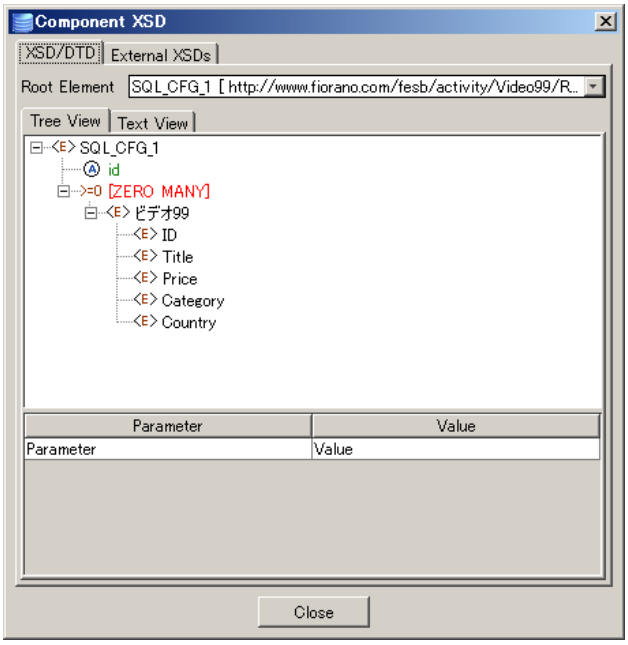
(4) コンポーネント レベルでのテスト

CPS 画面に戻り、**[Test]** ボタンをクリックします。

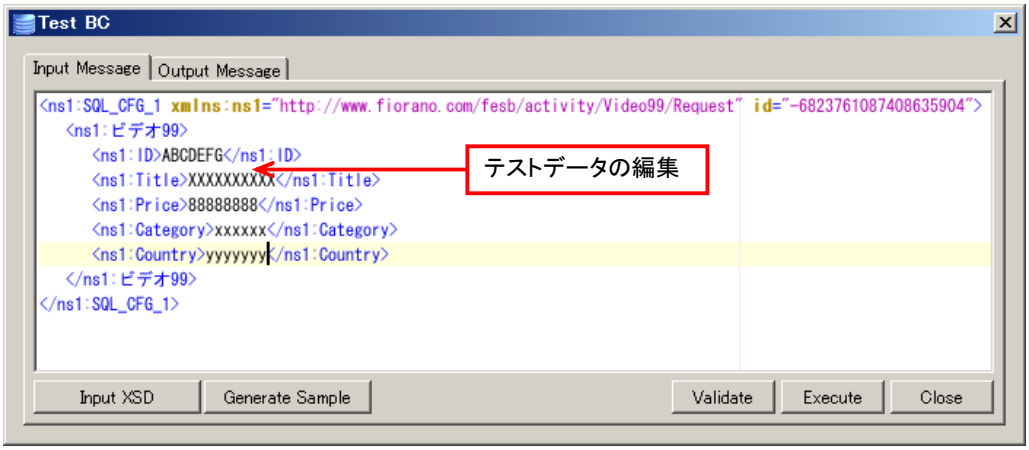
次の画面が表示されます。



この画面の **[Input Message]** タブは、インプット ポートの入力データをシミュレートするための画面です。**[Input XSD]** ボタンをクリックすると、インプット ポートで受信するデータの構造を、XSD 形式で表示します。**[Tree View]** と **[Text View]** の 2 種類があります。下のキャプチャ画面は、Tree View による表示を行ったものです。

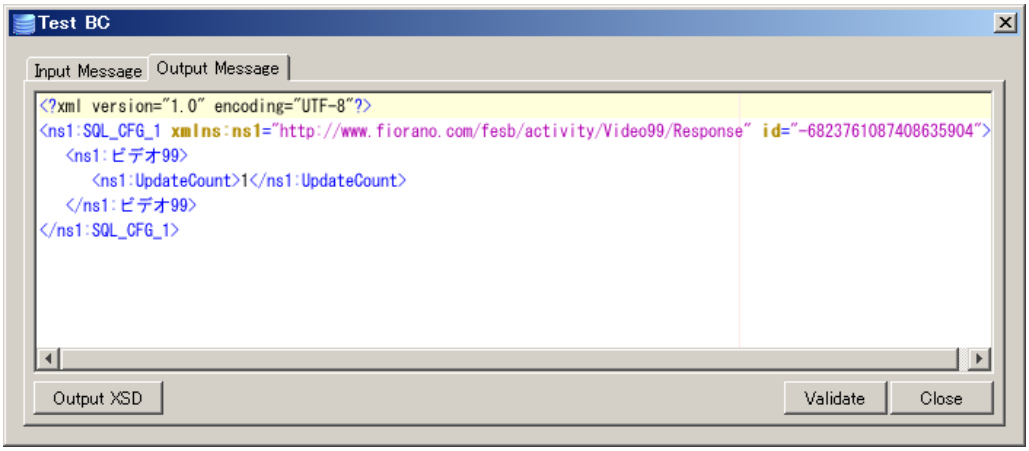


テスト データを入力するためには、XSD データ構造のウィンドウを閉じ、最初の画面で値を入力します。値の部分（黒字で表示されている部分）にマウス ポインターを合わせると、テキスト入力のカーソルが表示されます。元の値を削除し、テストする値を入力します。次の画面は、テスト値を入力した状態を示しています。



[Validate] ボタンで入力したデータのデータ タイプの整合性をチェックすることができます。

[Execute] ボタンで実行します。次の画面のように実行結果が表示されます。この画面は、アウトプット ポートから出力されるデータの構造となっています。



(5) スケジューラの設定は行いません。

[ビデオ] コンポーネントとは異なり、スケジューラとして動作させないため、トランスポート（JMS メッセージング）の設定ステップは追加されません。

(6) エラー処理はデフォルトのままとします。

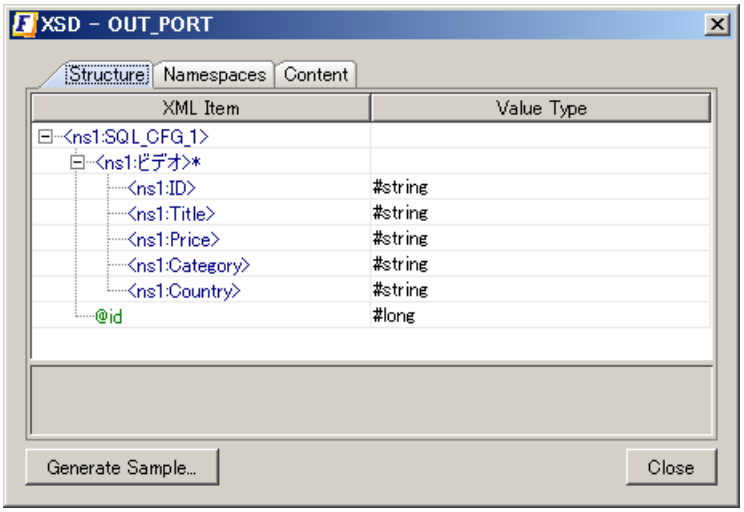
以上の 1 から 7 のプロパティ設定によって、2 つのコンポーネント（ビデオ と ビデオ 99）の設定が完了します。

3.8 ポートにおけるデータ スキーマの確認

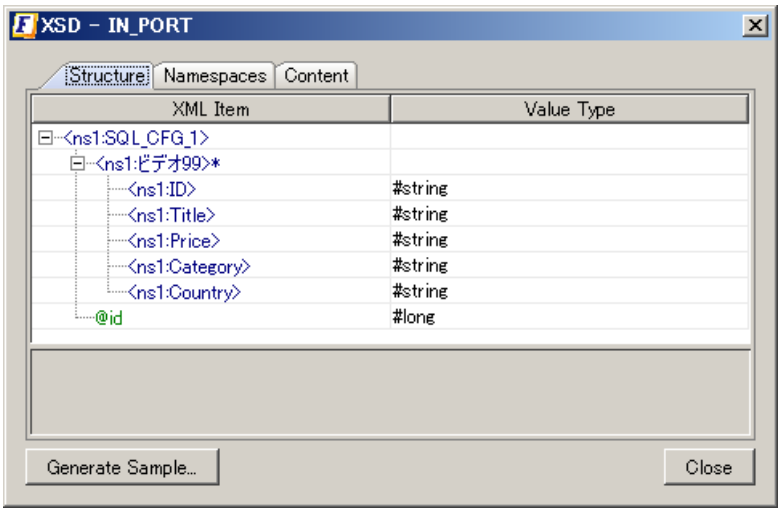
DB コンポーネントの CPS によるプロパティ設定が完了すると、設定に応じてインプット ポートおよびアウトプット ポートのデータ スキーマも自動的に決定されます。コンポーネントの種類によっては、CPS の設定内容に依存しないでデータ スキーマが決定されるものもあります。Fiorano がプリビルトして提供しているコンポーネントの多くは、CPS の設定内容に応じてポートのデータ スキーマが決定されるようになっています。

例題の [ビデオ] コンポーネントと [ビデオ 99] コンポーネントのデータ スキーマがどのように設定されているか見ることができます。[ビデオ] コンポーネントのアウトプット ポートを右クリックし、プルダウン メニューから **[View Schema Structure ...]** を選択します。

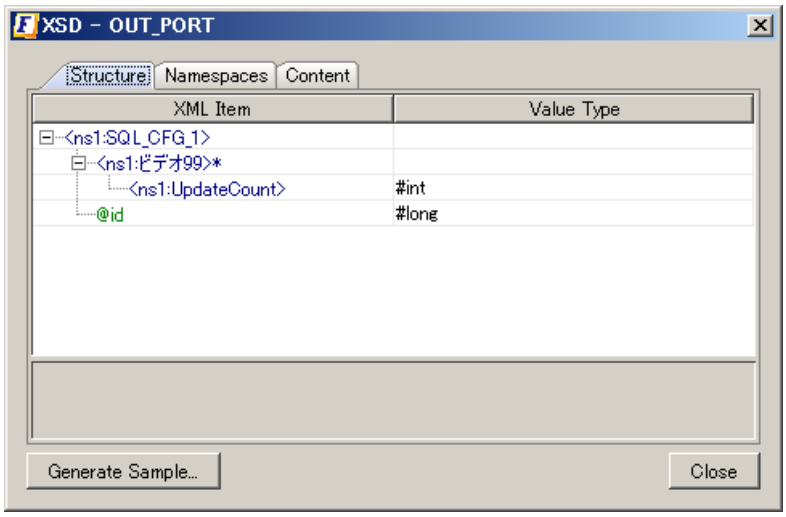
次のキャプチャ画面のようにウィンドウが表示されます。



次の画面は、[ビデオ 99] コンポーネントのインプット ポートのスキーマです。



[ビデオ 99] コンポーネントのアウトプット ポートのスキーマは、次のように表示されます。



上のスキーマ表示から、各ポートが次のように設定されていることがわかります。

[ビデオ] コンポーネント : インプット ポート --- スケジューラ指定を行ったため存在しない

アウトプット ポート -- SELECT ストートメントによってリトリーブするテーブル行を XML 形式に変換したスキーマ

[ビデオ 99] コンポーネント : インプット ポート – INSERT ステートメントによって挿入するテーブルの行を XML 形式に変換したスキーマ

アウトプット ポート – INSERT ステートメントの実行結果 (データベースからの戻り値) 通常のケースではインサートした行数

つまり、DB コンポーネントのクエリ ビルダーによって指定した SQL ステートメントに応じて、コンポーネントの各ポートのデータ スキーマが自動的に設定されるのです。

なお、ウィンドウの上部にあるタブは、それぞれ次の情報を表示します。

[Structure] : データ スキーマをツリー構造で表示

[Namespaces]: 使用しているネームスペースの表示

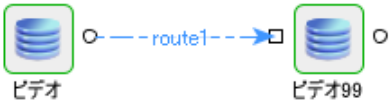
[content] : データスキーマを XML スキーマで表示

[Generate Samples …] ボタンは、このデータ スキーマに基づくサンプルの XML データを表示します。

3.9 ルートの設定

次に、サービス コンポーネント間のデータ転送経路を設定します。これは、コンポーネントのアウトプット ポートをクリックし、つなげるコンポーネントのインプット ポートへドラッグすることで行います。インプット ポートからアウトプット ポートへ逆方向にドラッグしてもルートは生成できません。

[ビデオ] コンポーネントのアウトプット ポートから [ビデオ 99] コンポーネントのインプット ポートへマウスをドラッグし、ルートを作成してみてください。



上図のようにルートが作成されます。

今作成されたルートを良く見ると、点線で表示されていることがわかります。これは、[ビデオ] コンポーネントのアウトプット データのデータ構造と、[ビデオ 99] コンポーネントのインプット データのデータ構造が異なっていることを示しています。これを解消するためには、この 2 つのデータ構造の間でマッピング (データ トランスfp-メンション) をする必要があります。

3.10 データ トランスフォーメンションの設定

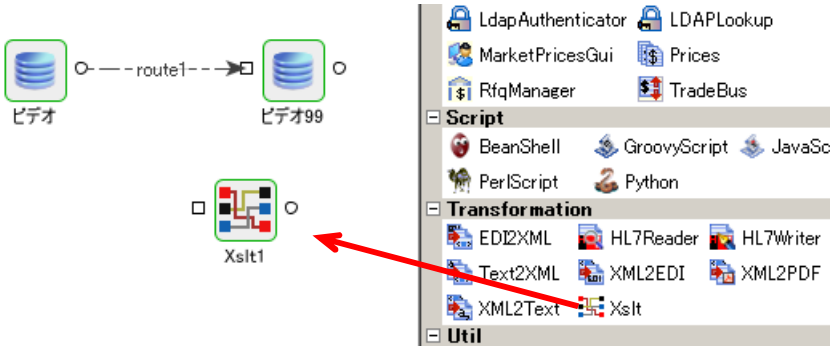
ポート間のマッピング指定には、Xslt コンポーネントを用います。

(旧バージョンで可能であった「ルート上のマッピング指定」は、2007 SP5 から廃止されています。廃止の理由は、ルート上でマッピング指定を行った場合、コンポーネント フロー上に明示的に表示されず、データフローやプロセス フローのメインテナンス性を低下させてしまうためです。このため、SP5 移行のバージョンでは、Xslt コンポーネントを用いる方法のみをサポートし、フロー上に明示的に表示させるようにしました。

なお、Xslt コンポーネントによる方法も、ルート上でマッピング指定を行う方法も、実行時には Fiorano マッパーによってデータ変換されますので、処理性能に差はありません。

1. Xslt コンポーネントのドラッグ&ドロップ

Xslt コンポーネントをパレットからイーゼルにドラッグ&ドロップします。Xslt コンポーネントは、パレットの `[Transformation]` カテゴリーにあります。



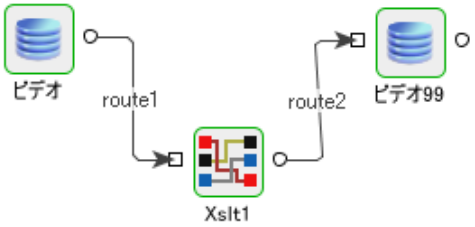
2. 既存ルートの削除

[ビデオ] コンポーネントと [ビデオ 99] コンポーネントの間にルートが設定されている場合には、ルートを削除します。

ルートを右クリックし、メニューから **[Edit]** → **[削除]** を選択すれば、ルートを削除できます。

3. 新規ルート作成

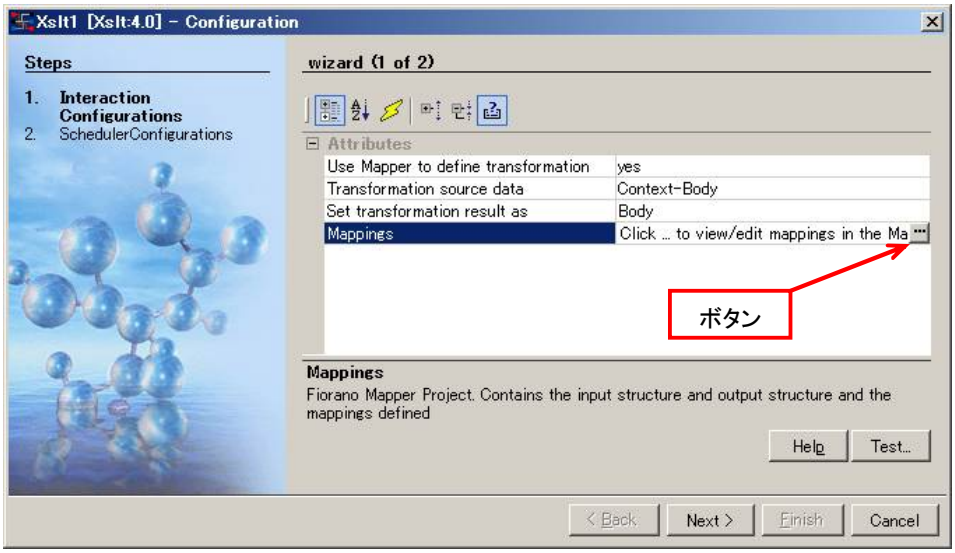
下図のように新しいルートを作成します。



4. Xslt コンポーネントの CPS の設定

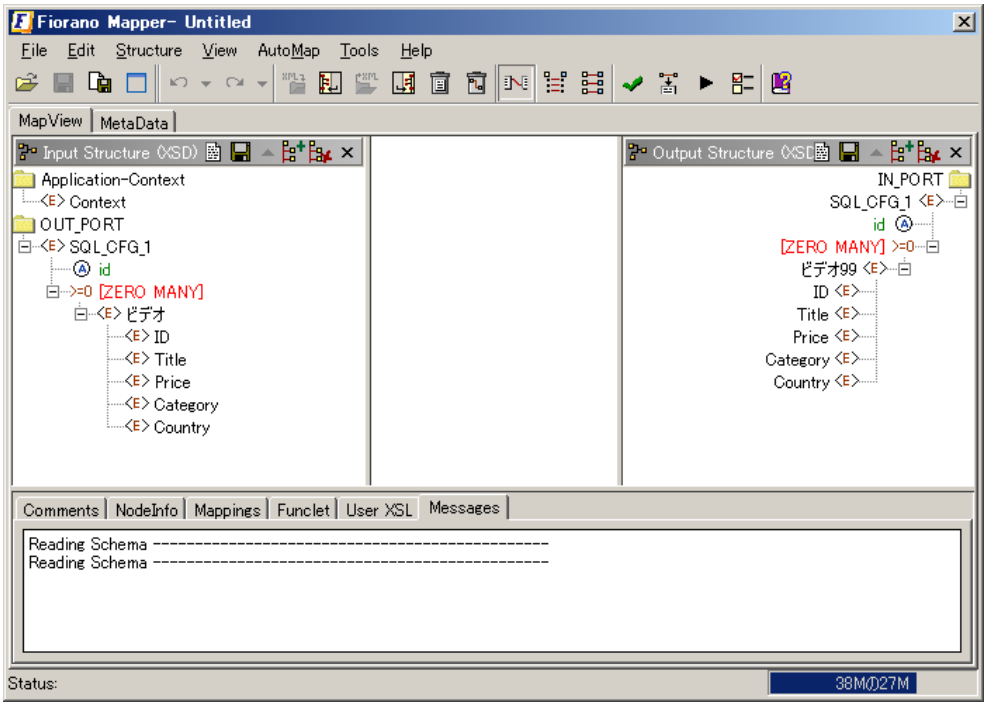
Xslt コンポーンネントをダブルクリックし、CPS を開きます。

次の画面は、Xslt コンポーネントの最初の CPS 画面です。



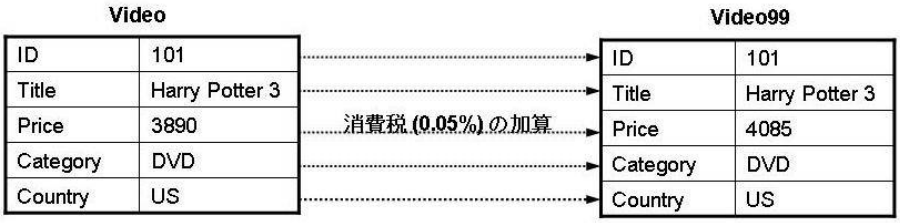
[Mappings] 項目の入力エリアをクリックし、右側の  ボタンをクリックすると、Fiorano マッパー ツールが起動します。

次の画面は、Fiorano マッパーが起動された状態を示しています。データ間の結びつけは何も指定されていません。

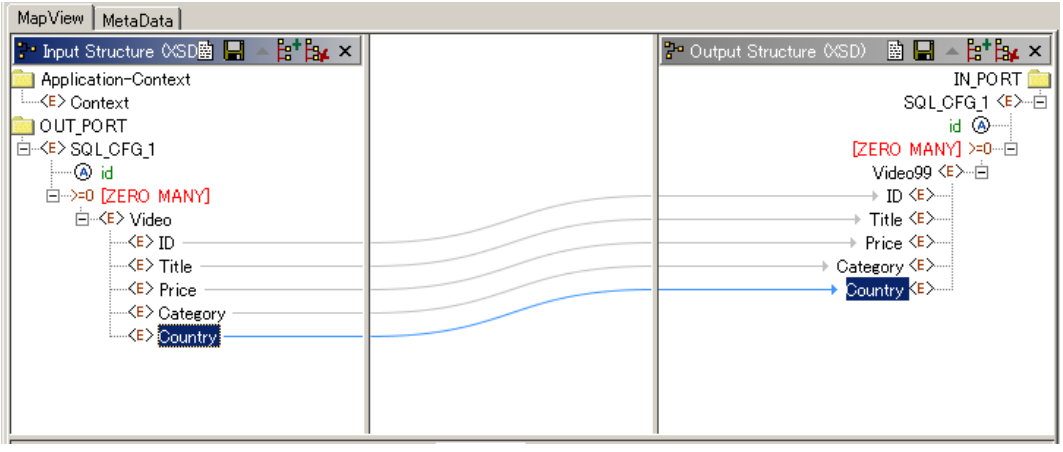


マッパー画面の左側は、[ビデオ] コンポーネントのアウトプット ポートのデータ構造を、右側は [ビデオ 99] コンポーネントのインプット ポートのデータ構造を示しています。

データ エレメント間の関係は、下図のようになっています。この関係を Fiorano マッパー上で指定していきます。



具体的な操作は、左側の “ID” エレメントを、右側の “ID” までマウスでドラッグします。以降、すべてのエレメントについて同じように結び付けていきます。次のキャプチャ画面のようになります。



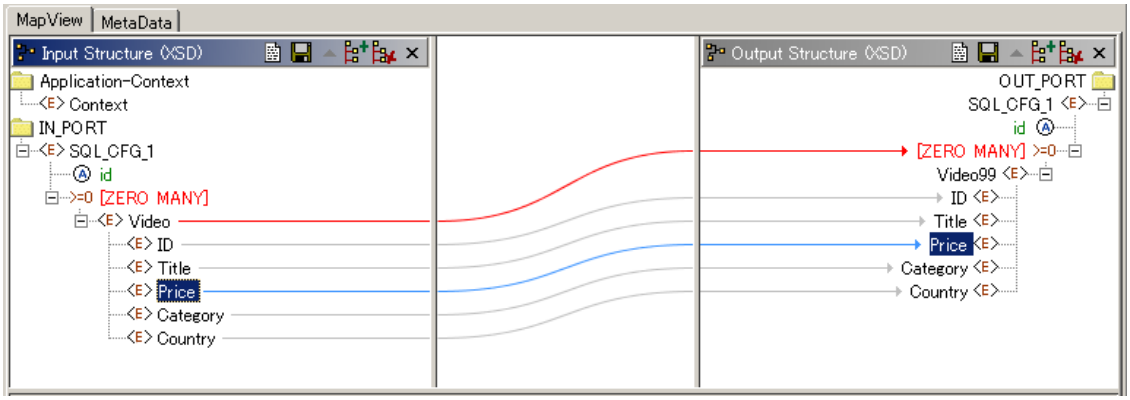
[ビデオ] コンポーネントから複数オカレンス（複数レコード）を持つデータが送られてくる場合のことも考慮しておきます。付録で説明している mckoi データベースのテーブル作成では 1 レコードしか登録していませんが、複数レコードが Video テーブルに存在する場合のことも考慮しておきます。

以下に示したデータ構造は、[ビデオ] コンポーネントにおいて SELECT ステートメントを実行した結果、3 レコードが検索された場合を示しています。

<ns1:SQL_CFG_1>	
@id	3568372652642530304
@xmlns:ns1	http://www.fiorano.com/fes
<ns1:Video>	
<ns1:ID>	101
<ns1:Title>	フーテンの寅さん
<ns1:Price>	4980
<ns1:Category>	video
<ns1:Country>	日本
<ns1:Video>	
<ns1:ID>	102
<ns1:Title>	ハリーポッター
<ns1:Price>	2980
<ns1:Category>	DVD
<ns1:Country>	USA
<ns1:Video>	
<ns1:ID>	201
<ns1:Title>	24 シーズン 1-1
<ns1:Price>	2980
<ns1:Category>	DVD
<ns1:Country>	USA

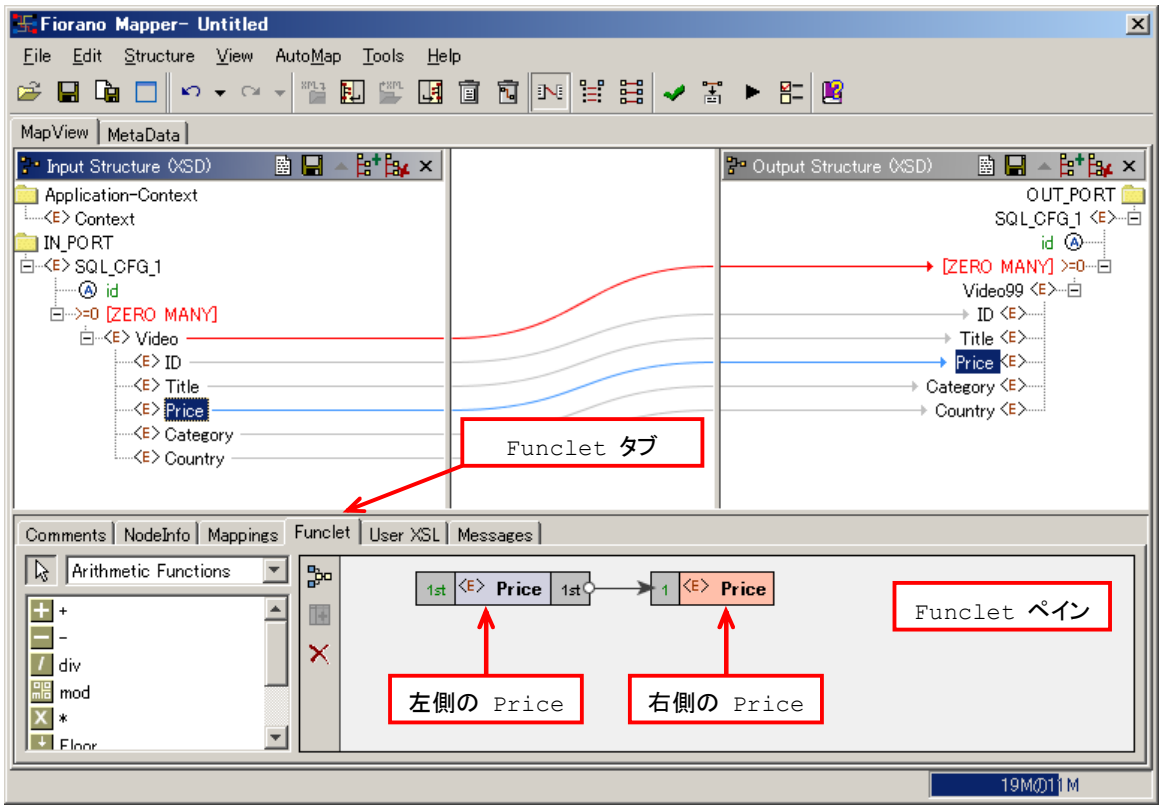
上のように複数のオカレンス（レコード）を持つデータが Xslt コンポーネントに入力されても、すべてのレコードに対してマッピングが実施され、アウトプット ポートからすべてのレコードが出力される必要があります。

これを実現するためには、最上位のエレメント **Video** をアウトプット側の **[ZERO MANY]** につなぎます。この指定をしないと、アウトプット ポートからは 1 レコード分しか出力されなくなります。



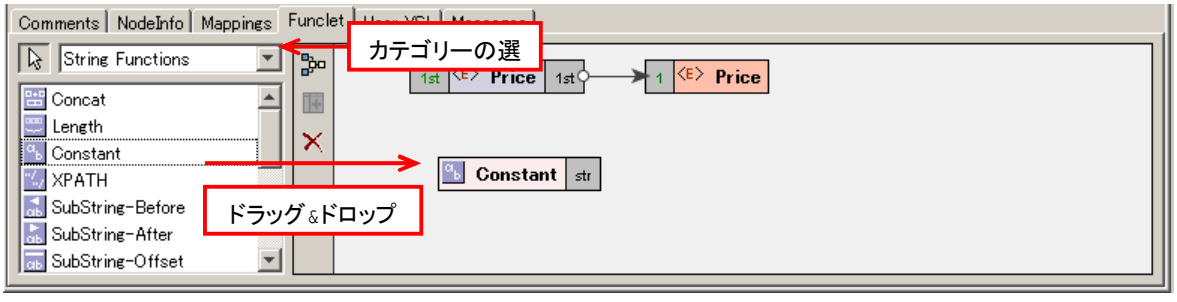
次に、消費税の設定を行います。

右側（[ビデオ 99] コンポーネントのインプット側）の “Price” エレメントを選択した状態で、**[Funclet]** タブをクリックしてください。次のように表示されます。

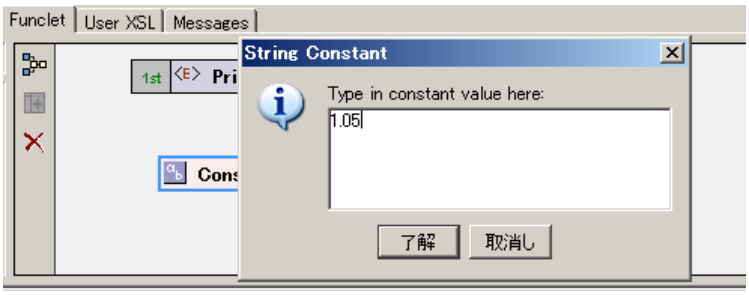


上の画面の Funclet ペインが示している意味は、左側 ([ビデオ] コンポーネントのアウトプット) の Price エレメントの値をそのまま右側 ([ビデオ 99] コンポーネントのインプット側) の Price エレメントの値にするということです。
消費税込みの値段に変更して代入する必要があります。具体的には、左側の Price の値を 1.05 倍してから右の Price に代入します。

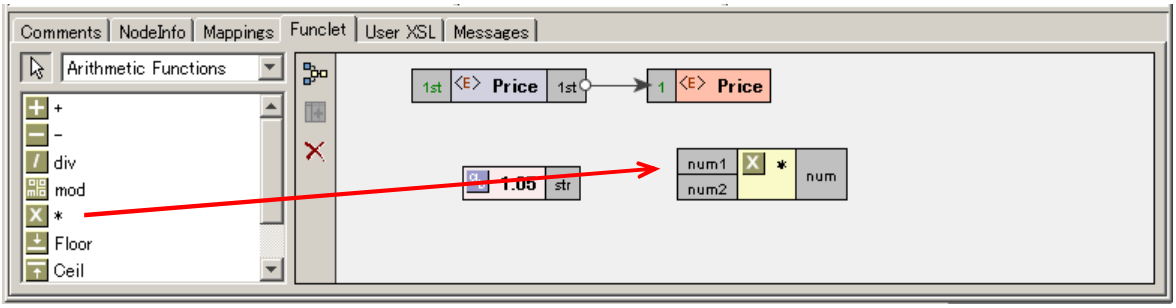
1. まず、左のペインから “Constant”（定数）を、右ペインにドラッグ&ドロップします。Constant は、**[String Functions]** カテゴリーにあります。



2. “Constant” の値を 1.05 に設定します。ドラッグしてきた Constant をダブルクリックすると入力ダイアログが表示されます。1.05 を入力し、[了解] ボタンをクリックすると、定数値の設定が完了します。

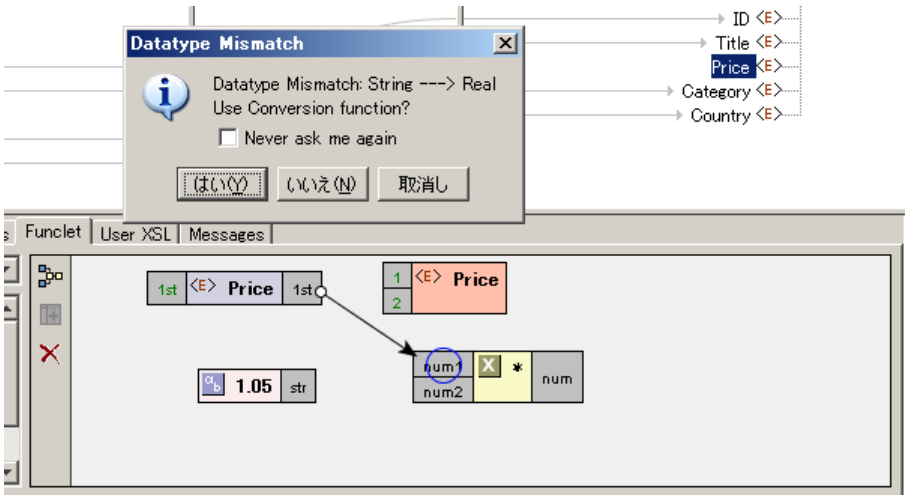


3. 四則演算のカテゴリーから掛け算のファンクションを、ドラッグ&ドロップします。四則演算のファンクションは **Arithmetic Functions** カテゴリーにあります。



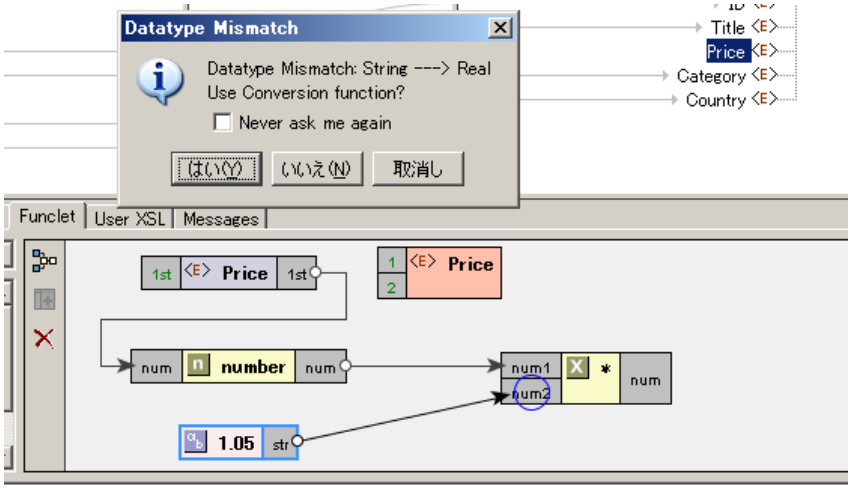
4. 次に、左側の Price から出ている矢印を、掛け算のファンクションに結びなおします。矢印の先端をクリックし、掛け算ファンクションの“**num1**”にドラッグします。

すると、次のキャプチャ画面のように、データ型を“ストリング”から“実数型”に変更するかと、問い合わせてきますので、[はい] をクリックします。

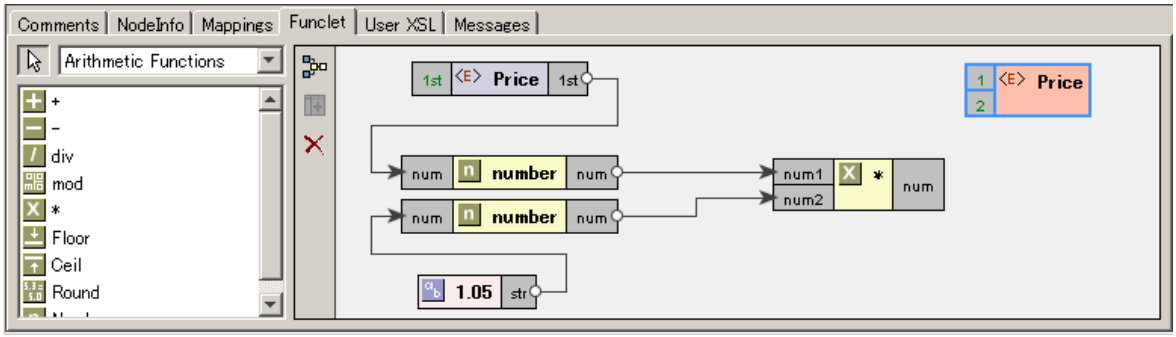


[はい] をクリックすると次のようになります。

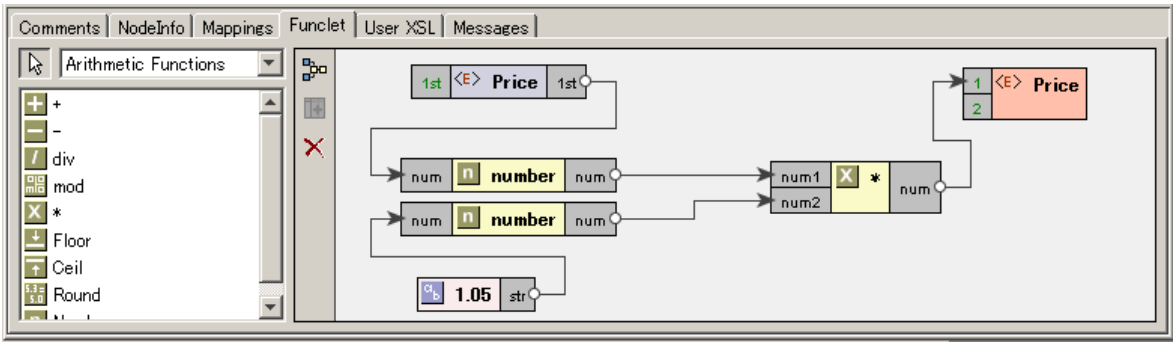
5. 次に Constant (1.05) を、掛け算ファンクションの num2 にドラッグします。先ほどと同様に、型変換を行うか聞いてきますので、[はい] を選択します。



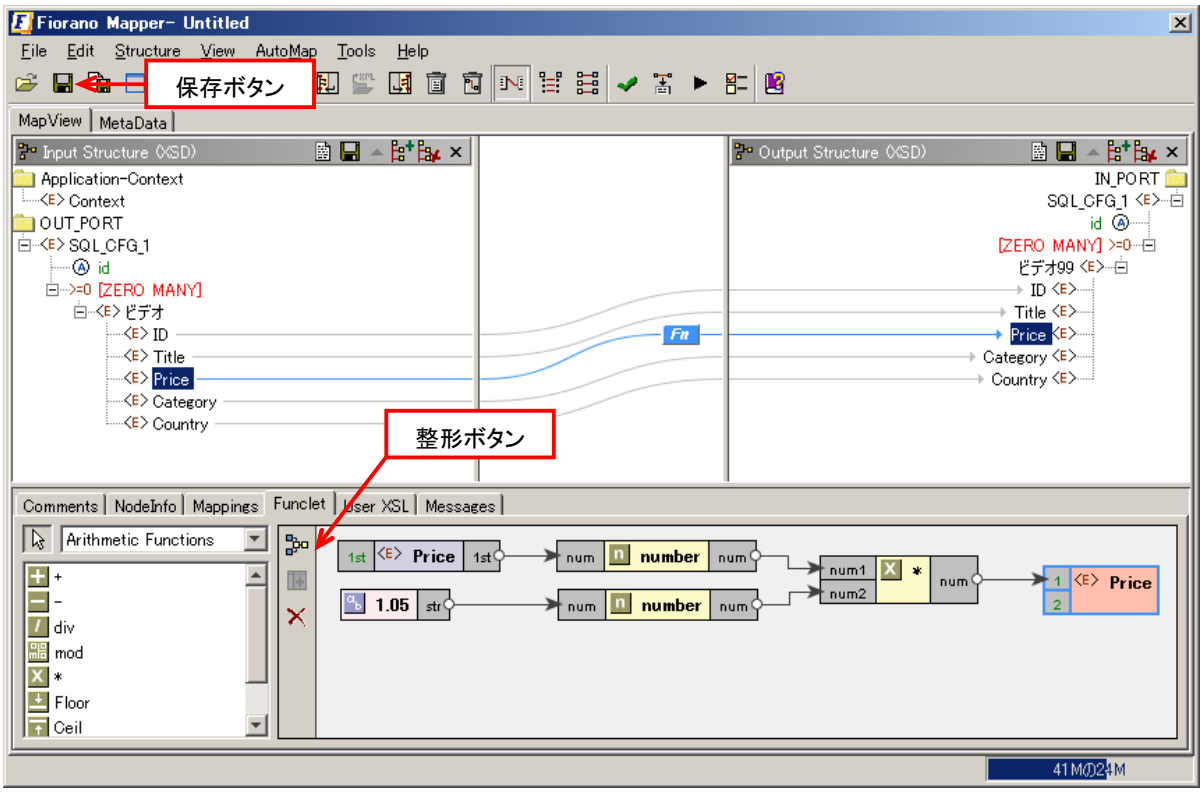
結果が次のようになります。



6. 最後に、掛け算ファンクションの結果を、Price に代入します。



マッパー画面の最終形は、次のようになります。



整形ボタンで Funclet ペイン内の図を整形することができます。

保存ボタンで設定を保存し、マッパー ウィンドウを閉じます。

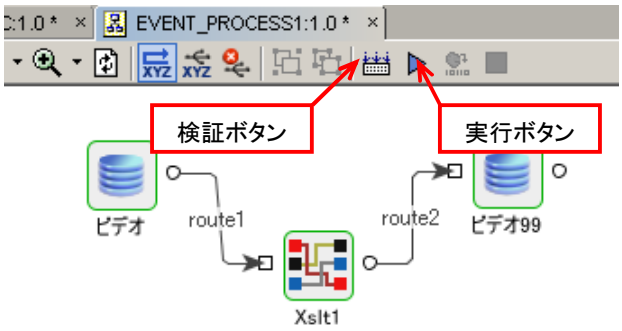
4 コンポーネント フローの実行

4.1 コンポーネント フローの検証と実行

今までの操作でコンポーネント フローの作成が完了しましたので、フローを実行してみます。

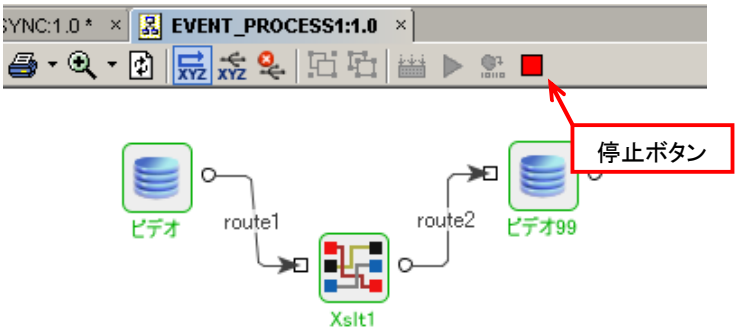
実行は、次の 2 つのステップによって行います。

1. フローの検証 (フローが論理的に正しいか、使用している ピア サーバーが稼動しているかなどのチェック)
   (フローを最初に実行するときには必須)
2. 実行
   (実行ボタンがグリーン色になっていないときは、検証が必要なことを表わしています)



実行ボタンをクリックすると、実行に必要なリソースが ESB サーバーからピア サーバーに転送され、ピア サーバー上でフローの実行が始まります。

次の画面は、コンポーネント フローが実行されている状態でキャプチャしたものです。



コンポーネント名が緑色に変わっています。また、フローの停止ボタンが赤色に変更され、停止ボタンが有効になったことを示しています。

コンポーネント名の色は、黒 (フローの停止中、フローの開発中) → 青 (デプロイメントの完了) → 緑 (実行中) の順に変わっていき、実行状況がわかるようになっています。

コンポーネント名が赤色で表示される場合は、「コンポーネント フローは稼動しているが、赤色のコンポーネントのみが停止している」状態を意味しています。コンポーネント フローも停止している場合には、コンポーネント名の色はすべて黒色で表示されます。

4.2 実行結果の確認

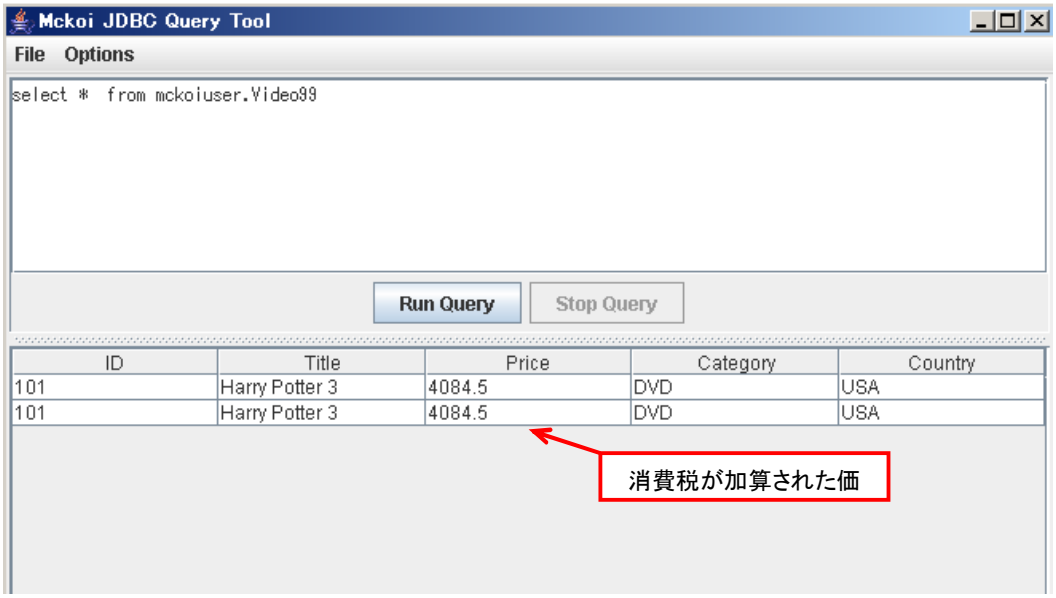
QueryMckoi.bat を用いて、Video99 テーブルが 20 秒ごとに更新 (レコードが追加) されることを確認してください。

QueryMckoi.bat の起動方法については、『付録 A : Mckoi データベースの使用方法』を参照してください。

下記の SQL ステートメントを、何回か実行してみてください。 [ビデオ] コンポーネントで指定したスケジューラのポーリング間隔の秒数が経過すると、新たなレコードが Video99 テーブルに追加されていくことが確認できます。

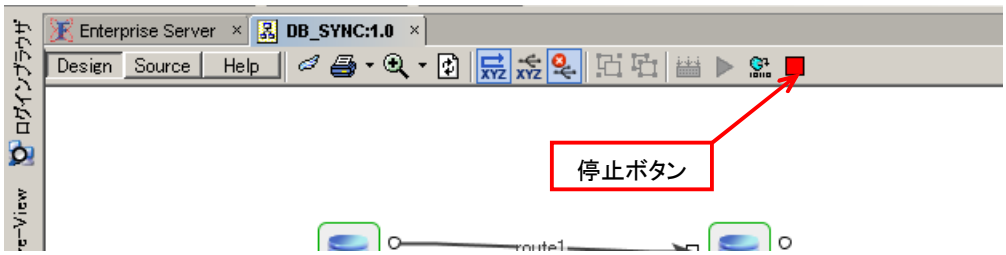
```
select * from mckoiuser.Video99
```

Price の項目が、Video テーブルのデータ値を 1.05 倍されていることも確認できます。
次の画面は、 [ビデオ 99] コンポーネントによって 2 回 INSERT された状態でキャプチャしたものです。



4.3 コンポーネント フローの停止

コンポーネント フローの停止には、停止ボタンをクリックします。



5 エラー フローの追加

コンポーネント フロー内の各コンポーネントで発生する例外を処理する方法には、次の 2 つがあります。

- エラー ポートから送信されるエラー情報を処理する方法
- エラー リスナー コンポーネントを用いて統一的に処理する方法

この章では、発生した例外を処理する、上記 2 つの方法を説明します。

5.1 エラー メッセージの形式

Fiorano がプリビルトしてバンドルしているコンポーネントが出力するエラー メッセージは、共通して次の内容となっています。

- エラー コード (番号)
- エラー メッセージ
- エラーの詳細説明
- エラーが発生した際の入力データ

5.2 エラー ポートを利用する方法

5.2.1 エラー フローの追加

各コンポーネントは、『3.5 サービス コンポーネントのプロパティ設定』のセクションで説明したように、例外発生時の処理を CPS (カスタム プロパティ シート) で定義できるようになっています。デフォルトでは、下の表に示すように、エラー ポートからエラー メッセージを送信するよう設定されています。

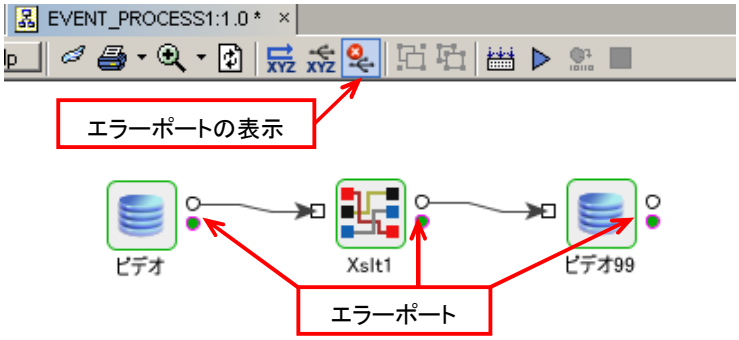
エラーのカテゴリー	選択できる処理	デフォルト設定
Request Processing Error **受信データの処理エラー** (受信データの処理中に例外が発生)	リトライ	
	ワーニングを throw	
	エラー ポートからエラー メッセージを送信	**デフォルト**
	コンポーネントを停止	
Connection Error **コネクション エラー** (処理中に各リソース (ライブラリ、ピア サーバー、データベースなど) への接続がロスト	再接続の試行	
	エラー ポートからエラー メッセージを送信	**デフォルト**
	コンポーネントを停止	
Invalid Request Error **入力データの形式エラー** (誤った形式のデータを入力)	エラー ポートからエラー メッセージを送信	**デフォルト**
	コンポーネントを停止しない	**デフォルト**

エラー ポートから送信されるエラー メッセージを処理するためには、次のようにします。

1. エラー ポートの表示

エラー ポート表示ボタンをクリックすると、各コンポーネントのエラー ポート (囲み線が赤色、中が緑色の丸型) が表示されます。

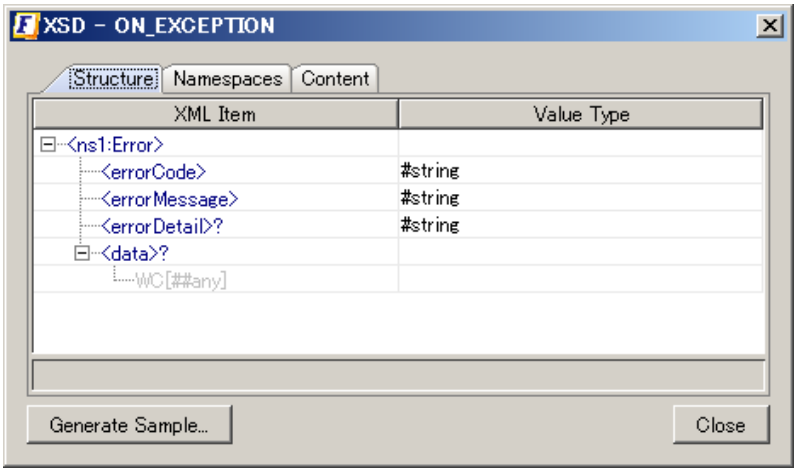
(下のキャプチャ画面では、ルート名表示ボタンをクリックし、ルート名を表示しないようにしています。)



エラー ポートから出力されるデータのスキーマは、次のようにして確認できます。

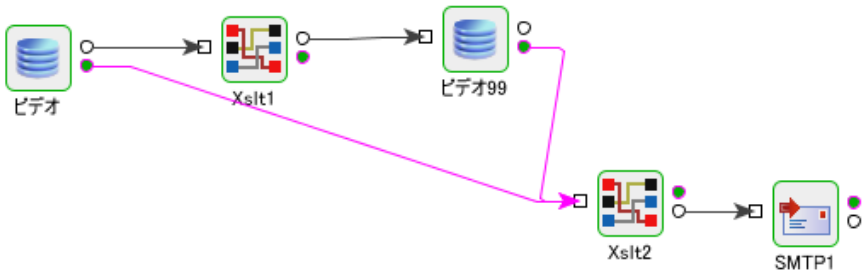
エラー ポートを右クリックし、メニューから [**View Schema Structure** …] を選択します。

下のように、エラー ポートのデータ スキーマが表示されます。



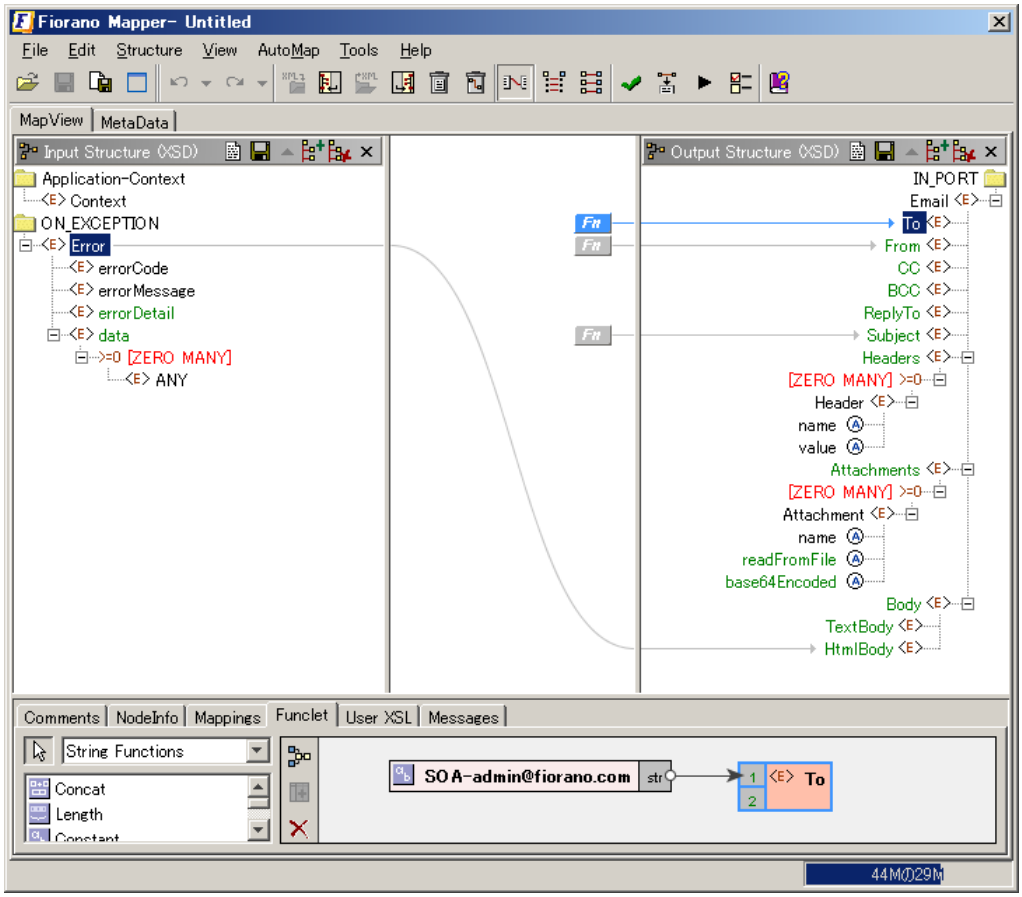
2. エラー ポートからのルートの設定

下の画面は、それぞれの DB コンポーネントで発生したエラーを、電子メールで担当者に送信するように設定したものです。



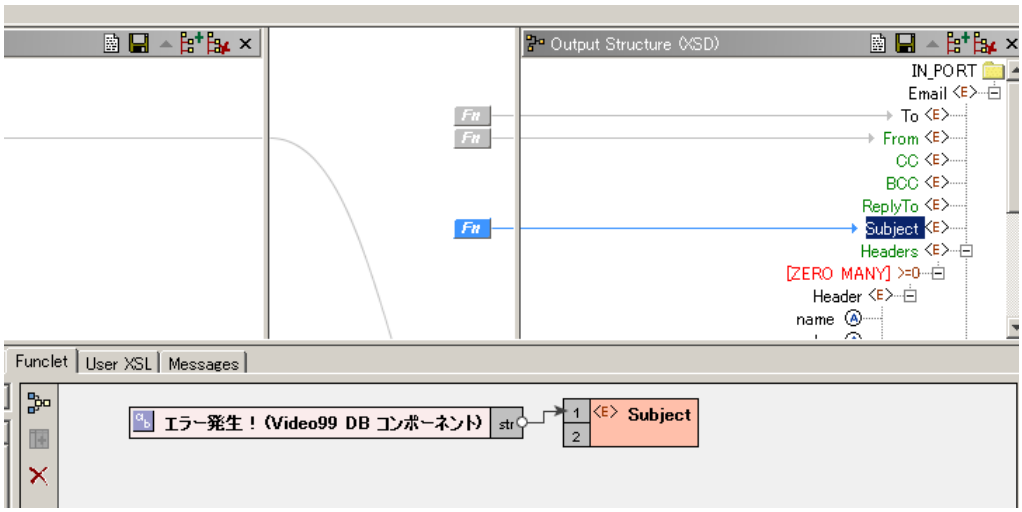
3. データ マッピング (Transformation) の設定

ビデオ および [ビデオ 99] コンポーネントのエラー ポートから送られてくるエラー メッセージをメール形式に変換するため、Xslt コンポーネントを用います。



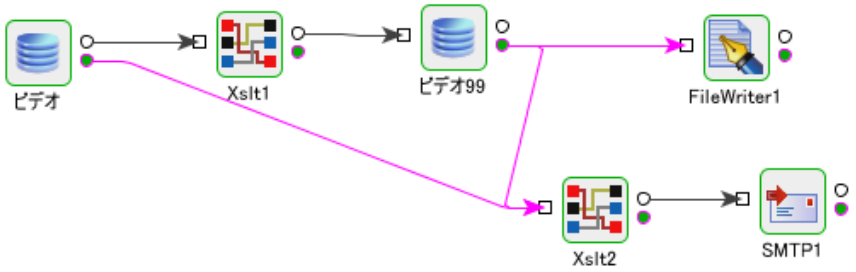
上の画面の設定では、エラー情報を電子メールの HTML 形式の本文 (HtmlBody) にセットし、宛先 (To) や件名 (Subject) は、ファンクレット機能を用いて設定しています。

例えば、下の画面のように、メールの件名 (Subject) にコンポーネント名も表示するように設定することで、メールの受信者にエラーが発生したコンポーネントを明示することもできます。



5. 他のコンポーネントの追加

さらに、データベースへの書き込みを行っている [ビデオ 99] コンポーネントでエラーが発生した場合には、そのエラー内容をファイルに書き出すようにします。こうすることで、データベースに書き込めなかったことによるデータの消失を防ぐことができます。下の画面のように、ファイルへの書き出しを行うコンポーネント FileWriter を追加します。



次の画面は、FileWriter の CPS の最初の画面をキャプチャしたものです。

各パラメータ値は、以下のように設定しています。

Compute Paths relative to Directory : **C:¥work**

Is Configured for Different Machine : no （デフォルト値）

File Name : **Video99エラー.txt**

Use file details from message header ? : no

Target Directory : C:¥work¥エラーメッセージ

Output Mode : **Append if exists**

Working Directory : **work**

Error Directory : **Err**

この設定の意味は、

a **C:¥work¥エラーメッセージ** に **Video99 エラー.txt** という名前のファイルを作成し、受信したエラー メッセージをこのファイルに追加していく。

b エラー メッセージを受信し、書き出すごとに **Video99エラー.txt** ファイルは閉じるようにする。

c FileWriter コンポーネントの処理用ワーキング ディレクトリとして **C:¥work¥work** を使用し、FileWriter 自身のエラーが発生した場合には、**C:¥work¥Err** をエラー処理用のワーキング ディレクトリとして使用する。

というものです。

ちなみに、FileWriter コンポーネントに渡す場合には、データ変換は必要ありません。FileWriter コンポーネントは、インプットのデータ スキーマを持っておらず、受け取ったデータをすべてそのまま書き出すためです。エラー情報に何らかの加工を行いたい場合には、Xslt コンポーネントなどを用いてエラー情報の加工を行った後に FileWriter に渡すようにします。

ここでは、ファイルに出力するようにしましたが、他のコンポーネント（例えば、ディスプレイへの表示など）にエラー メッセージを渡すことができます。

5.2.2 エラー フローの確認

実際にエラーが発生する状況を作り出して、作成したエラー フローが正しく機能するか確認します。

このセクションでは、ビデオ 99 のコンポーネントでエラーが発生する状況を作り出し、ビデオ 99 のエラー ポートから出力されたエラー メッセージがファイルに書き出されたことを確認する手順を説明します。
電子メールによるエラー メッセージの送信については、評価者の環境でご確認ください。

1. エラー状況の創出

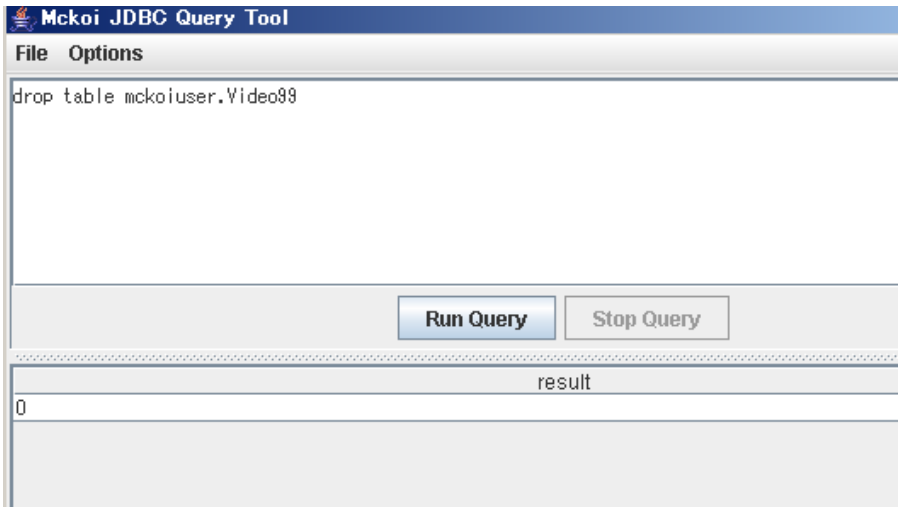
[ビデオ 99] コンポーネントで例外が発生する状況は何種類か考えられますが、ここでは [ビデオ 99] コンポーネントが INSERT 対象としている Video99 テーブルが存在しない状況を作り出します。
こうすることで、[ビデオ] コンポーネントが正しく Video テーブルからデータを SELECT でき、[ビデオ 99] コンポーネントでは INSERT ステートメントの実行において例外が発生する状況となります。

RunMckoi.bat によって Mckoi データベースを起動します。

QueryMckoi.bat によって、Mckoi クエリ ツールを起動します。

クエリ ツールで、Video99 テーブルを削除する、drop table mckoiuser.Video99 を実行します。

(Mckoi の実行、クエリ ツールの実行の詳細については、『付録 Mckoi データベースの使用方法と設定』を参照してください。)



2. DB_Sync コンポーネント フローの実行

DB_Sync を実行します。

[ビデオ] コンポーネントの最初のポーリング間隔が経過した後、Video99 に最初のデータが送られれてきます。

Video99 でクエリを実行しようとしますが、対象のテーブルが無いため、エラーが発生します。

3. ファイルの確認

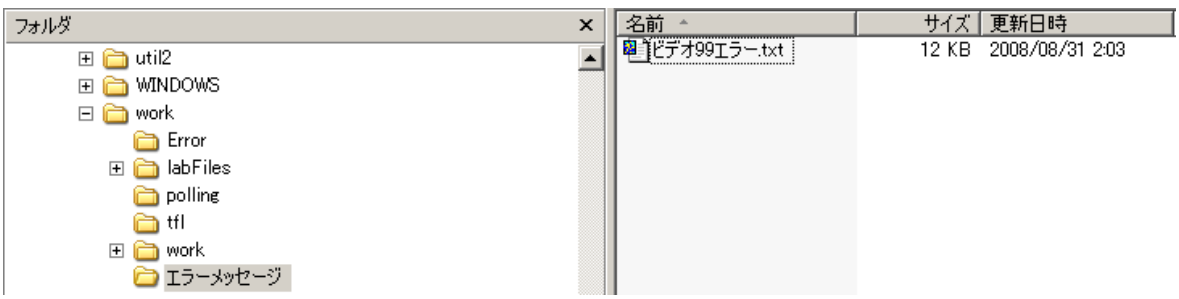
FileWriter で設定した

C:¥Work¥エラーメッセージ

に

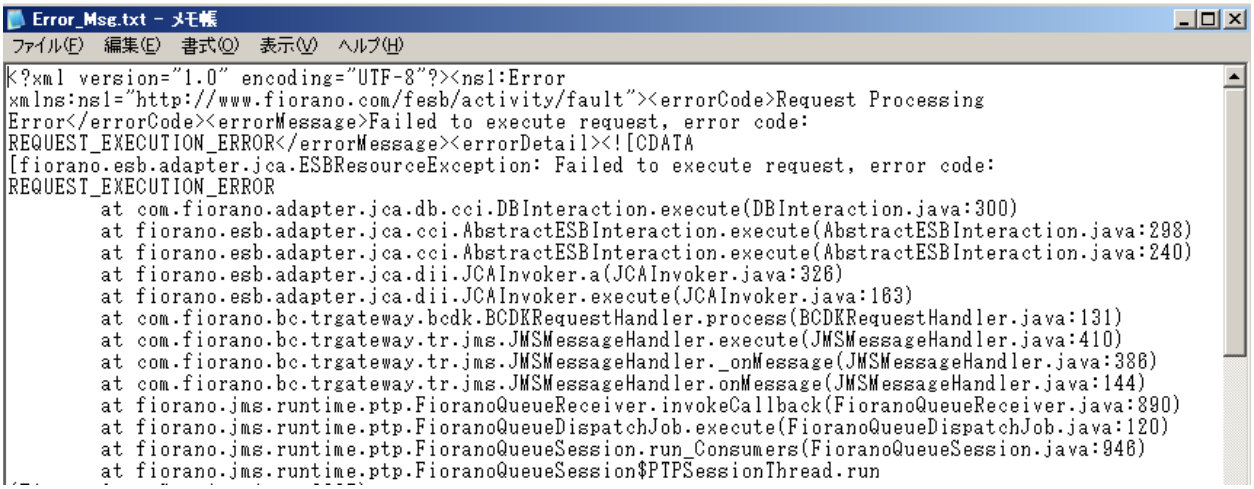
ビデオ 99 エラー.txt

というファイルが作成されていることを確認します。



4. エラー内容の確認

ファイルに書き出されたエラー メッセージの内容は、次のようにメモ帳などのテキスト エディタで閲覧できます。



書き出されたエラーのメッセージの先頭には、

```
<?xml version="1.0" encoding="UTF-8"?><ns1:Error
xmlns:ns1="http://www.fiorano.com/fesb/activity/fault"><errorCode>Request
Processing Error</errorCode><errorMessage>Failed to execute request, error code:
REQUEST_EXECUTION_ERROR</errorMessage><errorDetail>
```

とあり、受信したデータの処理中のエラーであることがわかります。

さらに、エラー メッセージの続きをみていくと、

```
(FioranoQueueSession.java:3067)
caused by
com.fiorano.adapter.db.jdbc.DBException: Error executing statement : INSERT INTO
"mckoiuser"."Video99"
( "ID", "Title", "Price", "Category", "Country" )
VALUES ( ?, ?, ?, ?, ? ). Reason : Table 'mckoiuser.Video99' does not exist.
```

とあり、INSERT ステートメントの実行中のエラーであり、その理由が Video99 テーブルが存在しないことだとわかります。

エラー メッセージの最後には、

```
<ns3:A>
  <ns3:ID>101</ns3:ID>
  <ns3:Title>Harry Potter 3</ns3:Title>
  <ns3:Price>4084.5</ns3:Price>
  <ns3:Category>DVD</ns3:Category>
  <ns3:Country>USA</ns3:Country>
</ns3:A>
```

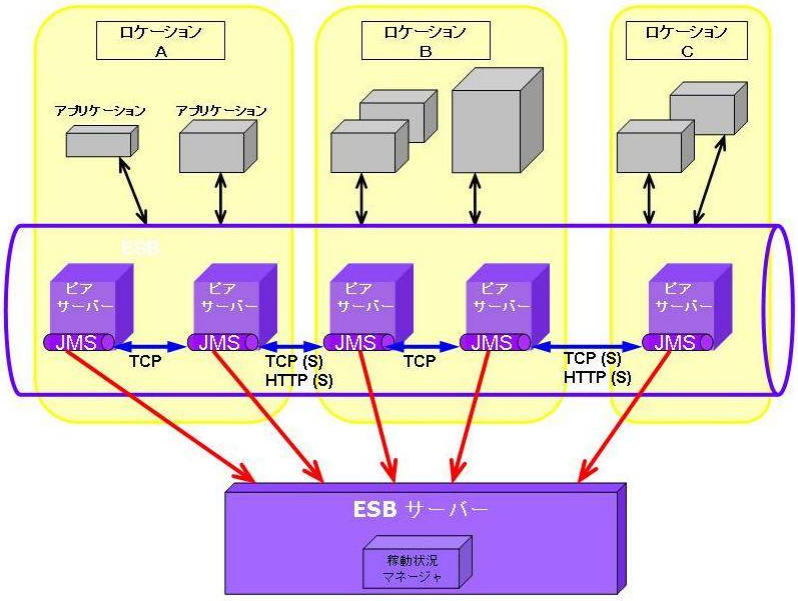
と、受信したデータが記録されています。

5.3 エラー リスナー コンポーネントの利用

5.3.1 エラー リスナーの作成

『5.2 エラーポートを利用する方法』は、エラー発生時の処理をコンポーネント毎に異なるものとする場合に有効です。
エラー リスナー コンポーネントは、エラー発生時の処理をコンポーネント毎に異なるものとするのではなく、統一的な共通した処理とする場合に有効です。

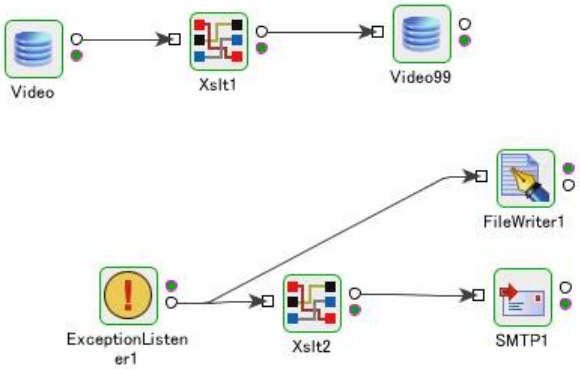
Fiorano SOA プラットフォームでは、稼動中のすべてのコンポーネント フローおよびコンポーネント フロー中の各コンポーネントで発生した例外のメッセージやシステム メッセージが、ESB サーバーに集められます。下図における、赤い線がこのメッセージの収集を示しています。ESB サーバーに集められたメッセージは、[稼動状況 マネージャ] によって管理されます。



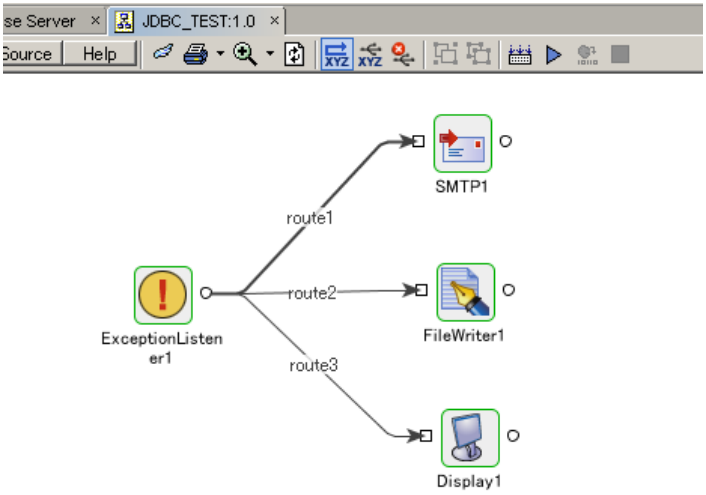
`ExceptionListner` コンポーネントは、ESB サーバーに届くエラー メッセージをリッスンしています。したがいまして、`ExceptionListner` コンポーネントがリッスンしたエラー メッセージを処理するフローを作成すれば、すべてのコンポーネント フローとその中のコンポーネントで発生したエラーを共通して処理することが可能となります。

エラー リスナーのフロー作成

エラー リスナーのフローは、下図のように本来の業務をおこなうコンポーネント フローの中に設定することもできます。



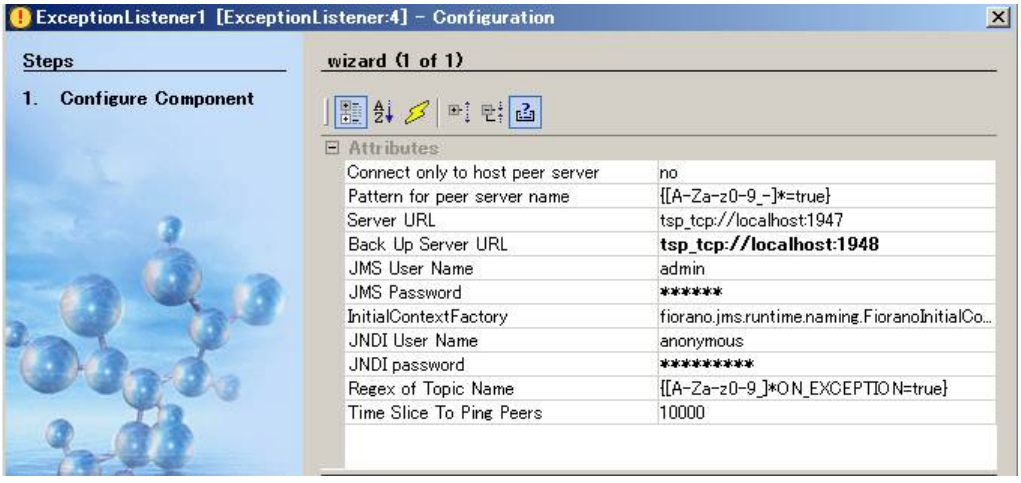
しかしながら、すべてのコンポーネント フローのエラーを統一的に扱うという性格を考えると、エラー リスナーのは独立した単一のフローとして構築するのがよいでしょう。



上図のフローは、ExceptionListner を用いた共通エラー処理フローの例です。

ESB ネットワーク上で稼動しているコンポーネント フローで発生した例外のエラー メッセージが ESB サーバーに届くと、ExceptionListner によって受信され、このコンポーネントのアウトプット ポートからリンクしているコンポーネントに渡されます。

下のキャプチャ画面は、ExceptionListner の CPS です。ここで指定するサーバーの URL とは、ESB サーバーの URL を指しています。



基本的に、ESB サーバーの URL 以外のパラメータ設定はデフォルト値のままでよく、変更する必要はありません。

5.3.2 エラー リスナーの実行と確認

『5.2 エラー ポートを利用する方法』と同じエラーを起こしてみます。

1. エラー リスナーのフローを実行し、エラーのリッスンを開始します。

2. DB_Sync のフローにおいて、エラー ポートからつながっているルートをすべて削除しておきます。
   エラー ポートからのリンクを残しておいてもよいのですが、削除したほうがエラー リスナーのフローによって処理されたことが明確に把握できます。

3. DB_Sync フローを実行します。
   エラー状況の創出
   1)『5.2.2 エラー フローの確認』で利用したのと同じ、Video99 テーブルが存在しない状況
   または、
   2) Mckoi データベースを起動しないまま、
   DB_Sync フローを実行すると、Video99 コンポーネンもしくは [ビデオ] コンポーネントでアクセス エラーが発生します。

4. 『5.2.2 エラー フローの確認』で説明したのと同様の方法で、エラー処理の結果を確認します。

付録 A : Mckoi データベースの使用方法と設定

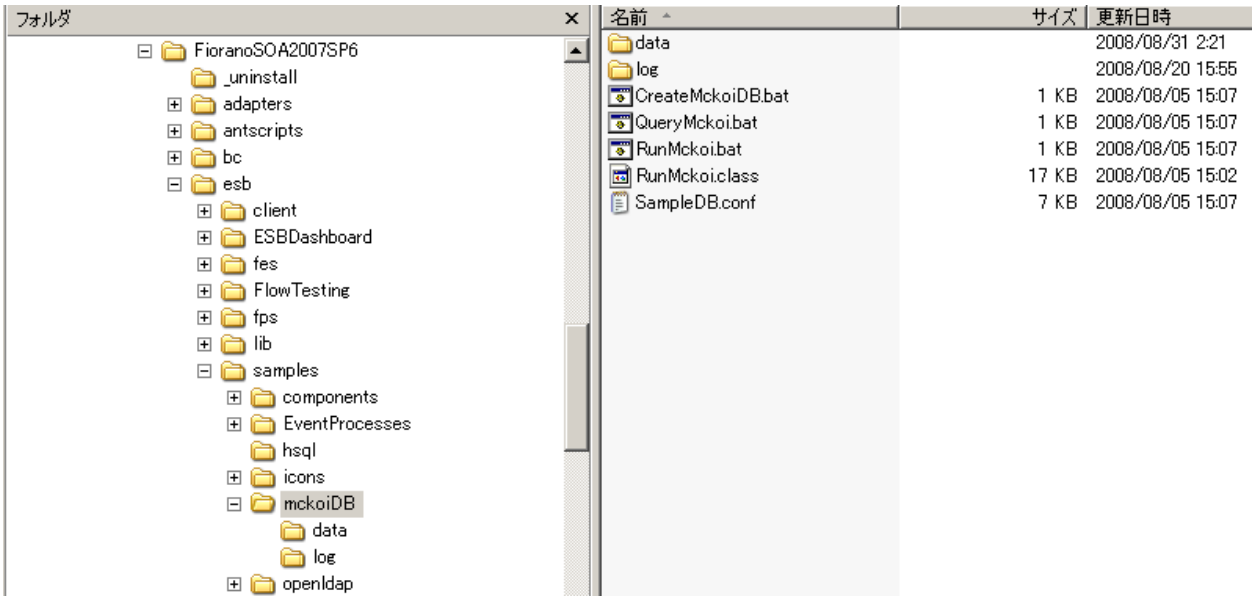
1. Mckoi 用 .bat ファイルの場所

Fiorano SOA プラットフォームには、フリー ソフトウェアのデータベース システムが数種類付属しています。

このガイドで説明している例題では、Mckoi データベースを使用します。

次の画面は、インストール フォルダー (通常は、**¥Program Files¥Fiorano¥FioranoSOAxxxx**) 下のツリー構造を示しています。(xxxx は、バージョン番号を示しています。)

¥Program Files¥Fiorano¥FioranoSOAxxxx¥esb¥samples¥mckoiDB フォルダーの下に、Mckoi DB を実行、操作するための .bat ファイルが格納されています。



2. Mckoi の起動

Mckoi を最初に起動するには、次の 2 つの bat ファイルを次の順に実行します。

1. **CreateMckoiDB.bat**
2. **RunMckoi.bat**

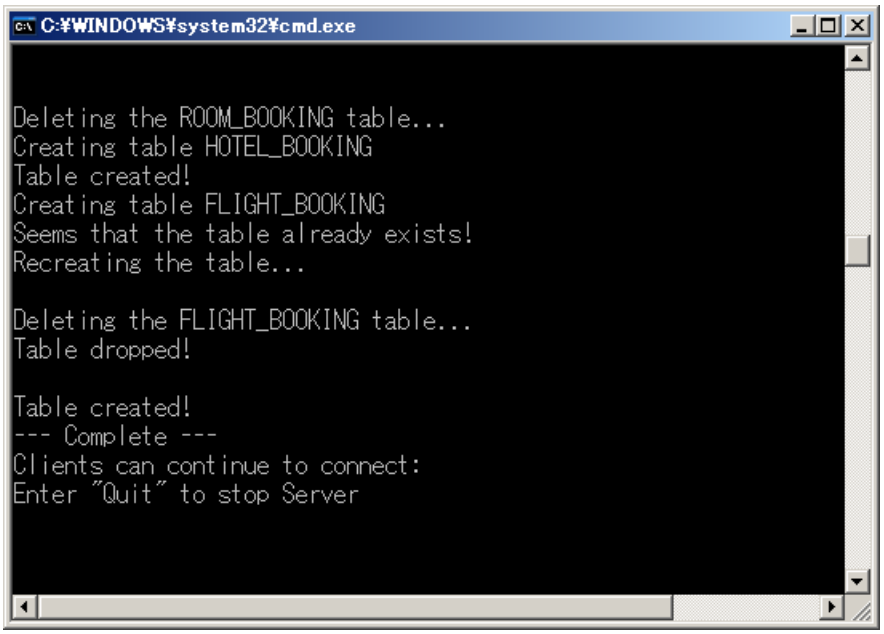
CreateMckoiDB.bat をダブルクリックすると、Fiorano によってあらかじめ指定されているデータベースやテーブルを生成します。ここで生成されたテーブルは、製品にバンドルされているサンプル プロセスで使用さるものです。この **CreateMckoiDB.bat** は、最初に 1 回だけ、**RunMckoi.bat** に先立って実行します。

RunMcki.bat は、Mckoi の DBMS を起動する bat ファイルです。

2 回目以降は、この RunMckoi.bat だけ実行すればよく、CreateMckoiDB.bat の実行は必要ありません。

正常に起動とテーブルの作成が完了すると、次のように表示されます。

```
--- Complete ---
Clients can continue to connect :
Enter "Quit" to Stop Server
```



3. 例題で使用するテーブルの作成

SQL を実行するためのツール **QueryMckoi.bat** を実行してください。

次の画面が表示されます。

入力エリアに SQL ステートメントを入力し、**[Run Query]** ボタンをクリックすると、クエリ結果ウィンドウに実行結果が表示されます。

このガイドの例題で使用するのは、次の 2 つのテーブルです。両テーブルともユーザー mckoiuser のもとに生成します。

mckoiuser.Video

ID	101
Title	Harry Potter 3
Price	3890
Category	DVD
Country	US

mckoiuser.Video99

ID	
Title	
Price	
Category	
Country	

また、Video テーブルにはデータとして上図の値を入力しておきます。Video99 には、データを入力しません。テーブルのみ生成しておきます。

上述の SQL クエリ実行ツール (**QueryMckoi.bat**) で、次の SQL を実行してください。

1. Video テーブルの生成

```
create table mckoiuser.Video(ID VARCHAR(5), Title VARCHAR(30), Price VARCHAR(10),
                             Category VARCHAR(5), Country VARCHAR(10))
```

2. Video99 テーブルの生成

```
create table mckoiuser.Video99(ID VARCHAR(5), Title VARCHAR(30), Price VARCHAR(10),
                               Category VARCHAR(5), Country VARCHAR(10))
```

3. Video に値を入力

```
insert into mckoiuser.Video(ID, Title, Price, Category, Country)
values('101', 'Harry Potter 3', '3890', 'DVD', 'USA')
```

Video テーブルを検索 (SELECT) して、次のように表示されれば完了です。