IBM WebSphere Adapters
バージョン 7 リリース 5

IBM WebSphere Adapter for JDBC ユーザーズ・ガイド
バージョン 7 リリース 5

IBM WebSphere Adapters
バージョン 7 リリース 5

IBM WebSphere Adapter for JDBC ユーザーズ・ガイド バージョン 7 リリース 5

お願い

本書および本書で紹介する製品をご使用になる前に、 399 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Adapter for JDBC バージョン 7 リリース 5 モディフィケーション 0 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

第1刷 2011.6

目次

IBM WebSphere Adapter for JDBC の資料

IBM® WebSphere® Adapter for JDBC を使用すれば、特殊なコーディングをすることなく、データベースとの情報交換を含む、統合処理を作成できます。

IBM WebSphere Adapter for JDBC の概要

IBM WebSphere Adapter for JDBC を使用して、データベースと相互作用し、情報交換を行う統合アプリケーションを作成できます。アプリケーションは、多くの場合 SQL コードを必要とせずに、アダプターを使用して、要求をデータベースに送信することも、データベースからイベントを受け取ることもできます。

アダプターにより、IBM Business Process Manager または WebSphere Enterprise Service Bus で実行されるアプリケーションと、データベースとの間で両方向の通信が可能になります。アプリケーションは、多くの場合 SQL コードを書かなくても、アダプターを使用して、データベースにあるデータの読み取り、作成、変更、または削除要求を送信できます。アプリケーションから受け取った要求を処理するため、アダプターは SQL 照会またはストアード・プロシージャーを使用してデータベース表を更新します。アプリケーションは、データベースからのイベントも受信できます。例えば、特定のデータベース表が更新されたという通知を受け取ることができます。データベースに対する変更によって生じたイベントを処理するため、アダプターはイベントをアプリケーションに送信します。イベント通知を使用して、データベース更新を自動的に他のアプリケーションに伝搬できます。IBM WebSphere Adapter for JDBC と別のアダプターによるイベント処理を組み合わせることで、Siebel、 PeopleSoft、Oracle などのエンタープライズ・アプリケーションに更新を自動的に伝搬できます。

アダプターは、さまざまなデータベース・ソフトウェアのベンダーやバージョンと統合する標準インターフェースを備えており、 Java Database Connectivity (JDBC) 2.0 またはそれ以降の仕様をサポートする JDBC ドライバーがインストールされているすべてのデータベース・サーバーをサポートします。 JDBC ドライバーが必要とする JRE バージョンは、ランタイム環境の JRE バージョンに等しいかそれ以下でなければなりません。そのようなサーバーの例としては、IBM DB2®、Oracle、Microsoft SQL Server、Sybase、Derby、および Informix® などがあります。アダプターは、ビジネス・オブジェクトを使用してアプリケーションとデータベース間でデータを交換するため、アプリケーションでは JDBC アプリケーション・プログラミング・インターフェース (API) を使用する必要がありません。ビジネス・オブジェクト とは、Business Function やビジネス・エレメントを表すアプリケーション・データ (データベース表や SQL 照会の結果など) のコンテナーです。アダプターは、アプリケーションのデータ・フォーマットを認識し、データを処理し、操作を実行し、結果をそのフォーマットで戻すことができます。

このリリースの新機能

このバージョンには、ビジネスの柔軟性、ユーザー・エクスペリエンス、およびアダプターの性能を強化するためのいくつかの新機能が含まれています。

IBM WebSphere Adapter for JDBC、バージョン 7.5 に含まれている新機能は、以下のとおりです。

- 新しいページング機能により、ページ単位でのレコード取り出しがサポートされます。
- マイグレーション機能の強化により、読み取り専用ファイルを処理し、必要に応じて更新することが可能です。
- ストアード・プロシージャー・ビジネス・オブジェクト、照会ビジネス・オブジェクト、およびバッチ照会ビジネス・オブジェクトで XML データ型がサポートされます。
- フォールトをビジネス・フォールトとしてカスタマイズできます。
- UpdateAll、DeleteAll、Upsert など、新しい Outbound 操作が追加されました。
- Outbound 処理中、テーブル・ビジネス・オブジェクト内の UpdateAll、DeleteAll、および RetrieveAll 操作で、柔軟な where 節条件がサポートされます。
- クラスター環境で Inbound イベント・ポーリングに対して高可用性 Active-Active フィーチャーがサポートされ、アダプターの複数インスタンスが、イベントのポーリングとイベント送達を重複なしでアクティブに実行できます。
- BatchCreate、BatchUpdate、および BatchDelete などのバッチ操作のサポートにより、トップレベルのビジネス・オブジェクトのレコードを単一の操作で追加/更新/削除することができます。
- マイグレーション
  - IBM Business Process Manager 上での IBM WebSphere Adapters バージョン 7.0 から IBM WebSphere Adapters バージョン 7.5 へのマイグレーション

IBM WebSphere Adapter for JDBC ソフトウェア、バージョン 7.5 から非推奨になった機能のリストについては、『マイグレーションに関する考慮事項』トピックを参照してください。

IBM WebSphere Adapter for JDBC バージョン 7.0 フィーチャー・パック 1 の新機能

IBM WebSphere Adapter for JDBC のフィーチャー・パックが使用可能であり、これによって製品の機能が拡張されます。IBM WebSphere Adapter for JDBC バージョン 7.0 フィーチャー・パック 1 について詳しくは、http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/topic/com.ibm.wsadapters.fep0610.doc/dochome.htm を参照してください。

IBM WebSphere Adapter for JDBC、バージョン 7.0 に含まれている新機能は、以下のとおりです。

- Oracle データベースのテーブル・ビジネス・オブジェクトおよび照会ビジネス・オブジェクト内で配列、テーブル、構造体、ネストされた構造体などのユーザー定義型または複合データ型がサポートされます。
- テーブル・ビジネス・オブジェクト内の XML データ型がサポートされます。
- DateFormat パラメーターを使用して、Date、Time、および Timestamp の各データ型の形式をカスタマイズできます。
- データベースに外部キー参照が定義されている場合、テーブル内の親子関係が自動的に検出されます。

この情報は、IBM® WebSphere® Adapters 製品のサポート Web サイト (http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Family) でも参照できます。このサイトは、定期的に最新の情報に更新されています。

サポートされるその他の機能についての詳しい情報は、 IBM WebSphere Adapter for JDBC ソフトウェアのインフォメーション・センター (http://bidoc.torolab.ibm.com:7500/help/topic/com.ibm.wsadapters.jca.jdbc.doc/doc/stbp_jdb_welcome.html) で参照できます。このサイトは、定期的に最新の情報に更新されています。

ハードウェア要件とソフトウェア要件

WebSphere Adapters のハードウェア要件とソフトウェア要件は、IBM Support Web サイトに記載されています。

WebSphere Adapters のハードウェア要件およびソフトウェア要件を確認するには、http://www.ibm.com/support/docview.wss?uid=swg27006249を参照してください。

追加情報

以下のリンク先には、アダプターの構成およびデプロイに必要となる場合がある追加情報が記載されています。

- WebSphere Business Integration Adapters および WebSphere Adapters の互換性マトリックスによって、ご使用のアダプターで必要となるソフトウェアのサポート対象バージョンが識別されます。この資料を表示するには、WebSphere Adapters のサポート・ページ (http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Family) にアクセスしてください。
- WebSphere Adapters の技術情報には、製品資料に記載されていない回避策および追加情報が記載されています。アダプターの技術情報を参照するには、Web ページ http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm にアクセスし、「**Product category**」リストからアダプターの名前を選択し、検索アイコンをクリックします。

WebSphere Adapter for JDBC の技術概要

アダプターは、JDBC アプリケーション・プログラミング・インターフェース (API) を介してアクセス可能なデータベースと、IBM Business Process Manager または WebSphere Enterprise Service Bus で稼働するアプリケーションとの統合をサポートします。アダプターは、 Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) の下で Outbound および Inbound 処理を提供し、Service Component Architecture (SCA) コンポーネントと統合します。

Outbound 処理により、アプリケーションはデータベースのデータに対してアクセスしたり変更を行ったりできます。アダプターは、アプリケーションからの要求を Outbound 操作に変換し、その操作を実行してデータベースのデータを作成、検索、更新、または削除したり、データベースに格納されたデータベース・プログラムを実行したりします。これらの要求を処理すると、対応するデータベース表の行が作成、検索、更新、または削除されます。また、アダプターによって、データベース

内に定義されたストアード・プロシージャーまたはストアード関数を実行でき、ユーザー定義の SELECT、INSERT、UPDATE、および DELETE ステートメントを実行することができます。アダプターを使用して、同じデータベースに対して複数のアプリケーションを統合できます。

図 1 は Outbound 処理のフローの概要を示したものです。

IBM Business Process Manager または WebSphere Enterprise Service Bus で実行中のアプリケーションが Outbound モジュール内のサービスを呼び出し、このサービスは 1 つ以上のビジネス・オブジェクトを処理する要求をアダプターに送信します。アダプターは JDBC API を使用してデータベース・サーバーに接続し、データベース内のテーブルやその他のオブジェクトにアクセスします。

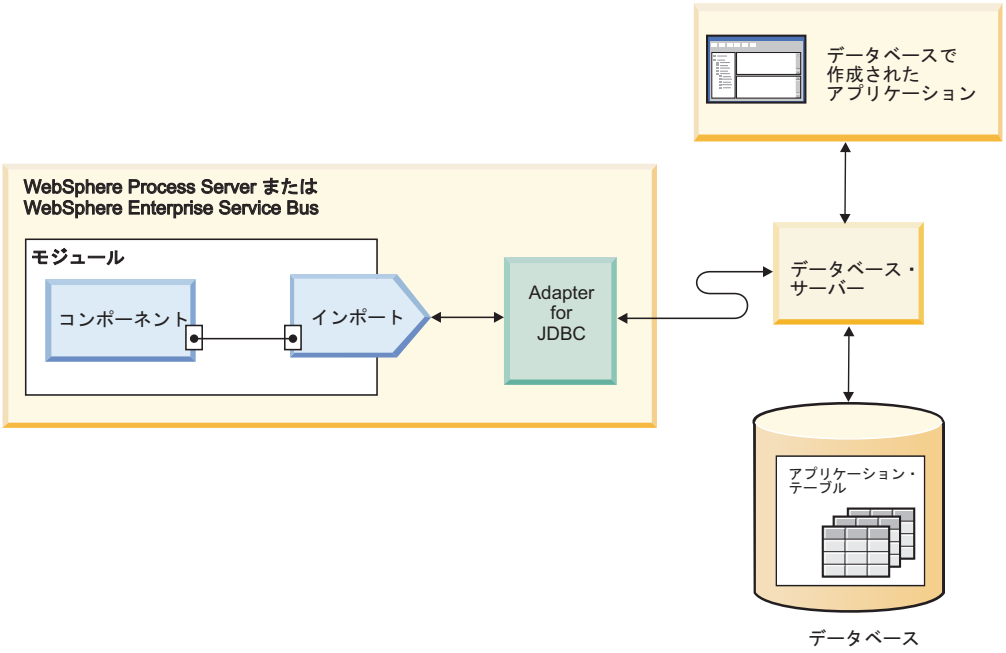


図 1. Outbound 要求の処理

Inbound 処理により、アプリケーションは、データベース内のオブジェクト変更時に通知を受け取ることができます。例えば、選択したデータベース表の行が作成、更新、または削除された場合にアプリケーションに通知できます。

5 ページの図 2 は Inbound 処理のフローの概要を示したものです。

データベース・アプリケーションがデータベース内の表を変更します。変更によってトリガーまたはその他の自動化機構が動作し、変更に関する情報でイベント・ストアを更新します。アダプターは定期的にイベント・ストアをポーリングし、イベントを取得および処理して、IBM Business Process Manager または WebSphere Enterprise Service Bus で実行するアプリケーションの一部であるモジュールのエクスポートにイベントを送達します。

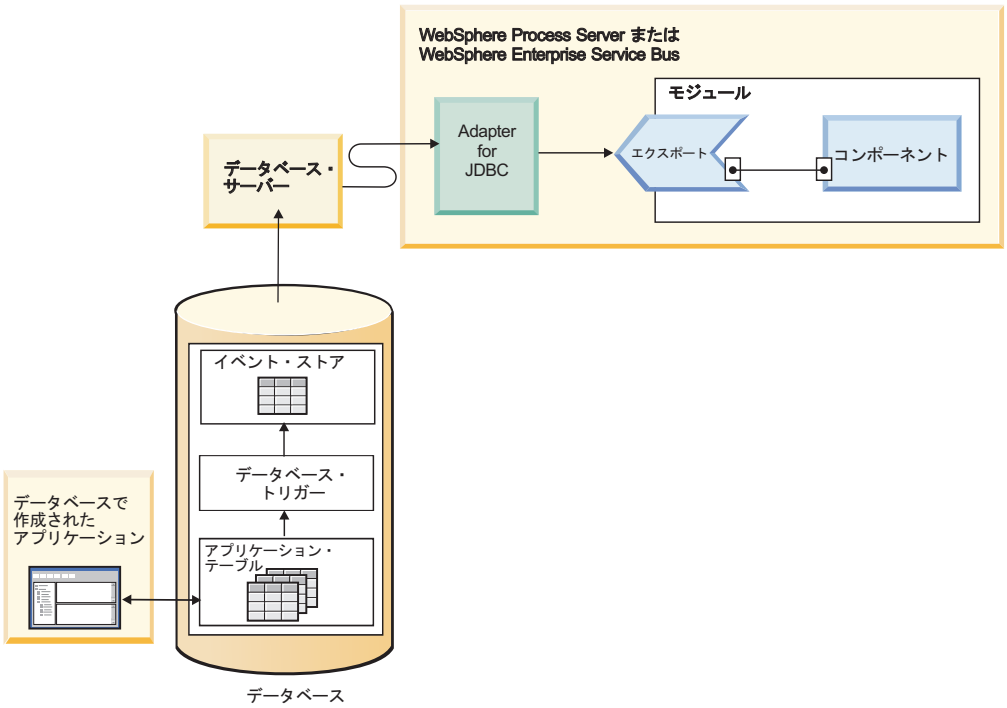


図 2. Inbound イベントの処理

アダプターは、次のいずれかの方法でイベントを処理できます。

- データベース・アプリケーションによって取り込まれたイベント・ストアを使用した標準イベント処理。
- ユーザー定義のデータベース照会を使用したカスタム・イベント処理。

標準イベント処理 では、データベース内のテーブルのデータが変更されると、対応するイベントがキー値などの関連情報と共に、イベント・ストアと呼ばれるデータベース表に挿入されます。変更されたデータを取り込むために、それぞれのテーブルにトリガーを設定することも、Oracle データベースに提供されている Oracle Change Data Capture のような他の方法を使用することもできます。アダプターは、イベント・ストアをポーリングし、数のイベントをひとまとめにして取り出します。イベントは、ビジネス・オブジェクト・タイプ、タイム・スタンプ、およびコネクター ID 別にフィルターに掛けることができます。アダプターは、各イベントを使用して、そのイベントによって変更されたビジネス・オブジェクトを含むビジネス・グラフまたはビジネス・オブジェクトを構成します。ビジネス・オブジェクトまたはビジネス・グラフは、その後、特定のビジネス・オブジェクトを受信するように構成されたエクスポートにディスパッチされます。

カスタム・イベント処理 では、アダプターは、ユーザーが標準 SQL ステートメント、ストアード・プロシージャー、またはストアード関数として指定した照会を実行します。これらのアクションでは、照会によって返されるデータについて、結果セットが戻されます。結果セットの各行は、イベント・ストアの行に対応します。アダプターは、各イベントのビジネス・オブジェクトを構成して、それを特定のビジネス・オブジェクト用に構成された (またはビジネス・オブジェクトをサブスクライブする) エクスポート (エンドポイントとも呼ばれる) に送信します。

標準およびカスタムの両方のイベント処理について、アダプターがイベントをポーリングする頻度と、各ポーリング期間に取得するイベント数を指定することができます。

Outbound 処理

アプリケーション・コンポーネントが、データベースにあるレコードの存在を照会したり、データベースのデータを取得または変更する必要がある場合に、アダプターは、アプリケーション・コンポーネントとデータベースの間のコネクターとして機能します。変更後イメージまたは差分スタイル・ビジネス・オブジェクトのいずれかが処理されます。アダプターの一連の標準 Outbound 操作では、また、アダプターは、Outbound 処理のためにローカル・トランザクションと XA (分散) トランザクションの両方をサポートしています。

アダプターのビジネス・オブジェクト・モデルでは、更新を行うビジネス・オブジェクトとして、変更後イメージと差分という 2 つのスタイルを使用します。*変更後イメージ* ・ビジネス・オブジェクトは、必要なすべての変更が行われた後のビジネス・オブジェクトの完全な状態を含んだものです。*差分* ビジネス・オブジェクトは、キー値および変更対象のデータのみを含んだものです。差分ビジネス・オブジェクトは、ビジネス・オブジェクトを更新する操作でのみ使用されます。

サポートされる操作

表 1 は、ビジネス・オブジェクトの各タイプでサポートされる Outbound 操作をリストしたもので、変更後イメージまたは差分のスタイルの処理をそれぞれサポートするかどうかを示しています。

表 1. ビジネス・オブジェクトの各タイプでサポートされる Outbound 操作

サポートされるビジネス・オブジェクト	操作	変更後イメージのサポート	差分のサポート
テーブル ビュー シノニム ニックネーム	Create	はい	いいえ
	Update	はい	いいえ
	UpdateAll	はい	いいえ
	Delete	はい	いいえ
	DeleteAll	はい	いいえ
	Retrieve	該当なし	該当なし
	RetrieveAll	該当なし	該当なし
	ApplyChanges	はい	はい
	Exists	該当なし	該当なし
	Upsert	はい	いいえ
ストアード・プロシージャー バッチ SQL	Execute	該当なし	該当なし
照会	RetrieveAll	該当なし	該当なし
ラッパー	Create	はい	いいえ
	Update	はい	いいえ
	Delete	はい	いいえ
	Retrieve	はい	いいえ

トランザクション管理

アダプターは、Outbound 処理のためにローカル・トランザクションと XA (分散) トランザクションの両方をサポートしています。このアダプターでは、トランザクションとは、データベースとの独立した相互作用です。トランザクションは、アトミックな単位で実行する、データベースへの複数の操作から構成されます。これらの操作は、データベースの他のユーザーが同時に実行する操作の影響は受けません。

アダプターがトランザクションをサポートするのは、データベース・サーバーがトランザクションをサポートする場合のみです。サポートされるトランザクションのタイプは、ローカル・トランザクションと XA トランザクションです。

- *ローカル・トランザクション* では、1 つのコンポーネントが、単一データベースを使用したトランザクションの開始および終了を定義します。このトランザクションでは、1 フェーズ・コミット・プロトコルを使用します。トランザクションは、データベースによって管理および実行されます。
- *XA トランザクション* では、トランザクションは複数の異種データベースに渡ることができます。このトランザクションでは、グローバル・プロトコル (2 フェーズ・コミット・プロトコル) を使用します。トランザクション・マネージャーがトランザクションを調整します。

XA トランザクション

このアダプターは、Outbound 処理の XA トランザクションをサポートします。XA トランザクション用にアダプターを構成するには、次のいずれかの方法を選択します。

- XA トランザクションをサポートする JNDI データ・ソースの指定 (337 ページの『XA DataSource JNDI 名 (XADataSourceJNDIName)』 プロパティーを使用)
- XA データ・ソースの指定 (335 ページの『XA DataSource 名 (XADataSourceName)』 プロパティーを使用)

XA DataSource JNDI 名プロパティーは、IBM Business Process Manager または WebSphere Enterprise Service Bus 内部で作成されたデータ・ソースを表します。サーバー上に XA トランザクションをサポートする JNDI データ・ソースを定義し、アダプターを構成するときにそのデータ・ソースを指定すると、アダプターは XA トランザクションに関与します。オプションで、XA データ・ソースを使用すると、アダプターは XA トランザクションに関与します。

XA トランザクションの構成方法については、 153 ページの『デプロイメント・プロパティーの設定およびサービスの生成』を参照してください。

Outbound 操作:

アプリケーション・コンポーネントでは、データベースからの取得などのアクションを実行するために操作を使用します。アダプターは特定の Outbound 操作を提供します。サポートされる操作ごとにアダプターがビジネス・オブジェクトをどう処理するかについての詳細を説明します。

操作を実行するには、アダプターによって提供される標準 SQL ステートメントを使用するか、あるいは定義したストアード・プロシージャーを使用します。ストアード・プロシージャーによって、操作を実行したり、操作の前後でカスタム処理を行ったりすることができます。それぞれの操作の実行方法は、各ビジネス・オブジェクト内で構成できます。

Create 操作:

Create 操作は、要求内のビジネス・オブジェクトに対応したデータベース表に行を作成します。階層ビジネス・オブジェクトの場合は、Create 操作によってビジネス・オブジェクトが再帰的に全探索され、階層内の各ビジネス・オブジェクトに対応する行が作成されます。

Create 操作を処理するため、アダプターは次の操作を実行します。

1. ビジネス・オブジェクトがラッパーであるかどうかを確認します。トップレベルのビジネス・オブジェクトがラッパー・ビジネス・オブジェクトの場合、アダプターはこのビジネス・オブジェクトを無視します。ラッパー・オブジェクトの行は作成されません。
2. 所有関係を伴う単一カーディナリティーの各子ビジネス・オブジェクトを、データベース内に再帰的に挿入します。つまり、アダプターは、子ビジネス・オブジェクトと、子および孫に含まれるすべての子ビジネス・オブジェクトを作成します。

   ビジネス・オブジェクト定義上、ある属性がある単一カーディナリティーの子ビジネス・オブジェクトを表すものとされている場合に、その属性が空であると、アダプターはその属性を無視します。ただし、ビジネス・オブジェクト定義により、その属性が子を表すことが必要であるにもかかわらず、子を表していない場合には、アダプターはエラーを戻して処理を停止します。
3. 所有関係を伴わない単一カーディナリティーの各子ビジネス・オブジェクトの有無を検索し、確認します。検索が失敗して、子がデータベース内に存在しないことが示された場合、アダプターはエラーを返して処理を停止します。Retrieve 操作が成功した場合、アダプターは子ビジネス・オブジェクトを再帰的に更新します。

   注: データベースに子ビジネス・オブジェクトが存在する場合に、このアプローチが正しく機能するには、子ビジネス・オブジェクト内の基本キー属性の相互参照が、Create 操作時に正しく行われる必要があります。アプリケーション・データベースに子ビジネス・オブジェクトが存在しない場合、基本キー属性は設定してはいけません。
4. トップレベルのビジネス・オブジェクトをデータベース内に挿入するため、次の操作を実行します。
   a. トップレベルのビジネス・オブジェクトの各外部キー値を、対応する単一カーディナリティーの子ビジネス・オブジェクトの基本キー値に設定します。子ビジネス・オブジェクトの値は、データベース・シーケンスまたはカウンター、あるいはデータベース自体によって、子の作成時に設定される場合があります。そのため、このステップでは、アダプターが親をデータベースに挿入する前に、親の外部キー値を正しいものにします。

b. データベースによって自動的に設定される属性のそれぞれに対して、新しい固有 ID 値を生成します。データベース・シーケンスまたはカウンターの名前は、属性のアプリケーション固有情報に格納されます。データベース・シーケンスまたはカウンターが属性に関連付けられている場合、アプリケーション・サーバーから渡された値があれば、アダプターによって生成された値によって上書きされます。

c. トップレベルのビジネス・オブジェクトをデータベース内に挿入します。

注: アダプターは空の複合列をヌル列として処理します。その値がヌルに設定されているかどうかは関係ありません。

5. 複数カーディナリティーの子ビジネス・オブジェクトのすべてを、次のように処理します。

a. それぞれの子の外部キー値を、親に含まれる対応する基本キー属性の値を参照するように設定します。親の基本キー値は親の作成時に生成されている可能性があるため、この処理によって、アダプターがデータベースに子を挿入する前に、それぞれの子の外部キー値を正しいものにします。

b. 複数カーディナリティーの子ビジネス・オブジェクトのすべてを、データベースに挿入します。

ヌル・データについて詳しくは、 279 ページの『ヌル・データ - よくある質問』を参照してください。

Retrieve 操作:

Retrieve 操作では、データベースからビジネス・オブジェクト階層のデータが抽出されます。

Retrieve 操作を処理するため、アダプターは次の操作を実行します。

1. 受信したトップレベルのビジネス・オブジェクトから、すべての子ビジネス・オブジェクトを削除します。すなわち、子のないトップレベルのビジネス・オブジェクトのコピーを作成します。
2. トップレベルのビジネス・オブジェクトを、データベース内で検索します。Retrieve 操作は基本キーのみを使用するため、最上位ビジネス・オブジェクトに基本キーを指定しておく必要があります。他の列は無視されます。

注: トップレベルのビジネス・オブジェクトの基本キーのみが、SQL ステートメントの where 節の構成に使用されます。

- トップレベルのビジネス・オブジェクトがラッパー・ビジネス・オブジェクトの場合、これは無視されます。ラッパー・ビジネス・オブジェクトの検索は実行されません。
- 検索の結果戻された行が 1 つの場合、アダプターは処理を継続します。
- 検索の結果、戻される行がない場合 (目的のトップレベル・ビジネス・オブジェクトがデータベース内に存在しないことを意味します)、アダプターは RecordNotFoundException エラーを戻します。
- 検索の結果戻された行が複数ある場合、アダプターは MultipleMatchingRecordsException エラーを戻します。

3. 複数カーディナリティーの子ビジネス・オブジェクトのすべてを、再帰的に検索します。

   注: アダプターは、ビジネス・オブジェクトの配列を取り込むときに、一意性を保証しません。一意性の保証は、データベース側で行われなければなりません。データベースから戻された子ビジネス・オブジェクトに重複があると、アダプターは、それらの重複する子を戻します。
4. 子ビジネス・オブジェクトが所有関係にあるかどうかに関係なく、各単一カーディナリティーの子を再帰的に検索します。所有権属性は無視されます。アダプターは、トップレベルのビジネス・オブジェクトの外部キーを使用して、子ビジネス・オブジェクトを検索します。子ビジネス・オブジェクトに構成された値はすべて無視されます。トップレベルのビジネス・オブジェクトに外部キーが構成されていて、子ビジネス・オブジェクトにレコードが検出されない場合、この操作は NULL のビジネス・オブジェクトを返します。

   注: 単一カーディナリティーの子ビジネス・オブジェクトは、ビジネス・オブジェクト内での出現順序に従って、親ビジネス・オブジェクトが処理された後に処理されます。

NULL データの取得

アダプターは、データベース表で列値が NULL のレコードを取得できます。例えば、Customer ビジネス・オブジェクトに、custid、ccode、fname、および lname という列があり、custid と ccode が複合キーを形成しているとします。複合キーとは、複数の属性を参照する基本キーであり、ビジネス・オブジェクトの一意性を定義するときに使用されます。ccode が NULL の Customer レコードを取得できます。アダプターは Retrieve 操作の SELECT ステートメントを次のように生成します。

```
select custid, ccode, fname, lname from customer where custid=? and ccode is NULL
```

ヌル・データについて詳しくは、279 ページの『ヌル・データ - よくある質問』を参照してください。

RetrieveAll 操作:

アダプターは、RetrieveAll 操作を使用してデータベースからビジネス・オブジェクトの配列を検索します。アダプターが使用するプロセスは、RetrieveAll 操作がデータベース表ビジネス・オブジェクトとユーザー指定 SQL ビジネス・オブジェクトのいずれを対象としているかによって異なります。

データベース表ビジネス・オブジェクトの場合

着信ビジネス・オブジェクト内に取り込まれているキー属性および非キー属性によって、取得のための選択基準が決まります。選択した属性によっては、アダプターは、データベースからトップレベルのビジネス・オブジェクトの複数の行を検索する場合もあります。トップレベルのビジネス・オブジェクトに指定された値はすべて使用されます。子ビジネス・オブジェクトの設定は無視されます。着信ビジネス・オブジェクト内に属性が取り込まれていない場合は、データベース内のそれぞれの表からすべての行が検索されます。

注: 子ビジネス・オブジェクト属性または操作用のユーザー定義照会基準を使用したレコード照会を行うために、SQL の機能を最大限に使用できます。

生成されるビジネス・オブジェクトの名前は、データベースの表の名前と一致します。例えば、データベース中の Customer 表は、「Customer」という名前のビジネス・オブジェクトとして表されます。

ビジネス・オブジェクトの配列を取得するため、アダプターは次の操作を実行します。

1. 取得したすべての行についてコンテナー・ビジネス・オブジェクトを構成します。コンテナー・ビジネス・オブジェクトの名前は、ビジネス・オブジェクトの名前にストリング「Container」を付加したものです。
2. ビジネス・グラフを使用するようモジュールが構成された場合 (これはオプションです)、取得した各行ごとに 1 つのトップレベルのビジネス・グラフを構築します。ビジネス・グラフの名前は、ビジネス・オブジェクト名にストリング「BG」を付加したものです。
3. Retrieve 操作を使用してコンテナー内の各ビジネス・グラフを取得します。

以下の図は、RetrieveAll 操作で戻されるオブジェクトの構造 (ビジネス・グラフがある場合とない場合) を示しています。

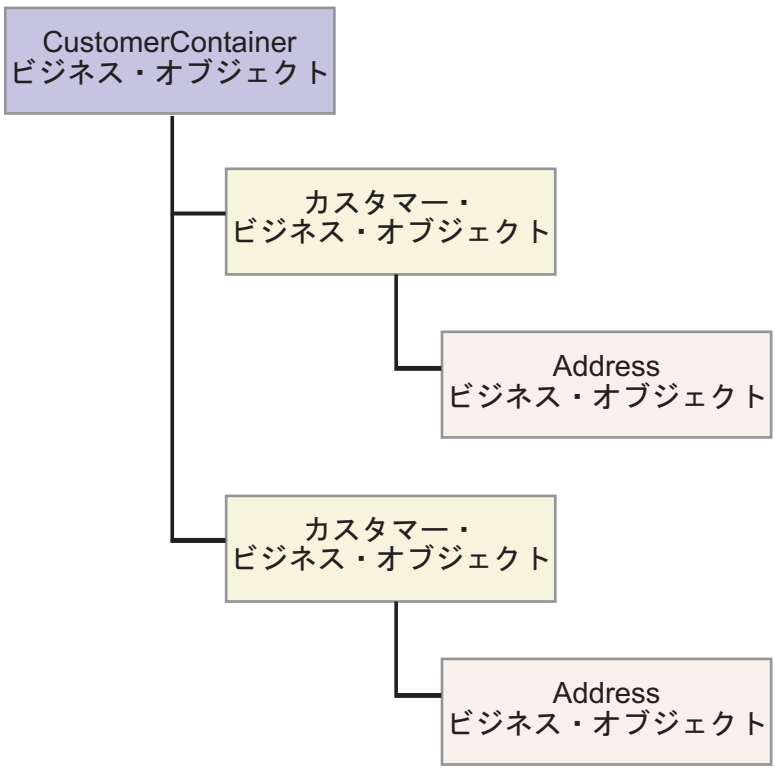


図 3. RetrieveAll 操作で戻されるビジネス・オブジェクトの構造 (オプションのビジネス・グラフなし)

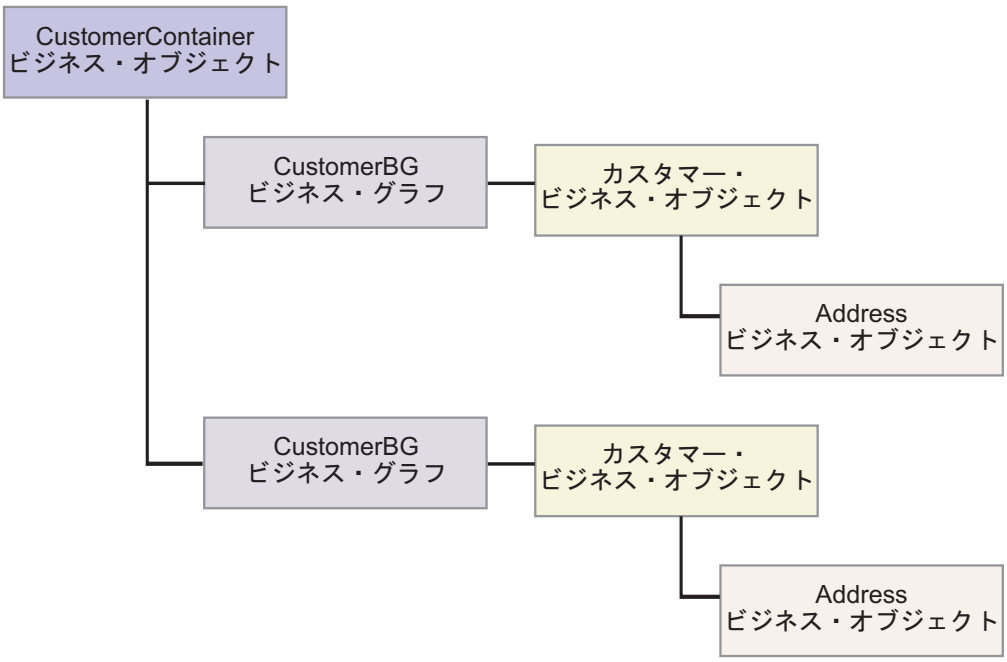


図 4. RetrieveAll 操作で戻されるビジネス・オブジェクトの構造 (オプションのビジネス・グラフあり)

RetrieveAll 操作によって、次のエラーが発生する可能性があります。

- `RecordNotFoundException` – 入力オブジェクト内の、データが取り込まれている 1 つ以上のビジネス・オブジェクトが、エンタープライズ情報システムに存在しないときに、ErrorOnEmptyResultset プロパティーの管理接続ファクトリー・プロパティーが True に設定されていると、この例外が生成されます。RecordNotFoundException なしで空の結果セットを取得するには、 .import ファイルを変更して 330 ページの『レコードが一切検出されない (ErrorOnEmptyResultSet) 場合は例外をスローします。』 プロパティーを False に設定するか、または、IBM Business Process Manager に Outbound アプリケーションをデプロイした後で、 MCF プロパティー 330 ページの『レコードが一切検出されない (ErrorOnEmptyResultSet) 場合は例外をスローします。』 を False に構成することができます。
- `MatchesExceededLimitException` – データベース内の一致するレコードの数が、対話仕様に定義された 341 ページの『返されるレコードの最大数』 プロパティーの値を超えると、この例外が生成されます。フォールトの MatchCount 属性にアダプターがデータベースで検出した一致の実数が含まれています。それを参照して、制限値を高くするか、あるいは検索を詳細化することができます。

  注: 341 ページの『返されるレコードの最大数』 のプロパティーを大きな数に設定すると、戻されるビジネス・オブジェクトのサイズと数によっては、メモリー不足による問題が発生する場合があります。
- `EISSystemException` – データベース (エンタープライズ情報システム) から 1 つ以上の回復不能エラーが報告されると、この例外が生成されます。

照会ビジネス・オブジェクトの場合

ユーザー指定 SELECT ステートメント (照会ビジネス・オブジェクト) に対して作成されたビジネス・オブジェクトも RetrieveAll 操作をサポートします。外部サービス・ウィザードは、ユーザー指定の SQL SELECT ステートメントを実行して、照会ビジネス・オブジェクトの階層を作成することによって、照会ビジネス・オブジェクトを生成します。

オプションのビジネス・グラフを使用する場合、階層は図 5に示すようになります。

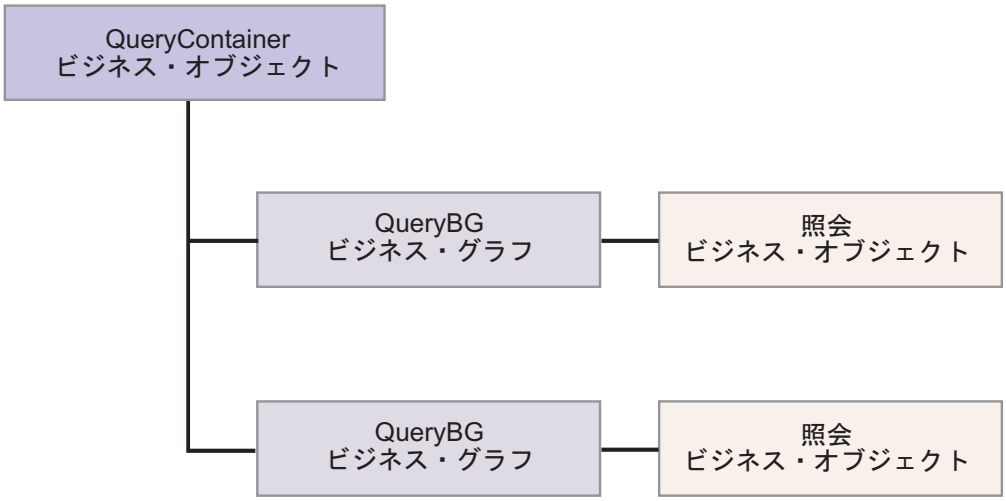


図 5. ユーザー指定の照会ビジネス・オブジェクト

オプションのビジネス・グラフを使用しない場合、階層は図 6に示すようになります。

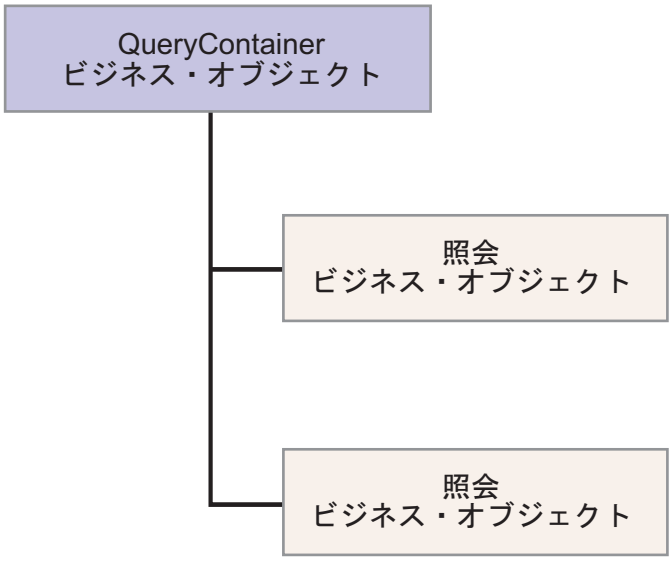


図 6. ユーザー指定の照会ビジネス・オブジェクト

外部サービス・ウィザードによってユーザー指定 SELECT ステートメントに対して生成された照会ビジネス・オブジェクトを処理するために、アダプターは以下の操作を実行します。

1. 照会ビジネス・オブジェクトから SELECT SQL ステートメントを取得します。
2. 照会ビジネス・オブジェクトで動的 WHERE 節が指定されているかどうかを判定します。
   - 動的 WHERE 文節がある場合、アダプターは、SELECT ステートメントのデフォルト WHERE 文節をその動的 WHERE 文節で置換します。
   - 動的 WHERE 文節がない場合、アダプターは、SELECT ステートメントのパラメーターを照会ビジネス・オブジェクトで指定された対応する値で置換します。
3. SELECT ステートメントを実行します。
4. 返された結果セットを取得し、照会ビジネス・オブジェクト値にデータベースから返されたデータを設定し、 13 ページの図 5に示した構造のコンテナー・ビジネス・オブジェクトを作成します。
5. 照会ビジネス・オブジェクトに子ビジネス・オブジェクトが定義されている場合、コンテナー内の各トップレベルの照会ビジネス・オブジェクトからなる階層全体を取得します (*下降検索*)。

   注: 照会ビジネス・オブジェクトをトップレベルのビジネス・オブジェクト以外にすることはできません。照会ビジネス・オブジェクトが子照会ビジネス・オブジェクトを持つことはできません。

NULL オブジェクトの取得

アダプターは、列値が NULL のレコードをデータベース表から取得できます。例えば、Customer ビジネス・オブジェクトに custid、ccode、fname、および lname という列があり、ccode は基本キーである必要はないとします。ccode 列が NULL である Customer レコードをすべて検索する照会を実行できます。アダプターは、RetrieveAll 操作の選択照会を次のように生成します。

```
select custid, ccode, fname, lname from customer where custid=? and ccode is NULL
```

ヌル・データについて詳しくは、 279 ページの『ヌル・データ - よくある質問』を参照してください。

Update 操作:

Update 操作は、ソース・ビジネス・オブジェクトを、トップレベルのソース・ビジネス・オブジェクトで指定された基本キーを使用してデータベースから検索されたビジネス・オブジェクトと比較することによって実行されます。

階層ビジネス・オブジェクトの更新時に、アダプターは次の操作を実行します。

1. ソース・ビジネス・オブジェクトの基本キー値を使用して、データベース内の対応するエンティティーを検索します。検索されたビジネス・オブジェクトは、データベース内のデータの現在の状態を正確に表したものです。

   検索が失敗した場合 (トップレベルのビジネス・オブジェクトがデータベース内に存在しないことを意味します)、アダプターは `RecordNotFoundException` エラーを戻し、更新は失敗します。

   検索に成功した場合、アダプターは、検索されたビジネス・オブジェクトをソース・ビジネス・オブジェクトと比較して、どの子ビジネス・オブジェクトに関し

てデータベースに変更を加える必要があるかを判別します。ただし、アダプターは、ソース・ビジネス・オブジェクトの単純属性の値と検索されたビジネス・オブジェクトの単純属性の値を比較しません。アダプターは、非キーの単純属性すべての値を更新します。

トップレベルのビジネス・オブジェクトのすべての単純属性がキーを表している場合、アダプターはそのトップレベルのビジネス・オブジェクト用の更新照会を生成できません。この場合、アダプターは、警告を記録してから次に進みます。

2. トップレベルのビジネス・オブジェクトの子のうち、単一カーディナリティーのものすべてを再帰的に更新します。

   ビジネス・オブジェクト定義上、ある属性がある子ビジネス・オブジェクトを表すことが必須である場合には、その子ビジネス・オブジェクトがソース・ビジネス・オブジェクトと検索されたビジネス・オブジェクトの両方に存在している必要があります。存在しない場合、Update 操作は失敗し、アダプターはエラーを戻します。

   アダプターでは、所有関係にある単一カーディナリティーの子を、次のいずれかの方法で処理します。

   - ソース・ビジネス・オブジェクトおよび検索したビジネス・オブジェクトの両方に子が存在する場合、アダプターはデータベース内の既存の子を更新する代わりに、既存の子を削除して新規の子を作成します。
   - その子がソース・ビジネス・オブジェクトには存在するにもかかわらず、検索されたビジネス・オブジェクトには存在しない場合、アダプターはデータベース内にその子を再帰的に作成します。
   - その子が検索されたビジネス・オブジェクトには存在するにもかかわらず、ソース・ビジネス・オブジェクトには存在しない場合、アダプターはデータベース内のその子を再帰的に削除します。

   所有関係にない単一カーディナリティーの子に関しては、アダプターは、ソース・ビジネス・オブジェクトに存在するそのような子のすべてを、データベースから検索しようとします。アダプターは、子の検索に成功すると、その子ビジネス・オブジェクトにデータを取り込みますが、更新は行いません。これは、所有関係にない単一カーディナリティーの子はアダプターによって変更されることがないためです。

3. 検索されたビジネス・オブジェクトのすべての単純属性を更新します。ただし、ソース・ビジネス・オブジェクト内の対応する属性が指定されていない場合を除きます。

   更新されるビジネス・オブジェクトは一意である必要があるため、アダプターは、結果として 1 行のみが処理されることを確認します。複数の行が戻されている場合、アダプターはエラーを戻します。

   トップレベルのビジネス・オブジェクトがラッパー・ビジネス・オブジェクトの場合、これは無視されます。ラッパー・ビジネス・オブジェクトの更新は実行されません。

4. 検索されたビジネス・オブジェクトの複数カーディナリティーの子のそれぞれを、次のいずれかの方法で処理します。

- その子がソース・ビジネス・オブジェクトの配列と検索されたビジネス・オブジェクトの配列の両方に存在する場合、アダプターはデータベース内でその子を再帰的に更新します。
- その子がソース・ビジネス・オブジェクトの配列には存在しても、検索されたビジネス・オブジェクトの配列には存在しない場合、アダプターはデータベース内でその子を再帰的に作成します。
- その子が検索されたビジネス・オブジェクトの配列には存在しても、ソース・ビジネス・オブジェクトの配列には存在しない場合、アダプターはデータベースからその子を再帰的に削除します。ただし、親に含まれているその子を表す属性のアプリケーション固有情報で、KeepRelationship プロパティーが `true` に設定されている場合を除きます。この場合、アダプターは、データベースからその子を削除しません。

NULL データと Update 操作

アダプターは、データベース表で列値が NULL のレコードを更新できます。例えば、Customer ビジネス・オブジェクトに、custid、ccode、fname、および lname という列があり、custid と ccode が複合キーを形成しているとします。複合キーとは、複数の属性を参照する基本キーであり、ビジネス・オブジェクトの一意性を定義するときに使用されます。ccode が NULL の Customer レコードを更新できます。アダプターにより、Update 操作の更新照会が次のように生成されます。

```
update customer set fname=?, lname=? where custid=? and ccode is null
```

注: アダプターは空の複合列をヌル列として処理します。その値がヌルに設定されているかどうかは関係ありません。

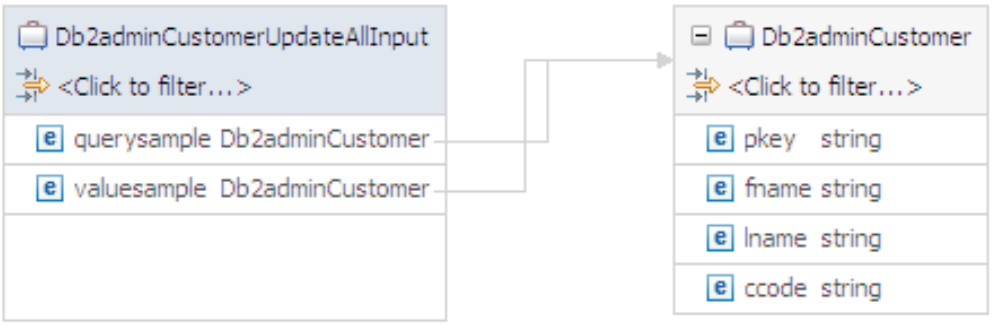
ヌル・データについて詳しくは、 279 ページの『ヌル・データ - よくある質問』を参照してください。

UpdateAll 操作:

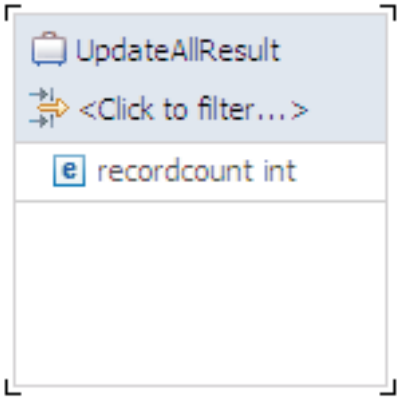
UpdateAll 操作は、複数のデータベース・レコードを一度に更新し、いくつのレコードが更新されたのかを示すレコード数を戻します。

UpdateAll 操作は、入力パラメーターとして UpdateAllInput ビジネス・オブジェクトを使用します。UpdateAllInput ビジネス・オブジェクトには 2 つの子ビジネス・オブジェクト属性 `querysample` と `valuesample` があります。これらの属性は、テーブル・ビジネス・オブジェクト・タイプに属します。アダプターは、更新が必要なレコードをフィルタリングするために `querysample` 属性を使用し、一致するすべてのレコードを更新するために `valuesample` 属性を使用します。

以下の画面は、UpdateAllInput ビジネス・オブジェクトの構造を示しています。このビジネス・オブジェクトの名前はスキーマ名を含み、その後にデータベース表名があり、さらにビジネス・オブジェクト名が続きます (<schemaName><tableName>UpdateAllInput)。



UpdateAll 操作の出力は UpdateAllResult ビジネス・オブジェクトであり、このビジネス・オブジェクトには recordcount 属性が含まれています。以下の画面は、UpdateAll 操作から返される UpdateAllResult ビジネス・オブジェクトの構造を示しています。



UpdateAll 操作を処理するため、アダプターは次の操作を実行します。

- アダプターは UpdateAllInput ビジネス・オブジェクト内の属性を使用して、1 つの UPDATE SQL ステートメントを生成します。UPDATE SQL ステートメントの照会規則 (WHERE 節) は querysample 属性に基づいて生成され、更新規則 (SET 節) は valuesample 属性に基づいて生成されます。

  valuesample のすべての属性が設定されていない場合、アダプターは MissingData フォールトを戻します。
- アダプターは SQL ステートメントを実行して、一致するすべてのレコードを更新します。

  ビジネス・オブジェクトが階層型の場合、トップレベルのビジネス・オブジェクト (階層型ビジネス・オブジェクトのトップレベルにある個別のビジネス・オブジェクト) のみが更新されます。階層型ビジネス・オブジェクト間の関係を UpdateAll 操作によって変更しようとしていて、その操作がデータベースのデータ保全性制約に違反する場合は、アダプターは IntegrityConstraintViolation フォールトを戻します。

valuesample に基本キー属性が設定されている場合、アダプターは対応する列を更新します。ただし、複数のレコードが一致し、1 次キー制約がデータベース中に定義されている場合、アダプターは UniqueConstraint フォールトを生成します。

- アダプターは、UPDATE SQL ステートメントの実行結果を取得し、UpdateAllResult ビジネス・オブジェクトに recordcount 属性を設定します。

  例えば、アダプターは、更新されたレコードが 10 レコードあると判定した場合は recordcount パラメーターを recordcount=10 と設定します。

  照会規則に一致するレコードがない場合、更新されるレコードはなく、recordcount 出力パラメーターは 0 です。

  注: UpdateAll 操作または DeleteAll 操作中に影響を受けるレコードがない場合はアダプターが RecordsNotFoundException を戻すようにしたい場合、外部サービス・ウィザードの「複合プロパティーの指定」ウィンドウで「**UpdateAll または DeleteAll 操作中に影響を受けるレコードがない場合は例外を返す**」チェック・ボックスを選択する必要があります。

注: 子ビジネス・オブジェクト属性または操作用のユーザー定義照会基準を使用したレコード照会を行うために、SQL の機能を最大限に使用できます。

ApplyChanges 操作:

ApplyChanges 操作では、ビジネス・オブジェクトの変更または削除のための差分および変更後イメージをサポートします。ApplyChanges 操作は、ビジネス・グラフを使用する場合にのみ使用可能です。

ビジネス・グラフの verb プロパティーを、create、update、delete などの操作の名前に設定した場合、アダプターは ApplyChanges 操作について変更後イメージ処理を実行します。例えば、verb を create に設定した場合、アダプターは ApplyChanges 操作を Create 操作と同様に処理します。

ビジネス・グラフで verb を設定しない場合、アダプターはビジネス・グラフの ChangeSummary を使用してビジネス・オブジェクトを更新します。このモードでは、ApplyChanges 操作は以下の点で Update 操作と異なります。

- ApplyChanges 操作では、更新の前に Retrieve 操作は実行されません。
- 着信ビジネス・オブジェクトとデータベース内のビジネス・オブジェクトの比較が行われません。
- 子はすべて、各子ビジネス・オブジェクトの ChangeSummary に設定されている操作に基づいて処理されます。子に操作が設定されていない場合、アダプターはエラーを戻します。

アダプターは、ChangeSummary からの階層ビジネス・オブジェクトの更新時に、以下のステップを実行します。このステップでは、ChangeSummary から、変更内容のみが処理されます。

1. 親オブジェクトの子のうち、単一カーディナリティーのものすべてを再帰的に処理します。ビジネス・オブジェクト仕様で必須にマークされている子は、Inbound オブジェクトに存在していなければなりません。存在しない場合、ApplyChanges 操作は失敗し、アダプターはエラーを戻します。

2. 親に含まれる外部キー値のうち、単一カーディナリティーの子の属性を参照するものすべてを、それぞれ対応する子の値に設定します。この処理が必要なのは、これ以前のステップで、単一カーディナリティーの子がデータベースに追加され、新しいシーケンス値が生成されている可能性があるためです。
3. 現在処理中のオブジェクトを、SQL UPDATE ステートメントまたはストアード・プロシージャーを使用して更新します。個々のビジネス・オブジェクトのすべての単純属性が更新されます。アダプターは UPDATE ステートメントにどの属性を追加しなければならないかを判断するのに、プロパティー・レベルの変更を使用しません。すべて更新されます。更新されているオブジェクトは固有であるため、アダプターは結果として 1 行のみが処理されていることをチェックします。複数の行が処理される場合、エラーが戻されます。
4. 現在のオブジェクトのカーディナリティー N のすべての子にある、親の属性を参照する外部キー値をすべて、対応する親の値に設定します。通常、これらの値はデータ・マッピング時に既に相互参照されています。ただし、これはカーディナリティーが N のコンテナーに含まれる新しい子には該当しない場合があります。このステップにより、カーディナリティーが N の子すべての外部キー値が正しい値になってから、それらの子の更新が行われることが徹底されます。
5. 現在のオブジェクトの、カーディナリティーが N のコンテナーをすべて更新します。

   子オブジェクトが処理されるときには、それぞれの子操作が取得され、適切な操作が実行されます。ApplyChanges で子に対して許可される操作は、Create、Delete、および Update です。

   - Create 操作が子で検出された場合、それが所有関係にある子であれば、検出された子がデータベース内に作成されます。所有関係にない子に関しては、検索により、データベースに存在するかどうかが確認されます。
   - Delete 操作が子で検出された場合、子は削除されます。
   - Update 操作が子で検出された場合、子はデータベース内で更新されます。

Delete 操作:

Delete 操作は、データベースからの着信ビジネス・オブジェクトのプルーニングと、その後の完全なビジネス・オブジェクトの検索によって実行されます。 Delete 操作は、その後階層内の各ビジネス・オブジェクトについて再帰的に適用されます。

Delete 操作では、ビジネス・オブジェクトのアプリケーション固有情報の StatusColumnName の値に応じて、物理削除と論理削除がサポートされます。ステータス列名の値が定義されている場合、アダプターは論理削除操作を実行します。ステータス列名の値が定義されていない場合、アダプターは物理削除操作を実行します。

物理削除

物理削除の場合、アダプターは次の処理を行います。

- 複数カーディナリティーの子ビジネス・オブジェクトすべてを再帰的に削除します。
- トップレベルのビジネス・オブジェクトを削除します。

トップレベルのビジネス・オブジェクトがラッパー・オブジェクトの場合、これは無視されます。ラッパー・ビジネス・オブジェクトの削除は実行されません。

- 所有関係にある単一カーディナリティーの子ビジネス・オブジェクトすべてを、再帰的に削除します。

論理削除

論理削除の場合、アダプターは次の処理を行います。

- ビジネス・オブジェクトの状況属性を、ビジネス・オブジェクト・レベルのアプリケーション固有情報によって指定されている値に設定する Update を発行します。アダプターでは、結果として 1 つのデータベース行だけが更新されることを確認します。それ以外の場合は、エラーを戻します。
- 所有関係にある単一カーディナリティーの子のすべて、および複数カーディナリティーの子のすべてに対し、論理削除を再帰的に実行します。アダプターは、所有関係にない単一カーディナリティーの子は削除しません。

NULL データと Delete 操作

アダプターは、データベース表で列値が NULL のレコードを削除できます。例えば、Customer ビジネス・オブジェクトに、custid、ccode、fname、および lname という列があり、custid と ccode が複合キーを形成しているとします。複合キーとは、複数の属性を参照する基本キーであり、ビジネス・オブジェクトの一意性を定義するときに使用されます。ccode が NULL の Customer レコードを削除できます。アダプターは、Delete 操作の削除照会を次のように生成します。

```
delete from customer where custid=? and ccode is null
```

ヌル・データについて詳しくは、 279 ページの『ヌル・データ - よくある質問』を参照してください。

DeleteAll 操作:

DeleteAll 操作は、複数のデータベース・レコードを一度に削除し、いくつのレコードが削除されたのかを示すレコード数を戻します。

DeleteAll 操作を処理するため、アダプターは次の操作を実行します。

- アダプターは、入力テーブル・ビジネス・オブジェクトの属性を使用して、SQL ステートメントを生成します。
- アダプターは、生成した SQL ステートメントを実行して、ビジネス・オブジェクト内に設定された属性に一致するすべてのレコードを削除します。

  ビジネス・オブジェクトが階層型の場合、トップレベルのビジネス・オブジェクト (階層型ビジネス・オブジェクトのトップレベルにある個別のビジネス・オブジェクト) のみが削除されます。操作がデータベースのデータ保全性制約に違反する場合、アダプターは IntegrityConstraint フォールトを返します。この例外を回避するには、階層型ビジネス・オブジェクトに対して特定の順序で DeleteAll 操作を複数回実行する必要があります。例えば、CUSTINFO 表が CUSTOMER 表への外部キー参照を持っている場合、最初に CUSTINFO 表のビジネス・オブジェクトに対して DeleteAll 操作を実行し、その後で CUSTOMER 表のビジネス・オブジェクトに対して DeleteAll 操作を実行する必要があります。

- アダプターは、DELETE SQL ステートメントの実行結果を取得し、DeleteAllResult ビジネス・オブジェクトに recordcount 属性を設定します。

  例えば、アダプターは、削除されたレコードが 10 レコードあると判定した場合は recordcount パラメーターを recordcount=10 と設定します。

  照会規則に一致するレコードがない場合、削除されるレコードはなく、recordcount 出力パラメーターは 0 です。

  注: UpdateAll 操作または DeleteAll 操作中に影響を受けるレコードがない場合はアダプターが RecordsNotFoundException を戻すようにしたい場合、外部サービス・ウィザードの「複合プロパティーの指定」ウィンドウで「**UpdateAll または DeleteAll 操作中に影響を受けるレコードがない場合は例外を返す**」チェック・ボックスを選択する必要があります。

注: 子ビジネス・オブジェクト属性または操作用のユーザー定義照会基準を使用したレコード照会を行うために、SQL の機能を最大限に使用できます。

Execute 操作:

ストアード・プロシージャーおよびストアード関数のほかに、ラッパー・ストアード・プロシージャーおよびラッパー・ストアード関数を実行するときにも Execute 操作が使用されます。外部サービス・ウィザードは、データベースのストアード・プロシージャーまたはストアード関数の定義に対応する必要なストアード・プロシージャー・ビジネス・オブジェクトを生成します。アダプターは、Execute 操作を使用してストアード・プロシージャー・ビジネス・オブジェクトを処理します。

ストアード・プロシージャー、そのストアード・プロシージャーから構成されるビジネス・オブジェクト、および Execute 操作でストアード・プロシージャー・ビジネス・オブジェクトを処理するためにアダプターが使用するステップの単純な例を以下に示します。

ストアード・プロシージャーの単純な例:

```
PROCEDURE testSP(IN int x,INOUT VARCHAR(10) msgSTR, OUT int status,
                 OUT struct outrec, OUT array retArr)
```

このプロシージャーは 2 つの結果セットを戻します。

このストアード・プロシージャーの場合に構成されるビジネス・オブジェクトの例を以下に示します。

```
BOLevel ASI
      SPName=testSP
      ResultSet=true
      MaxNumberOfResultSets=2
      ReturnValue = propName
                    ストアード・プロシージャーが関数の場合に戻されます。
                    戻り値が複合型 (array/struct/resultset) である場合は、
                    子ビジネス・オブジェクトに対応するプロパティー名になります。
          関数の場合にのみ定義されます。

プロパティー
      x Type=IP
      msgStr Type=IO
      status Type=OP
      outrec Type OP - outrec の子 BO、ASI ChildBOType = struct
```

```
retarr Type OP - retArr の n カーディナリティー子 BO、ASI ChildBOType = array
childBOName1 - 最初の結果セットに対する子 BO、ASI ChildBOType = resultset
childBOName2 - 2 番目の結果セットに対する子 BO、ASI ChildBOType = resultset
```

Execute 操作でこのストアード・プロシージャー・ビジネス・オブジェクトを処理するために、アダプターは以下の処理を行います。

1. ストアード・プロシージャー呼び出し CALL testSP(x, msgStr, status, outrec, retArr) を構成します。
2. 呼び出し可能ステートメントで入力パラメーター x および msgStr を設定します。
3. 呼び出し可能ステートメントを実行します。
4. 戻り値を取得し (関数の場合)、それがスカラー値であればその値を適切な属性に設定します。複合値 (構造体や配列など) であれば子ビジネス・オブジェクトに設定します。
5. 最初の結果セットを取得し、ResultSet1 のコンテナーを作成します。
6. 2 番目の結果セットを取得し、ResultSet2 のコンテナーを作成します。
7. 出力パラメーター msgStr および status を取得し、ビジネス・オブジェクトで対応する属性を設定します。
8. 出力パラメーター outrec を取得し、outrec で返されたデータから子ビジネス・オブジェクトを作成します。outrec がネストされた構造体型である場合、アダプターは階層子ビジネス・オブジェクトを再帰的に作成してデータを格納します。
9. 出力パラメーター retArr を取得し、retArr で返されたデータから複数のカーディナリティーの子ビジネス・オブジェクトを作成します。retArr がネストされた配列型である場合、アダプターは階層子ビジネス・オブジェクトを再帰的に作成してデータを格納します。

Exists 操作:

Exists 操作は、ビジネス・オブジェクトに設定された属性と一致するレコードが、データベースに含まれているかどうかを判断します。

選択基準には、キー属性および非キー属性のどちらも使用できます。

注: 外部サービス・ウィザードを使用してデータベース内のテーブル・オブジェクトをディスカバーする場合、複数のテーブルを選択し、それらのテーブルを「オブジェクトのディスカバリーと選択」画面の「選択されたオブジェクト」領域に追加することができます。ただし、外部サービス・ウィザードを使用して、選択したテーブルをリンクあるいは結合することはできません。ビジネス・アプリケーションの目的として、テーブル・ビジネス・オブジェクトで、結合テーブルに対する Exists 操作の実行が必要な場合は、データベース内のテーブルを結合し、結合テーブルのビューを作成しておく必要があります。結合テーブルのビューを作成してからであれば、そのビューに対するディスカバリーを実行できます。そのビューに対しては Exists 操作がサポートされます。

指定されたビジネス・オブジェクト属性に基づいて Exists 操作を実行し、結果を送信する際、アダプターは以下のアクションを行います。

1. アダプターは、インポートからテーブル・ビジネス・オブジェクトを受信します。このビジネス・オブジェクトは、フラット (単純。子ビジネス・オブジェクトがない)、あるいは階層型 (複雑。1 つ以上の子ビジネス・オブジェクトを含む) の可能性があります。

   ビジネス・オブジェクトが階層型の場合、アダプターが照会をビルドするのは、トップレベルのビジネス・オブジェクト (階層型ビジネス・オブジェクトのトップレベルにある個別のビジネス・オブジェクト) に対してのみです。

   注: Exists 操作をサポートする入力ビジネス・オブジェクトは、ビジネス・オブジェクトのタイプによって異なります。 Exists 操作は、テーブル・ビジネス・オブジェクトでのサポートに加えて、ビュー・ビジネス・オブジェクト、シノニムおよびニックネームのビジネス・オブジェクトでもサポートされています。
2. アダプターはテーブル・ビジネス・オブジェクトを使用して、サーバーに送信する SQL SELECT ステートメントを生成します。

   使用される SQL SELECT ステートメントを以下に示します。

   ```
   select count(*) from TABLENAME where column1=? AND column2=?
   ```

   以下に、今回の例でのサンプル SQL ステートメントを示します。

   ```
   select count(*) from CUSTOMER where fname='John' AND lname='Smith'
   ```

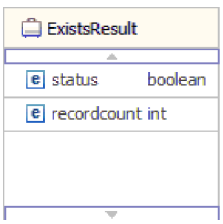
   この場合、SQL ステートメントでは、*非基本キー* 属性の `fname` と `lname` に、`John` と `Smith` という値を割り当てて指定しています。

   アダプターは、テーブル・ビジネス・オブジェクトの属性情報を SQL 照会の *where* 節に組み込みます。
3. データベース・サーバーが、SQL 照会を実行し、結果をアダプターに送信します。
4. アダプターは、SQL 照会の結果をデータベース・サーバーから取得し、**ExistsResults** ビジネス・オブジェクトの `recordcount` 属性と `status` 属性に設定します。

   例えば、ビジネス・オブジェクト内の属性および値の設定と一致するレコードが 2 つ存在することが Exists 操作によって判明した場合、アダプターは `status=true` と `recordcount=2` を設定します。

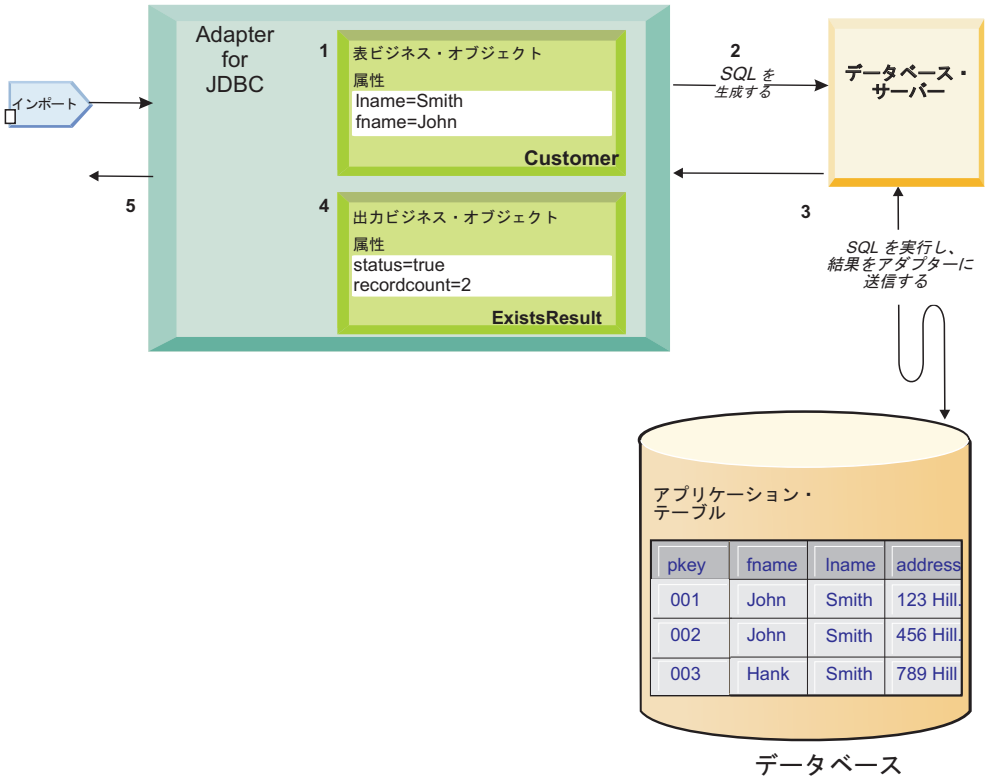
   指定された属性を持つレコードが検出されない場合、 `status` 出力パラメーターは **false** に、`recordcount` 出力パラメーターは **0** になります。

   以下の画面取りは、Exists 操作から返された ExistsResult ビジネス・オブジェクトの構造を示しています。



5. アダプターから呼び出し元に、ExistsResult ビジネス・オブジェクトが返されます。

以下の図は、アダプターが Exists 操作を使用してテーブル・ビジネス・オブジェクトを処理する方法を示しています。



注: 子ビジネス・オブジェクト属性または操作用のユーザー定義照会基準を使用したレコード照会を行うために、SQL の機能を最大限に使用できます。

ヌル・データについては、 279 ページの『ヌル・データ - よくある質問』を参照してください。

Upsert 操作:

Upsert 操作は、テーブル内のレコードを更新または挿入します。照会に一致するレコードがテーブルにない場合、 Upsert 操作は新規レコードを挿入します。照会に一致するレコードが既にテーブルにある場合、Upsert 操作はその既存レコードを更新します。

単一カーディナリティー関係

表 2. 単一カーディナリティー関係での Upsert 操作の動作

アプリケーション固有情報	Upsert 操作の動作
Ownership = True かつ KeepRelationship = True	• 親と子のビジネス・オブジェクトが両方とも存在しない場合、それらが作成されます。 • 親と子のビジネス・オブジェクトが両方とも存在する場合、それらが更新されます。 • 親ビジネス・オブジェクトが存在し、子ビジネス・オブジェクトが存在しない場合、親が更新され、子が作成されます。親が同じ子テーブル・レコードへの変更はありません。 • 親ビジネス・オブジェクトが存在せず、子ビジネス・オブジェクトが存在する場合、親が作成され、アダプターは子について UniqueConstraint フォールトを戻します。
Ownership = True かつ KeepRelationship = False	• 親と子のビジネス・オブジェクトが両方とも存在しない場合、それらが作成されます。 • 親と子のビジネス・オブジェクトが両方とも存在する場合、それらが更新されます。 • 親ビジネス・オブジェクトが存在し、子ビジネス・オブジェクトが存在しない場合、親が更新され、子が作成されます。親が同じ子テーブル・レコードは削除されます。 • 親ビジネス・オブジェクトが存在せず、子ビジネス・オブジェクトが存在する場合、親は作成されず、アダプターは子について UniqueConstraint フォールトを戻します。
Ownership = False かつ KeepRelationship = True または Ownership = False かつ KeepRelationship = False	• 親ビジネス・オブジェクトと子ビジネス・オブジェクトの両方が存在しない場合、それらは作成されません。アダプターは子についてビジネス・オブジェクトが見つからないというエラーを戻します。 • 親と子のビジネス・オブジェクトが両方とも存在する場合、親のみが更新されます。 • 親ビジネス・オブジェクトが存在し、子ビジネス・オブジェクトが存在しない場合、親は更新されず、アダプターは子についてビジネス・オブジェクトが見つからないというエラーを戻します。 • 親ビジネス・オブジェクトが存在せず、子ビジネス・オブジェクトが存在する場合、親が作成されます。

注: トップレベルのビジネス・オブジェクトに UID として基本キーがない階層型ビジネス・オブジェクトでは、トップレベルのビジネス・オブジェクトは存在する場合には更新され、それ以外の場合には作成されます。1 次キーが何も設定されていない場合、アダプターは MissingData フォールトを戻します。

UID として基本キーがある階層型ビジネス・オブジェクトでは、基本キーが設定されていて、ビジネス・オブジェクトが存在する場合、ビジネス・オブジェクトは更新され、そうでない場合は作成されます。 1 次キーが設定されていない場合、ビジネス・オブジェクトは作成されます。

複数カーディナリティー関係

表 3. 複数カーディナリティー関係での Upsert 操作の動作

アプリケーション固有情報	Upsert 操作の動作
Ownership = True かつ KeepRelationship = True	• 親と子のビジネス・オブジェクトが両方とも存在しない場合、それらが作成されます。 • 親と子のビジネス・オブジェクトが両方とも存在する場合、それらが更新されます。親が同じ子テーブル・レコードへの変更はありません。 • 親ビジネス・オブジェクトが存在し、子ビジネス・オブジェクトが存在しない場合、親が更新され、子が作成されます。親が同じ子テーブル・レコードへの変更はありません。 • 親ビジネス・オブジェクトが存在せず、子ビジネス・オブジェクトが存在する場合、親が作成され、アダプターは UniqueConstraint フォールトを戻します。
Ownership = True かつ KeepRelationship = False	• 親と子のビジネス・オブジェクトが両方とも存在しない場合、それらが作成されます。 • 親と子のビジネス・オブジェクトが両方とも存在する場合、それらが更新されます。親が同じ子テーブル・レコードは削除されます。 • 親ビジネス・オブジェクトが存在し、子ビジネス・オブジェクトが存在しない場合、親が更新され、子が作成されます。親が同じ子テーブル・レコードは削除されます。 • 親ビジネス・オブジェクトが存在せず、子ビジネス・オブジェクトが存在する場合、親が作成され、アダプターは UniqueConstraint フォールトを戻します。
Ownership = False かつ KeepRelationship = True または Ownership = False かつ KeepRelationship = False	• 親ビジネス・オブジェクトと子ビジネス・オブジェクトが両方とも存在しない場合、親が作成され、アダプターは子についてビジネス・オブジェクトが見つからないというエラーを戻します。 • 親と子のビジネス・オブジェクトが両方とも存在する場合、親のみが更新されます。 • 親ビジネス・オブジェクトが存在し、子ビジネス・オブジェクトが存在しない場合、親が更新されます。アダプターは子についてビジネス・オブジェクトが見つからないというエラーを戻します。 • 親ビジネス・オブジェクトが存在せず、子ビジネス・オブジェクトが存在する場合、親が作成されます。

バッチ操作:

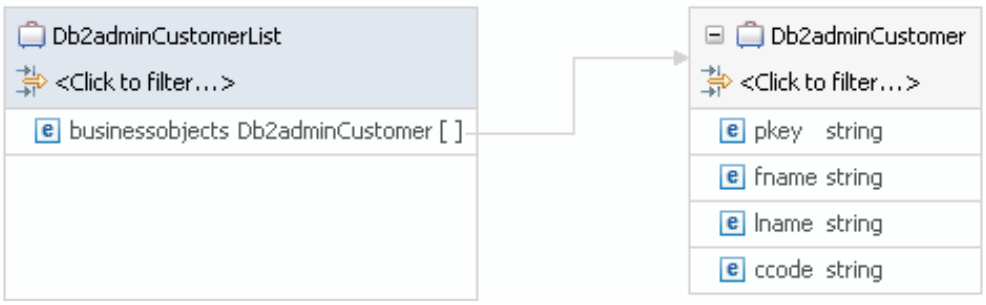
注: バッチ操作の振る舞いは、各データベース・ベンダーの JDBC ドライバーによって提供されるサポートに直接依存します。データベース・タイプに特有の動作についての注記に留意してください。

バッチ操作は、データベース内の大量レコードの挿入/更新、またはデータベースからの大量レコードの削除の効率を高めるためのソリューションとして提供されています。

バッチ操作は、ビジネス・オブジェクトのリストを受け入れ、トップレベルのビジネス・オブジェクトをバッチ・モードで処理します (子ビジネス・オブジェクトを持つ階層ビジネス・オブジェクトは現在はサポートされていません)。バッチ操作には、BatchCreate、BatchUpdate、および BatchDelete があります。詳しくは、 29 ページの『BatchCreate 操作』、 29 ページの『BatchUpdate 操作』、および 29 ページの『BatchDelete 操作』を参照してください。

入力ビジネス・オブジェクト

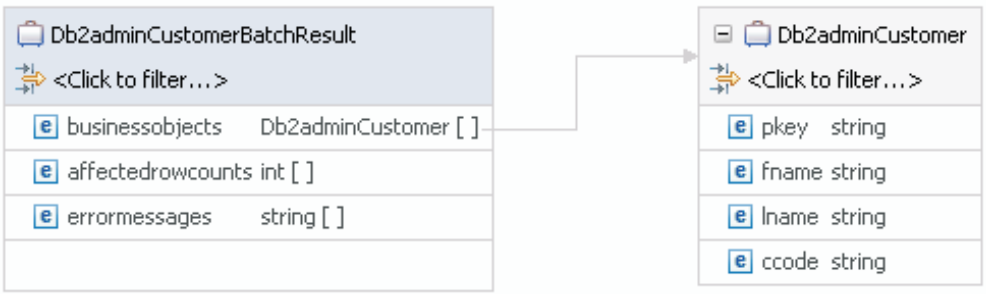
次の図は、バッチ操作の入力ビジネス・オブジェクトの例を示します。



入力ビジネス・オブジェクトには、ビジネス・オブジェクトのリスト (例えば、CustomerList) が含まれています。

戻されるビジネス・オブジェクト

次の図は、バッチ操作で戻されるビジネス・オブジェクトの例を示します (BatchResult ビジネス・オブジェクト)。



BatchResult ビジネス・オブジェクトには、ビジネス・オブジェクトのリスト (businessobjects)、更新カウント (affectedrowcounts)、およびエラー・メッセージ (errormessages) が含まれています。それぞれについての説明は以下のとおりです。

businessobjects

これには、戻される結果ビジネス・オブジェクトのリストが含まれています。ほとんどの場合、入力ビジネス・オブジェクトと同じです。しかし、入力ビジネス・オブジェクトのプロパティーに構成されている一連の UID または CopyAttributes ASI がある場合、更新されたビジネス・オブジェクトが戻されます。

デフォルトでは結果ビジネス・オブジェクトが戻されます。 345 ページの『returnBOsInBatch』 対話プロパティーの値を適切に設定することによって、結果ビジネス・オブジェクトを戻さないようアダプターを構成できます。

affectedrowcounts

データベース内の影響を受けたレコードの数を示します。これは、データベース・ドライバーによって戻されます。したがって、ここで表示される値は、各種ドライバーが使用する規則に基づきます。

表 4. affectedrowcounts の説明

値	説明
0	影響を受けたレコードがないことを示します (アダプターは、これを検出レコードなし例外と見なします)。
1	1 つのみのレコードが影響を受けたことを示します (予期される結果)。
-2	バッチ・ステートメントは正常に実行されたが、影響された行数カウントを入手できないことを示します。
-3	バッチ操作の実行が失敗したことを示します。詳しいエラー・メッセージは、対応する errormessages フィールドにあります。戻されるエラー・メッセージは、データベース・ドライバーに依存します。例えば、MS SQL Server データベースは、失敗したレコードに関する最初の例外メッセージのみを戻します。
-99998	予期せずにビジネス・オブジェクトがスキップされたことを示します。これはアダプター固有です。おそらくエラー・メッセージは入手できません。さらに支援が必要な場合は、Websphere Adapters サポートに連絡してください。
-99999	アダプター固有の、無効な入力を示します。内部例外または不明な例外がスローされます。エラー・メッセージの詳細は、対応する errormessages フィールドにあります。

errormessages

対応する入力ビジネス・オブジェクトの詳細エラー・メッセージを示します。

注:

- これらのバッチ操作では階層ビジネス・オブジェクトはサポートされていません。
- アダプターは、パフォーマンスを最大化するため、入力ビジネス・オブジェクトの実行順序を保証しません。
- アダプターは、バッチ操作用のフォールト・バインディングを自動的に生成することはありません。
- バッチ操作 (BatchCreate、BatchUpdate、および BatchDelete) とバッチ・ビジネス・オブジェクトとの間には何も関係はありません。バッチ操作は大量のビジネス・データを処理するときに使用でき、バッチ・ビジネス・オブジェクトは複数のユーザー定義 SQL ステートメントを処理するときに使用できます。
- バッチ操作では AutoCommit はサポートされていません。したがって、定義されているバッチ操作がある場合、アウトバウンド・インターフェースで 327 ページの『自動コミット (Auto commit) (AutoCommit)』 プロパティーは false に設定されている必要があります。AutoCommit = True の場合、例外がスローされます。

例外処理

バッチ操作中にエラーが発生する場合、 345 ページの『skipErrorsInBatch』 (デフォルト値は false) オプションを使用して、そのようなエラーの処理方法を構成できます。

BatchCreate 操作:

BatchCreate 操作は、入力としてビジネス・オブジェクトのリストを受け入れ、トップレベルのビジネス・オブジェクトをバッチ・モードで処理します。これによって、対応するトップレベルのレコードをデータベース内に効率的に作成することができます。

BatchCreate 操作を処理するため、アダプターは次のアクションを実行します。

- トップレベルの入力ビジネス・オブジェクトのみを考慮に入れ、すべての子入力ビジネス・オブジェクトを無視します。
- 対応するトップレベルの入力ビジネス・オブジェクトのレコードをデータベースに挿入します。1 回のバッチ対話で実行されるビジネス・オブジェクトの最大数は、 344 ページの『batchSize』 プロパティーに依存します。
- この操作の結果として BatchResult ビジネス・オブジェクトを戻します。
- ビジネス・オブジェクトの処理中にエラーが発生した場合、例外を生成します。例外について詳しくは、『例外処理』を参照してください。

UID の取り扱い

- JDBC API 制約のため、AUTO UID 値の取得はできません。
- BatchCreate 操作では SEQUENCE UID がサポートされています。

BatchUpdate 操作:

BatchUpdate 操作は、入力としてビジネス・オブジェクトのリストを受け入れ、トップレベルのビジネス・オブジェクトをバッチ・モードで処理します。これによって、トップレベルの複数ビジネス・オブジェクトを効率的に更新することができます。

BatchUpdate 操作を処理するため、アダプターは次のアクションを実行します。

- トップレベルの入力ビジネス・オブジェクトのみを考慮に入れ、すべての子入力ビジネス・オブジェクトを無視します。

  注: Update 操作とは異なり、 BatchUpdate 操作は Execute 操作を実行する前に Retrieve 操作を実行しません。
- トップレベルの入力ビジネス・オブジェクトのレコードをデータベース内で更新します。1 回のバッチ対話でのビジネス・オブジェクトの最大数は、 344 ページの『batchSize』 プロパティーに依存します。
- この操作の結果として BatchResult ビジネス・オブジェクトを戻します。
- ビジネス・オブジェクトの処理中にエラーが発生した場合、例外を生成します。例外について詳しくは、『例外処理』を参照してください。

BatchDelete 操作:

BatchDelete 操作は、入力としてビジネス・オブジェクトのリストを受け入れ、トップレベルのビジネス・オブジェクトをバッチ・モードで処理します。これによって、対応するトップレベルのレコードをデータベース内で効率的に削除することができます。

BatchDelete 操作を処理するため、アダプターは次のアクションを実行します。

- トップレベルの入力ビジネス・オブジェクトのみを考慮に入れ、すべての子入力ビジネス・オブジェクトを無視します。

  注: Delete 操作とは異なり、 BatchDelete 操作は Execute 操作を実行する前に Retrieve 操作を実行しません。
- トップレベルの入力ビジネス・オブジェクトのレコードをデータベースから削除します。1 回のバッチ対話でのビジネス・オブジェクトの最大数は、 344 ページの『batchSize』 プロパティーに依存します。
- この操作の結果として BatchResult ビジネス・オブジェクトを戻します。
- ビジネス・オブジェクトの処理中にエラーが発生した場合、例外を生成します。例外について詳しくは、 29 ページの『例外処理』を参照してください。

操作用のユーザー定義の照会基準:

IBM WebSphere Adapter for JDBC は、 RetrieveAll、UpdateAll、DeleteAll、および Exists の各操作で、テーブル・ビジネス・オブジェクトに対するユーザー定義の基準をサポートしています。このユーザー定義の基準を使用して、1 つの操作で複数のレコードの取得、更新、削除を実行でき、 1 つの操作で複数または 1 個のレコードの存在をチェックできます。

外部サービス・ウィザードを実行した後、ビジネス・オブジェクト・エディターで RetrieveAllCriteria、UpdateAllCriteria、DeleteAllCriteria、および ExistsCriteria の各 ASI を使用して、テーブル・ビジネス・オブジェクトに対する照会基準を設定することができます。

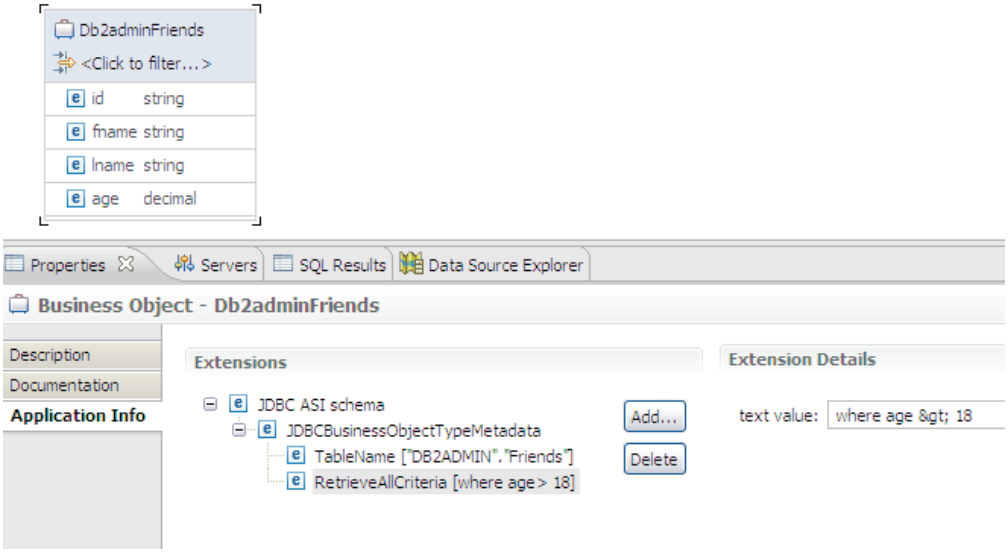


図 7. RetrieveAllCriteria アプリケーション固有情報を使用してテーブル・ビジネス・オブジェクトに対するユーザー定義の基準を設定する

操作用に生成される完全な SQL ステートメントは、基本 SQL ステートメントと基準 SQL ステートメントからなります。

ユーザーが条件を設定すると、実行時に、アダプターは完全な SQL ステートメントの一部として、操作に基づいて基本 SQL ステートメントを生成し、次に、生成した基本 SQL ステートメントに基準 SQL ステートメントを付加し、最後に操作用の完全な SQL ステートメントを生成します。

例えば、RetrieveAllCriteria ASI を `WHERE AGE > 18` と設定した場合、RetrieveAll 操作に使用される完全な SQL ステートメントは `SELECT ID, FNAME, LNAME, AGE FROM FRIENDS WHERE AGE > 18` のようになります。ここで、`SELECT ID, FNAME, LNAME, AGE FROM FRIENDS` が RetrieveAll 操作に基づいた基本 SQL ステートメントであり、 `WHERE AGE > 18` が基準 SQL ステートメントです。RetrieveAll 操作用の完全な SQL ステートメントは、これら 2 つのステートメントから構成されます。

注: 階層型ビジネス・オブジェクトの場合、トップレベルのビジネス・オブジェクトと子ビジネス・オブジェクトの両方に対して基準を設定すると、最終的な SQL ステートメントの生成には、トップレベルのビジネス・オブジェクトに指定された基準のみが使用されます。子ビジネス・オブジェクトに指定された基準は、トップレベルのビジネス・オブジェクトでの操作の実行には使用されません。それが使用されるのは、子ビジネス・オブジェクトで操作が実行されるときだけです。

各操作用に生成される基本 SQL ステートメント

ユーザーが条件を設定すると、アダプターは、各操作のために実行時に基本 SQL ステートメントとして以下の SQL ステートメントを生成します。この基本 SQL ステートメントは、操作用の完全な SQL ステートメントの一部です。

表 5. 各操作の基本 SQL ステートメント

操作	基本 SQL ステートメント
RetrieveAll	`SELECT <COLUMN_1>, <COLUMN_2> ... FROM <TOP_LEVEL_TABLE_NAME>`
UpdateAll	`UPDATE <TOP_LEVEL_TABLE_NAME> SET <COLUMN_1>=?, <COLUMN_2>=?`
DeleteAll	`DELETE FROM <TOP_LEVEL_TABLE_NAME>`
Exists	`SELECT <COLUMN_1>, <COLUMN_2> ... FROM <TOP_LEVEL_TABLE_NAME>`

各操作用に生成される基準 SQL ステートメント

基準 SQL ステートメントは、生成される完全な SQL ステートメントの一部です。基準 SQL ステートメントは以下を含むことができます。

- 有効な SQL 演算子

  `>`、`<`、`<=`、`>=`、`!=`、`EXISTS`、 `IN`、`LIKE`、および他の有効な SQL 演算子
- 任意の有効な SQL 演算子を使用する単純な `WHERE` 節

  ```
  WHERE (PKEY='XYZ' or FNAME='ABC') and (AGE<10 or AGE>70)
  ```
- SQL 照会がネストされている複雑な `WHERE` 節

  ```
  WHERE PKEY NOT IN(SELECT PKEY FROM FRIENDS WHERE AGE !=18)
  ```
- 他の任意の有効な SQL ステートメント部分

  ```
  WHERE PKEY NOT IN (SELECT PKEY FROM FRIENDS WHERE AGE != 18) ORDER BY
  AGE, FNAME
  ```

基準の内部で特殊な ID (例えば、特殊文字を含むデータベース・キーワードや ID) を使用する場合は、データベース固有のラッパー・ストリングを使用してその ID をラップする必要があります。デフォルトでは、ラッパー・ストリングは二重引用符です。例えば、 `WHERE "COLUMN SPEC 1" = 'ABC'` のようにします。

注: ASI に特殊 XML 文字が含まれていて、スキーマ・ファイルに直接保存される場合、ビジネス・オブジェクト定義に関するエラーが起こります。こういったエラーを回避するには、それらの XML 文字を手動でエンコードする必要があります。例えば、「<」は「<」にエンコードする必要があります。

ユーザー定義の照会基準内の名前付きパラメーター

アダプターは、ユーザー定義の照会基準 (基準 SQL ステートメント) 内の名前付きパラメーターをサポートします。名前付きパラメーターは、コロン (:) の後に ID を続けることで示されます。名前付きパラメーターを使用するときには、操作を実行するときに対応する値を入力ビジネス・オブジェクトに設定する必要があり、そうしない場合は未設定フィールドはヌルとして処理されます。

例えば、基準 SQL ステートメントを `WHERE AGE>:age` のように定義できます。

この例では、 WHERE 節に名前付きパラメーター age が含まれています。この名前付きパラメーターに対応する値は、実行時に入力ビジネス・オブジェクトに渡されます。

注: 名前付きパラメーターは、大/小文字の区別があります。

基準内で参照階層を指示するには、コロン (:) を使用し、その後にフィールド名を続けます。

例: `WHERE FNAME <> :fname AND EXISTS (SELECT * FROM SALARYINFO WHERE SALARYINFO.EMPNUM=EMPLOYEE.EMPNUM AND BASIC > :salaryinfoobj:basic)`

複数の入力インスタンスをパラメーター中で参照するには [n] を使用します (最初のインスタンスの索引は 0 です)。

注: インスタンスの開始索引は、1 ではなく 0 です。

例: `WHERE EXISTS (SELECT * FROM SALARYINFO WHERE SALARYINFO.EMPNUM=EMPLOYEE.EMPNUM AND BASIC < :salaryinfoobj[1]:basic)` は、2 番目の salaryinfoobj インスタンスの basic 属性を示します。

名前付きパラメーターが参照することができるのは、ビジネス・オブジェクト内のフィールドのみです。通常の子テーブル・ビジネス・オブジェクトを参照することは無効です。ただし、Struct タイプの子ビジネス・オブジェクトを参照することは有効です。

例 1: `WHERE STRUCTTYPEOBJ=:structtypeobj` は有効な参照です。

例 2: `WHERE CCODE=:custinfoobj:ccode` は有効な参照です。

例 3: `WHERE CUSTINFOOBJ=:custinfoobj` は無効な参照です。

ヌル値のフィールドを探すためにレコードを照会する場合、 SQL ステートメント内で IS NULL キーワードを使用する必要があります。

例: `WHERE <Column Name> is null`

条件の比較式の中に名前付きパラメーターを定義する場合、実行時に、対応するフィールドにヌル以外の値を入力する必要があります。そうしないと比較式は常に FALSE を戻します。

例: `WHERE CCODE=:custinfoobj:ccode` CUSTINFOOBJ の CCODE フィールドがヌルに設定されている場合、この条件に一致するレコードはありません。

子ビジネス・オブジェクト属性を使用したレコードのマッチング:

IBM WebSphere Adapter for JDBC は、 RetrieveAll、DeleteAll、Exists、および UpdateAll 操作で、子ビジネス・オブジェクト属性を使用したレコードのマッチングをサポートしています。

入力ビジネス・オブジェクトの子ビジネス・オブジェクト内で属性を指定すると、アダプターは子ビジネス・オブジェクトでの結合検索により、トップレベルのビジネス・オブジェクトのマッチングを行います。

- **単一インスタンス内に設定された子ビジネス・オブジェクト属性**

  単一の子ビジネス・オブジェクト・インスタンス内の子ビジネス・オブジェクト属性を指定すると、アダプターは、等号演算子 (=) を使用してそれらの子ビジネス・オブジェクト属性のマッチングを行い、マッチング・ルール用に AND 論理を生成します。図 1 の例では、アダプターは、company が「XYZ」と等しく、かつ customertype が「VIP」と等しいすべての顧客レコードを取り出すための照会を実行します。

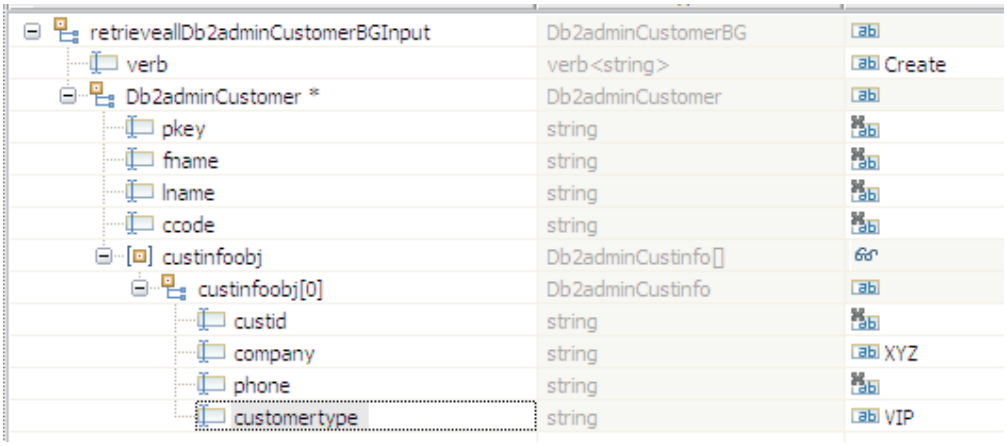


図 8. 単一の子ビジネス・オブジェクト・インスタンス内に設定された属性を使用してレコードを取り出す

- **同じ子ビジネス・オブジェクトの異なるインスタンスに設定された子ビジネス・オブジェクト属性**

  同じ子ビジネス・オブジェクトの異なるインスタンス内の子ビジネス・オブジェクト属性を指定すると、アダプターは、同じインスタンス内に設定された属性に対しては AND 論理を生成し、異なるインスタンスの場合は OR 論理を生成します。図 2 の例では、アダプターは、company が「XYZ」と等しくて customertype が「VIP」と等しいか、または company が「ABC」と等しくて customertype が「VVIP」と等しいすべての顧客レコードを取り出すための照会を実行します。

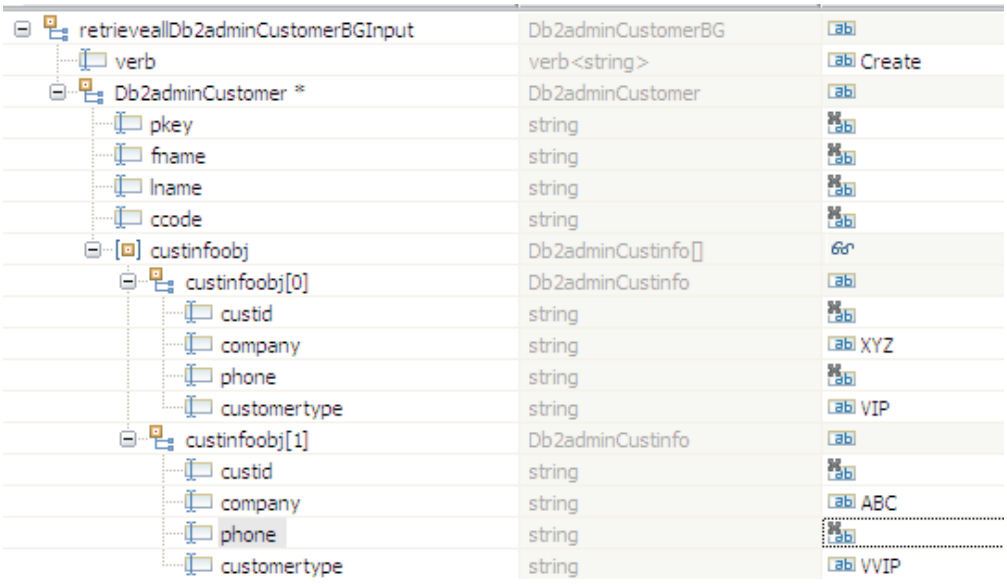


図 9. 異なる子インスタンス内に設定された属性を使用してレコードを取り出す

Inbound 処理

アダプターは、イベント送達を行う Inbound イベント管理をサポートします。イベントは、データベース・アプリケーションか、またはユーザーが指定したカスタム照会の結果のいずれかによってデータが取り込まれたイベント・ストアで処理されます。アダプターによるイベントのポーリング頻度と、1 回あたりのエクスポートへの送信レコード数を制御できます。

アダプターは、次のいずれかの方法で変更確認のためのポーリングを実行します。

- 標準イベント処理。アダプターはイベント・ストアを調べ、データベース・アプリケーションにより格納されたイベントがあるかどうかを確認します。
- カスタム・イベント処理。アダプターはユーザー定義の照会、ストアード・プロシージャー、またはストアード関数を実行します。

最初に外部サービス・ウィザードを使用してアダプターを構成するときに標準またはカスタムのイベント処理をカスタマイズできます。または、後でサーバーの管理コンソールを使用して活動化仕様プロパティーを変更することによってもイベント処理をカスタマイズすることができます。

イベントの対象であるデータベース・オブジェクトは、通知がエクスポートに送信された後に取得されます。この結果として、発生した取得エラーの検出と通知は、エクスポートの通知の完了後まで据え置かれます。これは、アダプターがエクスポートに通知する前に取得エラーを検出可能な、バージョン 6.2.x のアダプターでのイベント処理とは異なります。

標準イベント処理:

標準イベント処理では、アダプターはイベントをポーリングする SQL 照会を指定して、イベントが一度で確実に送達されるようにします。

Oracle Change Data Capture などのデータベース・トリガーまたはツールは、データベース内のテーブルのレコードが作成、更新、または削除されたときに実行します。トリガーまたはその他のツールはイベント・レコードをイベント・ストアに書き込みます。イベント・ストアは、ポーリング・アダプターがイベント・レコードを処理できるまでイベント・レコードが保存される永続キャッシュです。イベント・ストアはユーザー・テーブルと同じデータベース内にテーブルとして実装されます。これはアダプターによってアクセスされるデータベース・オブジェクトを含むテーブルです。

トリガーを定義するか、または他のツールをセットアップして、イベントの受け取りが必要な対象のデータベース表への変更をレポートする必要があります。アダプターには、アダプターのトリガーのセットアップ方法を示すサンプル・データベース・スクリプトがあります。サンプルは、*IID_installation_dir*/ResourceAdapters/JDBC_*version*/scripts ディレクトリーにあります。ここで、*version* はアダプターのバージョン (例えば、7.0.0.0) を示します。IBM DB2、IBM DB2 for z/OS®、Oracle、および Microsoft SQL Server に対応したサンプル・スクリプトが用意されています。

アダプターは「送達は 1 回のみ」を提供しています。これは、各イベントはエクスポートに 1 回だけ送達されることを保証するものです。モジュールについて「送達は 1 回のみ」を有効にした場合、イベント・ストア内の各イベントにトランザクション ID (XID) が設定されます。イベントが処理対象として取得されると、イベント・ストア内でそのイベントの XID 値が更新されます。次に、イベントが、対応するエクスポートに送達されて、イベント・ストアから削除されます。イベントが送達される前に、データベース接続が失われたか、アプリケーションが停止した場合、イベントは完全に処理されない可能性があります。この場合、イベントの再処理とエクスポートへの再送信が必要であることが XID 列によって示されます。データベース接続が再確立されるか、またはアダプターが再始動されると、アダプターは、イベント・ストアで XID 列に値を持つイベントがあるかどうかをチェックします。アダプターは、まずこれらのイベントを処理してから、ポーリング周期の間にその他のイベントをポーリングします。

アダプターは、すべてのイベントを処理するか、またはビジネス・オブジェクト・タイプによってイベントをフィルター操作できます。フィルターは、活動化仕様プロパティー EventFilterType を使用して設定します。このプロパティーは、ビジネス・オブジェクト・タイプをコンマで区切ったリストを持ちます。プロパティーで指定されたタイプのみが処理されます。プロパティーに値が指定されていない場合、フィルターは適用されず、すべてのイベントが処理されます。活動化仕様プロパティー FilterFutureEvents が true に設定されている場合、アダプターは、タイム・スタンプに基づいてイベントをフィルターに掛けます。アダプターは、各ポーリング周期のシステム時刻を各イベントのタイム・スタンプと比較します。イベントが将来発生するように設定されている場合は、その時刻になるまで処理されません。

カスタム・イベント処理:

カスタム・イベント処理では、イベントをポーリングする SQL 照会またはストアード・プロシージャーを提供します。

カスタム・イベント処理を使用して、どのイベントをエクスポートに送達するかを制御します。標準のイベント処理でイベント・ストアのポーリングに使用する SQL 照会の代わりに、アダプターで実行するデータベース照会 (*カスタム・イベント照会*) を指定します。カスタム・イベント照会では、必要なフィルタリングを実行する必要があります。カスタム・イベント処理が必要な場合、ウィザードでオプションを選択するか、管理コンソールの EventQueryType 活動化仕様プロパティーを設定して指定します。

XID 値の格納用に標準イベント・ストアを作成する場合、カスタム・イベント処理は「送達は 1 回のみ」をサポートします。 アダプターは、カスタム・イベント照会によって返されたイベントをイベント・ストアに格納し、そのイベントを XID 値で更新します。アダプターは、標準のイベント処理と同じ方法でイベントを処理します。標準イベント・ストアを照会するカスタム照会は作成しないでください。アダプターが「送達は 1 回のみ」に構成されている場合、そのテーブルにイベントが格納されるのは一時的であるためです。さらにこの状況では、アダプターが、カスタム照会から取得したイベント ID 値をイベント・ストアに取り込むため、イベント・ストアではイベント ID 値の自動生成が行われてはいけないことになります。

注: カスタム・イベント処理を使用するときには、 374 ページの『イベントを一度のみ送達する (AssuredOnceDelivery)』 のプロパティーを `True` に設定してください。

カスタム・イベント処理を有効にするには、アダプターを使用するようにモジュールを構成するときウィザードで拡張オプションを選択するか、EventQueryType 活動化仕様プロパティーを設定します。

注: カスタムのイベント処理を使用する場合は、 373 ページの『将来のタイム・スタンプを持つイベントを処理しない (FilterFutureEvents)』、 376 ページの『処理するイベント・タイプ (EventTypeFilter)』、および 364 ページの『イベント・フィルター用のアダプター・インスタンス (AdapterInstanceEventFilter)』 プロパティーはサポートされません。

イベント・テーブルからの処理済みイベントのアーカイブについて詳しくは、「WebSphere Adapter for JDBC の Inbound サービスのためのイベントのアーカイブ機能の実装」を参照してください。

カスタム・イベント照会

カスタム・イベント照会の実行の指定は、ウィザードの拡張オプションでユーザー定義のイベント照会を提供するか、CustomEventQuery 活動化仕様プロパティーを設定して行います。次のプログラム・タイプのいずれかを指定してください。

- 標準の SQL ステートメント
- ストアード・プロシージャー
- ストアード関数

これらのプログラムはすべて、入力パラメーターとしてポーリング数量 (アダプターが実行時に提供する活動化仕様プロパティー) を取ります。プログラムは他の入力パラメーターも受け取ることができます。これらのプログラムは、結果セットを戻す必要があります。この結果セットは、ポーリング数量に相当する数のレコード

を含み、event_id、object_key、object_name、および object_function の列をこの順序で含んでいます。アダプターは結果セットからイベント・オブジェクトを生成し、イベントを処理します。

標準の SQL ステートメント

処理するイベントを選択する SQL SELECT ステートメントを指定できます。照会には、ポーリング数量の入力パラメーターとその他の入力パラメーターを指定できます。

ストアード・プロシージャー

カスタム照会は、ポーリング数量を入力として受け取り、結果セットのタイプの出力パラメーターを戻すストアード・プロシージャーである場合があります。ストアード・プロシージャーを指定するときには、次の構文を使用してください。

```
call procedure_name (?, ?)
```

ここで *procedure_name* は、実行する必要のあるストアード・プロシージャーの名前です。最初のパラメーターはポーリング数量を表し、2 番目のパラメーターは結果セットを表します。

ストアード・プロシージャーは、その他の入力パラメーターを受け入れることもでき、以下のように call ステートメント自体でそれらを提供します。

```
call procedure_name (25, ?, ?)
```

ストアード関数

カスタム照会は、ポーリング数量を入力として受け取り、結果セットを戻すストアード関数である場合もあります。ストアード関数を指定するときには、次の構文を使用してください。

```
? = call function_name (?)
```

ここで、*function_name* は、ストアード関数の名前です。最初のパラメーターは結果セットを表し、2 番目のパラメーターはポーリング数量を表します。

ストアード関数は、その他の入力パラメーターを受け入れることもでき、以下のように call ステートメント自体でそれらを提供します。

```
? = call function_name (?, 'abc')
```

カスタム更新照会およびカスタム削除照会

カスタム・イベント処理では、カスタム更新/削除照会を提供することができます。これらの照会は、各イベントの処理後に実行されます。ユーザーは通常、*更新照会* を使用して、データベース・レコードが以降のポーリング周期中に処理対象として取り出されないようにします。*削除照会* は、各イベントの処理後にデータベース・レコードを削除する必要がある場合に使用します。更新照会と削除照会は、どちらもオプションです。

更新照会と削除照会は、CustomUpdateQuery と CustomDeleteQuery の各活動化仕様プロパティーで指定されます。これらの照会は、標準 SQL ステートメント、ストアード・プロシージャー、またはストアード関数として入力できます。カスタム更

新/削除照会の構文は、カスタム照会の構文と同じです。更新/削除照会は、イベント ID の入力パラメーターを取ります。アダプターは、実行時にイベント ID の値を提供します。照会には、これ以外の入力パラメーターを指定できます。このような入力パラメーターは、カスタム・イベント照会で説明した方法と同様に、照会構文自体に指定します。

イベント・ストア:

イベント・ストアは、ポーリング・アダプターがイベント・レコードを処理できるまでイベント・レコードが保存される永続キャッシュです。Inbound 要求がシステム内を通るときに、アダプターはイベント・ストアを使用してその Inbound 要求を追跡します。データベース・レコードの作成、更新、または削除が行われるたびに、アダプターはイベント・ストアのイベントの状況を更新します。アダプターによるイベント状況の更新は、イベントがサーバー上の構成済みエクスポートに渡されるまで、リカバリーの目的で継続的に行われます。

アダプターは、イベント・ストアから定期的な間隔でイベント・レコードをポーリングします。ポーリング呼び出しごとに、多数のイベントがアダプターにより処理されます。イベントは、優先度の昇順とイベントのタイム・スタンプの昇順に処理されます。各ポーリング周期において、アダプターはすべての新規イベントをピックアップします。新規イベントごとに、アダプターはイベントの object_key 列に設定されている値を取得し、オブジェクト名フィールドに指定されている値に対応するビジネス・オブジェクトをロードします。オブジェクトのロード後、アダプターは object_key 列に指定されている値に基づいて、ビジネス・オブジェクトの基本キー値を設定します。キーの設定後、アダプターはそれらのキーに基づいてオブジェクトの検索を実行します。検索した情報からビジネス・オブジェクトまたはオプションでビジネス・グラフが作成され、エクスポートにパブリッシュされます。

ストアード・プロシージャーをビジネス・オブジェクトの Retrieve 操作に関連付けている場合は、ストアード・プロシージャーの入力パラメーターとビジネス・オブジェクト属性 (通常は基本キー) の間のマッピングを定義できます。そうしたマッピングを定義すると、アダプターは、ストアード・プロシージャーの入力パラメーターを設定し、ストアード・プロシージャーを呼び出し、ストアード・プロシージャーから得られた結果に基づいてオブジェクトにデータを取り込みます。

ストアード・プロシージャーとストアード関数の場合、RetrieveSP アプリケーション固有情報を使用して、ストアード・プロシージャー/ストアード関数の入力パラメーターと、ビジネス・オブジェクト属性 (一般に基本キーを使用) の間にマッピングを定義していると、アダプターはストアード・プロシージャーの入力パラメーターを設定し、ストアード・プロシージャーを呼び出し、ストアード・プロシージャーの結果に基づいてビジネス・オブジェクトにデータを取り込みます。

object_function 列の値が `Delete` で、オブジェクトが削除されたことを示す場合、そのオブジェクトはデータベースから取得されません。データ・オブジェクトにキーが設定されると、ビジネス・オブジェクトとオプションでビジネス・グラフが作成されてエクスポートに送達されます。

イベントの通知が正常に完了すると、イベント・ストアからその項目が削除されます。失敗したイベントの場合は、エントリーがイベント・ストアに残り、

event_status 列が -1 に設定されます。高可用性 Active-Active サポートが有効になっている場合、イベントの処理が開始したときに event_status 列は 3 に設定されます。

表 6 に、イベント・ストアの表フォーマットと内容の説明を示します。

表 6. イベント・ストア・データベース表の定義

列名	タイプ	説明
XID	String	送達が 1 回のみの場合の固有トランザクション ID (XID) 値
event_id	Number	テーブルの基本キーである固有イベント ID。この ID には、object_key と同じ値を設定できます。
object_key	String	イベント・ストアの取得レコードのキーが含まれているストリング。 この列を NULL にすることはできません。 値を、1 つ以上の *key=value* ペアとして指定します。各ペアはセミコロン (;) で区切ります。 あるいは、基本キーの値のみをセミコロン (;) で区切って指定することもできます。この場合、ビジネス・オブジェクトでの基本キーの定義順序と同じ順序で値を指定する必要があります。
object_name	String	ビジネス・オブジェクトまたはビジネス・グラフの名前。ビジネス・オブジェクト (またはビジネス・グラフ内のビジネス・オブジェクト) は、階層ビジネス・オブジェクトであっても構いません。各ビジネス・オブジェクトまたはビジネス・グラフは、テーブルまたはビューを参照します。 この列を NULL にすることはできません。
object_function	String	イベントに対応した操作 (Delete、Create、Update など)。 この列を NULL にすることはできません。
event_priority	Number	イベント優先順位を識別します。この値は正整数でなければなりません。 この列を NULL にすることはできません。
event_time	Timestamp	イベントが生成された日時。形式は mm/dd/yyyy hh:mm:ss です。

表 6. イベント・ストア・データベース表の定義 (続き)

列名	タイプ	説明
event_status	Number	イベント状況。これは、最初は新規イベントの値に設定され、イベントが処理されるとともにアダプターによって更新されます。状況の値は次のいずれかです。 • -1: イベント処理中にエラーが発生しました。 • 0: 新規イベントを示します。 • 1: エクスポートに送達されたイベントを示します。 • 3: イベントがアダプターによって処理中かどうかを示します。このイベント状況は、高可用性 Active-Active モードの場合にのみ適用されます。詳しくは、84 ページの『クラスター環境での WebSphere Adapters』を参照してください。 この列を NULL にすることはできません。
event_comment	String	イベントに関連付けられているコメント
connector_ID	String	特定のイベントを受信するアダプター・インスタンスの固有 ID。
event_timeout	Timestamp	イベントがタイムアウトになったかどうかを示します。

ビジネス・オブジェクト

ビジネス・オブジェクトとは、データ、データ上で実行されるアクション、およびデータを処理するための追加の指示 (存在する場合) で構成される構造体のことです。IBM WebSphere Adapter for JDBC は、ビジネス・オブジェクトを使用して、データベースのテーブルとビュー、データベース照会、ストアード・プロシージャー、およびストアード関数の結果を表現します。ビジネス・オブジェクトにより、データベースのオブジェクトの階層を作成し、無関係なテーブルをグループ化できます。コンポーネントはビジネス・オブジェクトを使用してアダプターと通信します。

アダプターによるビジネス・オブジェクトの使用法

統合アプリケーションは、ビジネス・オブジェクトを使用してデータベースにアクセスします。アダプターは、Outbound 要求のビジネス・オブジェクトを、データベースへアクセスするための JDBC API 呼び出しに変換します。Inbound イベントの場合、アダプターはイベントのデータをビジネス・オブジェクトに変換し、このビジネス・オブジェクトがアプリケーションに戻されます。

アダプターは、ビジネス・オブジェクトを使用してデータベース内の次のタイプのオブジェクトを表現します。

- テーブルとビュー
- シノニムとニックネーム

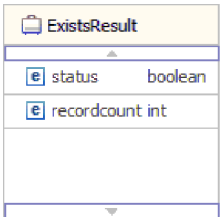
- ストアード・プロシージャーとストアード関数

一部のビジネス・オブジェクトは、データベース・オブジェクトを表現しません。これらのビジネス・オブジェクトには、以下のものがあります。

- バッチ SQL ビジネス・オブジェクト。一連のユーザー定義の insert ステートメント、update ステートメント、および delete ステートメントを表します。
- 照会ビジネス・オブジェクト。データベースに対して実行されるユーザー定義 SQL 照会を表します。
- ラッパー・ビジネス・オブジェクト。このオブジェクトにより、無関係なテーブル・オブジェクトとビュー・オブジェクトを単一のビジネス・オブジェクトにグループ化でき、複数のストアード・プロシージャーを単一のビジネス・オブジェクトにグループ化することができます。

アダプターは、出力に一部のビジネス・オブジェクトを使用します。これらのビジネス・オブジェクトには、以下のものがあります。

- コンテナー・ビジネス・オブジェクト。RetrieveAll 操作からの出力が入ります。
- ExistsResult ビジネス・オブジェクト。これには、Exists 操作からの出力が入ります。



ビジネス・オブジェクト内でのデータの表現方法

テーブルまたはビュー・ビジネス・オブジェクトの場合

テーブルまたはビューの各列は、テーブル・ビジネス・オブジェクトまたはビュー・ビジネス・オブジェクトの単純属性により表現されます。*単純属性* とは、String、Integer、または Date などの単一値を表す属性です。その他の属性は、子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表します。

同じビジネス・オブジェクトに含まれる単純属性を、別々のデータベース表に格納することはできませんが、次の状況が考えられます。

- データベース表に、対応するビジネス・オブジェクトに含まれる単純属性の数よりも多くの列が含まれる場合があります。すなわち、データベースの列の一部が、ビジネス・オブジェクト内に表されていません。アプリケーションによるビジネス・オブジェクトの処理で必要な列のみを実際の設計に含める必要があります。
- ビジネス・オブジェクトに、対応するデータベース表に含まれる列の数よりも多くの単純属性が含まれる場合があります。すなわち、ビジネス・オブジェクト内の属性の一部が、データベース内に表されていません。データベース内に列を持たない属性は、アプリケーション固有情報を持っていないか、デフォルト値が設定されているか、またはストアード・プロシージャーかストアード関数のパラメーターです。

- ビジネス・オブジェクトは、複数のデータベース表にまたがるビューを表すことができます。アダプターでは、Create、Update、および Delete 操作など、データベースに対する変更によって起動されるイベントを処理するときに、このようなビジネス・オブジェクトを使用できます。ただし、ビジネス・オブジェクトの要求を処理する場合には、Retrieve および RetrieveAll 要求に対してのみ、このようなビジネス・オブジェクトを使用できます。

テーブル・ビジネス・オブジェクトには、対応するデータベース表に基本キーがない場合でも、常に基本キーが設定されています。アダプターは、テーブル・ビジネス・オブジェクトを取得するときに、基本キー属性で指定される列を使用します。データベースで外部キー参照が定義されている場合、アダプターはテーブル間の親子関係を自動的にディスカバーして表示します。例えば、親テーブルである CUSTOMER テーブルと子テーブルである ADDRESS テーブルがあるとします。データベース内で ADDRESS から CUSTOMER への外部キー参照を定義した場合、アダプターは自動的に親子関係をディスカバーし、外部キー参照を「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウに表示します。外部キー参照が CUSTOMER から ADDRESS への場合、アダプターは自動的に「**単一カーディナリティー**」チェック・ボックスを選択し、外部キー参照を表示します。1 つのテーブルに複数の外部キー参照が定義されている場合、アダプターは外部キー関係を 1 つだけ生成します。

アダプターでは、複合の (すなわち複数の) 基本キーが設定されている表をサポートしています。データベースに基本キーが 1 つ以上存在する場合、ウィザードは、テーブル・ビジネス・オブジェクトのそれらの列に基本キー・プロパティーを設定します。データベース表に基本キーが存在しない場合、外部サービス・ウィザードでは、そのビジネス・オブジェクトをディスカバーして構成するときに基本キー情報の入力を求めるプロンプトが出されます。例えば CUSTOMER (pkey1, pkey2) > ADDRESS (fkey1, fkey2) といった複合基本キーがある場合、アダプターは ADDRESS (fkey1) と CUSTOMER (pkey1)、ADDRESS (fkey2) と CUSTOMER (pkey2) をそれぞれ関連付けます。シーケンス列や ID 列など、固有のデータを持つ列を指定してください。ID 列 (Informix では*シリアル列* と呼ばれる) は、データベースでテーブルの各行に自動的に固有の数値を生成する方法になります。テーブルには、ID 属性で定義される単一の列を作成できます。 ID 列の例として、オーダー番号、従業員番号、ストック番号、および問題番号などがあります。 ID 列は、DB2、Informix および Microsoft SQL Server のテーブルにのみ定義できます。

注: DB2 または Microsoft SQL Server データベースのいずれかの表に対してディスカバリー・プロセスを実行する場合に、その表で 1 つの列を ID 列として定義している場合、その表に対して生成されたビジネス・オブジェクトには、ID 列の固有 ID 属性は含まれていません。この場合、アプリケーション固有情報に属性を手動で追加することにより、生成されたビジネス・オブジェクトを編集する必要があります。これは、IBM Integration Designerのアセンブリー・エディターによって行うことができます。Informix データベースのテーブルに対してディスカバリー・プロセスを実行した場合は、固有 ID の属性を手動で追加する必要はありません。Informix の場合、生成されたビジネス・オブジェクトには、シリアル列の固有 ID 属性が含まれています。

ビジネス・オブジェクトに Date、Time、または Timestamp のデータ型が含まれている場合、これらの型の形式は DateFormat アプリケーション固有情報でカスタマイ

ズできます。日付の形式は、java.text.SimpleDateFormat に定義されたパターンに従っている必要があります。 SQL の Date、Time、または Timestamp をストリングに変換する (およびその逆の変換を行う) 必要があるときに、DateFormat アプリケーション固有情報でこれらの型がカスタマイズされている場合、アダプターはこのカスタマイズされた形式を使用します。例えば、日付を dd/MM/yy 形式で、タイム・スタンプを yyyy/MM/dd HH:mm 形式でそれぞれ指定できます。DateFormat アプリケーション固有情報を指定しない場合、アダプターはデフォルトの形式を使用します。Date 型のデフォルトの形式は、「yyyy-MM-dd」、Timestamp 型は、「yyyy-mm-dd hh:mm:ss.ffffffff」、Time 型は、「HH:mm:ss」です。

注: Timestamp 型の形式は、JDBC 規格で定義され、SimpleDateFormat パターンには従いません。

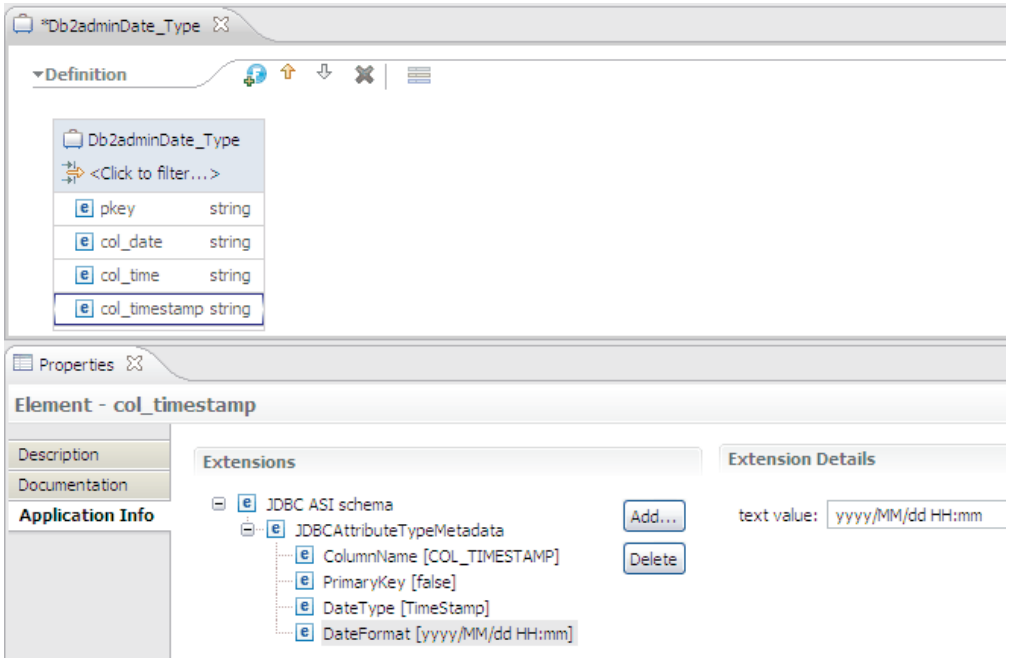


図 10. DateFormat アプリケーション固有情報とカスタマイズされた形式

テーブル・ビジネス・オブジェクトおよびビュー・ビジネス・オブジェクトは、Create、Update、Delete、Retrieve、RetrieveAll、Exists、および ApplyChanges の Outbound 操作をサポートします。階層型テーブル・ビジネス・オブジェクトに Exists 操作を実行すると、トップレベルのビジネス・オブジェクトのみが照会されます。

45 ページの図 11 に、ビジネス・オブジェクト・エディターに表示されたテーブル・ビジネス・オブジェクトを示します。このビジネス・オブジェクトでは、データベース表の列ごとに 1 つの属性が設定されています。表には子ビジネス・オブジェクトがないため、属性はすべて単純属性です。

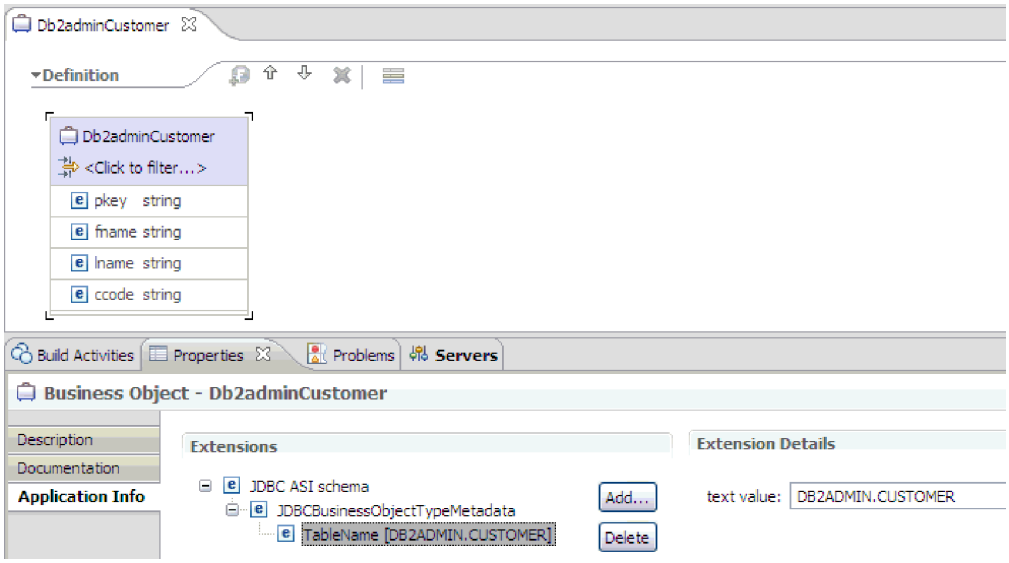


図 11. 子なしのテーブル・ビジネス・オブジェクト:

図 12 に、子テーブル・ビジネス・オブジェクトが 1 つあるテーブル・ビジネス・オブジェクトを示します。このビジネス・オブジェクトでは、データベース表の列ごとの単純属性に加えて、子ビジネス・オブジェクトを指す複合属性が設定されています。

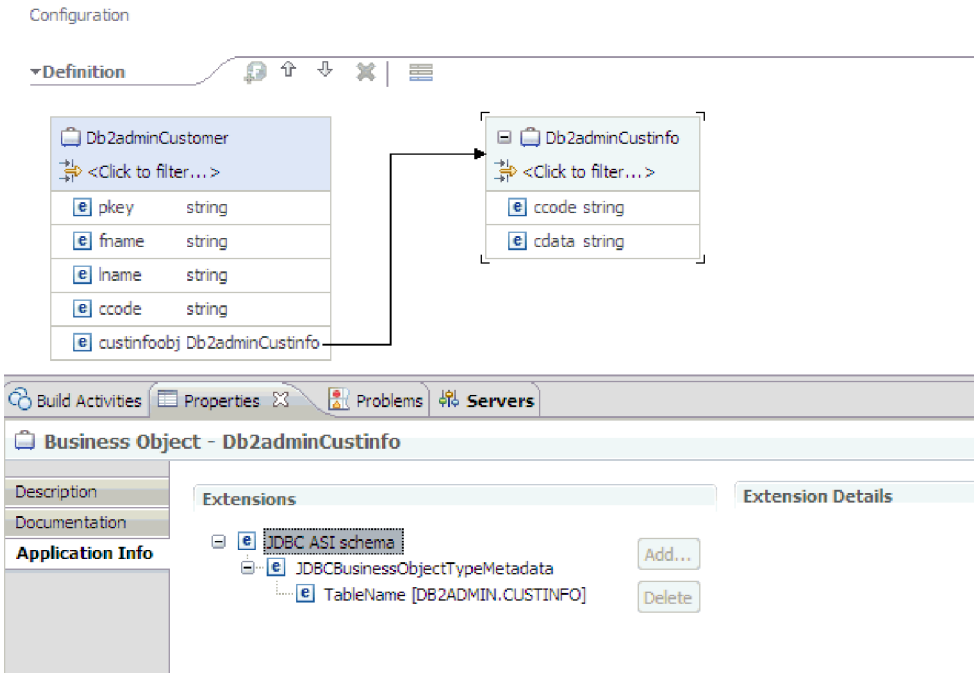


図 12. 子ビジネス・オブジェクトを 1 つ持つテーブル・ビジネス・オブジェクト:

Oracle データベースの場合、アダプターは、テーブル・ビジネス・オブジェクトで、配列、テーブル、構造体、ネストされた構造体などのユーザー定義型または複合データ型をサポートします。これらの型に対しては、型名および子属性の詳細が自動的にディスカバーされて、表示されます。アダプターでは、テーブル・ビジネス・オブジェクトの子ビジネス・オブジェクトとしてこれらのデータ型が処理され

ます。

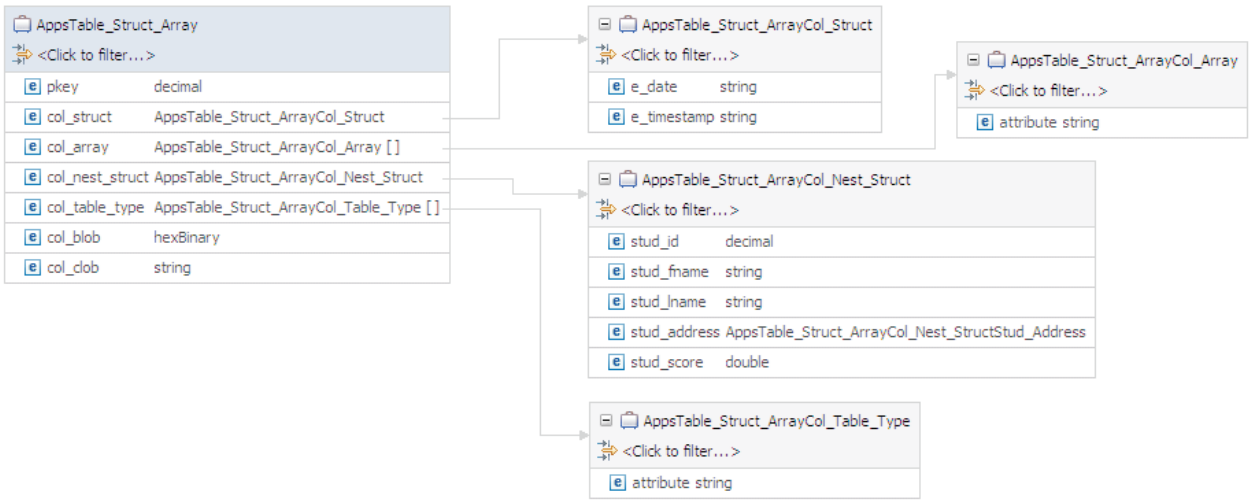


図 13. 列にユーザー定義型または複合型がある Oracle テーブル・ビジネス・オブジェクト

ストアード・プロシージャー・ビジネス・オブジェクトとストアード関数ビジネス・オブジェクトの場合

ストアード・プロシージャーまたはストアード関数のビジネス・オブジェクトでは、ストアード・プロシージャーまたはストアード関数のすべての入力パラメーターおよび出力パラメーターに、ビジネス・オブジェクトに対応する属性があります。入力または出力パラメーターのいずれかが、配列や構造体などの複合型である場合、対応するビジネス・オブジェクト属性は、配列または構造体の属性を含む子ビジネス・オブジェクトを持つ子ビジネス・オブジェクト型です。ストアード・プロシージャーが結果セットを戻した場合、戻された結果セットの属性を格納する子ビジネス・オブジェクトが作成されます。

ビジネス・オブジェクトに Date、Time、または Timestamp のデータ型が含まれている場合、これらの型の形式は DateFormat アプリケーション固有情報でカスタマイズできます。日付の形式は、java.text.SimpleDateFormat に定義されたパターンに従っている必要があります。 SQL の Date、Time、または Timestamp をストリングに変換する (およびその逆の変換を行う) 必要があるときに、DateFormat アプリケーション固有情報でこれらの型がカスタマイズされている場合、アダプターはこのカスタマイズされた形式を使用します。例えば、日付を dd/MM/yy 形式で、タイム・スタンプを yyyy/MM/dd HH:mm 形式でそれぞれ指定できます。DateFormat アプリケーション固有情報を指定しない場合、アダプターはデフォルトの形式を使用します。Date 型のデフォルトの形式は、「yyyy-MM-dd」、Timestamp 型は、「yyyy-mm-dd hh:mm:ss.ffffffff」、Time 型は、「HH:mm:ss」です。

注: Timestamp 型の形式は、JDBC 規格で定義され、SimpleDateFormat パターンには従いません。

ストアード・プロシージャーとストアード関数のビジネス・オブジェクトは、Execute Outbound 操作をサポートします。

ストアード・プロシージャー・ビジネス・オブジェクトの構造を下のサンプル・ファイルに示します。ビジネス・オブジェクト ScottStrtValues および ScottStrtValuesStrt が、1 つの入力タイプと 2 つの出力タイプを持つストアード・プロシージャーから生成されます。出力パラメーターの 1 つは、構造体データ型です。外部サービス・ウィザードによって、構造体型のビジネス・オブジェクト ScottStrtValuesStrt が生成され、子オブジェクトとして親ビジネス・オブジェクト ScottStrtValues に追加されます。親ビジネス・オブジェクト内の構造体型の属性については、ChildBOType アプリケーション固有情報が Struct に設定され、型が構造体であることを示します。ChildBOTypeName アプリケーション固有情報は、データベース内のユーザー定義構造体型の値に設定されます。次の例は、ストアード・プロシージャーのスキーマを示しています。

ScottStrtValues ビジネス・オブジェクトの例

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvalues" xmlns:scottstrtvalues=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvalues" xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata" xmlns:scottstrtvaluesstrt=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvaluesstrt">
<import namespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvaluesstrt"
schemaLocation="ScottStrtvaluesStrt.xsd"/>
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
schemaLocation="JDBCASI.xsd"/>

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvalues">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
<jdbcasi:MaxNumOfRetRS>0</jdbcasi:MaxNumOfRetRS>
<jdbcasi:ResultSet>false</jdbcasi:ResultSet>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="pkey" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>IP</jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="fname" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>OP</jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="strt" type="scottstrtvaluesstrt:ScottStrtvaluesStrt"
```

```
minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>OP</jdbcasi:SPParameterType>
<jdbcasi:ChildBOType>STRUCT</jdbcasi:ChildBOType>
<jdbcasi:ChildBOTypeName>STRUCT1</jdbcasi:ChildBOTypeName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

ScottStrtValuesStrt ビジネス・オブジェクトの例

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvaluesstrt" xmlns:scottstrtvaluesstrt=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvaluesstrt" xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
schemaLocation="JDBCASI.xsd"/>

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvaluesStrt">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="name" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="title" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="dept_num" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
```

```
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

照会ビジネス・オブジェクトの場合

データベース照会のビジネス・オブジェクトは、照会を実行する SQL ステートメントと、照会に必要なパラメーターを定義します。照会ビジネス・オブジェクトでは、Outbound 操作 RetrieveAll がサポートされています。

例えば、次の SELECT ステートメントを実行する照会ビジネス・オブジェクトがあるとします。

```
select C.pkey, C.fname, A.city from customer C, address A
    WHERE (C.pkey = A.custid) AND (C.fname like ?)
```

疑問符 (?) は、照会の入力パラメーターを示します。照会には複数のパラメーターを指定できます。各パラメーターは、SELECT ステートメントでは疑問符で示されています。サンプル照会ビジネス・オブジェクトの属性を表 7 に示します。照会ビジネス・オブジェクトには、抽出される列ごとの単純属性、パラメーターごとの単純属性、およびパラメーター置換の後も WHERE 節を保持する、照会の WHERE 節の「プレースホルダー・オブジェクト」があります。

表 7. 照会ビジネス・オブジェクトの属性

ビジネス・オブジェクト属性	**説明**
pkey	Customer 表のデータベース列 PKEY に対応
fname	Customer 表のデータベース列 FNAME に対応
city	Address 表のデータベース列 CITY に対応
parameter1	パラメーター。SELECT ステートメントの ? (疑問符) ごとに 1 つのパラメーターがあります。複数のパラメーターを持つ SELECT ステートメントでは、後続のパラメーターに parameter2、parameter3 のようにして名前が付けられます。
jdbcwhereclause	WHERE 節のプレースホルダー・オブジェクト

ビジネス・オブジェクトに Date、Time、または Timestamp のデータ型が含まれている場合、これらの型の形式は DateFormat アプリケーション固有情報でカスタマイズできます。日付の形式は、java.text.SimpleDateFormat に定義されたパターンに従っている必要があります。 SQL の Date、Time、または Timestamp をストリングに変換する (およびその逆の変換を行う) 必要があるときに、DateFormat アプリケーション固有情報でこれらの型がカスタマイズされている場合、アダプターはこのカスタマイズされた形式を使用します。例えば、日付を dd/MM/yy 形式で、タイム・スタンプを yyyy/MM/dd HH:mm 形式でそれぞれ指定できます。DateFormat アプリケーション固有情報を指定しない場合、アダプターはデフォルトの形式を使用します。Date 型のデフォルトの形式は、「yyyy-MM-dd」、Timestamp 型は、「yyyy-mm-dd hh:mm:ss.ffffffff」、Time 型は、「HH:mm:ss」です。

注: Timestamp 型の形式は、JDBC 規格で定義され、SimpleDateFormat パターンには従いません。

以下の図に、ビジネス・オブジェクト・エディターに表示されたサンプル照会のビジネス・オブジェクトを示します。

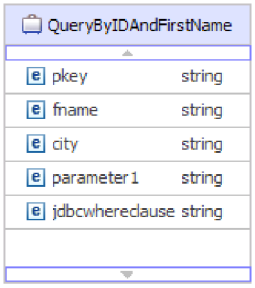


図 14. 照会ビジネス・オブジェクトの属性

この図は、照会ビジネス・オブジェクト例のアプリケーション固有情報を示しています。SelectStatement アプリケーション固有情報には、SELECT ステートメントが含まれています。

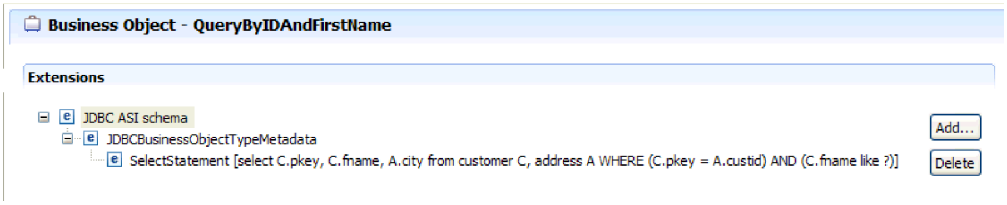


図 15. SELECT ステートメントは、ビジネス・オブジェクトのアプリケーション固有情報に保存されます。

Oracle データベースの場合、アダプターは、ビジネス・オブジェクトの照会結果で、配列、テーブル、構造体、ネストされた構造体などの複合データ型をサポートします。アダプターでは、バッチおよび照会のビジネス・オブジェクトにおいて、これらの複合型をパラメーターとしてサポートしていません。

バッチ SQL ビジネス・オブジェクトの場合

バッチ SQL ビジネス・オブジェクトは、データベース・アクションを実行する INSERT、UPDATE、および DELETE SQL ステートメントと、そのステートメントで必要とされるパラメーターを定義します。バッチ SQL ビジネス・オブジェクトでは、Outbound 操作 Execute がサポートされています。

例えば、次の INSERT および DELETE ステートメントを実行するバッチ SQL ビジネス・オブジェクトがあるとします。

```
Insert into customer (pkey,ccode,fname,lname) values(?,?,?,?);
Delete From Customer where pkey=?
```

各疑問符 (?) は、ステートメントのパラメーターを示します。バッチ SQL ビジネス・オブジェクト内の各ステートメントには複数のパラメーターを指定できます。各パラメーターは、ステートメントでは疑問符で示されています。バッチ SQL ビ

ジネス・オブジェクトには複数のステートメントを含めることができ、それぞれが独自のパラメーター・セットを持ちます。図 16 に、それぞれが 1 つ以上のパラメーターを持つ INSERT ステートメントと DELETE ステートメントが定義されている、バッチ SQL ビジネス・オブジェクトのビジネス・オブジェクトの形式を示します。

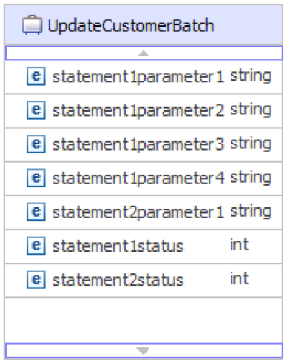


図 16. SQL ステートメントが 2 つのバッチ SQL ビジネス・オブジェクト

このビジネス・オブジェクトでは、statement1parameter1 や statement2parameter1 などの各ステートメントのパラメーターごとに属性が設定されています。また、statement1status や statement2status などの、各ステートメントの状況に関する属性もあります。ステートメント自体は、図 17 に示すように、ビジネス・オブジェクトについてのアプリケーション固有情報として格納されます。

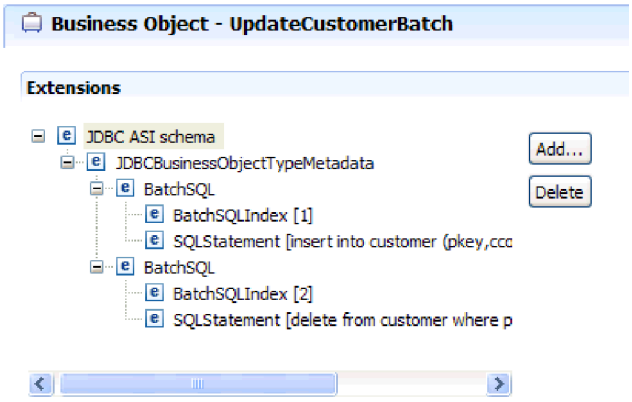


図 17. バッチ SQL ビジネス・オブジェクトのアプリケーション固有情報

ラッパー・ビジネス・オブジェクトの場合

ラッパー・ビジネス・オブジェクトにより、無関係のテーブル・ビジネス・オブジェクトおよびビュー・ビジネス・オブジェクトを 1 回の操作で扱うことができます。ラッパー・ビジネス・オブジェクトは、Create、Delete、Retrieve、および Update の Outbound 操作をサポートします。

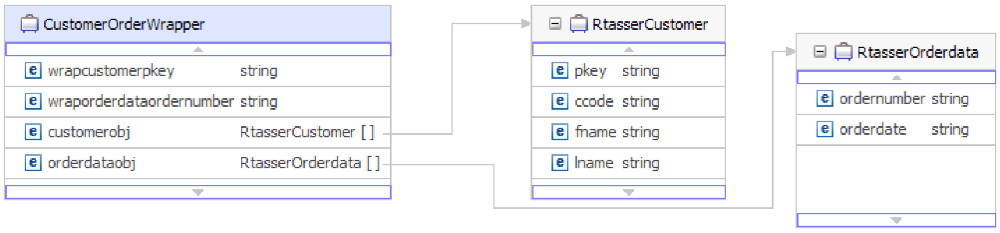


図 18. 2 つのテーブル・ビジネス・オブジェクトを含むラッパー・ビジネス・オブジェクト

ラッパー・ビジネス・オブジェクトには、子ビジネス・オブジェクトそれぞれの基本キーの単純属性が含まれています。フィールドの名前は「wrap」というストリングで、その後に、データベース表名とその表の基本キーの列名が続きます。ラッパー・ビジネス・オブジェクトには、テーブル・ビジネス・オブジェクトごとに 1 つの複合属性も含まれます。属性の名前は、ストリング「obj」を付加した表名です。複合属性のタイプは、対応するテーブル・ビジネス・オブジェクトの名前です。

ビジネス・グラフ

アダプターの構成時に、ビジネス・グラフを生成するオプションを選択することもできます。バージョン 6.0.2 では、トップレベルの各ビジネス・オブジェクトがビジネス・グラフに含まれていますが、このビジネス・オブジェクトには、実行する操作に関する追加情報を指定するために、バージョン 6.0.2 でアプリケーションが使用できる動詞が組み込まれています。バージョン 7.5 では、ビジネス・グラフが必要になるのは以下の状況に限られます。

- Outbound ApplyChanges 操作を使用する必要がある場合
- バージョン 7.5 より前のバージョンの IBM Integration Designer で作成されたモジュールにビジネス・オブジェクトを追加する場合

ビジネス・グラフが存在する場合、ビジネス・グラフは処理されますが、ApplyChanges 以外のすべての操作で動詞は無視されます。

ビジネス・オブジェクトの作成方法

ビジネス・オブジェクトを作成するには、IBM Integration Designer から起動される外部サービス・ウィザードを使用します。このウィザードにより、データベースに接続し、データベース・オブジェクトがディスカバーされ、表示されます。ビジネス・オブジェクトを作成するデータベース・オブジェクトを選択します。例えば、調べるスキーマを指定します。指定されたスキーマで、テーブル、ビュー、ストアード・プロシージャー、ストアード関数、シノニム、およびニックネームを選択します。また、ビジネス・オブジェクトを追加で作成できます。例えば、データベースに対して実行されるユーザー定義の SELECT、INSERT、UPDATE、または DELETE ステートメントの結果を表すビジネス・オブジェクトを作成できます。このウィザードでは、親子関係と、無関係なビジネス・オブジェクトをまとめるラッパーを使用してビジネス・オブジェクト階層を作成できます。

必要なビジネス・オブジェクトを指定し、これらのオブジェクトの階層を定義すると、ウィザードにより、選択されたオブジェクトを表すビジネス・オブジェクトが生成されます。また、アダプターに必要なその他の成果物も生成されます。

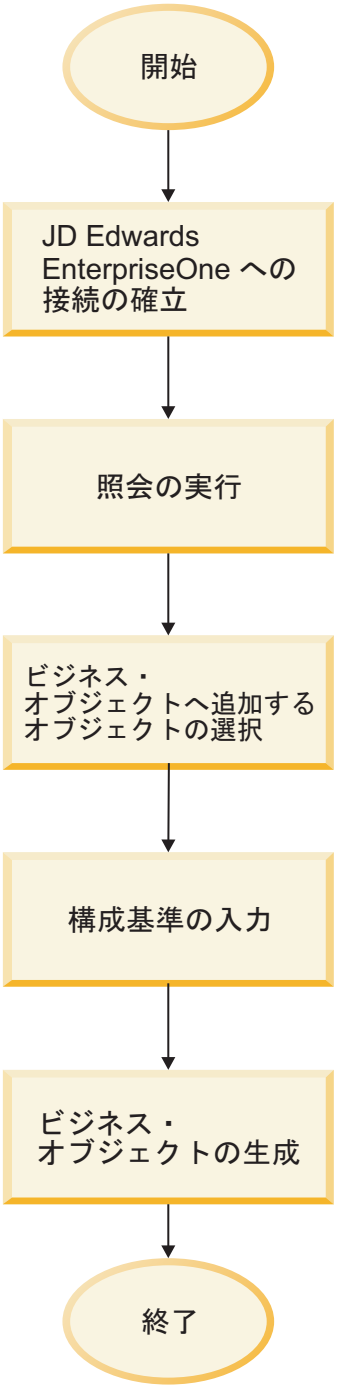


図 19. ビジネス・オブジェクトの作成方法

場合によっては、ウィザードで親子関係を完全に構成できないこともあります。これらの関係の場合は、IBM Integration Designer から起動するビジネス・オブジェクト・エディターを使用して、ウィザードによって作成されたビジネス・オブジェクト階層の定義を変更または完了します。詳しくは、 IBM Integration Designer インフォメーション・センター (リンク: http://bidoc.torolab.ibm.com:7500/help/index.jsp) で、ビジネス・オブジェクト・エディターを使用したビジネス・オブジェクトの変更方法を参照してください。

ビジネス・オブジェクト階層:

階層型ビジネス・オブジェクトでの親子関係とデータ所有権を使用して、データベース表の間の関係を定義します。無関係な表は、ラッパー・ビジネス・オブジェクトでグループ化できます。

ビジネス・オブジェクトは、フラットまたは階層です。フラットのビジネス・オブジェクトでは、すべての属性が単純属性であり、データベース表の 1 行を表します。階層には、関連するビジネス・オブジェクトまたは無関係なビジネス・オブジェクトを含めることができます。関連するビジネス・オブジェクトには、所有関係を伴う親子関係または所有権を伴わない親子関係があります。無関係なビジネス・オブジェクトの場合、ラッパー・ビジネス・オブジェクトが使用されます。

階層 ビジネス・オブジェクトという用語は、あらゆるレベルの子ビジネス・オブジェクトをすべて含む、完全なビジネス・オブジェクトを指します。*個別* ビジネス・オブジェクトという用語は、そのビジネス・オブジェクトが子ビジネス・オブジェクトを含んでいるか、そのビジネス・オブジェクトが親ビジネス・オブジェクトに属しているかに関係なく、1 つのビジネス・オブジェクトを指します。個別ビジネス・オブジェクトは、複数のデータベース表にまたがるビューを表すことができます。*トップレベルの* ビジネス・オブジェクトという用語は、階層の頂点にあり、それ自体は親ビジネス・オブジェクトを持たない個別ビジネス・オブジェクトを指します。

階層ビジネス・オブジェクトは、子ビジネス・オブジェクト、子ビジネス・オブジェクトの配列、またはその組み合わせを表す属性を持ちます。そして、子ビジネス・オブジェクトも、それぞれ自身の子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を持つことができます。この関係は階層の下に向かって続きます。

単一カーディナリティー関係 は、親ビジネス・オブジェクト内の属性が 1 つの子ビジネス・オブジェクトを表すときに発生します。属性は、子ビジネス・オブジェクトと同じ型です。アダプターは、単一カーディナリティー関係と、所有権のない単一カーディナリティー関係およびデータをサポートします。

複数カーディナリティー関係 は、親ビジネス・オブジェクト内の属性が子ビジネス・オブジェクトの配列を表すときに発生します。属性は、子ビジネス・オブジェクトと同じ型です。

以下のタイプの関係をビジネス・オブジェクトの間で使用して、データベース表を示す階層を定義します。

- 単一カーディナリティー関係
- 単一カーディナリティー関係および所有権のないデータ
- 複数カーディナリティー関係
- 複数の親を持つ子ビジネス・オブジェクト

また、無関係なビジネス・オブジェクトを 1 つのラッパー・ビジネス・オブジェクトにまとめることができます。

どちらのカーディナリティーのタイプでも、親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は、その関係を格納するビジネス・オブジェクトのキー属性に含まれるアプリケーション固有情報によって記述されます。

ビジネス・オブジェクトの単一カーディナリティー関係:

単一カーディナリティー関係では、親ビジネス・オブジェクト内の属性が 1 つの子ビジネス・オブジェクトを表します。この場合、その属性は、その子ビジネス・オブジェクトと同じタイプです。アダプターは、単一カーディナリティー関係と、所有権のない単一カーディナリティー関係およびデータをサポートします。

単一カーディナリティー関係

通常、単一カーディナリティーの子ビジネス・オブジェクトを含むビジネス・オブジェクトには、関係を表すための属性が 2 つ以上含まれます。それらの属性のうちの 1 つの属性のタイプは、子のタイプと同じです。もう 1 つの属性は、子の基本キーを、外部キーとして親に格納するための単純属性です。親には、子に含まれる基本キー属性と同数の外部キー属性が含まれます。

図 20 に、一般的な単一カーディナリティー関係を示します。この例では、ParentBOName オブジェクト内の FKey は、子の基本キーを含む単純属性であり、同様に、ParentBOName オブジェクト内にある Child(1) は、子ビジネス・オブジェクトを表す属性です。

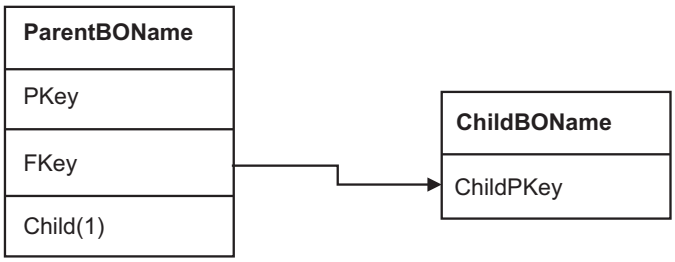
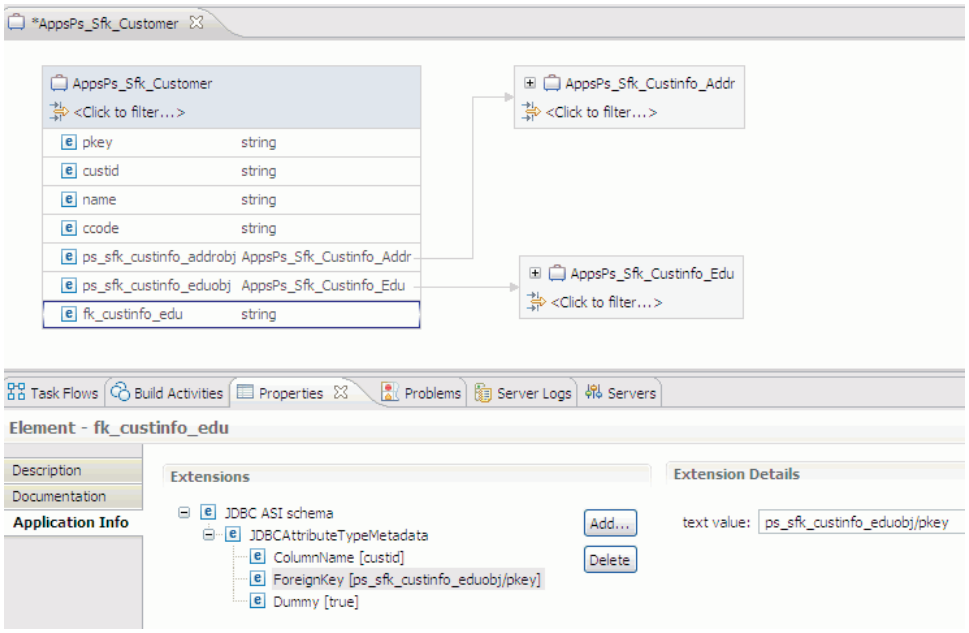


図 20. 典型的な単一カーディナリティー関係

関係を設定する外部キーが親に格納されるため、それぞれの親には特定のタイプの子ビジネス・オブジェクトを 1 つだけ格納できます。親に複数の子があり、それらの子の外部キーが親の同じ列を参照している場合は、親ビジネス・オブジェクトを編集し、親を参照している子と同じ数のダミーの列を追加します。次の例に示すように、列ごとに、ColumnName、 ForeignKey、および Dummy のアプリケーション固有情報を追加します。

1. ColumnName ASI を、子が参照する親の列に設定します。



2. 対応する ForeignKey ASI を設定します。
3. Dummy ASI を "true" に設定します。
4. *.xsd ファイルを開き、属性 nillable="true" を追加します。

```
<xsd:element minOccurs="0" name="fk_custinfo_edu" nillable="true" type="xsd:string">
  <xsd:annotation>
    <xsd:appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <metadata:JDBCAttributeTypeMetadataxmlns:metadata="http://www.ibm.com/xmlns/prod/web
      <metadata:ColumnName>custid</metadata:ColumnName>
      <metadata:ForeignKey>ps_sfk_custinfo_eduobj/pkey</metadata:ForeignKey>
      <metadata:Dummy>true</metadata:Dummy>
      </metadata:JDBCAttributeTypeMetadata>
    </xsd:appinfo>
   </xsd:annotation>
</xsd:element>
```

親ビジネス・オブジェクトは、所有関係にある単一カーディナリティーの子と、所有関係にない単一カーディナリティーの子を持つことができます。参照テーブルは所有関係を伴わない関係に使用します。所有権は、アプリケーション固有情報 Ownership の値により示されます。

単一カーディナリティー関係および所有権のないデータ

通常、各親ビジネス・オブジェクトは、その親ビジネス・オブジェクトに含まれる子ビジネス・オブジェクトの内部のデータを所有しています。例えば、各 Customer ビジネス・オブジェクトが Address ビジネス・オブジェクトを 1 つ含んでいる場合に、新しい顧客が作成されると、Customer テーブルと Address テーブルの両方に新しい行が 1 行挿入されます。挿入された新しい住所は、その新しい顧客に固有です。同様に、Customer テーブルから顧客を削除すると、Address テーブルからその顧客の住所も削除されます。

ただし、複数の階層ビジネス・オブジェクトに同一のデータが含まれており、どのビジネス・オブジェクトもそのデータを所有していない場合があります。例えば、Address データベース表に、StateProvince ルックアップ・テーブルへの参照が含まれていると仮定します。このルックアップ・テーブルはほとんど更新されることが

なく、住所データからは独立して保守されています。このため、住所データの作成または変更により、このルックアップ・テーブル内の都道府県データが影響を受けることはありません。ただし、StateProvince ビジネス・オブジェクトを Address ビジネス・オブジェクトと一緒に検索できるようにするには、StateProvince は Address の単一カーディナリティーの子であり、データの所有権のない関係を定義する必要があります。

データベース設計にルックアップ・テーブルが含まれている場合、ビジネス・オブジェクト設計はデータベース設計と少し異なります。この理由は、アダプターはテーブル・ビジネス・オブジェクトおよびその子テーブル・ビジネス・オブジェクトのためだけにデータを取得するためです。ルックアップ・テーブルを使用するには、テーブル間で単一カーディナリティーの親子関係を所有関係なしで作成する必要があります。StateProvince ルックアップ・テーブルはデータベース内の Address テーブルの子ではありませんが、対応する StateProvince ビジネス・オブジェクトは Address テーブル・ビジネス・オブジェクトの単一カーディナリティーの子です。これは、各アドレスに単一の都道府県が含まれているためです。しかし、Address ビジネス・オブジェクトは StateProvince ビジネス・オブジェクトを「所有」しません。住所を変更しても、都道府県のリストは変更されません。

アダプターが階層ビジネス・オブジェクトを Create、Delete、または Update 要求とともに受信した場合、アダプターは所有関係にない単一カーディナリティーの子ビジネス・オブジェクトの作成、削除、または更新を行いません。アダプターは、このようなビジネス・オブジェクトに対しては、Retrieve 操作のみを実行します。このような単一カーディナリティーのビジネス・オブジェクトの検索に失敗した場合、アダプターはエラーを戻して処理を停止します。ルックアップ・テーブルのビジネス・オブジェクトへの値の追加または変更は行いません。

非正規化データおよび所有権のないデータ

所有関係を伴わない包含関係には、静的参照テーブルの使用を容易にするだけでなく、正規化データと非正規化データを同期化するという別の機能があります。

正規化データから非正規化データへの同期: 所有関係を伴わない関係の場合、正規化アプリケーションから非正規化アプリケーションへの同期化を行うときに、データを作成または変更することができます。例えば、正規化されたソース・アプリケーションが、A と B という 2 つのテーブルにデータを格納するものとします。また、非正規化されている宛先アプリケーションでは、1 つのテーブルにすべてのデータが格納され、エンティティー A のそれぞれにエンティティー B のデータが重複して格納されるものとします。

この例では、テーブル B のデータの変更をソース・アプリケーションから宛先アプリケーションに同期化するには、テーブル B のデータが変更されるたびにテーブル A のイベントを起動する必要があります。さらに、テーブル B のデータはテーブル A に重複して格納されているので、テーブル A の行ごとに、テーブル B で変更されたデータが含まれるビジネス・オブジェクトを送信しなければなりません。

注: 非正規化されたテーブルを更新する場合、1 行を更新した結果複数の行が変更されることがないように、レコードごとに固有キーを持たせるようにしてください。そのようなキーが存在しない場合、アダプターでは複数レコードが更新されたことを示すエラーが発生します。

非正規化データから正規化データへの同期: 非正規化されているソース・アプリケーションから正規化されている宛先アプリケーションにデータを同期化する場合、アダプターは、正規化されているアプリケーションに含まれる所有関係にないデータに関しては、作成、削除、または更新を行いません。

正規化されているアプリケーションにデータを同期化する場合、アダプターは、所有関係にない単一カーディナリティーの子をすべて無視します。そのような子のデータを作成、除去、または変更するには、データを手動で処理する必要があります。

複数カーディナリティー関係:

複数カーディナリティー関係では、親ビジネス・オブジェクト内の属性が子ビジネス・オブジェクトの配列を表します。属性は、子ビジネス・オブジェクトと同じ型です。アプリケーションが単一の子エンティティーを格納する場合を除き、関係を記述する外部キーは子に格納されます。親子関係は親に格納されます。

通常、子ビジネス・オブジェクトの配列を含むビジネス・オブジェクトには、関係を表す属性が 1 つだけ含まれており、通常はこの属性が基本キーになります。この属性のタイプは、子ビジネス・オブジェクトと同じタイプの配列です。親が複数の子を含むようにするため、関係を設定する外部キーは子に格納されます。

したがって、どの子にも、親の基本キーを外部キーとして含む単純属性が 1 つ以上存在します。子には、親に含まれる基本キー属性と同数の外部キー属性が含まれます。

関係を設定する外部キーが子に格納されるので、それぞれの親は、1 つ以上の子を持つことができます (子を持たないことも可能です)。

図 21 に、複数カーディナリティー関係を示します。この例では、 3 つの ChildBOName ボックス内の ParentID は、親の基本キーを含む単純属性であり、ParentBOName ボックス内にある Child(1) は、子ビジネス・オブジェクトの配列を表す属性です。

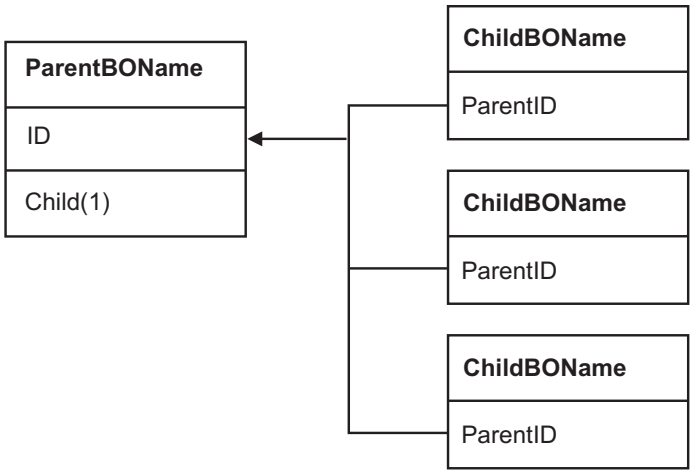


図 21. N>1 のビジネス・オブジェクトの複数カーディナリティーの関係

複数カーディナリティーの関係は、N=1 の関係である場合があります。アプリケーションによっては、親子関係を親ではなく子に格納するように、子エンティティーを 1 つ格納するものがあります。すなわち、子は、親の基本キーに格納されている値と同じ値の外部キーが格納されます。

このタイプの関係がアプリケーションで使用されるのは、子のデータが親から独立して存在しておらず、親を介してのみそのデータにアクセスできる場合です。このような子のデータでは、子とその外部キー値を作成するために、親とその基本キー値があらかじめ存在していなければなりません。

図 22 に、この関係タイプを示します。

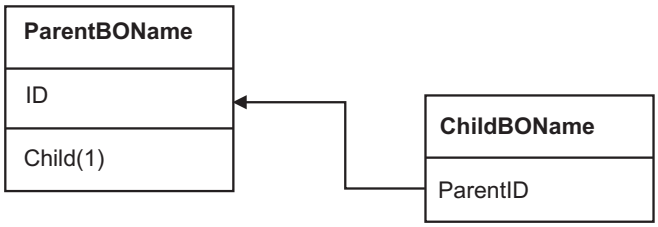


図 22. N=1 の場合の複数カーディナリティーの関係

複数の親テーブルを持つデータベース表:

データベース内の子テーブルが複数の親テーブルを持つ場合、アセンブリー・エディターを使用して追加の親ビジネス・オブジェクトを手動で構成する必要があります。外部サービス・ウィザードでは 1 つの親しか構成されません。

ビジネス・オブジェクト・スキーマ:

ビジネス・オブジェクト・スキーマは、外部サービス・ウィザードを実行するときに選択したデータベース・オブジェクトから作成されます。各データベース・オブジェクトはトップレベルのビジネス・オブジェクトになります。

スキーマは、ビジネス・オブジェクト名とアプリケーション固有情報を定義します。ビジネス・オブジェクトと、その属性およびアプリケーション固有情報は、スキーマでは次のように表現されます。

- ビジネス・オブジェクトは、複合タイプの定義にマップします。
- ビジネス・オブジェクトのアプリケーション固有の情報は、複合タイプでの注釈に含まれます。
- ビジネス・オブジェクトの属性は、エレメント・タイプの定義にマップします。
- ビジネス・オブジェクト内の各プロパティーのアプリケーション固有情報は、エレメント・タイプに関する注釈に含まれます。

ビジネス・オブジェクトおよび属性のためのアプリケーション固有のプロパティーのテンプレートが、アダプターのメタデータ・スキーマに定義されています。このスキーマ・ファイルの名前は `JDBCASI.xsd` です。アダプターに対して生成されたスキーマ・ファイルの注釈には、このテンプレートへの参照が含まれます。

テーブル、ビュー、およびシノニムについての概要

データベースは、テーブル、ビュー、シノニムなど、一般的データベース・オブジェクトを提供します。

テーブルは、データベース内の一般的なデータベース・スキーマ・オブジェクトです。テーブル・オブジェクトは、データベース内にデータを保管するために使用されます。DB2、Oracle、SQL Server などのデータベースでは、データベースにデータを入力するための多くのテーブル・タイプが用意されていて、例えば、オブジェクト、標準、ネスト化、クラスター化、索引編成などがあります。

ビューは、仮想テーブルまたは保管照会文と見なされるデータベース・オブジェクトです。ビューは、それ自体にデータを物理的に格納するのではなく、ビューが作成された基本テーブルからデータを取り出します。テーブルを対象に実行されるすべての操作は、ビューに対しても実行できます。

シノニムは、テーブルやビューといったデータベース・オブジェクト、およびその他のデータベース・オブジェクトに付与される別名または代替名です。

ラッパー・ビジネス・オブジェクト

ラッパー・ビジネス・オブジェクトにより、無関係のテーブル・ビジネス・オブジェクトおよびビュー・ビジネス・オブジェクトを 1 回の操作で扱うことができます。Wrapper ビジネス・オブジェクトには、子ビジネス・オブジェクトそれぞれの基本キーの単純属性が含まれています。フィールドの名前は「wrap」というストリングで、その後に、データベース表名とその表の基本キーの列名が続きます。ラッパー・ビジネス・オブジェクトには、テーブル・ビジネス・オブジェクトごとに 1 つの複合属性も含まれます。属性の名前は、ストリング「obj」を付加した表名です。複合属性のタイプは、対応するテーブル・ビジネス・オブジェクトの名前です。

以下の例では、生成されたラッパー・ビジネス・オブジェクト WrapperBO に 3 個の子ビジネス・オブジェクトが含まれています。基本キー (pkey) を持つ Db2adminCustomer、基本キー (ccode) を持つ Db2adminCustInfo、および、複合基本キー (fname、lname) を持つ Db2adminCustomer_Compositekey です。このラッパー・ビジネス・オブジェクトに対する Create、Update、Delete、および Retrieve 操作の実行方法について以下のセクションで説明します。

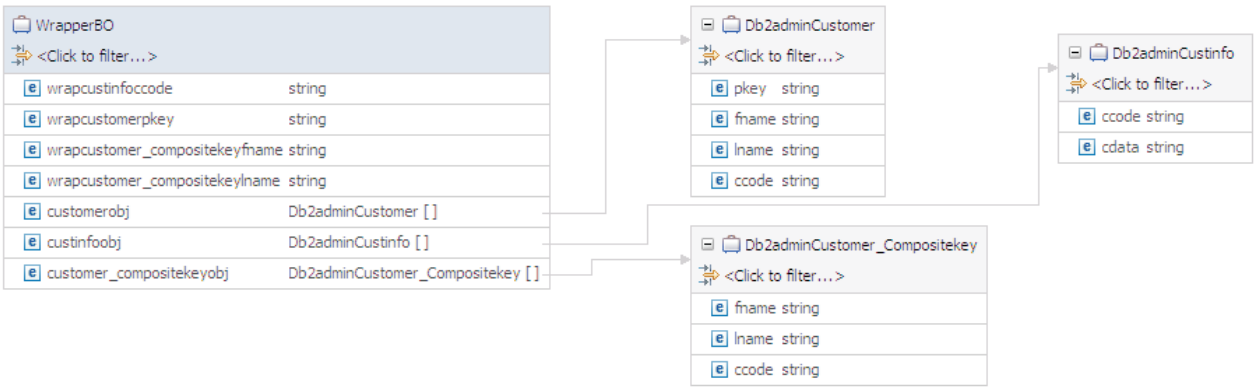


図 23. ラッパー・ビジネス・オブジェクト

ラッパー・ビジネス・オブジェクトは外部サービス・ウィザードの「複合プロパティーの指定」ウィンドウで作成することができます。ラッパー・ビジネス・オブジェクトの作成方法については、 147 ページの『操作のグローバル・プロパティーの設定およびラッパー・ビジネス・オブジェクトの作成』を参照してください。 ラッパー・ビジネス・オブジェクトは、Create、Retrieve、Update、および Delete 操作をサポートします。各操作の動作を次の表に示します。

表 8. ラッパー・ビジネス・オブジェクトに対する操作の動作

	アクション	
操作	*ラッパー・ビジネス・オブジェクトのいずれかの値を設定し、そのキーに対応する子ビジネス・オブジェクト・インスタンスがない場合*	*ラッパー・ビジネス・オブジェクトにキーの値を設定し、子ビジネス・オブジェクト・インスタンスを追加する場合:* 1. **子ビジネス・オブジェクト・インスタンスの基本キーが設定されていない場合、子ビジネス・オブジェクト・インスタンスの基本キー値は、ラッパー・ビジネス・オブジェクト内の対応するキーからコピーされ、操作の実行に使用されます。** 2. **子ビジネス・オブジェクト・インスタンスの基本キー値が設定されていて、ラッパー・ビジネス・オブジェクト内の対応するキー値と異なる場合、ラッパー・ビジネス・オブジェクト内のキー値は無視され、子ビジネス・オブジェクトのキー値が操作の実行に使用されます。**
Create	Create 操作は正常に実行され、返される子ビジネス・オブジェクトは空です。データベースに挿入されるレコードはありません。	子ビジネス・オブジェクト・インスタンスのレコードがデータベース内に存在しない場合、レコードはデータベースに挿入されます。子ビジネス・オブジェクト・インスタンスのレコードがデータベース内に存在する場合、アダプターは UniqueConstraintViolatedException を戻します。
Retrieve	• キーが有効な場合、キーに対応するレコードが戻されます。 • キーが無効な場合、戻されるレコードはありません。	• 子ビジネス・オブジェクト・インスタンスのレコードがデータベース内に存在する場合、そのレコードが戻されます。 • 子ビジネス・オブジェクト・インスタンスのレコードがデータベース内に見つからない場合、アダプターは RecordNotFoundException を戻します。

表 8. ラッパー・ビジネス・オブジェクトに対する操作の動作 (続き)

操作	アクション	
	ラッパー・ビジネス・オブジェクトのいずれかの値を設定し、そのキーに対応する子ビジネス・オブジェクト・インスタンスがない場合	*ラッパー・ビジネス・オブジェクトにキーの値を設定し、子ビジネス・オブジェクト・インスタンスを追加する場合:* 1. **子ビジネス・オブジェクト・インスタンスの基本キーが設定されていない場合、子ビジネス・オブジェクト・インスタンスの基本キー値は、ラッパー・ビジネス・オブジェクト内の対応するキーからコピーされ、操作の実行に使用されます。** 2. **子ビジネス・オブジェクト・インスタンスの基本キー値が設定されていて、ラッパー・ビジネス・オブジェクト内の対応するキー値と異なる場合、ラッパー・ビジネス・オブジェクト内のキー値は無視され、子ビジネス・オブジェクトのキー値が操作の実行に使用されます。**
Update	Update 操作は正常に実行されます。データベース内で更新されるレコードはありません。	• 子ビジネス・オブジェクトが存在しない場合、レコードはデータベースに挿入されます。 • 子ビジネス・オブジェクトが存在している場合、レコードはデータベース内で更新されます。 • いずれかの子ビジネス・オブジェクトで基本キーが設定されていない場合、レコードはラッパー・ビジネス・オブジェクト内の対応するキーからコピーされたキー値で更新されます。
Delete	Delete 操作は、キーが有効な場合は正常に実行され、対応する子ビジネス・オブジェクトが削除されます。	指定された子ビジネス・オブジェクトのみが削除されます。 • いずれかの子ビジネス・オブジェクトで基本キーが設定されていない場合、ラッパー・ビジネス・オブジェクトのキー値を持つレコードが削除されます。 • 子ビジネス・オブジェクトがデータベース内に存在しない場合、アダプターは RecordNotFoundException を戻します。

ストアード・プロシージャーの概要

モジュールから Execute 操作で実行するビジネス・オブジェクトをストアード・プロシージャーとすることができます。ストアード・プロシージャーは、標準 SQL の代わりに任意のビジネス・オブジェクトに対する操作を実行するか、または、ある操作の実行前または実行後に追加のアクションを実行することができます。

ストアード・プロシージャーは、複数の SQL ステートメントのグループであり、1 つの論理単位を形成して特定のタスクを実行します。ストアード・プロシージャーは、アダプターがオブジェクトに対して実行する一連の操作または照会を、データベース・サーバー内にカプセル化したものです。アダプターは次のいずれかの方法でストアード・プロシージャーを使用します。

- データベースに対して実行するストアード・プロシージャー・ビジネス・オブジェクトを作成する
- ビジネス・オブジェクトの操作用に提供されている SQL ステートメントを置換することによって、または、操作を実行する前または後にアクションを実行することによって、ビジネス・オブジェクトの操作を拡張する
- ストアード・プロシージャーのラッパー・ビジネス・オブジェクトを作成する

  ストアード・プロシージャーの Wrapper ビジネス・オブジェクトにより、複数のストアード・プロシージャーをグループ化したり、あるいは、同一ストアード・プロシージャーを複数回実行することができます。ストアード・プロシージャーに対応するビジネス・オブジェクトは、ラッパー・ストアード・プロシージャーの複数カーディナリティーの子として追加されます。

ストアード・プロシージャー・ビジネス・オブジェクトの概要

データベースのストアード・プロシージャーまたはストアード関数に対応するストアード・プロシージャー・ビジネス・オブジェクトを作成することができます。次に Execute 操作を使用して、データベース内のデータに対してストアード・プロシージャーを実行できます。

外部サービス・ウィザードで、ストアード・プロシージャーまたはストアード関数を実行するストアード・プロシージャー・ビジネス・オブジェクトを作成できます。ウィザードではビジネス・オブジェクトを作成するために、データベース内のストアード・プロシージャーまたはストアード関数を検査します。ストアード・プロシージャー・ビジネス・オブジェクトでは、各パラメーターごとに属性が 1 つあります。

注: データベース・スキーマに、同じ名前を持つ複数のストアード・プロシージャーがあり、かつ、それらのストアード・プロシージャーが異なるパラメーターを持っている場合、外部サービス・ウィザードは、どのストアード・プロシージャーが選択されるかを識別できません。したがって、ストアード・プロシージャーは固有の名前を持つ必要があります。

パラメーター属性が単純データ型の場合は、パラメーターのサンプル値についての属性が 1 つあります。ウィザードはストアード・プロシージャーを保存する前にストアード・プロシージャーを検証するとき、サンプル値を使用します。アダプターはストアード・プロシージャーからの結果を使用してパラメーターを検証し、返さ

れる結果セットの最大数を取得し、これらの結果セットのメタデータを使用して子ビジネス・オブジェクトを生成できるようにします。ストアード・プロシージャー・ビジネス・オブジェクトを検証すると、ウィザードではストアード・プロシージャー・ビジネス・オブジェクトの階層を自動的に生成します。

ストアード・プロシージャーに、Struct、Array、または結果セットなどの複合データ型の入力または出力パラメーターまたは戻り値パラメーターがある場合、これらの各パラメーターに対応するデータ型をウィザードで選択し、対応するユーザー定義タイプの名前を指定する必要があります。 Struct または Array 型のパラメーターの場合、プロパティー SPComplexParameterTypeName に保存されている対応するユーザー定義タイプ名を指定する必要があります。

例えば、Struct_TEMP という名前の Struct オブジェクトをデータベースに作成し、1 つの入力パラメーターとして型を設定する場合は、このプロパティーの値を Struct_TEMP に設定する必要があります。ウィザードはこの型名を使用して、対応する子ビジネス・オブジェクトの生成のためのメタデータを決定します。ストアード・プロシージャーから結果セットが返される場合は、その結果セットの数をプロパティー MaxNumberOfResultSets に設定する必要があります。この値は、アダプター・ランタイムによって処理される、返される結果セットの最大数を表します。

ディスカバリー時および実行時に、IBM WebSphere Adapter for JDBC は、ストアード・プロシージャーの実行から返された結果セットに含まれる列に名前が付いているものと見なします。ストアード・プロシージャーによっては、名前のない列を含む結果セットを返すことがあります。例えば、次の例のような SQL ステートメントを使用するストアード・プロシージャーは、名前のない列を含む結果セットを返します。

```
SELECT COUNT(*) FROM EMPLOYEE;
SELECT 111,222,333 FROM CUSTOMER;
```

Oracle は、返される結果セットのテーブル列に「ダミー」の名前を割り当てることで、そのような SQL SELECT ステートメントを処理します。例えば、例の SELECT ステートメントでは、テーブル列に、count(*)、d1, d2, d3 などの名前を割り当てます。

返される結果セットに名前のないテーブル列が含まれている (データベースによりダミーの名前が割り当てられなかった) 場合、アダプターがそうした列にダミーの名前を作成します。

データベースあるいはアダプターのいずれかによって生成されたダミーの列名は、ストアード・プロシージャー・ビジネス・オブジェクトの属性に割り当てられます。

名前のないテーブル列にダミー名を割り当てる (アダプターまたはデータベースによる) 動作によって、ディスカバリー時あるいは実行時のストアード・プロシージャーの実行が確実に正常に行われるようになります。

ストアード・プロシージャー・ビジネス・オブジェクトの場合、ウィザードは、ネストされた Struct または Array オブジェクトをサポートし、任意の数の層にわたる

ネストされた階層をサポートできます。ウィザードは、これらのネストされたすべての Struct または Array オブジェクトに対応する子ビジネス・オブジェクトを生成できます。

表 9. ストアード・プロシージャー・ビジネス・オブジェクトの複合データ型プロパティー

プロパティー名	タイプ	説明

SPComplexParameterType	String	値は次のいずれかです。
		
		`Array`
		`ResultSet`
		`Struct`

SPComplexParameterTypeName	String	ユーザー定義タイプの名前。このプロパティーは、SPComplexParameterType の値が `Struct` または `Array` の場合に必要です。
MaxNumberOfResultSets	Integer	アダプター・ランタイムによって処理される、返される結果セットの最大数。ウィザードにより、この数のビジネス・オブジェクトが作成されます。

操作の代わりまたは追加で使用するストアード・プロシージャー

アダプターがデータベース内で、アダプターが操作を実行するのに使用する SQL ステートメントの代わりか、その前または後でストアード・プロシージャーを使用するよう指定することができます。各ビジネス・オブジェクトはそれぞれの操作で使用する、異なるセットのストアード・プロシージャーを持つことができます。

アダプターは、単純な SQL ステートメントを使用して、
Create、Update、Delete、Retrieve、または RetrieveAll 操作を実行できます。SQL ステートメントで使用される列名は、属性のアプリケーション固有情報から取り出されます。WHERE 文節は、ビジネス・オブジェクトで指定されたキー値を使用して構成されます。各照会は、複数のテーブルにまたがることはできません。ただし、ビューに送ることはできます。しかし、アダプターによって提供された SQL ステートメントは、ストアード・プロシージャーおよびストアード関数を使用して置換または拡張することができます。

アダプターは以下の状況で、ストアード・プロシージャーまたはストアード関数を呼び出すことができます。

- ビジネス・オブジェクトを処理する前に、操作準備処理を行う。
- ビジネス・オブジェクトを処理した後で、操作後の処理を行う。
- 単純な Create、Update、Delete、Retrieve または RetrieveAll ステートメントを使用せずにビジネス・オブジェクトに対して一連の操作を実行する。

階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、トップレベルのビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーをトップレベルのビジネス・オブジェクトに関連付けても、各子ビジ

ネス・オブジェクトに関連付けないと、そのトップレベルのビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

ストアード・プロシージャーのアプリケーション固有情報エレメント、各エレメントの目的および使用法を表 10 に示します。各エレメントの完全な説明は、表に続くセクションで説明します。ストアード・プロシージャーの定義例を 70 ページの『ストアード・プロシージャーの例』 に示します。

表 10. ストアード・プロシージャーのアプリケーション固有情報 (テーブルおよびビュー・ビジネス・オブジェクト内)

記述名	エレメント名	目的
ストアード・プロシージャー・タイプ	StoredProcedureType	ストアード・プロシージャー・タイプは、使用するストアード・プロシージャーのタイプを定義します。これにより、ストアード・プロシージャーが呼び出される時点 (ビジネス・オブジェクトの処理前など) が決まります。
ストアード・プロシージャー名	StoredProcedureName	適切な StoredProcedureType と関連したストアード・プロシージャーの名前です。
結果セット	ResultSet	この値は、ストアード・プロシージャーが結果セットを戻すかどうかを指定します。結果のセットが戻される場合は、結果のセットの行で戻される値を使用して、現在のビジネス・オブジェクトの複数カーディナリティーの子が作成されます。
パラメーター	Parameters	各 Parameters エレメントは、ストアード・プロシージャーまたはストアード関数の 1 つのパラメーターを記述します。
戻り値	ReturnValue	関数によって値が返されるため、プロシージャー呼び出しではなく関数呼び出しであることを示す値。

ストアード・プロシージャー・タイプ

ストアード・プロシージャー・タイプは、使用するストアード・プロシージャーのタイプを定義します。このストアード・プロシージャー・タイプによって、そのストアード・プロシージャーがいつ呼び出されるか (ビジネス・オブジェクトの処理前など) が決まります。

表 11. 「ストアード・プロシージャー・タイプ」エレメントの特性

必須	はい
デフォルト	なし

表 11. 「ストアード・プロシージャー・タイプ」エレメントの特性 (続き)

使用可能な値	次のいずれかです。 • Before*Operation*SP • After*Operation*SP • *Operation*SP *Operation* には、操作名 (Create、Update、Delete、Retrieve、または RetrieveAll) の 1 つを指定します。
サポートされる双方向変換	いいえ
プロパティー・タイプ	String
使用上の注意	RetrieveAll に関連するストアード・プロシージャーのタイプは、トップレベルのビジネス・オブジェクトにのみ適用されます。 選択した任意のアプリケーション固有情報を StoredProcedureType プロパティーから除去できます。対応する操作のアプリケーション固有情報プロパティー・グループもすべて削除されます。
例	• CreateSP: 作成操作を実行します • UpdateSP: 更新操作を実行します • BeforeCreateSP: ビジネス・オブジェクトの作成前に実行します • AfterCreateSP: ビジネス・オブジェクトの作成後に実行します • AfterDeleteSP: ビジネス・オブジェクトの削除後に実行します

ストアード・プロシージャー名

適切な StoredProcedureType と関連したストアード・プロシージャーの名前です。

表 12. 「ストアード・プロシージャー名」エレメントの特性

必須	はい
デフォルト	なし
サポートされる双方向変換	はい
プロパティー・タイプ	String

結果セット

この値は、ストアード・プロシージャーが結果セットを戻すかどうかを決定します。結果のセットが戻される場合は、結果のセットの行で戻される値を使用して、現在のビジネス・オブジェクトの複数カーディナリティーの子が作成されます。

表 13. 「結果セット」エレメントの特性

必須	はい
デフォルト	なし
使用可能な値	`True` `False`

表 13. 「結果セット」エレメントの特性 (続き)

サポートされる双方向変換	いいえ
プロパティー・タイプ	Boolean
使用上の注意	Oracle ユーザーについて: ストアード・プロシージャーが結果セットを戻す場合、外部サービス・ウィザードが終了した後でビジネス・オブジェクト・エディターを使用して、この属性が `true` に設定されていることを確認してください。Oracle JDBC ドライバーがこの値を正しく戻さない場合があります。

Parameters

ストアード・プロシージャーまたはストアード関数のパラメーターごとに 1 つの Parameters エレメントがあります。各 Parameters エレメントは、1 つのパラメーターの名前と型を定義します。

表 14. 「Parameters」エレメントの特性

必須	はい
デフォルト	なし
内容	各 Parameters エレメントは、以下の情報を指定します。 • PropertyName: パラメーターとして受け渡すビジネス・オブジェクト属性の名前を指定します。 • Type: パラメーターのタイプ (以下のいずれかの値) を指定します。 – `IP`: 入力専用 – `OP`: 出力専用 – `IO`: 入出力 – `RS`: 結果セット
サポートされる双方向変換	いいえ
プロパティー・タイプ	String
使用上の注意	Oracle ストアード・プロシージャーの場合、結果セットは出力パラメーターとしてのみ返すことができます。この場合、1 つのパラメーターのタイプが、結果セットを示す `RS` でなければなりません。

戻り値

関数によって値が返されるため、プロシージャー呼び出しではなく関数呼び出しであることを示す値。

表 15. 「戻り値」エレメントの特性

必須	いいえ
デフォルト	なし
使用可能な値	`RS`、ビジネス・オブジェクト属性の名前、または子ビジネス・オブジェクトの名前を指定できます。
サポートされる双方向変換	いいえ
プロパティー・タイプ	String

表 15. 「戻り値」エレメントの特性 (続き)

使用上の注意	戻り値が RS である場合、戻り値は結果セットです。この結果セットは、このビジネス・オブジェクトに対応する複数カーディナリティー・コンテナーの作成に使用されます。戻り値が属性名である場合、値はビジネス・オブジェクトの特定の属性に割り当てられます。属性が別の子ビジネス・オブジェクトである場合、アダプターはエラーを返します。 テーブルまたはビューから生成されたビジネス・オブジェクトにストアード・プロシージャーを関連付けるときに、ストアード・プロシージャーが関数である場合は、そのストアード・プロシージャーから値が返されます。 1 つの ReturnValue アプリケーション固有情報の値が操作のアプリケーション固有情報に追加されます。このアプリケーション固有情報が存在する場合は、関数によって値が返されるため、関数呼び出しであってプロシージャー呼び出しではないことが示されます。 このアプリケーション固有情報の値がビジネス・オブジェクト属性名である場合は、戻り値がビジネス・オブジェクトの特定の属性に割り当てられます。 このアプリケーション固有情報の値が別の子ビジネス・オブジェクトである場合、アダプター・ランタイムはエラーを返します。 要約すると、戻り値が単純データ型である場合はウィザードによって 1 つのビジネス・オブジェクト属性を戻り値にバインドでき、このアプリケーション固有情報の値がそのビジネス・オブジェクト属性の名前に設定されます。しかし、戻り値が結果セットである場合、ウィザードはこのアプリケーション固有情報の値を RS に設定します。 **注:** Oracle データベースの場合、結果セットは戻り値としてではなく、出力パラメーターとして返さなければなりません。出力パラメーターのタイプは RS と設定され、結果セットを返すためにこのパラメーターが使用されることを示します。 **注:** 階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、トップレベルのビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーをトップレベルのビジネス・オブジェクトに関連付けても、各子ビジネス・オブジェクトに関連付けないと、そのトップレベルのビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

ストアード・プロシージャーの例

以下は、RtCustomer.xsd ファイルのカスタマー・ビジネス・オブジェクトの XML 定義を示した例で、Retrieve 操作の RetrieveSP および AfterRetrieveSP のストアード・プロシージャーの定義を表しています。アダプターは標準 SQL の代わりに

RT.RETR_CUST ストアード・プロシージャーを実行して、テーブル・ビジネス・オブジェクトを取得します。ビジネス・オブジェクトを取得した後、アダプターは RT.CUSTINFO ストアード・プロシージャーを実行します。

```
<jdbcasi:JDBCBusinessObjectTypeMetadata
   xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:TableName>RTASSER.CUSTOMER</jdbcasi:TableName
     <jdbcasi:Operation> <jdbc asi:Name>Retrieve</jdbcasi:Name>
       <jdbcasi:StoredProcedures>
         <jdbcasi:StoredProcedureType>AfterRetrieveSP</jdbcasi:StoredProcedureType>
         <jdbcasi:StoredProcedureName>RT.CUSTINFO</jdbcasi:StoredProcedureName>
           <jdbcasi:Parameters>
             <jdbcasi:Type>IP</jdbcasi:Type>
             <jdbcasi:PropertyName>pkey</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>lname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>ccode</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
         </jdbcasi:StoredProcedures>
         <jdbcasi:StoredProcedures>
         <jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
         <jdbcasi:StoredProcedureName>RT.RETR_CUST</jdbcasi:StoredProcedureName>
           <jdbcasi:Parameters>
             <jdbcasi:Type>IP</jdbcasi:Type>
             <jdbcasi:PropertyName>ccode</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>lname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
       </jdbcasi:StoredProcedures>
   </jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
```

ストアード関数の概要

データベースによっては、ストアード・プロシージャーの他にストアード関数をサポートしています。ストアード関数は、常に値を返す点を除き、ストアード・プロシージャーと同じです。アダプターはこれらを同じようにサポートします。

Oracle データベースでは、アダプターはユーザーが CREATE FUNCTION ステートメントで作成するストアード関数をサポートします。この型の関数はユーザー定義*関数* (UDF) と呼ばれることがありますが、この用語は、Java のストアード関数またはストアード・プロシージャーを指すことが多く、アダプターはこれらをサポートしません。

DB2 データベースでは、アダプターは値を戻すストアード・プロシージャーをサポートします。この機能を DB2 のユーザー定義関数と混同してはいけません。この用語はこの場合、SQL 言語の既存の組み込み関数への機能の拡張または追加のことを示しています。DB2 ユーザー定義関数は、アダプターで実行される操作、照会、およびバッチ SQL ステートメント用に作成した SQL 内で使用できますが、

CREATE FUNCTION ステートメントで作成したユーザー定義関数は、通常の場合、ビジネス・オブジェクトとしてアダプターに示されません。

関数呼び出しの構文を次に示します。

```
? = call FunctionName parameter_list
```

ストアード・プロシージャーの構文を次に示します。

```
call SPName parameter_list
```

ReturnValue ビジネス・オブジェクト・アプリケーション固有情報を使用して、戻り値を含む属性を指定します。ReturnValue について詳しくは、 301 ページの『ビジネス・オブジェクト・レベルのアプリケーション固有情報』を参照してください。

照会ビジネス・オブジェクトの概要

照会ビジネス・オブジェクトはユーザー定義の SELECT ステートメントをデータベースに対して実行して、ビジネス・オブジェクト内の一致するレコードを戻します。

外部サービス・ウィザードでは、データベースに対してユーザー定義 SELECT ステートメントを実行する照会ビジネス・オブジェクトを作成できます。 SELECT ステートメントを指定します。このとき、SELECT ステートメントの置換可能なパラメーターの代わりに ? (疑問符) を使用します。ウィザードに、各パラメーターのデータ型とサンプル値を指定できる領域が表示されます。ウィザードは SELECT ステートメントの結果を使用して照会ビジネス・オブジェクトを作成するため、サンプル値は、データベース内のデータと一致している必要があります。

ウィザードで照会の構成を保存する前に、照会を検証する必要があります。照会を検証するときには、サンプル値を使用した SELECT ステートメントの実行がウィザードによって行われます。結果セットを取得した後、ウィザードはメタデータを分析し、すべての列の列名および列の型を取得します。返された結果セットの列ごとに、ウィザードは対応する属性を照会ビジネス・オブジェクトに生成します。ウィザードは、WHERE 節のパラメーターごとに 1 つずつの属性を照会ビジネス・オブジェクト内に生成します。これらの属性は、実行時に結果を動的にフィルター操作するために使用されます。

例えば、以下の SELECT ステートメントを指定したと仮定します。

```
sSELECT * FROM CUSTOMER WHERE FNAME=? and AGE=?
```

Oracle データベースの場合、アダプターは、ビジネス・オブジェクトの照会結果で、配列、テーブル、構造体、ネストされた構造体などの複合データ型をサポートします。アダプターでは、バッチおよび照会のビジネス・オブジェクトにおいて、これらの複合型をパラメーターとしてサポートしていません。

この WHERE 節には 2 つのパラメーターがあります。最初のパラメーターのデータ型は **string** で、FNAME 列のデータ型と突き合わせます。2 番目のパラメーターのデータ型は **int** で、AGE 列と突き合わせます。データベースに、FNAME 列にストリング Mike、AGE 列に整数値 27 が含まれている顧客レコードがある場合、照会ビジネス・オブジェクトの構成時にこれらの値をサンプル値として指定できます。アダプターは、照会ビジネス・オブジェクト用に parameter1 および parameter2

という名前の 2 つの属性を生成します。実行時にこれらの属性に適切な値を指定することによって、照会結果を動的にフィルターに掛けることができます。

さらに、結果をフィルターに掛けるために実行中に動的に WHERE 節属性を設定することもできます。例えば、LNAME が `Mike` で始まる結果が戻されるよう、`WHERE LNAME LIKE 'Mike%'` のように WHERE 節を設定できます。

注: 動的に設定される WHERE 節でパラメーターを使用することはできません。

バッチ SQL ビジネス・オブジェクトの概要

バッチ SQL ビジネス・オブジェクトは 1 つ以上のユーザー定義の INSERT、UPDATE、および DELETE ステートメントを実行して、ステートメントの状況を戻します。

外部サービス・ウィザードでは、データベースに対してユーザー定義の INSERT、UPDATE、および DELETE ステートメントのセットを実行するバッチ SQL ビジネス・オブジェクトを作成できます。ステートメントを指定しますが、このとき、ステートメントの置換可能なパラメーターの代わりに ? (疑問符) を使用します。ウィザードに、各パラメーターのデータ型とサンプル値を指定できる領域が表示されます。ウィザードは構成を保存する前にビジネス・オブジェクトを検証するとき、サンプル値を使用します。

バッチ SQL ステートメントは UPDATE および DELETE ステートメントで動的 WHERE 節をサポートしません。アダプターは JOIN または SUBSELECT などの複合ステートメントを受け取りますが、アダプターではこれらのステートメントは構文解析されません。

バッチ SQL ビジネス・オブジェクトに単一の INSERT ステートメントが含まれる場合、自動生成 ID または挿入された行の識別値を検索できます。

ビジネス・オブジェクトの各 SQL ステートメントは状況値を戻しますが、これは Statement*N*Status という名前の属性に配置されます。例えば、ビジネス・オブジェクトの最初の SQL ステートメントの状況は Statement1Status に、2 番目のステートメントの状況は Statement2Status に、といったように入れられます。

単一の INSERT ステートメントが失敗すると、アダプターは例外を生成します。バッチ UPDATE のいずれかのステートメントが正しく実行されないと、JDBC ドライバーは java.sql.BatchUpdateException を生成します。この例外が生成されると、アダプターはトランザクションをロールバックして、どの SQL ステートメントもデータベースにコミットされないようにします。

外部サービス・ウィザード

IBM Integration Designer の外部サービス・ウィザードを使用して、データベースのオブジェクトをディスカバーし、バッチ SQL、照会、およびラッパー・ビジネス・オブジェクトを生成し、選択されているデータベース・オブジェクトからビジネス・オブジェクトを生成します。このウィザードでは、アダプターが Service Component Architecture (SCA) コンポーネントとして稼働できるようにするサービス成果物とモジュールも生成します。

Log and Trace Analyzer

アダプターは、Log and Trace Analyzer で表示できるログ・ファイルとトレース・ファイルを作成します。

Log and Trace Analyzer は、ログ・ファイルとトレース・ファイルをフィルタリングして、アダプターのメッセージとトレース情報を分離することができます。また、ログ・ビューアーの中で、アダプターのメッセージとトレース情報を強調表示することもできます。

フィルタリングおよび強調表示の際のアダプターのコンポーネント ID は、JDBCRA にアダプター ID プロパティーの値を付加した文字で構成されるストリングです。例えば、アダプター ID プロパティーが、001 に設定されている場合、コンポーネント ID は、JDBCRA001 となります。

同じアダプターの複数のインスタンスを実行する場合、アダプター ID プロパティーの最初の 7 文字は、必ずインスタンスごとに固有のものにし、ログおよびトレース情報を特定のアダプター・インスタンスに相互に関連付けられるようにしてください。アダプター ID プロパティーの最初の 7 文字を固有のものにすることにより、そのアダプターの複数インスタンスのコンポーネント ID も固有のものになり、アダプターの特定インスタンスにログおよびトレース情報を相互に関連付けることができるようになります。 例えば、WebSphere Adapter for JDBC の 2 つのインスタンスのアダプター ID プロパティーを 001 および 002 に設定するとします。これらのインスタンスのコンポーネント ID、JDBCRA001 および JDBCRA002 は、短いので固有性を保つことができ、別のアダプター・インスタンスとして区別することができます。しかし、もっと長いアダプター ID プロパティーのインスタンスの場合、互いを区別できなくなります。2 つのインスタンスのアダプター ID プロパティーを Instance01 と Instance02 に設定した場合、各アダプター・インスタンスのログおよびトレース情報を調べることはできなくなります。これは、両方のインスタンスのコンポーネント ID が JDBCRAInstanc に切り捨てられるためです。

Outbound 処理については、アダプター ID プロパティーは、リソース・アダプターおよび管理接続ファクトリー・プロパティー・グループの両方にあります。外部サービス・ウィザードを使用して Outbound 処理用にアダプターを構成後、アダプター ID プロパティーを更新する場合は、リソース・アダプター・プロパティーと管理接続ファクトリー・プロパティーの設定に矛盾がないことを必ず確認してください。そのようにすることで、ログおよびトレース・エントリーのマーキングが不整合になることを防ぐことができます。 Inbound 処理については、アダプター ID プロパティーは、リソース・アダプター・プロパティーのみに設定されますので、このような配慮は不要です。

アダプター ID プロパティーについて詳しくは、次を参照してください。 320 ページの『アダプ ター ID (AdapterID)』.

ビジネス・フォールト

アダプターは、予想される例外で Outbound サービス記述で宣言されている例外であるビジネス・フォールトか、インポートをサポートします。ビジネス・フォールトは、ビジネス・ルールの違反または制約違反が原因で、ビジネス・プロセスの予測可能なポイントに発生します。

IBM Business Process Manager または WebSphere Enterprise Service Bus は他のタイプのフォールトもサポートしていますが、アダプターはビジネス・フォールトのみを戻します。この資料ではビジネス・フォールトを単にフォールト と呼びます。すべての例外がフォールトになるわけではありません。フォールトは、アクション可能なエラー、すなわち、アプリケーションの終了を必要としないリカバリー・アクションが可能なエラーに対して生成されます。例えば、アダプターが受け取った Outbound 処理用のビジネス・オブジェクトに必須データが含まれていない場合や、アダプターで Outbound 処理中に特定のエラーが発生した場合に、アダプターはフォールトを戻します。

フォールト・ビジネス・オブジェクト

外部サービス・ウィザードにより、アダプターで生成可能な各フォールトに対して、ビジネス・オブジェクトが作成されます。さらに、このウィザードは WBIFault スーパーセット・ビジネス・オブジェクトを作成します。このオブジェクトには、図 24 に示されているように、message、errorCode、primaryKeySet の各属性など、すべてのフォールトに共通の情報が含まれています。

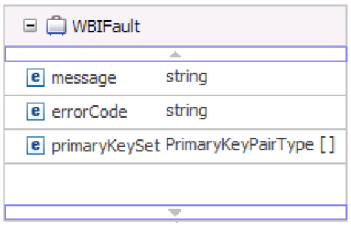


図 24. WBIFault ビジネス・オブジェクトの構造

カスタム・フォールト

外部サービス・ウィザードの「複合プロパティーの指定」ウィンドウでフォールト・パターンを事前定義することによって、サービス実行時例外をビジネス・フォールトとしてキャッチすることができます。カスタム・フォールトを定義すると、アダプターは実行時例外の代わりにカスタマイズされたフォールト・ビジネス・オブジェクトを返すようになるので、システム・ログ・エントリーが大量になるのを回避できます。例えば、データベースでロックされているレコードがある場合、「.*Record is locked.*」のような正規表現を追加することによってフォールト・パターン・ストリングを定義できます。実行時に、例外が発生すると、アダプターは例外メッセージとフォールト・パターン・ストリングを突き合わせます。例外メッセージが、使用可能な事前定義済みのフォールト・パターン・ストリングのいずれかと一致すると、アダプターはカスタム BusinessFault ビジネス・オブジェクトを戻します。このビジネス・オブジェクトは、事前定義されたフォールト・パターン・ストリングおよびフォールト名である faultPattern 属性および faultName 属性を含んでいます。

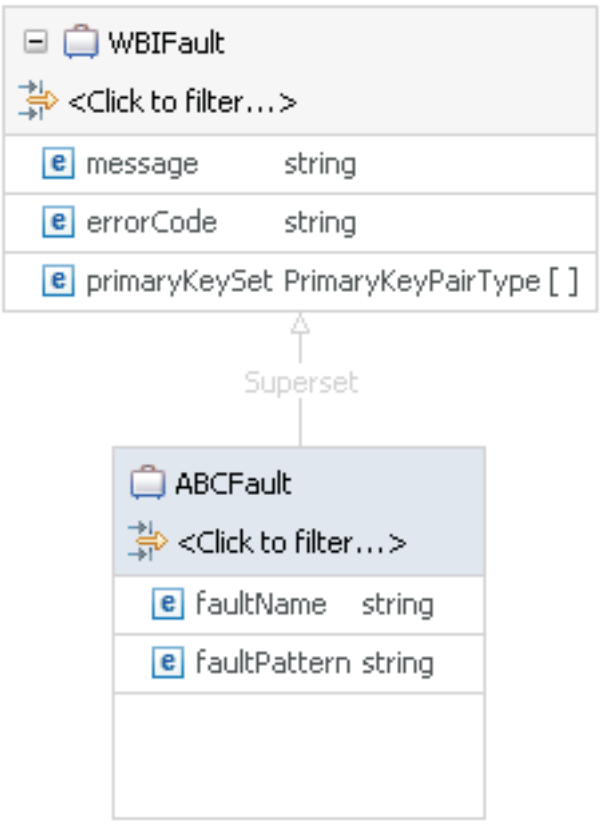


図 25. カスタム BusinessFault オブジェクトの構造

アダプター実装の計画

IBM WebSphere Adapter for JDBC を使用する前に、作業者に必要な経験と、アダプターが稼働するサーバー環境について理解しておきます。アダプターをサーバー環境にデプロイする上での考慮事項を理解すると共に、クラスター・サーバー環境の使用によってアダプターのパフォーマンスおよび可用性を向上させる方法を検討します。

始める前に

モジュールの構成とデプロイを開始する前に、ビジネス・インテグレーションの概念、Java Database Connectivity (JDBC)、ご使用の環境のデータベース製品、IBM Integration Designer および IBM Business Process Manager または WebSphere Enterprise Service Bus の機能について十分に理解しておく必要があります。

IBM WebSphere Adapter for JDBC を構成してデプロイするには、以下の概念、ツール、および作業に関する知識と経験が必要です。

- 構築するソリューションの業務要件。
- ご使用の環境の JDBC およびデータベース製品。これには、データ・アクセスの問題、トランザクション・モデル、および異種のリレーショナル・データベース、キュー、および Web サービス間の接続が含まれています。

- Service Component Architecture (SCA) プログラミング・モデルなどのビジネス・インテグレーションの概念およびモデル。
- IBM Business Process Manager または WebSphere Enterprise Service Bus などの統合ソリューションに使用する予定のサーバーの機能と要件。ホスト・サーバーの構成と管理の方法、および管理コンソールの使用によるプロパティー定義の設定と変更、接続ファクトリーの構成、イベントの管理の各方法を理解しておく必要があります。
- IBM Integration Designer によって提供されるツールおよび機能。これらのツールの使用によるモジュールの作成方法、コンポーネントの接続およびテスト方法、その他の統合作業の実行方法を理解しておく必要があります。

セキュリティー

アダプターは、Java 2 セキュリティーの J2C 認証データ項目 (認証別名) 機能を使用して、ユーザー名およびパスワードの安全な認証機能を提供します。 セキュリティー機能について詳しくは、IBM Business Process Manager または WebSphere Enterprise Service Bus の資料を参照してください。

ログ・ファイルとトレース・ファイルの中の機密ユーザー・データ保護のサポート

アダプターにより、ログ・ファイルおよびトレース・ファイル内の重要データまたは機密データを、許可なく表示できないように保護することができます。

アダプターのログ・ファイルおよびトレース・ファイルには、重要情報または機密情報が入っている可能性のある データベース からのデータが含まれる場合があります。このようなファイルは、重要データの表示許可を持たない人によって見られることがあります。例えば、サポート・スペシャリストはログ・ファイルおよびトレース・ファイルを使用して、問題のトラブルシューティングを行う必要があります。

そのような状況でデータを保護するために、アダプターでは、アダプターのログ・ファイルおよびトレース・ファイル内にあるユーザーの機密データを非表示にするかどうかを指定できます。このオプションは、外部サービス・ウィザードの中で選択したり、HideConfidentialTrace プロパティーを変更したりできます。このプロパティーが有効な場合、アダプターは、機密データを XXX で置き換えます。

このオプション・プロパティーについては、 324 ページの『管理接続ファクトリー・プロパティー』 を参照してください。

次のタイプの情報が基本的に機密データであるとみなされ、隠蔽されます。

- ビジネス・オブジェクトの内容
- イベント・レコードのオブジェクト・キーの内容
- ユーザー名、パスワード、環境、およびロール
- データベースへの接続に使用される URL

次のタイプの情報はユーザー・データであるとはみなされず、隠蔽されません。

- イベント・レコード・オブジェクト・キーの部分ではないイベント・レコードの内容。例えば、XID、イベント ID、ビジネス・オブジェクト名、およびイベント状況
- ビジネス・オブジェクト・スキーマ
- トランザクション ID
- 呼び出しシーケンス

ユーザー認証

アダプターでは、データベースへの接続に必要なユーザー名およびパスワードを指定する方法がいくつかサポートされています。それぞれの方法の特徴および制限を理解した上で、ご使用のアプリケーションにとって適切なセキュリティー・レベルであり、かつ都合のよい方法を選択してください。

アダプターをアプリケーションに統合するには、以下の場合にユーザー名およびパスワードが必要になります。

- ユーザーがアダプターでアクセスできるオブジェクトおよびサービスに関する情報を抽出、すなわちディスカバー するために外部サービス・ウィザードがデータベースに接続するとき。
- IBM Business Process Manager または WebSphere Enterprise Service Bus での実行時に、アダプターが Outbound 要求および Inbound イベントを処理するためにデータベースに接続するとき。

ウィザードでの認証

外部サービス・ウィザードはディスカバリー処理のために接続情報を要求し、その後、それを、実行時に使用する接続情報を指定するアダプター・プロパティーのデフォルト値として再利用します。ウィザードの実行中に使用するユーザー名およびパスワードは、アプリケーションをサーバーにデプロイするときとは別のものを使用できます。別の データベースに接続することもできます。ただし、2 つのデータベースのスキーマ名が同じである必要があります。例えば、WebSphere Adapter for JDBC を使用するアプリケーションの開発および統合中は、実動データベースを使用しないことがあります。テスト・データベースを使用し、同じデータ・フォーマットで、より少ない数の模擬レコードを使用することにより、実動データベースのパフォーマンスに影響を与えることなく、また顧客データのプライバシー要件に起因する制限が生じることなく、アプリケーションを開発および統合できます。

ウィザードは、ディスカバリー・プロセス用に指定されたユーザー名およびパスワードをディスカバリー・プロセスでのみ使用します。これらは、ウィザードの完了後はアクセス不能になります。

実行時の認証

実行時、アダプターは、データベースに接続するためにユーザー名およびパスワードを提供する必要があります。ユーザー介入なしに接続するためには、アダプターは保存されているユーザー情報のコピーにアクセスしなければなりません。サーバー環境では、ユーザー情報の保存方法はいくつかあります。外部サービス・ウィザードでは、次のいずれかの方法を使用してユーザー情報を取得するようにアダプターを構成できます。

- アダプター・プロパティー
- データ・ソース
- J2C 認証別名

アダプター・プロパティーへのユーザー名およびパスワードの保存は、実行時にこの情報を提供するための直接的な方法です。外部サービス・ウィザードを使用してモジュールを構成するときに、このユーザー名およびパスワードを指定します。ユーザー名とパスワードを直接指定する方法は最も簡単なように見えますが、この方法には重要な制限があります。アダプター・プロパティーは暗号化されません。パスワードは、サーバー上で他のユーザーがアクセスできるフィールドに平文で格納されます。さらに、パスワードが変更された場合は、そのデータベースにアクセスするすべてのアダプター・インスタンスで、パスワードを更新しなければなりません。これは、アプリケーション EAR ファイルに組み込まれているアダプターだけでなく、サーバーに個別にインストールされたアダプターも該当します。

データ・ソースを使用して、他のアプリケーション用に既に確立された接続を使用します。例えば、複数のアプリケーションが同じユーザー名およびパスワードを使用して同じデータベースにアクセスする場合は、同じデータ・ソースを使用してそれらのアプリケーションをデプロイできます。ユーザー名およびパスワードを知るユーザーを、そのデータ・ソースにアプリケーションをデプロイする最初のユーザー、またはデータ・ソースを個別に定義する最初のユーザーのみに限定できます。

Java 2 セキュリティーの Java 認証・承認サービス (JAAS、Authentication and Authorization Service) フィーチャーで作成された J2C 認証データ項目、すなわち認証別名を使用する方法は、堅固でセキュアなアプリケーション・デプロイメント方法です。 管理者は、システムにアクセスする必要がある 1 つ以上のアプリケーションで使用される認証別名を作成します。ユーザー名およびパスワードを知るユーザーを、その管理者のみに限定できます。管理者は、変更が必要な場合は単一の場所でパスワードを変更できます。

デプロイメント・オプション

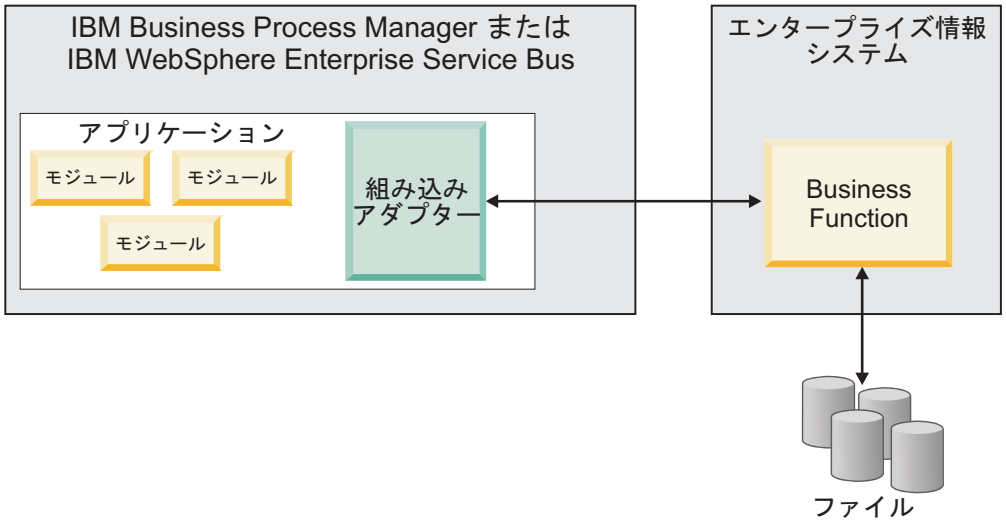
アダプターをデプロイする方法は、2 とおりあります。デプロイされたアプリケーションの一部としてアダプターを組み込むか、アダプターをスタンドアロン RAR ファイルとしてデプロイできます。ご利用の環境の要件によって、選択するデプロイメント・オプションのタイプが異なります。

デプロイメント・オプションについて以下で説明します。

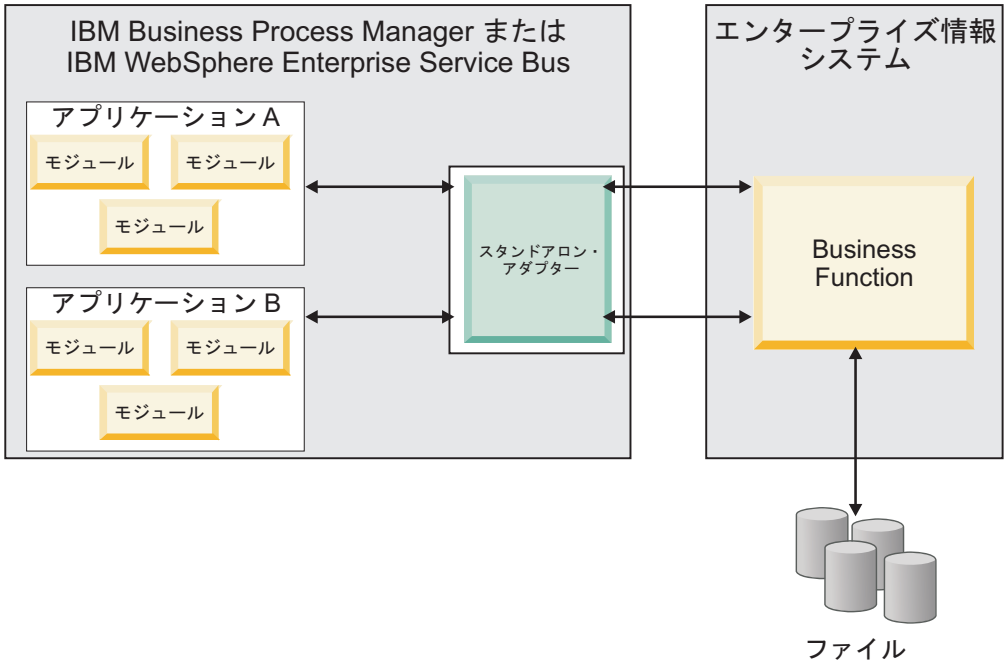
- **単一アプリケーションが使用するモジュールで (With module for use by single application)**: アダプター・ファイルをモジュール内に組み込むと、モジュールをすべてのアプリケーション・サーバーにデプロイすることができます。組み込みアダプターを使用するのは、組み込みアダプターを使用するモジュールが 1 つある場合か、複数のモジュールでバージョンの異なるアダプターを実行する必要がある場合です。組み込みアダプターを使用すると、他のモジュールのアダプター・バージョンを変更することで、それらのモジュールを不安定にするリスクを生じることなく、1 つのモジュール内でアダプターをアップグレードできます。
- **複数アプリケーションが使用するサーバー上 (On server for use by multiple applications)**: モジュール内にアダプター・ファイルを組み込まない場合は、このモジュールを実行するアプリケーション・サーバーごとに、アダプター・ファイ

ルをスタンドアロン・アダプターとしてインストールする必要があります。複数のモジュールが同じバージョンのアダプターを使用可能で、アダプターを中央の場所で管理する場合は、スタンドアロン・アダプターを使用します。スタンドアロン・アダプターの場合も、複数のモジュールに対して単一のアダプター・インスタンスを実行することにより、必要なリソースが軽減されます。

エンタープライズ・アーカイブ (EAR) ファイル内には、組み込みアダプターがバンドルされています。この組み込みアダプターは、一緒にパッケージされ、デプロイされたアプリケーションでのみ使用することができます。



スタンドアロン・アダプターを表すのは、スタンドアロンのリソース・アダプター・アーカイブ (RAR) ファイルです。これは、デプロイされた後、サーバー・インスタンス内のすべてのデプロイ済みアプリケーションから使用することができます。



ご使用のアプリケーションのプロジェクトを IBM Integration Designer を使用して作成する場合は、アダプターのパッケージ方法 (EAR ファイルによるバンドル、またはスタンドアロン RAR ファイル) を選択できます。この選択に応じて、ランタイム環境におけるアダプターの使用方法や、管理コンソールにおけるアダプターのプロパティーの表示方法が異なります。

アダプターをアプリケーションに組み込む方法と、スタンドアロン・モジュールとしてデプロイする方法のどちらを選択するかは、アダプターの管理の仕方によって決まります。アダプターの 1 つのコピーのみを保持して、アダプターのアップグレード時に複数のアプリケーションが中断してもかまわない場合は、アダプターをスタンドアロン・モジュールとしてデプロイすることが多くなります。

複数のバージョンを稼働させる計画があり、アダプターのアップグレード時に起こりうる中断を避けたい場合は、アダプターをアプリケーションに組み込むことが多くなります。アダプターをアプリケーションに組み込む場合、アダプターのバージョンをアプリケーションのバージョンに関連付けて、単一のモジュールとして管理することができます。

アダプターのアプリケーションへの組み込みに関する考慮事項

アダプターをアプリケーションに組み込む計画がある場合は、以下の点を考慮してください。

- 組み込みアダプターには、クラス・ローダーの独立性があります。

  クラス・ローダーは、アプリケーションのパッケージ化、およびランタイム環境にデプロイされたパッケージ済みアプリケーションの動作に影響を与えます。 *クラス・ローダーの独立性* とは、アダプターが、他のアプリケーションまたはモジュールからクラスをロードできないということを意味します。 クラス・ローダーの独立性により、異なるアプリケーション内の類似する名前を持つ 2 つのクラスは互いに干渉しなくなります。
- アダプターが組み込まれた各アプリケーションを、別々に管理する必要があります。

スタンドアロン・アダプターを使用する際の考慮事項

スタンドアロン・アダプターを使用する計画がある場合は、以下の点を考慮してください。

- スタンドアロン・アダプターには、クラス・ローダーの独立性がありません。

  スタンドアロン・アダプターにはクラス・ローダーの独立性がないため、指定された任意の Java 成果物の 1 つのバージョンのみが実行され、その成果物のバージョンおよびシーケンスは確定されません。例えば、スタンドアロン・アダプターを使用する場合は、*1* つの リソース・アダプター・バージョン、*1* つの アダプター・ファウンデーション・クラス (AFC) バージョン、または *1* つの サード・パーティー JAR バージョンのみが存在します。 スタンドアロン・アダプターとしてデプロイされたアダプターはすべて、単一の AFC バージョンを共有し、1 つのアダプターのすべてのインスタンスは同じコードのバージョンを共有します。1 つのサード・パーティー・ライブラリーを使用するアダプター・インスタンスはすべて、そのライブラリーを共有しなければなりません。

- これらの共有成果物のいずれかを更新する場合、その成果物を使用するすべてのアプリケーションが影響を受けることになります。

  例えば、サーバー・バージョン X で動作しているアダプターを使用しているときに、クライアント・アプリケーションのバージョンをバージョン Y に更新すると、元のアプリケーションが動作しなくなることがあります。
- アダプター・ファウンデーション・クラス (AFC) には前のバージョンとの互換性がありますが、スタンドアロン・フォーマットでデプロイされる各 RAR ファイルには、最新バージョンの AFC を入れておく必要があります。

  スタンドアロン・アダプターのクラスパス内に JAR ファイルの複数のコピーがある場合、使用される JAR ファイルはランダムになります。このため、すべての JAR ファイルを最新バージョンにしておく必要があります。

注:

CWYBS_AdapterFoundation.jar のバージョンが異なるアダプターを複数インストールしており、実行時に古いバージョンの CWYBS_AdapterFoundation.jar がロードされると、バージョンの競合が発生し、アダプターにより ResourceAdapterInternalException エラー・メッセージが返されます。例えば、Oracle E-Business Suite アダプター バージョン 7.0.0.3 と WebSphere Adapter for JDBC バージョン 7.5 をインストールした場合、次のエラー・メッセージが表示されます。IBM WebSphere Adapter for JDBC には、バージョン 7.0.0.3 のファイル /C:/IBM/WebSphere/ProcServer7/profiles/ProcSrv01/installedConnectors/CWYOE_OracleEBS.rar/CWYBS_AdapterFoundation.jar がロードされています。しかし、要求されているこの jar のベース・レベルはバージョン 7.5 です。CWYBS_AdapterFoundation.jar のバージョンが異なるアダプターを複数インストールしている場合、バージョンの競合が発生し、アダプターにより ResourceAdapterInternalException メッセージが返されます。この競合を解決するには、すべてのアダプターを同じバージョン・レベルにマイグレーションする必要があります。さらに支援が必要な場合は、WebSphere Adapters サポートにアクセスしてください。

WebSphere Adapter 7.5 を他のバージョンとデプロイする際の考慮事項

クライアント/サーバー・コミュニケーションが不要な組み込みアダプターを使用する必要がある場合、サーバー接続を必要とするスタンドアロン・アダプターを使用する必要がある場合、あるいはさまざまなアダプター接続を混用する必要がある場合があります。

以下のシナリオで、それぞれの場合において AFC バージョンの競合が検出される動作を示します。

スタンドアロン・アダプターをデプロイする場合

1. IBM Business Process Manager 管理コンソールを使用して、WebSphere Adapter for Flat Files バージョン 7.0.1.0 をインストールします。
2. 管理コンソールを使用して、WebSphere Adapter for SAP Software バージョン 7.5.0.0 をインストールします。

3. ALE パススルー Inbound 操作の活動化仕様を作成します。
4. スタンドアロン ALE パススルー Inbound 操作のアプリケーションを IBM Integration Designer で作成します。
5. 管理コンソールを使用して、アプリケーションをインストールし、開始します。
6. エラーを検証します。

注: IBM Business Process Manager のログ/トレース・エリアに、 AFC バージョンの競合を示すエラー・メッセージが生成されます。

組み込みアダプターをデプロイする場合

1. RAR ファイルを使用して、WebSphere Adapter for FTP バージョン 7.0.1.0 のビルドをインポートします。
2. FTP Inbound EMD 操作を作成します。
3. RAR ファイルを使用して、WebSphere Adapter for Oracle E-Business Suite バージョン 7.5.0.0 のビルドをインポートします。
4. FTP Inbound EMD 操作を作成したモジュールと同じモジュールに、Oracle Inbound EMD 操作を作成します。
5. モジュールを IBM Business Process Manager にデプロイします。
6. トレースを確認します。

ステップ 5 で、デプロイメントが失敗するはずです。ステップ 6 で、 AFC バージョンの競合による内部エラー・メッセージを受け取ります。

注: 2 つのアダプターによって生成されたビジネス・オブジェクト間で名前が競合しないようにするために、成果物を別々のフォルダーに生成する必要があるかもしれません。

スタンドアロン・アダプターと組み込みアダプターを組み合わせてデプロイする場合

1. IBM Business Process Manager 管理コンソールを使用して、WebSphere Adapter for JDBC バージョン 7.0.1.0 をインストールします。
2. JDBC Inbound 操作の活動化仕様を作成します。
3. スタンドアロン・アダプター・デプロイメント用の JDBC Inbound 操作のアプリケーションを IBM Integration Designer で作成します。
4. JDBC Inbound アプリケーションをデプロイして、Inbound イベントをトリガーします。
5. WebSphere Adapter for SAP Software バージョン 7.5.0.0 Inbound 組み込みアダプター・デプロイメント用のアプリケーションを IBM Integration Designer で作成します。
6. SAP Inbound アプリケーションをデプロイして、Inbound イベントをトリガーします。

注: スタンドアロン・デプロイメントおよび組み込みデプロイメントにそれぞれ異なるクラス・ローダーを使用することにより、AFC バージョンの競合を解決することができます。この方法を使用すると、マイグレーション・プロセスは異なる CWYBS_AdapterFoundation.jar ファイルを処理するため、互いに競合することはあり

ません。JDBC Inbound アプリケーションと SAP Inbound アプリケーションを正常に開始することができ、例外が発生することなく Inbound イベントを処理できます。

さらに支援が必要な場合は、 http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Familyにアクセスしてください。

クラスター環境での WebSphere Adapters

モジュールをクラスター・サーバー環境にデプロイすることで、アダプターのパフォーマンスおよび可用性を向上させることができます。クラスターとは、ワークロードの平衡を取り、高可用性とスケーラビリティーを提供するために、一緒に管理されるサーバー・グループのことです。

デプロイしたモジュールはクラスター内のすべてのサーバーで複製されます。これは、モジュールをデプロイするのに使用したアダプターがスタンドアロンであっても組み込みであっても関係ありません。以下の IBM 製品は、クラスター環境で WebSphere Adapters をサポートします。

- IBM Business Process Manager または WebSphere Enterprise Service Bus
- WebSphere Application Server Network Deployment
- WebSphere Extended Deployment

サーバー・クラスターをセットアップするときには、デプロイメント・マネージャー・プロファイルを作成してください。デプロイメント・マネージャーのサブコンポーネントである HAManager が、アダプター・インスタンスを活動化するよう Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) コンテナーに通知します。クラスター環境の作成について詳しくは、リンク http://publib.boulder.ibm.com/infocenter/wasinfo/beta/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm_cluster_v61.html を参照してください。

必要に応じて、WebSphere Extended Deployment を使用して、クラスター環境内のアダプター・インスタンスのパフォーマンスを向上させることができます。WebSphere Extended Deployment は、静的ワークロード・マネージャーではなく動的ワークロード・マネージャーのインスタンスを使用して、WebSphere Application Server Network Deployment の機能を拡張します。動的ワークロード・マネージャー・インスタンスは、要求の負荷を動的に平衡化することによって、クラスター内のアダプター・インスタンスのパフォーマンスを最適化することができます。つまり、負荷の変動に応じてアプリケーション・サーバー・インスタンスを自動的に停止したり始動したりできるため、能力や構成が異なるシステムが負荷変動に一様に対処できるようになります。 WebSphere Extended Deployment の利点について詳しくは、http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r1m1/index.jspを参照してください。

クラスター環境では、アダプター・インスタンスは、Inbound 処理および Outbound 処理の両方を処理することができます。

Inbound 処理の高可用性

Inbound 処理は、データベースのデータを更新した結果、起動するイベントに基づいています。WebSphere Adapter for JDBC は、イベント・テーブルをポーリングすることで更新を検出するよう構成されます。 その後、アダプターはイベントをそのエンドポイントにパブリッシュします。

モジュールをクラスターにデプロイすると、 Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) コンテナーにより、enableHASupport リソース・アダプター・プロパティーが検査されます。enableHASupport プロパティーの値が真である場合 (デフォルトの設定)、すべてのアダプター・インスタンスはポリシー N のうちの 1 つを持つ HAManager に登録されます。 このポリシーは、アダプター・インスタンスのうちの 1 つのみがイベントのポーリングを開始することを意味します。 クラスター内のその他のアダプター・インスタンスが開始していても、それらのインスタンスは、アクティブなアダプター・インスタンスがイベントの処理を完了するまで、アクティブ・イベントに関して休止のままとなります。ポーリング・スレッドが開始しているサーバーが何らかの理由でシャットダウンした場合は、バックアップ・サーバーのいずれかで稼働しているアダプター・インスタンスが活動状態になります。

注: アダプターのアクティブ/パッシブ構成モードで、`enableHASupport` プロパティーが `True` に設定されている場合でも、パッシブ・アダプター・インスタンスのエンドポイント・アプリケーションもイベント/メッセージを listen します。これは、JMS 活動化仕様の `alwaysactivateAllMDBs` プロパティーが `True` に設定されているからです。パッシブ・アダプター・インスタンスのエンドポイント・アプリケーションで、イベントの listen を止めるようにするには、`alwaysactivateAllMDBs` プロパティー値を `False` に設定する必要があります。詳しくは、 267 ページの『enableHASupport が True に設定されていると、パッシブ・アダプター・インスタンスの エンドポイント・アプリケーションがイベントを listen する』トピックを参照してください。

WebSphere 高可用性環境での WebSphere Adapter の使用について詳しくは、WebSphere 高可用性環境での WebSphere Adapter の使用を参照してください。

高可用性 Active-Active が有効になっていると、つまり enableHASupport プロパティーの値が false に設定されていると、すべてのアダプター・インスタンスが Inbound クラスターでのイベントをポーリングし、アダプターは Active-Active 構成で作動します。アクティブ構成モードでは、高可用性クラスター内で WebSphere Adapter for JDBC のインスタンスを複数アクティブにすることができます。各アダプター・インスタンスが別々のイベントを並行して処理します。1 つのクラスター・セットアップ内で複数のアダプター・インスタンスがアクティブにポーリングを行う場合、それらはロード・バランサーとして機能します。クラスター内のアダプター・インスタンスのうちの 1 つで障害が起こった場合、クラスター内の他のアクティブ・インスタンスがイベントを処理します。

高可用性 Active-Active が有効にされると、JDBC アダプターは、新しいテーブル構成に基づいてイベントを処理します。イベント・テーブル構成が正しくない場合、アダプターは始動中に例外をスローします。アダプターは Inbound 処理の場合にのみ高可用性 Active-Active をサポートするので、Inbound 構成のアダプターは実行時

に例外をスローします。テーブル構造の変更または新規作成について詳しくは、105 ページの『イベント・ストアの作成』を参照してください。

注: クラスター環境では、アダプターが高可用性 Active-Active 構成で動作する場合、高可用性とロード・バランシングのサポートの両方を提供します。この機能は、ハイパフォーマンスを必要とする実稼働環境で有用です。

高可用性 Active-Active 構成では、WebSphere Adapter for JDBC は、 1 つのイベントが複数のアダプター・インスタンスによって処理されることのないよう保証します。その結果として、各アダプター・インスタンスがそれぞれ固有のイベントをポーリングし、そのイベントを重複なしでエンドポイントに送達するようになります。ただし、高可用性 Active-Active が有効になっていると、イベントの順序付けは保証されません。いずれかのアダプター・インスタンスで障害が起こると、フェッチされたイベントは指定の時間内にクリーンアップされ、すべての未処理イベントは新規イベントとして処理されます。

高可用性 Active-Active 構成におけるデータベース・サポート

現在のところ、アダプターは、高可用性 Active-Active 構成で稼働しているとき、次のデータベースをサポートします。

- IBM DB2
- Oracle
- Microsoft SQL Server

Outbound 処理の高可用性

クラスター環境では、Outbound 処理要求の実行に、複数のアダプター・インスタンスが使用可能です。そのため、Outbound 要求について WebSphere Adapter for JDBC と対話するアプリケーションが、ご使用の環境に複数存在する場合は、クラスター環境にモジュールをデプロイすることにより、パフォーマンスが向上することがあります。クラスター環境では、複数の Outbound 要求が同じレコードを処理しようとしない限り、複数の Outbound 要求を同時に処理することができます。

複数の Outbound 要求が、顧客の住所などの同じレコードを処理しようとした場合、WebSphere Application Server Network Deployment のワークロード管理機能により、その要求は、受信された順に使用可能なアダプター・インスタンスの間で分配されます。このため、クラスター環境では、この種の Outbound 要求は、単一サーバー環境内と同じように処理されます。すなわち、1 つのアダプター・インスタンスが一度に処理するのは、1 つの Outbound 要求のみです。ワークロード管理について詳しくは、リンク http://publib.boulder.ibm.com/infocenter/wasinfo/beta/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm.html を参照してください。

準備済みステートメントのキャッシュのサポート

IBM WebSphere Adapter for JDBC では、サーバーでの準備済みステートメントのキャッシュがサポートされています。これにより、Outbound/Inbound 操作または操作のバッチの実行にかかる時間が短縮されます。

アダプターは*準備済みステートメント* を使用します。これは、1 回コンパイルされたが、繰り返し実行可能な SQL QUERY ステートメントが含まれている Java オブジェクトです。サーバーは、準備済みステートメントをキャッシュに格納することで、このステートメントの処理を最適化します。アダプターで準備済みステートメントのキャッシュを使用する場合は、管理コンソールでデータ・ソースを定義し、定義したデータ・ソースのキャッシュを有効にします。次に、以下のいずれかの方法で、このデータ・ソースを使用するようにアダプターを構成します。

- アダプターを初めて構成するときに、外部サービス・ウィザードを使用し、データ・ソースの JNDI 名を使用するように構成する
- 管理コンソールを使用して、Inbound 処理に対する DatasourceJNDIName プロパティーを設定する
- 管理コンソールを使用して、Outbound 処理に対する poolDataSourceJNDIName プロパティーまたは XADataSourceJNDIName プロパティーを設定する

WebSphere Adapter for JDBC のバージョン 7.5 へのマイグレーション

WebSphere Adapter for JDBC のバージョン 7.5 へのマイグレーションを行うことにより、アダプターの前のバージョンから自動的にアップグレードします。さらに、前のバージョンのアダプターを組み込んだアプリケーションをマイグレーションできるため、アプリケーションで、バージョン 7.5 が備えているフィーチャーや機能を利用できます。

マイグレーションに関する考慮事項

WebSphere Adapter for JDBC バージョン 7.5 には、既存のアダプター・アプリケーションに影響を与える可能性のあるフィーチャーおよび更新がいくつか含まれている場合があります。WebSphere Adapter for JDBC を使用するアプリケーションのマイグレーションを行う前に、既存のアプリケーションに影響を与える可能性のある要因について考慮する必要があります。

旧バージョンとの互換性

WebSphere Adapter for JDBC バージョン 7.5 は、バージョン 6.1x、バージョン 6.2x、およびバージョン 7.0 のアダプターを使用して作成されるカスタム・ビジネス・オブジェクト (XSD ファイル) およびデータ・バインディングと完全に互換性があり、既存のビジネス・オブジェクトおよびデータ・バインディングは最新バージョンのアダプターでも正常に動作します。

バージョン 7.5 の WebSphere Adapter for JDBC はバージョン 6.1x、バージョン 6.2x、およびバージョン 7.0 と完全に互換性があるため、旧バージョンの WebSphere Adapter for JDBC を使用していたアプリケーションはすべて、バージョン 7.5 にアップグレードしても変更なしで稼働します。ただし、バージョン 7.5 のアダプターのフィーチャーおよび機能をアプリケーションで使用したい場合は、アダプターのアップグレードに加えて、成果物のマイグレーションを実行してください。

マイグレーション・ウィザードは、バージョン 6.1.x、バージョン 6.2.x、またはバージョン 7.0 のアダプターをバージョン 7.5 に置換 (アップグレード) し、バージョン 7.5 のフィーチャーおよび機能をアプリケーションで使用できるようにします。

注: マイグレーション・ウィザードは、バージョン 7.5 のアダプターで動作するマッパーやメディエーターなどのコンポーネントを作成したり、既存のコンポーネントを変更したりすることはありません。バージョン 7.0 またはそれより前のアダプターが組み込まれたアプリケーションがあり、バージョン 7.5 にアップグレードしようとしていて、バージョン 7.5 のフィーチャーおよび機能をそれらのアプリケーションで利用したい場合は、アプリケーションの変更が必要になる場合があります。

モジュール内で成果物のバージョンが整合しない場合は、モジュール全体がマイグレーション不可としてマークが付けられるため、選択できません。バージョンの不整合は、プロジェクトが破損している可能性があることを示すものであるため、ワークスペース・ログに記録されます。

IBM Integration Designer バージョン 7.5 のアダプター・マイグレーション・ウィザードは、バージョン 6.1x、バージョン 6.2x、およびバージョン 7.0 からバージョン 7.5 へのアダプターのマイグレーションのみをサポートします。古いバージョンからバージョン 7.5 より前のバージョンへのアダプターのマイグレーションはサポートされていません。

アップグレードかアップグレード後にマイグレーションかの決定

マイグレーション・ウィザードのデフォルト処理では、アダプターのアップグレードを行い、アプリケーションで バージョン 7.5 のアダプターのフィーチャーと機能を使用できるように、アプリケーション成果物をマイグレーションします。プロジェクトを選択することによってアダプターのアップグレードを選択すると、ウィザードは関連する成果物をマイグレーション用に自動的に選択します。

アダプターを 6.1.x、バージョン 6.2.x、およびバージョン 7.0 からバージョン 7.5 にアップグレードするが、アダプター成果物はマイグレーションしないと決定した場合、アダプター成果物をマイグレーション・ウィザードの適切な領域から選択解除することでそれを実現できます。

アダプター成果物が何も選択されていない状態でマイグレーション・ウィザードを実行すると、アダプターがインストールされ、アップグレードされます。成果物はマイグレーションされないため、ご使用のアプリケーションでは バージョン 7.5 のアダプターで提供される機能は利用できません。

プロジェクト内で参照される複数のアダプターのマイグレーション

モジュールに 1 つ以上のコネクター・プロジェクトが含まれていて、そのそれぞれが別々のアダプターを参照している場合 (例えば、JDBC および SAP の各アダプターを参照する複数のコネクター・プロジェクトが含まれているモジュール・プロジェクトなど)、マイグレーション・ウィザードはそれぞれのアダプターに属する成果物を識別し、その他のアダプターの成果物に悪影響を与えることなく、それらの成果物をマイグレーションします。

モジュール・プロジェクトを選択してマイグレーション・ウィザードを起動した場合、以下のようになります。

- 「**ソース・コネクター**」フィールドには、選択されたモジュール・プロジェクトとともにコネクター・プロジェクトがリストされます。
- 「**依存関係のある成果物プロジェクト**」領域には、選択されたモジュール・プロジェクトのみがリストされます。

コネクター・プロジェクトを選択して、マイグレーション・ウィザードを起動する場合:

- 「**ソース・コネクター**」フィールドには、選択されたコネクター・プロジェクトのみがリストされます。
- 「**依存関係のある成果物プロジェクト**」領域には、モジュール・プロジェクトなど、選択されたコネクター・プロジェクトを参照するすべてのプロジェクトがリストされます。

テスト環境でのマイグレーション・ウィザードの実行

アダプターのマイグレーションでは、WebSphere Adapter for JDBC のバージョン 7.5 を使用するアプリケーションを変更しなければならない場合があるため、アプリケーションを実稼働環境にデプロイする前に、まず開発環境でマイグレーションを実行して、アプリケーションをテストする必要があります。

マイグレーション・ウィザードは、開発環境に完全に統合されています。

非推奨の機能

非推奨の機能とは、サポートされていても推奨されてはおらず、廃止される可能性がある機能です。従来のバージョンの WebSphere Adapter for JDBC の機能のうち、バージョン 7.5 で非推奨になったもので、アプリケーションの変更が必要になる可能性があるのは、次のとおりです。

- Outbound 処理での DataSourceJNDIName プロパティーのサポート

  このプロパティーはバージョン 7.0 では非推奨であり、代わりに 2 つの新規プロパティー 337 ページの『XA DataSource JNDI 名 (XADataSourceJNDIName)』 および 338 ページの『接続プール・データ・ソースの JNDI 名 (PoolDataSourceJNDIName)』 が提供されています。前のバージョンの成果物が DataSourceJNDIName プロパティーで構成されている場合、マイグレーション・プロセス時に、新規プロパティー 337 ページの『XA DataSource JNDI 名 (XADataSourceJNDIName)』 が DataSourceJNDIName プロパティーと同じ値に設定されます。この非推奨プロパティーは、実行時にはアダプターによって引き続きサポートされるため、アダプター バージョン 7.5 を円滑に使用することができ、このアダプターはアセンブリー・エディターに表示されます。ただし、このプロパティーは編集できません。

マイグレーションの実行

アダプター・マイグレーション・ウィザードを使用して、プロジェクトまたは EAR ファイルをバージョン 7.5 にマイグレーションできます。 ツールが終了したらマイグレーションは完了するため、プロジェクトで作業したり、モジュールをデプロイしたりできます。

始める前に

『*マイグレーションに関する考慮事項*』の情報を見直します。

このタスクについて

IBM Integration Designer でマイグレーションを実行するには、以下のステップを完了してください。

注: マイグレーションが完了すると、そのモジュールは以前のバージョンの IBM Business Process Manager または WebSphere Enterprise Service Bus ランタイムおよび IBM Integration Designer とは互換性がなくなります。

以下の手順では、IBM Integration Designer の Java EE パースペクティブでコネクター・プロジェクトのメニューからアダプター・マイグレーション・ウィザードを実行する方法について説明します。

手順

1. 既存のプロジェクトの場合は PI (プロジェクト交換) ファイルをワークスペースにインポートします。

   注: RAR の内容を変更したり、コネクター・プロジェクトの外部でアダプターの JAR ファイルをコピーしたりしないでください。
2. 以前のバージョンの IBM Integration Designer で作成したプロジェクトがある場合は、ワークスペース・マイグレーション・ウィザードが自動的に開始され、マイグレーション対象のプロジェクトが選択されます。ウィザードに従って、ワークスペースのマイグレーションを完了します。詳しくは、http://bidoc.torolab.ibm.com:7500/help/index.jsp?topic=/com.ibm.wbpm.wid.imuc.doc/topics/tmigsrcart.htmlを参照してください。
3. Java EE パースペクティブに切り替えます。
4. モジュールを右クリックして、「**コネクター・プロジェクトのマイグレーション**」を選択します。例えば、アダプター RAR モジュールです。

   また、以下の方法でアダプター・マイグレーション・ウィザードを起動することもできます。
   - Java EE パースペクティブで、プロジェクトを右クリックし、「**アダプター成果物のマイグレーション**」を選択します。
   - 問題ビューで、マイグレーション固有のメッセージを右クリックし、「**クイック・フィックス**」を選択して問題を解消します。
5. 「プロジェクトの選択」ウィンドウで、以下の手順を実行します。
   a. 「**ソース・コネクター**」フィールドに、マイグレーションするコネクター・プロジェクトの名前が表示されます。モジュール・プロジェクトをマイグレーションする場合、このフィールドにはモジュール・プロジェクト内のすべてのコネクター・プロジェクトがリストされます。リストからソース・プロジェクトを選択します。詳しくは、88 ページの『プロジェクト内で参照される複数のアダプターのマイグレーション』を参照してください。
   b. 「**ターゲット・コネクター**」フィールドに、マイグレーションするコネクターの名前が表示されます。複数のバージョンのアダプターを処理する場合、

このリストには互換性のあるすべてのコネクターの名前が表示されます。マイグレーションするコネクターを選択します。

c. 「**ターゲットのバージョン**」フィールドに、前のステップで選択したターゲット・コネクターに対応するバージョンが表示されます。

d. 「**依存関係のある成果物プロジェクト**」の領域には、マイグレーションされるアダプター成果物がリストされます。モジュール・プロジェクトをマイグレーションする場合、この領域には選択されたモジュール・プロジェクトのみがリストされます。モジュール・プロジェクト内のコネクター・プロジェクトをマイグレーションする場合、この領域には、モジュール・プロジェクトを含めて、選択されたコネクター・プロジェクトを参照するすべてのプロジェクトがリストされます。 デフォルトでは、依存関係のある成果物プロジェクトがすべて選択されています。依存関係のある成果物プロジェクトを選択しないと、そのプロジェクトはマイグレーションされません。選択しなかったプロジェクトは、後でマイグレーションすることができます。以前にマイグレーション済みのプロジェクト、現行バージョンのプロジェクト、エラーのあるプロジェクトはマイグレーションの対象外であり、選択されません。詳しくは、92 ページの『マイグレーションしない場合のプロジェクトのアップグレード』を参照してください。

e. 「**次へ**」をクリックします。 警告ウィンドウが表示され、「このバージョンのターゲット・アダプターでサポートされないプロパティーは、マイグレーション中に除去されます。」というメッセージが表示されます。

f. 「**OK**」をクリックします。

マイグレーションするプロジェクトの中に Inbound プロジェクトが少なくとも 1 つあると、次のメッセージが表示されます。「WebSphere JDBC Adapter 7.5 以降、Inbound 処理用の高可用性 Active-Active 機能は、Oracle データベース、IBM DB2、および Microsoft SQL Server の場合にサポートされます。この機能を使用すると、オプションで、同じイベント・テーブルに関連付けられた複数の Inbound インスタンスが同時に実行されるようにして、イベント処理のパフォーマンス全体を向上させることができます。この機能をサポートするためには、イベント・テーブル構造を更新する必要があります。高可用性 Active-Active 機能について詳しくは、インフォメーション・センターの『WebSphere JDBC Adapters』セクションを参照してください。」

g. 「**OK**」をクリックします。

6. 「変更内容の確認」ウィンドウで、マイグレーション対象の成果物のそれぞれで発生するマイグレーションの変更点を確認します。詳細を表示するには、+ 記号をクリックして各ノードを展開します。

7. マイグレーションを完了するため、以下の操作を実行します。

- 「**終了**」をクリックします。
- マイグレーション中に更新が必要なファイルが読み取り専用モードの場合、「**終了**」ボタンをクリックすることはできません。これらのファイルを表示するには、「**次へ**」をクリックします。「読み取り専用ファイルの更新」ウィンドウに読み取り専用ファイルが表示されます。これらのファイルを更新し、マイグレーションを続行するには、「**終了**」をクリックします。アダプターをマイグレーションしないでウィザードを終了するには、「**取り消し**」をクリックします。

マイグレーション・プロセスを実行する前に、ウィザードは、マイグレーションによって影響を受けるすべてのプロジェクトをバックアップします。 プロジェクトは、ワークスペース内の一時フォルダーにバックアップされます。何らかの理由でマイグレーションが失敗した場合、あるいは、終了前にマイグレーションを取り消すことにした場合、ウィザードは変更されたプロジェクトをすべて削除し、一時フォルダーに格納されていたプロジェクトで置き換えます。

マイグレーションが正常に完了したら、バックアップされたプロジェクトはすべて削除されます。

8. マイグレーションを完了した後、高可用性 Active-Active サポートを有効にするため、イベント・テーブル構造を手動でアップグレードする必要があります。**高可用性 Active-Active サポートの有効化**について詳しくは、182 ページの『デプロイメント・プロパティーの設定およびサービスの生成』を参照してください。イベント・テーブル構造を変更または作成するには、105 ページの『イベント・ストアの作成』を参照してください。
9. EAR ファイルをマイグレーションしている場合は、マイグレーション済みのアダプターおよび成果物を持つ新規の EAR ファイルを作成して、IBM Business Process Manager または WebSphere Enterprise Service Bus にデプロイすることもできます。 EAR ファイルのエクスポートおよびデプロイについて詳しくは、この資料で EAR ファイルについて説明しているトピックを参照してください。

タスクの結果

プロジェクトまたは EAR ファイルは、バージョン 7.5 へマイグレーションされます。アダプター・マイグレーション・ウィザードの終了後に外部サービス・ウィザードを実行する必要はありません。

マイグレーションしない場合のプロジェクトのアップグレード

アダプターを以前のバージョンからバージョン 7.5 にアップグレードする一方で、アダプター・プロジェクトの成果物をマイグレーションしないように選択できます。

このタスクについて

アダプター成果物が何も選択されていない状態でマイグレーション・ウィザードを実行すると、アダプターがインストールされ、アップグレードされます。成果物はマイグレーションされないため、ご使用のアプリケーションでは バージョン 7.5 のアダプターで提供される機能は利用できません。

手順

1. PI (プロジェクト交換) ファイルをワークスペースにインポートします。
2. 以前のバージョンの IBM Integration Designer で作成したプロジェクトがある場合は、ワークスペース・マイグレーション・ウィザードが自動的に開始され、マイグレーション対象のプロジェクトが選択されます。ウィザードに従って、ワークスペースのマイグレーションを完了します。詳しくは、http://bidoc.torolab.ibm.com:7500/help/index.jsp?topic=/com.ibm.wbpm.wid.imuc.doc/topics/tmigsrcart.htmlを参照してください。

3. Java EE パースペクティブで、プロジェクト名を右クリックし、「**コネクター・プロジェクトのマイグレーション**」をクリックします。 「**アダプター・マイグレーション**」ウィザードが表示されます。
4. 「プロジェクトの選択」ウィンドウで、依存関係のある成果物プロジェクトをクリアして、「**次へ**」をクリックします。 警告ウィンドウが開き、「このバージョンのターゲット・アダプターでサポートされないプロパティーは、マイグレーション中に除去されます。」というメッセージが表示されます。
5. 「**OK**」をクリックします。
6. 「変更内容の確認」ウィンドウで、プロジェクトの更新で発生したマイグレーションの変更点を確認します。詳細を表示するには、+ 記号をクリックして各ノードを展開します。
7. マイグレーションを完了するため、以下の操作を実行します。
   - 「**終了**」をクリックします。
   - マイグレーション中に更新が必要なファイルが読み取り専用モードの場合、「**終了**」ボタンをクリックすることはできません。これらのファイルを表示するには、「**次へ**」をクリックします。「読み取り専用ファイルの更新」ウィンドウに読み取り専用ファイルが表示されます。これらのファイルを更新し、マイグレーションを続行するには、「**終了**」をクリックします。アダプターをマイグレーションしないでウィザードを終了するには、「**取り消し**」をクリックします。

タスクの結果

これで、プロジェクトを WebSphere Adapter for JDBC バージョン 7.5 で使用できるようになりました。

バージョン 7.5 の WebSphere Adapters で使用するための、WebSphere Business Integration アプリケーションのマイグレーション

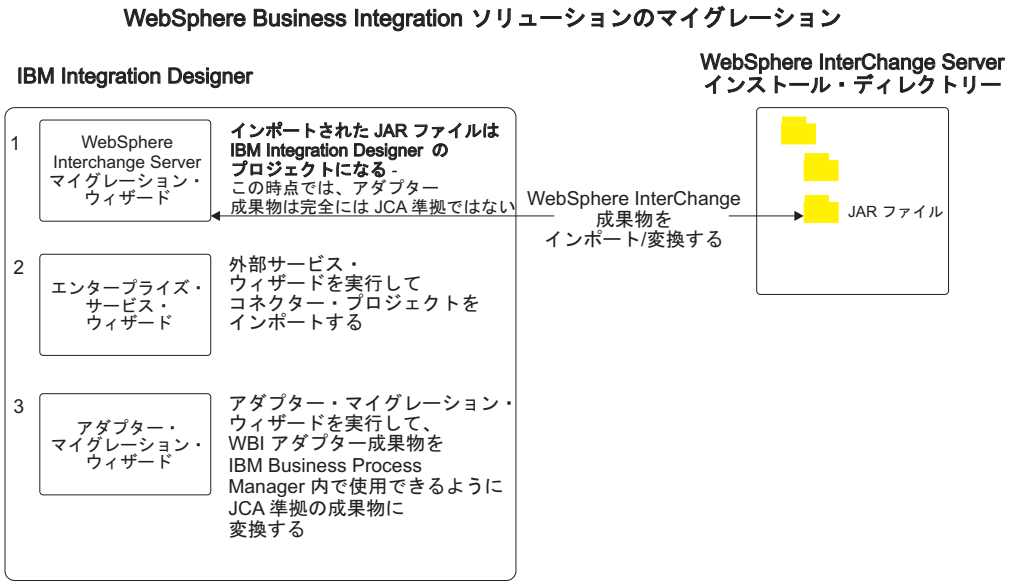
ご使用のアダプターのバージョン 7.5 との互換性を持たせるために、WebSphere Business Integration アプリケーションのマイグレーションを行う必要があります。

このタスクについて

WebSphere アダプターのバージョン 7.5 で使用するための、WebSphere Business Integration アプリケーションのマイグレーションは、複数ステップの処理です。まず、WebSphere InterChange Server からの成果物がマイグレーションされて変換されます。次に、IBM Integration Designer で成果物に対してプロジェクトが作成されます。残りの手順では、アダプター固有の成果物がマイグレーションされて、アダプターの バージョン 7.5 でサポートされる JCA 準拠の形式に変換されます。

例

以下の図は、WebSphere Business Integration のソリューションを WebSphere InterChange Server からマイグレーションして、これらのアプリケーションをバージョン 7.5 のアダプターで使用できるようにするためのウィザードを示しています。



WebSphere InterChange Serverからアプリケーションをマイグレーションするためのロードマップ

バージョン 7.5 の WebSphere Adapter for JDBC を WebSphere InterChange Server からのアプリケーションで使用するには、IBM Business Process Manager または WebSphere Enterprise Service Bus にデプロイして実行できるように、アプリケーション成果物をマイグレーションして、変換する必要があります。このタスクの概要を理解すれば、タスクを達成するのに必要な手順を実行できるようになります。

以下の図には、マイグレーション・タスクのフローを示しています。図の後に示す手順で、この作業の概要を説明します。これらの各ステップの実行方法の詳細については、このロードマップの後に記載するトピックを参照してください。

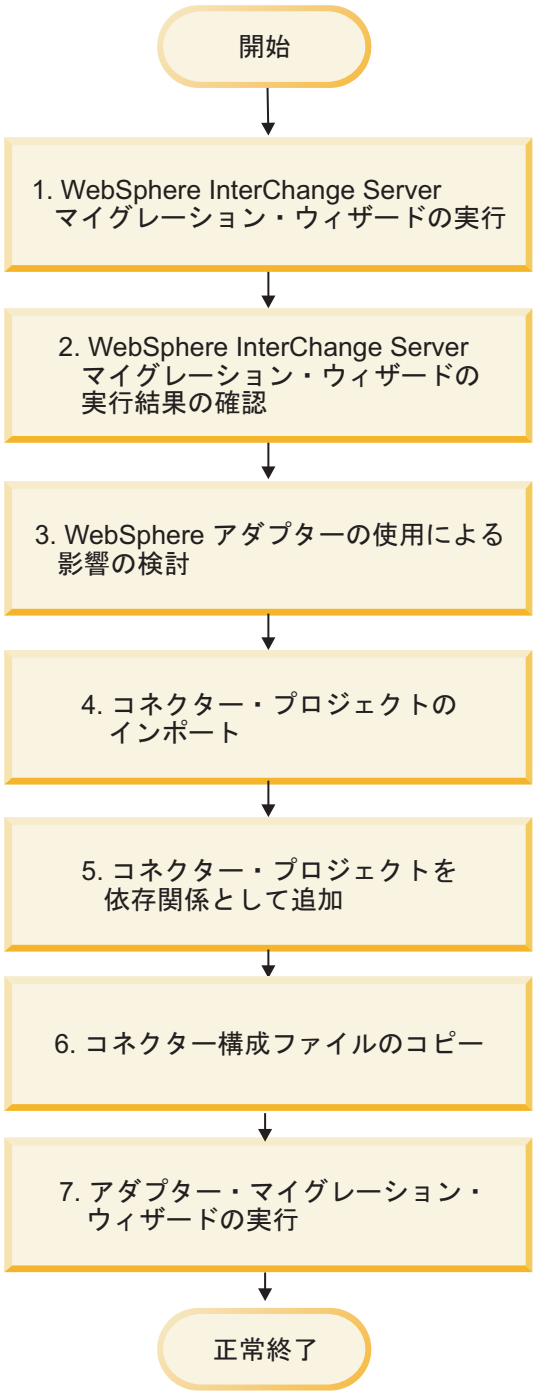


図 26. WebSphere InterChange Server からアプリケーションをマイグレーションする場合のロードマップ

WebSphere InterChange Server からのアプリケーションのマイグレーション

このタスクは、以下のステップからなります。

1. WebSphere InterChange Server マイグレーション・ウィザードを実行します。

WebSphere InterChange Server マイグレーション・ウィザードは、アプリケーション成果物を IBM Integration Designer へ移動します。 このタスクの完了時、マイグレーションされたアダプター成果物は、完全に JCA 準拠であるとは限りません。

2. WebSphere InterChange Server マイグレーションが成功したことを検証します。

   マイグレーション結果ウィンドウのすべてのメッセージを確認して、必要があれば対処します。

3. バージョン 7.5 の WebSphere Adapter for JDBCを使用する影響について考慮します。

   WebSphere InterChange Server アプリケーションのマイグレーションに関する考慮事項のほかに、マイグレーションされたアプリケーションにバージョン 7.5 の WebSphere Adapter for JDBCがどのように作用するかを考慮する必要があります。WebSphere InterChange Server アプリケーションでサポートされる一部のアダプター操作は、バージョン 7.5のアダプターではサポートおよび実装の仕方が異なる場合があります。

4. アダプター・マイグレーション・ウィザードを実行します。

   アダプター・マイグレーション・ウィザードを実行して、スキーマやサービス定義ファイル (.import、.export、.wsdl) などのアダプター固有成果物を、バージョン 7.5のアダプターで使用するために更新します。

WebSphere Business Integration Adapter のマイグレーションの考慮事項

WebSphere Adapter for JDBC バージョン 7.5 にマイグレーションすることにより、Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) に準拠した、サービス指向アーキテクチャー用に特別に設計されたアダプターになります。

注: WebSphere Business Integration Adapter と JCA 準拠アダプターは、異なる Inbound イベント・テーブルを使用します。マイグレーションが完了したら、JCA 準拠アダプターがサポートする新しい Inbound イベント・テーブルに手動で変更する必要があります。

操作の考慮事項

いくつかの WebSphere Business Integration Adapter 操作は、WebSphere Adapter for JDBC バージョン 7.5 ではサポート方法が異なります。次の操作を使用する予定であれば、追加の開発を行う必要があります。

DeltaUpdate

DeltaUpdate は、WebSphere Adapter for JDBCではサポートされていません。 ApplyChanges 操作を使用して、コンポーネントのデルタ処理を実装する必要があります。

RetrieveByContent

RetrieveAll 操作は、WebSphere Adapter for JDBC の中で、RetrieveByContent 操作と同等の操作としてサポートされます。

アプリケーション成果物

アダプターのマイグレーション・ウィザードを実行する前に、WebSphere InterChange Server マイグレーション・ウィザードを使用して、WebSphere Business Integration Adapter のアプリケーション成果物 (ビジネス・オブジェクト、マップ、コラボレーションなど) を生成してください。その後、アダプター・マイグレーション・ウィザードを実行してスキーマおよびサービス定義ファイル (.import、.export、および .wsdl) などのアダプター固有の成果物を更新し、アダプター固有の成果物が JCA 準拠のフォーマットに適切に変換されるようにします。

テスト環境におけるマイグレーション・ウィザードの最初の実行

WebSphere Business Integration Adapter から WebSphere Adapter for JDBC へのマイグレーションは、WebSphere Adapter for JDBC の バージョン 7.5 を使用するアプリケーションの変更を要する場合があるため、必ず最初に開発環境でマイグレーションを実行してアプリケーションをテストしてから、アプリケーションを実稼働環境にデプロイしてください。

WebSphere InterChange Server からのアプリケーション成果物のマイグレーション

アプリケーション成果物を IBM Integration Designer にマイグレーションするには、WebSphere InterChange Server マイグレーション・ウィザードを実行します。ウィザードにより、ほとんどの成果物がインポートされ、IBM Business Process Manager または WebSphere Enterprise Service Bus と互換性のあるフォーマットに変換されます。

始める前に

アプリケーション成果物を WebSphere InterChange Server 形式から IBM Business Process Manager または WebSphere Enterprise Service Bus と互換性のある成果物にマイグレーションするには、IBM Integration Designer 内から WebSphere InterChange Server マイグレーション・ウィザードを起動します。

WebSphere InterChange Server からの成果物のマイグレーションの準備について、およびマイグレーションの実行とマイグレーションが正常に行われたかどうかの検証を行う詳しい手順については、 http://bidoc.torolab.ibm.com:7500/help/topic/com.ibm.wbpm.wid.imuc.doc/topics/twics.html を参照してください。

このタスクについて

WebSphere InterChange Server マイグレーション・ウィザードを実行しても、アダプター固有成果物 (サービス記述子、サービス定義、ビジネス・オブジェクトなど) が IBM Business Process Manager または WebSphere Enterprise Service Bus 互換成果物に完全には変換されない場合があります。アダプター固有成果物のマイグレーションを完了するには、WebSphere InterChange Server マイグレーション・ウィザードを正常に実行した後で、アダプター・マイグレーション・ウィザードを実行します。

注: WebSphere InterChange Server マイグレーション・ウィザードの実行時には、リポジトリー内の各コネクターが同じアダプターのバージョンに設定されていることを確認してください。

タスクの結果

プロジェクトおよびアプリケーション成果物がマイグレーションされ、 IBM Business Process Manager 互換の成果物に変換されます。

次のタスク

アダプター・マイグレーション・ウィザードを実行して、アダプター固有成果物をマイグレーションします。

アダプター固有の成果物のマイグレーション

IBM Integration Designer で成果物に対してプロジェクトを作成すると、アダプター・マイグレーション・ウィザードを使用して、そのプロジェクトをマイグレーションすることができます。アダプター・マイグレーション・ウィザードを使用すると、バージョン 7.5 のアダプターで使用されるスキーマおよびサービス定義ファイル (.import、.export、および .wsdl) など、アダプター固有の成果物が更新されます。アダプター・マイグレーション・ウィザードの実行を完了したら、マイグレーションが完了するので、プロジェクトで作業したり、モジュールをデプロイしたりできます。

始める前に

アダプター・マイグレーション・ウィザードを実行する前に、以下の手順を行う必要があります。

- 87 ページの『マイグレーションに関する考慮事項』に記載されている情報を確認します。
- WebSphere InterChange Server マイグレーション・ウィザードを実行して、プロジェクトをマイグレーションし、IBM Business Process Manager または WebSphere Enterprise Service Bus で使用できるようにデータ・オブジェクトを変換します。

このタスクについて

マイグレーションが完了した後は、モジュールは、アダプターのバージョン 7.5 でのみ機能します。

IBM Integration Designer でマイグレーションを実行するには、以下のステップを完了してください。

手順

1. 既存のプロジェクトの場合は PI (プロジェクト交換) ファイルをワークスペースにインポートします。
2. 以前のバージョンの IBM Integration Designer で作成したプロジェクトがある場合は、ワークスペース・マイグレーション・ウィザードが自動的に開始され、マイグレーション対象のプロジェクトが選択されます。ウィザードに従って、ワークスペースのマイグレーションを完了します。詳しくは、

http://bidoc.torolab.ibm.com:7500/help/index.jsp?topic=/com.ibm.wbpm.wid.imuc.doc/topics/tmigsrcart.htmlを参照してください。

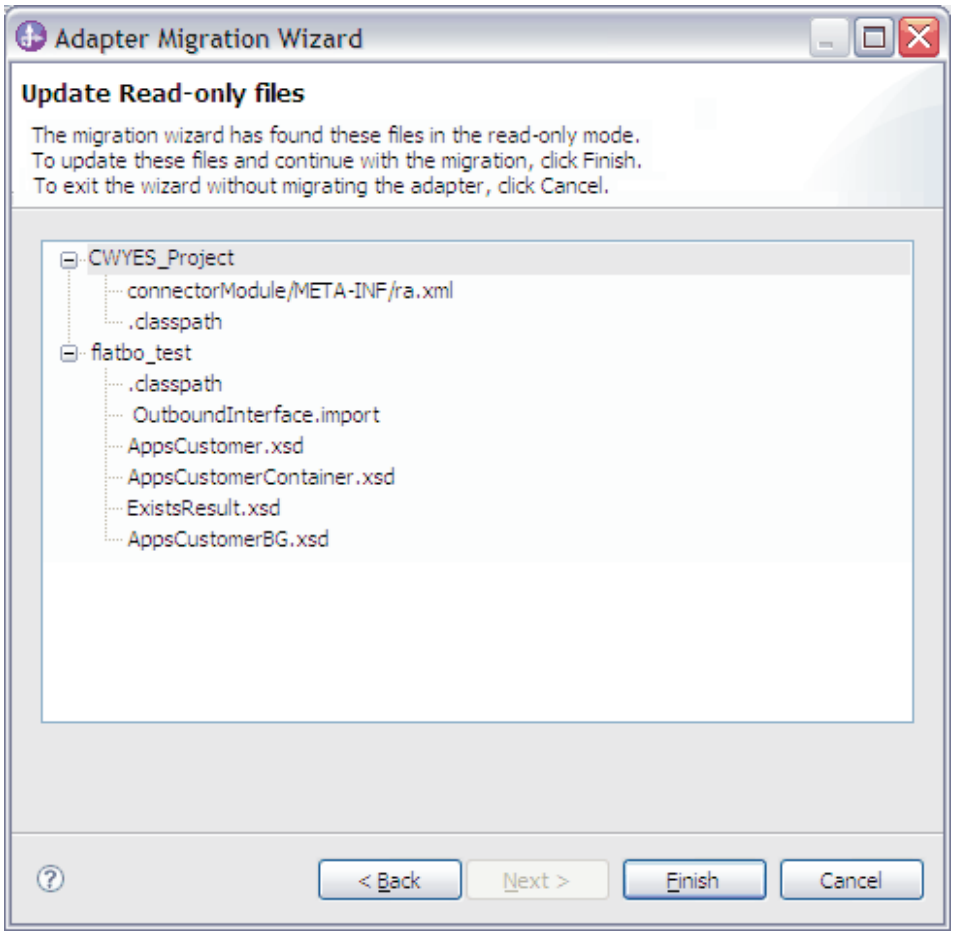
3. Java EE パースペクティブに切り替えます。
4. コネクター・プロジェクトを右クリックして、「**コネクター・プロジェクトのマイグレーション**」を選択します。

   Java EE パースペクティブで、右クリック・オプションを使用してモジュール・プロジェクトを選択し、「**アダプター成果物のマイグレーション**」を選択することにより、アダプター・マイグレーション・ウィザードを起動することもできます。

   注:

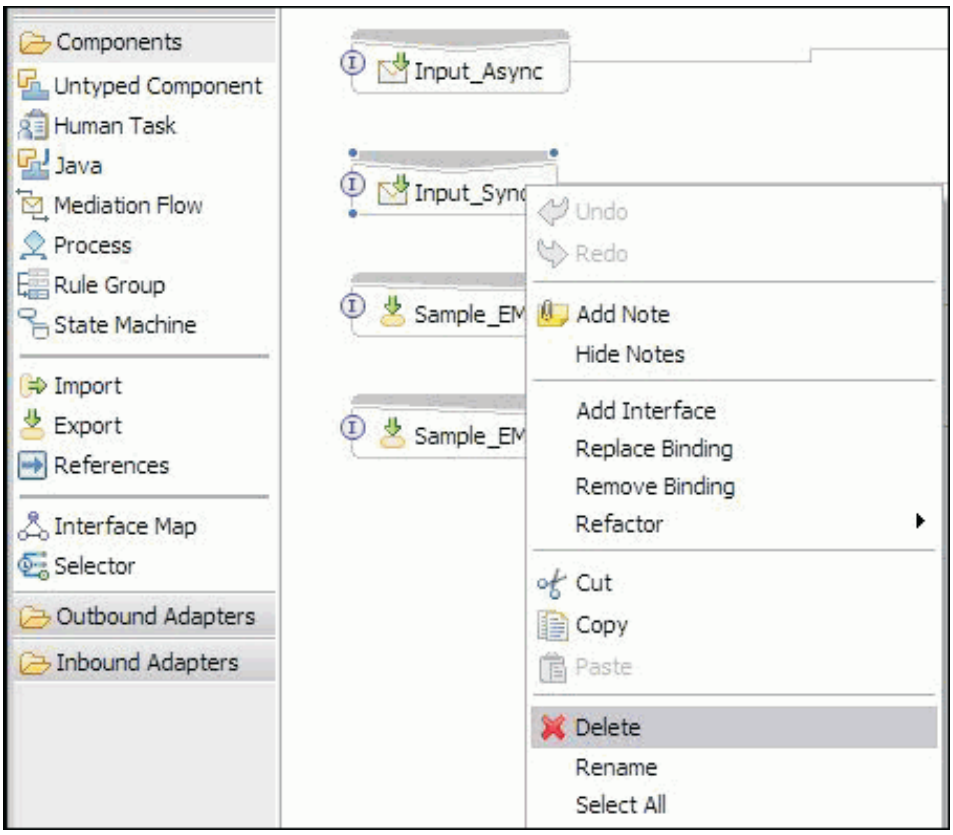
   マイグレーション・ウィザードでサポートされていないアダプター・タイプ (例えば、CICS/IMS アダプター) の場合、「**コネクター・プロジェクトのマイグレーション**」および「**アダプター成果物のマイグレーション**」のメニューは選択できません。アダプター・プロジェクトが最新のバージョンであり、このアダプター・プロジェクトを参照するモジュール・プロジェクトも最新のバージョンである場合、これらのメニューは使用不可になります。
   Java EE パースペクティブで、コネクター・プロジェクトからマイグレーション・ウィザードを起動する場合、デフォルトでは、依存関係のある成果物がすべて選択されます。依存関係のある成果物プロジェクトを選択しないと、そのプロジェクトはマイグレーションされません。
5. 「プロジェクトの選択」ウィンドウで、以下の手順を実行します。
   a. 「**ソース・コネクター**」フィールドに、マイグレーションするコネクター・プロジェクトの名前が表示されます。リストからソース・プロジェクトを選択します。
   b. 「**ターゲット・コネクター**」フィールドに、マイグレーションするコネクターの名前が表示されます。複数のバージョンのアダプターを処理する場合、このリストには互換性のあるすべてのコネクターの名前が表示されます。マイグレーションするコネクターを選択します。
   c. 「**ターゲットのバージョン**」フィールドに、前の手順で選択したターゲット・コネクターに対応するバージョンが表示されます。
   d. 「**依存関係のある成果物プロジェクト**」の領域には、マイグレーションされるアダプター成果物がリストされます。
   e. 「ようこそ」ページに表示されたタスクおよび警告を確認して、「**次へ**」をクリックします。 警告ウィンドウが開き、「このバージョンのターゲット・アダプターでサポートされないプロパティーは、マイグレーション中に除去されます。」というメッセージが表示されます。
   f. 「**OK**」をクリックします。
6. 「変更内容の確認」ウィンドウで、マイグレーション対象の成果物のそれぞれで発生するマイグレーションの変更点を確認します。詳細を表示するには、+ 記号をクリックして各ノードを展開します。
7. マイグレーションを完了するため、以下の操作を実行します。
   - 「**終了**」をクリックします。

- マイグレーション中に更新が必要なファイルが読み取り専用モードの場合、「**終了**」ボタンをクリックすることはできません。これらのファイルを表示するには、「**次へ**」をクリックします。「読み取り専用ファイルの更新」ウィンドウに読み取り専用ファイルが表示されます。これらのファイルを更新し、マイグレーションを続行するには、「**終了**」をクリックします。アダプターをマイグレーションしないでウィザードを終了するには、「**取り消し**」をクリックします。



マイグレーション・プロセスを実行する前に、ウィザードは、マイグレーションによって影響を受けるすべてのプロジェクトをバックアップします。プロジェクトは、ワークスペース内の一時フォルダーにバックアップされます。何らかの理由でマイグレーションが失敗した場合、あるいは、終了前にマイグレーションを取り消すことにした場合、ウィザードは変更されたプロジェクトをすべて削除し、一時フォルダーに保管されたプロジェクトで置き換えます。

8. 変更を有効にするために、「**プロジェクト**」>「**クリーン**」を選択して、ワークスペースをリフレッシュして再ビルドします。
9. マイグレーションが正常に完了すると、バックアップ・プロジェクトはすべて削除されます。 Sync Inbound フローはアダプターによって使用されないため、手動で削除します。 マイグレーションされたプロジェクトから、Input_Sync Inbound フローを選択して右クリックし、「**削除**」を選択します。



10. EAR ファイルをマイグレーションしている場合は、マイグレーション済みアダプターおよび成果物のある新規の EAR ファイルを作成して、IBM Business Process Manager または WebSphere Enterprise Service Bus にデプロイします。EAR ファイルのエクスポートおよびデプロイについては、 207 ページの『実稼働のためのモジュールのデプロイ』を参照してください。

タスクの結果

プロジェクトがバージョン 7.5 にマイグレーションされます。アダプター・マイグレーション・ウィザードの終了後に外部サービス・ウィザードを実行する必要はありません。

マイグレーション後のインポート、エクスポート、および WSDL ファイルの変更

WebSphere InterChange Server マイグレーション・ウィザードによりアプリケーション成果物が IBM Integration Designer に移動すると、各サービス定義ファイル (インポート・ファイル、エクスポート・ファイル、WSDL ファイル) に変更内容が反映されます。

このタスクの完了時、マイグレーションされたアダプター成果物は、完全に JCA 準拠であるとは限りません。アダプター・マイグレーション・ウィザードを実行することによって、アダプター固有成果物 (サービス記述子、サービス定義、ビジネス・オブジェクトなど) の JCA 互換形式へのマイグレーションを完了させることができます。

インポート・ファイルの変更

マイグレーション中、影響を受けるモジュール成果物がインポート・ファイルにマイグレーションされます。既存の JMS バインディング・プロパティーは、インポート・ファイルの EIS バインディング・プロパティーに変更されます。インポート・ファイルに追加されるその他のプロパティー詳細には、データ・バインディング構成、管理接続ファクトリー・プロパティー内の接続情報への変更、およびいくつかの新規メソッド・バインディングに関する情報が含まれています。

エクスポート・ファイルの変更

マイグレーション中、影響を受けるモジュール成果物がエクスポート・ファイルにマイグレーションされます。既存の JMS バインディング・プロパティーは、エクスポート・ファイルの EIS バインディング・プロパティーに変更されます。エクスポート・ファイルに追加されるその他のプロパティー詳細には、データ・バインディング構成、活動化仕様プロパティー内の接続情報への変更、およびいくつかの新規メソッド・バインディングに関する情報が含まれています。

マイグレーションの後の WSDL ファイルの変更

マイグレーション時に、影響を受けたモジュール成果物は対応する WSDL ファイルにマイグレーションされます。これには、 JDBC 固有のサービス記述 WSDL 成果物が含まれます。サービス記述ファイルは JCA 互換となります。WSDL ファイルには、各操作の入力タイプと出力タイプが入ります。特定の入力タイプに対して Inbound 操作および Outbound 操作の両方が機能して、その操作の実行後、対応する出力タイプが生成されます。

注:

- プロジェクトの複数のトップレベル Inbound ビジネス・オブジェクトをマイグレーションする場合、最初のトップレベル・ビジネス・オブジェクトの Inbound 機能のみが正しく動作します。 他のトップレベル Inbound ビジネス・オブジェクトが正しく機能するようにするには、正しい宛先サービスを呼び出すように Input_Processing.java と Input_Async_Processing.java クラスの "emit + [verb name] + after image + [business object name]" メソッドを手動で変更する必要があります。
- 有効でない、あるいは、WebSphere Adapter for JDBC でサポートされていない WebSphere Business Integration Adapter for JDBC のプロパティーは、マイグレーションされた成果物からは除去されます。

サンプルおよびチュートリアル

ユーザーが、WebSphere Adapters を使用する際に役立つように、サンプルおよびチュートリアルがビジネス・プロセス・マネージメントのサンプルおよびチュートリアルの Web サイトから入手できます。

サンプルおよびチュートリアルには、以下のいずれかの方法でアクセスできます。

- IBM Integration Designer のウェルカム・ページで、「**サンプルおよびチュートリアルに移動**」をクリックします。「サンプルおよびチュートリアル」ペインで、

「サンプルの詳細 (More samples)」の下の「**取得 (Retrieve)**」をクリックします。表示されたカテゴリーをブラウズして、選択を行います。

- ビジネス・プロセス・マネージメントのサンプルおよびチュートリアルの Web サイト (http://publib.boulder.ibm.com/bpcsamp/index.html) から入手できます。

デプロイメント用のモジュールの構成

アダプターを IBM Business Process Manager または WebSphere Enterprise Service Bus 上にデプロイできるように構成するには、IBM Integration Designer を使用して、アダプターをデプロイするときに EAR ファイルとしてエクスポートされるモジュールを作成します。次に、ディスカバーの対象となるビジネス・オブジェクトと、そのディスカバーを行うシステムを指定します。

モジュールの構成のためのロードマップ

ランタイム環境で WebSphere Adapter for JDBC を使用できるようにするには、まずモジュールを構成する必要があります。このタスクの概要を理解すれば、タスクを達成するのに必要な手順を実行できるようになります。

IBM Integration Designer を使用してアダプターのモジュールを構成し、使用できるようにします。以下の図は、構成作業の流れを示しています。また、図の後に示す手順で、この作業の概要を説明します。これらの各ステップの実行方法の詳細については、このロードマップの後に記載するトピックを参照してください。

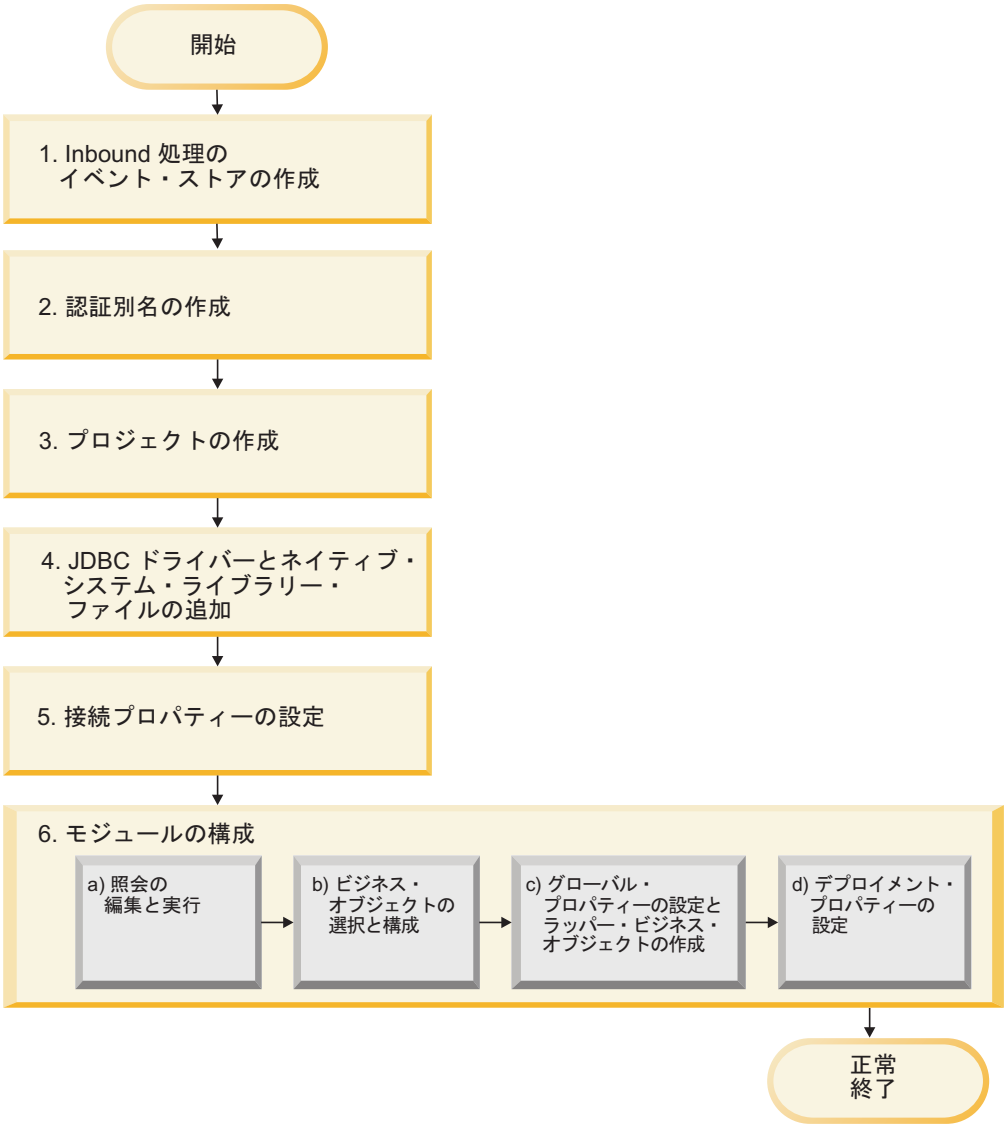


図 27. モジュールの構成のためのロードマップ

デプロイメント環境のためのモジュールの構成

この作業は、次のステップから成ります。

1. Inbound 処理のイベント・ストアを作成します。
2. 暗号化パスワードで、データベース サーバーにアクセスするための認証別名を作成します。この手順は、オプションです。パスワードおよび ID の扱いに関するポリシーに応じて実行してください。サーバーを使用して、この手順を実行してください。
3. プロジェクトを作成します。最初に、IBM Integration Designer で外部サービス・ウィザードを開始し、モジュールを作成してデプロイするプロセスを開始してください。このウィザードによって、プロジェクトが作成されます。これは、モジュールに関連付けられたファイルを編成するために使用されます。
4. WebSphere Adapter for JDBC が必要とする JDBC ドライバーおよびネイティブ・システム・ライブラリー・ファイルをプロジェクトに追加します。モジュー

ルを EAR ファイルとしてエクスポートし、その EAR ファイルをサーバーにデプロイする際にも、これらの依存性が必要になります。

5. 接続プロパティーを設定します。これは、外部サービス・ウィザードが データベース に接続してオブジェクトおよびサービスをディスカバーする際に必要です。
6. 外部サービス・ウィザードを使用して、データベースからビジネス・オブジェクトおよびサービスを検出して選択し、ビジネス・オブジェクト定義および関連する成果物を生成することで、Inbound 処理または Outbound 処理用にモジュールを構成します。
   a. アクセス可能なデータベース・オブジェクトをディスカバーする照会を編集して、実行します。
   b. ビジネス・オブジェクトを選択して、Inbound 処理または Outbound 処理用に構成します。
   c. グローバル・プロパティーを操作用に設定して、ラッパー・ビジネス・オブジェクトを作成します。
   d. デプロイメント・プロパティーを設定します。アダプターは、実行時にこれを使用して、データベース に接続します。次に、サービスを生成します。外部サービス・ウィザードを使用して、新規モジュールを保存してください。ここには、構成済みのビジネス・オブジェクト、インポート・ファイルまたはエクスポート・ファイル、およびサービス・インターフェースが含まれています。 .

イベント・ストアの作成

アダプターが Inbound イベントを処理できるようにするには、事前にデータベースにイベント・ストアを作成する必要があります。必要に応じて、ユーザー・テーブル上にトリガーを設定して、イベント・テーブルを取り込むことができます。

このタスクについて

イベントの Inbound 処理が必要な場合にのみ、このタスクを実行してください。イベントの報告対象のテーブルを含むデータベースにイベント・ストアを作成します。

手順

1. イベント・ストアを作成します。 IBM DB2、IBM DB2 for z/OS、 Oracle、または MicrosoftSQL Server データベース用のイベント・ストアを作成するために、以下のサンプル・スクリプトが用意されています。
   - `scripts_db2.sql`
   - `scripts_db2_zOS.sql`
   - `scripts_oracle.sql`
   - `scripts_mssql.sql`

   注: `scripts_db2_zOS.sql` スクリプトを実行する前に、ストレージ・グループ JDDBSTO が存在することを確認してください。

これらのファイルは、*IID_installation_dir*/ResourceAdapters/JDBC_*version*/Scripts ディレクトリーにあります。ここで、*IID_installation_dir* は IBM Integration Designer のインストール・ディレクトリーであり、 *version* はアダプターのバージョンを示します。

2. ユーザー・テーブルへの変更によって、イベント・ストアに保管されるイベントが自動的に生成されるように、必要に応じてトリガーをユーザー・テーブル上にセットアップします。サンプル・スクリプトには、トリガーを作成して、イベント・ストアを取り込む方法についての例が含まれています。

タスクの結果

イベント・ストアは、イベント処理で使用可能です。

認証別名の作成

認証別名は、アダプターがデータベースへのアクセスに使用するパスワードを暗号化する機能です。アダプターは、アダプター・プロパティーに保管されたユーザー ID とパスワードを使用する代わりに、この認証別名を使用して データベースに接続することができます。

始める前に

認証別名を作成するには、IBM Business Process Manager または WebSphere Enterprise Service Bus の管理コンソールへのアクセス権が必要です。また、データベースに接続するために使用するユーザー名とパスワードを知っておく必要があります。

以下の手順は、IBM Integration Designer によって管理コンソールにアクセスする方法を示しています。管理コンソールを直接使用する (IBM Integration Designer からアクセスしない) 場合は、管理コンソールにログオンしてステップ 2 (107 ページ) に進んでください。

このタスクについて

認証別名を使用すると、他のユーザーから見える可能性があるアダプター構成プロパティーに、パスワードを平文で格納する必要がなくなります。

認証別名を作成するには、以下の手順に従います。

手順

1. 管理コンソールを開始します。

   IBM Integration Designer によって管理コンソールを開始するには、以下の手順を実行します。

   a. Integration Designer の Business Integration パースペクティブで、「**サーバー**」タブをクリックします。

   b. サーバーの状況が「**開始済み**」でない場合は、サーバーの名前 (例えば、**IBM Business Process Manager**) を右クリックし、「**開始**」をクリックします。 サーバーの状況が Started になるのを待ちます。

c. サーバーの名前を右クリックし、「**管理コンソールの実行**」をクリックします。

d. 管理コンソールにログオンします。管理コンソールにユーザー ID およびパスワードが必要な場合は、ID およびパスワードを入力して、「**ログイン**」をクリックします。ユーザー ID およびパスワードが必要ない場合は、「**ログイン**」をクリックします。

2. 管理コンソールで、「**セキュリティー**」 > 「**グローバル・セキュリティー**」をクリックします。

3. 「**Java 認証・承認サービス**」の下にある「**J2C 認証データ**」をクリックします。

4. 認証別名を作成します。

a. 表示された J2C 認証別名のリストで、「**新規作成**」をクリックします。

b. 「一般プロパティー」エリアで、「**別名**」フィールドに認証別名の名前を入力します。

c. データベースへの接続の確立に必要なユーザー ID およびパスワードを入力します。

d. オプション: 別名の説明を入力します。

e. 「**OK**」をクリックします。

新規に作成された別名が表示されます。

別名のフルネームは、ノード名および指定した認証別名で構成されます。例えば、ノード widNode に ProductionServerAlias という名前で別名を作成する場合、フルネームは widNode/ProductionServerAlias となります。このフルネームは、後続の構成ウィンドウで使用する名前です。

f. 「**保存**」をクリックします。

タスクの結果

ウィザードの後半でアダプター・プロパティーを構成するときに指定する認証別名が作成されました。

プロジェクトの作成

モジュールの作成とデプロイのプロセスを開始するには、IBM Integration Designer の外部サービス・ウィザードを開始します。このウィザードは、モジュールに関連付けられたファイルを編成するために使用される、コネクター・プロジェクトを作成します。

始める前に

データベースへの接続の確立に必要な情報を収集済みであることを確認します。例えば、データベース の名前または IP アドレス、およびアクセスに必要なユーザー ID とパスワードが必要です。

このタスクについて

既存のプロジェクトがある場合は、新規オブジェクトを作成する代わりに、そのプロジェクトを使用できます。ウィザードの開始前に選択してください。

手順

1. 外部サービス・ウィザードを開始するには、IBM Integration Designer の Business Integration パースペクティブに進み、「**ファイル**」 > 「**新規**」 > 「**外部サービス**」の順にクリックします。
2. 「新規外部サービス」ウィンドウで、「**アダプター**」ノードを展開し「**JDBC**」を選択します。
3. 「**次へ**」をクリックします。
4. 「アダプターの選択」ウィンドウで「**IBM WebSphere Adapter for JDBC (IBM : *version*)**」を選択します。ここで、*version* は使用したいアダプターのバージョンです。
5. 「**次へ**」をクリックします。
6. 「RAR ファイルのインポート」ウィンドウで、「**コネクター・プロジェクト**」にあるデフォルト・プロジェクト名を受け入れるか、別の名前を入力します。
7. 「**ターゲット・ランタイム環境**」 フィールドで、モジュールをデプロイするサーバーのタイプを選択します。ウィザードは、そのサーバーに対して適切な成果物を作成します。
8. 「**次へ**」をクリックします。 必要なファイルおよびライブラリーの位置指定ウィンドウが表示されます。

タスクの結果

アダプターの RAR ファイルを含む新規のコネクター・プロジェクトが作成されます。プロジェクトは、ビジネス・インテグレーション・パースペクティブにリストされます。

次のタスク

外部サービス・ウィザードでの作業を続行します。次のステップでは、データベース固有のファイルをプロジェクトに追加します。

外部ソフトウェア依存関係の追加

外部サービス・ウィザード がデータベース・サーバーと通信できるようにするには、データベースの特定ファイルのコピーが必要です。外部サービス・ウィザードを使用して、JDBC ドライバーと必要なネイティブ・システム・ライブラリー・ファイルが格納されている JAR ファイルの場所を指定します。

始める前に

このタスクを実行するには、IBM Integration Designer で外部サービス・ウィザードを実行します。

このタスクについて

モジュール構成時にこのタスクを実行するだけでなく、場合によっては、IBM Business Process Manager または WebSphere Enterprise Service Bus にファイルをデプロイする必要があります。

手順

1. データベース管理者またはデータベース・ソフトウェアの Web サイトから、ご使用のデータベース・ソフトウェアおよびオペレーティング・システムの JDBC ドライバー固有ファイルまたはネイティブ・ライブラリーを入手します。 必要なファイルの種類は、データベース・サーバーによって異なります。JDBC ドライバー固有のファイルが JRE 1.6 と互換性があることを確認してください。次の表に、一般的なデータベース・ソフトウェアに必要な JDBC ドライバー・ファイルをリストします。

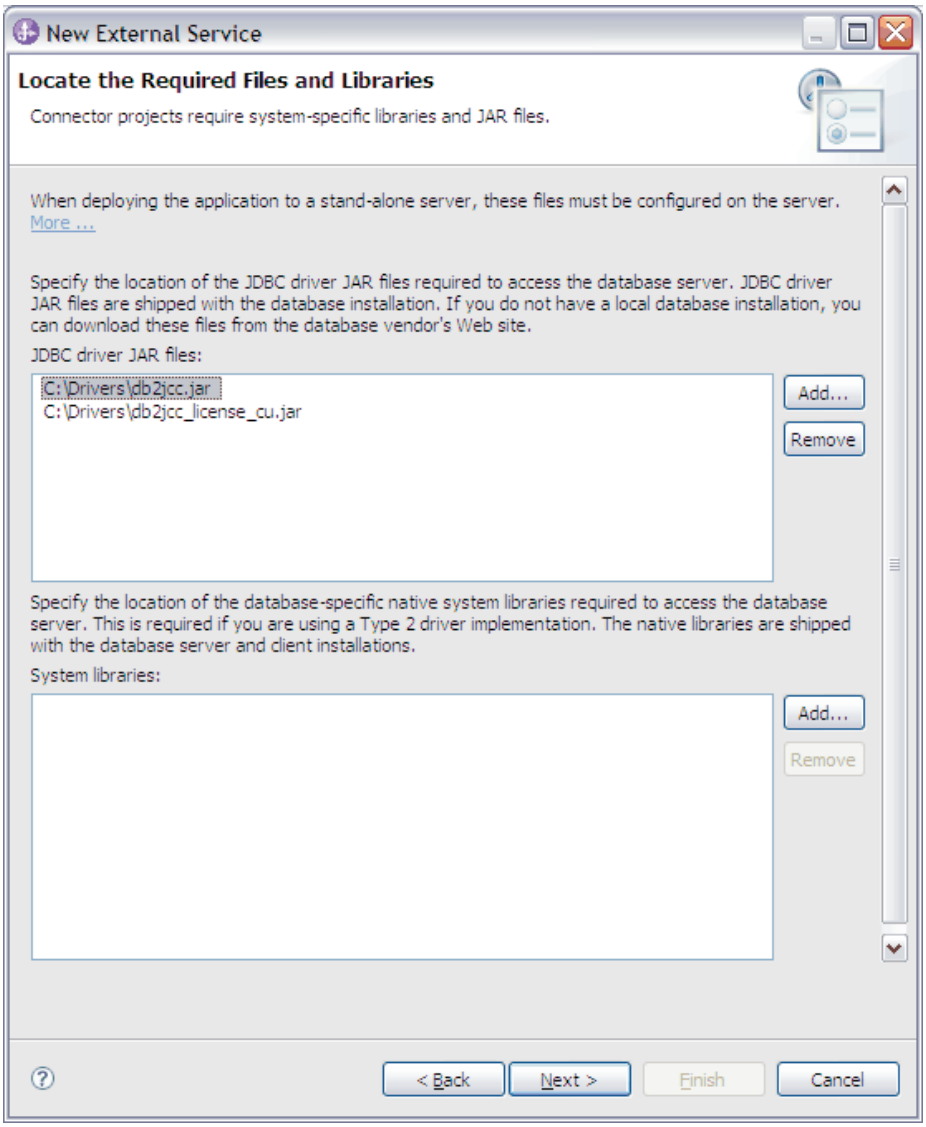
表 16. 共通データベース・ソフトウェアの JDBC ドライバー・ファイル

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
IBM DB2 Universal Database™ (Linux、UNIX、および Windows 版)	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cu.jar	なし
IBM DB2 for z/OS	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cisuz.jar	なし
IBM DB2 for IBM i	IBM Toolbox for Java リモート・ドライバー (Type 4)	jt400.jar db2jcc_license_cisuz.jar	なし
	IBM Toolkit for Java ネイティブ・ドライバー* (Type 2)	db2_classes.jar	なし
IBM DB2 Universal Database (Linux、UNIX、および Windows 版)	IBM DB2 Universal (Type 2)	db2java.zip	なし
Oracle	Thin ドライバー	ojdbc6.jar ランタイム環境で XML データ・タイプを扱うには、さらに以下の追加ライブラリーが必要です。 xdb.jar xmlparserv2.jar	なし
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	なし

表 16. 共通データベース・ソフトウェアの JDBC ドライバー・ファイル (続き)

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
Informix	IBM Informix JDBC ドライバー	ifxjdbc.jar ifxjdbcx.jar および いくつかのサポート .jar ファイル 詳しくは、「IBM Informix JDBC ドライバ プログラマーズ ガイド」を参照してください。	なし
* IBM Toolkit for Java ネイティブ・ドライバーを使用して、アダプター実行時のデータベースに接続することはできますが、これをウィザード実行時の接続に使用することはできません。ディスカバリー・プロセス中は、IBM Toolbox for Java リモート・ドライバー、または、IBM DB2 Universal ドライバーを使用する必要があります。ただし、実行時にネイティブ・ドライバーを使用するようにモジュールを構成することはできます。これは、「サービス生成およびデプロイメント・プロパティーの指定」ウィンドウで行います。			

2. 「必要なファイルおよびライブラリーの位置指定」ウィンドウで、プロジェクトに必要な JDBC ドライバー固有ファイルの場所を指定します。
   a. 「**JDBC ドライバー JAR ファイル**」フィールドで、「**追加**」をクリックして、JDBC ドライバー・ファイルを選択します。 JDBC ドライバーについて詳しくは、「JDBC ドライバーについてのよくある質問 (Frequently asked questions about JDBC drivers)」を参照してください。
   b. JDBC タイプ 2 ドライバーを使用する場合は、「**システム・ライブラリー**」フィールドで「**追加**」をクリックして、データベース・サーバーへのアクセスに必要なネイティブ・ システム・ライブラリーを追加します。 タイプ 4 の JDBC ドライバーを使用する場合のみ、このフィールドを空のままにします。



3. 「**次へ**」をクリックします。 処理方向の選択ウィンドウが表示されます。

タスクの結果

ウィザードに、データベース・サーバーと通信するために必要なファイルがあります。

次のタスク

外部サービス・ウィザードでの作業を続行します。次のステップでは、ウィザードがデータベースに接続するために必要となる情報を設定します。

外部サービス・ウィザードの接続プロパティーの設定

外部サービス・ウィザードがデータベース・オブジェクトをディスカバーするためにデータベース・インスタンスに接続する際に使用する接続プロパティーを指定します。

始める前に

接続プロパティーを構成する前に、外部サービス・ウィザードを始動しておく必要があります。

Sybase データベースでストアード・プロシージャーのディスカバーを計画している場合、そのストアード・プロシージャーのトランザクション・モードの設定が「**データベース接続時に自動コミットを設定**」プロパティーを使用するかどうかの決定に影響する可能性があることに注意してください。ストアード・プロシージャーのトランザクション・モードの設定がどのように「**データベース接続時に自動コミットを設定**」にチェック・マークを付けるかどうかの決定に影響するかは、トラブルシューティングおよびサポート情報のトピックの*「一般的な問題の解決策」* を参照してください。

このタスクについて

外部サービス・ウィザードでは、ディスカバリーのためのデータベース接続やサービス記述の作成にこれらのプロパティーが必要となります。プロパティーについて詳しくは、310 ページの『ウィザードの接続プロパティー』を参照してください。

手順

1. 「処理方向の選択」ウィンドウで、「**Outbound**」または「**Inbound**」を選択し、「**次へ**」をクリックします。
2. 「ディスカバリー・プロパティーの指定」ウィンドウで、ウィザードがデータベースへの接続に使用する接続プロパティーを指定します。接続プロパティーは、アダプターが接続するデータベースによって異なります。サポートされるデータベースに適用されるプロパティーは、以下のステップで説明されます。
   a. データベース・ソフトウェアのリストで、ご使用の製品およびバージョンを選択します。 「**プロパティー (Properties)**」領域には、データベース固有の接続プロパティーを指定するためのフィールドが表示されます。

      注: IBM DB2 Version 9.1 for z/OS の場合、「**V8 (新機能モード) (V8 (New-Function Mode))**」を選択します。

      注: Derby の場合、「**汎用 JDBC (Generic JDBC)**」を選択してください。
   b. 「**JDBC ドライバー・タイプ**」フィールドで、使用する JDBC ドライバーのタイプを選択します。

      注: IBM DB2 for IBM i の場合、 `AS/400 Toolbox for Java` または `IBM DB2 Universal` を選択して、データベース・オブジェクトをディスカバーします。実行時にサーバー上のローカル・アクセスでネイティブ・ドライバーを使用するように、後でモジュールを構成できます。

      注: Derby データベースを使用していて、データベース・ソフトウェアとして「**汎用 JDBC**」を選択している場合、「**JDBC ドライバー・タイプ**」フィールドには、あらかじめ値 **その他** が入ります。
   c. 「**データベース**」フィールドに、データベース名を指定します。Oracle データベースの場合、これはシステム ID (SID) です。

d. 「**ホスト名**」フィールドには、データベース・サーバーのホスト名または IP アドレスを指定します。IP アドレスを IPv6 形式で指定する場合は、アドレスを大括弧 (**[]**) で囲んでください。

e. 「**ポート番号**」フィールドで、データベースに接続するためのポート番号を指定します。

f. DB2、Oracle、Microsoft SQL Server および Informix データベースで、「**JDBC ドライバー・タイプ**」で特定のドライバーを選択した場合、ウィザードは 「**JDBC ドライバー・クラス名**」フィールドに対してデフォルト値を指定し、他の接続フィールドから「**データベース URL**」フィールドの値を作成します。さらに、一部の特定の JDBC ドライバー・タイプでは、ウィザードにより、「**ポート番号**」にもデフォルト値が入ります。任意のデータベース・ソフトウェア (例えば、Derby など) および他の特定のドライバーについて、ドライバーに 「**その他**」 を選択した場合、ドライバー・クラス名およびデータベース URL を指定する必要があります (ただし、データベース URL の一部が入力済みの場合があります)。 313 ページの『データベース URL』および 315 ページの『JDBC ドライバー・クラス名』を参照してください。

g. **Informix データベースの場合のみ**: 「**サーバー名**」フィールドに、アダプターが接続する Informix データベース・サーバーの名前を指定します。

h. 「**追加の JDBC ドライバー接続プロパティー (Additional JDBC driver connection properties)**」フィールドで、データベース接続時に設定される追加プロパティーを指定します。1 組以上の *name*:*value* ペアをセミコロン文字 (**;**) で区切って指定します。 例えば、次のようになります。

```
loginTimeout:20;readOnly:true;
securityMechanism:USER_ONLY_SECURITY
```

接続情報はディスカバリー・プロセスでのみ使用されます。ウィザードの後のほうでは、実行時用として別の接続情報を指定できます。

3. ウィザードからデータベースに接続するために使用するユーザー名とパスワードを、「**ユーザー名**」フィールドと「**パスワード**」フィールドに入力します。 このユーザー名はディスカバリー・プロセスでのみ使用され、保存されません。ウィザードの後のほうでは、実行時用として別のユーザー名およびパスワード、または別の認証方式を指定できます。

4. オプション: 「**ビジネス・オブジェクト名のプレフィックス**」フィールドに、ビジネス・オブジェクト名の先頭に付けるストリングを入力します。

5. 実行時にアダプターの双方向言語サポートを有効にするには、以下の手順を実行します。

   a. 「**拡張**」をクリックします。

   b. 「**BiDi プロパティー**」領域で、「**BiDi 変換**」を選択します。

   c. 順序付けスキーマ、テキスト方向、対称スワッピング、文字シェーピング、および数字シェーピングの各プロパティーを設定して、双方向変換の実行方法を制御します。

6. オプション: Sybase データベースで JConnect ドライバーを使用してディスカバリーを実行しているとき、ストアード・プロシージャーのトランザクション・モード設定により、アダプターがストアード・プロシージャーからの結果セットをディスカバーしないようにするシナリオもありえます。 ストアード・プロシージャーのトランザクション・モード・プロパティーがデフォルト値の「非チェー

ン・モード」または「Transact-SQL モード」に設定されている場合は、**「拡張プロパティー」**エリアで**「データベース接続時に自動コミットを設定」**チェック・ボックスを選択します。**「データベース接続時に自動コミットを設定」**チェック・ボックスを選択すると、ストアード・プロシージャーの「非チェーン・モード」構成をバイパスし、アダプターがストアード・プロシージャーからの結果セットをディスカバーできるようになります。**「データベース接続時に自動コミットを設定」**チェック・ボックスを選択せず、ストアード・プロシージャーのトランザクション・モード・プロパティーを「非チェーン・モード」に設定していると、アダプターは**「オブジェクトを選択に追加できません」**というエラーを返します。Sybase におけるストアード・プロシージャーのトランザクション・モードの設定がディスカバリー時のアダプターの処理にどのように影響するかについて詳しくは、トラブルシューティングおよびサポート情報のトピックの*「一般的な問題の解決策」* を参照してください。

注: ストアード・プロシージャーのトランザクション・モード・プロパティーが「チェーン・モード」に設定されている場合は、**「データベース接続時に自動コミットを設定」**チェック・ボックスを選択する必要はありません。**「データベース接続時に自動コミットを設定」**のプロパティーの詳細な説明は、317 ページの『データベース接続時に自動コミットを設定』を参照してください。

7. ウィザードのログ・ファイルの場所またはログに含まれる情報量を変更するには、**「ウィザードのロギング・プロパティーを変更する (Change the logging properties for the wizard)」**チェック・ボックスを選択してから、以下の情報を指定します。
   - **「ログ・ファイル出力場所 (Log file output location)」**フィールドに、ウィザードのログ・ファイルの場所を指定します。
   - **「ロギング・レベル (Logging level)」**フィールドに、記録するエラーの重大度を指定します。

   このログ情報はウィザードでのみ使用されます。実行時は、アダプターはサーバーの標準ログ・ファイルおよびトレース・ファイルにメッセージおよびトレース情報を書き込みます。
8. **「次へ」**をクリックします。

   ウィザードに、例外 eccom.ibm.adapter.framework.BaseException を示すウィンドウが表示された場合は、データベース・サーバーに接続できません。メッセージには、問題の考えられる原因について追加情報が含まれています。また、**「ログ・ファイル出力場所 (Log file output location)」**で指定されたディレクトリーにあるログを確認できます。接続情報が正しいことを確認してください。

タスクの結果

外部サービス・ウィザードは、データベースに接続され、「エンタープライズ・システムでのオブジェクトの検索」ウィンドウが表示されます。

次のタスク

ウィザードでの作業を続行します。次のステップでは、データベースを調べて、ウィザードでビジネス・オブジェクトを作成する対象となるオブジェクトを見つけます。

Outbound 処理のモジュールの構成

アダプターを Outbound 処理に使用するようにモジュールを構成するには、IBM Integration Designer 内で外部サービス・ウィザードを使用して、データベースからビジネス・オブジェクトおよびサービスを検出して選択し、ビジネス・オブジェクト定義および関連する成果物を生成します。

Outbound 処理のデータベース・オブジェクトのディスカバー

データベースに接続した後は、データベース・オブジェクトを検索する照会を実行します。ディスカバーされたオブジェクトのツリーを参照して、データベース内のオブジェクトの構造を理解します。また、フィルターを使用して、参照したいデータベース・オブジェクトのみを表示します。ユーザー定義データベース照会およびユーザー定義バッチ SQL ステートメントとして作成するビジネス・オブジェクトの数を定義します。

始める前に

データベースのアクセスに必要なプログラムのデータ要件を知っていなければなりません。例えば、データベースに関する以下の情報が必要になります。

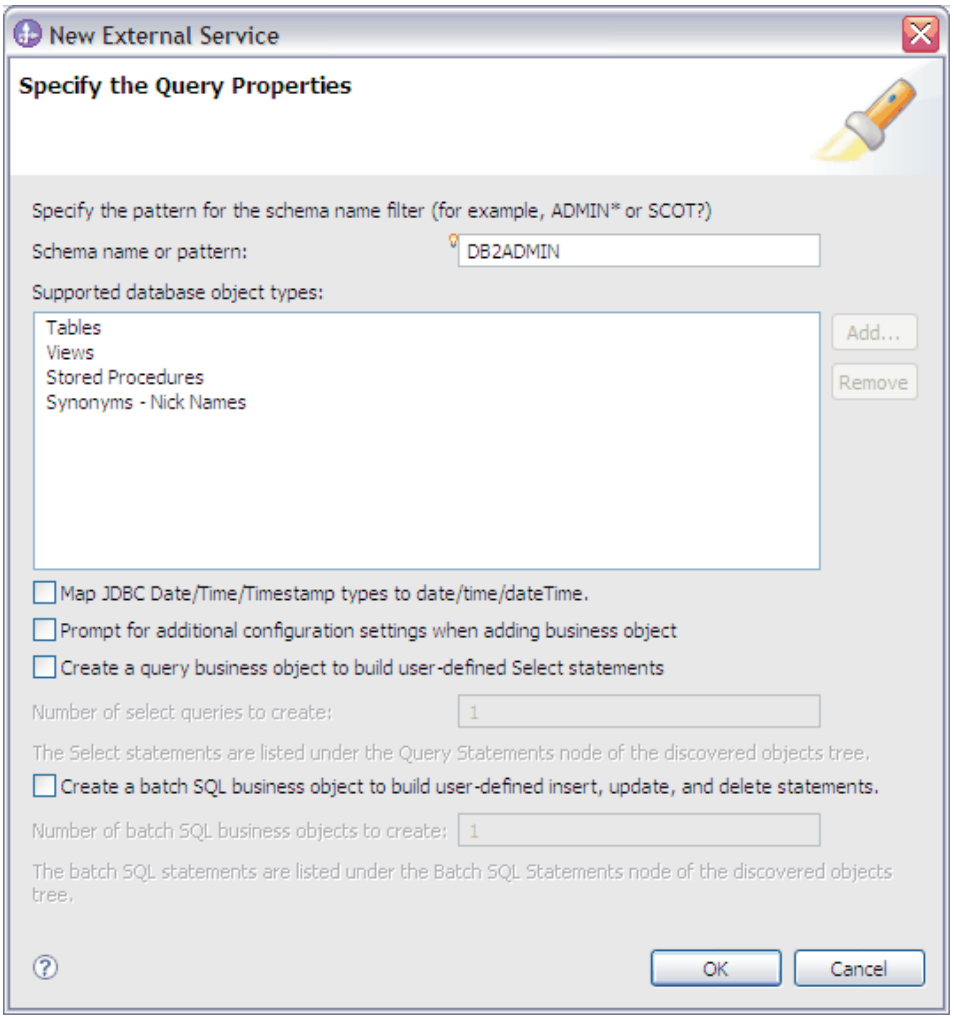
- モジュールはどのスキーマにアクセスする必要があるか
- これらのスキーマ内のどのタイプのデータベース・オブジェクトにアクセスする必要があるか
- どのテーブル、ビュー、シノニムまたはニックネーム、ストアード・プロシージャーまたはストアード関数にアクセスする必要があるか
- 作成する必要がある照会ビジネス・オブジェクトおよびバッチ SQL ビジネス・オブジェクトの数。パラメーター値およびそのパラメーターのサンプル・データベース値を含む

このタスクについて

この作業は、外部サービス・ウィザードの「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで開始します。

手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで、「**照会の編集 (Edit Query)**」をクリックします。 照会プロパティーの指定ウィンドウが表示されます。



「照会プロパティーの指定」ウィンドウを使用して、次のタスクを実行します。

- データベース・スキーマのサブセットを検索することにより、検索時間を削減します。
- 1 つ以上のタイプのデータベース・オブジェクトを検索から除外します。
- データベース内の情報からは自動的に判別できないアプリケーション固有情報の入力をウィザードがユーザーに求めるようにします。
- 作成する照会ビジネス・オブジェクトおよびバッチ SQL ビジネス・オブジェクトの数を指定します。
- JDBC のデータ型 Date、Time、および Timestamp を date、time、および dateTime にマップします。

注: バージョン 6.2.x、フィックスパック 2 では、作成したいラッパー・ビジネス・オブジェクトの数もこのウィンドウで指定できました。バージョン 7.0 では、ウィザードは後で wrapper 情報の入力を求めます。

2. 取得されるデータベース・スキーマの数を制限するには、**「スキーマ名またはパターン」**にスキーマの名前またはスキーマ名パターンを入力します。1 文字に対応させる場合は疑問符または下線 (? または _) を使用し、複数文字に対応

させる場合はアスタリスクまたはパーセント記号 (* または %) を使用します。 照会を実行すると、そのストリングで始まるスキーマか、またはスキーマ名パターンに一致するスキーマのみが表示されます。スキーマ名パターンを指定しない場合は、データベース内のすべてのスキーマが表示されます。データベースに多数のスキーマがある場合は、フィルターを使用してディスカバリー・プロセスを高速化できます。

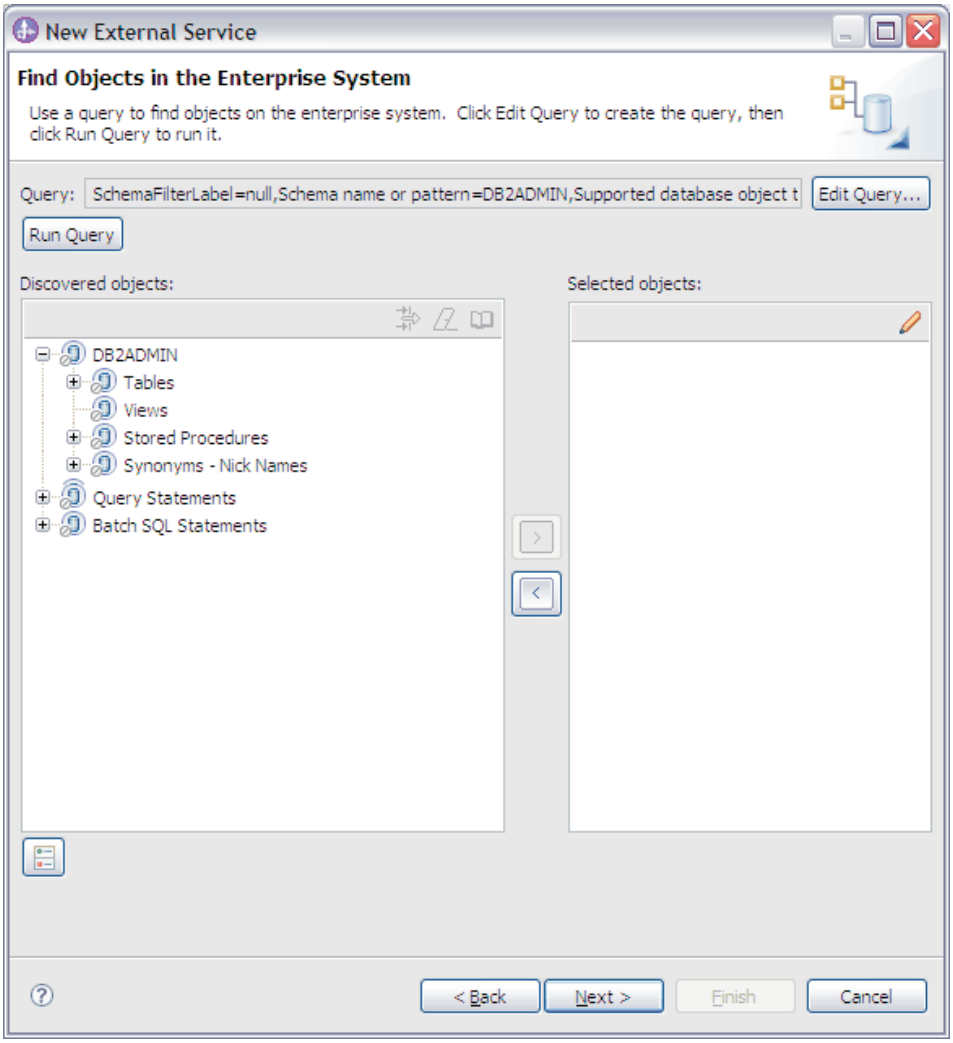
3. 検索から 1 つ以上のタイプのオブジェクトを除外するには、除外したいオブジェクトのタイプ (テーブル、ビュー、ストアード・プロシージャーおよびストアード関数、シノニムまたはニックネーム) を「**サポートされるデータベース・オブジェクト・タイプ**」フィールドで選択して、「**除去**」をクリックします。そのオブジェクト・タイプをもう一度追加するには、「**追加**」をクリックします。 特定のタイプのデータベース・オブジェクトにのみアクセスする必要がある場合は、必要のないオブジェクトを除外することで、ディスカバリー・プロセスを高速化できます。
4. データ型が Date、Time、および Timestamp であるテーブル、ストアード・プロシージャー、およびストアード関数の各オブジェクトは、デフォルトでは String データ型にマップされます。これらのオブジェクトを、JDBC ドライバーでサポートされる実際のデータ型 (Date、Time、Datetime など) にマップするには、「**JDBC の Date/Time/Timestamp の型を date/time/dateTime にマップ**」チェック・ボックスを選択します。

   注: デフォルトのデータ型マッピングは、JDBC ドライバーのバージョンによって異なります。例えば、Oracle JDBC ドライバーを使用する場合、Date データ型は、Date ではなく dateTime データ型にマップされます。このような場合は、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウで、適切なデータ型を手動で選択する必要があります。
5. 「**ビジネス・オブジェクトの追加時に追加構成設定のプロンプトを出す**」チェック・ボックスを選択します。 これにより、作成するビジネス・オブジェクトのリストにデータベース・オブジェクトを追加すると、そのオブジェクトに関してユーザーが構成可能なすべてのアプリケーション固有情報を入力するよう求められます。例えば、このオプションを選択した場合は、ビジネス・オブジェクトの単純親子階層を作成するプロセスが、順を追ってウィザードで示されます。 2 つの異なるテーブルの属性を参照する 2 つの属性をテーブル・ビジネス・オブジェクトが持つ (すなわち、2 つの親ビジネス・オブジェクトを持つ) 階層が必要な場合は、アセンブリー・エディターで構成を実行します。このエディターは、IBM Integration Designer から起動されるツールです。また、データベースで外部キー参照が定義されている場合は、アダプターはテーブル間の親子関係を自動的にディスカバーして表示します。

   重要: このオプションを選択しない場合、ウィザードは必須情報のみを入力するようプロンプトを出します。アセンブリー・エディターを使用して、ビジネス・オブジェクトの構成を実行する必要があります。また、データベースで外部キー参照を定義していない場合、アダプターは親子関係を自動的には生成しません。
6. ユーザー定義データベース照会を実行するビジネス・オブジェクトを作成するには、「**ユーザー定義の Select ステートメントを作成するための照会ビジネス・オブジェクトを作成する**」を選択してから、作成する照会ビジネス・オブ

ジェクトの数を入力します。 ここではビジネス・オブジェクトの数のみを指定します。このビジネス・オブジェクトについては、後で名前およびその他の詳細の入力をウィザードから求められます。

7. 一連の SQL ステートメントを実行するビジネス・オブジェクトを作成するには、「**ユーザー定義の insert、update、および delete ステートメントを作成するためのバッチ SQL ビジネス・オブジェクトを作成する**」を選択して、作成するバッチ SQL ビジネス・オブジェクトの数を入力します。 ここではビジネス・オブジェクトの数のみを指定します。このビジネス・オブジェクトについては、後で名前およびその他の詳細の入力をウィザードから求められます。
8. 「**OK**」をクリックして、データベース照会への変更を保存します。
9. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで、「**照会の実行 (Run Query)**」をクリックします。これによりこの照会を使用してデータベース・オブジェクトがディスカバーされ、照会およびバッチ SQL ビジネス・オブジェクトのテンプレートが作成されます。 標準的な照会の実行結果を次の図に示します。



注: 有効期限が切れたデータベース接続を復元するには、外部サービス・ウィザードを再始動します。

「**ディスカバーされたオブジェクト**」ペインには、ディスカバーされたデータベース・オブジェクトがリストされます。

10. 「**ディスカバーされたオブジェクト**」リストで、スキーマ・ノードおよびその下の「**テーブル**」、「**ビュー (Views)**」、「**ストアード・プロシージャー (Stored Procedures)**」、「**シノニム - ニックネーム (Synonyms - Nicknames)**」の各ノードを展開するには、「**+**」 (正符号) をクリックします。これにより、ウィザードによってディスカバーされたデータベース・オブジェクトが表示されます。
11. 「**照会ステートメント (Query Statements)**」および「**バッチ SQL ステートメント (Batch SQL Statements)**」のノードを展開するには、「**+**」 (正符号) をクリックします。これによって、照会ビジネス・オブジェクトおよびバッチ SQL ビジネス・オブジェクトのテンプレートが表示されます。

タスクの結果

アダプター、および照会ビジネス・オブジェクトとバッチ SQL ビジネス・オブジェクトのビジネス・オブジェクト・テンプレートを使用してアクセスできる、データベース・オブジェクトがウィザードによって表示されました。

次のタスク

外部サービス・ウィザードでの作業を続行します。次の手順では、モジュールで使用するオブジェクトの選択、各ビジネス・オブジェクトの構成、およびビジネス・オブジェクトの階層の作成を行います。

ビジネス・オブジェクトの選択および構成

外部サービス・ウィザード でディスカバーされたデータベース・オブジェクトのリストと、指定した照会オブジェクト・テンプレートおよびバッチ SQL オブジェクト・テンプレートを使用することにより、引き続きこのウィザードを使用して、モジュールでアクセスする必要のあるデータベース・オブジェクトを選択します。次に、新規ビジネス・オブジェクトの構成情報を入力します。

このタスクについて

「エンタープライズ・システムでのオブジェクトの検索」ウィンドウでは、オブジェクトを任意の順序で選択して構成できます。ただし、例外があり、親テーブルを選択して構成した後で、その子テーブルを選択して構成する必要があります。この制限を除けば、オブジェクトを個々に追加することも、複数のオブジェクトを一度に追加することも柔軟に行えます。「**ディスカバーされたオブジェクト**」リストのさまざまなノード内のオブジェクトを組み合わせることができます。例えば、テーブル・オブジェクト数個と、ストアード・プロシージャー・オブジェクト、および照会ステートメントを選択して、それらを同時に追加および構成することができます。

ビジネス・オブジェクトの選択および構成のおおまかな流れは以下のとおりです。

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの「**検出オブジェクト (Discovered objects)**」リストで 1 つ以上のオブジェクトを選択します。
2. 「**>**」 (追加) ボタンをクリックします。

3. 「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが開きます。
   - 1 つのオブジェクトを選択すると、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが 1 つ表示されます。

     このウィンドウで、ユーザーが構成可能な属性や、ウィザードによるデータベース検査ではディスカバーできないその他の情報をすべて入力したら、**「OK」**をクリックして構成を保存します。
   - 複数のオブジェクトを選択すると、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが、選択したオブジェクトごとに 1 ページずつ表示されます。

     各オブジェクトの名前を順にクリックします。ウィンドウには、該当するオブジェクトを個別に選択した場合と同じ情報が表示されます。

     重要: すべてのオブジェクトの構成ページで操作が完了するまでは、**「OK」**はクリックしないでください。ウィザードは、すべての必須フィールドに値が指定されるまではノートブックを閉じません。ただし、ユーザーは、オプション・フィールドに値を指定する前にウィンドウを閉じることができます。ウィザードでオプション・フィールドを構成しない場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後にそれらのフィールドを構成する必要があります。
4. ウィザードにより、構成されたオブジェクトが**「選択済みオブジェクト (Selected objects)」**リストに追加されます。

ウィザードを終了しない限り、操作を繰り返してモジュールに必要なビジネス・オブジェクトを選択および構成できます。ただし、ウィザードを開始して既存のモジュールにオブジェクトを追加する前に、ビジネス・オブジェクトを使用するプログラムの要件を十分に理解してください。ウィザードによって、同じパス内の既存のビジネス・オブジェクトが上書きされます。

Outbound 処理のテーブル、ビュー、およびシノニムまたはニックネームの選択および構成:

モジュールで使用するテーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを選択して構成するには、ビジネス・オブジェクトの構成プロパティーを指定します。

始める前に

このタスクを実行するには、データベース内のデータの構造や、モジュールがどんなデータベース・オブジェクトにアクセスする必要があるかを理解しなければなりません。具体的には、以下の情報を認識しておく必要があります。

- テーブル、ビュー、およびシノニムまたはニックネームの構造 (必要な列や、データ型などの列属性を含む)。
- テーブル間の関係 (親子関係のカーディナリティーおよび所有権を含む)。

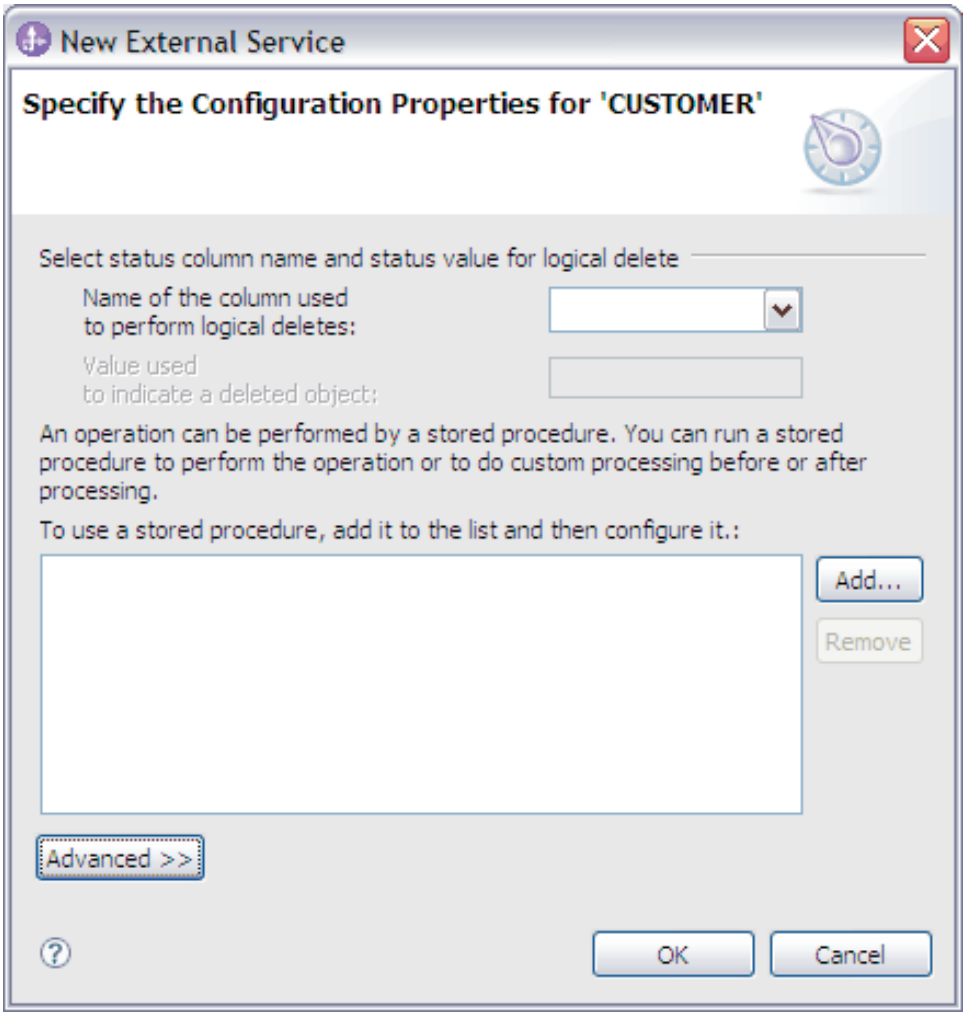
このタスクについて

このタスクは、外部サービス・ウィザードを介して実行されます。「エンタープライズ・システムでのオブジェクトの検索」ウィンドウから操作を開始し、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

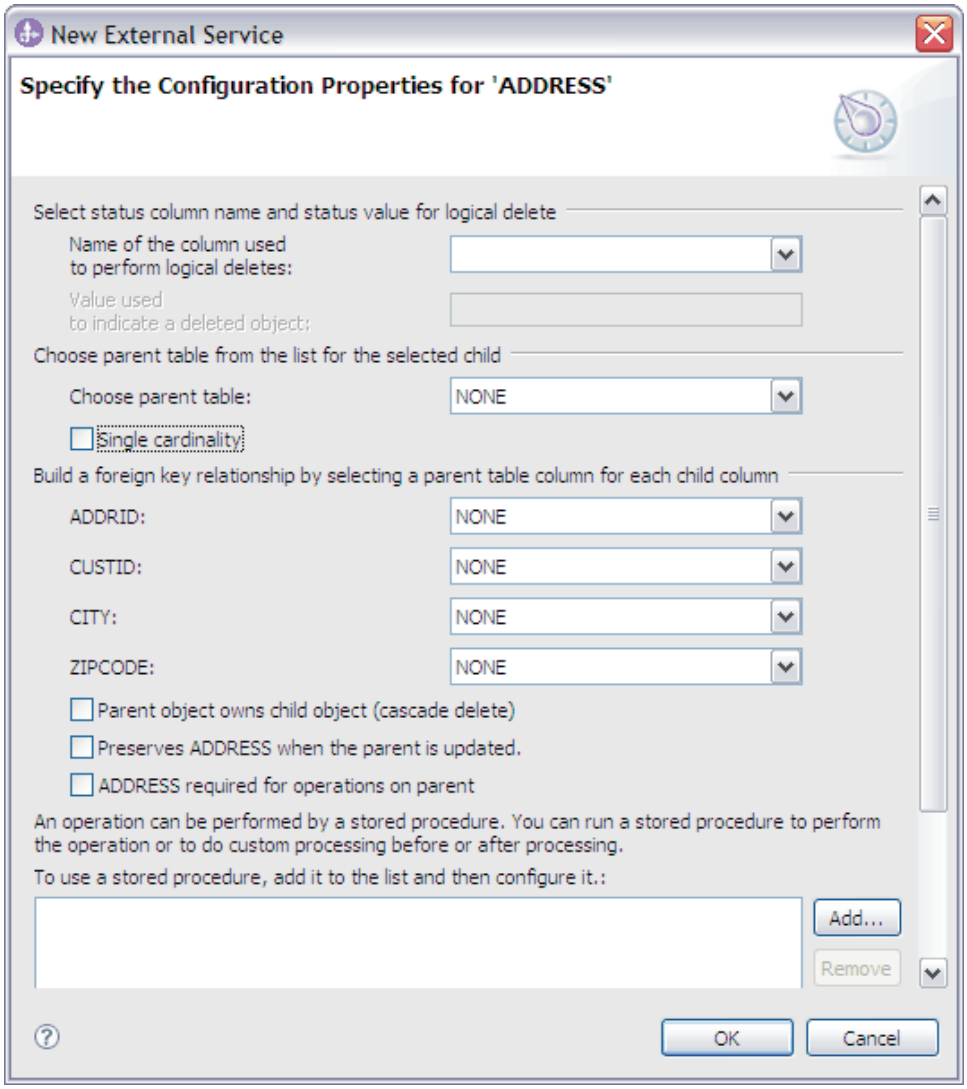
手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの「**ディスカバーされたオブジェクト**」リストで、テーブル、ビュー、またはシノニムを 1 つ以上選択し、「**>**」 (追加) ボタンをクリックします。オブジェクトが「**選択済みオブジェクト**」リストに追加されます。

   以下の 2 つの図に、ビジネス・オブジェクト (テーブル、ビュー、シノニム、またはニックネーム) の標準的な「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウを示します。最初の図に、選択する最初のテーブルまたはテーブル・グループの標準的なウィンドウを示します。



以下の図に、選択する後続のテーブルの標準的なウィンドウを示します。少なくとも 1 つのテーブルを選択して構成した後は、後続テーブルの「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウに、テーブル間の親子階層をオプションで定義することができる領域が表示されます。



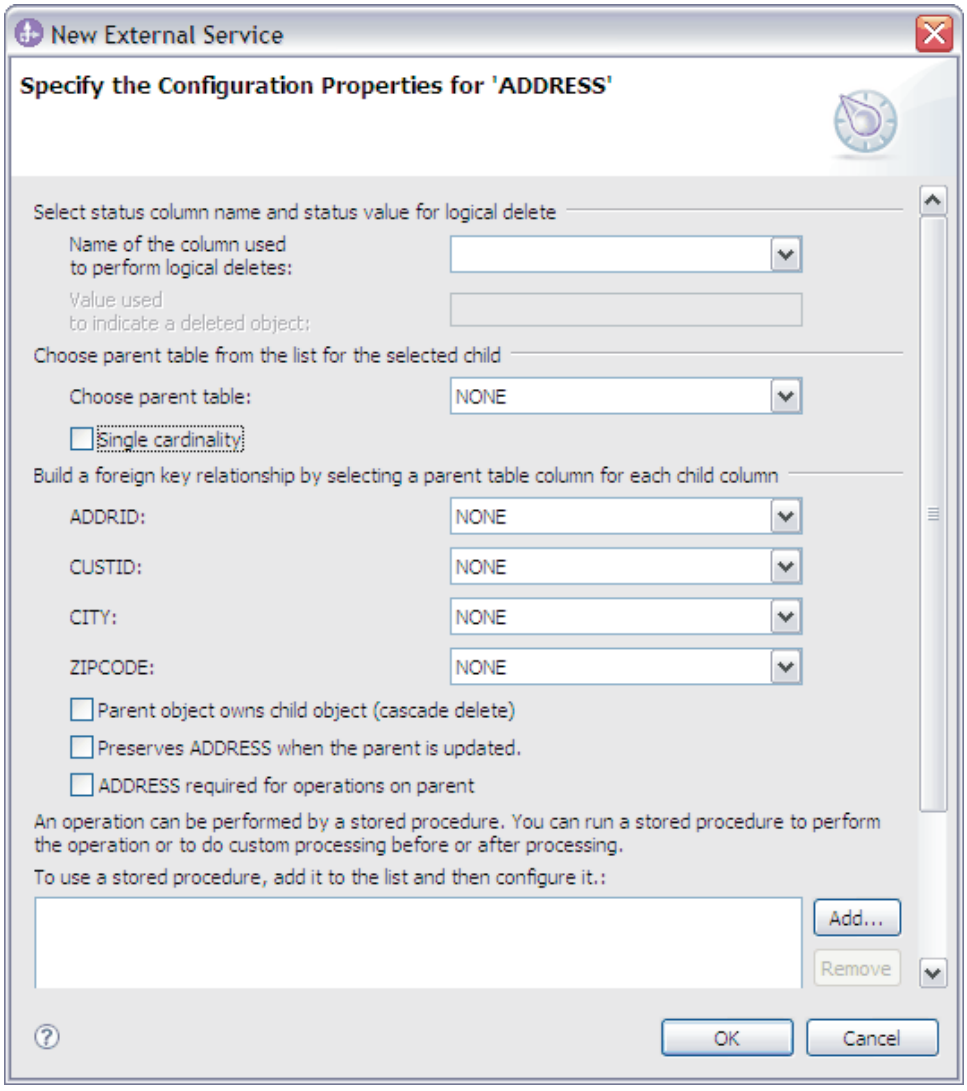
オブジェクトの構成時には、拡張構成を必要とする選択を行うと、このウィンドウに追加のフィールドが表示され、ウィンドウがスクロールされる場合があります。必ずウィンドウのすべてのフィールドを確認してから、「**OK**」をクリックしてください。

2. 論理削除を示すのに使用される列がテーブルにある場合は、次の手順に従います。
   a. 「**論理削除を実行するのに使用される列の名前**」フィールドで列名を選択します。
   b. 「**削除されたオブジェクトを示すために使用する値**」フィールドに、行が論理的に削除されていることを示す値を入力します。 この値については、データベース管理者に確認できます。

3. 「**テーブル *table_name* の基本キーの選択**」エリアが表示されたら、「**追加**」をクリックし、テーブル・ビジネス・オブジェクトの基本キーとして使用する列を選択してから、「**OK**」をクリックします。テーブルに複合キーがある場合は、複数の列を選択できます。 「**テーブル *table_name* の基本キーの選択**」エリアは、データベース表に基本キーとして指定された列が存在しない場合にのみ表示されます。各テーブル・ビジネス・オブジェクトには、関連付けられたデータベース表にキーがない場合でも、基本キーが定義されている必要があります。データベースで基本キーが定義されている場合、ウィンドウのこのセクションは表示されません。
4. オプション: ビジネス・オブジェクト間の親子関係を定義します。

   親子階層を作成する場合、まず親テーブルを構成して「エンタープライズ・システムでのオブジェクトの検索」ウィンドウに戻り、子テーブルを選択して構成します。

   以下の図に示す「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウの領域を使用して、親子関係を構成します。これらのフィールドは、構成する最初のテーブルの場合には表示されません。



a. 「**親テーブルの選択**」フィールドで、構成する親テーブルの名前を選択します。 リストに親テーブルが表示されていない場合は、親テーブルがまだ構成されていません。戻って親オブジェクトを構成してから、子オブジェクトを構成してください。データベースで外部キー参照を定義した場合、親テーブルを選択すると、アダプターはテーブル間の親子関係を自動的にディスカバーして表示します。テーブルとその親のテーブル間に単一カーディナリティー関係がある場合は、「**単一カーディナリティー**」チェック・ボックスが自動的に選択されます。

b. 関係のカーディナリティーを指定します。

- テーブルとその親テーブルの間に単一カーディナリティー関係がある場合は、「**単一カーディナリティー**」チェック・ボックスを選択します。単一カーディナリティー関係では、親はこのタイプの子ビジネス・オブジェクトを 1 つのみ持つことができます。単一カーディナリティー関係は、所有関係を伴って実際の子を表すか、または所有関係を伴わずにルックアップ・テーブルまたはデータベース内の他の対等オブジェクトを表すために使用できます。

- テーブルに複数カーディナリティー関係がある場合は、「**単一カーディナリティー**」チェック・ボックスを選択しないでください。複数カーディナリティー関係では、親がこのタイプの子ビジネス・オブジェクトの配列を持つことができます。

c. 親と子の間に外部キー関係を作成するため、子の列ごとに、親テーブルの外部キーであるかどうかを指定します。
   - 子の列が外部キーでない場合は、「なし」を選択します。
   - 子の列が外部キーの場合は、その子の列に対応する親テーブルの列を選択します。

     注: ウィザードは、1 つの親テーブルのみを構成できます。子テーブルに複数の親テーブルがある場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後に残りの親テーブルを構成する必要があります。

d. 親オブジェクトが子オブジェクトを所有している場合、データベース内の子オブジェクトは親が削除されるときに削除されます。この子がその親によって所有されていることを示すには、「**親オブジェクトが子オブジェクトを所有する (カスケード削除)**」チェック・ボックスを選択します。あるいは、このオプションをクリアして、ルックアップ・テーブルなどの子オブジェクトが、親の削除時に削除されないようにします。

e. Update 操作の一環として子オブジェクトが削除されることのないようにするには、「**親の更新時に *child_table_name* を保持する**」チェック・ボックスを選択します。

   親テーブルが更新されると、アダプターは入力に存在する子ビジネス・オブジェクトを、データベースから返される子ビジネス・オブジェクトと比較します。デフォルトでは、アダプターは、入力ビジネス・オブジェクト内に存在しない、データベースから返されたすべての子オブジェクトを削除します。

f. デフォルトでは、子ビジネス・オブジェクトを指定せずに、親ビジネス・オブジェクトに対して操作を実行できます。親ビジネス・オブジェクトを変更対象として実行依頼するときに、その親ビジネス・オブジェクトで子ビジネス・オブジェクトが必ず指定されるようにしたい場合は、「***Child_table_name* は、親に対する操作で必須**」チェック・ボックスを選択します。

5. 操作を実行するには、アダプターによって生成される標準 SQL ステートメントを使用するか、あるいはデータベース内のストアード・プロシージャーまたはストアード関数を使用します。ストアード・プロシージャーまたはストアード関数を使用する場合は、以下の手順を実行します。

   a. 「**追加**」をクリックします。

   b. 「追加」ウィンドウで、実行するストアード・プロシージャーのタイプを選択します。 操作ごとに、その操作を実行するストアード・プロシージャーと、操作の前後に実行するストアード・プロシージャーを選択できます。例えば、Create 操作の場合は、ストアード・プロシージャー CreateSP、BeforeCreateSP、および AfterCreateSP のどれでも指定できます。

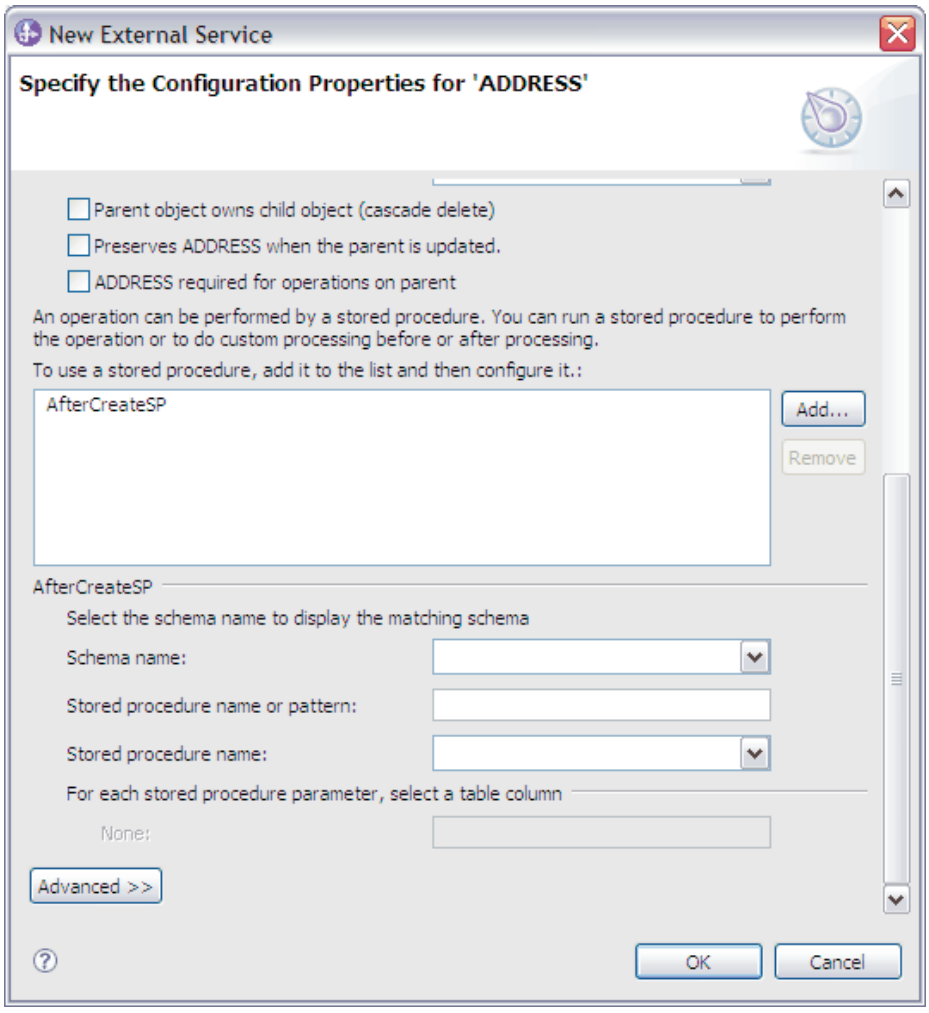
      注: RetrieveAllSP を指定してテーブルを構成する場合は、ストアード・プロシージャーが 1 つの結果セットのみを返すことを確認します。ストアード・

プロシージャーの ResultSet ASI を true に設定して、実行時に例外「ストアード・プロシージャーに関連した結果セットが見つかりませんでした (No resultset found associated with the stored procedure)」、 「結果セットが返されませんでした (No resultset returned)」、 「複数の結果セットが返されました (More than one resultset returned)」のいずれも生成されないようにします。

注:

1) Oracle データベースの場合、WebSphere Adapter for JDBC は、 Cursor タイプの OUT パラメーターを使用するストアード・プロシージャーのみをサポートします。Cursor タイプの IN または INOUT パラメーターを使用するストアード・プロシージャーはサポートしません。
2) DB2 データベースおよび MSSQLServer データベースの場合、WebSphere Adapter for JDBC は、 Cursor タイプの IN、OUT、および INOUT パラメーターを使用するストアード・プロシージャーをサポートしません。

c. 「**OK**」をクリックします。選択したストアード・プロシージャーのタイプが「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウに表示されます。このウィンドウは、各ストアード・プロシージャーの構成用の領域を表示するために拡張されます。新しい領域を表示するには、スクロールダウンする必要がある場合もあります。



注: 階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、トップレベルのビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーをトップレベルのビジネス・オブジェクトに関連付けても、各子ビジネス・オブジェクトに関連付けないと、そのトップレベルのビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

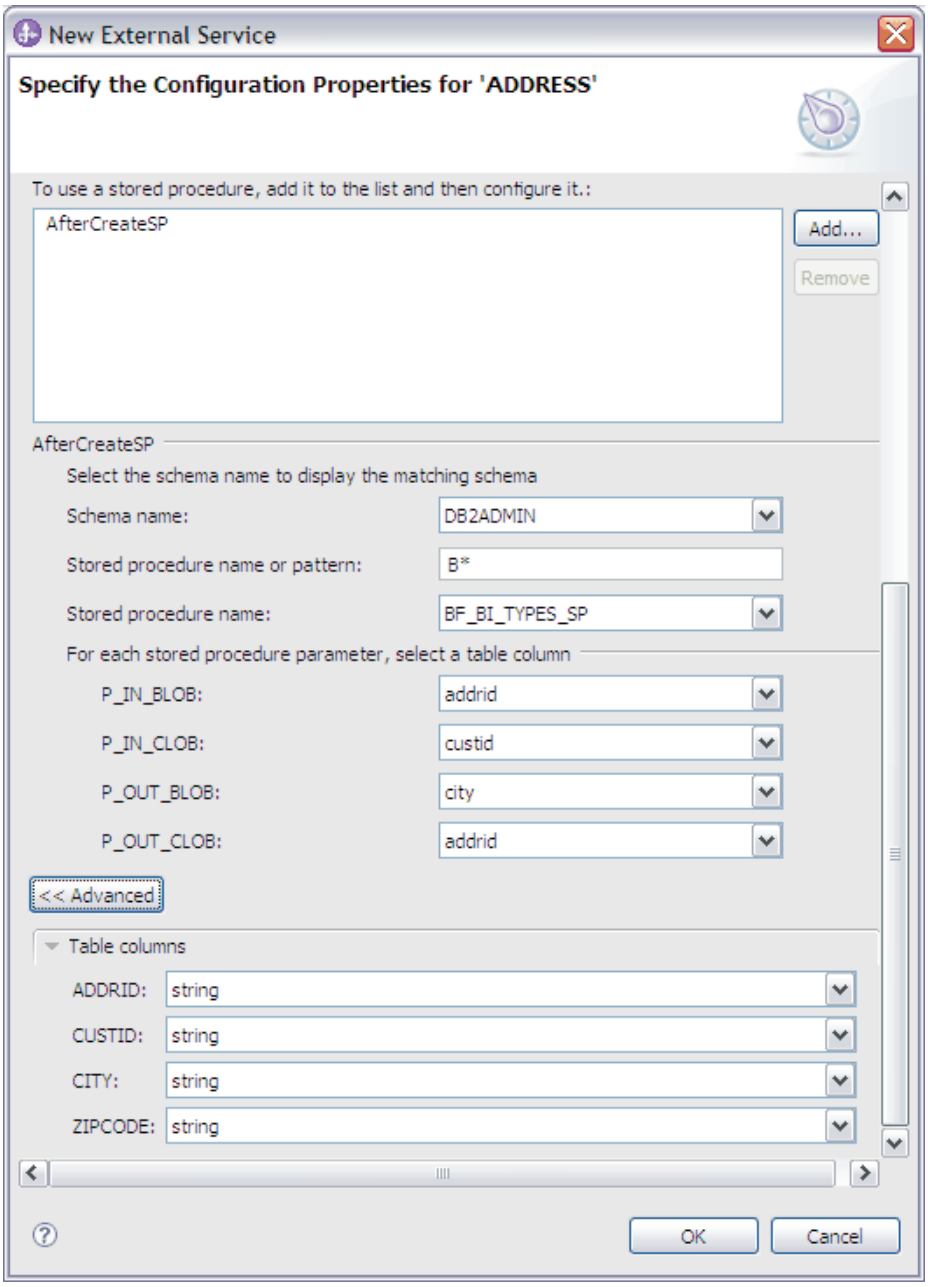
6. 選択したストアード・プロシージャー・タイプごとに、データベース内でのストアード・プロシージャーの名前を指定し、ビジネス・オブジェクトを構成します。
   a. 「**スキーマ名**」フィールドで、ストアード・プロシージャーが含まれるスキーマの名前を選択します。
   b. ストアード・プロシージャーまたはストアード関数の名前を指定します。
      1) 「**ストアード・プロシージャー名またはパターン**」フィールドで、ストアード・プロシージャーまたはストアード関数の名前を入力するか、または名前パターンを入力します。1 つの文字と一致させる場合は疑問符または

下線 (? または _) を使用し、複数の文字と一致させる場合はアスタリスクまたはパーセント記号 (* または %) を使用します。

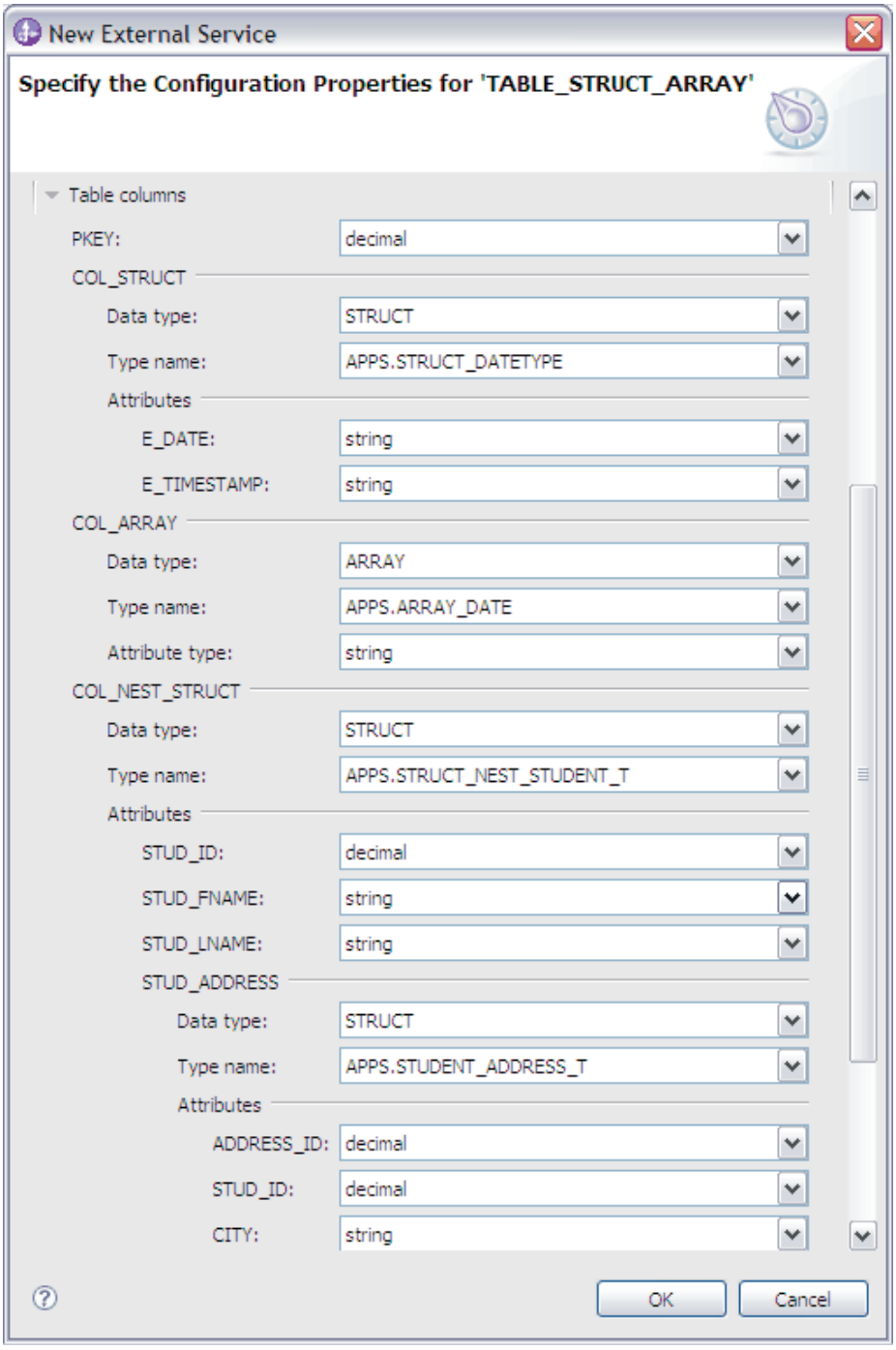
2) 「**ストアード・プロシージャー名**」フィールドで、目的のプロシージャーの名前を選択します。

「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが拡張して、ストアード・プロシージャーを構成するための領域が表示されます。ウィザードは、データベース内のストアード・プロシージャーを調べることにより、パラメーターのリストを自動生成します。

c. ストアード・プロシージャーのパラメーターごと (左側) に、そのパラメーターでストアード・プロシージャーに渡すテーブル列 (右側) を選択します。次の図に、ストアード・プロシージャーを構成した後のウィンドウの一部を示します。



7. テーブル内の各列のデータ型マッピングを指定するには、以下の手順を実行します。
   a. 「**拡張**」をクリックします。
   b. 「**テーブル列**」を展開します。テーブル内の列ごとに、デフォルトのデータ型マッピングが表示されます。 Oracle データベースにおいて、配列、構造体、ネストされた構造体、テーブルなど、何らかのユーザー定義型または複合データ型がテーブルに含まれている場合は、型名および子属性の詳細も自動的にディスカバーされて、表示されます。次の図に、複合データ型を含む Oracle テーブルの型名および子属性の詳細を示します。



c. マッピングを確認して、必要な場合は変更します。

8. ウィンドウのすべてのフィールドの操作が完了したら、「**OK**」をクリックします。ビジネス・オブジェクトの構成が保存されます。 定義したビジネス・オブジェクト (テーブル、ビュー、シノニム、およびニックネーム) が、「エンタープライズ・システムでのオブジェクトの検索」ウィンドウにリストされます。
9. 「**選択済みオブジェクト**」リストのオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。

次のタスク

エンタープライズ・システムでのオブジェクトの検索ウィンドウで、他のタイプのビジネス・オブジェクトの選択と構成を続行します。完了したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

ストアード・プロシージャーおよびストアード関数の選択および構成:

データベース内のストアード・プロシージャーおよびストアード関数に対応するビジネス・オブジェクトを選択および構成します。

始める前に

ストアード・プロシージャーまたはストアード関数のビジネス・オブジェクトを選択して構成するには、データベース内のデータの構造や、モジュールがどのオブジェクトにアクセスする必要があるかを理解しなければなりません。特に、モジュールがアクセスする必要があるストアード・プロシージャーまたはストアード関数に渡すパラメーターについて知る必要があります。

このタスクについて

このタスクは、外部サービス・ウィザードを介して実行されます。「エンタープライズ・システムでのオブジェクトの検索」ウィンドウから操作を開始し、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの「**ディスカバーされたオブジェクト**」リストで、必要なストアード・プロシージャーまたはストアード関数を含むスキーマのノードを展開してから、「**ストアード・プロシージャー (Stored Procedures)**」 ノードを展開します。
2. 「フィルター・プロパティー」ウィンドウの 1 つ以上のフィルター・フィールドに有効な名前またはパターンを指定して、ストアード・プロシージャーをフィルタリングします。
   a. 「**ストアード・プロシージャー (Stored Procedures)**」をクリックして、「**ディスカバーされたオブジェクト**」ペインの上部にある  (フィルターの作成または編集) ボタンをクリックします。
   b. 「フィルター・プロパティー」ウィンドウで、「**オブジェクト名またはパターン**」フィールド内に名前または文字パターンを入力します。 1 つの文字と一致させる場合は疑問符 (?) または下線 (_) を使用し、複数の文字と一致させる場合はアスタリスク (*) またはパーセント (%) を使用します。 名前には、大/小文字の区別があります。
   c. 「**カタログ名またはパターン**」フィールドに、名前またはパターンを入力します。 1 つの文字と一致させる場合は疑問符 (?) または下線 (_) を使用し、複数の文字と一致させる場合はアスタリスク (*) またはパーセント (%) を使用します。

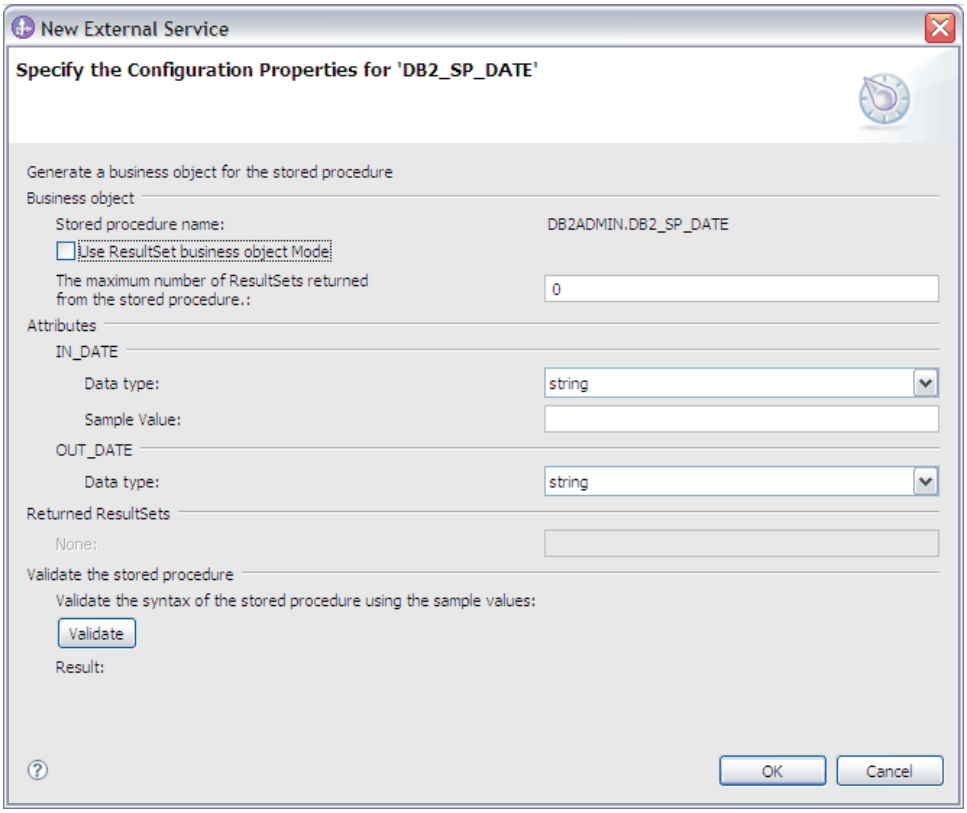
d. 「**OK**」をクリックします。

3. 「**ストアード・プロシージャー (Stored Procedures)**」リストからオブジェクトを 1 つ以上選択し、「**>**」 (追加) ボタンをクリックして、「**選択済みオブジェクト**」リストにオブジェクトを追加します。

   PL/SQL パッケージで定義されているストアード・プロシージャーが、*SPName*(*PackageName*) の形式で表示されます。例えば、EMP_MGMT パッケージに CREATE_DEPT ストアード・プロシージャーが含まれている場合、このストアード・プロシージャーは CREATE_DEPT(EMP_MGMT) としてリストに表示されます。

   注: データベース・スキーマに、同じ名前を持つ複数のストアード・プロシージャーがあり、かつ、それらのストアード・プロシージャーが異なるパラメーターを持っている場合、外部サービス・ウィザードは、どのストアード・プロシージャーが選択されるかを識別できません。したがって、ストアード・プロシージャーは固有の名前を持つ必要があります。
   「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウに、ストアード・プロシージャー・ビジネス・オブジェクトの属性がリストされます。このリストには、ストアード・プロシージャーのパラメーターの名前とデータ型、および返される結果セットに関する情報が含まれます。

   注: DB2 データベースの場合、テーブルに構造化データ型が含まれていると、アダプターはその構造化型の列名のみを表示します。アダプターは型の詳細を取得することはできません。



4. ストアード・プロシージャーから返された結果セットをビジネス・オブジェクト・モードで処理するために、「**結果セット・ビジネス・オブジェクト・モードを使用**」チェック・ボックスを選択します。このモードでは、アダプターは、外部サービス・ウィザードから返された結果セットの数や順序を認識する必要がありません。

   注: 「**結果セット・ビジネス・オブジェクト・モードを使用**」チェック・ボックスを選択すると、「**ストアード・プロシージャーから返される結果セットの最大数**」フィールドが使用できなくなります。

5. ストアード・プロシージャーが結果セットを返す場合は、「**ストアード・プロシージャーから返される結果セットの最大数**」フィールドの値が、期待される最大数を反映するようにしてください。ウィザードにより、結果を保持するのに必要な数の結果セット・ビジネス・オブジェクトが作成されます。

   注: Oracle データベースの場合は、ストアード・プロシージャーの構文を検証後、結果セットの数が正しいことを確認してください。Oracle ドライバーは、必ず情報を返すとは限りません。返された結果セットの数が正しくない場合は、検証後、「**OK**」をクリックしてウィンドウを終了する前に、数を設定してください。ウィザードを終了した後は、オプションで、ストアード・プロシージャー・ビジネス・オブジェクトの MaxNumOfRetRS アプリケーション固有情報の設定を検証できます。

   注:

   a. Oracle データベースの場合、WebSphere Adapter for JDBC は、 Cursor タイプの OUT パラメーターを使用するストアード・プロシージャーのみをサポートします。Cursor タイプの IN または INOUT パラメーターを使用するストアード・プロシージャーはサポートしません。

   b. DB2 データベースおよび MSSQLServer データベースの場合、WebSphere Adapter for JDBC は、 Cursor タイプの IN、OUT、および INOUT パラメーターを使用するストアード・プロシージャーをサポートしません。

6. 各パラメーターを構成します。

   a. 「**データ・タイプ**」フィールドに正しいデータ型が表示されていることを確認します。 標準 JDBC データ型を持つパラメーターのデータ型は、ウィザードによって自動的にディスカバーされます。他の特定の型については、手動でデータ型を選択する必要があります。

   b. 「**サンプル値**」フィールドに、有効な値を入力します。

      注: Oracle データベースの場合、ストアード・プロシージャーまたはストアード関数に、データ型が Date である Varray 属性または Object 属性のどちらかが含まれ、データ型のマッピングが Date から String である場合には、ストアード・プロシージャーまたはストアード関数の検証が正常に実行されるように、yyyy-mm-dd フォーマットではなく yyyy-mm-dd hh:mm:ss フォーマットでサンプル値を指定する必要があります。次の表に、各 Date データ型マッピングで準拠すべきフォーマットを示します。

表 17. Date データ型フォーマット

データ型マッピング	データ型が Date である SP/SF	データ型が Date である SP/SF の VArray/Object
Date から String	フォーマット: Day-Month-Year (例: '01-JAN-0001')	フォーマット: Year-Month-Day Hour:Minute:Second.[Millisecond] (例: '0001-01-01 01:00:00.000000000')
Date から Date	フォーマット: Year-Month-Day (例: '0001-01-01')	フォーマット: Year-Month-Day (例: '0001-01-01')

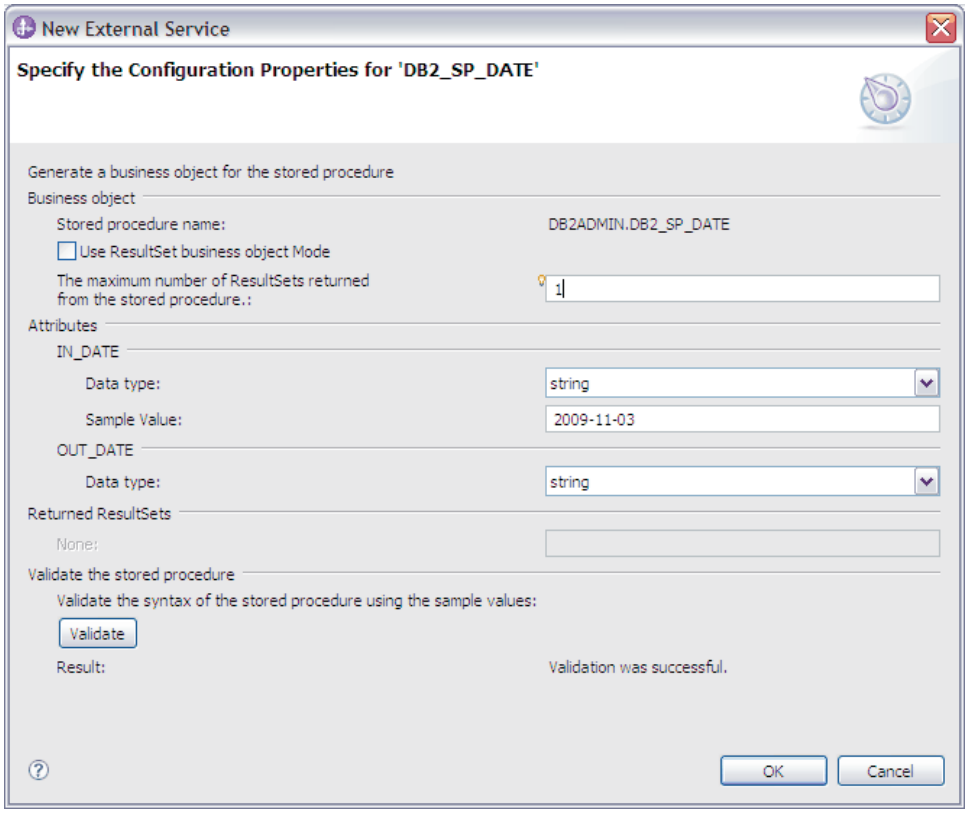
- Year パラメーターは、すべてのフォーマットで、最小 1 桁から最大 4 桁で指定できます。
- Month パラメーターは、すべてのフォーマットで、最小 1 桁から最大 2 桁で指定できます。ただし、データ型が Date である SP/SF の Date から String へのマッピングは例外で、この場合、Month パラメーターは、前述の表の例で指定されているように、3 文字の英字で指定します。
- Day、Hour、Minute、および Second の各パラメーターは、すべてのフォーマットで、最小 1 桁から最大 2 桁で指定できます。
- Milliseconds はオプション・パラメーターです。このパラメーターは、すべてのフォーマットで、最小 1 桁から最大 9 桁で指定できます。

7. サンプル値を使用してストアード・プロシージャーの構文を検証するには、「**検証**」をクリックします。 検証の結果が「**結果**」領域に表示されます。

   「**結果**」領域に「検証は失敗しました」のメッセージが表示された場合は、指定した情報に問題があります。「検証は失敗しました」メッセージの後に表示されるデータベース・サーバーからのエラー・メッセージを参考にして、定義を訂正します。パラメーターおよびサンプル・データのデータ型が正しいことを確認してください。

   ワークスペースの `.metadata` フォルダー内の `.log` ファイルには、問題に関する追加情報が含まれています。

   以下の図に、ストアード・プロシージャーが検証された後のウィンドウを示します。



メッセージ「検証は成功しました。」が表示された場合は、「**OK**」をクリックして、ストアード・プロシージャー・ビジネス・オブジェクトの定義を保存します。

重要: ストアード・プロシージャーまたはストアード関数が結果セットを返す場合は、検証が正常に終了するまで「**OK**」をクリックしないでください。ウィザードは、検証時に返された結果を使用して、その結果を保持するビジネス・オブジェクトを作成します。ストアード・プロシージャーが正常に検証されない場合、アダプターは実行時に結果セットを返しません。

8. 「**選択済みオブジェクト**」リストのオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。

タスクの結果

ストアード・プロシージャーおよびストアード関数として構成したビジネス・オブジェクトが「エンタープライズ・システムでのオブジェクトの検索」ウィンドウにリストされます。

次のタスク

エンタープライズ・システムでのオブジェクトの検索ウィンドウで、他のタイプのビジネス・オブジェクトの選択と構成を続行します。完了したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

バッチ SQL ビジネス・オブジェクトの選択および構成:

バッチ SQL ビジネス・オブジェクトを使用して、データベース操作を実行する一連の INSERT、UPDATE、および DELETE SQL ステートメントを定義します。

始める前に

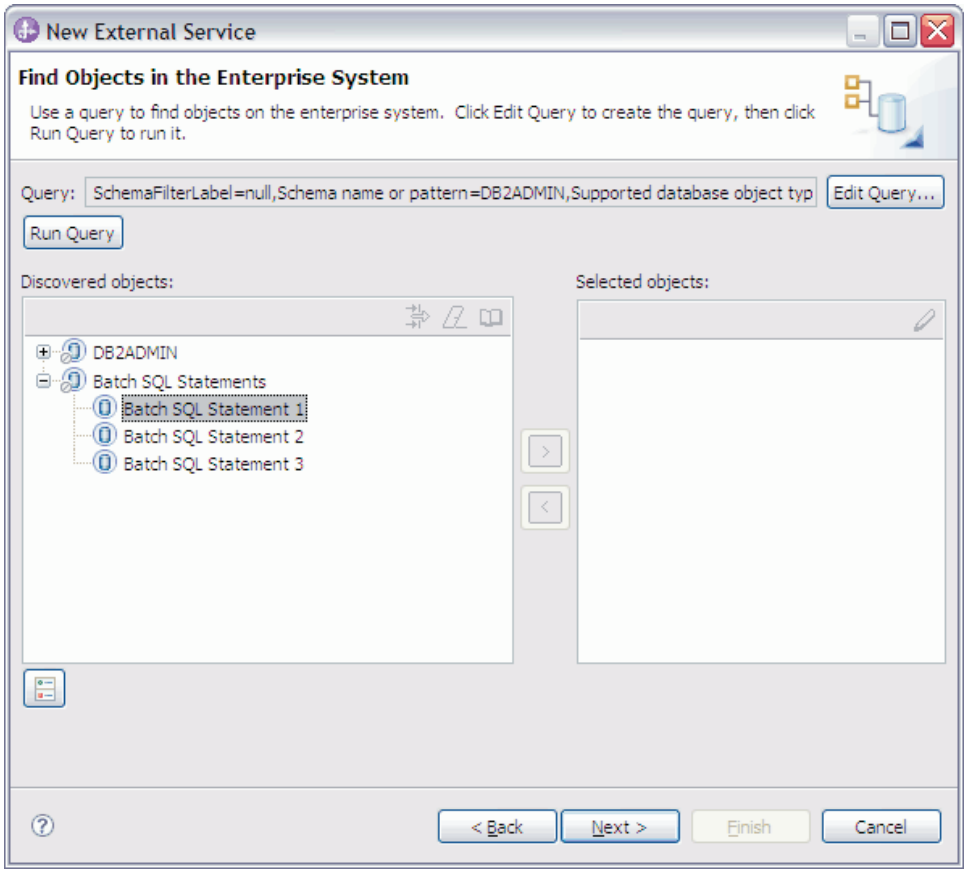
バッチ SQL ビジネス・オブジェクトを構成するには、テーブルおよびビューをはじめとする、データベース内のデータ構造がわかっていなければなりません。SQL ステートメントで処理する必要がある列の名前およびデータ型を把握しておく必要があります。さらに、SQL INSERT、UPDATE、および DELETE ステートメントを記述できなければなりません。

このタスクについて

このタスクは、外部サービス・ウィザードを介して実行されます。「エンタープライズ・システムでのオブジェクトの検索」ウィンドウから操作を開始し、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの 「**ディスカバーされたオブジェクト**」リストで、「**バッチ SQL ステートメント (Batch SQL Statements)**」ノードを展開します。 このノードには、「照会プロパティーの指定」ウィンドウで要求した各バッチ SQL ビジネス・オブジェクトのオブジェクト・テンプレート「**バッチ SQL ステートメント (Batch SQL Statement) *n***」があります。例えば、前述のウィンドウでバッチ SQL ビジネス・オブジェクト数を 3 と指定した場合、「**ディスカバーされたオブジェクト**」リストには 3 つのオブジェクト・テンプレートが表示されます (次の図を参照)。



2. オブジェクト・テンプレートを 1 つ以上選択し、「>」 (追加) ボタンをクリックして、オブジェクトを「**選択済みオブジェクト**」リストに追加します。
3. 「**バッチ SQL ビジネス・オブジェクト名**」フィールドに、ビジネス・オブジェクトの名前を入力します。 名前にスペースを使用することはできませんが、各国語文字は使用できます。
4. 「**SQL ステートメント**」フィールドに、1 つ以上の SQL INSERT、UPDATE、または DELETE ステートメントをセミコロン (**;**) で区切って入力します。ステートメント内の各パラメーターは疑問符 (**?**) で示します。 以下の例では、バッチ SQL ビジネス・オブジェクトの柔軟性が示されています。
   - insert into autoid (con1) values ('Smith')
   - insert into customer (pkey, fname, lname, ccode) values (?, ?, ?, 12345)
   - update customer set fname=?, lname=? where custid=? and ccode is null
   - delete from customer where ccode like ?
   - insert into customer (pkey,ccode,fname,lname) values (?,?,?,?); delete from customer where pkey=?
5. DB2 または Microsoft SQL データベースで、単一の INSERT ステートメントを指定する場合は、オプションで、シーケンスにより自動生成された固有 ID をアダプターが取得するようにできます。ID を取得するようにビジネス・オブジェクトを構成するには、「**生成された固有 ID の取得**」チェック・ボックスを選択して、その ID が格納されている列の名前を入力します。

このオプションは、単一の INSERT ステートメントを指定する場合で、かつ指定した列の ID を生成するようにデータベースが構成されている場合に限り有効です。

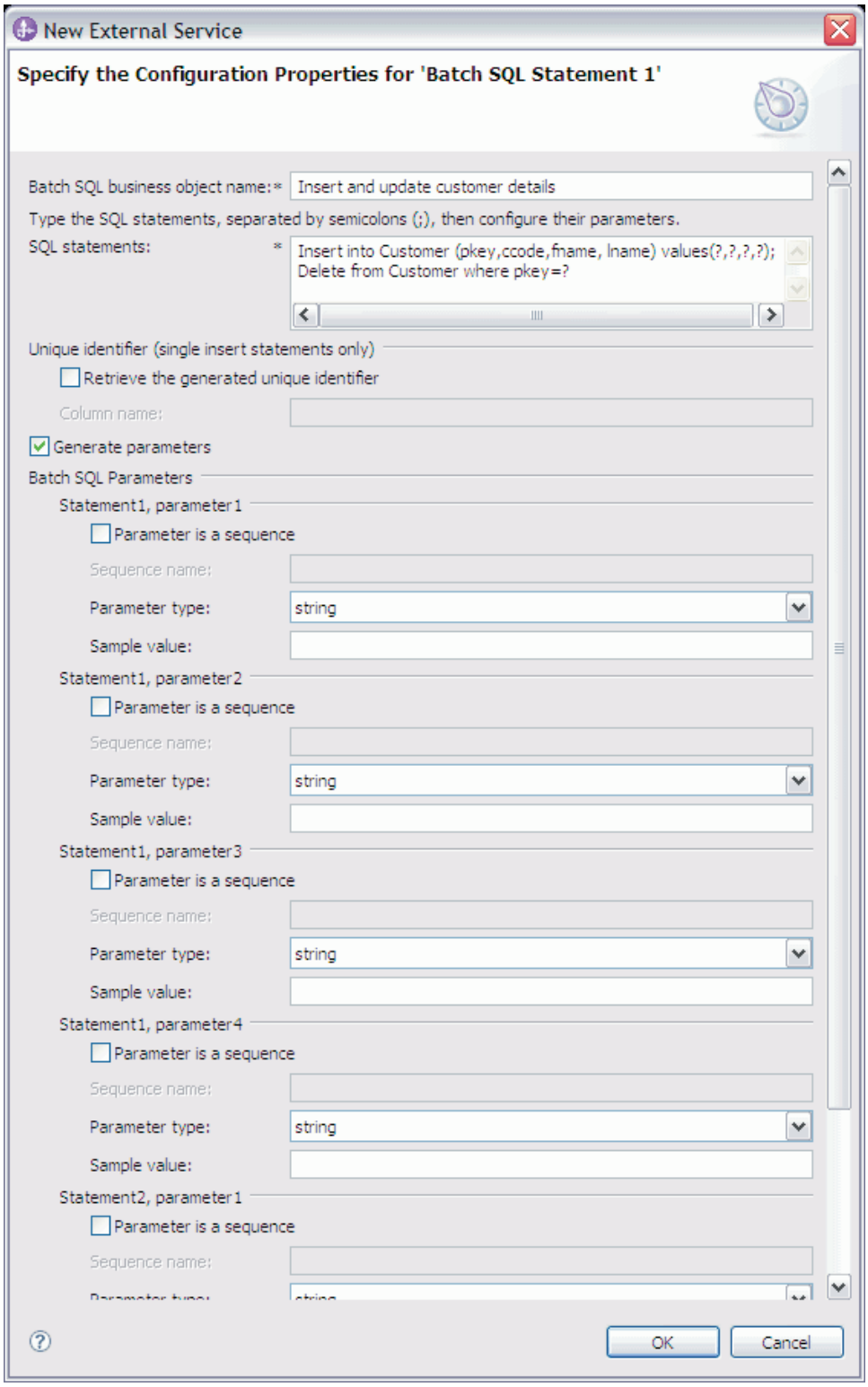
注: Oracle データベースは固有 ID の使用をサポートしないため、ご使用の構成で Oracle データベースを使用している場合は、「**生成された固有 ID の取得**」のチェック・ボックスは使用不可になります。

6. 「**パラメーターの生成**」チェック・ボックスを選択します。 ウィンドウが拡張されて、各パラメーターを定義するための領域が表示されます。これにより、ウィンドウがスクロールすることがあります。ウィンドウを拡張すると、見やすくなります。パラメーターを構成するための領域には、「**ステートメント 1、パラメーター 1 (Statement 1, parameter 1)**」、「**ステートメント *n*、パラメーター *m*(Statement n, parameter m)**」 などのラベルが付いています。

   例えば、以下の SQL ステートメントを指定して、「**パラメーターの生成**」をクリックするとします。`Insert into customer (pkey,ccode,fname,lname) values (?,?,?,?); Delete from Customer where pkey=?`

   「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')ウィンドウが拡張して、5 つのパラメーターが表示されます。最初のステートメント (`Insert`) には 4 つのパラメーターがあり、これらのパラメーターは「**ステートメント 1、パラメーター 1 (Statement 1, parameter 1)**」から「**ステートメント 1、パラメーター 4 (Statement 1, parameter 4)**」に対応します。2 番目のステートメント (`Delete`) には 1 つのパラメーターがあり、このパラメーターは「**ステートメント 2、パラメーター 1 (Statement 2, parameter 1)**」です。

   次の図に、2 つの SQL ステートメントが含まれている「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウを示します。 1 つ目のステートメントには 4 つのパラメーターがあり、2 つ目のステートメントには 1 つのパラメーターがあります。



7. 各パラメーターを、SQL ステートメントで指定した順に構成します。
   - パラメーターが DB2 または Oracle データベースのシーケンス列である場合:
     a. 「**パラメーターはシーケンス**」チェック・ボックスを選択します。
     b. 「**シーケンス名**」フィールドに、シーケンス列の名前を入力します。

シーケンス列は、整数データ型でなければならないため、「**パラメーター・タイプ**」は「int」に変更されます。

シーケンス列にサンプル値は必要ありません。

- パラメーターがシーケンス列でない場合:
  a. 「**パラメーターはシーケンス**」チェック・ボックスをクリアします。
  b. 「**パラメーター・タイプ**」フィールドで、パラメーターのデータ型を選択します。
  c. 「**サンプル値**」フィールドに、パラメーターのサンプル値を入力します。この値は、入力した SQL ステートメントの構文が正しいことを検証するために使用されます。

  INSERT ステートメントでは、パラメーターのデータ型と一致する任意の値を使用できます。

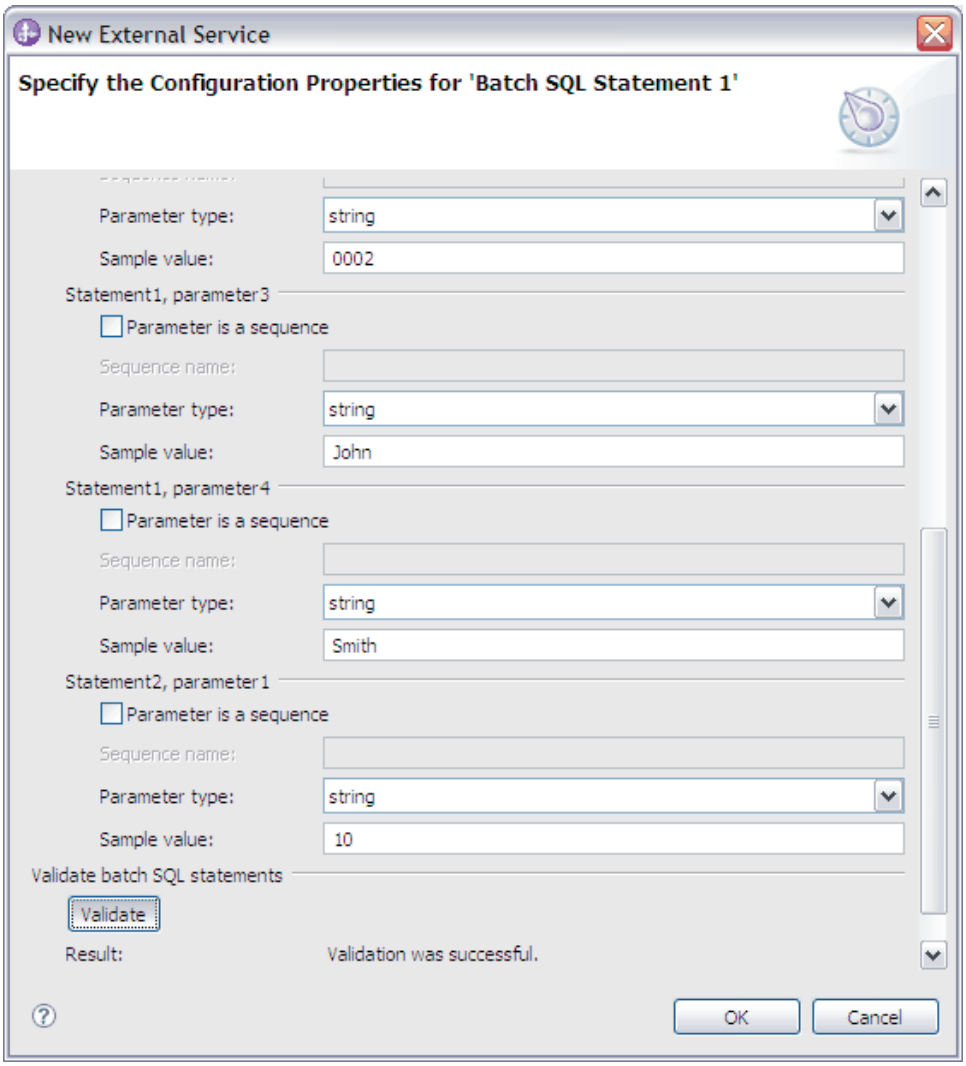
  UPDATE および DELETE ステートメントでは、データベースに存在する値を指定する必要があります。ウィザードは、このサンプル・データでステートメントを実行して、結果セットを取得し、この結果セットを使用して、バッチ SQL ビジネス・オブジェクトの属性を設定します。ウィザードはステートメントを実行しますが、結果を COMMIT しないため、データベースのデータが更新されることも、削除されることもありません。

  例えば、顧客の姓が格納されている列に対応するパラメーターの場合は、データ型として string を選択し、サンプル値 Smith を指定します。

8. 「**検証**」をクリックします。 「**結果**」エリアに検証結果が表示されます。

「**結果**」領域に「検証は失敗しました」メッセージが表示された場合は、指定した情報に問題があります。「検証は失敗しました。」の後に表示されているデータベース・サーバーからのエラー・メッセージを参考にして、定義を訂正します。SQL ステートメントの構文、およびパラメーターのデータ型を確認してください。UPDATE および DELETE ステートメントの場合は、サンプル・データがデータベースに存在することも確認してください。

次の図は、検証済みのバッチ SQL ビジネス・オブジェクトの「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウを示します。



9. メッセージ「検証は成功しました。」が表示された場合は、「**OK**」をクリックして、バッチ SQL ビジネス・オブジェクトの定義を保存します。

タスクの結果

構成したバッチ SQL ビジネス・オブジェクトが「エンタープライズ・システムでのオブジェクトの検索」ウィンドウにリストされます。

次のタスク

エンタープライズ・システムでのオブジェクトの検索ウィンドウで、他のタイプのビジネス・オブジェクトの選択と構成を続行します。完了したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

照会ビジネス・オブジェクトの選択および構成:

モジュールで使用するユーザー定義 SELECT ステートメントの照会ビジネス・オブジェクトを選択および構成します。

始める前に

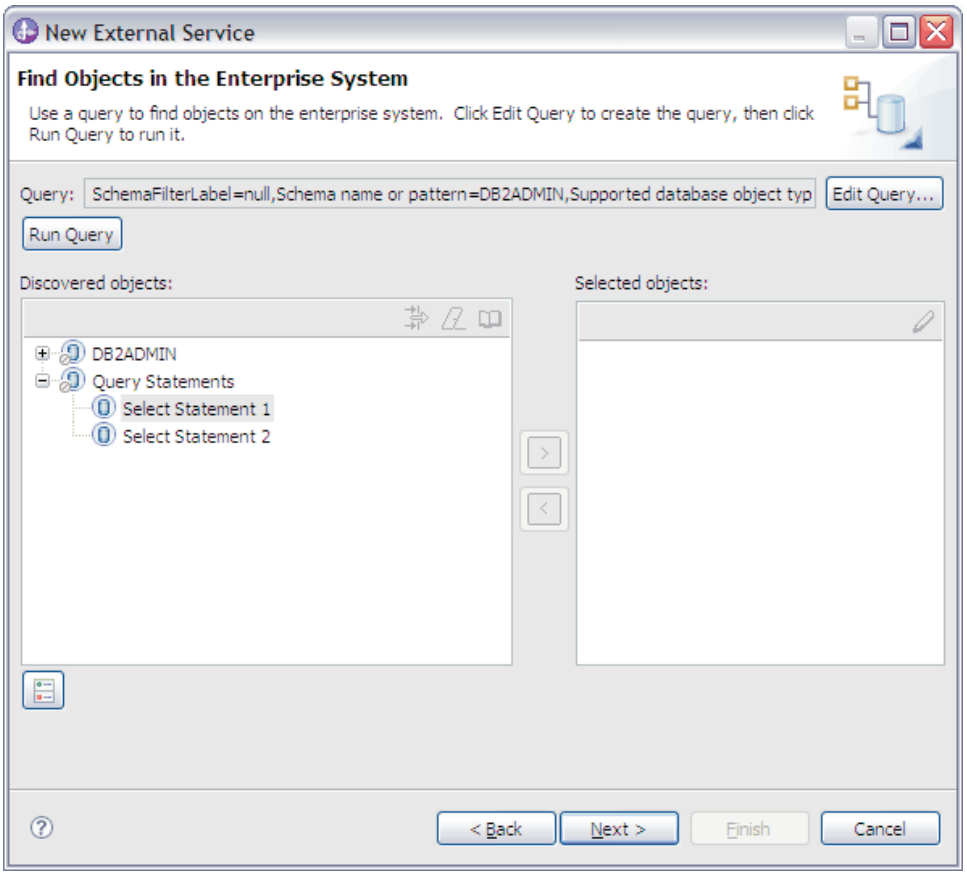
照会ビジネス・オブジェクトを構成するには、テーブルおよびビューをはじめとする、データベース内のデータ構造がわかっていなければなりません。モジュールがアクセスするべき列の名前およびデータ型を知っておく必要があります。さらに、SQL SELECT ステートメントを記述できなければなりません。

このタスクについて

このタスクは、外部サービス・ウィザードを介して実行されます。「エンタープライズ・システムでのオブジェクトの検索」ウィンドウから操作を開始し、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

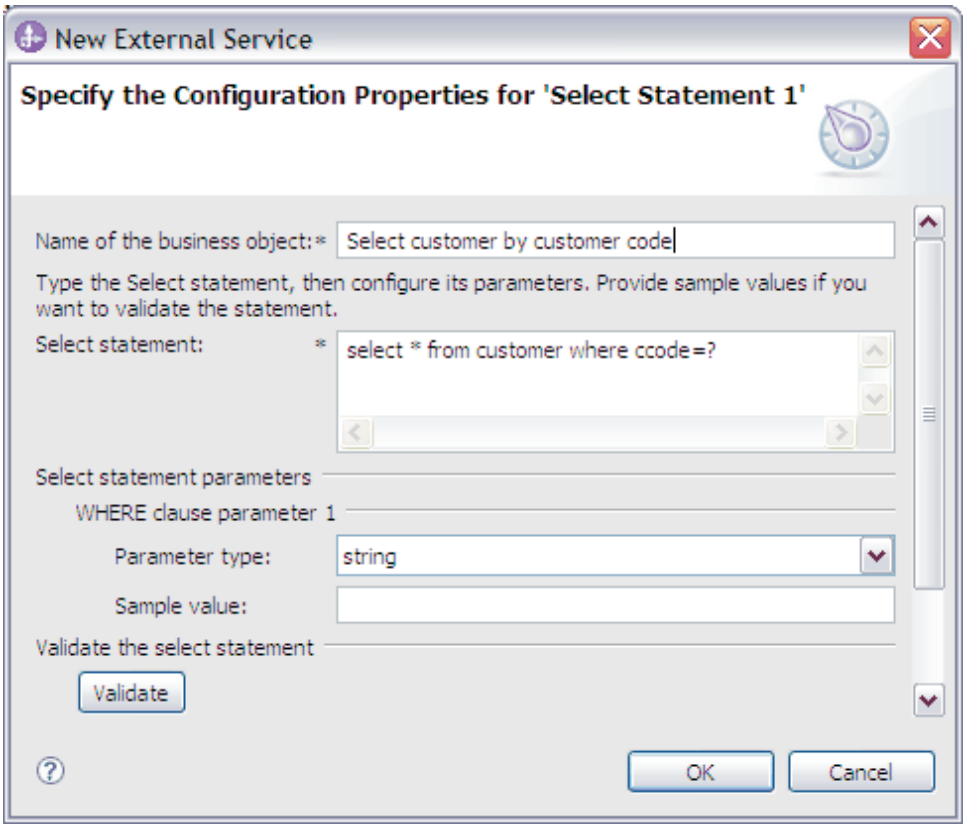
手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの **「ディスカバーされたオブジェクト」**リストで、**「照会ステートメント (Query Statements)」** ノードを展開します。 このノードには、「照会プロパティーの指定」ウィンドウで要求した各照会ビジネス・オブジェクトのオブジェクト・テンプレート**「Select ステートメント *n*」**があります。例えば、前述のウィンドウで照会ビジネス・オブジェクト数を 2 と指定した場合、**「ディスカバーされたオブジェクト」**リストには 2 つのオブジェクト・テンプレートが表示されます (以下の図を参照)。



2. オブジェクト・テンプレートを 1 つ以上選択し、「>」 (追加) ボタンをクリックして、オブジェクトを「**選択されたオブジェクト (Selected objects)**」リストに追加します。
3. 「**ビジネス・オブジェクトの名前**」フィールドに、ビジネス・オブジェクトの名前を入力します。 この名前には、スペースおよび各国語文字を使用できます。
4. 「**select ステートメント**」フィールドに、実行する SELECT ステートメントを入力します。各パラメーターは疑問符 (**?**) で示します。 次のサンプル SELECT ステートメントは、照会ビジネス・オブジェクトの柔軟性を示しています。
   - `select * from customer where ccode=?`
   - `select * from customer where id=? and age=?`
   - `select * from customer where lname like ?`
   - `select C.pkey, C.fname, A.city from customer C, address A WHERE (C.pkey = A.custid) AND (C.fname like ?)`

   注: SELECT ステートメントの FROM 節にネストされた SELECT ステートメントが含まれないようにしてください。
   ? を入力するごとに、ウィンドウが拡張して、そのパラメーターの WHERE 節を定義するための領域が表示されます。次の図は、単一のパラメーターを持つ照会ビジネス・オブジェクトの「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウを表しています。



5. 「**WHERE 節パラメーター *n***」エリアに、 SELECT ステートメントの各パラメーターに関する情報を入力します。

a. 「**パラメーター・タイプ**」フィールドで、パラメーターのデータ型を選択します。Oracle データベースの場合、アダプターはバッチおよび照会のビジネス・オブジェクトにおいて、配列、テーブル、構造体、ネストされた構造体などの複合型をパラメーターとしてサポートしていません。

b. 「**サンプル値**」フィールドに、パラメーターのサンプル値を入力します。

例えば、顧客の姓が格納されている列に対応するパラメーターの場合は、データ型として `string` を選択し、サンプル値 `Smith` を指定します。

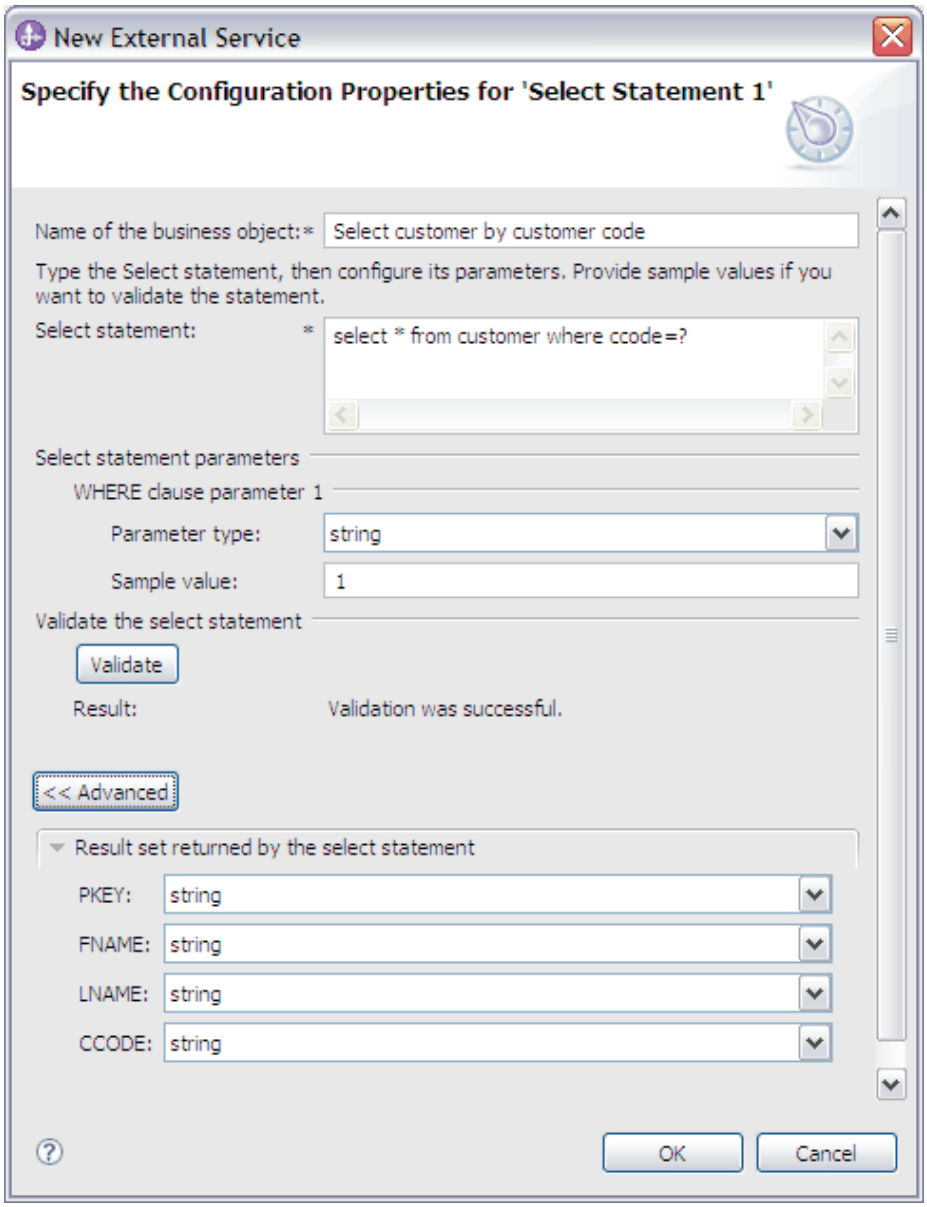
6. 「**検証**」をクリックします。「**結果**」エリアに検証結果が表示されます。

「**結果**」エリアに「検証は失敗しました。」と表示された場合は、入力した情報に問題があります。 「検証は失敗しました。」の後に表示されているデータベース・サーバーからのエラー・メッセージを参考にして、定義を訂正します。SELECT ステートメントの構文、パラメーターのデータ型、およびサンプル・データを確認してください。検証が成功した場合は、「**拡張**」ボタンが表示されます。

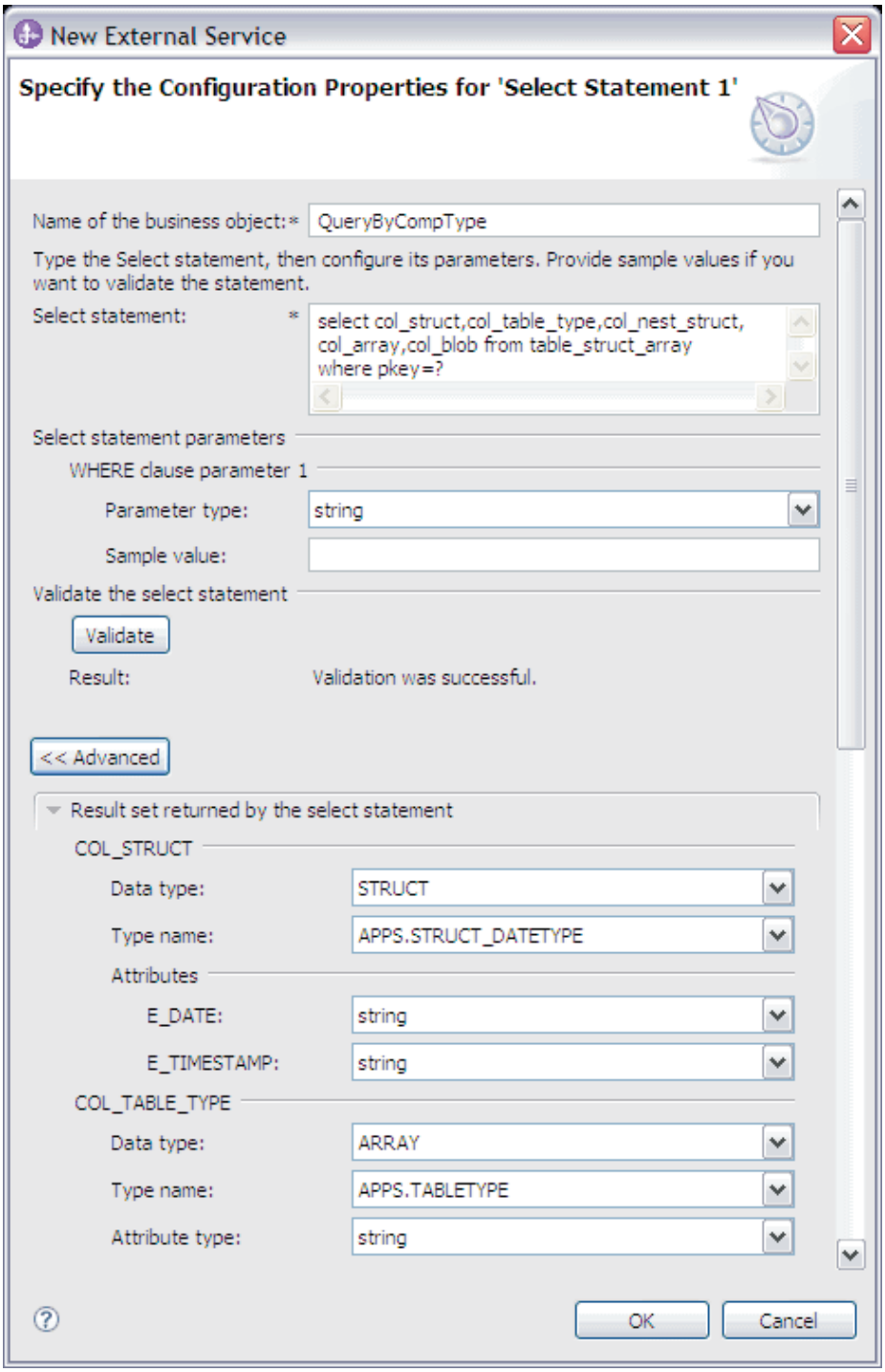
7. SELECT ステートメントから返される結果セット内の各列のデータ型マッピングを指定するには、以下の手順を実行します。

a. 「**拡張**」をクリックします。

b. 「**Select ステートメントによって返された結果セット**」を展開します。結果セット内の列ごとに、デフォルトのデータ型マッピングが表示されます。



Oracle データベースにおいて、配列、構造体、ネストされた構造体、テーブルなど、何らかの複合データ型が照会結果に含まれている場合は、型名および子属性の詳細も自動的にディスカバーされて、表示されます。以下の図は、Oracle テーブルの照会結果の型名および子属性の詳細を表しています。



c. マッピングを確認して、必要な場合は変更します。

8. 「**OK**」をクリックして、照会ビジネス・オブジェクトの定義を保存します。

タスクの結果

定義した照会ビジネス・オブジェクトが「エンタープライズ・システムでのオブジェクトの検索」ウィンドウにリストされます。

次のタスク

エンタープライズ・システムでのオブジェクトの検索ウィンドウで、他のタイプのビジネス・オブジェクトの選択と構成を続行します。完了したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

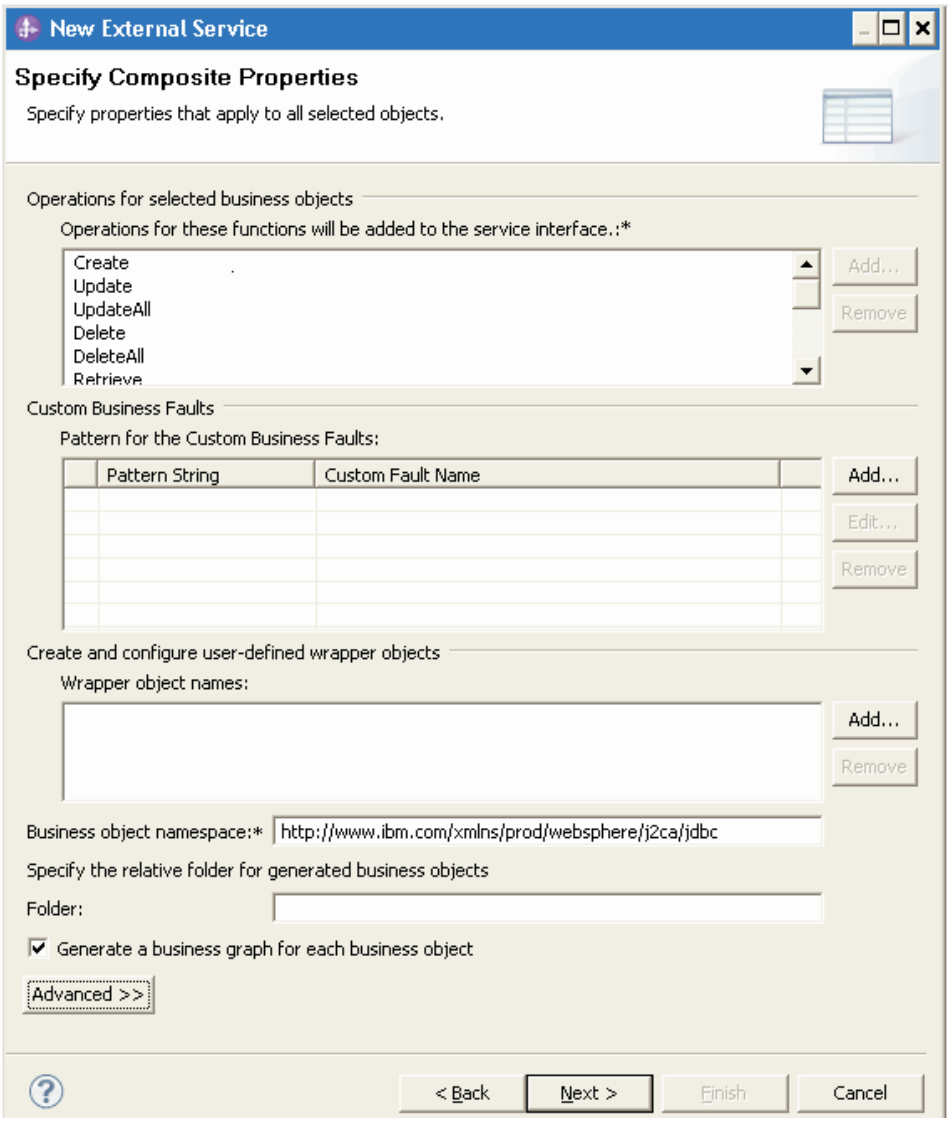
操作のグローバル・プロパティーの設定およびラッパー・ビジネス・オブジェクトの作成

外部サービス・ウィザードでデータベース・オブジェクトを選択した後、wrapper のビジネス・オブジェクトを定義し、すべてのビジネス・オブジェクトに適用するプロパティーを指定する必要があります。

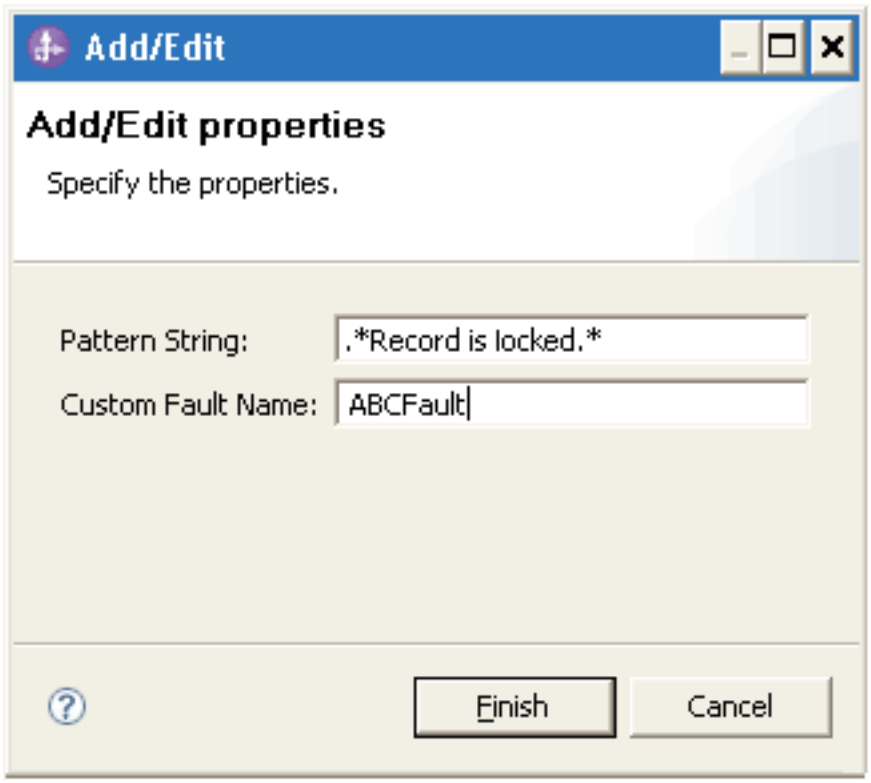
手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの「**選択済みオブジェクト**」リストに、アプリケーションで使用するすべてのビジネス・オブジェクト (ラッパー・ビジネス・オブジェクトを除く) が含まれている場合は、「**次へ**」をクリックします。
2. 「複合プロパティーの指定」ウィンドウで、操作リストを確認します。

   このウィンドウには、前のウィンドウで選択したすべてのビジネス・オブジェクトに対する Outbound サービスで、アダプターがサポートするすべての操作がリストされます。各ビジネス・オブジェクトがすべての操作をサポートするわけではありません。例えば、照会ビジネス・オブジェクトは、RetrieveAll 操作のみをサポートします。ストアード・プロシージャーおよびバッチ SQL ビジネス・オブジェクトは、 Execute 操作のみをサポートします。



3. 不要な操作を除去するには、その操作名を選択して「**除去 (Remove)**」をクリックします。 操作を復元する必要がある場合、「**追加**」をクリックして、削除した操作を復元します。
4. カスタム・フォールトのパターンを定義するには、次のようにします。
   a. 「カスタム・ビジネス・フォールトのパターン」領域で「**追加**」をクリックします。「プロパティーの追加/編集」ウィンドウが表示されます。
   b. 「**パターン・ストリング**」フィールドに正規表現を入力してフォールト・パターンを定義します。例えば、「.*Record is locked*」と入力します。



c. 「**カスタム・フォールト名**」フィールドに、フォールト名を入力します。この名前は .xsd ファイルを生成するのに使用されます。

注: カスタム・フォールト名は、モジュールの「データ・タイプ」の下にある、ビジネス・オブジェクト名、事前定義済みフォールト xsd 名、または他の xsd 名と同じであってはなりません。

d. 「**終了**」をクリックします。実行時に、アダプターはユーザー定義の正規表現に基づいてカスタム・フォールトを返します。

カスタム・フォールト・パターン・ストリングが、事前定義されたフォールト例外メッセージに一致すると、事前定義されたフォールトに優先して、カスタム・フォールトが戻されます。

5. ラッパー・ビジネス・オブジェクトを作成するため、以下の手順を実行します。

a. 「**Wrapper オブジェクト名**」領域で、「**追加**」をクリックします。

b. 「値の追加」ウィンドウに、ラッパー・ビジネス・オブジェクトの名前を入力して、「**OK**」をクリックします。 スペースを使用しないでください。名前には、各国語文字を使用できます。

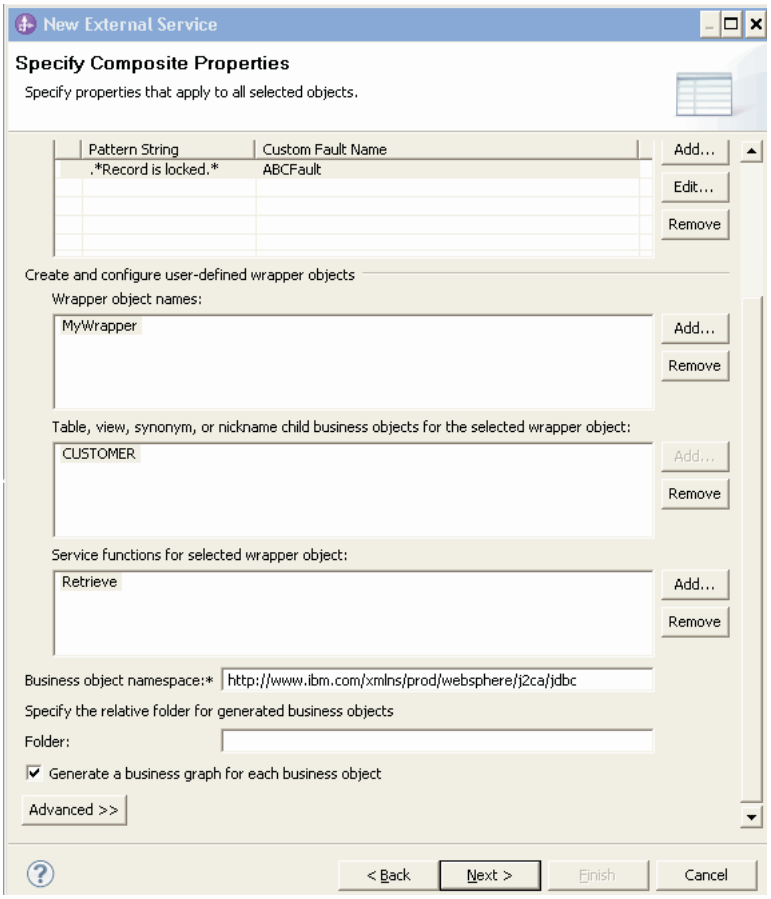
c. 「**選択した wrapper オブジェクトのテーブル、ビュー、シノニム、またはニックネーム子ビジネス・オブジェクト**」エリアで、「**追加**」をクリックします。

d. 「値の追加」ウィンドウで、ラッパーに組み込む 1 つ以上のビジネス・オブジェクトを選択して、「**OK**」をクリックします。

e. 「**選択した wrapper オブジェクトのサービス機能**」エリアで、「**追加**」をクリックします。

f. 「 値の追加」ウィンドウで、wrapper オブジェクトで実行する操作を 1 つ以上選択して、「**OK**」をクリックします。 RetrieveAll および ApplyChanges 操作はラッパー・ビジネス・オブジェクトに適用されないため、リストされません。

g. 作成するラッパー・ビジネス・オブジェクトごとに、この手順を繰り返します。 次の図は、2 つのラッパー・ビジネス・オブジェクトが定義されている「複合プロパティーの指定」ウィンドウを示します。このウィンドウでは、一度に 1 つのラッパー・ビジネス・オブジェクトについてのプロパティーが表示されます。



6. 「**ビジネス・オブジェクト名前空間**」フィールドで、デフォルトの名前空間を受け入れるか、または別の名前空間の完全な名前を入力します。

   ビジネス・オブジェクト・スキーマを論理的に分離するため、名前空間がビジネス・オブジェクト名の前に付加されます。

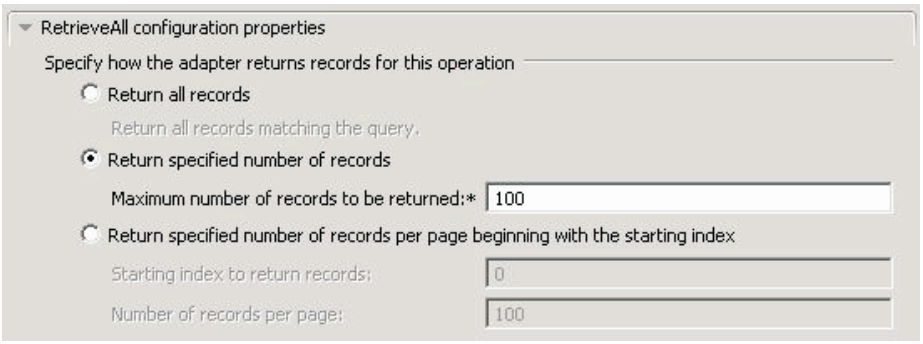
7. オプション: 「**フォルダー**」フィールドに、生成されたビジネス・オブジェクトを格納するフォルダーの相対パスを入力します。

   注: 1 つのモジュール内に複数のアダプター成果物を作成する場合は、モジュール内の各アダプターに対して、別々のビジネス・オブジェクト・フォルダーを指定するようにしてください。例えば、1 つのモジュール内に Oracle、JDBC、SAP、および JDE 用の成果物を作成する場合は、それらの各ア

ダプターに対して、別々の相対フォルダーを作成する必要があります。別々の相対フォルダーを指定していない場合、新規成果物を生成すると、既存の成果物が上書きされます。

8. ビジネス・オブジェクトごとにビジネス・グラフを作成する場合は、「**ビジネス・オブジェクトごとにビジネス・グラフを生成**」チェック・ボックスを選択します。 ビジネス・グラフは、以下の場合のみ必要になります。
   - ApplyChanges 操作を使用する必要がある場合
   - バージョン 7.0 より前のバージョンの IBM Integration Designer で作成されたモジュールにビジネス・オブジェクトを追加する場合

   注: 前のバージョンの IBM Integration Designer によって作成されたモジュールにビジネス・オブジェクトを追加する場合は、このオプションを選択する必要があります。それ以外の場合は、インターフェースを再接続する必要があります。
9. オプション: 「拡張」をクリックして拡張プロパティーを指定します。拡張セクションをそれぞれ展開して、プロパティーを確認します。
   - **RetrieveAll 構成プロパティー**

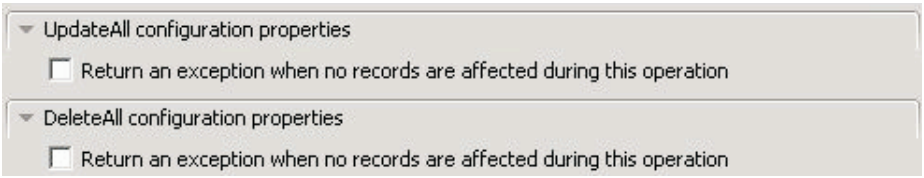


   注: 「**RetrieveAll 操作でアダプターがレコードを返す方法の指定**」領域にあるプロパティーは、RetrieveAll 操作にのみ適用可能です。これらのプロパティーは、 ステップ 3 で RetrieveAll 操作を削除すると使用不可になります。

   a. RetrieveAll 操作で照会と一致するすべてのレコードを返すようにする場合は、「**すべてのレコードを返す**」を選択します。
   b. RetrieveAll 操作が返す必要があるレコード数を指定する場合は、「**指定された数のレコードを返す**」を選択し、「**返されるレコードの最大数**」フィールドに値を入力します。デフォルト値は 100 です。指定した最大レコード数の値を超える数のレコードがデータベースから取り出された場合、アダプターは例外を返します。
   c. 指定した開始索引からのレコードをアダプターが返すようにする場合は、「**指定されたページ当たりの数のレコードを開始索引から返す**」を選択します。
   d. 「**レコードを返す開始索引**」フィールドに、 RetrieveAll 操作がレコードの取り出しを開始しなければならない索引を指定します。
   e. 「**ページ当たりのレコード数**」フィールドに、開始索引から取り出されるレコードの数を指定します。

注: 「**レコードを返す開始索引**」プロパティーまたは「**ページ当たりのレコード数**」プロパティーの値を実行時に変更するには、「**ビジネス・オブジェクトごとにビジネス・グラフを生成**」チェック・ボックスを選択します。実行時に動的にこれらのプロパティーを変更するには、実行時の対話プロパティーの動的変更を参照してください。

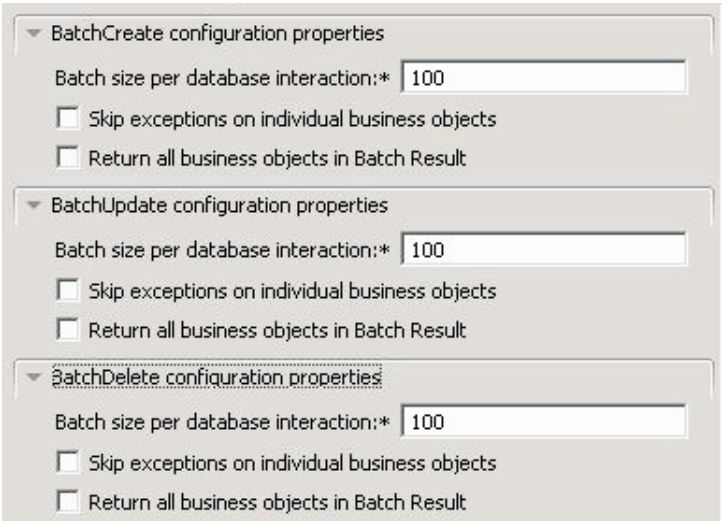
- **UpdateAll 構成プロパティー**および **DeleteAll 構成プロパティー**



  – データベース内のレコードが UpdateAll 操作または DeleteAll 操作中に影響を受けない場合はアダプターが RecordsNotFoundException を返すようにしたい場合、「**この操作中に影響を受けるレコードがない場合は例外を返す**」チェック・ボックスを選択します。

注: ステップ 3 で UpdateAll 操作と DeleteAll 操作の両方を削除すると、このプロパティーは使用不可になります。

- **BatchCreate、BatchUpdate、および BatchDelete の各構成プロパティー**



  – 「**データベースとの 1 回の対話でのバッチ・サイズ**」に、 1 回のバッチ操作で処理されるビジネス・オブジェクトの数を指定します。詳しくは、344 ページの『batchSize』を参照してください。
  – バッチ操作中にスローされる例外をスキップするには、「**個別ビジネス・オブジェクトでの例外をスキップする**」チェック・ボックスを選択します。詳しくは、345 ページの『skipErrorsInBatch』を参照してください。
  – バッチ操作の完了後に結果のビジネス・オブジェクトが返されるようにするには、「**すべてのビジネス・オブジェクトをバッチ結果に返す**」チェック・ボックスを選択します。詳しくは、345 ページの『returnBOsInBatch』を参照してください。

10. 「**次へ**」をクリックします。

タスクの結果

ラッパー・ビジネス・オブジェクトを作成して、モジュール内のすべてのビジネス・オブジェクトに適用する情報を設定しました。

次のタスク

ウィザードでの作業を続行します。次のステップでは、実行時に使用するデプロイメント情報、およびサービスをモジュールとして保存するための情報を指定します。

デプロイメント・プロパティーの設定およびサービスの生成

モジュールのビジネス・オブジェクトを選択して構成した後、外部サービス・ウィザードを使用して、アダプターが特定のデータベースに接続するために使用するプロパティーを構成します。ウィザードは、すべての成果物とプロパティー値を保存する、新規のビジネス・インテグレーション・モジュールを作成します。

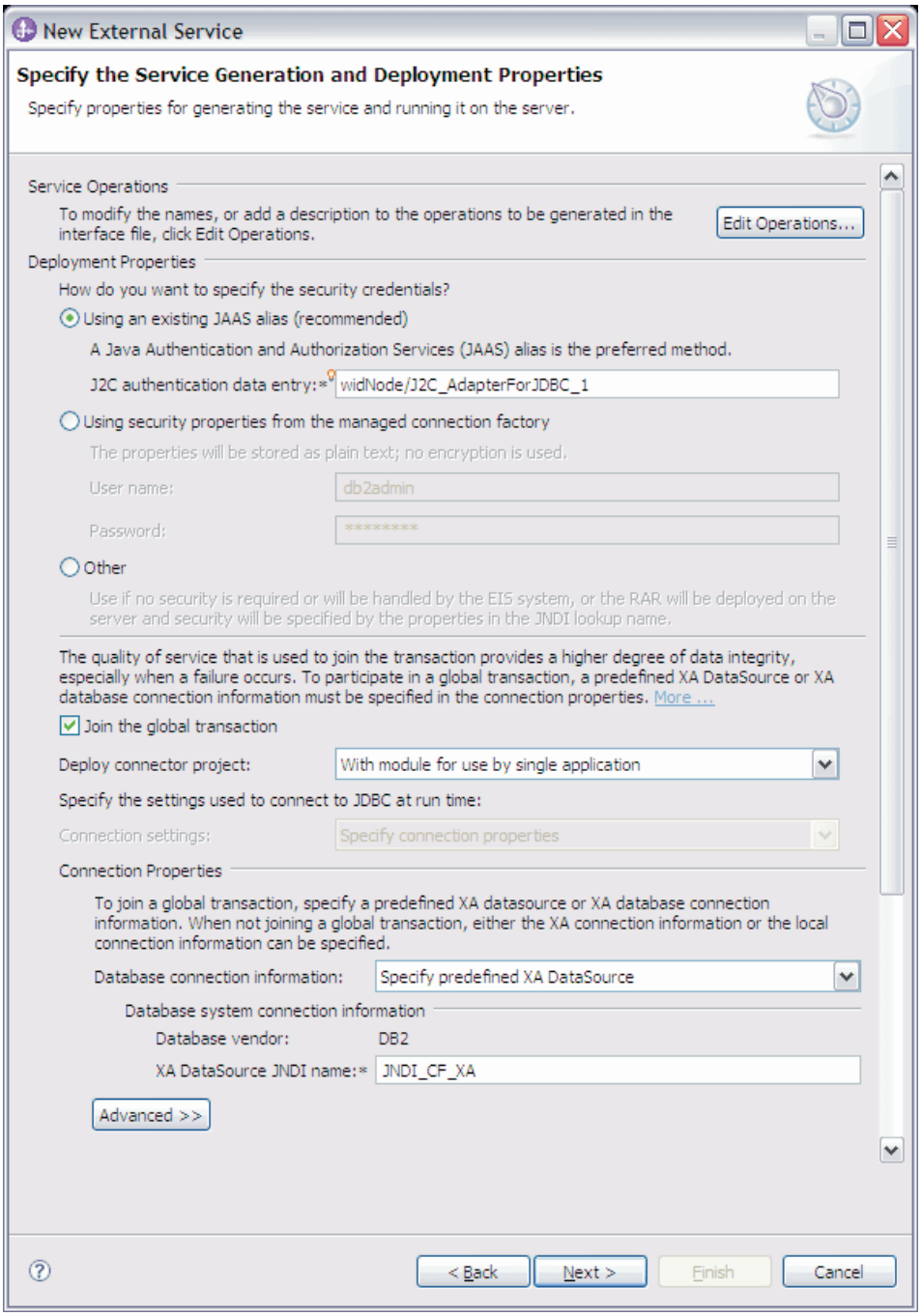
このタスクについて

このタスクは、「外部サービス・ウィザードのサービス生成およびデプロイメント・プロパティーの指定」ウィンドウおよび「ロケーション・プロパティーの指定」ウィンドウを介して実行されます。

このタスクの接続プロパティーは、ウィザードがデータベースに接続するために使用した値に初期化されます。他の値を使用するようにモジュールを構成するには、ここで値を変更します。例えば、実行時に IBM i で IBM Toolkit for Java ネイティブ・ドライバーを使用するには、ここでドライバー情報を設定します。

手順

1. 「サービス生成およびデプロイメント・プロパティーの指定」ウィンドウで、**「操作の編集」**をクリックして、作成するビジネス・オブジェクトの操作の名前を確認するか、その操作の説明を追加します。



2. 「**デプロイメント・プロパティー**」エリアで、アダプターが実行時にユーザー名およびパスワードを取得する方法を指定します。
   - J2C 認証別名を使用するには、「**既存の JAAS 別名を使用する (推奨)**」をクリックして、「**J2C 認証データ項目**」フィールドに別名の名前を入力します。既存の認証別名を指定することも、(モジュールをデプロイする前に) 認証別名を作成することもできます。名前は大文字小文字が区別されます。また、名前にはノード名が含まれます。
   - 管理接続プロパティーを使用するには、「**管理接続ファクトリーのセキュリティー・プロパティーを使用**」をクリックして、「**ユーザー名**」フィールドと「**パスワード**」フィールドに値を入力します。

- 他の手段でユーザー名とパスワードを管理するには「**その他**」をクリックします。

注: データベース接続の確立にローカル接続情報を使用する場合、セキュリティー資格情報が必要です。セキュリティー・メカニズムとして、「**既存の JAAS 別名を使用する (推奨)**」または「**管理接続ファクトリーのセキュリティー・プロパティーを使用**」のいずれかを選択できます。サーバー上の既存のデータ・ソースを使用する場合には、セキュリティー資格情報は必要ありません。この場合、セキュリティー・メカニズムとして「**その他**」を選択できます。また、「**J2C 認証データ項目**」フィールドを設定するか、「**ユーザー名**」フィールドと「**パスワード**」フィールドを設定すると、データ・ソースのユーザー名とパスワードがこれらのフィールドの値によって指定変更されます。

3. デフォルトでは、アダプターはグローバル・トランザクションを結合するように構成されています。アダプターは、グローバル・トランザクションに XA 接続を使用します。定義済みの XA データ・ソースまたは XA データベース接続情報を指定することにより、XA 接続を構成できます。ローカル・トランザクションの場合、「**グローバル・トランザクションの結合**」チェック・ボックスをクリアします。定義済みの接続プール・データ・ソース、ローカル・データベース接続情報、定義済みの XA データ・ソースまたは XA データ・ソース接続情報を指定することにより、ローカル・トランザクションを構成できます。

   注: 「**グローバル・トランザクションの結合**」 チェック・ボックスを選択すると、「**データベース接続情報**」リストには、XA 関連のデータベース接続オプションのみが表示されます。前のバージョンの成果物にローカル接続プロパティーのみが含まれており、「**グローバル・トランザクションの結合**」チェック・ボックスが選択されている場合、マイグレーション後に、アセンブリー・エディターの「**データベース接続情報**」リストには、ローカル・データベース接続のオプションが表示されます。

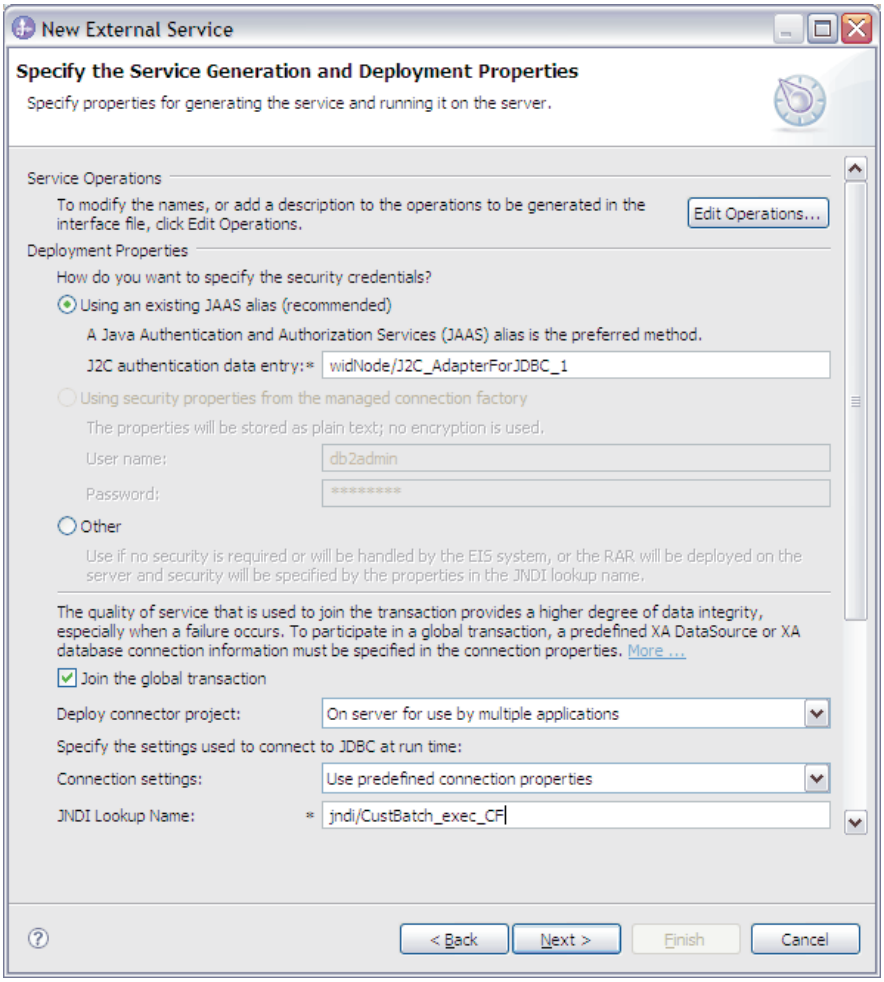
   注: グローバル・トランザクションに有効な接続情報を指定しなかった場合、アダプターは ResourceException を戻します。

4. 「**コネクター・プロジェクトのデプロイ**」フィールドで、モジュールにアダプター・ファイルを組み込むかどうかを指定します。次の値のいずれかを選択してください。
   - **単一アプリケーションが使用するモジュールで (With module for use by single application)**: アダプター・ファイルをモジュール内に組み込むと、モジュールをすべてのアプリケーション・サーバーにデプロイすることができます。組み込みアダプターを使用するのは、組み込みアダプターを使用するモジュールが 1 つある場合か、複数のモジュールでバージョンの異なるアダプターを実行する必要がある場合です。組み込みアダプターを使用すると、他のモジュールのアダプター・バージョンを変更することで、それらのモジュールを不安定にするリスクを生じることなく、1 つのモジュール内でアダプターをアップグレードできます。
   - **複数アプリケーションが使用するサーバー上 (On server for use by multiple applications)**: モジュール内にアダプター・ファイルを組み込まない場合は、このモジュールを実行するアプリケーション・サーバーごとに、アダプター・ファイルをスタンドアロン・アダプターとしてインストールする必要が

あります。複数のモジュールが同じバージョンのアダプターを使用可能で、アダプターを中央の場所で管理する場合は、スタンドアロン・アダプターを使用します。スタンドアロン・アダプターの場合も、複数のモジュールに対して単一のアダプター・インスタンスを実行することにより、必要なリソースが軽減されます。

5. 前のステップで「**複数アプリケーションが使用するサーバー上 (On server for use by multiple applications)**」を選択した場合は、実行時に使用される接続プロパティーを指定します。
   - サーバーで管理接続ファクトリーまたは活動化仕様を手動で作成および構成した場合、あるいは同じ管理接続ファクトリーまたは活動化仕様のプロパティーを使用して同じデータベースに接続するアプリケーションを既にデプロイ済みの場合は、その Java Naming and Directory Interface (JNDI) データ・ソースの名前を指定することによって、管理接続ファクトリーまたは活動化仕様を再利用できます。
     a. 「**接続設定 (Connection settings)**」リストで、「**事前定義された接続プロパティーを使用する**」を選択します。
     b. 「**JNDI ルックアップ名**」フィールドに、既存の管理接続ファクトリーまたは活動化仕様の JNDI データ・ソースの名前を入力します。

        以下の図に、アダプターのスタンドアロン・デプロイメントで管理接続ファクトリーまたは活動化仕様を再利用する場合の標準的な設定を示します。



c. 「**次へ**」をクリックしてこのタスクを完了します。

- これが、特定のユーザー名とパスワードを使用してデータベースに接続する最初のアプリケーションである場合、または他のアプリケーションとは別々にユーザー名とパスワードを管理する場合は、「**接続プロパティーの指定**」を選択します。

6. 「**接続プロパティー**」エリアで、アダプターが実行時にデータベース接続を確立する方法を指定します。以下のいずれかの方法で、実行時にアダプターとの接続を確立できます。
   - サーバー上の事前定義の XA データ・ソースを使用するには、以下のようにします (XA 接続用)。
     a. 「**グローバル・トランザクションの結合**」チェック・ボックスを選択します。
     b. 「**データベース接続情報**」リストから、「**事前定義 XA DataSource の指定**」を選択します。
     c. 「**データベース・システム接続情報**」エリアで、「**XA DataSource JNDI 名**」フィールドに値を入力します。この値は、IBM Business Process Manager または WebSphere Enterprise Service Bus で作成される、XA トランザクションをサポートする JNDI データ・ソースに設定する必要があ

ります。このプロパティーについて詳しくは、 337 ページの『XA DataSource JNDI 名 (XADataSourceJNDIName)』を参照してください。

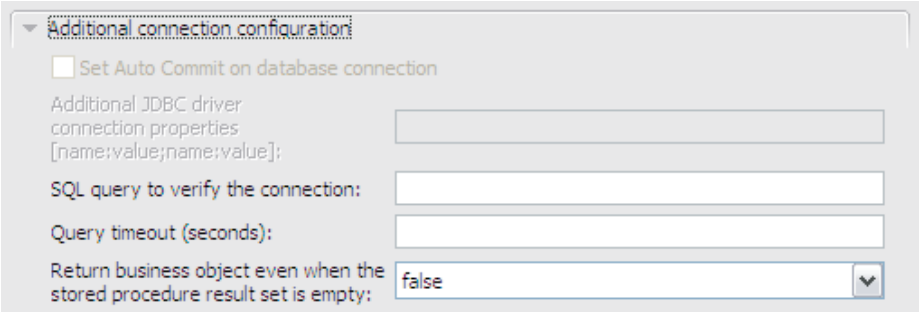
- アダプター・プロパティーに保存される接続情報を指定するには、以下のようにします (XA 接続用)。
  a. 「**グローバル・トランザクションの結合**」チェック・ボックスを選択します。
  b. 「**データベース接続情報**」リストから、「**XA データベース接続情報の指定**」を選択します。

     注: 「**XA データベース接続情報の指定**」オプションを使用すると、Oracle データベースおよび DB2 データベースの場合にのみ、グローバル・トランザクションをサポートするようにアダプターを構成することができます。
  c. 「**データベース・システム接続情報**」エリアで、DB2 データベースの「**XA DataSource 名**」フィールドと「**XA データベース名**」フィールドに値を入力します。Oracle データベースの場合は、「**XA DataSource 名**」フィールドと「**データベース URL**」フィールドに値を入力します。これらのプロパティーについて詳しくは、 328 ページの『データベース URL (DatabaseURL)』および 335 ページの『XA DataSource 名 (XADataSourceName)』を参照してください。
- 事前定義接続プール・データ・ソースを使用するには、以下のようにします (ローカル接続用)。
  a. 「**グローバル・トランザクションの結合**」チェック・ボックスをクリアします。
  b. 「**データベース接続情報**」リストから、「**事前定義接続プール DataSource の指定**」を選択します。
  c. 「**データベース・システム接続情報**」エリアで、「**接続プール DataSource JNDI 名**」フィールドに既存の JNDI データ・ソースの名前を入力します。このプロパティーについて詳しくは、 338 ページの『接続プール・データ・ソースの JNDI 名 (PoolDataSourceJNDIName)』を参照してください。
- アダプター・プロパティーに保存される接続情報を指定するには、以下のようにします (ローカル接続用)。
  a. 「**グローバル・トランザクションの結合**」チェック・ボックスをクリアします。
  b. 「**データベース接続情報**」リストから、「**ローカル・データベース接続情報の指定**」を選択します。
  c. 「**データベース・システム接続情報**」エリアで、「**データベース URL**」フィールドと「**JDBC ドライバー・クラス名**」フィールドに値を入力します。これらのプロパティーについて詳しくは、 328 ページの『データベース URL (DatabaseURL)』および 330 ページの『JDBC ドライバー・クラス (JDBC driver class) (JDBCDriverClass)』を参照してください。

7. 必須の接続プロパティーの値を確認し、必要に応じて変更します。 フィールドは、ウィザードの開始時に指定した接続情報で初期化されます。これらの値を変更して、別のユーザー名およびパスワードを実行時に指定できます。また、別のデータベースに接続できますが、スキーマ名は両方のデータベースで同じ

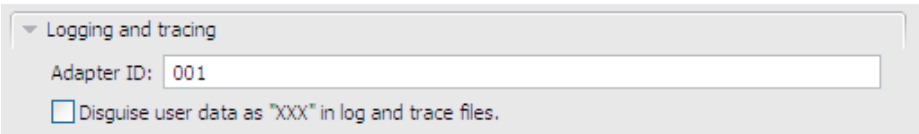
でなければなりません。接続プロパティーの形式はデータベース固有です。プロパティーについて詳しくは、 324 ページの『管理接続ファクトリー・プロパティー』を参照してください。

8. オプション: 「**拡張**」をクリックして拡張プロパティーを指定します。 拡張セクションをそれぞれ展開して、プロパティーを確認します。
   - **追加の接続構成**



     a. データベースの AUTOCOMMIT をオンにする場合は、「**データベース接続時に自動コミットを設定**」チェック・ボックスを選択します。このプロパティーについて詳しくは、 327 ページの『自動コミット (Auto commit) (AutoCommit)』を参照してください。
     b. 「**追加の JDBC ドライバー接続プロパティー (Additional JDBC driver connection properties)**」を設定します。このプロパティーについて詳しくは、 327 ページの『追加の JDBC ドライバー接続プロパティー [name:value;name:value] (JDBCDriverConnectionProperties)』を参照してください。
     c. 「**接続を検証するための SQL 照会**」を設定します。このプロパティーについて詳しくは、 333 ページの『接続を検証するための SQL 照会 (PingQuery)』を参照してください。
     d. 「**照会タイムアウト**」フィールドに、アダプターがデータベース照会への応答を待機する時間 (秒) を入力します。このプロパティーについて詳しくは、 332 ページの『照会タイムアウト (秒) (QueryTimeOut)』を参照してください。
     e. 「**ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す**」を設定します。このプロパティーについて詳しくは、 333 ページの『ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)』を参照してください。

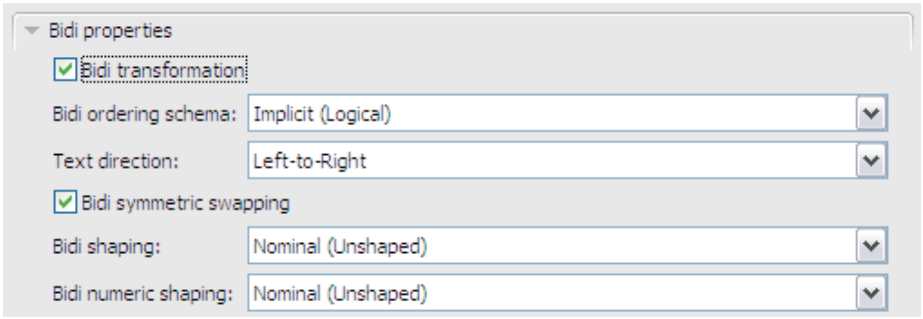
   - **ロギングおよびトレース**



     – アダプターのインスタンスが複数ある場合は、**アダプター ID** をこのインスタンスに固有の値に設定します。このプロパティーについて詳しくは、 326 ページの『アダプ ター ID (AdapterID)』を参照してください。

– ログあるいはトレースに特定の情報を表示しないよう情報をマスクする必要がある場合は、**「ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する」**を選択します。このプロパティーについて詳しくは、329 ページの『ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する (HideConfidentialTrace)』を参照してください。

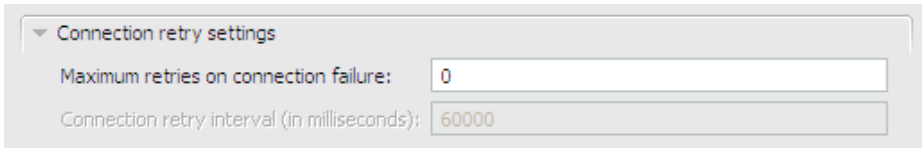
- **BiDi プロパティー**



– 実行時にアダプターの双方向言語サポートを有効にするには、**「BiDi 変換」**チェック・ボックスを選択します。

– 順序付けスキーマ、テキスト方向、対称スワッピング、文字シェーピング、および数字シェーピングの各プロパティーを設定して、双方向変換の実行方法を制御します。

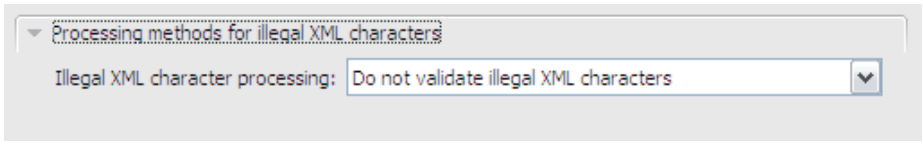
- **接続再試行の設定**



– 接続に失敗した場合にアダプターが データベース への再接続を試行できる回数を指定するには、**「接続に失敗した場合の最大再試行回数」**にゼロ以上の値を設定します。詳しくは、336 ページの『接続に失敗した場合の最大再試行回数 (connectionRetryLimit)』を参照してください。

– 接続に失敗した場合の再試行の時間間隔を指定するには、**「接続の再試行間隔 (ミリ秒単位)」**にミリ秒単位の値を設定します。このプロパティーが有効になるのは、connectionRetryLimit プロパティーの値がゼロより大きい場合のみです。詳しくは、336 ページの『接続の再試行間隔 (ミリ秒単位) (ConnectionRetryInterval)』を参照してください。

- **正しくない XML 文字の処理方法**



– デフォルトのアダプター動作を使用するには、**「正しくない XML 文字を検証しない」**を選択します。

– 例外メッセージを受け取った後で続行し、実行時に正しくない XML 文字をトレース・ファイルに記録するには、「**BO の内容に正しくない XML 文字が含まれる場合は例外をスローする**」を選択します。
– 正しくない XML 文字を廃棄し、実行時にそれらの文字をトレース・ファイルに記録するには、「**すべての正しくない XML 文字および関連ログを廃棄する**」を選択します。

9. 「**次へ**」をクリックします。 ロケーション・プロパティーの指定ウィンドウが表示されます。
10. 「ロケーション・プロパティーの指定」ウィンドウで、作成するモジュールの名前を指定します。新規モジュールを指定することも、既存のモジュールを指定することもできます。
    - 目的のモジュール名が「**モジュール**」リストに表示されている場合は、その名前を選択します。

      重要: モジュールに、現在構成しているものと同じ名前のインターフェースまたはビジネス・オブジェクトが含まれている場合、そのモジュールにある元のインターフェースまたはビジネス・オブジェクトは新しいバージョンによって置き換えられます。
    - それ以外の場合は、新規モジュールを作成します。
      a. 「**新規**」をクリックします。
      b. 「ビジネス・インテグレーション・プロジェクト・タイプの選択」ウィンドウで、「**モジュール**」を選択して「**次へ**」をクリックします。
      c. 「モジュールの作成」ウィンドウで、モジュールの名前を入力します。例えば、JDBCOutboundModule です。
      d. サービス記述ファイル (`.import` ファイルおよび `.wsdl` ファイル) をモジュール内のデフォルト・フォルダーの中に入れる場合は、「**デフォルト・ロケーションの使用**」を選択したままにします。モジュール内の別のフォルダーを指定する場合は、このオプションをクリアしてから、「**参照**」をクリックして、「**場所**」フィールドに別のフォルダーを指定します。
      e. ウィザードを閉じたときに IBM Integration Designer のアセンブリー・ダイアグラムでこのモジュールが自動的に開くようにする場合は、「**モジュール・アセンブリー・ダイアグラムを開く**」チェック・ボックスを選択します。それ以外の場合は、このオプションをクリアします。
      f. 「**終了**」をクリックすると、新規モジュールが作成されます。
11. 成果物に使用する名前空間を指定します。
    - モジュールのビジネス・オブジェクトにデフォルトの派生名前空間を使用させる場合は、「**デフォルトの名前空間を使用する**」チェック・ボックスを選択します。
    - 別の名前空間を指定するには、「**デフォルトの名前空間を使用する**」チェック・ボックスをクリアして、「**名前空間**」フィールドに別の値を入力します。
12. オプション: サービス記述を保存する新規モジュール内のフォルダーを指定します。「**フォルダー**」にフォルダー名を入力するか、既存フォルダーを参照します。 フォルダー名を指定しない場合、成果物 (インポート・ファイル、XSD

ファイル、および WSDL ファイル) は、モジュールのルート・フォルダー (すなわち、モジュール名を持つフォルダー) に保管されます。

13. 「**名前**」フィールドで、デフォルトのインポート名を受け入れるか、または別の名前を入力します。
14. オプション: ビジネス・オブジェクトをライブラリーに保存して他のモジュールで使用できるようにする場合は、「**ライブラリーにビジネス・オブジェクトを保存する**」を選択し、「**ライブラリー**」フィールドでライブラリーの場所を指定します。
15. オプション: 「**説明**」フィールドに、モジュールを説明するコメントを入力します。
16. プロパティーの設定が完了したら、「**終了**」をクリックします。

タスクの結果

ウィザードは終了します。モジュールがプロジェクトに作成され、成果物が生成されます。

次のタスク

インスタンスによっては、構成を完了するためにアセンブリー・エディターを使用しなければならない場合があります。完了したら、モジュールをテストまたはデプロイできます。

構成の完了

場合によって、ビジネス・オブジェクトの構成を完了するために、手動による構成ステップが必要になります。

このタスクについて

ウィザードによって生成された成果物をカスタマイズする必要がある場合に、このタスクを実行してください。以下の状態でこのタスクを行う可能性があります。

- ある列の CopyAttribute パラメーターの値を、他の列と同じ値に設定する場合。
- ビジネス・オブジェクトから属性を除去する場合。例えば、参照する必要のないデータベース列に対応した単純な属性を除去することによって、ビジネス・オブジェクト設計を簡素化できます。
- ビジネス・オブジェクトに属性を追加する場合。例えば、 DB2 または Microsoft SQL Server データベースのいずれかのデータベース内のテーブルに対してディスカバリー処理を実行するときに、そのテーブルで ID 列として 1 つの列が定義されている場合、そのテーブルに対して生成されるビジネス・オブジェクトには固有 ID 属性は含まれません。アダプターは実行時に ID 列の固有 ID を必要とするため、これを属性のアプリケーション固有情報に追加する必要があります。この場合、<UID>AUTO</UID> を属性のアプリケーション固有情報に追加します。Oracle データベースの場合、Oracle では ID 列がサポートされていないため、シーケンス名に UID を指定して、自動生成のフィールドとしてフィールドを定義するようにします。

  注: Informix データベースからテーブル・ビジネス・オブジェクトを生成した場合は、同様の変更を加える必要はありません。Informix データベースのテーブルに対してディスカバリー・プロセスを実行し、そのテーブルで列がシリアルとし

て定義されている場合 (ID 列は、Informix ではシリアル列 と呼ばれる)、結果のビジネス・オブジェクトにシリアル列の固有 ID 属性が含まれます。したがって、ビジネス・オブジェクトのアプリケーション固有情報を編集する必要はありません。 Informix データベース・テーブルのシリアル列に生成される固有 ID パラメーター値は、 `serial` または `serial8` のいずれかです。

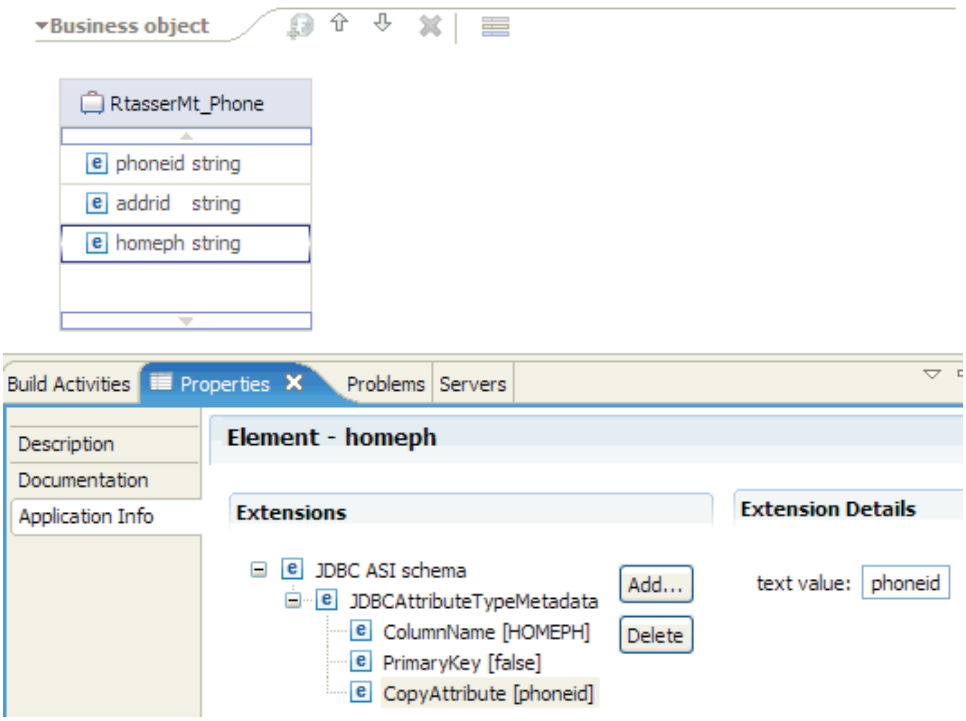
- 複数の親を持つテーブル・ビジネス・オブジェクトについて追加の親を構成する場合。ウィザードは、1 つのテーブル・ビジネス・オブジェクトにつき、親を 1 つのみ構成します。

このトピックでは、CopyAttribute パラメーターをテーブル・ビジネス・オブジェクトに設定する、詳細な手順を説明します。ビジネス・オブジェクト構造に対する他の変更 (上に述べた変更など) は、同じ手法を使用して、実行できます。

CopyAttribute パラメーターは、他の列の値およびアプリケーション固有情報を取り込む対象となる列についての属性のプロパティーに含まれます。例えば、テーブルの新しい行の `contact` 列に `e-mail` 列と同じ値を組み込むには、`contact` 属性の CopyAttribute パラメーターを `e-mail` に設定します。 IBM Integration Designer でアセンブリー・エディターを使用して、値を設定します。

手順

1. IBM Integration Designer の「ビジネス・インテグレーション」パースペクティブで、モジュール名、「**データ・タイプ**」を展開し、テーブル・ビジネス・オブジェクトを見つけます。 ビジネス・オブジェクト名は、データベース・スキーマ名にデータベース表名を加えたものです。オプションの名前空間が、名前の最初に含まれる場合もあります。
2. ビジネス・オブジェクト名を右クリックして、「**開く**」を選択します。 アセンブリー・エディターに、ビジネス・オブジェクトが表示され、それぞれの列に関するフィールドが表示されます。
3. アセンブリー・エディターで、他の列と一致させるために設定したい列を選択します。
4. 「プロパティー」ビューで、「アプリケーション情報」を選択します。 「プロパティー」ビューが表示されていない場合は、列の名前を右クリックして、「**プロパティーに表示 (Show in Properties)**」をクリックします。
5. 「**JDBC ASI スキーマ (JDBC ASI schema)**」を展開し、「**JDBCAttributeTypeMetadata**」を展開します。
6. 「**JDBCAttributeTypeMetadata**」を右クリックして、「**新規**」 > 「**jdbcasi:CopyAttribute**」を選択します。
7. 「**CopyAttribute**」プロパティーを選択します。
8. 「拡張子の詳細 (Extension Details)」領域で、コピーする情報を含む列の名前に、テキスト値を設定します。列は、現行ビジネス・オブジェクトに含めることも、親のビジネス・オブジェクトに含めることもできます。現行ビジネス・オブジェクトの列からコピーするには、値を列の名前に設定します (`phoneid` など)。親ビジネス・オブジェクトの列からコピーするには、列の名前のプレフィックスに 2 つのピリオド (..) を使用します (`..phone` など)。以下の図は、CopyAttribute プロパティーが現行テーブルの列に設定されているアセンブリー・エディターを表しています。



タスクの結果

CopyAttribute プロパティーを使用して、別の列の情報に基づいてデータベース列のビジネス・オブジェクト属性およびプロパティーを設定するように、ビジネス・オブジェクトは構成済みです。

次のタスク

これで、モジュールをテストおよびデプロイできます。

Inbound 処理のモジュールの構成

アダプターを Inbound 処理に使用するようにモジュールを構成するには、IBM Integration Designer 内で 外部サービス・ウィザードを使用して、データベースからビジネス・オブジェクトおよびサービスを検出して選択し、ビジネス・オブジェクト定義および関連する成果物を生成します。

Inbound 処理のデータベース・オブジェクトのディスカバー

接続プロパティーを構成した後は、データベース・オブジェクトを検索する照会を実行します。ディスカバーされたオブジェクトのツリーを参照して、データベース内のオブジェクトの構造を理解します。また、フィルターを使用して、参照したいデータベース・オブジェクトのみを表示します。

始める前に

データベースのアクセスに必要なプログラムのデータ要件を知っていなければなりません。例えば、データベースに関する以下の情報が必要になります。

- モジュールはどのスキーマにアクセスする必要があるか

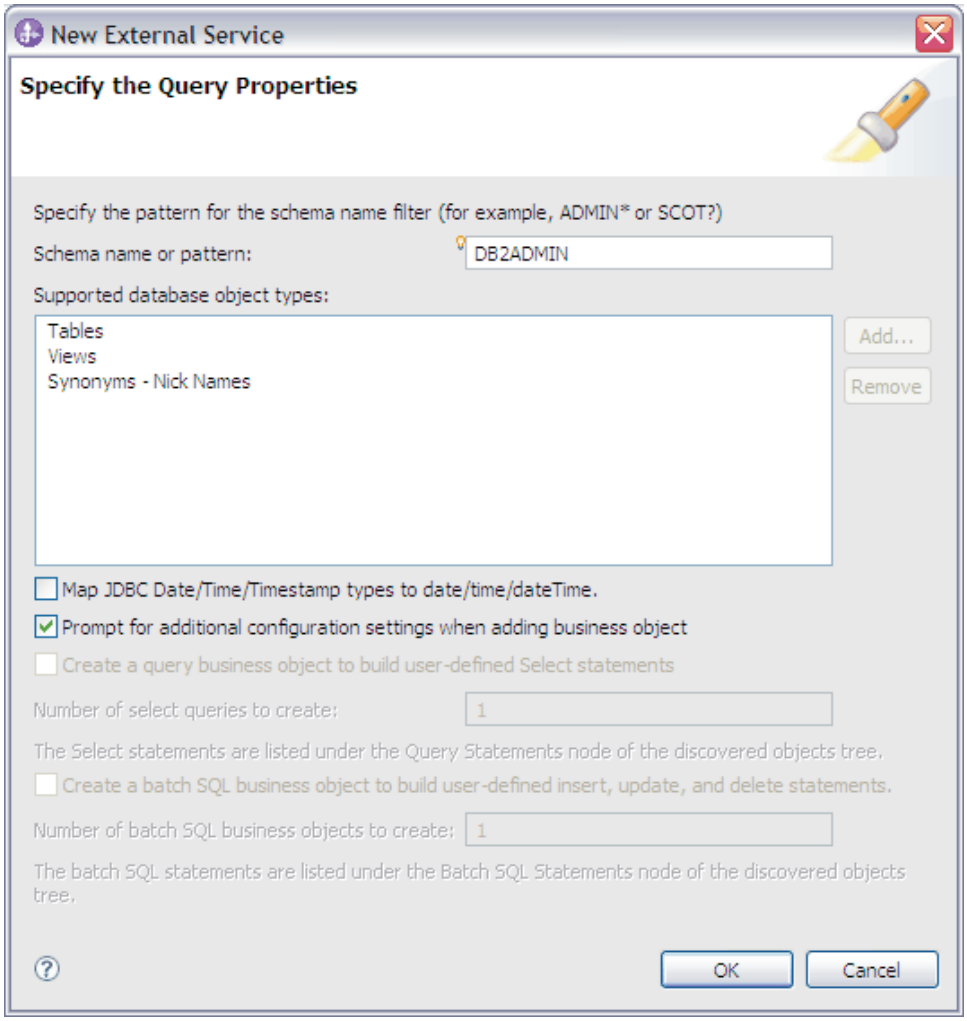
- これらのスキーマ内のどのタイプのデータベース・オブジェクトにアクセスする必要があるか

このタスクについて

この作業は、外部サービス・ウィザードの「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで開始します。

手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで、**「照会の編集 (Edit Query)」**をクリックします。 「照会プロパティーの指定」ウィンドウが表示されます。



注: 「ユーザー定義の select ステートメントを作成するための照会ビジネス・オブジェクトを作成する」および**「ユーザー定義の insert、update、および delete ステートメントを作成するためのバッチ SQL ビジネス・オブジェクトを作成する」**の各オプションは、Outbound 処理にのみ使用できます。

「照会プロパティーの指定」ウィンドウを使用して、次のタスクを実行します。

- データベース・スキーマのサブセットを検索することにより、検索時間を削減します。

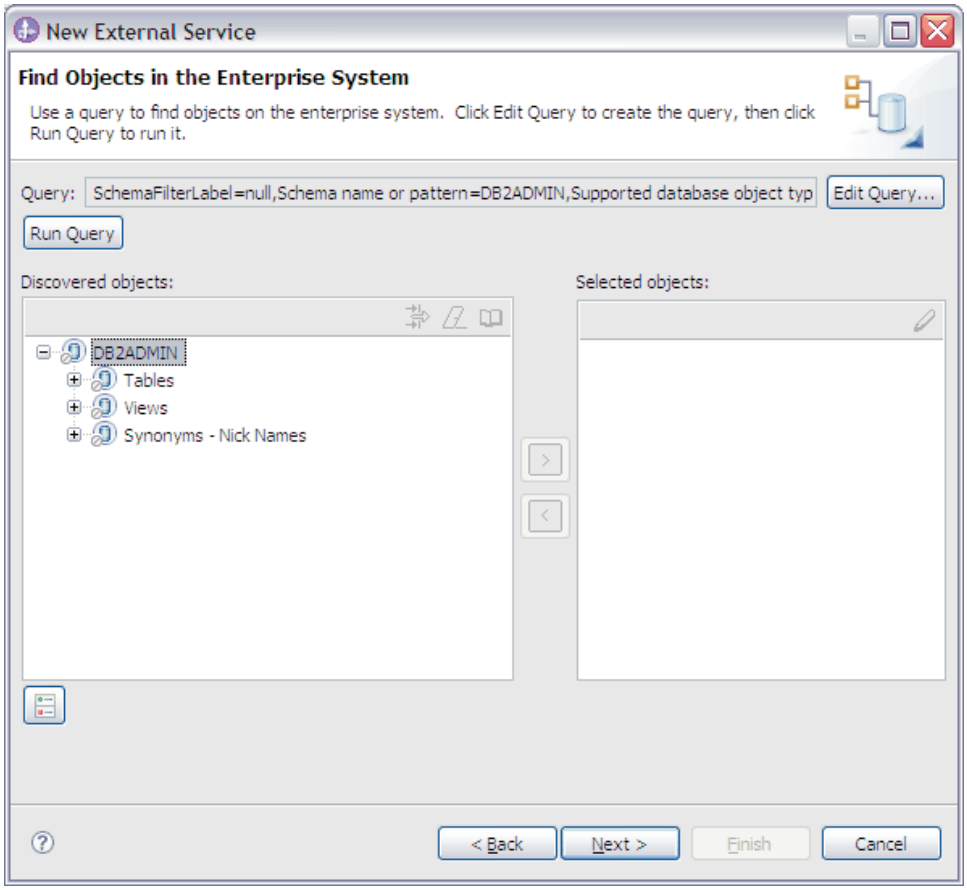
- 1 つ以上のタイプのデータベース・オブジェクトを検索から除外します。
- データベース内の情報からは自動的に判別できないアプリケーション固有情報の入力をウィザードがユーザーに求めるようにします。
- JDBC のデータ型 Date、Time、および Timestamp を date、time、および dateTime にマップします。

2. 取得されるデータベース・スキーマの数を制限するには、「**スキーマ名またはパターン**」にスキーマの名前またはスキーマ名パターンを入力します。1 文字に対応させる場合は疑問符または下線 (? または _) を使用し、複数文字に対応させる場合はアスタリスクまたはパーセント記号 (* または %) を使用します。 照会を実行すると、そのストリングで始まるスキーマか、またはスキーマ名パターンに一致するスキーマのみが表示されます。スキーマ名パターンを指定しない場合は、データベース内のすべてのスキーマが表示されます。データベースに多数のスキーマがある場合は、フィルターを使用してディスカバリー・プロセスを高速化できます。
3. 検索から 1 つ以上のタイプのオブジェクトを除外するには、除外したいオブジェクトのタイプ (テーブル、ビュー、およびシノニムまたはニックネーム) を「**サポートされるデータベース・オブジェクト・タイプ**」フィールドで選択して、「**除去**」をクリックします。そのオブジェクト・タイプをもう一度追加するには、「**追加**」をクリックします。 アクセスする必要のないオブジェクト・タイプがデータベースに含まれている場合は、それらを除外すると、ディスカバリー・プロセスを高速化できます。
4. データ型が Date、Time、および Timestamp であるテーブル・オブジェクトは、デフォルトでは String データ型にマップされます。これらのオブジェクトを、JDBC ドライバーでサポートされる実際のデータ型 (Date、Time、Datetime など) にマップするには、「**JDBC の Date/Time/Timestamp の型を date/time/dateTime にマップ**」チェック・ボックスを選択します。

   注: デフォルトのデータ型マッピングは、JDBC ドライバーのバージョンによって異なります。例えば、Oracle JDBC ドライバーを使用する場合、Date データ型は、Date ではなく dateTime データ型にマップされます。このような場合は、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウで、適切なデータ型を手動で選択する必要があります。
5. 「**ビジネス・オブジェクトの追加時に追加構成設定のプロンプトを出す**」チェック・ボックスを選択します。 これにより、作成するビジネス・オブジェクトのリストにデータベース・オブジェクトを追加すると、そのオブジェクトに関してユーザーが構成可能なすべてのアプリケーション固有情報を入力するよう求められます。例えば、このオプションを選択した場合は、ビジネス・オブジェクトの単純親子階層を作成するプロセスが、順を追ってウィザードで示されます。 2 つの異なるテーブルの属性を参照する 2 つの属性をテーブル・ビジネス・オブジェクトが持つ (すなわち、2 つの親ビジネス・オブジェクトを持つ) 階層が必要な場合は、アセンブリー・エディターで構成を実行します。このエディターは、IBM Integration Designer から起動されるツールです。また、データベースで外部キー参照が定義されている場合は、アダプターはテーブル間の親子関係を自動的にディスカバーして表示します。

重要: このオプションを選択しない場合、ウィザードは必須情報のみを入力するようプロンプトを出します。アセンブリー・エディターを使用して、ビジネス・オブジェクトの構成を実行する必要があります。また、データベースで外部キー参照を定義していない場合、アダプターは親子関係を自動的には生成しません。

6. 「**OK**」をクリックして、照会への変更を保存します。
7. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで、「**照会の実行 (Run Query)**」をクリックします。これによりこの照会を使用してデータベース・オブジェクトがディスカバーされます。 標準的な照会の実行結果を次の図に示します。



「**ディスカバーされたオブジェクト**」ペインには、ディスカバーされたオブジェクトがリストされます。テーブル、ビュー、シノニム/ニックネームは、スキーマ名によってソートされます。

8. 「**ディスカバーされたオブジェクト**」リストで、スキーマ・ノードおよびその下の「**テーブル**」、「**ビュー (Views)**」、「**シノニム - ニックネーム (Synonyms - Nicknames)**」の各ノードを展開するには、「**+**」 (正符号) をクリックします。これにより、ウィザードによってディスカバーされたデータベース・オブジェクトが表示されます。

タスクの結果

アダプターを使用してアクセスできるデータベース・オブジェクトがウィザードによってディスカバーされました。

次のタスク

外部サービス・ウィザードでの作業を続行します。次の手順では、モジュールで使用するオブジェクトの選択、各ビジネス・オブジェクトの構成、およびビジネス・オブジェクトの階層の作成を行います。

ビジネス・オブジェクトの選択および構成

外部サービス・ウィザード でディスカバーされたデータベース・オブジェクトのリストと、指定した照会オブジェクト・テンプレートおよびバッチ SQL オブジェクト・テンプレートを使用することにより、引き続きこのウィザードを使用して、モジュールでアクセスする必要のあるデータベース・オブジェクトを選択します。次に、新規ビジネス・オブジェクトの構成情報を入力します。

このタスクについて

「エンタープライズ・システムでのオブジェクトの検索」ウィンドウでは、オブジェクトを任意の順序で選択して構成できます。ただし、1 つだけ例外があり、親テーブルを選択して構成した後で、その子テーブルを選択して構成する必要があります。この制限を除けば、オブジェクトを個々に追加することも、複数のオブジェクトを一度に追加することも柔軟に行えます。**「ディスカバーされたオブジェクト」**リストのさまざまなノード内のオブジェクトを組み合わせることができます。例えば、テーブルおよびビュー・オブジェクト数個とストアード・プロシージャー・オブジェクトを選択して、それらを同時に追加できます。

ビジネス・オブジェクトの選択および構成のおおまかな流れは以下のとおりです。

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの**「検出オブジェクト (Discovered objects)」**リストで 1 つ以上のオブジェクトを選択します。
2. **「>」** (追加) ボタンをクリックします。
3. 「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが開きます。
   - 1 つのオブジェクトを選択すると、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが 1 つ表示されます。

     このウィンドウで、ユーザーが構成可能な属性や、ウィザードによるデータベース検査ではディスカバーできないその他の情報をすべて入力したら、**「OK」**をクリックして構成を保存します。
   - 複数のオブジェクトを選択すると、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが、選択したオブジェクトごとに 1 ページずつ表示されます。

     各オブジェクトの名前を順にクリックします。ウィンドウには、該当するオブジェクトを個別に選択した場合と同じ情報が表示されます。

     重要: すべてのオブジェクトの構成ページで操作が完了するまでは、**「OK」**はクリックしないでください。ウィザードは、すべての必須フィールドに値が指定されるまではノートブックを閉じません。ただし、ユーザーは、オプション・フィールドに値を指定する前にウィンドウを閉じることができます。ウィ

ザードでオプション・フィールドを構成しない場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後にそれらのフィールドを構成する必要があります。

4. ウィザードにより、構成されたオブジェクトが「**選択済みオブジェクト (Selected objects)**」リストに追加されます。

ウィザードを終了しない限り、操作を繰り返してモジュールに必要なビジネス・オブジェクトを選択および構成できます。ただし、ウィザードを開始して既存のモジュールにオブジェクトを追加する前に、ビジネス・オブジェクトを使用するプログラムの要件を十分に理解してください。ウィザードによって、同じパス内の既存のビジネス・オブジェクトが上書きされます。

Inbound 処理のテーブル、ビュー、およびシノニムまたはニックネームの選択および構成:

モジュールで使用するテーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを選択および構成します。Inbound 処理の場合、これらはイベントで送達されるビジネス・オブジェクトです。

始める前に

このタスクを実行するには、データベース内のデータの構造や、モジュールがどんなデータベース・オブジェクトにアクセスする必要があるかを理解しなければなりません。具体的には、以下の情報を認識しておく必要があります。

- テーブル、ビュー、およびシノニムまたはニックネームの構造 (必要な列や、データ型などの列属性を含む)。
- テーブル間の関係 (親子関係のカーディナリティーおよび所有権を含む)。

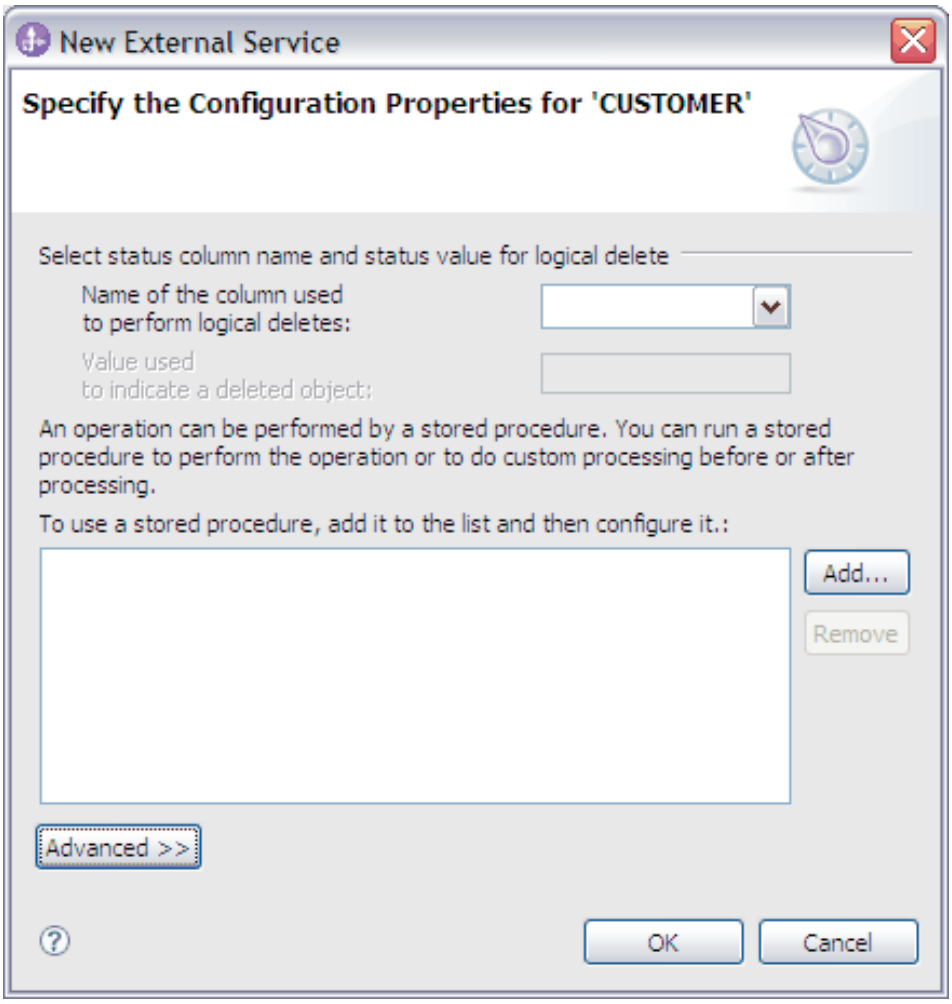
このタスクについて

このタスクは、外部サービス・ウィザードを介して実行されます。「エンタープライズ・システムでのオブジェクトの検索」ウィンドウから操作を開始し、「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

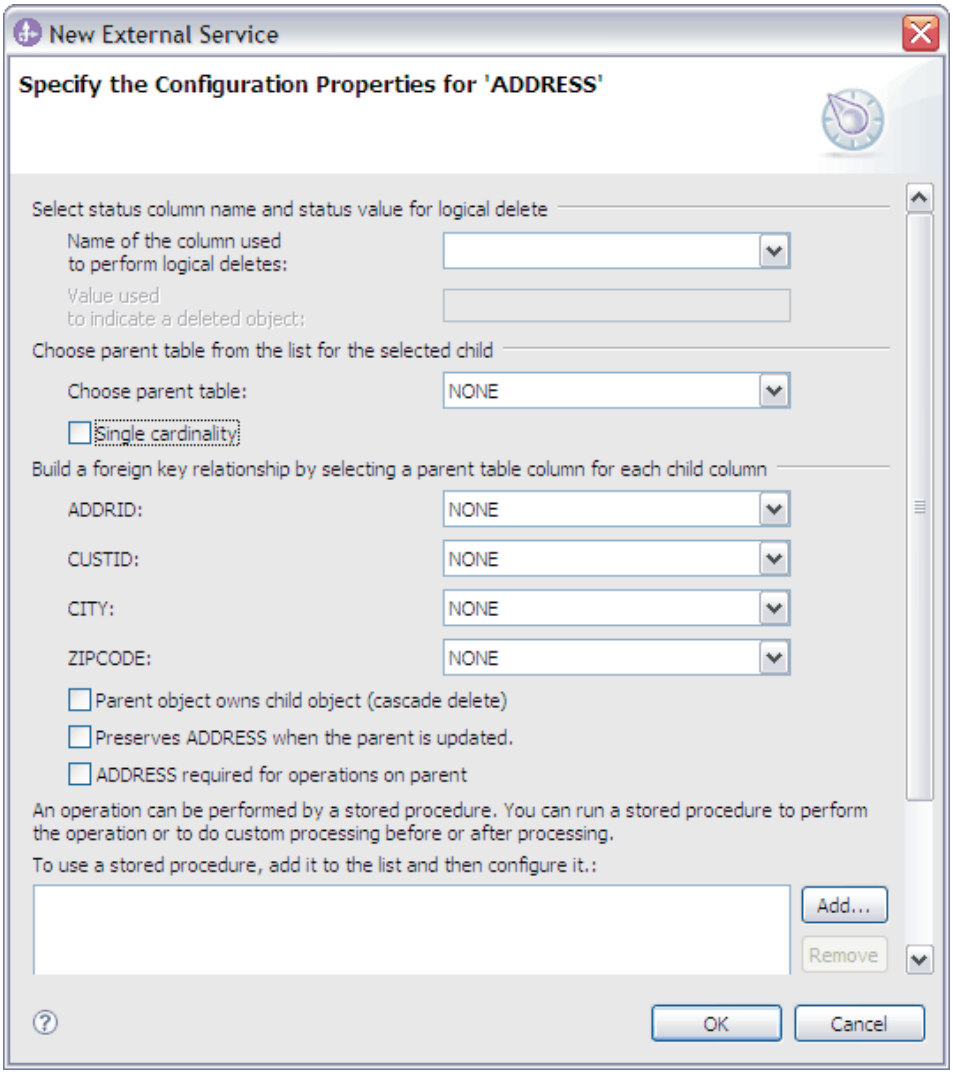
手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの「**ディスカバーされたオブジェクト**」リストで、テーブル、ビュー、またはシノニムを 1 つ以上選択し、「**>**」 (追加) ボタンをクリックします。オブジェクトが「**選択済みオブジェクト**」リストに追加されます。

   以下の 2 つの図に、ビジネス・オブジェクト (テーブル、ビュー、シノニム、またはニックネーム) の標準的な「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウを示します。最初の図に、選択する最初のテーブルまたはテーブル・グループの標準的なウィンドウを示します。



以下の図に、選択する後続のテーブルの標準的なウィンドウを示します。少なくとも 1 つのテーブルを選択して構成した後は、後続テーブルの「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウに、テーブル間の親子階層をオプションで定義することができる領域が表示されます。



オブジェクトの構成時には、拡張構成を必要とする選択を行うと、このウィンドウに追加のフィールドが表示され、ウィンドウがスクロールされる場合があります。必ずウィンドウのすべてのフィールドを確認してから、「**OK**」をクリックしてください。

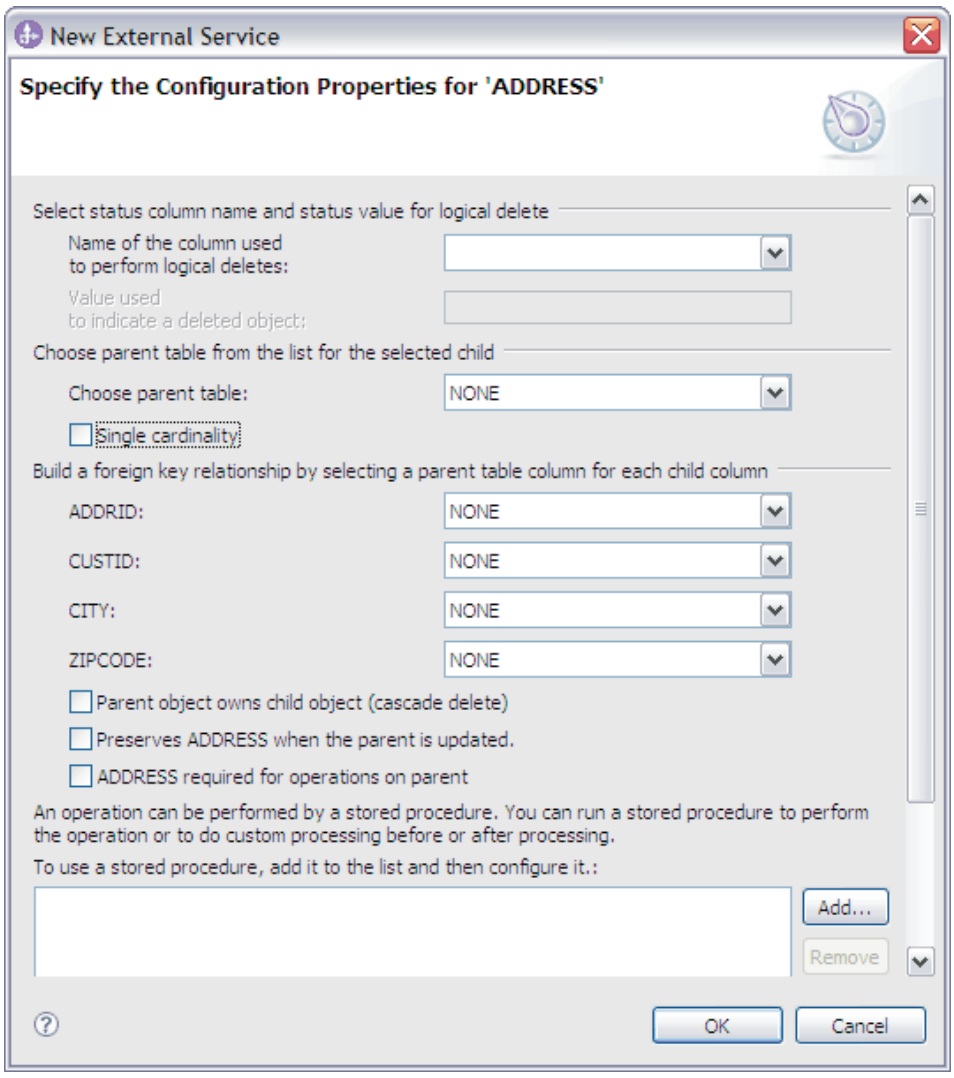
2. 論理削除を示すのに使用される列がテーブルにある場合は、次の手順に従います。
   a. 「**論理削除を実行するのに使用される列の名前**」フィールドで列名を選択します。
   b. 「**削除されたオブジェクトを示すために使用する値**」フィールドに、行が論理的に削除されていることを示す値を入力します。 この値については、データベース管理者に確認できます。
3. 「**テーブル *table_name* の基本キーの選択**」エリアが表示されたら、「**追加**」をクリックし、テーブル・ビジネス・オブジェクトの基本キーとして使用する列を選択してから、「**OK**」をクリックします。テーブルに複合キーがある場合は、複数の列を選択できます。 「**テーブル *table_name* の基本キーの選択**」エリアは、データベース表に基本キーとして指定された列が存在しない場合にのみ表示されます。各テーブル・ビジネス・オブジェクトには、関連付けられた

データベース表にキーがない場合でも、基本キーが定義されている必要があります。データベースで基本キーが定義されている場合、ウィンドウのこのセクションは表示されません。

4. オプション: ビジネス・オブジェクト間の親子関係を定義します。

   親子階層を作成する場合、まず親テーブルを構成して「エンタープライズ・システムでのオブジェクトの検索」ウィンドウに戻り、子テーブルを選択して構成します。

   以下の図に示す「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウの領域を使用して、親子関係を構成します。これらのフィールドは、構成する最初のテーブルの場合には表示されません。



   a. 「**親テーブルの選択**」フィールドで、構成する親テーブルの名前を選択します。 リストに親テーブルが表示されていない場合は、親テーブルがまだ構成されていません。戻って親オブジェクトを構成してから、子オブジェクトを構成してください。データベースで外部キー参照を定義した場合、親テーブルを選択すると、アダプターはテーブル間の親子関係を自動的にディスカ

バーして表示します。テーブルとその親のテーブル間に単一カーディナリティー関係がある場合は、「**単一カーディナリティー**」チェック・ボックスが自動的に選択されます。

b. 関係のカーディナリティーを指定します。
   - テーブルとその親テーブルの間に単一カーディナリティー関係がある場合は、「**単一カーディナリティー**」チェック・ボックスを選択します。単一カーディナリティー関係では、親はこのタイプの子ビジネス・オブジェクトを 1 つのみ持つことができます。単一カーディナリティー関係は、所有関係を伴って実際の子を表すか、または所有関係を伴わずにルックアップ・テーブルまたはデータベース内の他の対等オブジェクトを表すために使用できます。
   - テーブルに複数カーディナリティー関係がある場合は、「**単一カーディナリティー**」チェック・ボックスを選択しないでください。複数カーディナリティー関係では、親がこのタイプの子ビジネス・オブジェクトの配列を持つことができます。

c. 親と子の間に外部キー関係を作成するため、子の列ごとに、親テーブルの外部キーであるかどうかを指定します。
   - 子の列が外部キーでない場合は、「なし」を選択します。
   - 子の列が外部キーの場合は、その子の列に対応する親テーブルの列を選択します。

     注: ウィザードは、1 つの親テーブルのみを構成できます。子テーブルに複数の親テーブルがある場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後に残りの親テーブルを構成する必要があります。

d. 親オブジェクトが子オブジェクトを所有している場合、データベース内の子オブジェクトは親が削除されるときに削除されます。この子がその親によって所有されていることを示すには、「**親オブジェクトが子オブジェクトを所有する (カスケード削除)**」チェック・ボックスを選択します。あるいは、このオプションをクリアして、ルックアップ・テーブルなどの子オブジェクトが、親の削除時に削除されないようにします。

e. Update 操作の一環として子オブジェクトが削除されることのないようにするには、「**親の更新時に *child_table_name* を保持する**」チェック・ボックスを選択します。

   親テーブルが更新されると、アダプターは入力に存在する子ビジネス・オブジェクトを、データベースから返される子ビジネス・オブジェクトと比較します。デフォルトでは、アダプターは、入力ビジネス・オブジェクト内に存在しない、データベースから返されたすべての子オブジェクトを削除します。

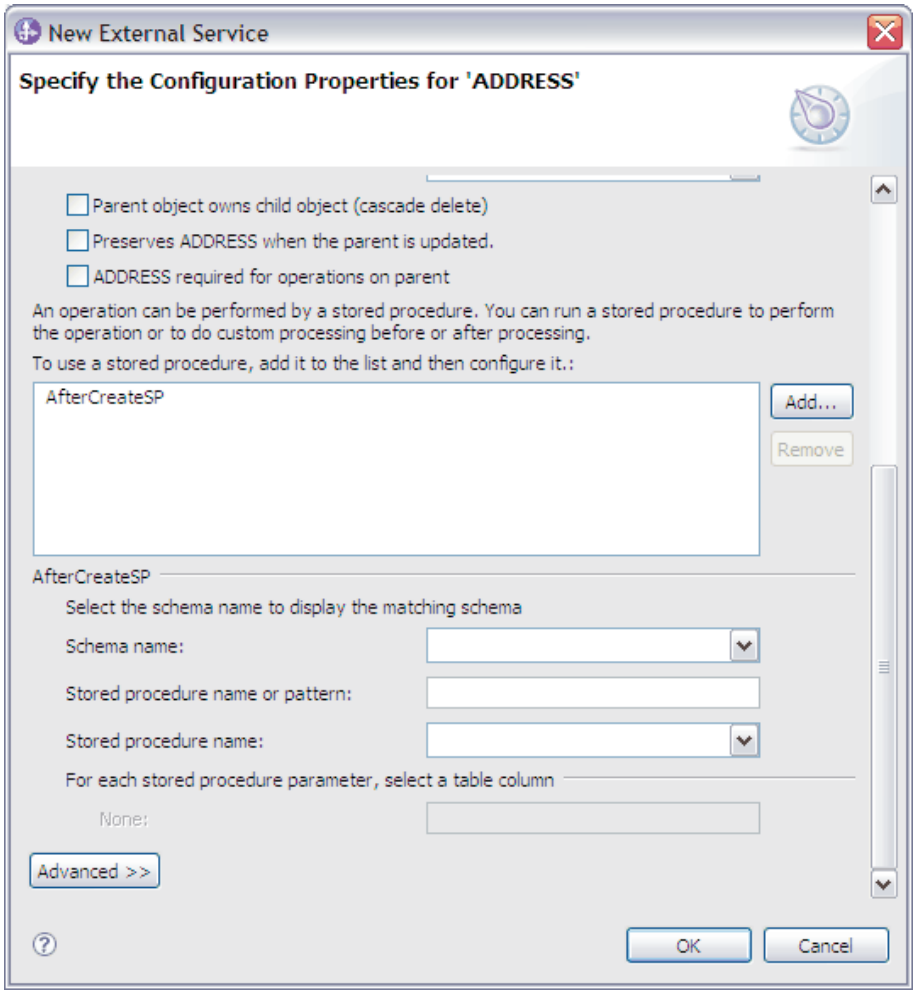
f. デフォルトでは、子ビジネス・オブジェクトを指定せずに、親ビジネス・オブジェクトに対して操作を実行できます。親ビジネス・オブジェクトを変更対象として実行依頼するときに、その親ビジネス・オブジェクトで子ビジネス・オブジェクトが必ず指定されるようにしたい場合は、「***Child_table_name* は、親に対する操作で必須**」チェック・ボックスを選択します。

5. 操作を実行するには、アダプターによって生成される標準 SQL ステートメントを使用するか、あるいはデータベース内のストアード・プロシージャーまたはストアード関数を使用します。ストアード・プロシージャーまたはストアード関数を使用する場合は、以下の手順を実行します。
   a. 「**追加**」をクリックします。
   b. 「追加」ウィンドウで、実行するストアード・プロシージャーのタイプを選択します。 操作ごとに、その操作を実行するストアード・プロシージャーと、操作の前後に実行するストアード・プロシージャーを選択できます。例えば、Create 操作の場合は、ストアード・プロシージャー CreateSP、BeforeCreateSP、および AfterCreateSP のどれでも指定できます。

      注: RetrieveAllSP を指定してテーブルを構成する場合は、ストアード・プロシージャーが 1 つの結果セットのみを返すことを確認します。ストアード・プロシージャーの ResultSet ASI を `true` に設定して、実行時に例外「ストアード・プロシージャーに関連した結果セットが見つかりませんでした (`No resultset found associated with the stored procedure`)」、「結果セットが返されませんでした (`No resultset returned`)」、 「複数の結果セットが返されました (`More than one resultset returned`)」のいずれも生成されないようにします。

      注:

      1) Oracle データベースの場合、WebSphere Adapter for JDBC は、 Cursor タイプの OUT パラメーターを使用するストアード・プロシージャーのみをサポートします。Cursor タイプの IN または INOUT パラメーターを使用するストアード・プロシージャーはサポートしません。
      2) DB2 データベースおよび MSSQLServer データベースの場合、WebSphere Adapter for JDBC は、 Cursor タイプの IN、OUT、および INOUT パラメーターを使用するストアード・プロシージャーをサポートしません。
   c. 「**OK**」をクリックします。選択したストアード・プロシージャーのタイプが「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウに表示されます。このウィンドウは、各ストアード・プロシージャーの構成用の領域を表示するために拡張されます。新しい領域を表示するには、スクロールダウンする必要がある場合もあります。



注: 階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、トップレベルのビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーをトップレベルのビジネス・オブジェクトに関連付けても、各子ビジネス・オブジェクトに関連付けないと、そのトップレベルのビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

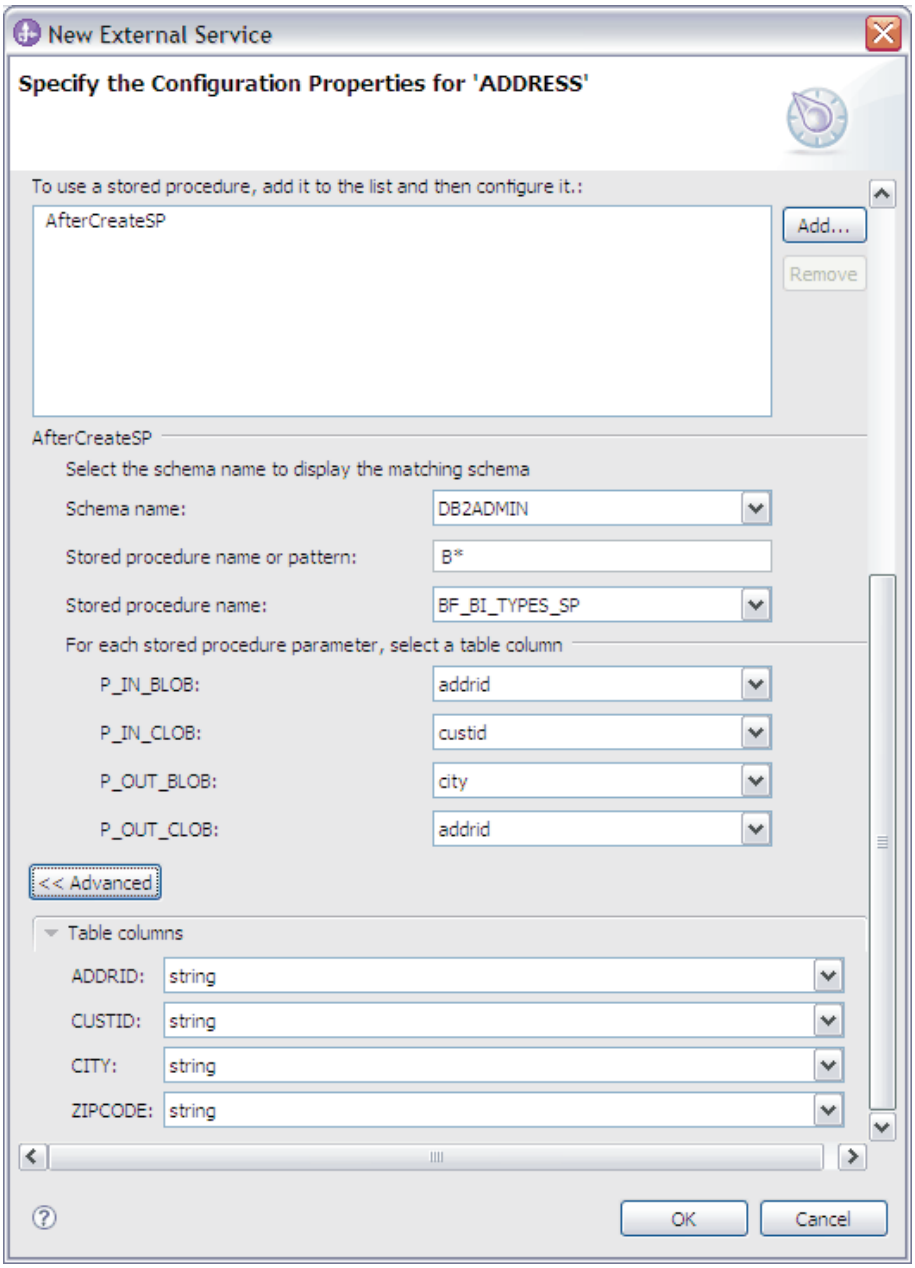
6. 選択したストアード・プロシージャー・タイプごとに、データベース内でのストアード・プロシージャーの名前を指定し、ビジネス・オブジェクトを構成します。
   a. 「**スキーマ名**」フィールドで、ストアード・プロシージャーが含まれるスキーマの名前を選択します。
   b. ストアード・プロシージャーまたはストアード関数の名前を指定します。
      1) 「**ストアード・プロシージャー名またはパターン**」フィールドで、ストアード・プロシージャーまたはストアード関数の名前を入力するか、または名前パターンを入力します。1 つの文字と一致させる場合は疑問符

または下線 (? または _) を使用し、複数の文字と一致させる場合はアスタリスクまたはパーセント記号 (* または %) を使用します。

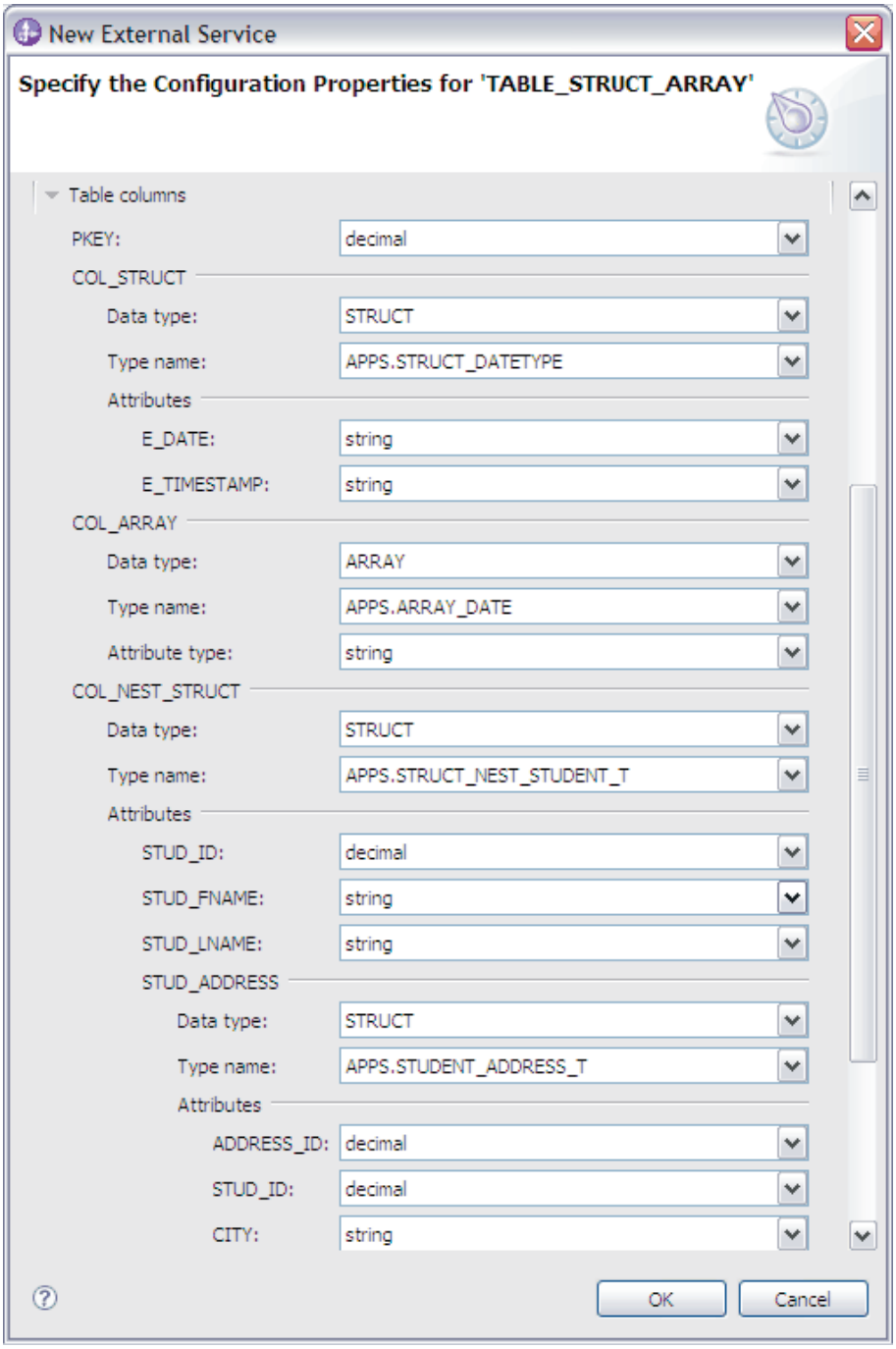
2) 「**ストアード・プロシージャー名**」フィールドで、目的のプロシージャーの名前を選択します。

「「オブジェクト」の構成プロパティーの指定 (Specify the Configuration Properties for 'object')」ウィンドウが拡張して、ストアード・プロシージャーを構成するための領域が表示されます。ウィザードは、データベース内のストアード・プロシージャーを調べることにより、パラメーターのリストを自動生成します。

c. ストアード・プロシージャーのパラメーターごと (左側) に、そのパラメーターでストアード・プロシージャーに渡すテーブル列 (右側) を選択します。 次の図に、ストアード・プロシージャーを構成した後のウィンドウの一部を示します。



7. テーブル内の各列のデータ型マッピングを指定するには、以下の手順を実行します。
   a. 「**拡張**」をクリックします。
   b. 「**テーブル列**」を展開します。テーブル内の列ごとに、デフォルトのデータ型マッピングが表示されます。 Oracle データベースにおいて、配列、構造体、ネストされた構造体、テーブルなど、何らかのユーザー定義型または複合データ型がテーブルに含まれている場合は、型名および子属性の詳細も自動的にディスカバーされて、表示されます。次の図に、複合データ型を含む Oracle テーブルの型名および子属性の詳細を示します。



c. マッピングを確認して、必要な場合は変更します。

注: テーブル内の基本キーのデータ型が Date または Timestamp である場合、event_table 内の object_key のフォーマットは「yyyy-mm-dd hh-mm-ss」である必要があります。

8. ウィンドウのすべてのフィールドの操作が完了したら、「**OK**」をクリックします。ビジネス・オブジェクトの構成が保存されます。 定義したビジネス・オブジェクト (テーブル、ビュー、シノニム、およびニックネーム) が、「エンタープライズ・システムでのオブジェクトの検索」ウィンドウにリストされます。

9. 「**選択済みオブジェクト**」リストのオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。
10. 必要なすべてのビジネス・オブジェクトを選択および構成したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

次のタスク

エンタープライズ・システムでのオブジェクトの検索ウィンドウで、他のタイプのビジネス・オブジェクトの選択と構成を続行します。完了したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

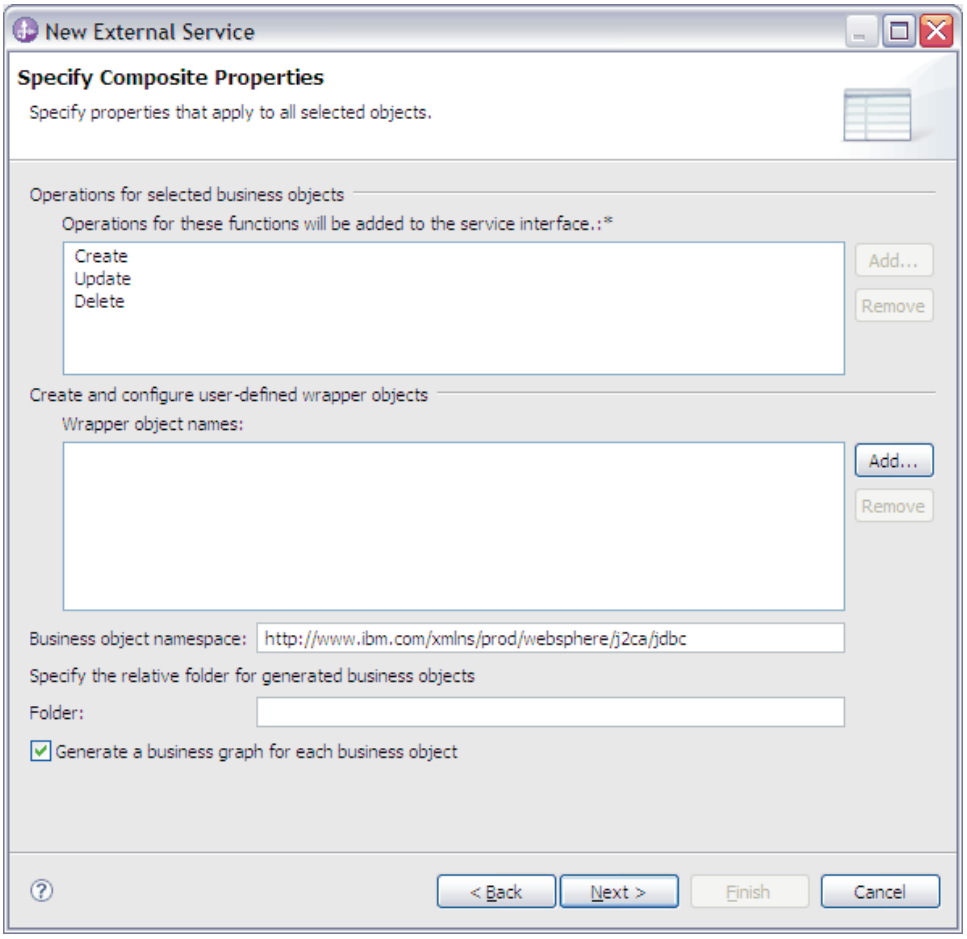
操作のグローバル・プロパティーの設定およびラッパー・ビジネス・オブジェクトの作成

外部サービス・ウィザードでデータベース・オブジェクトを選択した後は、すべてのビジネス・オブジェクトに適用するプロパティーを指定する必要があります。

手順

1. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウの「**選択済みオブジェクト**」リストに、アプリケーションで使用するすべてのビジネス・オブジェクト (ラッパー・ビジネス・オブジェクトを除く) が含まれている場合は、「**次へ**」をクリックします。
2. 「複合プロパティーの指定」ウィンドウで、操作リストを確認します。このリストには、アダプターが Inbound サービスでサポートする操作が含まれています。操作リストへの追加には、前のウィンドウで選択したすべてのビジネス・オブジェクトの操作が含まれます。

   指定された操作は、生成されるすべてのビジネス・オブジェクトに対して設定されます。



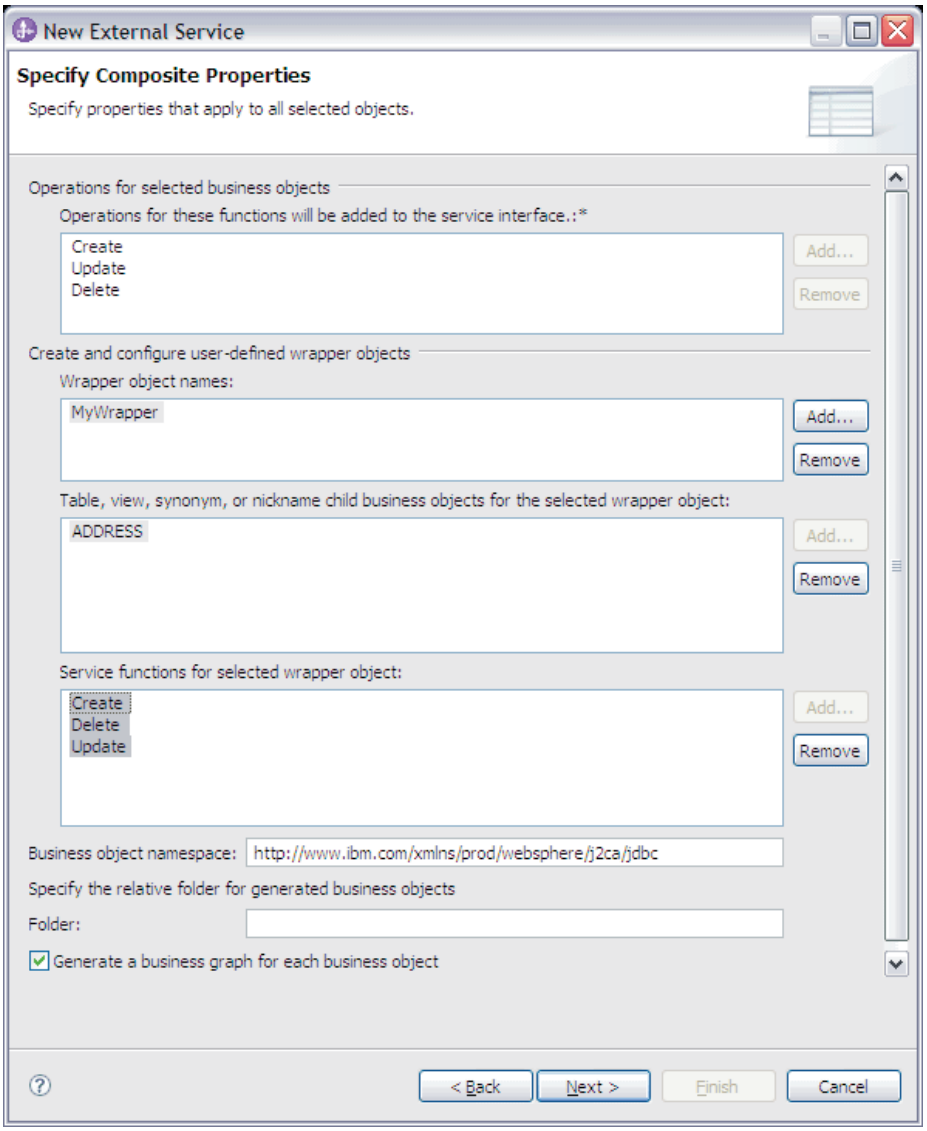
3. 不要な操作を除去するには、その操作名を選択して「**除去 (Remove)**」をクリックします。 削除した操作を復元する場合は、「**追加**」をクリックして、除去した操作を復元します。
4. ラッパー・ビジネス・オブジェクトを作成するため、以下の手順を実行します。

   注: テーブルに複数の基本キーがあり、このテーブルから複数のレコードを取得したい場合は、ラッパー・オブジェクトを使用できます。基本キー列を設定解除するには、unsetValueKeyWord プロパティーを使用します。

   a. 「**Wrapper オブジェクト名**」領域で、「**追加**」をクリックします。
   b. 「値の追加」ウィンドウに、ラッパー・ビジネス・オブジェクトの名前を入力して、「**OK**」をクリックします。 スペースを使用しないでください。名前には、各国語文字を使用できます。
   c. 「**選択した wrapper オブジェクトのテーブル、ビュー、シノニム、またはニックネーム子ビジネス・オブジェクト**」エリアで、「**追加**」をクリックします。
   d. 「値の追加」ウィンドウで、wrapper に組み込むビジネス・オブジェクトを 1 つ以上選択して、「**OK**」をクリックします。
   e. 「**選択した wrapper オブジェクトのサービス機能**」エリアで、「**追加**」をクリックします。

f. 「 値の追加」ウィンドウで、wrapper オブジェクトで実行する操作を 1 つ以上選択して、「**OK**」をクリックします。

g. 作成するラッパー・ビジネス・オブジェクトごとに、この手順を繰り返します。 次の図に、1 つのラッパー・ビジネス・オブジェクトが定義されている「複合プロパティーの指定」ウィンドウを示します。



5. 「**ビジネス・オブジェクト名前空間**」フィールドで、デフォルトの名前空間を受け入れるか、または別の名前空間の完全な名前を入力します。

   名前空間は、ビジネス・オブジェクト・スキーマを論理的に分離するため、ビジネス・オブジェクト名の前に付加されます。このプロパティーについて詳しくは、 367 ページの『ビジネス・オブジェクト名前空間 (BONameSpace)』を参照してください。

6. オプション: 「**フォルダー**」フィールドに、生成されたビジネス・オブジェクトを格納するフォルダーの相対パスを入力します。

   注: 1 つのモジュール内に複数のアダプター成果物を作成する場合は、モジュール内の各アダプターに対して、別々のビジネス・オブジェクト・フォルダーを指

定するようにしてください。例えば、1 つのモジュール内に Oracle、JDBC、SAP、および JDE 用の成果物を作成する場合は、それらの各アダプターに対して、別々の相対フォルダーを作成する必要があります。別々の相対フォルダーを指定していない場合、新規成果物を生成すると、既存の成果物が上書きされます。

7. ビジネス・オブジェクトごとにビジネス・グラフを作成する場合は、「**ビジネス・オブジェクトごとにビジネス・グラフを生成**」チェック・ボックスを選択します。 ビジネス・グラフは、バージョン 7.0 より前のバージョンの IBM Integration Designer で作成されたモジュールにビジネス・オブジェクトを追加する場合のみ必要になります。

   注: 前のバージョンの IBM Integration Designer によって作成されたモジュールにビジネス・オブジェクトを追加する場合は、このオプションを選択する必要があります。それ以外の場合は、インターフェースを再接続する必要があります。
8. 完了したら、「**次へ**」をクリックします。

タスクの結果

モジュール内のすべてのビジネス・オブジェクトに適用する情報を設定しました。

次のタスク

ウィザードでの作業を続行します。次の手順では、実行時に使用するデプロイメント情報、およびサービスをモジュールとして保存するための情報を指定します。

デプロイメント・プロパティーの設定およびサービスの生成

モジュールのビジネス・オブジェクトを選択して構成した後、外部サービス・ウィザードを使用して、アダプターが特定のデータベースに接続するために使用するプロパティーを構成します。ウィザードは、すべての成果物とプロパティー値を保存する、新規のビジネス・インテグレーション・モジュールを作成します。

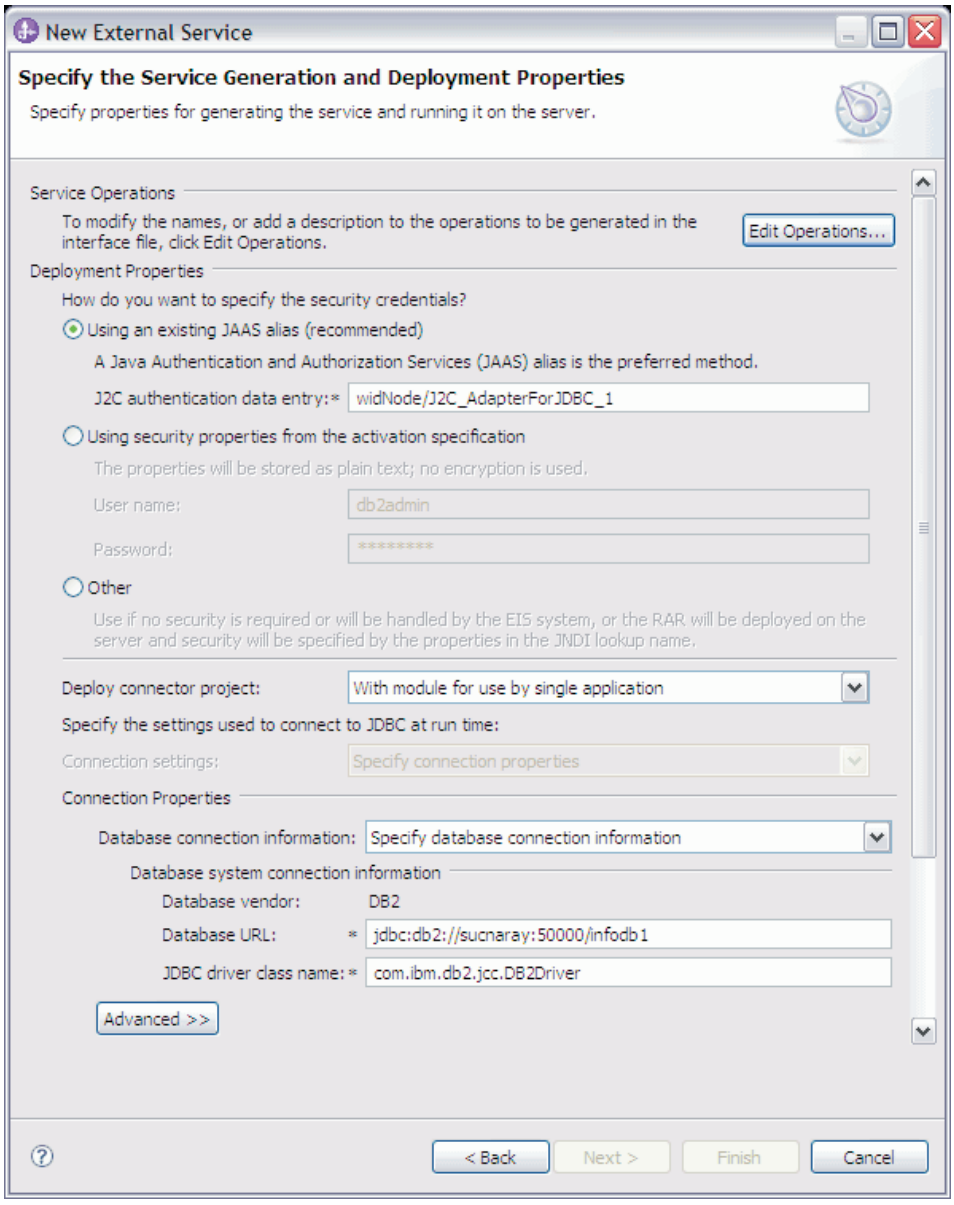
このタスクについて

このタスクは、「外部サービス・ウィザードのサービス生成およびデプロイメント・プロパティーの指定」ウィンドウおよび「ロケーション・プロパティーの指定」ウィンドウを介して実行されます。

このタスクの接続プロパティーは、ウィザードがデータベースに接続するために使用した値に初期化されます。他の値を使用するようにモジュールを構成するには、ここで値を変更します。例えば、実行時に IBM i で IBM Toolkit for Java ネイティブ・ドライバーを使用するには、ここでドライバー情報を設定します。

手順

1. 「サービス生成およびデプロイメント・プロパティーの指定」ウィンドウで、「**操作の編集**」をクリックして、作成するビジネス・オブジェクトの操作の名前を確認するか、その操作の説明を追加します。

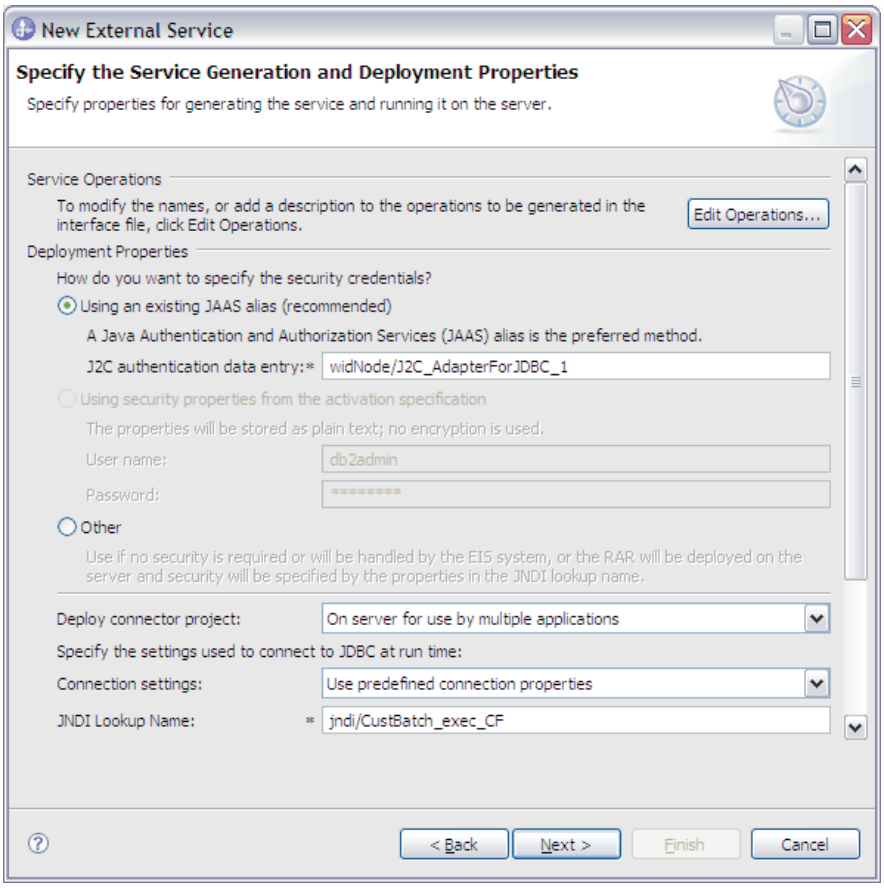


2. 「**デプロイメント・プロパティー**」エリアで、アダプターが実行時にユーザー名およびパスワードを取得する方法を指定します。
   - J2C 認証別名を使用するには、「**既存の JAAS 別名を使用する (推奨)**」をクリックして、「**J2C 認証データ項目**」フィールドに別名の名前を入力します。既存の認証別名を指定することも、(モジュールをデプロイする前に) 認証別名を作成することもできます。名前は大文字小文字が区別されます。また、名前にはノード名が含まれます。
   - 管理接続プロパティーを使用するには、「**管理接続ファクトリーのセキュリティー・プロパティーを使用**」をクリックして、「**ユーザー名**」フィールドと「**パスワード**」フィールドに値を入力します。
   - 他の手段でユーザー名とパスワードを管理するには「**その他**」をクリックします。

注: データベース接続の確立にローカル接続情報を使用する場合、セキュリティー資格情報が必要です。セキュリティー・メカニズムとして、「**既存の JAAS 別名を使用する (推奨)**」または「**管理接続ファクトリーのセキュリティー・プロパティーを使用**」のいずれかを選択できます。サーバー上の既存のデータ・ソースを使用する場合には、セキュリティー資格情報は必要ありません。この場合、セキュリティー・メカニズムとして「**その他**」を選択できます。また、「**J2C 認証データ項目**」フィールドを設定するか、「**ユーザー名**」フィールドと「**パスワード**」フィールドを設定すると、データ・ソースのユーザー名とパスワードがこれらのフィールドの値によって指定変更されます。

3. 「**コネクター・プロジェクトのデプロイ**」フィールドで、モジュールにアダプター・ファイルを組み込むかどうかを指定します。次の値のいずれかを選択してください。
   - **単一アプリケーションが使用するモジュールで (With module for use by single application)**: アダプター・ファイルをモジュール内に組み込むと、モジュールをすべてのアプリケーション・サーバーにデプロイすることができます。組み込みアダプターを使用するのは、組み込みアダプターを使用するモジュールが 1 つある場合か、複数のモジュールでバージョンの異なるアダプターを実行する必要がある場合です。組み込みアダプターを使用すると、他のモジュールのアダプター・バージョンを変更することで、それらのモジュールを不安定にするリスクを生じることなく、1 つのモジュール内でアダプターをアップグレードできます。
   - **複数アプリケーションが使用するサーバー上 (On server for use by multiple applications)**: モジュール内にアダプター・ファイルを組み込まない場合は、このモジュールを実行するアプリケーション・サーバーごとに、アダプター・ファイルをスタンドアロン・アダプターとしてインストールする必要があります。複数のモジュールが同じバージョンのアダプターを使用可能で、アダプターを中央の場所で管理する場合は、スタンドアロン・アダプターを使用します。スタンドアロン・アダプターの場合も、複数のモジュールに対して単一のアダプター・インスタンスを実行することにより、必要なリソースが軽減されます。
4. 前のステップで「**複数アダプターが使用するサーバー上 (On server for use by multiple adapters)**」を選択した場合は、実行時に接続する接続プロパティーを指定します。
   - サーバーで管理接続ファクトリーまたは活動化仕様を手動で作成および構成した場合、あるいは同じ管理接続ファクトリーまたは活動化仕様のプロパティーを使用して同じデータベースに接続するアプリケーションを既にデプロイ済みの場合は、その Java Naming and Directory Interface (JNDI) データ・ソースの名前を指定することによって、管理接続ファクトリーまたは活動化仕様を再利用できます。
     a. 「**接続設定 (Connection settings)**」リストで、「**事前定義された接続プロパティーを使用する**」を選択します。
     b. 「**JNDI ルックアップ名**」フィールドに、既存の管理接続ファクトリーまたは活動化仕様の JNDI データ・ソースの名前を入力します。

        以下の図に、アダプターのスタンドアロン・デプロイメントで管理接続ファクトリーまたは活動化仕様を再利用する場合の標準的な設定を示します。



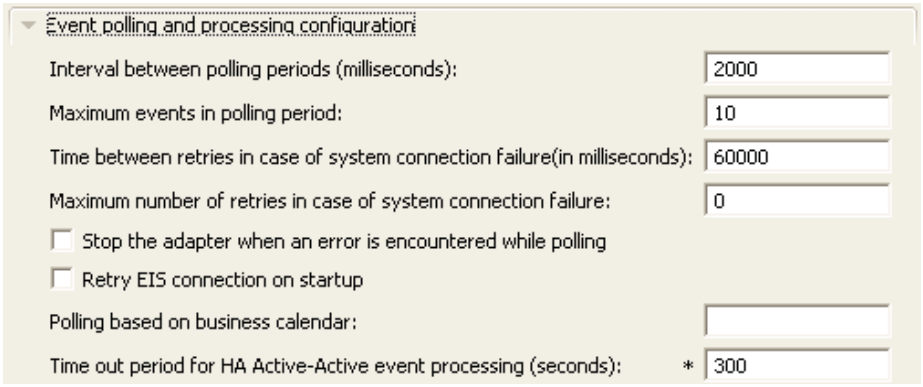
c. 「**次へ**」をクリックしてこのタスクを完了します。

- これが、特定のユーザー名とパスワードを使用してデータベースに接続する最初のアプリケーションである場合、または他のアプリケーションとは別々にユーザー名とパスワードを管理する場合は、「**接続プロパティーの指定**」を選択します。

5. 「**接続プロパティー**」エリアで、アダプターが実行時にデータベース接続を確立する方法を指定します。
   - サーバー上の既存のデータ・ソースを使用するには、以下の手順を実行します。
     a. 「**データベース接続情報**」リストから、「**事前定義 DataSource の指定**」を選択します。
     b. 「**データベース・システム接続情報**」エリアで、「**DataSource JNDI 名**」フィールドに既存の JNDI データ・ソースの名前を入力します。このプロパティーについて詳しくは、 370 ページの『データ・ソース JNDI 名 (DataSourceJNDIName)』を参照してください。
   - アダプター・プロパティーに保存される接続情報を指定するには、以下のようにします。
     a. 「**データベース接続情報**」リストから、「**データベース接続情報の指定**」を選択します。
     b. 「**データベース・システム接続情報**」エリアで、「**データベース URL**」フィールドと「**JDBC ドライバー・クラス名**」フィールドに値を入力しま

す。これらのプロパティーについて詳しくは、 371 ページの『データベース URL (DatabaseURL)』および 378 ページの『JDBC ドライバー・クラス (JDBC driver class) (JDBCDriverClass)』を参照してください。

6. 必須の接続プロパティーの値を確認し、必要に応じて変更します。 フィールドは、ウィザードの開始時に指定した接続情報で初期化されます。これらの値を変更して、別のユーザー名およびパスワードを実行時に指定できます。また、別のデータベースに接続できますが、スキーマ名は両方のデータベースで同じでなければなりません。接続プロパティーの形式はデータベース固有です。プロパティーについて詳しくは、 361 ページの『活動化仕様プロパティー』を参照してください。
7. オプション: 「**拡張**」をクリックして拡張プロパティーを指定します。 拡張セクションをそれぞれ展開して、プロパティーを確認します。
   - **イベント・ポーリングおよび処理の構成**



   a. 「**ポーリング期間の間隔 (ミリ秒単位)**」フィールドに、アダプターがポーリング期間から次の期間まで待機する時間 (ミリ秒) を入力します。このプロパティーについて詳しくは、 381 ページの『ポーリング期間の間隔 (PollPeriod)』を参照してください。
   b. 「**ポーリング期間内の最大イベント数**」フィールドに、各ポーリング期間で送達するイベント数を入力します。このプロパティーについて詳しくは、 382 ページの『ポーリング期間内の最大イベント数 (ポーリング数量)』を参照してください。
   c. 「**システム接続に失敗した場合の再試行間隔 (ミリ秒)**」フィールドに、アダプターがポーリング中に接続が失敗してから接続を試行するまで待機する時間 (ミリ秒) を入力します。詳しくは、 383 ページの『接続が失敗した場合の再試行間隔 (RetryInterval)』を参照してください。
   d. 「**システム接続に失敗した場合の最大再試行回数**」フィールドに、アダプターがポーリング・エラーを報告するまでに接続を再試行する回数を入力します。詳しくは、 383 ページの『システム接続を再試行する回数 (RetryLimit)』を参照してください。
   e. ポーリング・エラーが発生したらアダプターを停止するようにしたい場合は、「**ポーリング時にエラーが検出された場合はアダプターを停止する**」チェック・ボックスを選択します。このオプションを選択しない場合、アダプターは例外をログに記録しますが、稼働し続けます。詳しくは、 385 ページの『ポーリング時にエラーが検出された場合はアダプターを停止する (StopPollingOnError)』を参照してください。

f. 始動時に、失敗した接続をアダプターに再試行させる場合は、「**開始時に EIS 接続を再試行する**」チェック・ボックスを選択します。詳しくは、383 ページの『開始時に EIS 接続を再試行する (RetryConnectionOnStartup)』を参照してください。

g. Inbound アクティビティーのポーリングをカレンダーに基づいて作成するには、「カレンダー・ベースのスケジューリング」オプションを選択します。 IBM Integration Designer で新規カレンダーを作成する際に、ビジネス・アクティビティーをスケジュールすることができます。カレンダー・ベースのスケジューリング機能を使用するオプションは、ツール環境として IBM Integration Designer を使用している場合にのみ有効です。モジュールまたはライブラリー用に、ブランクのカレンダーを選択するか、または新しいカレンダーを作成することができます。ブランクのカレンダーを選択する場合は、定義済みの時間間隔を設定することはできません。ユーザー固有の時間間隔を定義する必要があります。事前定義されたテンプレートを使用してカレンダーを作成する場合は、テンプレートごとに時間間隔を定義することができます。

1) 「**新規**」をクリックして、モジュールまたはライブラリー用の新しいカレンダー項目を作成します。

   既存のカレンダーを選択するか、または新規カレンダー・インスタンスを作成するかを選択することができます。

   – 「**参照**」をクリックして、既存のカレンダー・モジュールを選択します。または、「**新規**」をクリックして、新規カレンダー・モジュールを作成します。
   – 「**参照**」をクリックして、カレンダー用のフォルダーを選択します。(オプション)。
   – 新規カレンダーの名前を入力します。
   – 事前定義テンプレートを使用してカレンダーを生成する場合は、「**次へ**」をクリックします。または、非テンプレート・カレンダーを作成する場合は、「**終了**」をクリックします。

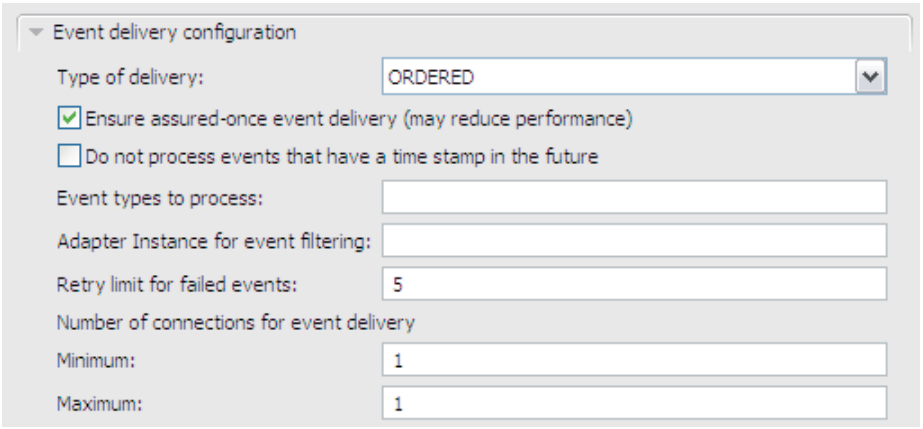
2) 「**参照**」をクリックして、モジュールまたはライブラリー用の既存のカレンダーを選択します。「**ビジネス・カレンダーの選択**」画面で、IBM Integration Designer ワークスペースに現在存在しているすべてのカレンダー・ファイル (`*cal`) を検索することができます。

   – 「**名前**」フィールドにカレンダー名を入力するか、または「**一致するビジネス・カレンダー**」画面でカレンダーをクリックします。「**OK**」をクリックして、外部サービス・ウィザードを開きます。
   – 「**IBM Integration Designer ワークスペース**」で、「**カレンダー**」モジュールを選択して、「**IBM Integration Designer ロジック**」-> 「**カレンダー**」と参照して、カレンダー・スケジュールを表示または変更します。時間間隔および例外を変更したり、これらのエレメントに新しい項目を追加したりすることができます。詳しくは、http://bidoc.torolab.ibm.com:7500/help/index.jsp?topic=/com.ibm.wbpm.main.z.doc/topics/cadm_buscal.htmlの情報を参照してください。

注: ビジネス・カレンダー・モジュールは、 Inbound アプリケーションとともに同じ IBM Business Process Manager または WebSphere Enterprise Service Bus インスタンスにデプロイする必要があります。これら 2 つの関連付けを同じサーバー・インスタンスにマップしないと、ビジネス・カレンダーを使用する Inbound アプリケーションは、デフォルトで、カレンダーが構成されていない場合のようにポーリングを行います。

h. 「**HA Active-Active イベント処理のタイムアウト期間 (秒)**」フィールドに、HA Active-Active イベント処理のタイムアウト期間を秒数で指定します。Inbound Active-Active モードを有効にするには、 322 ページの『高可用性 サポートを使用可能にする (enableHASupport)』 を false に設定します。詳しくは、 390 ページの『HA Active-Active イベント 処理のタイムアウト期間 (秒)』を参照してください。

- **イベント送達構成**



a. 「**送達のタイプ**」フィールドで、送達方法を選択します。この方式については、 373 ページの『送達タイプ (DeliveryType)』で説明します。

b. 確実にイベントが 1 回のみ、1 つのエクスポートにのみ送達されるようにする場合は、「**イベントを 1 回のみ送達する**」チェック・ボックスを選択します。このオプションはパフォーマンスを低下させる可能性がありますが、イベント送達が重複したり欠落したりすることはありません。詳しくは、 374 ページの『イベントを一度のみ送達する (AssuredOnceDelivery)』を参照してください。

c. デフォルトでは、アダプターはポーリング時に検出したすべてのイベントを処理します。現在時刻より後のタイム・スタンプを持つイベントをアダプターで処理しないようにする場合は、「**将来のタイム・スタンプを持つイベントを処理しない**」チェック・ボックスを選択します。詳しくは、 373 ページの『将来のタイム・スタンプを持つイベントを処理しない (FilterFutureEvents)』を参照してください。

d. 「**処理するイベント・タイプ**」フィールドに、必要な値を入力します。

表 18. イベント・タイプの値

値	説明
,	異なるフィルターを分離する (旧バージョンと互換)

表 18. イベント・タイプの値 (続き)

値	説明
:	ビジネス・オブジェクト名と、フィルター用の操作名およびそれ以上の追加の拡張とを分離する
|	サポートされる異なる値を分離する

例えば、Customer ビジネス・オブジェクトと Order ビジネス・オブジェクトに関連するイベントのみを受け取るには、`SchemaNameCustomerBG,SchemaNameOrderBG` と指定します。 Customer ビジネス・オブジェクトの Create 操作のみに関連するイベント、および Order ビジネス・オブジェクトの Create 操作または Delete 操作に関連するイベントを受け取るには、 `SchemaNameCustomerBG:Create,SchemaNameOrderBG:Create|Delete` と指定します。この例の SchemaName は実際のスキーマ名を使用して置き換えてください。

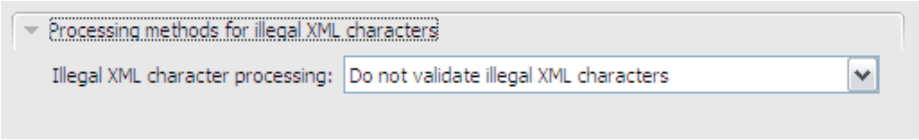
詳しくは、 376 ページの『処理するイベント・タイプ (EventTypeFilter)』を参照してください。

e. 「**イベント・フィルター用のアダプター・インスタンス**」フィールドに、イベントが送達されるコネクター ID を入力します。詳しくは、 364 ページの『イベント・フィルター用のアダプター・インスタンス (AdapterInstanceEventFilter)』を参照してください。

f. イベント送達の失敗後にその送達を試行する回数を「**失敗したイベントの再試行制限**」フィールドに指定します。詳しくは、 378 ページの『失敗したイベントの再試行制限 (FailedEventRetryLimit)』を参照してください。

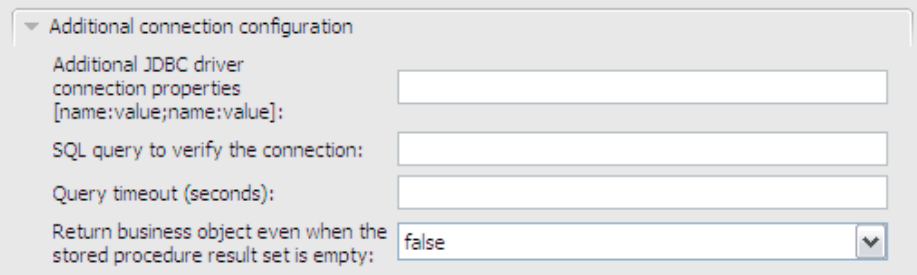
g. 「**イベントを送達するための接続数**」領域で、イベント送達に使用する接続の最小数および最大数を指定します。詳しくは、 380 ページの『最小接続数 (Minimum connections) (MinimumConnections)』および 379 ページの『最大接続数 (Maximum connections) (MaximumConnections)』を参照してください。

- **正しくない XML 文字の処理方法**



  - デフォルトのアダプター動作を使用するには、「**正しくない XML 文字を検証しない**」を選択します。
  - 例外メッセージを受け取った後で続行し、実行時に正しくない XML 文字をトレース・ファイルに記録するには、「**BO の内容に正しくない XML 文字が含まれる場合は例外をスローする**」を選択します。
  - 正しくない XML 文字を廃棄し、実行時にそれらの文字をトレース・ファイルに記録するには、「**すべての正しくない XML 文字および関連ログを廃棄する**」を選択します。

- **追加の接続構成**



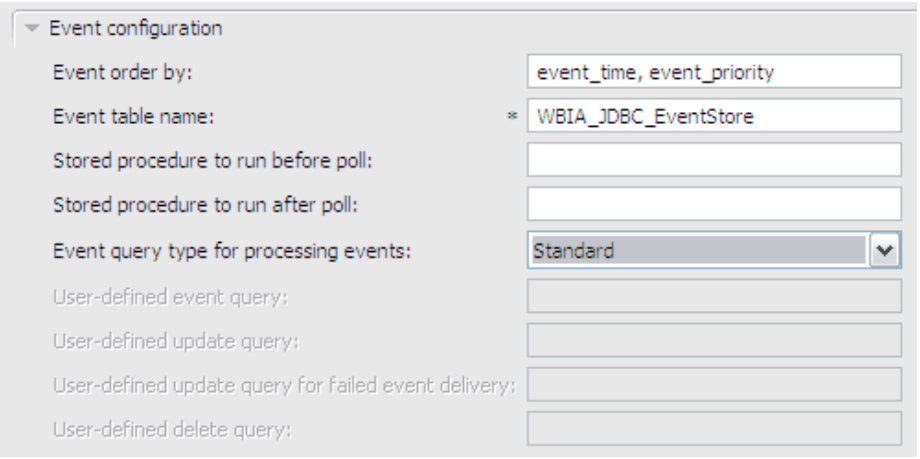
a. 「**追加の JDBC ドライバー接続プロパティー (Additional JDBC driver connection properties)**」を設定します。このプロパティーについて詳しくは、366 ページの『追加の JDBC ドライバー接続プロパティー [name:value;name:value] (JDBCDriverConnectionProperties)』を参照してください。

b. 「**接続を検証するための SQL 照会**」を設定します。このプロパティーについて詳しくは、381 ページの『接続を検証するための SQL 照会 (PingQuery)』を参照してください。

c. 「**照会タイムアウト (秒)**」フィールドに、アダプターがデータベース照会の応答を待機する時間 (秒) を入力します。このプロパティーについて詳しくは、382 ページの『照会タイムアウト (秒) (QueryTimeOut)』を参照してください。

d. 「**ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す**」を設定します。このプロパティーについて詳しくは、384 ページの『ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)』を参照してください。

- **イベント構成**



a. 「**イベント順序**」フィールドで、イベントが取得および処理される順序を指定します。これは、イベント・テーブルの列名と、各列のソート順を制御するキーワードからなるコンマ区切りのリストです。昇順の場合は `asc` を使用し、降順の場合は `desc` を使用します。詳しくは、375 ページの『イベント順序 (EventOrderBy)』を参照してください。

b. 「**イベント・テーブル名**」フィールドで、イベント・ストアを格納するテーブルのデフォルト名を受け入れるか、あるいは別のテーブル名を入力します。詳しくは、 376 ページの『イベント・テーブル名 (EventTableName)』を参照してください。

c. 「**ポーリング前に実行するストアード・プロシージャー**」フィールドで、実際のポーリング照会が呼び出される前に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。詳しくは、 386 ページの『ポーリング前に実行するストアード・プロシージャー (SPBeforePoll)』を参照してください。

d. 「**ポーリング後に実行するストアード・プロシージャー**」フィールドに、各ポーリング周期の後に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。詳しくは、 385 ページの『ポーリング後に実行するストアード・プロシージャー (SPAfterPoll)』を参照してください。

e. 「**イベント処理用のイベント照会タイプ**」フィールドで、使用するイベント処理のタイプを選択します。

   – アダプターによって提供される標準イベント処理を使用するには、「**標準 (Standard)**」を選択します。

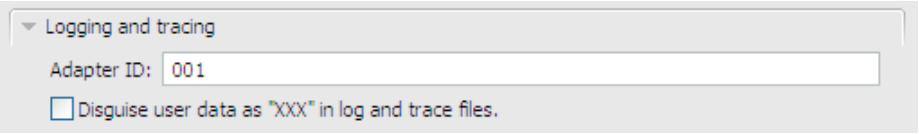
   – ユーザー独自の照会を提供してイベント処理をカスタマイズする場合は、「**ユーザー定義 (Dynamic) (User-Defined (Dynamic))**」を選択します。このオプションを選択する場合は、次の表で説明する追加フィールドに値を入力してください。

     注: カスタムのイベント処理を使用する場合は、 373 ページの『将来のタイム・スタンプを持つイベントを処理しない (FilterFutureEvents)』、 376 ページの『処理するイベント・タイプ (EventTypeFilter)』、および 364 ページの『イベント・フィルター用のアダプター・インスタンス (AdapterInstanceEventFilter)』 プロパティーはサポートされません。

フィールド	指定内容	詳細情報
ユーザー定義削除照会	各イベントの処理後に実行され、イベントの送達後に削除可能なレコードを削除する照会、ストアード・プロシージャー、またはストアード関数の名前	367 ページの『ユーザー定義削除照会 (CustomDeleteQuery)』
ユーザー定義イベント照会	イベントのポーリングを実行する照会、ストアード・プロシージャー、またはストアード関数の名前	368 ページの『ユーザー定義イベント照会 (CustomEventQuery)』 ユーザー定義イベント照会のための JDBC イベント・テーブルの使用
失敗したイベント送達のためのユーザー定義の更新照会	イベントが正常に送達されなかった場合に実行される、照会、ストアード・プロシージャー、またはストアード関数の名前	370 ページの『失敗したイベント送達のためのユーザー定義の更新照会 (CustomUpdateQueryForFailedEvent)』

フィールド	指定内容	詳細情報
ユーザー定義の更新照会	各イベントの処理後に実行され、後続のイベント・サイクルで処理対象としてイベントが選択されることを防ぐ照会、ストアード・プロシージャー、またはストアード関数の名前	369 ページの『ユーザー定義の更新照会 (CustomUpdateQuery)』

- **ロギングおよびトレース**



  a. アダプターのインスタンスが複数ある場合は、**アダプター ID** をこのインスタンスに固有の値に設定します。このプロパティーについて詳しくは、320 ページの『アダプ ター ID (AdapterID)』を参照してください。

  b. ログあるいはトレースに特定の情報を表示しないよう情報をマスクする必要がある場合は、「**ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する**」フィールドを選択します。

8. 「**次へ**」をクリックします。 ロケーション・プロパティーの指定ウィンドウが表示されます。
9. 「ロケーション・プロパティーの指定」ウィンドウで、作成するモジュールの名前を指定します。新規モジュールを指定することも、既存のモジュールを指定することもできます。
   - モジュール名が「**モジュール**」リストに表示されている場合は、その名前を選択します。

     重要: モジュールに、現在構成しているものと同じ名前のインターフェースまたはビジネス・オブジェクトが含まれている場合、そのモジュールにある元のインターフェースまたはビジネス・オブジェクトは新しいバージョンによって置き換えられます。
   - それ以外の場合は、新規モジュールを作成します。
     a. 「**新規**」をクリックします。
     b. 「ビジネス・インテグレーション・プロジェクト・タイプの選択」ウィンドウで、「**モジュール**」を選択して「**次へ**」をクリックします。
     c. 「モジュールの作成」ウィンドウで、モジュールの名前を入力します。例えば、JDBCInboundModule です。
     d. サービス記述ファイル (`.export` ファイルおよび `.wsdl` ファイル) をモジュールのデフォルト・フォルダーに置きたい場合は、「**デフォルト・ロケーションを使用する (Use default location)**」チェック・ボックスを選択したままにします。モジュール内の別のフォルダーを指定する場合は、このオプションをクリアし、「**参照**」をクリックして、「**場所**」フィールドに別のフォルダーを指定します。

e. ウィザードを閉じたときに IBM Integration Designer のアセンブリー・ダイアグラムでこのモジュールが自動的に開くようにする場合は、「**モジュール・アセンブリー・ダイアグラムを開く**」チェック・ボックスを選択します。それ以外の場合は、このオプションをクリアします。

f. 「**終了**」をクリックすると、新規モジュールが作成されます。

10. ビジネス・オブジェクトに使用する名前空間を指定します。
    - モジュールのビジネス・オブジェクトにデフォルトの名前空間を使用させる場合は、「**デフォルトの名前空間を使用する**」チェック・ボックスを選択します。
    - 別の名前空間を指定するには、「**デフォルトの名前空間を使用する**」チェック・ボックスをクリアして、「**名前空間**」フィールドに別の値を入力します。
11. オプション: サービス記述を保存する新規モジュール内のフォルダーを指定します。「**フォルダー**」にフォルダー名を入力するか、既存フォルダーを参照します。 フォルダー名を指定しない場合、成果物 (エクスポート・ファイル、XSD ファイル、および WSDL ファイル) は、モジュールのルート・フォルダー (すなわちモジュール名のフォルダー) に保管されます。
12. 「**名前**」フィールドで、デフォルトのインポート名を受け入れるか、または別の名前を入力します。
13. オプション: 他のモジュールがビジネス・オブジェクトを使用できるようにそのビジネス・オブジェクトをライブラリーに保存する場合は、「**ライブラリーにビジネス・オブジェクトを保存する**」を選択して、「**ライブラリー**」にライブラリーの場所を指定します。
14. オプション: 「**説明**」フィールドに、モジュールを説明するコメントを入力します。
15. プロパティーの設定が完了したら、「**終了**」をクリックします。

タスクの結果

ウィザードは終了します。モジュールがプロジェクトに作成され、成果物が生成されます。

次のタスク

インスタンスによっては、構成を完了するためにアセンブリー・エディターを使用しなければならない場合があります。完了したら、モジュールをテストまたはデプロイできます。

構成の完了

場合によって、ビジネス・オブジェクトの構成を完了するために、手動による構成ステップが必要になります。

このタスクについて

ウィザードによって生成された成果物をカスタマイズする必要がある場合に、このタスクを実行してください。以下の状態でこのタスクを行う可能性があります。

- ある列の CopyAttribute パラメーターの値を、他の列と同じ値に設定する場合。

- ビジネス・オブジェクトから属性を除去する場合。例えば、参照する必要のないデータベース列に対応した単純な属性を除去することによって、ビジネス・オブジェクト設計を簡素化できます。
- ビジネス・オブジェクトに属性を追加する場合。例えば、 DB2 または Microsoft SQL Server データベースのいずれかのデータベース内のテーブルに対してディスカバリー処理を実行するときに、そのテーブルで ID 列として 1 つの列が定義されている場合、そのテーブルに対して生成されるビジネス・オブジェクトには固有 ID 属性は含まれません。アダプターは実行時に ID 列の固有 ID を必要とするため、これを属性のアプリケーション固有情報に追加する必要があります。この場合、<UID>AUTO</UID> を属性のアプリケーション固有情報に追加します。Oracle データベースの場合、Oracle では ID 列がサポートされていないため、シーケンス名に UID を指定して、自動生成のフィールドとしてフィールドを定義するようにします。

  注: Informix データベースからテーブル・ビジネス・オブジェクトを生成した場合は、同様の変更を加える必要はありません。Informix データベースのテーブルに対してディスカバリー・プロセスを実行し、そのテーブルで列がシリアルとして定義されている場合 (ID 列は、Informix では*シリアル列* と呼ばれる)、結果のビジネス・オブジェクトにシリアル列の固有 ID 属性が含まれます。したがって、ビジネス・オブジェクトのアプリケーション固有情報を編集する必要はありません。 Informix データベース・テーブルのシリアル列に生成される固有 ID パラメーター値は、 `serial` または `serial8` のいずれかです。
- 複数の親を持つテーブル・ビジネス・オブジェクトについて追加の親を構成する場合。ウィザードは、1 つのテーブル・ビジネス・オブジェクトにつき、親を 1 つのみ構成します。

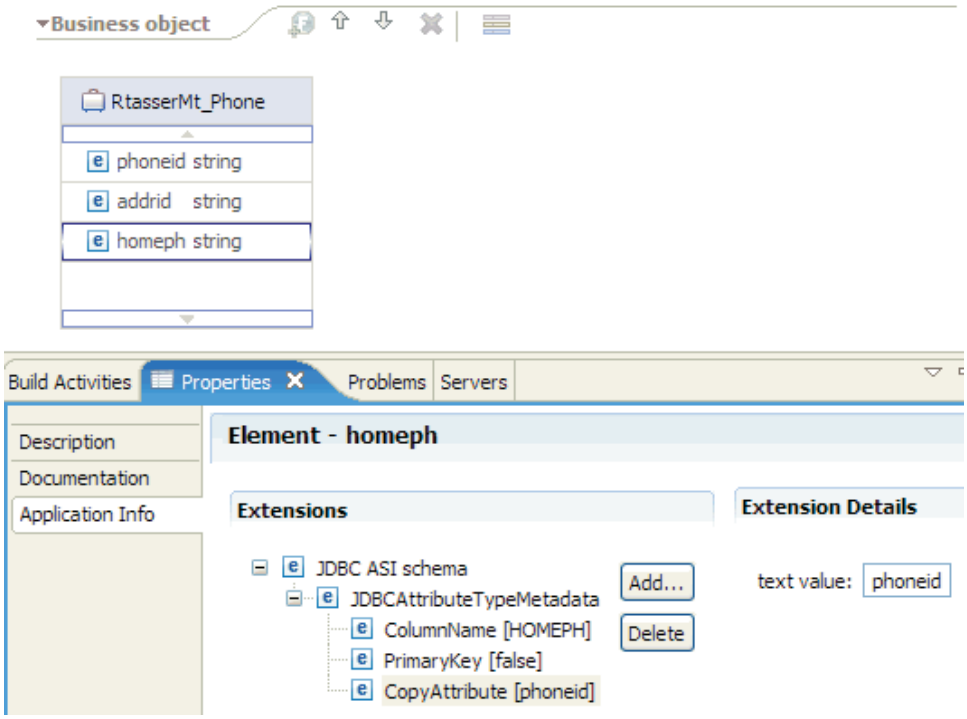
このトピックでは、CopyAttribute パラメーターをテーブル・ビジネス・オブジェクトに設定する、詳細な手順を説明します。ビジネス・オブジェクト構造に対する他の変更 (上に述べた変更など) は、同じ手法を使用して、実行できます。

CopyAttribute パラメーターは、他の列の値およびアプリケーション固有情報を取り込む対象となる列についての属性のプロパティーに含まれます。例えば、テーブルの新しい行の `contact` 列に `e-mail` 列と同じ値を組み込むには、`contact` 属性の CopyAttribute パラメーターを `e-mail` に設定します。 IBM Integration Designer でアセンブリー・エディターを使用して、値を設定します。

手順

1. IBM Integration Designer の「ビジネス・インテグレーション」パースペクティブで、モジュール名、「**データ・タイプ**」を展開し、テーブル・ビジネス・オブジェクトを見つけます。 ビジネス・オブジェクト名は、データベース・スキーマ名にデータベース表名を加えたものです。オプションの名前空間が、名前の最初に含まれる場合もあります。
2. ビジネス・オブジェクト名を右クリックして、「**開く**」を選択します。 アセンブリー・エディターに、ビジネス・オブジェクトが表示され、それぞれの列に関するフィールドが表示されます。
3. アセンブリー・エディターで、他の列と一致させるために設定したい列を選択します。

4. 「プロパティー」ビューで、「アプリケーション情報」を選択します。 「プロパティー」ビューが表示されていない場合は、列の名前を右クリックして、「**プロパティーに表示 (Show in Properties)**」をクリックします。
5. 「**JDBC ASI スキーマ (JDBC ASI schema)**」を展開し、「**JDBCAttributeTypeMetadata**」を展開します。
6. 「**JDBCAttributeTypeMetadata**」を右クリックして、「**新規**」 > 「**jdbcasi:CopyAttribute**」を選択します。
7. 「**CopyAttribute**」プロパティーを選択します。
8. 「拡張子の詳細 (Extension Details)」領域で、コピーする情報を含む列の名前に、テキスト値を設定します。列は、現行ビジネス・オブジェクトに含めることも、親のビジネス・オブジェクトに含めることもできます。現行ビジネス・オブジェクトの列からコピーするには、値を列の名前に設定します (`phoneid` など)。親ビジネス・オブジェクトの列からコピーするには、列の名前のプレフィックスに 2 つのピリオド (..) を使用します (`..phone` など)。以下の図は、CopyAttribute プロパティーが現行テーブルの列に設定されているアセンブリー・エディターを表しています。



タスクの結果

CopyAttribute プロパティーを使用して、別の列の情報に基づいてデータベース列のビジネス・オブジェクト属性およびプロパティーを設定するように、ビジネス・オブジェクトは構成済みです。

次のタスク

これで、モジュールをテストおよびデプロイできます。

成果物の変更

しばしば、ビジネスの要件により、バックエンドのエンタープライズ情報システム (EIS) のデータ構造を変更する必要が生じることがあります。これらの変更を行う場合、外部サービス・ウィザードを使用して以前に生成された成果物の再生成および再構成 (インポートおよびエクスポート) が必要になります。

以下に、エンタープライズ・サービスのディスカバリー・フローからの出力を後続のフローで再利用できる、いくつかのビジネス・シナリオについて概説します。

- 新規のオブジェクトをオブジェクト・セットに追加する場合。
- 選択されたオブジェクトの構成を変更する場合 (操作や操作名の変更、セキュリティー、トランザクション、信頼性などのサービス・レベル設定の変更)。
- ディスカバーされたオブジェクトをオブジェクト・セットから除去する場合。
- サービス内の既存のオブジェクトを再ディスカバーし、バックエンド・システム内のオブジェクトが更新されたときにそのサービスを同期化する場合。

既存の成果物を変更するには、以下のいずれかの方法でウィザードを起動します。外部サービス・ウィザードは、以前に構成された設定で初期化されます。

- アセンブリー・エディターで、変更するコンポーネントを選択して右クリックし、「**編集、バインディングの**」を選択します。
- 「ビジネス・インテグレーション」ビューで、変更するコンポーネントを選択して右クリックし、「**編集、バインディングの**」を選択します。
- アセンブリー・エディターでコンポーネントを選択し、「プロパティー」ビューを選択します。「バインディング」タブで、「**編集**」リンクをクリックします。

注: 「編集、バインディングの」オプションは、IBM Integration Designer 7.0 を使用して生成された成果物でのみ使用可能です。以前のバージョンの IBM Integration Designer からプロジェクト交換をインポートする場合、編集、バインディングの オプションは使用できません。構成を手動で変更した場合、ウィザードを再実行するとそれらの変更が上書きされます。

サービス・インポートの変更

Integration Designer で 編集、バインディングの オプションを使用し、オブジェクトを再ディスカバーおよび再構成して、インポート・コンポーネントを変更します。

このタスクについて

外部サービス・ウィザードを起動して、サービスのインポート・インターフェースの情報を変更できます。ウィザードにより、選択したインポート・インターフェースの既存の情報が自動的に取り込まれます。オブジェクトおよびサービスを変更した後、変更されたデータでインポート・コンポーネントを再生成できます。

手順

1. 以下のいずれかの方法を使用して、選択したサービス・インポート・インターフェースに対して外部サービス・ウィザードを起動します。
   - アセンブリー・エディターで、変更するコンポーネントを選択して右クリックし、「**編集、バインディングの**」を選択します。

- 「ビジネス・インテグレーション」ビューで、変更するコンポーネントを選択して右クリックし、**「編集、バインディングの」**を選択します。
- アセンブリー・エディターでインターフェースを選択し、「プロパティー」ビューを選択します。「バインディング」タブで、**「編集」**リンクをクリックします。

外部サービス・ウィザードの「エンタープライズ・システムでのオブジェクトの検索」ウィンドウが表示されます。外部サービス・ウィザードにより、選択したインポート・インターフェースの既存の構成の詳細が自動的に取り込まれます。

2. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで、必要な変更を行います。オブジェクトのディスカバーについて詳しくは、115 ページの『Outbound 処理のデータベース・オブジェクトのディスカバー』を参照してください。

   注: 外部サービス・ウィザードの接続プロパティーを変更するには、**「戻る (Back)」**をクリックして、「ディスカバリー・プロパティーの指定」ウィンドウでプロパティーを変更します。詳しくは、111 ページの『外部サービス・ウィザードの接続プロパティーの設定』を参照してください。

   a. 以下のオブジェクトを選択して構成することができます。
      - モジュールで使用するテーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを構成するには、120 ページの『Outbound 処理のテーブル、ビュー、およびシノニムまたはニックネームの選択および構成』を参照してください。

        注: 前回のエンタープライズ・サービス・ディスカバリー時に選択したテーブルがデータベース内で削除されていると、アダプターは、**「オブジェクトが見つかりません」**例外を返します。
      - データベース内のストアード・プロシージャーおよびストアード関数に対応するビジネス・オブジェクトを構成するには、131 ページの『ストアード・プロシージャーおよびストアード関数の選択および構成』を参照してください。

        注: データベース内のストアード・プロシージャー定義が変更された場合は、ストアード・プロシージャーを再構成し、検証が成功することを確認する必要があります。
      - バッチ SQL ビジネス・オブジェクトを構成するには、135 ページの『バッチ SQL ビジネス・オブジェクトの選択および構成』を参照してください。
      - 照会ビジネス・オブジェクトを構成するには、141 ページの『照会ビジネス・オブジェクトの選択および構成』を参照してください。

        注: 照会定義が変更された場合は、検証が成功することを確認する必要があります。

   b. **「選択されたオブジェクト」**リストのオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。

   c. **「選択されたオブジェクト」**リストからオブジェクトを除去するには、オブジェクト名を選択して、  (除去) ボタンをクリックします。

3. 「**次へ**」をクリックします。「**取り消し**」をクリックすると、前のステップで行った変更が無効になります。
4. 「複合プロパティーの指定」ウィンドウで、すべてのビジネス・オブジェクトに適用するプロパティーを指定します。詳しくは、 147 ページの『操作のグローバル・プロパティーの設定およびラッパー・ビジネス・オブジェクトの作成』を参照してください。
5. 「**次へ**」をクリックします。
6. 「サービス生成」ウィンドウで、必要に応じてサービス操作を変更します。
7. 「**終了**」をクリックします。 成果物が再生成されます。
8. ほかに手動で行う必須構成があれば完了します。詳しくは、 162 ページの『構成の完了』を参照してください。

タスクの結果

成果物が再生成されます。

次のタスク

モジュールをテストしてデプロイすることができます。

サービス・エクスポートの変更

Integration Designer で 編集、バインディングの オプションを使用し、オブジェクトを再ディスカバーおよび再構成して、エクスポート・コンポーネントを変更します。

このタスクについて

外部サービス・ウィザードを起動して、サービスのエクスポート・インターフェースの情報を変更できます。ウィザードにより、選択したエクスポート・インターフェースの既存の情報が自動的に取り込まれます。オブジェクトおよびサービスを変更した後、変更されたデータでエクスポート・コンポーネントを再生成できます。

手順

1. 以下のいずれかの方法を使用して、選択したサービス・エクスポート・インターフェースに対して外部サービス・ウィザードを起動します。
   - アセンブリー・エディターで、変更するコンポーネントを選択して右クリックし、「**編集、バインディングの**」を選択します。
   - 「ビジネス・インテグレーション」ビューで、変更するコンポーネントを選択して右クリックし、「**編集、バインディングの**」を選択します。
   - アセンブリー・エディターでインターフェースを選択し、「プロパティー」ビューを選択します。「バインディング」タブで、「**編集**」リンクをクリックします。

   外部サービス・ウィザードの「エンタープライズ・システムでのオブジェクトの検索」ウィンドウが表示されます。外部サービス・ウィザードにより、選択したエクスポート・インターフェースの既存の構成の詳細が自動的に取り込まれます。

2. 「エンタープライズ・システムでのオブジェクトの検索」ウィンドウで、必要な変更を行います。オブジェクトのディスカバーについて詳しくは、 164 ページの『Inbound 処理のデータベース・オブジェクトのディスカバー』を参照してください。

   注: 外部サービス・ウィザードの接続プロパティーを変更するには、**「戻る (Back)」**をクリックして、「ディスカバリー・プロパティーの指定」ウィンドウでプロパティーを変更します。詳しくは、 111 ページの『外部サービス・ウィザードの接続プロパティーの設定』を参照してください。

   a. 以下のオブジェクトを選択して構成することができます。
      - モジュールで使用するテーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを構成するには、 169 ページの『Inbound 処理のテーブル、ビュー、およびシノニムまたはニックネームの選択および構成』を参照してください。

        注: 前回のエンタープライズ・サービス・ディスカバリー時に選択したテーブルがデータベースで削除されていると、アダプターは、**「オブジェクトが見つかりません (Object not found)」**例外を生成します。

   b. **「選択されたオブジェクト」**リストのオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。
   c. **「選択されたオブジェクト」**リストからオブジェクトを除去するには、オブジェクト名を選択して、 (除去) ボタンをクリックします。
3. **「次へ」**をクリックします。**「取り消し」**をクリックすると、前のステップで行った変更は無効になります。
4. 「複合プロパティーの指定」ウィンドウで、すべてのビジネス・オブジェクトに適用するプロパティーを指定します。詳しくは、 179 ページの『操作のグローバル・プロパティーの設定およびラッパー・ビジネス・オブジェクトの作成』を参照してください。
5. **「次へ」**をクリックします。
6. 「サービス生成」ウィンドウで、必要に応じてサービス操作を変更します。
7. **「終了」**をクリックします。 成果物が再生成されます。
8. ほかに手動で行う必須構成があれば完了します。詳しくは、 162 ページの『構成の完了』を参照してください。

タスクの結果

成果物が再生成されます。

次のタスク

モジュールをテストしてデプロイすることができます。

アセンブリー・エディターによる対話仕様プロパティーの変更

サービスの生成後にアダプター・モジュールの対話仕様プロパティーを変更するには、IBM Integration Designer のアセンブリー・エディターを使用します。

始める前に

アダプターに対してサービスを生成するには、あらかじめ外部サービス・ウィザードを使用しておく必要があります。

このタスクについて

アダプターのサービスを生成後に、対話仕様プロパティーの変更が必要になる場合があります。対話仕様プロパティーはオプションですが、特定のビジネス・オブジェクトの特定の操作に対して、メソッド・レベルで設定されます。指定した値は、外部サービス・ウィザードによって生成されるすべての親ビジネス・オブジェクトのデフォルトとして表示されます。 これらのプロパティーは、EAR ファイルをエクスポートする前に変更できます。アプリケーションをデプロイした後にこれらのプロパティーを変更することはできません。

対話仕様プロパティーを変更するには、以下の手順を使用します。

手順

1. IBM Integration Designer の Business Integration パースペクティブで、モジュール名を展開します。
2. 「**アセンブリー・ダイアグラム**」を展開して、インターフェースをダブルクリックします。
3. アセンブリー・エディターでインターフェースをクリックします。 (追加のクリックをしない限り、モジュールのプロパティーが表示されています。)
4. 「**プロパティー**」タブをクリックします。(ダイアグラム内でインターフェースを右クリックし、「**プロパティーを表示**」をクリックすることもできます。)
5. 「**バインディング**」で、「**メソッド・バインディング**」をクリックします。インターフェースのメソッドが、ビジネス・オブジェクトと操作の組み合わせごとに 1 つずつ表示されます。
6. 変更する対話仕様プロパティーを持つメソッドを選択します。
7. 「**汎用**」タブ内のプロパティーを変更します。変更する対話仕様プロパティーを持つメソッドごとにこの手順を繰り返します。

タスクの結果

アダプター・モジュールに関連付けられている対話仕様プロパティーが変更されました。

次のタスク

モジュールをデプロイします。

モジュールのデプロイ

モジュールをデプロイし、モジュールおよびアダプターを構成するファイルを、実稼働またはテストのための動作環境に配置します。 IBM Integration Designerの統合テスト環境では、インストール時に選択したテスト環境のプロファイルに応じて、IBM Business Process Manager または WebSphere Enterprise Service Bus、あるいは両方のランタイムがサポートされます。

デプロイメント環境

モジュールおよびアダプターのデプロイ先には、テスト環境と実稼働環境があります。

IBM Integration Designer では、モジュールをテスト環境内の 1 つ以上のサーバーにデプロイできます。通常は、これがビジネス・インテグレーション・モジュールの実行およびテストを行うための最も一般的な手法です。ただし、IBM Business Process Manager または WebSphere Enterprise Service Bus 上で管理コンソールまたはコマンド行ツールを使用して、サーバーへのデプロイメント用のモジュールを EAR ファイルとしてエクスポートすることもできます。

テスト用のモジュールのデプロイ

IBM Integration Designer では、組み込みアダプターを含むモジュールをテスト環境にデプロイし、サーバー構成の編集、サーバーの始動と停止、およびモジュール・コードでのエラーのテストなどのタスクを実行できるサーバー・ツールで作業を行うことができます。 テストは通常、コンポーネントのインターフェース操作について実行されますが、このテストを実行すると、コンポーネントが正しく実装され、参照先が正しく接続されているかどうかを判断できます。

外部依存関係の追加

依存関係のある JAR は、ライブラリー・ディレクトリーに追加するか、または EAR にパッケージ化する必要があります。

このタスクについて

JAR は、クラス・パスに設定され、これら依存ライブラリーは、モジュールのデプロイ時に、ランタイムに使用できるようにする必要があります。依存ライブラリーを使用可能にするには、スタンドアロン・デプロイメントまたは組み込みデプロイメント用と、組み込みデプロイメント専用の 2 つの方法があります。

サーバーでの外部ソフトウェア依存関係の追加:

アダプターがデータベースと通信するには、IBM Business Process Manager または WebSphere Enterprise Service Bus サーバーにインストールされた特定のファイルが必要です。

始める前に

データベースが IBM Business Process Manager または WebSphere Enterprise Service Bus と同じコンピューター・システム上にインストールされている場合は、この作業を実行する必要はありません。ファイルは既にアダプターで使用できる状態になっています。

このタスクについて

アダプターが、データベース・サーバーと通信するには、そのデータベース・サーバーの JDBC ドライバー・ファイルまたはネイティブ・システム・ライブラリーが必要です。

手順

1. データベース管理者またはデータベース・ソフトウェアの Web サイトから、ご使用のデータベース・ソフトウェアおよびオペレーティング・システムの JDBC ドライバー固有ファイルまたはネイティブ・ライブラリーを入手します。必要なファイルの種類は、データベース・サーバーによって異なります。

   次の表に、一般的なデータベース・ソフトウェアで必要とされるファイルをリストします。

表 19. 共通データベース・ソフトウェアの JDBC ドライバー・ファイル

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
IBM DB2 Universal Database (Linux、UNIX、および Windows 版)	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cu.jar	なし
IBM DB2 for z/OS	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cisuz.jar	なし
IBM DB2 for IBM i	IBM Toolbox for Java リモート・ドライバー (Type 4)	jt400.jar db2jcc_license_cisuz.jar	なし
	IBM Toolkit for Java ネイティブ・ドライバー* (Type 2)	db2_classes.jar	なし
IBM DB2 Universal Database (Linux、UNIX、および Windows 版)	IBM DB2 Universal (Type 2)	db2java.zip	なし
Oracle	Thin ドライバー	ojdbc6.jar ランタイム環境で XML データ・タイプを扱うには、さらに以下の追加ライブラリーが必要です。 xdb.jar xmlparserv2.jar	なし
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	なし

表 19. 共通データベース・ソフトウェアの JDBC ドライバー・ファイル (続き)

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
Informix	IBM Informix JDBC ドライバー	ifxjdbc.jar ifxjdbcx.jar および いくつかのサポート .jar ファイル 詳しくは、「IBM Informix JDBC ドライバ プログラマーズ ガイド」を参照してください。	なし
* IBM Toolkit for Java ネイティブ・ドライバーを使用して、アダプター実行時のデータベースに接続することはできますが、これをウィザード実行時の接続に使用することはできません。ディスカバリー・プロセス中は、IBM Toolbox for Java リモート・ドライバー、または、IBM DB2 Universal ドライバーを使用する必要があります。ただし、実行時にネイティブ・ドライバーを使用するようにモジュールを構成することはできます。これは、「サービス生成およびデプロイメント・プロパティーの指定」ウィンドウで行います。			

注: JDBC ドライバーが必要とする JRE バージョンは、ランタイム環境の JRE バージョンに等しいかそれ以下でなければなりません。

2. ファイルをサーバーにコピーします。
   - Integration Designer のテスト環境の場合は、ファイルを ${WAS_INSTALL_ROOT}/runtimes/bi_v7/lib/ext ディレクトリーにコピーします。
   - 実稼働環境の場合は、ファイルを IBM Business Process Manager または WebSphere Enterprise Service Bus の ${WAS_INSTALL_ROOT}/lib/ext ディレクトリーにコピーします。

アダプターがバンドルされる場合の外部ソフトウェア依存関係の追加:

アダプター・アプリケーションを実行するには、まず、依存関係のある JAR ファイルを EAR アプリケーションにコピーする必要があります。この方法を使用するのは、組み込みデプロイメントの場合に限られます。

このタスクについて

必要なファイルを入手して、それらを EAR アプリケーションにコピーするには、以下の手順に従います。

手順

1. 該当するモジュールから、ワークスペースに進み、JAR ファイルをディレクトリーにコピーします。例えば、モジュール名が「ModuleName」の場合、ワークスペースに進んで、JAR ファイルを `ModuleNameApp/EarContent` ディレクトリーにコピーします。

2. アダプター RAR のマニフェスト・ファイル、manifest.mf を、アダプターが必要とする JAR ファイルのリストで変更します。JAR ファイルを「Class-Path: dependantjar1.jar, dependantjar2.jar」の形式で追加します。
3. ネイティブ・ライブラリーをランタイム bin ディレクトリーにコピーし、アプリケーションをデプロイします。

タスクの結果

これでサード・パーティー・ライブラリーがランタイム環境の一部となります。

Inbound 処理をテストするためのターゲット・コンポーネントの生成および接続

Inbound 処理用のアダプターが組み込まれているモジュールをテスト環境にデプロイする前に、まずターゲット・コンポーネントを生成して接続する必要があります。このターゲット・コンポーネントは、アダプターがイベントを送信する*宛先* として機能します。

始める前に

外部サービス・ウィザードを使用してエクスポート・モジュールを生成してあるはずです。

このタスクについて

Inbound 処理のためにターゲット・コンポーネントを生成して接続する必要があるのは、テスト環境のみです。実稼働環境でアダプターをデプロイする際には必要ありません。

ターゲット・コンポーネントは、イベントを受信します。 IBM Integration Designer のアセンブリー・エディターを使用して、エクスポート・コンポーネントを (2 つのコンポーネントを接続している) ターゲット・コンポーネントに*接続* します。アダプターはこの接続を使用して、(エクスポート・コンポーネントからターゲット・コンポーネントへ) イベント・データを受け渡します。

手順

1. ターゲット・コンポーネントを作成します。
   a. IBM Integration Designer の Business Integration パースペクティブで、「**アセンブリー・ダイアグラム**」を展開して、エクスポート・コンポーネントをダブルクリックします。 デフォルト値を変更しなかった場合、エクスポート・コンポーネントの名前は、ご使用のアダプター + **InboundInterface** になります。

      インターフェースにより、呼び出すことができる操作と渡されるデータ (入力引数、戻り値、例外など) が指定されます。 **InboundInterface** コンポーネントには、Inbound 処理をサポートするためにアダプターが必要とする操作が格納されています。また、このコンポーネントは外部サービス・ウィザードを実行すると作成されます。
   b. 「**コンポーネント**」を展開して「**型なしコンポーネント**」を選択し、そのコンポーネントをアセンブリー・ダイアグラムまでドラッグして、新規コンポーネントを作成します。

カーソルが配置アイコンに変わります。

c. アセンブリー・ダイアグラムに表示させるにはコンポーネントをクリックします。

2. コンポーネントを接続します。

a. エクスポート・コンポーネントをクリックして新規コンポーネントにドラッグします。

b. アセンブリー・ダイアグラムを保存します。 「**ファイル**」 > 「**保存**」とクリックします。

3. 新規コンポーネントの実装を生成します。

a. 新規コンポーネントを右クリックして、「**実装の生成**」 > 「**Java**」を選択します。

b. 「**(デフォルト・パッケージ)**」を選択して、「**OK**」をクリックします。 これにより、Inbound モジュールのエンドポイントが作成されます。

別のタブに Java 実装環境が表示されます。

c. **オプション:** print ステートメントを追加して、各エンドポイント・メソッドのエンドポイントで受信したデータ・オブジェクトを出力します。

d. 「**ファイル**」 > 「**保存**」をクリックして、変更内容を保存します。

次のタスク

テストを行うモジュールのデプロイを続行します。

Outbound 操作のテスト準備

IBM Integration Designer テスト・クライアントでご使用のモジュールの Outbound 処理をテストする前に、ビジネス・オブジェクトのいくつかを変更しなければならない場合があります。

このタスクについて

このステップは、IBM Integration Designer テスト・クライアントで実行されます。プロジェクトの名前を右クリックして、「**テスト**」 > 「**モジュールのテスト**」をクリックし、ビジネス・インテグレーション・パースペクティブから IBM Integration Designer を開きます。

手順

- **照会ビジネス・オブジェクト**

  ご使用の照会ビジネス・オブジェクトが WHERE 節なしで作成されていた場合 (例えば、`Select * from Customer` などの SELECT ステートメントで定義されていた場合)、テスト・クライアントでテストする前に、照会ビジネス・オブジェクトの jdbcwhereclause 属性を設定解除します。

- **テーブル、ビュー、およびシノニムまたはニックネーム・ビジネス・オブジェクト**

  RetrieveAll 操作をテストする前に、テストの一環として設定しない値を持つ、あらゆる属性を設定解除する必要があります。

- **照会ビジネス・オブジェクト**

RetrieveAll 操作をテストする前に、テストの一環として設定しない値を持つ、あらゆる属性を設定解除する必要があります。

サーバーへのモジュールの追加

IBM Integration Designerでは、モジュールをテスト環境内の 1 つ以上のサーバーに追加できます。

始める前に

テスト対象のモジュールが Inbound 処理の実行にアダプターを使用する場合は、そのアダプターのイベントの送信先となるターゲット・コンポーネント を生成し、接続してください。

このタスクについて

モジュール、およびモジュールによるアダプターの使用をテストするために、サーバーへモジュールを追加する必要があります。

手順

1. *条件付き:* 「**サーバー**」ビューにサーバーがない場合は、以下の手順を実行し、新規サーバーを追加して定義します。
   a. 「**サーバー**」ビューにカーソルを置き、右クリックして「**新規**」 > 「**サーバー**」と選択します。
   b. 「新規サーバーの定義」ウィンドウで、サーバー・タイプを選択します。
   c. サーバーの設定値を構成します。
   d. 「**終了**」をクリックして、サーバーを公開します。
2. モジュールをサーバーに追加します。
   a. 「サーバー」ビューに切り替えます。 IBM Integration Designer で、「**ウィンドウ**」 > 「**ビューの表示**」 > 「**サーバー**」を選択します。
   a. サーバーを始動します。 IBM Integration Designer の画面の右下のペインの「**サーバー**」タブで、「サーバー」を右クリックして、「**開始**」を選択します。
3. サーバーの状況が「*開始済み*」である場合は、サーバーを右クリックし、「**プロジェクトの追加および除去**」を選択します。
4. 「プロジェクトの追加および除去」画面で、対象のプロジェクトを選択して「**追加**」をクリックします。 プロジェクトは、「**使用可能プロジェクト**」のリストから「**構成プロジェクト**」のリストに移動します。
5. 「**終了**」をクリックします。 これにより、モジュールがサーバーにデプロイされます。

   モジュールがサーバーに追加されている間に、右下のペインの「コンソール」タブに、ログが表示されます。

次のタスク

モジュールおよびアダプターの機能をテストします。

テスト・クライアントを使用した Outbound 処理用モジュールのテスト

Outbound 処理用のアセンブル済みモジュールおよびアダプターを、IBM Integration Designer の統合テスト・クライアントを使用してテストします。

始める前に

最初にモジュールをサーバーに追加する必要があります。

このタスクについて

モジュールのテストは、コンポーネントのインターフェース操作を対象に実行されます。そのため、コンポーネントが正しく実装されているかどうか、および参照先が正しく接続されているかどうかを確認できます。

手順

1. テストするモジュールを選択し、右クリックして、「**テスト**」 > 「**テスト・モジュール**」を選択します。
2. テスト・クライアントを使用したモジュールのテストについて詳しくは、IBM Integration Designer インフォメーション・センターの『モジュールおよびコンポーネントのテスト *(Testing modules and components)*』のトピックを参照してください。

次のタスク

ご使用のモジュールおよびアダプターのテスト結果に納得したら、モジュールおよびアダプターを実稼働環境にデプロイできます。

実稼働のためのモジュールのデプロイ

外部サービス・ウィザードを使用して作成したモジュールを、実稼働環境で IBM Business Process Manager または WebSphere Enterprise Service Bus にデプロイする処理は、2 段階構成になっています。最初に、IBM Integration Designer 内にモジュールをエンタープライズ・アーカイブ (EAR) ファイルの形でエクスポートします。次に、IBM Business Process Manager または WebSphere Enterprise Service Bus 管理コンソール を使用して、EAR ファイルをデプロイします。

サーバーでの外部ソフトウェア依存関係の追加

アダプターがデータベースと通信するには、IBM Business Process Manager または WebSphere Enterprise Service Bus サーバーにインストールされた特定のファイルが必要です。

始める前に

データベースが IBM Business Process Manager または WebSphere Enterprise Service Bus と同じコンピューター・システム上にインストールされている場合は、この作業を実行する必要はありません。ファイルは既にアダプターで使用できる状態になっています。

このタスクについて

アダプターが、データベース・サーバーと通信するには、そのデータベース・サーバーの JDBC ドライバー・ファイルまたはネイティブ・システム・ライブラリーが必要です。

手順

1. データベース管理者またはデータベース・ソフトウェアの Web サイトから、ご使用のデータベース・ソフトウェアおよびオペレーティング・システムの JDBC ドライバー固有ファイルまたはネイティブ・ライブラリーを入手します。必要なファイルの種類は、データベース・サーバーによって異なります。

   次の表に、一般的なデータベース・ソフトウェアで必要とされるファイルをリストします。

表 20. 共通データベース・ソフトウェアの JDBC ドライバー・ファイル

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
IBM DB2 Universal Database (Linux、UNIX、および Windows 版)	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cu.jar	なし
IBM DB2 for z/OS	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cisuz.jar	なし
IBM DB2 for IBM i	IBM Toolbox for Java リモート・ドライバー (Type 4)	jt400.jar db2jcc_license_cisuz.jar	なし
	IBM Toolkit for Java ネイティブ・ドライバー* (Type 2)	db2_classes.jar	なし
IBM DB2 Universal Database (Linux、UNIX、および Windows 版)	IBM DB2 Universal (Type 2)	db2java.zip	なし
Oracle	Thin ドライバー	ojdbc6.jar ランタイム環境で XML データ・タイプを扱うには、さらに以下の追加ライブラリーが必要です。 xdb.jar xmlparserv2.jar	なし
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	なし

表 20. 共通データベース・ソフトウェアの JDBC ドライバー・ファイル (続き)

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
Informix	IBM Informix JDBC ドライバー	ifxjdbc.jar ifxjdbcx.jar および いくつかのサポート .jar ファイル 詳しくは、「IBM Informix JDBC ドライバ プログラマーズ ガイド」を参照してください。	なし
* IBM Toolkit for Java ネイティブ・ドライバーを使用して、アダプター実行時のデータベースに接続することはできますが、これをウィザード実行時の接続に使用することはできません。ディスカバリー・プロセス中は、IBM Toolbox for Java リモート・ドライバー、または、IBM DB2 Universal ドライバーを使用する必要があります。ただし、実行時にネイティブ・ドライバーを使用するようにモジュールを構成することはできます。これは、「サービス生成およびデプロイメント・プロパティーの指定」ウィンドウで行います。			

注: JDBC ドライバーが必要とする JRE バージョンは、ランタイム環境の JRE バージョンに等しいかそれ以下でなければなりません。

2. ファイルをサーバーにコピーします。
   - Integration Designer のテスト環境の場合は、ファイルを ${WAS_INSTALL_ROOT}/runtimes/bi_v7/lib/ext ディレクトリーにコピーします。
   - 実稼働環境の場合は、ファイルを IBM Business Process Manager または WebSphere Enterprise Service Bus の ${WAS_INSTALL_ROOT}/lib/ext ディレクトリーにコピーします。

RAR ファイルのインストール (スタンドアロン・アダプターを使用するモジュールの場合のみ)

アダプターをモジュールに組み込まないが、サーバー・インスタンスのデプロイされたすべてのアプリケーションで使用可能にすることを選択する場合は、RAR ファイルのフォーマットでアダプターをアプリケーション・サーバーにインストールする必要があります。 RAR ファイルとは、Java 2 Connector (J2C) アーキテクチャーに合わせてリソース・アダプターを圧縮するときに使用する Java アーカイブ (JAR) ファイルのことです。

始める前に

外部サービス・ウィザードの「サービス生成およびデプロイメント・プロパティーの指定」ウィンドウで、「**コネクター・プロジェクトのデプロイ**」を「**複数アダプターが使用するサーバー上**」に設定する必要があります。

このタスクについて

アダプターを RAR ファイルのフォーマットでインストールすると、そのアダプターは、サーバー・ランタイムで実行されているすべての J2EE アプリケーション・コンポーネントで使用可能になります。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**リソース**」 > 「**リソース・アダプター**」 > 「**リソース・アダプター**」をクリックします。
5. 「リソース・アダプター」ページで、「**RAR のインストール**」をクリックします。

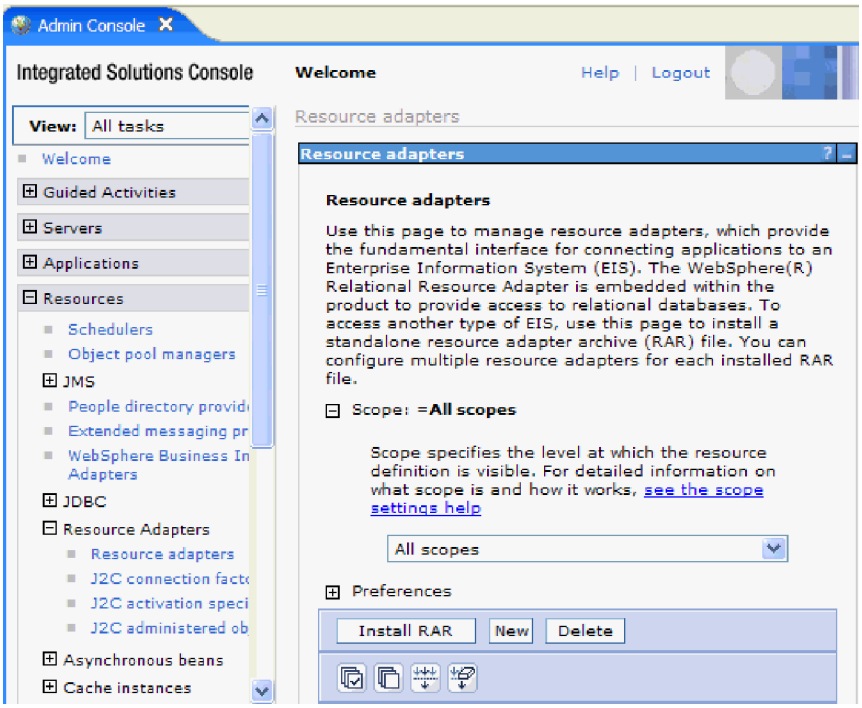
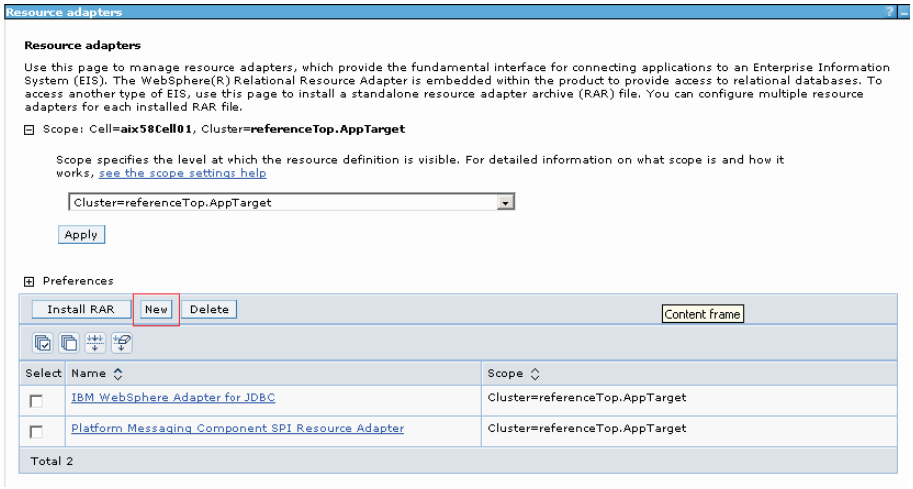


図 28. 「リソース・アダプター」ページの「RAR のインストール」ボタン

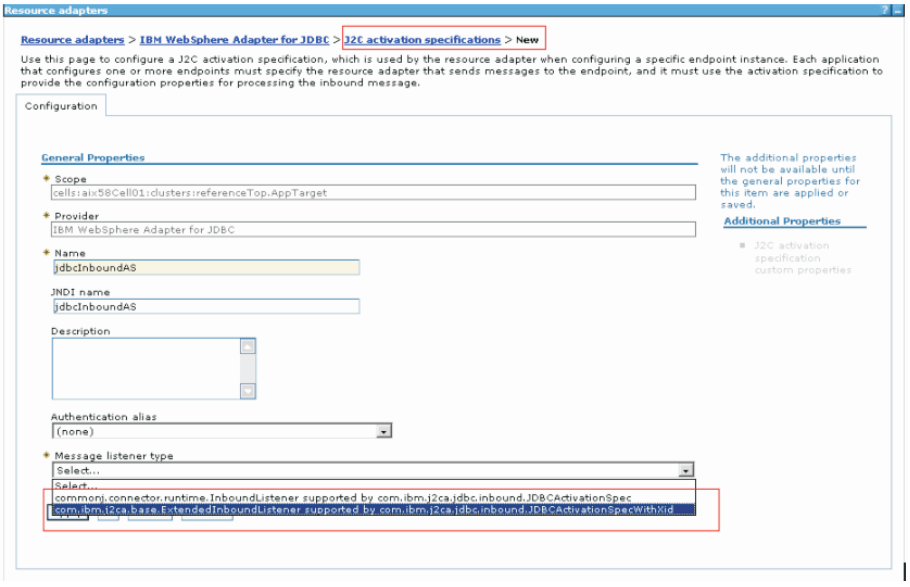
6. 「RAR ファイルのインストール」ページで、「**参照**」をクリックし、ご使用のアダプターの RAR ファイルへ移動します。

   RAR ファイルは、通常、パス *IID_installation_directory*/ResourceAdapters/*adapter_name*/*adapter*.rar にインストールされます。
7. 「**次へ**」をクリックします。
8. オプション: 「リソース・アダプター」ページで、アダプターの名前を変更し、説明を追加します。
9. 「**OK**」をクリックします。

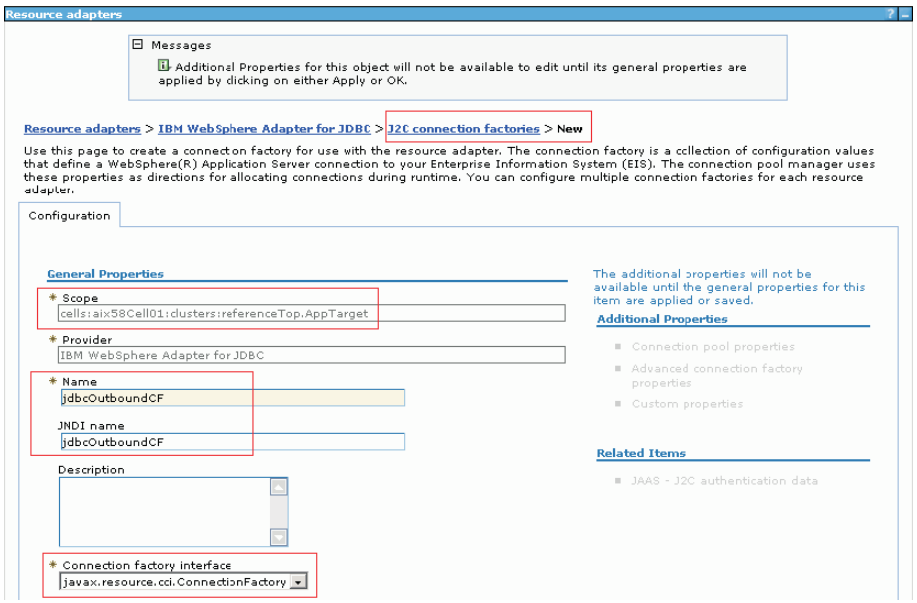
10. ページの上部にある「**メッセージ**」ボックスで「**保存**」をクリックします。
11. オプション: クラスター環境の場合は、それぞれのノードに個別にアダプターをインストールする必要があります。クラスター環境にアダプターをインストールした後、以下の手順を実行します。
    a. クラスター環境に JDBC アダプターを作成します。「**リソース・アダプター**」 > 「***target_cluster***」 > 「**新規**」とクリックして、クラスター用の新規のリソース・アダプターを作成します。例えば、以下の図の場合、ターゲット・クラスターは `Cluster=referenceTop.AppTarget` です。



    b. 「**リソース・アダプター**」 > 「***adapter_name***」 > 「**J2C 活動化仕様**」 > 「**新規**」とクリックして、Inbound サービス用の新規活動化仕様を作成します。



    c. 「**リソース・アダプター**」 > 「***adapter_name***」 > 「**J2C 接続ファクトリー**」 > 「**新規**」とクリックし、Outbound サービス用の新規接続ファクトリーを作成します。



d. 要件に応じて、リソース・アダプターのカスタム・プロパティー (活動化仕様と接続ファクトリーの両方) を構成します。

次のタスク

次の手順は、サーバーにデプロイできる EAR ファイルとしてモジュールをエクスポートすることです。

EAR ファイルとしてのモジュールのエクスポート

IBM Integration Designer を使用して、モジュールを EAR ファイルとしてエクスポートします。EAR ファイルを作成することによって、モジュールのすべての内容を IBM Business Process Manager または WebSphere Enterprise Service Bus に容易にデプロイできる形式で取り込みます。

始める前に

モジュールを EAR ファイルとしてエクスポートするには、事前にサービスと通信するためのモジュールを作成しておく必要があります。このモジュールを、IBM Integration Designer ビジネス・インテグレーション・パースペクティブ内に表示する必要があります。

このタスクについて

モジュールを EAR ファイルとしてエクスポートするには、以下の手順を実行します。

手順

1. モジュールを右クリックして、「**エクスポート**」を選択します。
2. 「選択」ウィンドウで、「**Java EE**」を展開します。
3. 「**EAR ファイル**」を選択して、「**次へ**」をクリックします。

4. オプション: 正しい EAR アプリケーションを選択します。 EAR アプリケーションにはモジュールと同じ名前が付けられますが、名前の末尾に「App」が追加されます。
5. EAR ファイルを格納するローカル・ファイル・システム上で、フォルダーを参照します。
6. ソース・ファイルをエクスポートする場合は、「**ソース・ファイルのエクスポート**」チェック・ボックスを選択します。 このオプションは、EAR ファイルのほかにソース・ファイルをエクスポートする場合に表示されます。 ソース・ファイルには、Java コンポーネント、データ・マップなどに関連付けられているファイルがあります。
7. 既存のファイルを上書きする場合は、「**既存ファイルの上書き**」をクリックします。
8. 「**終了**」をクリックします。

タスクの結果

モジュールの内容が EAR ファイルとしてエクスポートされます。

次のタスク

このモジュールを管理コンソールにインストールします。これにより、モジュールが IBM Business Process Manager または WebSphere Enterprise Service Bus にデプロイされます。

EAR ファイルのインストール

EAR ファイルのインストールは、デプロイメント・プロセスの最終手順です。 EAR ファイルをサーバーにインストールして実行すると、EAR ファイルの一部として組み込まれているアダプターが、インストール済みアプリケーションの一部として稼働します。

始める前に

モジュールを IBM Business Process Manager または WebSphere Enterprise Service Bus にインストールするには、その前にモジュールを EAR ファイルとしてエクスポートしておく必要があります。

このタスクについて

EAR ファイルをインストールするには、次の手順を実行します。アダプター・モジュール・アプリケーションのクラスター化について詳しくは、 http://www.ibm.com/software/webservers/appserv/was/library/ を参照してください。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。

4. 「**アプリケーション**」 > 「**新規アプリケーション**」 > 「**新規エンタープライズ・アプリケーション**」 とクリックします。

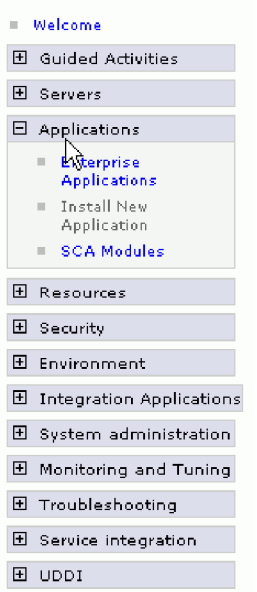


図 29. 「アプリケーション・インストールの準備」ウィンドウ

5. 「**参照**」をクリックして、EAR ファイルを位置指定し、「**次へ**」をクリックします。 EAR ファイル名は、モジュール名の後に「App」が付いたものです。
6. オプション: クラスター環境にデプロイする場合は、以下の手順を実行します。
   a. 「**ステップ 2: サーバーにモジュールをマップ**」ウィンドウで、モジュールを選択し、「**次へ**」をクリックします。
   b. サーバー・クラスターの名前を選択します。
   c. 「**適用**」をクリックします。
7. 「**次へ**」をクリックします。「要約」ページで設定を確認して、「**終了**」をクリックします。
8. オプション: 認証別名を使用している場合は、以下の手順を実行します。
   a. 「**セキュリティー**」を展開して、「**ビジネス・インテグレーション・セキュリティー**」を選択します。
   b. 構成する認証別名を選択します。 認証別名構成に変更を加えるためには、管理者権限またはオペレーター権限が必要です。
   c. オプション: 「**ユーザー名**」を入力します (まだ入力されていない場合)。
   d. 「**パスワード**」を入力します (まだ入力されていない場合)。
   e. 「**確認パスワード (Confirm Password)**」フィールドに再度パスワードを入力します (まだ入力されていない場合)。
   f. 「**OK**」をクリックします。

タスクの結果

この時点で、プロジェクトがデプロイメントされ、「エンタープライズ・アプリケーション」ウィンドウが表示されます。

次のタスク

いずれかのプロパティーを設定または再設定する場合、あるいは、アダプター・プロジェクトのアプリケーションをクラスター化する場合は、トラブルシューティング・ツールを構成する前に、管理コンソールを使用してそれらの変更を行ってください。

アダプター・モジュールの管理

アダプターをスタンドアロンのデプロイメントで稼働している場合は、アダプター・モジュールの開始、停止、モニター、およびトラブルシューティングには、サーバーの管理コンソールを使用します。組み込みアダプターを使用しているアプリケーションでは、アプリケーションの開始時または停止時にアダプター・モジュールが開始または停止します。

組み込みアダプターの構成プロパティーの変更

アダプターをモジュールの一部としてデプロイした後に構成プロパティーを変更するには、ランタイム環境の管理コンソールを使用します。 リソース・アダプター・プロパティー (一般的なアダプター操作に使用)、管理接続ファクトリー・プロパティー (Outbound 処理に使用)、および活動化仕様プロパティー (Inbound 処理に使用) を更新できます。

組み込みアダプターのリソース・アダプター・プロパティーの設定

アダプターをモジュールの一部としてデプロイした後に、このアダプターのリソース・アダプター・プロパティーを設定するには、管理コンソールを使用します。 構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプター・モジュールを IBM Business Process Manager または WebSphere Enterprise Service Bus 上にデプロイする必要があります。

このタスクについて

カスタム・プロパティーとは、すべての IBM WebSphere Adapters が共有するデフォルト構成プロパティーです。

管理コンソールを使用してプロパティーを構成するには、以下の手順を使用します。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**アプリケーション**」 > 「**アプリケーション・タイプ**」 > 「**WebSphere エンタープライズ・アプリケーション (WebSphere enterprise application)**」と選択します。

5. 「エンタープライズ・アプリケーション」リストから、プロパティーを変更するアダプター・モジュールの名前をクリックします。「**構成**」ページが表示されます。

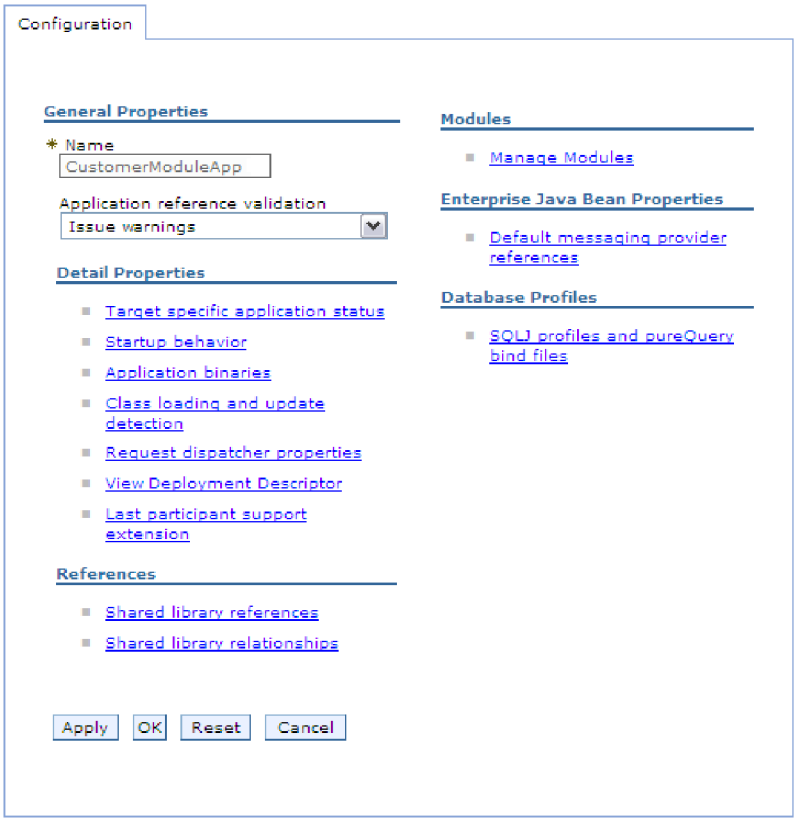


図 30. 「構成」タブでの「モジュールの管理」の選択

6. 「**モジュール**」の下で、「**モジュールの管理**」をクリックします。
7. 「**IBM WebSphere Adapter for JDBC**」をクリックします。
8. 「**追加プロパティー**」リストから、「**リソース・アダプター**」をクリックします。
9. 次のページで、「**追加プロパティー**」リストから、「**カスタム・プロパティー**」をクリックします。
10. 変更するプロパティーごとに、以下の手順を実行します。

    注: これらのプロパティーについて詳しくは、 318 ページの『リソース・アダプター・プロパティー』 を参照してください。

    a. プロパティーの名前をクリックします。選択されたプロパティーの 「**構成**」 ページが表示されます。
    b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
    c. 「**OK**」をクリックします。
11. 「メッセージ」領域で「**保存**」をクリックします。

タスクの結果

アダプター・モジュールに関連付けられているリソース・アダプター・プロパティーが変更されました。

組み込みアダプターの管理 (J2C) 接続ファクトリー・プロパティーの設定

アダプターをモジュールの一部としてデプロイした後に、アダプターの管理接続ファクトリー・プロパティーを設定するには、管理コンソールを使用します。構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプター・モジュールを IBM Business Process Manager または WebSphere Enterprise Service Bus 上にデプロイする必要があります。

このタスクについて

管理接続ファクトリー・プロパティーは、ターゲット データベースのインスタンスを構成する場合に使用します。

注: 管理コンソール内では、このプロパティーを「J2C 接続ファクトリー・プロパティー」と呼びます。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**アプリケーション**」 > 「**アプリケーション・タイプ**」 > 「**WebSphere エンタープライズ・アプリケーション (WebSphere enterprise application)**」と選択します。
5. 「エンタープライズ・アプリケーション」リストで、プロパティーを変更するアダプター・モジュールの名前をクリックします。

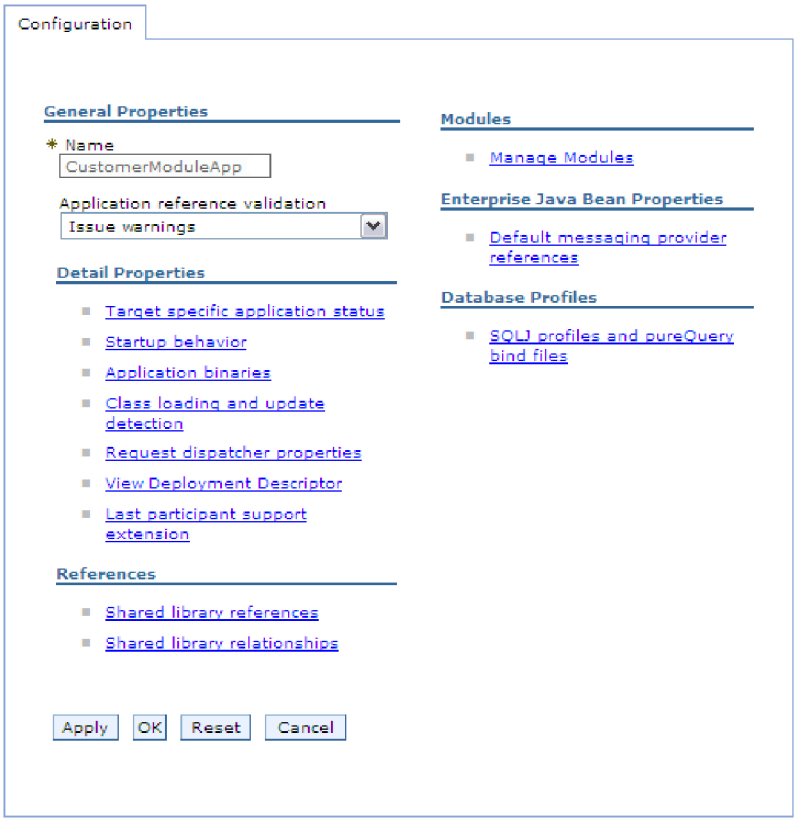


図 31. 「構成」タブでの「モジュールの管理」の選択

6. 「**モジュール**」の下で、「**モジュールの管理**」をクリックします。
7. 「**IBM WebSphere Adapter for JDBC**」をクリックします。
8. 「**追加プロパティー**」リストで、「**リソース・アダプター**」をクリックします。
9. 次のページで、「**追加プロパティー**」リストから「**J2C 接続ファクトリー**」をクリックします。
10. アダプター・モジュールに関連付けられた接続ファクトリーの名前をクリックします。
11. 「**追加プロパティー**」リストで、「**カスタム・プロパティー**」をクリックします。

    カスタム・プロパティーは、IBM WebSphere Adapter for JDBC に特有の J2C 接続ファクトリー・プロパティーです。接続プールおよび拡張接続ファクトリー・プロパティーは、ユーザーが独自にアダプターを作成する場合に構成するプロパティーです。
12. 変更するプロパティーごとに、以下の手順を実行します。

    注: これらのプロパティーについて詳しくは、324 ページの『管理接続ファクトリー・プロパティー』 を参照してください。

    a. プロパティーの名前をクリックします。
    b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。

c. 「**OK**」をクリックします。

13. 「メッセージ」領域で「**保存**」をクリックします。

タスクの結果

アダプター・モジュールに関連付けられた管理接続ファクトリー・プロパティーが変更されます。

組み込みアダプターの活動化仕様プロパティーの設定

アダプターをモジュールの一部としてデプロイした後に、そのアダプターの活動化仕様プロパティーを設定するには、管理コンソールを使用します。 構成するメッセージ・エンドポイント・プロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプター・モジュールを IBM Business Process Manager または WebSphere Enterprise Service Bus 上にデプロイする必要があります。

このタスクについて

活動化仕様プロパティーは、エンドポイントを Inbound 処理用に構成する場合に使用します。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**アプリケーション**」 > 「**アプリケーション・タイプ**」 > 「**WebSphere エンタープライズ・アプリケーション (WebSphere enterprise application)**」と選択します。
5. 「エンタープライズ・アプリケーション」リストから、プロパティーを変更するアダプター・モジュールの名前をクリックします。

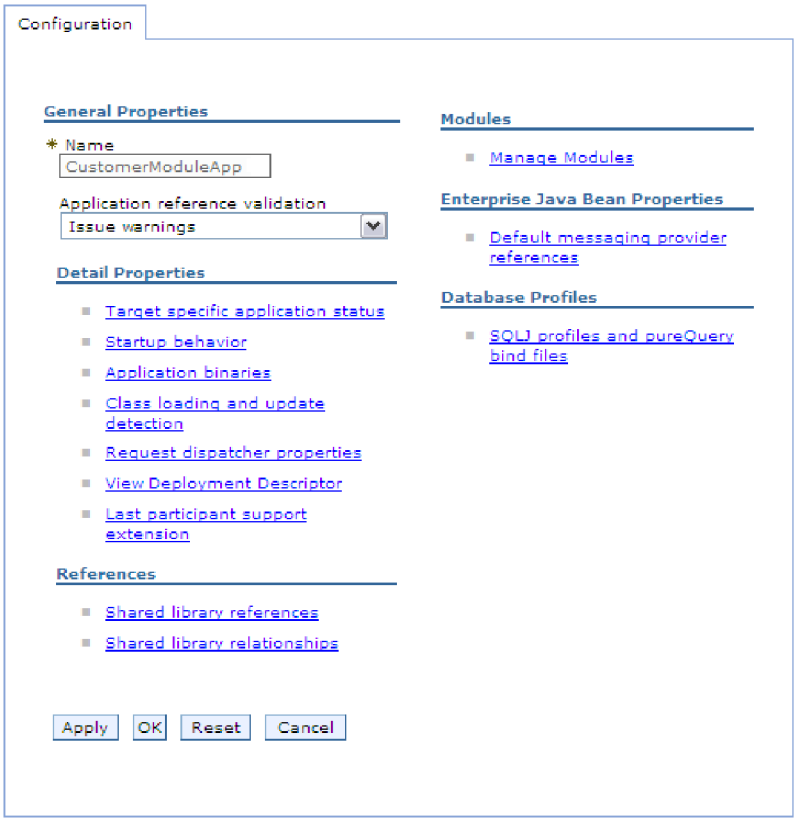


図 32. 「構成」タブでの「モジュールの管理」の選択

6. 「**モジュール**」の下で、「**モジュールの管理**」をクリックします。
7. 「**IBM WebSphere Adapter for JDBC**」をクリックします。
8. 「**追加プロパティー**」リストから、「**リソース・アダプター**」をクリックします。
9. 次のページで、「**追加プロパティー**」リストから、「**J2C 活動化仕様**」をクリックします。
10. アダプター・モジュールに関連付けられている活動化仕様の名前をクリックします。
11. 「**追加プロパティー**」リストから、「**J2C 活動化仕様のカスタム・プロパティー**」をクリックします。
12. 変更するプロパティーごとに、以下の手順を実行します。

    注: これらのプロパティーについて詳しくは、 361 ページの『活動化仕様プロパティー』を参照してください。

    a. プロパティーの名前をクリックします。
    b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
    c. 「**OK**」をクリックします。
13. 「メッセージ」領域で「**保存**」をクリックします。

タスクの結果

アダプター・モジュールに関連付けられている活動化仕様プロパティーが変更されました。

スタンドアロン・アダプターの構成プロパティーの変更

スタンドアロン・アダプターのインストール後に構成プロパティーを設定するには、ランタイム環境の管理コンソールを使用します。アダプターに関する一般情報を入力して、(汎用のアダプター操作に使用される) リソース・アダプター・プロパティーを設定します。アダプターを Outbound 操作に使用する場合は、接続ファクトリーを作成して、それに関するプロパティーを設定します。 アダプターを Inbound 操作に使用する場合は、活動化仕様を作成して、それに関するプロパティーを設定します。

スタンドアロン・アダプターのリソース・アダプター・プロパティーの設定

スタンドアロン・アダプターを IBM Business Process Manager または WebSphere Enterprise Service Bus にインストールした後に、そのアダプターのリソース・アダプター・プロパティーを設定するには、管理コンソールを使用します。構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプターを IBM Business Process Manager または WebSphere Enterprise Service Bus にインストールしておく必要があります。

このタスクについて

カスタム・プロパティーとは、すべての IBM WebSphere アダプターが共有するデフォルト構成プロパティーです。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**リソース**」 > 「**リソース・アダプター**」 > 「**リソース・アダプター**」をクリックします。
5. 「リソース・アダプター」ページで、「**IBM WebSphere Adapter for JDBC**」をクリックします。
6. 「**追加プロパティー**」リストで、「**カスタム・プロパティー**」をクリックします。
7. 変更するプロパティーごとに、以下の手順を実行します。

注: これらのプロパティーについて詳しくは、 318 ページの『リソース・アダプター・プロパティー』 を参照してください。

a. プロパティーの名前をクリックします。

b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。

c. 「**OK**」をクリックします。

8. 「メッセージ」領域で「**保存**」をクリックします。

タスクの結果

アダプターに関連付けられているリソース・アダプター・プロパティーが変更されました。

スタンドアロン・アダプターの管理 (J2C) 接続ファクトリー・プロパティーの設定

スタンドアロン・アダプターを IBM Business Process Manager または WebSphere Enterprise Service Bus にインストールした後に、そのアダプターの管理接続ファクトリー・プロパティーを設定するには、管理コンソールを使用します。構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプターを IBM Business Process Manager または WebSphere Enterprise Service Bus にインストールしておく必要があります。

このタスクについて

管理接続ファクトリー・プロパティーは、ターゲット データベースのインスタンスを構成する場合に使用します。

注: 管理コンソール内では、このプロパティーを「J2C 接続ファクトリー・プロパティー」と呼びます。

管理コンソールを使用してプロパティーを構成するには、以下の手順を使用します。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**リソース**」 > 「**リソース・アダプター**」 > 「**リソース・アダプター**」をクリックします。
5. 「リソース・アダプター」ページで、「**IBM WebSphere Adapter for JDBC**」をクリックします。
6. 「**追加プロパティー**」リストで、「**J2C 接続ファクトリー**」をクリックします。

7. 既存の接続ファクトリーを使用する場合は、既存の接続ファクトリーのリストからの選択に進んでください。

   注: 外部サービス・ウィザードを使用してアダプター・モジュールを構成するときに「**接続プロパティーを指定する**」を選択した場合は、接続ファクトリーを作成する必要がありません。

   接続ファクトリーを作成する場合は、以下の手順を実行します。

   a. 「**新規**」をクリックします。
   b. 「**構成**」タブの「**一般プロパティー**」セクションで、接続ファクトリーの名前を入力します。例えば、`AdapterCF` と入力できます。
   c. 「**JNDI 名**」に値を入力します。 例えば、`com/eis/AdapterCF` と入力できます。
   d. オプション: 「**コンポーネント管理認証別名**」リストから認証別名を選択します。
   e. 「**OK**」をクリックします。
   f. 「メッセージ」領域で「**保存**」をクリックします。

   新規に作成された接続ファクトリーが表示されます。

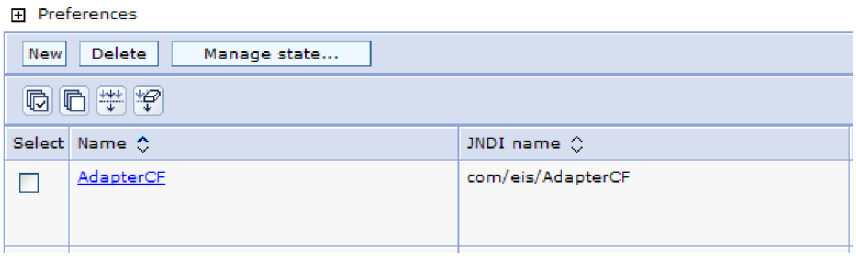


図 33. リソース・アダプターと併用するためのユーザー定義接続ファクトリー

8. 接続ファクトリーのリストで、使用するものをクリックします。
9. 「**追加プロパティー**」リストで、「**カスタム・プロパティー**」をクリックします。

   カスタム・プロパティーは、WebSphere Adapter for JDBC に特有の J2C 接続ファクトリー・プロパティーです。接続プールおよび拡張接続ファクトリー・プロパティーは、ユーザーが独自にアダプターを作成する場合に構成するプロパティーです。
10. 変更するプロパティーごとに、以下の手順を実行します。

    注: これらのプロパティーについて詳しくは、 324 ページの『管理接続ファクトリー・プロパティー』 を参照してください。

    a. プロパティーの名前をクリックします。
    b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
    c. 「**OK**」をクリックします。
11. プロパティーの設定が終了したら、「**適用**」をクリックします。

12. 「メッセージ」領域で「**保存**」をクリックします。

タスクの結果

アダプターに関連付けられている管理接続ファクトリー・プロパティーが設定されます。

スタンドアロン・アダプターの活動化仕様プロパティーの設定

スタンドアロン・アダプターを IBM Business Process Manager または WebSphere Enterprise Service Bus にインストールした後に、そのアダプターの活動化仕様プロパティーを設定するには、管理コンソールを使用します。構成するメッセージ・エンドポイント・プロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプターを IBM Business Process Manager または WebSphere Enterprise Service Bus にインストールしておく必要があります。

このタスクについて

活動化仕様プロパティーは、エンドポイントを Inbound 処理用に構成する場合に使用します。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**リソース**」 > 「**リソース・アダプター**」 > 「**リソース・アダプター**」をクリックします。
5. 「リソース・アダプター」ページで、「**IBM WebSphere Adapter for JDBC**」をクリックします。
6. 「**追加プロパティー**」リストで、「**J2C 活動化仕様**」をクリックします。
7. 既存の活動化仕様を使用する場合は、既存の活動化仕様のリストからの選択に進んでください。

   注: 外部サービス・ウィザードを使用してアダプター・モジュールを構成するときに「**事前定義された接続プロパティーを使用する**」を選択した場合は、活動化仕様を作成する必要はありません。

   活動化仕様を作成する場合は、以下の手順を実行します。

   a. 「**新規**」をクリックします。
   b. 「**構成**」タブの「**一般プロパティー**」セクションで、活動化仕様の名前を入力します。例えば、`AdapterAS` と入力できます。

c. 「**JNDI 名**」に値を入力します。 例えば、com/eis/AdapterAS と入力できます。
d. オプション: 「**認証別名**」リストから認証別名を選択します。
e. メッセージ・リスナー・タイプを選択します。
f. 「**OK**」をクリックします。
g. ページの上部にある「**メッセージ**」ボックスで「**保存**」をクリックします。

新規に作成された活動化仕様が表示されます。

8. 活動化仕様のリストで、使用するものをクリックします。
9. 「追加プロパティー」リストで、「**J2C 活動化仕様のカスタム・プロパティー**」をクリックします。
10. 設定するプロパティーごとに、次の手順を実行します。

注: これらのプロパティーについて詳しくは、361 ページの『活動化仕様プロパティー』を参照してください。

a. プロパティーの名前をクリックします。
b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
c. 「**OK**」をクリックします。

11. プロパティーの設定が終了したら、「**適用**」をクリックします。
12. 「メッセージ」領域で「**保存**」をクリックします。

タスクの結果

アダプターに関連付けられた活動化仕様プロパティーが設定されます。

アダプターを使用するアプリケーションの開始

アダプターを使用するアプリケーションを開始するには、サーバーの管理コンソールを使用します。デフォルトでは、サーバーが始動すると、アプリケーションは自動的に開始します。

このタスクについて

アプリケーションを開始するには、アプリケーションが組み込みアダプターを使用している場合でもスタンドアロン・アダプターを使用している場合でも、この手順を使用します。組み込みアダプターを使用するアプリケーションの場合、アダプターはアプリケーションの開始時に開始されます。 スタンドアロン・アダプターを使用するアプリケーションの場合、アダプターはアプリケーション・サーバーの始動時に開始されます。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。

4. 「**アプリケーション**」 > 「**アプリケーション・タイプ**」 > 「**WebSphere エンタープライズ・アプリケーション**」 とクリックします。

   注: 管理コンソールには、「Integrated Solutions Console」というラベルが付いています。
5. 開始したいアプリケーションを選択します。アプリケーション名は、インストールした EAR ファイルの名前からファイル拡張子 .EAR を除いたものです。
6. 「**開始**」をクリックします。

タスクの結果

アプリケーションの状況が「開始済み」に変化し、アプリケーションが開始されたことを示すメッセージが管理コンソールの上部に表示されます。

アダプターを使用するアプリケーションの停止

アダプターを使用するアプリケーションを停止するには、サーバーの管理コンソールを使用します。デフォルトでは、サーバーが停止すると、アプリケーションは自動的に停止します。

このタスクについて

アプリケーションを停止するには、アプリケーションが組み込みアダプターを使用している場合でもスタンドアロン・アダプターを使用している場合でも、この手順を使用します。アプリケーションと組み込みアダプターの組み合わせの場合、アダプターはアプリケーションの停止時に停止します。 スタンドアロン・アダプターを使用するアプリケーションの場合、アダプターはアプリケーション・サーバーの停止時に停止します。

手順

1. サーバーが稼働していない場合は、「**サーバー**」ビューでご使用のサーバーを右クリックして、「**開始**」を選択します。
2. サーバー状況が「**開始済み**」に変わったら、サーバーを右クリックして「**管理**」 > 「**管理コンソールの実行**」と選択します。
3. 管理コンソールにログオンします。
4. 「**アプリケーション**」 > 「**アプリケーション・タイプ**」 > 「**WebSphere エンタープライズ・アプリケーション**」 とクリックします。

   注: 管理コンソールには、「Integrated Solutions Console」というラベルが付いています。
5. 停止したいアプリケーションを選択します。アプリケーション名は、インストールした EAR ファイルの名前からファイル拡張子 .EAR を除いたものです。
6. 「**停止 (Stop)**」をクリックします。

タスクの結果

アプリケーションの状況が「停止」に変化し、アプリケーションが停止したことを示すメッセージが管理コンソールの上部に表示されます。

Performance Monitoring Infrastructure を使用したパフォーマンスのモニター

Performance Monitoring Infrastructure (PMI) は、管理コンソールの機能の 1 つで、これを使用すると、実稼働環境内で IBM WebSphere Adapter for JDBC を含む、コンポーネントのパフォーマンスを動的にモニターすることができます。PMI は、サーバー内のさまざまなコンポーネントから、平均応答時間や要求の総数などのアダプターのパフォーマンス・データを収集して、そのデータをツリー構造に編成します。このデータは、Tivoli® Performance Viewer (IBM Business Process Manager または WebSphere Enterprise Service Bus の管理コンソールに統合されているグラフィカル・モニター・ツール) を通して表示することができます。

このタスクについて

PMI により、以下の時点のデータを収集することによって、アダプターのパフォーマンスをモニターすることができます。

- Outbound 処理時。Outbound 要求をモニターします。
- Inbound イベントの取り出し時。イベント・テーブルからのイベントの取り出しをモニターします。
- Inbound イベントの送達時。エンドポイント (1 つまたは複数の) へのイベントの送達をモニターします。

ご使用のアダプターに対して PMI を使用可能に設定し、構成するためには、まず、トレースの詳細レベルを設定し、パフォーマンス・データの収集元となるいくつかのイベントを実行する必要があります。

ご使用のアダプター環境の全体的なパフォーマンスをモニターし、それを向上させるために PMI を役立てる方法について詳しくは、 IBM Business Process Manager または WebSphere Enterprise Service Bus の Web サイト (http://www.ibm.com/software/webservers/appserv/was/library/) で PMI を検索してください。

Performance Monitoring Infrastructure の構成

Performance Monitoring Infrastructure (PMI) を、アダプターのパフォーマンス・データ (平均応答時間や要求の総数など) を収集するように構成することができます。使用するアダプター用に PMI を構成した後、Tivoli Performance Viewer を使用してアダプターのパフォーマンスをモニターすることができます。

始める前に

アダプター用に PMI を構成するためには、まずトレースの詳細レベルを設定し、パフォーマンス・データの収集元となるいくつかのイベントを実行する必要があります。

1. トレース機能を使用可能にしてイベント・データを受け取るためには、トレース・レベルを fine、finer、finest、または all のいずれかに設定する必要があります。*=info の後に、コロンとストリングを追加します。例えば、次のように入力します。

```
*=info: WBILocationMonitor.CEI.ResourceAdapter.
*=finest: WBILocationMonitor.LOG.ResourceAdapter.*=finest:
```

トレース・レベルの設定方法については、 231 ページの『Common Event Infrastructure (CEI) を使用したトレースの使用可能化』を参照してください。

2. 1 つ以上の Outbound 要求または Inbound イベントを生成して、構成可能なパフォーマンス・データを生成します。

手順

1. アダプターに対して PMI を使用可能にします。
   a. 管理コンソールで、「**モニターおよびチューニング**」を展開してから、「**Performance Monitoring Infrastructure (PMI)**」を選択します。
   b. サーバーのリストから、ご使用のサーバーの名前をクリックします。
   c. 「構成」タブを選択してから、「 **Performance Monitoring (PMI) を使用可能にする (Enable Performance Monitoring (PMI))**」チェック・ボックスを選択します。
   d. 「**カスタム**」を選択して、選択的に統計を使用可能または使用不可に設定します。

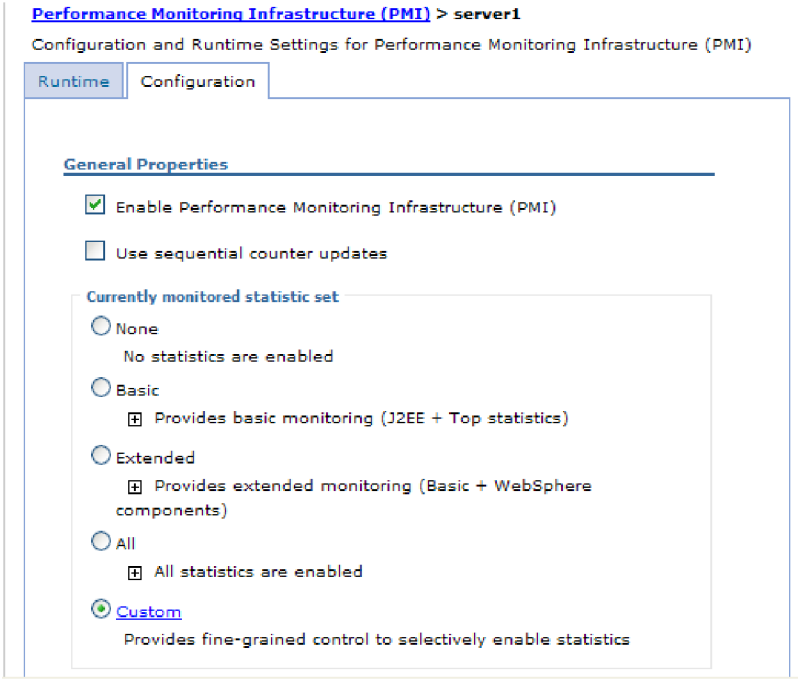


図 34. Performance Monitoring Infrastructure の使用可能化

   e. 「**適用**」または「**OK**」をクリックします。
   f. 「**保存**」をクリックします。 これで、PMI が使用可能になりました。
2. アダプター用に PMI を構成します。
   a. 管理コンソールで、「**モニターおよびチューニング**」を展開してから、「**Performance Monitoring Infrastructure (PMI)**」を選択します。
   b. サーバーのリストから、ご使用のサーバーの名前をクリックします。
   c. 「**カスタム**」を選択します。

d. 「**ランタイム**」タブを選択します。 以下の図は、「ランタイム」タブを示しています。

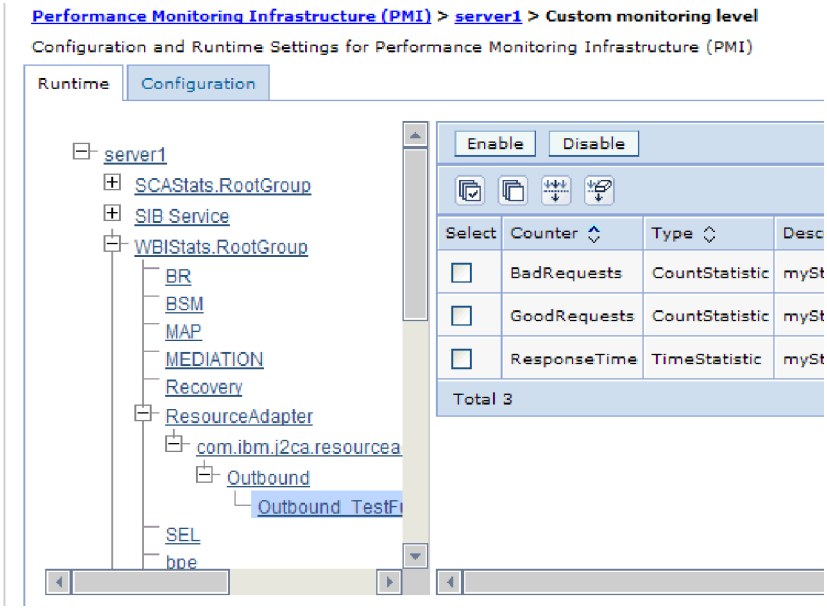


図 35. PMI の構成に使用される「ランタイム」タブ

e. 「**WBIStats.RootGroup**」をクリックします。 これは、ルート・グループで収集されるデータ用の PMI サブモジュールです。この例では、ルート・グループに WBIStats という名前を使用しています。

f. 「**ResourceAdapter**」をクリックします。 これは、JCA アダプターについて収集されるデータ用のサブモジュールです。

g. アダプターの名前をクリックして、モニターするプロセスを選択します。

h. 右側のペインで、収集する統計のチェック・ボックスを選択してから、「**使用可能**」をクリックします。

タスクの結果

PMI がアダプター用に構成されます。

次のタスク

これで、アダプターのパフォーマンス統計を表示することができるようになりました。

パフォーマンスに関する統計の表示

アダプターのパフォーマンス・データは、グラフィカル・モニター・ツール Tivoli Performance Viewer を使用して表示することができます。Tivoli Performance Viewer は、IBM Business Process Manager または WebSphere Enterprise Service Bus の管理コンソールに組み込まれています。

始める前に

アダプターで Performance Monitoring Infrastructure を使用可能にするように構成します。

手順

1. 管理コンソールで、「**モニターおよびチューニング**」を展開し、「**Performance Viewer**」を展開した後、「**現行アクティビティー**」を選択します。
2. サーバーのリストにて、ご使用のサーバーの名前をクリックします。
3. サーバー名の下で、「**パフォーマンス・モジュール**」を展開します。
4. 「**WBIStatsRootGroup**」をクリックします。
5. 「**ResourceAdapter**」およびアダプター・モジュールの名前をクリックします。
6. 複数のプロセスがある場合は、統計を表示させるプロセスのチェック・ボックスを選択します。

タスクの結果

右側のパネルに統計が表示されます。「**グラフの表示**」をクリックして、データのグラフを表示するか、または「**表の表示**」をクリックして、統計を表形式で表示することができます。

以下の図では、アダプターのパフォーマンス統計を表示しています。

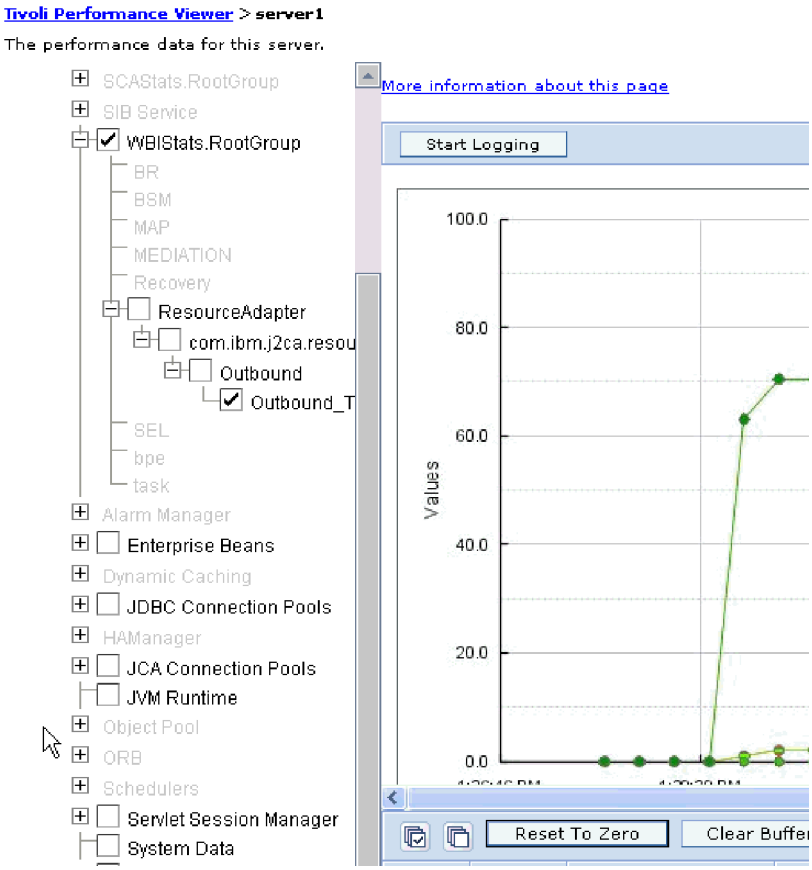


図 36. グラフ表示によるアダプターのパフォーマンス統計

Common Event Infrastructure (CEI) を使用したトレースの使用可能化

アダプターは、サーバー内に組み込まれたコンポーネントである Common Event Infrastructure を使用して、ポーリング周期の開始または停止などの重要なビジネス・イベントに関するデータを通知できます。 イベント・データの書き込み先は、構成設定に応じてデータベースまたはトレース・ログ・ファイルになります。

このタスクについて

トレース・ログ・ファイル内の CEI 項目を報告するには、この手順によって、管理コンソール内で Common Base Event Browser を使用します。

手順

1. 管理コンソールで、「**トラブルシューティング**」をクリックします。
2. 「**ログおよびトレース**」をクリックします。
3. サーバーのリストから、ご使用のサーバーの名前をクリックします。
4. 「**ログ詳細レベルの変更**」ボックスで、アダプターによるイベント・データの書き込み先にする CEI データベースの名前 (例えば、

WBIEventMonitor.CEI.ResourceAdapter.*) またはトレース・ログ・ファイルの名前 (例えば、WBIEventMonitor.LOG.ResourceAdapter.*) をクリックします。

5. アダプターを使用してデータベースまたはトレース・ログ・ファイルに書き込むビジネス・イベントの詳細レベルを選択し、(必要に応じて) メッセージおよびトレースに関連付けられている詳細レベルの細分度を調整します。
   - **ロギングなし**。イベント・ロギングをオフにします。
   - **メッセージのみ**。アダプターはイベントを通知します。
   - **すべてのメッセージおよびトレース**。アダプターは、イベントの詳細を通知します。
   - **メッセージとトレースのレベル**。イベントに関連付けられているビジネス・オブジェクト・ペイロードについてアダプターが通知する詳細度を制御するための設定です。詳細レベルを調整する場合は、以下のオプションのいずれかを選択してください。

     詳細 - 中。アダプターはイベントを通知しますが、ビジネス・オブジェクト・ペイロードについては通知しません。

     詳細 - 高。アダプターは、イベントおよびビジネス・オブジェクト・ペイロードの説明を通知します。

     詳細 - 最高。アダプターは、イベントおよびビジネス・オブジェクト・ペイロード全体を通知します。
6. 「**OK**」をクリックします。

タスクの結果

イベント・ロギングが使用可能になります。CEI 項目は、トレース・ログ・ファイル内で参照できます。または、管理コンソール内で Common Base Event Browser を使用して表示することもできます。

トラブルシューティングおよびサポート

共通のトラブルシューティング手法とセルフ・ヘルプ情報は、問題を迅速に識別して解決するのに役立ちます。

ロギングおよびトレースの構成

要件に合うようロギングおよびトレースを構成します。アダプターのロギングを使用可能にし、イベント処理の状況を制御します。アダプターのログ・ファイル名およびトレース・ファイル名を変更して、ほかのログ・ファイルおよびトレース・ファイルと区別します。

ロギング・プロパティーの構成

管理コンソールを使用して、ロギングを使用可能にし、ログの出力プロパティー (ログの場所、詳細レベル、および出力フォーマットなど) を設定します。

このタスクについて

アダプターでモニター対象イベントをログに記録できるようにするには、まず、モニター対象サービス・コンポーネントのイベント・ポイント、イベントごとに必要となる詳細レベル、およびイベントをログに公開するために使用する出力のフォーマットを指定する必要があります。管理コンソールを使用して、次のタスクを実行します。

- 特定のイベント・ログを使用可能または使用不可に設定する
- ログの詳細レベルを指定する
- ログ・ファイルの格納場所と保持数を指定する
- ログ出力のフォーマットを指定する

  ログ・アナライザーの出力形式を設定した場合は、ログ・アナライザー・ツール (IBM Process Serverに付属するアプリケーション) を使用して、トレース出力を開くことができます。これは、2 つの異なるサーバー・プロセスからのトレースを相関しようとする場合に便利です。なぜなら、これにより、ログ・アナライザーのマージ機能が使用できるからです。

IBM Process Server (サービス・コンポーネントとイベント・ポイントを含む) のモニターの詳細については、ご使用のIBM Process Serverの資料を参照してください。

ログ構成は、静的または動的に変更できます。 アプリケーション・サーバーを開始または再始動すると、静的構成が有効になります。動的構成 (ランタイム構成) の変更は、直ちに適用されます。

ログが作成されると、そのログの詳細レベルが構成データから設定されます。 特定のログ名に対して、構成データが使用可能でない場合、そのログのレベルは、ログの親から取得されます。 親ログに構成データが存在しない場合は、さらにその親ログを検査するという動作を繰り返し、非ヌル・レベルの値を持つログが見つかるまで、ツリーをさかのぼっていきます。 ログのレベルを変更すると、その変更はログの子に伝搬されます。また、必要に応じて、ログの子からその子へと変更が再帰的に伝搬されます。

ロギングを使用可能にし、ログの出力プロパティーを設定するには、以下の手順を実行します。

手順

1. 管理コンソールのナビゲーション・ペインで、「**サーバー**」 > 「**アプリケーション・サーバー**」をクリックします。
2. 操作するサーバーの名前をクリックします。
3. 「**トラブルシューティング**」で、「**ログおよびトレース**」をクリックします。
4. 「**ログ詳細レベルの変更**」をクリックします。
5. いつ変更を有効にするのかを指定します。
   - 構成を静的に変更する場合は、「**構成**」タブをクリックします。
   - 構成を動的に変更する場合は、「**ランタイム**」タブをクリックします。
6. 変更したいロギング・レベルのパッケージの名前をクリックします。
   WebSphere Adapters 用のパッケージ名は、**com.ibm.j2ca.*** で始まります。

- アダプターの基本コンポーネントの場合は、**com.ibm.j2ca.base.*** を選択します。
- アダプターの基本コンポーネントとすべてのデプロイ済みアダプターの場合は、**com.ibm.j2ca.*** を選択します。
- WebSphere Adapter for JDBC と WebSphere Adapter for Oracle E-Business Suite の間で共通のコア・コンポーネントの場合は、**com.ibm.j2ca.dbadapter.core.*** を選択します。
- WebSphere Adapter for JDBC の場合のみ、**com.ibm.j2ca.jdbc.*** パッケージを選択します。

7. ロギング・レベルを選択します。

ロギング・レベル	説明
致命的	タスクを続行できない。または、コンポーネントが機能しない。
重大	タスクを続行できないが、コンポーネントは機能する。このロギング・レベルには、差し迫った致命的エラーを示す (すなわち、リソースが枯渇寸前であることを強く示唆する) 状況も含まれる。
警告	潜在的なエラーが発生したか、重大エラーが差し迫っている。このロギング・レベルには、例えばリソース・リークの可能性など、進行性の障害を示す状況も含まれる。
監査	サーバーの状態やリソースに影響を与える重大なイベントが発生した。
情報	タスクが稼働中である。このロギング・レベルには、タスクの全体的な進行を概説する一般情報が含まれる。
構成	構成の状況が報告されるか、構成変更が発生した。
詳細	サブタスクが稼働中である。このロギング・レベルには、サブタスクの進行を詳細に説明した一般情報が含まれる。

8. 「**適用**」をクリックします。
9. 「**OK**」をクリックします。
10. 静的な構成変更を有効にするには、IBM Process Serverを停止し、再始動します。

タスクの結果

これ以降、ログ項目には、選択したアダプター・コンポーネントについての指定したレベルの情報が格納されます。

ログ・ファイル名およびトレース・ファイル名の変更

アダプター・ログおよびトレース情報を他のプロセスとは分離して保持するには、管理コンソールを使用してファイル名を変更します。デフォルトでは、IBM Process Server上にあるすべてのプロセスおよびアプリケーションのログ情報およびトレース情報は、`SystemOut.log` ファイルおよび `trace.log` ファイルに書き込まれます。

始める前に

アダプター・モジュールをアプリケーション・サーバーにデプロイした後は、ログ・ファイル名およびトレース・ファイル名はいつでも変更できます。

このタスクについて

ログ・ファイルおよびトレース・ファイルは、静的または動的に変更できます。アプリケーション・サーバーを開始または再始動すると、静的変更が有効になります。動的変更またはランタイム構成変更は、即座に適用されます。

ログ・ファイルおよびトレース・ファイルは、*install_root*/profiles/*profile_name*/logs/*server_name* フォルダーにあります。

ログ・ファイル名およびトレース・ファイル名を設定または変更するには、次の手順を実行します。

手順

1. 管理コンソールのナビゲーション・ペインで、「**アプリケーション**」>「**エンタープライズ・アプリケーション**」を選択します。
2. 「エンタープライズ・アプリケーション」リストから、アダプター・アプリケーションの名前をクリックします。これは、アダプターの EAR ファイルの名前から ear ファイル拡張子を除いたものです。 例えば、EAR ファイルの名前が Accounting_OutboundApp.ear である場合は、**Accounting_OutboundApp** をクリックします。
3. 「構成」タブの「モジュール」リストから、「**モジュールの管理**」をクリックします。
4. モジュールのリストで、IBM WebSphere Adapter for JDBC をクリックします。
5. 「構成」タブの「追加プロパティー」の下で、「**リソース・アダプター**」をクリックします。
6. 「構成」タブの「追加プロパティー」の下で、「**カスタム・プロパティー**」をクリックします。
7. 「カスタム・プロパティー」テーブル内で、ファイル名を変更します。
   a. 「**logFilename**」をクリックして、ログ・ファイルの名前を変更します。あるいは、「**traceFilename**」をクリックして、トレース・ファイルの名前を変更します。
   b. 「構成」タブで、「**値**」フィールドに新しい名前を入力します。 デフォルトでは、ログ・ファイルの名前は SystemOut.log、トレース・ファイルの名前は trace.log になります。
   c. 「**適用**」または「**OK**」をクリックします。 変更内容がローカル・マシン上に保存されます。
   d. 変更内容をサーバー上のマスター構成に保存するには、次のいずれかの手順を実行します。
      - **静的変更**: サーバーを停止してから再始動します。この方法では、変更を行うことは可能ですが、サーバーを停止してから始動するまで、行った変更は有効になりません。
      - **動的変更**:「カスタム・プロパティー」テーブルの上にあるメッセージ・ボックス内にある「**保存**」リンクをクリックします。プロンプトが出されたら、再度「**保存**」をクリックします。

First Failure Data Capture (FFDC) サポート

アダプターは、IBM Business Process Manager または WebSphere Enterprise Service Bus の実行時に発生する障害や重大なソフトウェアの問題の永続的な記録を提供する First Failure Data Capture (FFDC) をサポートしています。

FFDC 機能はバックグラウンドで実行され、実行時に発生するイベントやエラーを収集します。この機能はさまざまな障害を相互に関連付ける手段を提供するため、この機能を利用すると、ソフトウェアは、ある 1 つの障害の影響をその原因に結びつけ、その結果、障害の根本原因を素早く突き止めることが容易になります。取り込まれたデータは、アダプターの実行時に発生した例外処理を識別するときに使用できます。

問題が発生すると、例外メッセージおよびコンテキスト・データがアダプターによってログ・ファイルに書き込まれます。このログ・ファイルは *install_root*/profiles/*profile*/logs/ffdc ディレクトリーに置かれます。

First Failure Data Capture (FFDC) について詳しくは、IBM Business Process Manager または WebSphere Enterprise Service Bus の資料を参照してください。

一般的な問題の解決策

ご使用のデータベースで IBM WebSphere Adapter for JDBC を実行するときに発生する可能性のあるいくつかの問題と、その解決策および回避策を説明します。これらの問題および解決策は、ソフトウェア・サポート Web サイトに技術情報として文書化されている問題や解決策を補足するものです。

IBM WebSphere Adaptersについての技術情報の詳細なリストについては、http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm を参照してください。

オブジェクトを選択に追加できない

問題

設計時に、JConnect ドライバーを使用してアダプターで Sybase からストアード・プロシージャーをインポートする場合、IBM WebSphere Adapter for JDBC エンタープライズ・サービスのディスカバリー処理が失敗します。

注: この問題は、jTDS 1.2.2 ドライバーを使用する場合には発生しません。

次のメッセージが生成されます: オブジェクトを選択に追加できません: com.sybase.jdbc2.jdbc.SybSQLException:「CREATE TABLE」は、「tempdb」データベースの複数ステートメント・トランザクション内では許可されていません。

原因

エンタープライズ・サービスのディスカバリー処理において、自動コミットのプロパティー「**データベース接続時に自動コミットを設定**」が選択されておらず、Sybase データベースのストアード・プロシージャーのトランザクション・モードがデフォルト値の「非チェーン・モード」に設定されています。デフォルトの「非チ

ェーン・モード」では、トランザクションを完了させるために、コミット・トランザクションやロールバック・トランザクションと対になった明示的な開始トランザクション・ステートメントが必要です。

解決策

ストアード・プロシージャーの定義を検討して、トランザクションを適切に処理するように変更できるかを判断してください。ストアード・プロシージャーの定義を変更できない場合、外部サービス・ウィザード の「ディスカバリー・プロパティーの指定」ウィンドウから「**データベース接続時に自動コミットを設定**」を選択して、ディスカバリー・プロセスを再実行することができます。

「**データベース接続時に自動コミットを設定**」を選択すると、「非チェーン・モード」構成に関連したデフォルトの処理が自動的に指定変更されます。トランザクション・モードが Sybase データベースでどのように機能するかについて詳しくは、Sybase データベースの資料を参照してください。

注: 外部サービス・ウィザード の「ディスカバリー・プロパティーの指定」ウィンドウから「**データベース接続時に自動コミットを設定**」を選択する場合、外部サービス・ウィザード の最後の画面でも「**データベース接続時に自動コミットを設定**」を選択する必要があります。この最後の画面の「**データベース接続時に自動コミットを設定**」の値は、実行時にアダプターが、データベースとの outbound 接続インスタンスを作成するために使用する管理接続ファクトリー・プロパティーに適用されます。

Oracle 9i または 10g データベースに 4K 以上の CLOB データ型を挿入できない

問題

4 K 以上の CLOB (文字ラージ・オブジェクト) を Oracle 9i または 10g データベースに挿入しようとすると、次の例外が生成されます。

- Oracle 9i: `ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: データベースでの操作が SQL 例外で失敗しました。失敗の理由はソケットから読み取るデータがありません (No more data to read from socket) です。`
- Oracle 10g: `ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: データベースでの操作が SQL 例外で失敗しました。失敗の理由は ORA-01460: 実装されていないまたは不当な変換が要求されました (ORA-01460: unimplemented or unreasonable conversion requested) です。`

原因

4K を超える CLOB を正しくサポートしない古いバージョンのドライバーを使用しています。

解決策

Oracle 10.1.0.2 以降のリリースの Oracle シン・ドライバーを使用してください。

生成された一部のビジネス・オブジェクトには Oracle データベース・オブジェクトの属性がない

問題

Oracle データベース・オブジェクトから生成される一部のビジネス・オブジェクトに、テーブル列の属性がありません。

原因

特定の条件下では、Oracle JDBC ドライバーはデータベース・オブジェクトの列情報を返しません。この問題については、以下のバグを現在 Oracle に提出しています。

- 2281705。シノニムがある場合、DATABASEMETADA.GETCOLUMNS は基盤となるテーブルを返さない
- 2696213。JDBC GETPROCEDURECOLUMNS はプロシージャーのシノニムの列を返さない

また、他のスキーマ内のオブジェクトを参照するプライベート・シノニムが使用されている場合、列情報は返されません。

解決策

シノニムを持つテーブルの場合は、テーブルのシノニムを使用してビジネス・オブジェクトを生成します。

プロシージャーのシノニムの場合は、シノニムの基盤となるオリジナルのプロシージャーを使用してビジネス・オブジェクトを生成します。

他のスキーマ内のオブジェクトを参照するプライベート・シノニムの場合は、オリジナルのテーブルを使用するか、現在のスキーマにシノニムを作成します。

IBM WebSphere Adapter for JDBC を使用して JDBC (タイプ 2 またはタイプ 4) ユニバーサル・ドライバーにより IBM DB2 for z/OS に接続する

問題と原因

DB2 for z/OS は、位置インデックスをデフォルトで使用し、列名を使用しないことによって、アダプターが使用するストアード・プロシージャー・メタデータの照会をサポートします。解決策では、z/OS プラットフォームの DB2 でアダプターを使用する手順を示しています。

解決策

アダプターを使用して DB2 for z/OS に接続するには、以下の接続要件が満たされていることを確認してください。

- ユニバーサル JDBC ドライバーの物理表現は、db2jcc.jar ファイルです。このファイルへのパスが、クラス・パスに設定されていることを確認してください。
- データベース URL: タイプ 2 またはタイプ 4 のどちらのドライバーを使用するかを決定するには、接続の形式を検討します。

タイプ 2: jdbc:db2:database
(例: jdbc:db2:MyDB。MyDB はデータベース名。)

タイプ 4: jdbc:db2://server:port/database
(例: jdbc:db2://9.182.15.129:50000/MyDB。MyDB はデータベース名。)

- ドライバー・クラス: com.ibm.db2.jcc.DB2Driver。

  タイプ 2 ドライバーとタイプ 4 ドライバーの両方で、ドライバー・クラスは同一です。

- クラス・パスに db2jcc_license_cisuz.jar ファイルのパスを設定します。

  タイプ 2 ドライバーとタイプ 4 ドライバーの両方で、ライセンス JAR ファイルは同一です。DB2 for z/OS および DB2 for IBM i サーバーへのアクセスには、有効な DB2 Connect™ ライセンスが必要です。DB2 クライアントは、DB2 Connect ライセンスがなければ、zSeries® サーバーおよび iSeries® サーバーへの直接接続を提供しません。

  DB2 Connect のライセンス交付および使用法について詳しくは、以下のページを参照してください。
  http://www-128.ibm.com/developerworks/db2/library/techarticle/0303zikopoulos1/0303zikopoulos1.html
  http://www-128.ibm.com/developerworks/db2/library/techarticle/0301zikopoulos/0301zikopoulos.html

ウィザードを使用することによるストアード・プロシージャーのメタデータのインポートでは、問題が発生する可能性があります。アダプターを使用して、ストアード・プロシージャーを使用し、DB2 からメタデータをインポートするには、DB2 を以下のステップで説明するように再構成する必要があります。前述のステップに加えて、以下のステップを実行してください。

- DB2 に APAR の PQ62695、PQ55393、PQ56616、PQ54605、PQ46183、および PQ62139 を適用します。
- アダプターでストアード・プロシージャーを使用したい場合は、下記のステップを実行します。これは、PQ62695 のフィックスの一部です。このフィックスでは、JDBC および ODBC 仕様で文書化されているスキーマ・メタデータ API に対応する結果セットを生成することが可能なストアード・プロシージャーが導入されています。

  これらのプロシージャーは、DB2 Universal Driver で提供される JDBC ドライバーおよび ODBC ドライバーによって使用されます。以下のステップを実行して、ストアード・プロシージャーのサポートを使用可能にします。

  1. APAR を適用します。
  2. ZPARM アセンブリー・ジョブ DSNTIJUZ 内の DESCSTAT 変数の値を確認します。 DESCSTAT 変数の値が NO である場合は、YES に変更します。

     注: DESCSTAT のデフォルトは、V7 では NO ですが、V8 では YES に変更されました。
  3. ZPARM モジュールを再アセンブルし、再初期化します。

4. DSNTIJMS という名前の JCL ジョブを実行します。このメンバーは、db2prefix.SDSNSAMP データ・セット内にあります。
5. DB2 を再始動します。

リモート DB2 データベースを用いた outbound サポートのための XA トランザクションの使用

Universal Driver を使用した IBM WebSphere Adapter for JDBC での XA トランザクションの使用

アダプターおよび Universal ドライバーで XA トランザクションを使用してリモート DB2 データベースに接続するには、以下のバージョンのソフトウェアおよび構成プロパティーが必要です。

- DB2 バージョン: 8.2 以降
- JDBC ドライバー: UDB ドライバーのタイプ 4 およびタイプ 2
- XA データ・ソース名: com.ibm.db2.jcc.DB2XADataSource
- XA データベース名: これは、ローカル DB2 クライアントで構成されたリモート・データベース別名です。
- データベース URL: jdbc:db2://hostname:port/databasename
- JDBC ドライバー・クラス: com.ibm.db2.jcc.DB2Driver

リモート DB2 データベースを用いた Outbound サポートのための XA トランザクションの使用

リモート DB2 データベースを使用する IBM WebSphere Adapter for JDBC 用の XA サポートの構成要件について、以下に示します。

リモート DB2 データベースでの XA トランザクションの使用

リモート DB2 データベースの追加

1. DB2 サーバーで **db2admin** (*DB2_InstallPath*¥**SQLLIB¥BIN**) コマンドを実行します。
2. DB2 構成アシスタントを開きます。
3. 「**表示 (View)**」 > 「**拡張表示 (Advanced View)**」に移動します。

次のステップを順番に実行してください。

1. **リモート・システムの追加**
   a. 「**システム**」タブを選択します。
   b. メニューから、「**選択済み**」 > 「**システムの追加 (Add System)**」を選択します。
   c. 「**システム名 (System name)**」フィールドで、ターゲット・データベースが配置されている物理マシン、サーバーシステム、またはワークステーションを指定します。サーバー・システム上のシステム名は、DB2SYSTEM DAS 構成パラメーターによって定義されます。ユーザーはこの値を使用する必要があります。
   d. 「**ホスト名**」フィールドに、ホスト名か、またはターゲット・データベースが存在するインターネット・プロトコル (IP) アドレスを入力します。

e. 「**ノード名 (Node name)**」フィールドで、データベースが配置されているリモート・ノードのローカル・ニックネームを指定します。選択するノード名は、ノード・ディレクトリーまたは管理ノード・ディレクトリーに存在していないものにする必要があります。

f. オペレーティング・システムを選択して、「**OK**」をクリックします。

2. **インスタンス・ノードの追加**

a. 「**インスタンス・ノード (Instance Nodes)**」タブを選択します。

b. メニューから、「**選択済み**」 > 「**インスタンス・ノードの追加 (Add Instance Node)**」を選択します。

c. 「**システム名 (System name)**」フィールドで、ターゲット・データベースが配置されている物理マシン、サーバーシステム、またはワークステーションを指定します。リモート・システムの追加タスクで追加したシステムを選択します。

d. 「**インスタンス名**」フィールドに、ターゲット・データベースが配置されているインスタンスの名前 (DB2 など) を入力します。

e. 「**インスタンス・ノード名 (Instance node name)**」フィールドで、データベースが配置されているカタログされたシステム (ノード) の固有のニックネームを指定します。選択するノード名は、ノード・ディレクトリーまたは管理ノード・ディレクトリーに存在していないものにする必要があります。

f. オペレーティング・システムを選択して、ホスト名を入力します。リモート・システムの追加タスクのステップ 4 と同じホスト名を使用します。

g. リモート DB2 インスタンスが実行されているポート番号を入力します。

h. 「**OK**」をクリックします。

3. **データベースの追加**

a. 「**データベース**」タブを選択します。

b. メニューから、「**選択済み**」 > 「**データベースの追加**」を選択します。

c. 「**インスタンス・ノード (Instance node)**」フィールドで、インスタンス・ノードの追加タスクで作成したインスタンスを選択します。「**データベース名 (Database name)**」フィールドで追加するデータベースの名前を指定します。

d. 「**別名 (Alias)**」フィールドで、ワークステーション上で実行中のアプリケーションが使用できるローカル・ニックネームを指定します。何も入力されない場合、別名はデータベース名と同じになります。別名は、固有でなければなりません。

注: この別名値を、アダプターの XADatabaseName プロパティーに入力する必要があります。

4. **データベース接続のテスト**

a. 「**データベース**」タブを選択します。

b. データベースの追加タスクで追加されたデータベースを選択します。

c. メニューから、「**選択済み**」 > 「**接続のテスト**」を選択します。

d. 「**CLI**」チェック・ボックスを選択し、ユーザー ID とパスワードを入力して、「**接続のテスト**」をクリックします。これにより、接続の成功が戻されます。

イベント・テーブルのトランザクション (XID) 列を調べる

アダプターが送達は 1 回のみとして構成されている場合は、XID 列と共に状況列を使用して、イベントが処理されたかどうかを判別します。

- XID 列に 0 が含まれている場合、イベントはまだ処理対象として選出されていません。
- XID 列にトランザクション ID が含まれている (すなわち 0 ではない) 場合、アダプターはイベントの処理を開始しましたが、処理は完了していません。イベントの処理中にアダプターまたはアプリケーション・サーバーが異常終了した場合は、この組み合わせが発生することがあります。トランザクション・マネージャーはリカバリー時にこれらのトランザクションを COMMIT または ROLLBACK します。

照会 SQL ステートメントからの予期しない結果の処理

照会から予期しない結果を受け取った場合は、トレースをオンにして、照会 SQL をログで確認してください。トレースをオンにすると、テスト・クライアントの場合に、不必要な属性をすべて*設定解除* し忘れていないかを確認できるため、特に便利です。また、入力ビジネス・オブジェクトが正確に入力されているかどうかを判断する場合にも、トレースをオンにすると効果的です。

XA トランザクションをサポートする SQL Server 2000 の構成

XA トランザクションをサポートするように SQL SERVER 2000 を構成するには、以下のようにします。

1. `Microsoft SQL Server 2000 Driver for JDBC¥SQLServer JTA¥` からパス *`sqlserver_install_directory`*`¥MSSQL¥Binn` に sqljdbc.dll をコピーします。
2. SQL クエリ アナライザーを開き、`Microsoft SQL Server 2000 Driver for JDBC¥SQLServer JTA¥instjdbc.sql` を実行します。

IBM WebSphere Adapter for JDBC が SQL Server 2000 JDBC ドライバーによる SQL Server 2000 への接続に失敗する

原因

これは、SQL Server 2000 JDBC ドライバーでの制限です。 .

解決策

以下のいずれかの方法を使用してこの問題を解決できます。

- 値 `SelectMethod=Cursor` をデータベース URL プロパティーに付加する。例: jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=Partner;SelectMethod=Cursor
- SQL Server 2005 JDBC ドライバーを使用する。 SQL Server 2005 JDBC ドライバーは、SQL Server 2000 および SQL Server 2005 の両方を知る方法を提供します。詳しくは、Microsoft のサポート Web サイト (http://msdn.microsoft.com/ja-jp/data/aa937724.aspx) を参照してください。

複数の IBM WebSphere Adapter for JDBC エクスポート・コンポーネントが同一の SCA モジュール内にある

複数のアダプター・エクスポート・コンポーネントが同一の SCA モジュール内に存在する場合、これらのエクスポートでは、同じイベント・テーブルから同じイベント・レコードが取り出されることはなく、そのレコードに対する操作は行われません。複数のエクスポートを定義して同一のイベント・レコードにアクセスする場合は、データベース・デッドロック・エラーなどの潜在的な問題が発生する可能性があります。複数のアダプター・エクスポート・コンポーネントが同一のイベント・テーブルからレコードを取り出すためには、エクスポートごとに異なる EventTypeFilter (処理するイベント・タイプ) 条件を構成して、 1 つのイベント・レコードが複数のエクスポートによって取り出されるリスクを回避する必要があります。

iSeries DB2 において異なる固有の名前があるストアード・プロシージャーに対して正しいビジネス・オブジェクトを作成できない

原因

DB2 iSeries JDBC ドライバー Toolbox for Java™ & JTOpen では、接続プロパティー・ソース「metadata source」を使用して、データベースからの DatabaseMetaData の取得方法が指定されます。これが「0」に設定されていると、メタデータはオブジェクト情報取得 (ROI) データ・フローを使用して取得されます。これが「1」に設定されていると、メタデータはシステム・ストアード・プロシージャーを呼び出すことにより取得されます。バージョン 6.1 以前の場合、デフォルトでは、metadata source は 1 に設定されていません。データベース・メタデータはオブジェクト情報取得 (ROI) データ・フローを使用して取得されます。これが原因で問題が発生します。

DB2 iSeries バージョン 7.1 のデフォルトでは、ストアード・プロシージャーを呼び出すことによってデータベース・メタデータを取得します。

JTOpen ダウンロード Web サイト (http://sourceforge.net/projects/jt400/files/JTOpen-full/6.4/) から入手できる `jtopen_6_4_source¥com¥ibm¥as400¥access¥doc-files¥JDBCProperties.html` に、データベース・メタデータに関する詳細情報が記載されています。

解決策

metadata `source=1` を接続 URL に追加し、DB2 iSeries バージョン 6.1 以前への接続時には「;」の後にスペースがないようにします。例: jdbc:as400://wsbcas12.rtp.raleigh.ibm.com;metadata source=1

成果物を再生成せずにスキーマを変更する

スキーマのみが変更され、表名は変更されていない場合は、以下に説明されているように、Inbound 用の *.export ファイルまたは Outbound 処理用の *.import ファイルと、*.xsd ファイルを編集し、元のスキーマを新規のスキーマに変更します。

import ファイルまたは export ファイルを変更するには、以下のようにします。

1. Java パースペクティブで、テキスト・エディターで *.import ファイルまたは *.export ファイルを開きます。例えば、 JDBCInboundInterface.export などです。

   以下は、JDBCInboundInterface.export のコード・スニペットです。

```
<connection type="com.ibm.j2ca.jdbc.inbound.JDBCActivationSpecWithXid"
listenerType="com.ibm.j2ca.base.ExtendedInboundListener"
selectorType=
"com.ibm.j2ca.extension.emd.runtime.StructuredDataFunctionSelector">
      <properties>
        <databaseURL>jdbc:oracle:thin:@localhost:1521:ORA92
</databaseURL>
        <databaseVendor>ORACLE</databaseVendor>
        <jdbcDriverClass>oracle.jdbc.driver.OracleDriver
</jdbcDriverClass>
        <password>jcajdbc</password>
        <returnDummyBOForSP>false</returnDummyBOForSP>
        <userName>jcajdbc</userName>
</properties>
    </connection>
```

2. 既存のユーザー名とパスワードを、新規のスキーマで使用されるユーザー名とパスワードに変更します。

*.xsd ファイルを変更するには、以下のようにします。

1. テキスト・エディターで *.xsd ファイルを開きます。例えば、 JcajdbcCustomer.xsd などです。

   以下は、 JcajdbcCustomer.xsd のコード・スニペットです。

```
<jdbcasi:TableName>JCAJDBC.CUSTOMER</jdbcasi:TableName>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>BeforeRetrieveSP
</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>JCAJDBC.fn_beforeRetrievesSP
</jdbcasi:StoredProcedureName>
<jdbcasi:ReturnValue>RS</jdbcasi:ReturnValue>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>pkey</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>AfterRetrieveSP
</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>JCAJDBC.sp_afterRetrieveSP
</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
</jdbcasi:Parameters>
<jdbcasi:Parameters>
<jdbcasi:Type>OP</jdbcasi:Type>
<jdbcasi:PropertyName>RS</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
```

2. 表およびプロシージャーの名前のスキーマ名のプレフィックス (BO で定義した場合) を新規のスキーマ名に更新します。例えば、JCAJDBC.CUSTOMER は <NewSchema>.CUSTOMER に変更し、JCAJDBC.fn_beforeRetrievesSP は <NewSchema>.fn_beforeRetrievesSP に変更し、JCAJDBC.sp_afterRetrieveSP は <NewSchema>.sp_afterRetrieveSP に変更します。

ストアード・プロシージャーの検証時に、結果セットに値がまったく含まれていない

問題

アダプターが MS SQL Server 2000 でストアード・プロシージャーを呼び出すように構成されている場合、「構文を検証...」チェック・ボックスを選択した後では、ウィザードは結果セットから値を返しません。

解決策

AutoCommit ='true' に設定するか、またはドライバーを SQL Server バージョン 2005 にアップグレードしてください。

MS SQL Server 用のストアード・プロシージャー名の BIDI サポート

問題

MS SQL Server では、ストアード・プロシージャー名が右から左に書かれる言語で記述されている場合、ストアード・プロシージャーのビジネス・オブジェクト構成ウィンドウが表示されません。ログ・ファイルに例外「java.lang.RuntimeException: com.microsoft.sqlserver.jdbc.SQLServerException: nvarchar 値「??」をデータ型 int に変換する際に変換に失敗しました (java.lang.RuntimeException: com.microsoft.sqlserver.jdbc.SQLServerException: Conversion failed when converting the nvarchar value '??' to data type int)」が表示されます。EMD ウィザードがこの例外を無視して完了した場合、対応するストアード・プロシージャーのビジネス・オブジェクトは生成されません。

解決策

ストアード・プロシージャー名は、左から右に書かれる言語で記述する必要があります。ストアード・プロシージャー名を表す言語を変更してください。

Oracle データベースのスキーマ名 BIDI サポート

問題

Oracle データベースでは、スキーマ名が右から左に書かれる言語で記述されている場合、ストアード・プロシージャーの検証が失敗します。ログ・ファイルに例外「java.sql.SQLException: ORA-06550: 行 xx、列 xx: PLS-00302: コンポーネント「xxx」を宣言する必要があります (java.sql.SQLException: ORA-06550: line xx, column xx: PLS-00302: component 'xxx' must be declared)」が表示されます。EMD ウィザードがこの例外を無視して完了した場合、対応するストアード・プロシージャーのビジネス・オブジェクトは生成されます。ただし、対応する Outbound 操作が処理されるときに、例外が生成されます。

解決策

スキーマ名は、左から右に書かれる言語で記述する必要があります。スキーマ名を表す言語を変更してください。

テーブル名または列名に SQL キーワードまたはその他の特殊文字が含まれていると IBM WebSphere Adapter for JDBC が失敗する

問題

テーブル名または列名に、スペースや単一引用符などの特殊文字、または SQL 予約キーワードが含まれていると、アダプターが失敗します。

原因

テーブル名に特殊文字が含まれるものがあり、例えば、Oracle サーバーでは、「TABLE」、「TABLE 2」、「table」、「table 2」、「'TABLE'」、「'TABLE 2'」などです。または、テーブル内の列に特殊文字が含まれるものがあり、例えば、「DESC」、「DESC 2」、「desc」、「desc 2」、「'COLUMN'」、「'COLUMN 2'」などです。

解決策

用語「TABLE」および「DESC」は SQL 予約キーワードです。これらのテーブル名または列名は SQL ステートメントの処理中に正しく扱われないため、 `TableName` または `ColumnName` のアプリケーション固有情報を編集して、テーブル名または列を二重引用符で囲む必要があります。

以下に、SQL キーワードおよび特殊文字 (スペース) を含むテーブル名および列名の例を示します。

```
<jdbcasi:TableName>YUANJS.table 2</jdbcasi:TableName>
<jdbcasi:ColumnName>DESC</jdbcasi:ColumnName>
```

テーブル名および列名を二重引用符で囲むように ASI を編集します。

```
<jdbcasi:TableName>"YUANJS.table 2"</jdbcasi:TableName>
<jdbcasi:ColumnName>"DESC"</jdbcasi:ColumnName>
```

IBM WebSphere Adapter for JDBC および MySQL データベースのサポート

メタデータ・ディスカバリー中に、JDBC エンタープライズ・メタデータ・ディスカバリー (EMD) はまずデータベース固有の JDBC ドライバーからスキーマ情報を取得し、その後でスキーマの選択に応じて、他のデータベース・エンティティー (テーブル、プロシージャー、ビュー、シノニムなど) に関する情報を取得します。しかし、MySQL の場合、JDBC ドライバーはスキーマ情報を返しません。これは、MySQL の JDBC ドライバーの実装環境の場合、バージョン 5.0.4 より前のバージョンではデータベース・スキーマに関する情報を返さないためです。したがって、ディスカバリーおよび成果物の生成をそれ以上進めることはできません。

IBM i でのテーブルおよびトリガーの生成用のサンプル SQL スクリプト

問題

JDBC J2EE コネクター・アーキテクチャー (JCA) のリソース・アダプターには、scripts_db2.sql という DB2 用のサンプル・スクリプトが付属しています。IBM i 上の UDB では、DB2 サンプル・スクリプトの SQL 構文が記述どおりに機能しません。

原因

UDB には、 SCHEMA.TABLE 形式の完全修飾スキーマ SQL が必要です。現在のスクリプトには、修飾されたスキーマは含まれません。

解決策

IBM i で使用できるように scripts_db2.sql を変更して、テーブル・スクリプトとトリガー・スクリプトのテーブル名とトリガー名に完全修飾スキーマ名を使用するようにしてください。 例として、scripts_db2.sql からのテーブル作成スクリプトのサンプルを以下に示します。

```
CREATE TABLE customer
  (
   pkey VARCHAR(10) NOT NULL PRIMARY KEY,
   fname VARCHAR(20),
   lname VARCHAR(20),
   ccode VARCHAR(10)
);
```

IBM i で使用するためには、以下に示すようにスクリプトを変更し、<schema_name> を完全修飾スキーマ名に置き換えます。

```
CREATE TABLE <schema_name>.customer
(
  pkey VARCHAR(10) NOT NULL PRIMARY KEY,
  fname VARCHAR(20),
  lname VARCHAR(20),
  ccode VARCHAR(10)
);
```

IBM WebSphere Adapter for JDBC がテーブル内の行をロックしてしまい、それによって他のアプリケーションがタイムアウトする

問題

アダプターは、グローバル・トランザクションの一部として組み込まれていて、データベース・テーブル内の共有データにアクセスしていました。データベース・テーブル内のこの共有データには、他のアプリケーションもアクセスしていました。グローバル・トランザクションの一部として、アダプターは、行をロックしたままにしていたため、他のアプリケーションが同じ共有データにアクセスしようとするとタイムアウトが発生しました。

解決策

共有データにアクセスする場合は、アダプターをグローバル・トランザクションから除去します。その代わりに、ビジネス・レベルで何らかの手段を使用して、ビジネス・データの整合性が保たれるようにしてください。例えば、変数を使用して、共有データへの操作が成功したかどうかを示し、このデータへの重要な操作をログに記録します。

誤ったバイナリー・データが返される (照会 DB2/AS400)

問題

バージョン 6.1 のアダプターを使用して DB2/AS400 に対する照会を実行すると、IBM i ではバイナリー・フィールドが EBCDIC 文字として返されます。例えば、フィールドは 9910001761 の代わりに 4040f9f9f1f0f0f0f1f7f6f1 を返します。AS400 のフィールドは、 BINCHAR として定義されていて、CCSID 65535 のタグが付けられています。

原因

IBM i データベース内のフィールドには、CCSID 65535 のタグが付けられています。Toolbox の JDBC ドライバーは、この CCSID を変換すべきでないフィールドとして認識します。

解決策

変換したいフィールドに、有効な CCSID のタグを付けます。あるいは、「translate binary」接続プロパティーを「true」に設定することもできます。これは、CCSID 65535 のタグが付けられたフィールドを含む、すべてのフィールドを変換するように JDBC ドライバーに指示するものです。これを簡単に行う方法としては、データベースへの接続時に、使用される URL の終わりに「;translate binary=true」を追加します。

戻り値のあるストアード・プロシージャーの使用時のエラー

問題

戻り値があるストアード・プロシージャーを処理すると、アダプターが失敗します。生成されるエラー・メッセージは、「プロシージャー 'GetPolicyCount' はパラメーター '@count' を予測していましたが、提供されませんでした。(Procedure 'GetPolicyCount' expects parameter '@count', which was not supplied.)」のようなものになります。

原因

アダプターは現在、戻り値があるストアード・プロシージャーの処理をサポートしていません。以下の例は、アダプターでサポートされないことが原因でエラーを引き起こしたエンタープライズ・メタデータ・ディスカバリーで生成された XSD 内の行を強調表示しています。

次の例では、状況値を返すストアード・プロシージャーを使用するときに、生成される XSD 内に、戻り値に対する以下のような参照 (太字で示しています) があります。

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/pocusergetpolicycount"
xmlns:pocusergetpolicycount=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/pocusergetpolicycount"
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
schemaLocation="JDBCASI.xsd"/>
```

```
<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi"
asiNSURI="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="PocuserGetpolicycount">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:TableName>POCUser.GetPolicyCount</jdbcasi:TableName>
<jdbcasi:StatusColumnName>returnvalue</jdbcasi:StatusColumnName>
<jdbcasi:StatusValue></jdbcasi:StatusValue>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>GetPolicyCount</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>OP</jdbcasi:Type>
<jdbcasi:PropertyName>returnvalue</jdbcasi:PropertyName>
</jdbcasi:Parameters>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>ipolicynum</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="returnvalue" type="int" minOccurs="1" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:ColumnName>@RETURN_VALUE</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="ipolicynum" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:ColumnName>@iPolicyNum</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

解決策

ストアード・プロシージャーの実行をエラーなしで処理するために、戻り値へのすべての参照 (上記の太字で示したもの) を XSD ファイルから削除します。上記の例の XSD を変更したものは、以下のようになっているはずです。

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/pocusergetpolicycount"
```

```
xmlns:pocusergetpolicycount=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/pocusergetpolicycount"
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
 schemaLocation="JDBCASI.xsd"/>

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi"
asiNSURI="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="PocuserGetpolicycount">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:TableName>POCUser.GetPolicyCount</jdbcasi:TableName>
<jdbcasi:StatusColumnName>returnvalue</jdbcasi:StatusColumnName>
<jdbcasi:StatusValue></jdbcasi:StatusValue>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>GetPolicyCount</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>ipolicynum</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="ipolicynum" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:ColumnName>@iPolicyNum</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

外部サービス・ウィザードの始動時にクラス・ローダー違反が発生する

問題

データ・パースペクティブでデータベースへの接続を使用した後に、外部サービス・ウィザードを使用することができません。ウィザードの 2 番目のパネルの最後に、例外

com.ibm.adapter.framework.api.ImportException が生成されます。
理由: pc:0 でクラス・ロードの制約に違反がありました
(クラス: oracle/jdbc/driver/OracleConnection
メソッド: getWrapper()Loracle/jdbc/OracleConnection;)。

このエラーは、以下のどちらの状態でも発生します。

- 外部サービス・ウィザードを通じてデータベースへの接続を確立すると、データ・パースペクティブからデータベースに接続しようとしたときにエラーが発生します。
- データ・パースペクティブを通じてデータベースへの接続を確立すると、外部サービス・ウィザードからデータベースに接続しようとしたときにエラーが発生します。

原因

データ・パースペクティブとウィザードは独自のクラス・ローダーを使用するため、エラーが発生します。DLL (JDBC ドライバーが使用するネイティブ・ライブラリー) は、データ・パースペクティブにいったんロードすると、ウィザードに再度ロードできなくなります。JVM には、どの時点でも 1 つのクラス・ローダーのみがネイティブ・ライブラリーをロード可能であるという、固有の制約事項があります。そのため、クラス・ローダー A が DLL B をロードした場合、クラス・ローダー A が解放され、ガーベッジ・コレクションが実行されるまで、その他のクラス・ローダーは DLL B をロードできません。実際にはユーザーはガーベッジ・コレクションを制御できないため、通常、別のクラス・ローダーで DLL B をロードする場合は、JVM を再始動する必要があるということになります。この制限は、既知のものであり、IBM WebSphere Application Server では文書化されています。

解決策

このエラーが発生したときは、IBM Integration Designer を再始動することが唯一の解決策です。

Oracle 10g と共に XA を使用すると接続のクローズのエラーが発生する

問題

Oracle 10g を使用して XA トランザクションを実行するためにアダプターが使用されると、アダプターは接続のクローズ例外「javax.resource.ResourceException: Closed Connection」を戻します。

原因

これは、Oracle 10g データベース・ドライバーの既知の問題です。この問題については、Oracle で「3488761 Connection closed error from OracleConnection.getConnection() - 10G drivers」というバグが記録されています。

解決策

このバグは、Oracle 10g リリース 2 ドライバーで修正されています。予備手段としては、Oracle 9i JDBC Thin ドライバーを使用して、XA トランザクションのためにデータベースに接続します。

Oracle でのトランザクションの開始中のエラー

問題

Oracle データベースを使用して XA トランザクションを実行するためにアダプターが使用されると、次のエラーが生成されます: WTRN0078E: トランザクション・マネージャーがトランザクションのリソースで start を呼び出そうとして、エラーが発生しました。エラー・コードは XAER_RMERR でした。

解決策

Oracle ディレクトリーに含まれているスクリプト initxa.sql および initjvm.sql を実行します。

```
<ORACLE_HOME>javavm¥install
file: initxa.sql
file: initjvm.sql
```

これらのスクリプトを実行するために必要な許可を持つためには、SYSOPER または SYSDBA として Oracle にログオンしていなければならないため、おそらくこのアクティビティーは、Oracle データベース管理者が実行する必要があります。

initxa.sql スクリプトは、データベースを XA 用に構成します。正常に実行されると、データベースは XA 向けに構成されます。このスクリプトは、一度目の試行で正常に実行される場合もあります。データベースのメモリー・スペースが小さすぎると、正常に実行されないことがあります。

これを修正するには、initjvm.sql スクリプトを実行します。これにより、調整が必要なパラメーターが示されます。パラメーターは、次のファイルに格納されます。

```
<ORACLE_HOME>¥database
file: init<DATABASE_SID>.ora
```

表 21に、通常増加させる必要のある 2 つのパラメーターを示します。ご使用の特定のデータベース構成では、異なるパラメーターの調整が必要となる可能性があります。

表 21. 標準的なパラメーター・サイズ

パラメーター名	最小値
java_pool_size	12000000
shared_pool_size	24000000

Outbound 処理中に ResourceException が発生する

ResourceException を受け取った場合は、根本原因のフィールドを調べて原因を判別します。よくある問題の根本原因を以下に示します。

- SQLException 例外

  SQLException にテキスト UserID or password is invalid が含まれている場合は、Outbound 接続に指定されたユーザー ID またはパスワードが正しくありません。

  例えば、次のようになります。

  ```
  javax.resource.ResourceException: [ibm][db2][jcc][t4][2013][11249] Connection
  authorization failure occurred.  Reason: User ID or Password invalid.
  ```

- ConnectException 例外

ConnectException に、connection refused または could not establish connection to the server というテキストが含まれている場合は、データベース・サーバーが作動可能ではないか、または接続を妨げるネットワーク問題が発生している可能性があります。

例えば、次のようになります。

```
javax.resource.ResourceException: [ibm][db2][jcc][t4][2043][11550] Exception
java.net.ConnectException: Error opening socket to server /9.26.237.55 on port
50,000 with message: Connection refused: connect.
```

Inbound 処理中に ResourceException が発生する

この例外は、データベース接続時に繰り返される問題があることを示します。イベントをポーリングするには、アダプターはデータベースに接続する必要があります。接続が失敗した場合、アダプターは、構成済みの一定の時間を待機した後、データベースへの接続を再試行します。アダプターは、構成された回数だけ接続を試行した後、ポーリングを停止します。アダプターは、ポーリングを停止した場合、ResourceException を戻します。

テスト・クライアントで RetrieveAll 操作または Retrieve 操作を処理していると RecordNotFoundException が生成される

問題

IBM Integration Designer テスト・クライアントで RetrieveAll 操作または Retrieve 操作を処理するときに、WHERE 条件 (SELECT ステートメント内) において不要な属性がブランクのままであると、RetrieveAll 操作または Retrieve 操作は失敗し、「RecordNotFoundException: Record not found in EIS」という例外が生成されることがあります。

解決策

テスト・クライアントで、必要とされる属性の値を <unset> に設定します。RetrieveAll 操作を処理します。再び例外が生成される場合は、データベース表に一致するレコードが存在しないと考えられます。

IBM WebSphere Adapter for JDBC が RecordNotFoundException をスローする

問題

アダプターがデータベースからデータを取得しようとしたときに、要求されたキーの項目がない場合、アダプターは、空またはヌルのオブジェクトを返すのではなく、「5/13/09 12:28:29:332 GST com.ibm.j2ca.base.exceptions.RecordNotFoundException」というような例外をスローします。

原因

RetrieveAll 操作を処理するときに、データベースから返すレコードがないと、アダプターは RecordNotFoundException を生成します。アダプターは空の項目を正しく処理しません。

問題

IBM サポートに連絡して、アダプターの暫定フィックス V6.0.2.3IF10 を入手してください。このフィックスでは、管理接続ファクトリーに errorOnEmptyResultset という新規プロパティーが追加されています。このプロパティーのデフォルト値は、「true」で、RetrieveAll 操作に対して行が返されないと、 RecordNotFoundException がスローされます。

RetrieveAll 操作の動作をオーバーライドするには、アプリケーションのデプロイ後に、管理コンソールを使用して errorOnEmptyResultset プロパティーの値を変更し、「False」に設定することができます。レコードが検出されないと、RetrieveAll 操作の処理後にアダプターは空のコンテナーを返します。このプロパティーは、このフィックスの EMD では構成できないため、管理コンソールで変更する必要があります。

以下に示すように、「errorOnEmptyResultset」プロパティーを *.import ファイルに追加することもできます。

```
<connection type="com.ibm.j2ca.jdbc.JDBCManagedConnectionFactory"
interactionType="com.ibm.j2ca.jdbc.JDBCInteractionSpec">
<properties>
<databaseURL>
jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=adapter
</databaseURL>
<jdbcDriverClass>
com.microsoft.jdbc.sqlserver.SQLServerDriver</jdbcDriverClass>
<password>adrienne</password>
<userName>sa</userName>
<errorOnEmptyResultset>false</errorOnEmptyResultset>
</properties>
```

データの取得中にレコードが検出されない場合に JDBC JCA アダプター 6.x が例外をスローする

253 ページの『IBM WebSphere Adapter for JDBC が RecordNotFoundException をスローする』 を参照してください。

ストアード・プロシージャーの ResultSet ASI に、結果セットを返すための手操作による介入が必要である

問題

IBM WebSphere Adapter for JDBC エンタープライズ・メタデータ・ディスカバリー・プロセスは、結果セットを返し、動詞 ASI としてビジネス・オブジェクトに添付されるストアード・プロシージャーについては、ResultSet ASI を設定しません。

原因

Oracle データベースだけが、出力パラメーターとして結果セットを返します。その他のすべてのデータベースについては、ストアード・プロシージャーが結果セットを返すかどうかを判別する方法はありません。

解決策

結果セットを返し、動詞 ASI としてビジネス・オブジェクトに添付されるすべてのストアード・プロシージャーに対して、手動で ResultSet ASI を true に設定し、ストアード・プロシージャーが結果セットを返すように指示してください。

次の例で、ビジネス・オブジェクト内で ASI を設定する方法を示します。

```
<jdbcasi:Operation>
<jdbcasi:Name>RetrieveAll</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveAllSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>SCOTT.GETCUSTS1</jdbcasi:StoredProcedureName>
<jdbcasi:ResultSet>true</jdbcasi:ResultSet>
<jdbcasi:Parameters>
```

エンタープライズ・サービス・ディスカバリーは、生成したビジネス・オブジェクトに MaxNumOfRetRS および ResultSet ASI の値を保存できない

解決策

ウィザードで、まず「**ストアード・プロシージャーから返される結果セットの最大数**」フィールドの値を設定して、ストアード・プロシージャーを検証すると、「**ストアード・プロシージャーから返される結果セットの最大数**」フィールドの値は自動的に 0 にリセットされ、それまでの値は無視されます。生成されたビジネス・オブジェクトに値を正しく保存するには、まずストアード・プロシージャーを正常に検証してから、「**ストアード・プロシージャーから返される結果セットの最大数**」フィールドに値を指定してください。

『ストアード・プロシージャーおよびストアード関数の選択および構成』も参照してください。

IBM WebSphere Adapter for JDBC が、単一のスキーマで同じ名前を持つ複数のストアード・プロシージャーをサポートしていない

『ストアード・プロシージャーおよびストアード関数の選択および構成』を参照してください。

ビジネス・オブジェクト属性の Date 型 ASI が、 Date および Time 型の列を参照する

問題

JDBC Enterprise Metadata Discovery は、すべての Time、TimeStamp、および Date のデータベース列を、ビジネス・オブジェクト内のサービス・データ・オブジェクトの Date 型の属性にマップします。SDO Date 型は、Date 型属性が、データベースに保管された実際の時間ではなく、 GMT で表示されるような方法で処理されます。値が SDO またはデータベースに設定されるときに、時間部分は消失します。

解決策

SQL の型が DATE、TIME、および TIMESTAMP である列について、JDBC エンタープライズ・メタデータ・ディスカバリーは対応する属性の型を String に設定し、DataType という名前の、属性のアプリケーション固有情報が追加されます。この値は、Date、Time、TimeStamp のいずれかに設定されます。適切な属性の型およびア

プリケーション固有情報を持つビジネス・オブジェクト XSD を生成するには、JDBC エンタープライズ・メタデータ・ディスカバリーを再実行してください。

Outbound 中にビジネス・オブジェクトに日時の値を設定するときには、以下の形式を使用します。

- 日付形式は yyyy-mm-dd です。
- 時刻形式は hh:mm:ss です。
- タイム・スタンプ形式は yyyy-mm-dd hh:mm:ss です。

同様に、Inbound については、キー値がイベント・テーブル内の object_key 列について date または time 型である場合、上記で説明した形式で値を入力する必要があります。

XML データ型および XQuery が J2CA JDBC アダプターで直接サポートされない

「How J2CA Adapter works with XML datatype in DB2」を参照してください。

BPEL が特定のビジネス・フォールトを catch できない

「BPEL が特定のビジネス・フォールトを catch できない」を参照してください。

グローバル・トランザクションでの例外によって Outbound が失敗する

問題

Outbound がグローバル・トランザクションでの例外によって失敗し、トランザクションを正常にロールバックできません。

原因

バージョン 6.2 以降、IBM WebSphere Adapter for JDBC は、例外 (IntegrityConstraintViolationException、 MatchesExceededLimitException、 MissingDataException、 MultipleMatchingRecordsException、 ObjectNotFoundException、 RecordNotFoundException、 および UniqueConstraintViolatedException) 用にビジネス・オブジェクトをサポートしています。これらのいずれかの例外がスローされると、EIS バインディングによって、対応するフォールト・ビジネス・オブジェクト (IntegrityConstraintFault、 MatchesExceededLimitFault、 MissingDataFault、 MultipleMatchingRecordsFault、 ObjectNotFoundFault、 RecordNotFoundFault、 UniqueConstraintFault など) として例外がカプセル化されます。

EIS バインディングは、フォールト・ビジネス・オブジェクトを受け取るとすぐに、 ServiceRuntimeException の代わりに ServiceBusinessException をスローします。グローバル・トランザクションで、コンポーネントが ServiceRuntimeException をスローすると、トランザクション・マネージャーはグローバル・トランザクションをロールバックします。しかし、コンポーネントが ServiceBusinessException をスローすると、トランザクション・マネージャーはグローバル・トランザクションをコミットします。

解決策

1. .import ファイル内のすべてのフォールト・ビジネス・オブジェクト・バインディングを削除します。

```
<methodBinding
      inDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.db2admi
      ncustom erbg.Db2adminCustomerBGDataBinding"
      method="createDb2adminCustomerBG"
      outDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.db2adm
      incusto merbg.Db2adminCustomerBGDataBinding">
      <faultBinding fault="INTEGRITY_CONSTRAINT_VIOLATION"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI     mpl"/>
      <faultBinding fault="MISSING_DATA"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI     mpl"/>
      <faultBinding fault="OBJECT_NOTFOUND_EXCEPTION"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI     mpl"/>
      <faultBinding fault="UNIQUECONSTRAINT_VIOLATION"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI     mpl"/>
      <interaction>
        <properties>
            <functionName>Create</functionName>
        </properties>
      </interaction>
  </methodBinding>
```

2. 以下に示すように、ビジネス・フォールトを処理するようにフォールト・ハンドラーを定義します。

```
DataObject bo = null;
  try {
   bo = (DataObject) locateService_JDBCOutboundInterfacePartner().invoke
("createDb2adminAddressBG",createDb2adminAddressBGInput);
  } catch (ServiceBusinessException e) {
   // TODO Auto-generated catch block
   throw new ServiceRuntimeException(e);
  }
```

アダプターの Outbound に例外がある場合は常に、 ServiceRuntimeException が生成され、グローバル・トランザクションはロールバックされます。それ以外の場合、グローバル・トランザクションはコミットされます。

EAR ファイル内の 1 つ以上のモジュールとの JDBC アダプターの共用

組み込みデプロイメントによって、 EAR ファイル内の 1 つ以上のモジュールと JDBC アダプターを共用するには、以下のようにします。

1. IBM Integration Designer のビジネス・インテグレーション・パースペクティブで、「**ファイル**」>「**新規**」>「**外部サービス**」と選択します。
2. IBM WebSphere Adapter for JDBC を選択し、Outbound サービスまたは Inbound サービスを作成します。各種のデータベースに接続するために、 JDBC ドライバーをアダプター・コネクター・プロジェクトに確実に追加します。アダプターを各種のデータベースに接続したい場合、すべての JDBC ドライバーを一度にアダプター・コネクター・プロジェクトに追加できます。
3. 既存のコネクター・プロジェクトを選択して、もう 1 つ Outbound または Inbound サービスを作成します。それまでに成果物用に作成したものと同じモジュールを使用します。

4. Inbound または Outbound サービスを作成したら、選択したモジュールの「依存関係」ページの「**J2EE**」エリアに、 JDBC アダプター・コネクター・プロジェクトのみがリストされているかどうかを確認します。

詳しくは、「EAR ファイル内の 1 つ以上のモジュールとの JDBC アダプターの共用」を参照してください。

イベント・テーブルでの複合キー用の object_key 列値の設定

問題

IBM WebSphere Adapter for JDBC がポーリングするイベント・テーブルには、複合基本キーとして設定されている列がいくつかあります。アダプターがイベントをポーリングできるように object_key 列をイベント・テーブルに設定するには、どうすればよいでしょうか?

解決策

次の例で、このシナリオについて説明します。以下に、データベース内の Employee テーブルを示します。

```
CREATE TABLE ADMIN.EMPLOYEE ( EMPLOYEEID VARCHAR (10) NOT NULL ,
JOBCODE VARCHAR (10) NOT NULL , EMPLOYEENAME VARCHAR (10) NOT NULL ,
AGE VARCHAR (10) NOT NULL , CONSTRAINT CC1182727951922 PRIMARY KEY (
EMPLOYEEID, JOBCODE) ) ;
```

EmployeeID と JobCode は両方とも、Employee テーブル内の複合キーです。

Employee テーブルにデータを挿入するには、以下の SQL ステートメントを使用します。

```
INSERT INTO employee (employeeid,jobcode,employeename,age) VALUES
('1','8','Mike','30');
```

「Employee」テーブルには、1 (EmployeeID) と 8 (JobCode) の両方に「Mike」という従業員のレコードがあります。

トリガーを使用してイベント・テーブルにレコードを挿入する場合は、以下のような SQL ステートメントが必要です。

```
INSERT INTO wbia_jdbc_eventstore (object_key, object_name,
object_function, event_priority, event_status)
VALUES ('1;8', 'AdminEmployeeBG', 'Create', 1, 0);
```

ESD を実行して「Employee」テーブルのビジネス・オブジェクトを生成した後、そのビジネス・オブジェクトから生成されるキー属性は両方とも、テーブル定義と同じ順序になります。正しい順序は、「EmployeeID」が先頭で、「JobCode」がその次です。

したがって、「object_key」の値をイベント・テーブルに挿入するときは、ビジネス・オブジェクト定義とテーブル定義を同じ順序にする必要があります。例えば、「EmployeeID」列の値が「1」で、「JobCode」列の値が「8」の場合、「object_key」の正しい値は、「8;1」ではなく、「1;8」です。

さらに、デリミッターとしてセミコロン文字 (;) を使用して、「object_key」列内の各複合キー値を区切る必要があります。

ユーザー定義イベント照会用の WebSphere Business Integration Adapter for JDBC での IBM WebSphere Adapter for JDBC の使用

問題

ユーザー定義イベント照会に、IBM WebSphere Adapter for JDBC の WBIA_JDBC_EventStore イベント・テーブルではなく、WebSphere Business Integration Adapter for JDBC の XWORLDS_EVENTS イベント・テーブルを使用すると、エラーが発生し、「フィールド object_function がありません (field object_function not there)」または「無効な列名です (Invalid column name)」というメッセージが生成されます。

原因

WebSphere Business Integration Adapters のイベント・テーブル XWORLDS_EVENTS に、フィールド「object_function」がありません。また、イベント・テーブル WBIA_JDBC_EVENTSTORE に、フィールド「connector_ID」がありません。

解決策

以下に、マイグレーション・ステップを示します。

1. connector_ID でテーブル「WBIA_JDBC_EVENTSTORE」を作成します。

   ```
   CREATE TABLE WBIA_JDBC_EventStore
   (
   event_id INTEGER NOT NULL PRIMARY KEY,
   xid VARCHAR(200),
   object_key VARCHAR(80) NOT NULL,
   object_name VARCHAR(40) NOT NULL,
   object_function VARCHAR(40) NOT NULL,
   event_priority INTEGER NOT NULL,
   event_time TIMESTAMP default CURRENT TIMESTAMP NOT NULL,
   event_status INTEGER NOT NULL,
   connector_ID VARCHAR(40),
   event_comment VARCHAR(100)
   );
   ```

   いくつかのイベントを JDBC アダプター・インスタンスからフィルター処理するために、「connector_ID」フィールドが使用されます。「connector_ID」によって、現在、connectorID フィルター処理を使用している WebSphere Business Integration Adapter のお客様がシームレスに JCA にマイグレーションすることができます。この機能により、お客様は同じタイプのイベントが数多くある場合にロード・バランシングを行うことができます。

2. 以下の SQL ステートメントを作成して実行します。

   ユーザー定義の照会:

   ```
   select event_id, object_key, object_name, object_verb as object_function ,
   connector_id from xworlds_events where event_status = 0
   ```

   ユーザー定義の update:

   ```
    update xworlds_events set event_status= 1 where event_id = ?
   ```

   ユーザー定義の delete:

   ```
   delete from xworlds_events where event_id= ?
   ```

IBM WebSphere Adapter for JDBC の Inbound サービス用のアーカイブ・イベント機能の実装

「WebSphere Adapter for JDBC の Inbound サービス用のアーカイブ・イベント機能の実装」を参照してください。

IBM WebSphere Adapter for JDBC アダプターが Inbound 処理中にネイティブ・メソッド用のメソッドを検出できない

問題

アダプターが、Inbound 処理中にネイティブ・メソッド用のメソッドを検出できません。

原因

アダプターは、内部でネイティブ・メソッドを使用して、サポートされる各操作にマップします。Inbound 操作とネイティブ・メソッドの関係は、エクスポート・コンポーネント・プロパティーのメソッド・バインディングに定義されます。

アダプターは、プレフィックス「emitCreateAfterImage」と、アダプターのイベント・テーブルから取得されるオブジェクト名を使用することで、 Create 操作用のネイティブ・メソッドを構成します。アダプターのエクスポート・コンポーネントに定義されたネイティブ・メソッドが emitCreateAfterImageAdminCustomerBG で、アダプターのイベント・テーブルに挿入されたオブジェクト名が AdminCUSTOMERBG の場合は、2 つのネイティブ・メソッドの名前が互いに一致しないため、エラーが生成されます。

解決策

この問題を修正するには、次のいずれかの方法を使用できます。

- オブジェクト名として AdminCustomerBG をイベント・テーブルに挿入します。または
- アダプターのエクスポート・コンポーネントのプロパティーで、emitCreateAfterImageAdminCustomerBG を emitCreateAfterImageAdminCUSTOMERBG に置き換えます。

BLOB 列を含む結果セットから内容を取得中に、ドライバーによって NULL ポイント例外が生成される

問題

アダプターが DB2 ストアード・プロシージャーで返される BLOB 列を含む結果セットから内容を取得しようとすると、ドライバーによって NULL ポイント例外が生成されます。

原因

DB2 JCC ドライバーのバージョンが 4.7.85 です。

解決策

DB2 JCC ドライバー 3.50.152 を使用してください。

Oracle の Date 型が、Date ではなく dateTime にマップされる

問題

Oracle の一部のフィールドにおいて、Date データ型が Date ではなく dateTime にデフォルトでマップされます。

原因

これは、Oracle JDBC ドライバーの問題です。Oracle データベースにおける Date 型は、JDBC 仕様に定義されている Timestamp 型と似ています。 Date 型には、時間の情報も含まれています。アダプターは、ドライバーから戻されたデータ型に基づいて、JDBC 型を SDO 型にマップします。ドライバーが JDBC 型の Date を戻すと、アダプターはそれを SDO 型の Date にマップします。ドライバーが JDBC 型の Timestamp を戻すと、アダプターはそれを SDO 型の dateTime にマップします。

解決策

目的の型に手動でマップできます。

Oracle での XML データ型のサポート

問題 1

xmlparserv2.jar がアプリケーション・ライブラリーに追加されると、アプリケーションをデプロイしようとするときに管理コンソールで例外が生成されます。

原因

xmlparserv2.jar がアプリケーション・ライブラリーに追加されると、XML 処理インターフェースである DocumentBuilderFactory、SAXParserFactory、および TransformerFactory の実装として登録されます。xmlparserv2.jar が TransformerFactory の実装として登録されると、互換性の問題から、アプリケーションのデプロイメント・フェーズで以下の例外が生成されます。

```
java.lang.IllegalArgumentException
 at oracle.xml.jaxp.JXTransformer.setOutputProperty(JXTransformer.java:793)
 at org.eclipse.xsd.util.DefaultJAXPConfiguration.createTransformer
  (DefaultJAXPConfiguration.java:63)
 at org.eclipse.xsd.util.XSDResourceImpl.doSerialize(XSDResourceImpl.java:153)
 at org.eclipse.xsd.util.XSDResourceImpl.serialize(XSDResourceImpl.java:136)
 at org.eclipse.xsd.ecore.XSDEcoreBuilder.setAnnotations(XSDEcoreBuilder.java:3087)
 at com.ibm.ws.bo.BOModelBuilder.access$201(BOModelBuilder.java:103)
 at com.ibm.ws.bo.BOModelBuilder$1.run(BOModelBuilder.java:1778)
 at java.security.AccessController.doPrivileged(AccessController.java:202)
 at com.ibm.ws.bo.BOModelBuilder.setAnnotations(BOModelBuilder.java:1774)
 at org.eclipse.xsd.ecore.XSDEcoreBuilder.getEPackage(XSDEcoreBuilder.java:161)
 at com.ibm.ws.bo.BOModelBuilder.getEStructuralFeature(BOModelBuilder.java:983)
 at org.eclipse.xsd.ecore.XSDEcoreBuilder.generate(XSDEcoreBuilder.java:2709)
 at com.ibm.ws.bo.BOModelBuilder.generate(BOModelBuilder.java:340)
 at com.ibm.ws.bo.BOModelBuilder.generate(BOModelBuilder.java:650)
 at com.ibm.ws.bo.BOModelBuilder.build(BOModelBuilder.java:309)
 at com.ibm.ws.bo.BOModelHolder.loadModels(BOModelHolder.java:591)
 at com.ibm.ws.bo.BOModelHolder.loadAllModels(BOModelHolder.java:626)
```

```
at com.ibm.ws.bo.BOModelHolder.loadNamespace(BOModelHolder.java:545)
at com.ibm.ws.bo.BOEPackageRegistry.loadEPackage(BOEPackageRegistry.java:218)
at com.ibm.ws.bo.BOEPackageRegistry.getEPackage(BOEPackageRegistry.java:295)
```

解決策

<WPS_HOME>/java/jre/lib/ の下で jaxp.properties.sample を jaxp.properties にリネームすることで、互換性のある実装をクラス・ローダーにロードさせるようにします。さらに、jaxp.properties ファイルで、javax.xml.transform.TransformerFactory、javax.xml.parsers.SAXParserFactory、および javax.xml.parsers.DocumentBuilderFactory の各プロパティーのコメントを外します。サーバーを再始動します。

注: Oracle で、JDNI データ・ソースを使用してデータベースに接続している場合は、データ・ソースの作成時に必ず xdb.jar と xmlparserv2.jar をクラスパスに追加してください。

問題 2

Oracle (ドライバー・バージョン 11.1.0.7.0) で、XML データ型があるテーブルにおいて、Retrieve 操作および RetrieveAll 操作が正しくない結果を戻します。

原因

Oracle のドライバー・バージョン 11.1.0.7.0 では、複合型の XML 型属性の内容は正しく読み取られません。ドライバーからは常にヌルが戻されます。

解決策

ドライバー・バージョン 11.2.0.1.0 を使用してください。

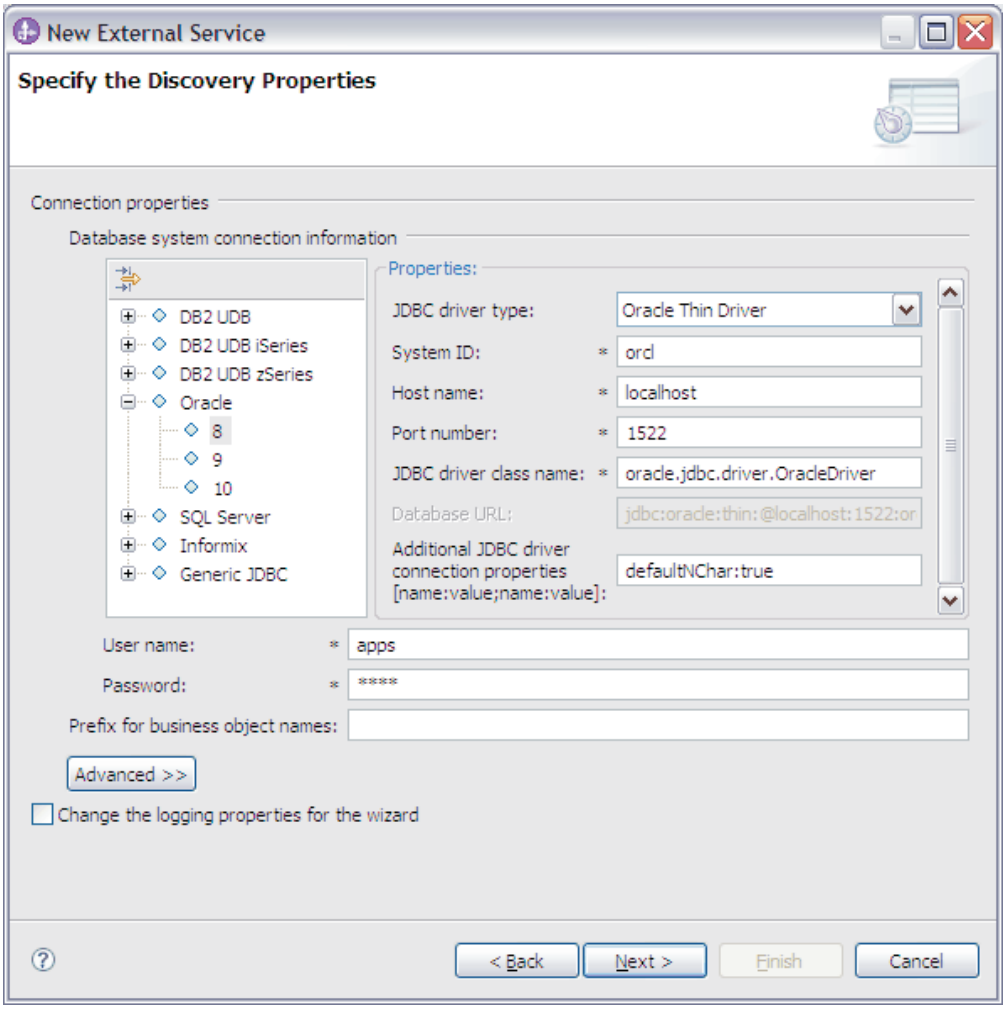
Oracle NCHAR 型または NVARCHAR 型を取り扱う際のエンコード問題

問題

IBM WebSphere Adapter for JDBC には、 Oracle NCHAR 型または NVARCHAR 型を取り扱う際のエンコードの問題があります。Retrieve 操作および RetrieveAll 操作は、NCHAR 型または NVARCHAR 型の列の場合には読めないコードを返します。

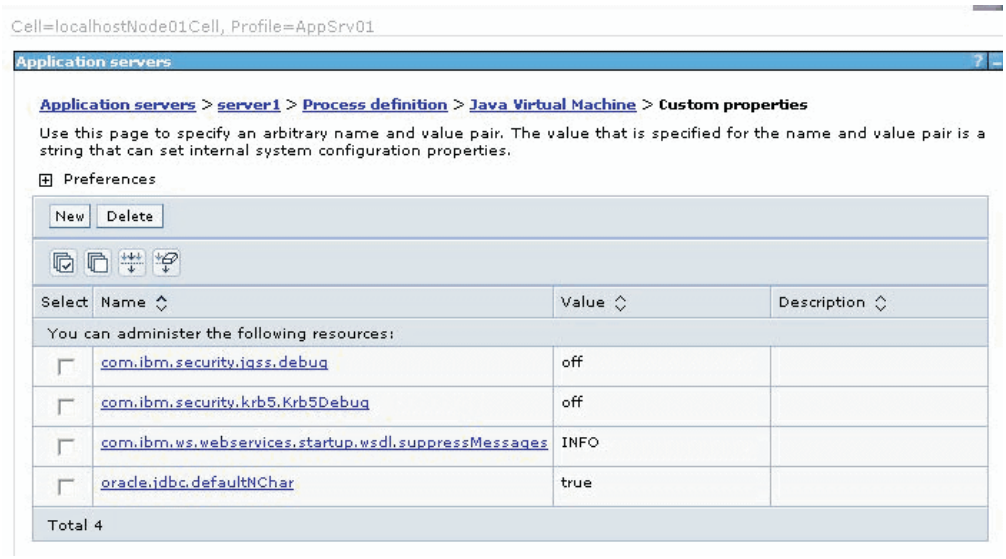
解決策 1

接続タイプとして接続プロパティー (URL) を使用する場合、以下の図に示すように接続プロパティー defaultNChar:true を追加します。Retrieve 操作および RetrieveAll 操作は、NCHAR 列または NVARCHAR 列に対して正しい値をリストします。



解決策 2

接続タイプとして接続プロパティー (URL) を使用しない場合、図に示すように Java 仮想マシン・プロパティー oracle.jdbc.defaultNChar=true を追加し、サーバーを再始動します。Retrieve 操作および RetrieveAll 操作は、NCHAR 列または NVARCHAR 列に対して正しい値をリストします。



IBM WebSphere Adapter for JDBC は外部サービス・ディスカバリー中にユーザー定義関数のリストを表示しない

問題

外部サービス・ディスカバリー中に、アダプターがユーザー定義関数のリストを表示しません。

解決策

DB2 JDBC ドライバーでの制限のため、この機能はサポートされていません。

アダプターによりバージョン競合例外メッセージが返される

問題

CWYBS_AdapterFoundation.jar のバージョンが異なるアダプターを複数インストールしており、実行時に古いバージョンの CWYBS_AdapterFoundation.jar がロードされると、バージョンの競合が発生し、アダプターにより ResourceAdapterInternalException エラー・メッセージが返されます。例えば、Oracle E-Business Suite アダプター バージョン 7.0.0.3 と WebSphere Adapter for JDBC バージョン 7.5 をインストールした場合、次のエラー・メッセージが表示されます。IBM WebSphere Adapter for JDBC には、バージョン 7.0.0.3 のファイル /C:/IBM/WebSphere/ProcServer7/profiles/ProcSrv01/installedConnectors/CWYOE_OracleEBS.rar/CWYBS_AdapterFoundation.jar がロードされています。しかし、要求されているこの jar のベース・レベルはバージョン 7.5 です。CWYBS_AdapterFoundation.jar のバージョンが異なるアダプターを複数インストールしている場合、バージョンの競合が発生し、アダプターにより ResourceAdapterInternalException メッセージが返されます。この競合を解決するには、すべてのアダプターを同じバージョン・レベルにマイグレーションする必要があります。さらに支援が必要な場合は、WebSphere Adapters サポートにアクセスしてください。

解決策

すべてのアダプターを同じバージョン・レベルにマイグレーションしてください。

さらに支援が必要な場合は、 http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Familyにアクセスしてください。

IBM WebSphere Adapter for JDBC はテーブルとタイプでスキーマが異なる場合はテーブル BO 内の Oracle システム複合タイプをサポートしない

問題

Oracle テーブルで、列タイプがシステム・タイプ・スキーマの場合、そのスキーマを Oracle JDBC ドライバーは取り出すことができません。

原因

これは、 Oracle JDBC ドライバーの制約が原因です。

解決策

テーブルのスキーマとタイプのスキーマが同じであるようにしてください。

操作でユーザー定義タイプを扱う際に例外がスローされる

問題 1

ARRAY 型または STRUCT 型に含まれる ARRAY 属性または LOB 属性が、RetrieveAll、UpdateAll、DeleteAll、または Exists の各操作における SQL ステートメントの WHERE 節で使用されたものである場合、データベースは以下の例外をスローします。

ORA-22901: ネストした表、オブジェクト型の VARRAY 属性または LOB 属性を比較できません。

問題 2

アダプターは、 `null` に設定された属性値を持つ STRUCT データ型を使用したときにレコードが検出されないと、RecordNotFoundException エラーを生成します。

原因

これは、 Oracle データベースの制約が原因です。

解決策 1

対応するユーザー定義の型 (STRUCT、ARRAY) 属性が、SQL ステートメントの where 節を構成するのに使用された場合、それらを設定解除します。

解決策 2

RecordNotFoundException を回避するためには STRUCT 型を設定解除します。

古いバージョンの IBM WebSphere Adapter for JDBC が新しいバージョンにマイグレーションされるときに例外がスローされる

問題

原因

階層ビジネス・オブジェクト内の、1 次キー - 外部キー関係を持つビジネス・オブジェクト属性のデータ型が異なります。

例えば、 Customer という名前の親ビジネス・オブジェクトにおいて、親ビジネス・オブジェクトの 1 つの属性 (Customer) が外部キーとして機能し Payment という名前の子ビジネス・オブジェクトでは、子ビジネス・オブジェクトの 1 つの対応する属性 (Payment) が基本キーとして機能するとします。親ビジネス・オブジェクト内の外部キーのデータ型は STRING であり、子ビジネス・オブジェクト内の基本キーのデータ型は INT です。1 次キー - 外部キー関係のデータ型の不整合が原因で、新しいバージョンのアダプターでの実行時に IntegrityConstraintViolationException メッセージが表示されます。

解決策

1 次キー - 外部キー関係を持つ属性のデータ型が整合するようにしてください。

IBM WebSphere Adapter for JDBC が WebSphere Application Server DataSource と共に実行するときに例外がスローされる

問題

IBM WebSphere Adapter for JDBC は、 WebSphere Application Server DataSource と共に実行すると、以下の StaleConnectionException をスローします。

```
Caused by: com.ibm.websphere.ce.cm.StaleConnectionException:
Io exception: Software caused connection abort: socket write errorDSRA0010E:
SQL State = 08006, Error Code = 17,002
 at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
 at sun.reflect.NativeConstructorAccessorImpl.newInstance
(NativeConstructorAccessorImpl.java:44)
 at sun.reflect.DelegatingConstructorAccessorImpl.newInstance
(DelegatingConstructorAccessorImpl.java:39)
 at java.lang.reflect.Constructor.newInstance(Constructor.java:516)
 at com.ibm.websphere.rsadapter.GenericDataStoreHelper.
mapExceptionHelper(GenericDataStoreHelper.java:605)
 at com.ibm.websphere.rsadapter.GenericDataStoreHelper.
mapException(GenericDataStoreHelper.java:667)
 at com.ibm.ws.rsadapter.AdapterUtil.mapException(AdapterUtil.java:2111)
 ... 75 more
---- Begin backtrace for Nested Throwables
java.sql.SQLRecoverableException: Io exception: Software caused connection abort:
socket write errorDSRA0010E: SQL State = 08006, Error Code = 17,002
 at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java:101)
 at oracle.jdbc.driver.DatabaseError.newSQLException(DatabaseError.java:133)
 at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:199)
 at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:263)
 at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:521)
 at oracle.jdbc.driver.T4CConnection.doRollback(T4CConnection.java:634)
 at oracle.jdbc.driver.PhysicalConnection.rollback(PhysicalConnection.java:3470)
 at oracle.jdbc.OracleConnectionWrapper.rollback(OracleConnectionWrapper.java:135)
 at com.ibm.ws.rsadapter.spi.WSRdbSpiLocalTransactionImpl.
rollback(WSRdbSpiLocalTransactionImpl.java:604)
```

原因

これは、次の理由で発生します。

- データベースが開始していなかったため、アダプターがデータベースに接続しようとして失敗します。
- この接続はデータベース障害のためにもう使用できません。以前に取得した接続をアダプターが使用しようとすると、その接続は無効になります。この場合、アダプターが現在使用中のすべての接続が、データベースに接続しようとしたときにエラーをスローします。
- アダプターが、現在 stale 状態の接続を取得します。

解決策

preTestSQLString を構成して、空きプールから取得する各接続をテストします。これによって問題は解決しますが、取得する接続の数によっては、パフォーマンスに影響することがあります。詳しくは、http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tdat_pretestconn.htmlを参照してください。

enableHASupport が True に設定されていると、パッシブ・アダプター・インスタンスのエンドポイント・アプリケーションがイベントを listen する

問題

アダプターのアクティブ/パッシブ構成モードで、enableHASupport プロパティーが True に設定されている場合でも、パッシブ・アダプター・インスタンスのエンドポイント・アプリケーションもイベント/メッセージを listen します。

原因

WebSphere Application Server バージョン 7.0 では、デフォルトで、JMS 活動化仕様の alwaysactivateAllMDBs プロパティーが True に設定されます。このため、すべてのアダプター (アクティブ/パッシブ) インスタンスのエンドポイント・アプリケーションで、イベントの listen が有効になっています。

解決策

パッシブ・アダプター・インスタンスのエンドポイント・アプリケーションで、イベントの listen を止めるようにするには、alwaysactivateAllMDBs プロパティー値を False に設定する必要があります。 JMS 活動化仕様は 1 つ以上の MDB に関連付けられていて、イベントの受信に必要な構成を提供しています。
alwaysActivateAllMDBs プロパティーが False に設定されていると、アクティブ・アダプター・インスタンスのエンドポイント・アプリケーションのみがイベントを受け取ります。

alwaysActivateAllMDBs プロパティーを False に設定するには、以下の手順を実行します。

1. 管理コンソールにログオンします。
2. 「**リソース**」>「**JMS**」>「**活動化仕様**」に移動します。

3. リストからアプリケーションに対応する活動化仕様を選んでクリックします。
4. 「**追加プロパティー**」で、「**カスタム・プロパティー**」をクリックします。
5. 「**alwaysActivateAllMDBs**」をクリックします。
6. 値を **False** に変更します。
7. 「**適用**」に続いて「**OK**」をクリックします。

結果

アクティブ・アダプター・インスタンスのエンドポイント・アプリケーションのみがイベントを listen するようになります。

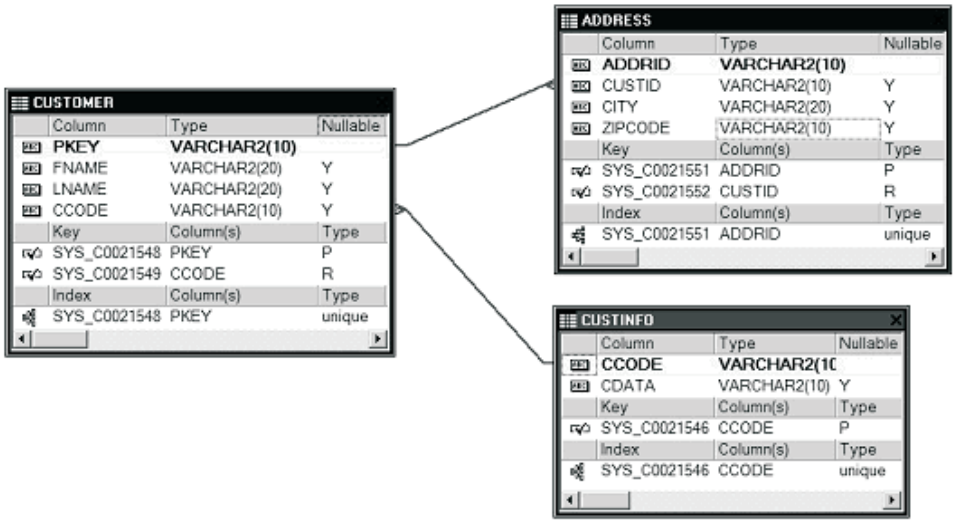
親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の単一/複数カーディナリティー関係の作成

問題

IBM WebSphere Adapter for JDBC を使用しているときに、親ビジネス・オブジェクトと子ビジネス・オブジェクトとの間に単一カーディナリティー関係または複数カーディナリティー関係を作成するには、どのようにすればいいでしょうか?

解決策

下の図のように、データベースには 3 つのテーブルがあります。CUSTOMER テーブルには、ADDRESS テーブルに対応する複数のレコードが含まれていて、CUSTINFO テーブルには、CUSTOMER テーブルに対応する複数のレコードが含まれています。



SQL スクリプトを使用してこれらのテーブルを作成するには、以下のプロシージャーを参照してください。

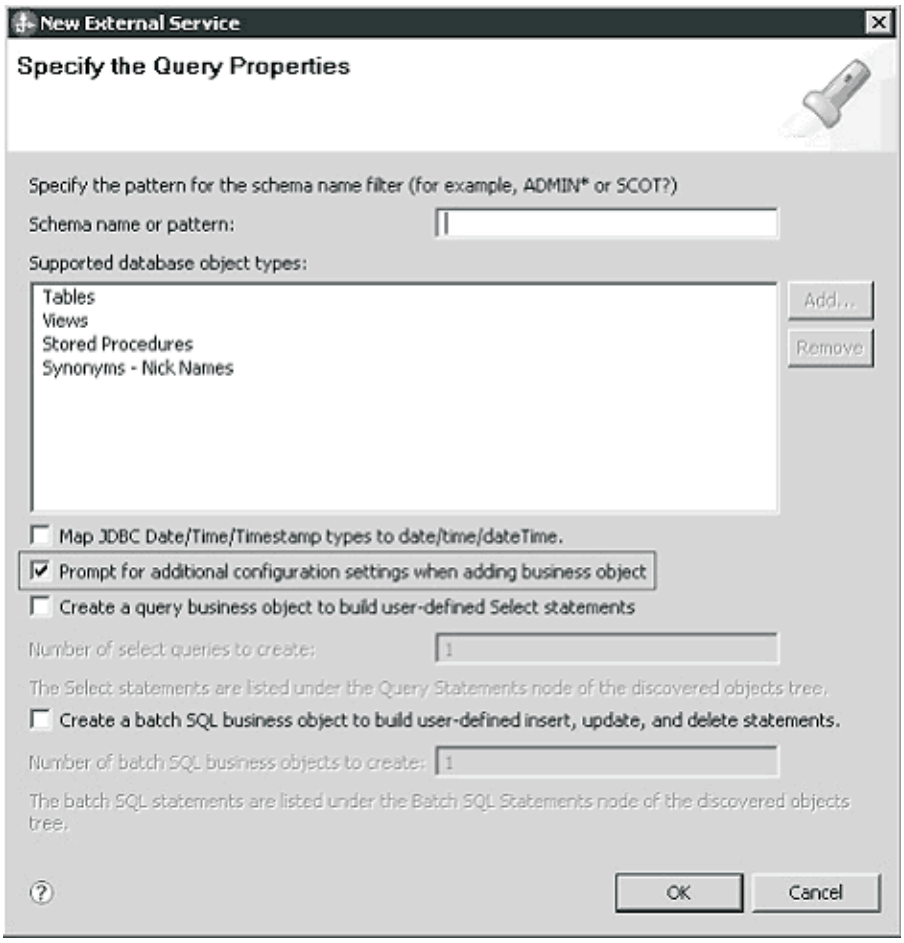
```
--------------------
--0. clean up
--------------------
DROP TABLE address;
DROP TABLE customer;
DROP TABLE custinfo;
```

```
--------------------
--1.create "custinfo"
--------------------
CREATE TABLE custinfo
(
  ccode    VARCHAR(10) NOT NULL  PRIMARY KEY,
  cdata    VARCHAR(10)
);

--------------------
--2.create "customer"
--------------------
CREATE TABLE customer
(
  pkey    VARCHAR(10) NOT NULL  PRIMARY KEY,
  fname    VARCHAR(20),
  lname    VARCHAR(20),
  ccode    VARCHAR(10),
  FOREIGN KEY(ccode) REFERENCES custinfo(ccode)
);

--------------------
--3.create "address"
--------------------
CREATE TABLE address
(
  addrid    VARCHAR(10) NOT NULL  PRIMARY KEY,
  custid    VARCHAR(10),
  city      VARCHAR(20),
  zipcode  VARCHAR(10),
  FOREIGN KEY(custid) REFERENCES customer(pkey)
);

--------------------
--4. prepare for data
--------------------
--1. custinfo
insert into custinfo (CCODE, CDATA)
values ('1', '1');

insert into custinfo (CCODE, CDATA)
values ('2', '2');

insert into custinfo (CCODE, CDATA)
values ('3', '3');

insert into custinfo (CCODE, CDATA)
values ('4', '4');

insert into custinfo (CCODE, CDATA)
values ('5', '5');

insert into custinfo (CCODE, CDATA)
values ('6', '6');

insert into custinfo (CCODE, CDATA)
values ('7', '7');

insert into custinfo (CCODE, CDATA)
values ('8', '8');

insert into custinfo (CCODE, CDATA)
values ('9', '9');

insert into custinfo (CCODE, CDATA)
values ('10', '10');
```

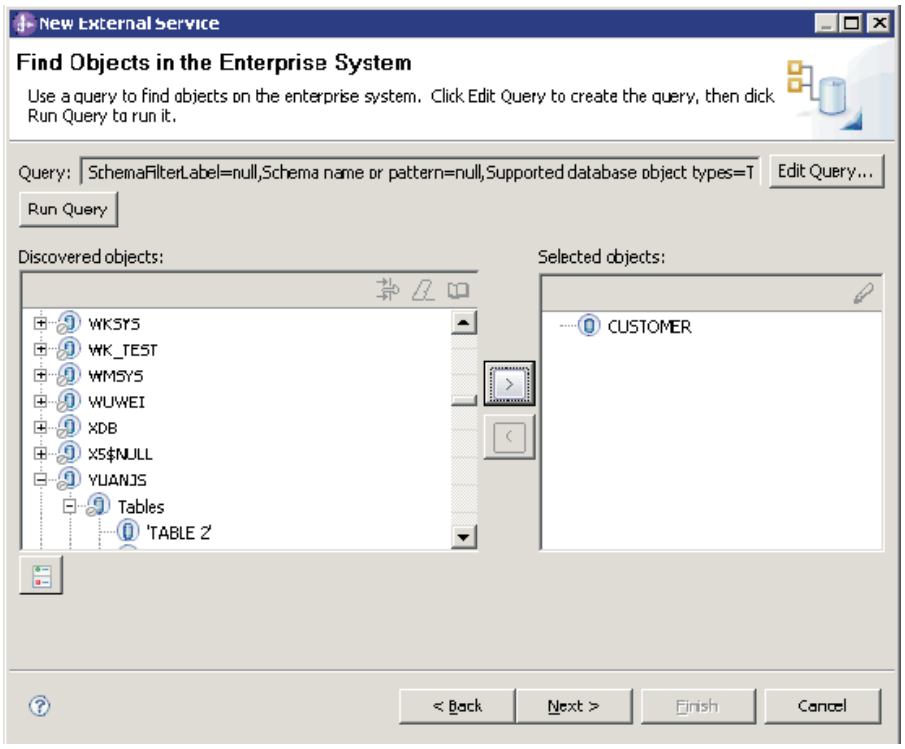
```
--2. customer
insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('1', 'fname', 'lname', '1');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('2', 'fname', 'lname', '2');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('3', 'fname', 'lname', '3');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('4', 'fname', 'lname', '4');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('5', 'fname', 'lname', '5');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('6', 'fname', 'lname', '6');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('7', 'fname', 'lname', '7');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('8', 'fname', 'lname', '8');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('9', 'fname', 'lname', '9');

insert into customer (PKEY, FNAME, LNAME, CCODE)
values ('10', 'fname', 'lname', '10');

--3. address
insert into address (ADDRID, CUSTID, CITY, ZIPCODE)
values ('1', '1', 'BeiJing', '100000');

insert into address (ADDRID, CUSTID, CITY, ZIPCODE)
values ('2', '1', 'ShangHai', '200000');

insert into address (ADDRID, CUSTID, CITY, ZIPCODE)
values ('3', '2', 'NanJin', '300000');

insert into address (ADDRID, CUSTID, CITY, ZIPCODE)
values ('4', '2', 'TianJin', '400000');

insert into address (ADDRID, CUSTID, CITY, ZIPCODE)
values ('5', '2', 'ChongQing', '500000');

insert into address (ADDRID, CUSTID, CITY, ZIPCODE)
values ('6', '3', 'ChangSha', '410000');

Commit;
```

これら 3 つのテーブルを使用して、親ビジネス・オブジェクトと子ビジネス・オブジェクトとの間に単一カーディナリティー関係または複数カーディナリティー関係を作成します。

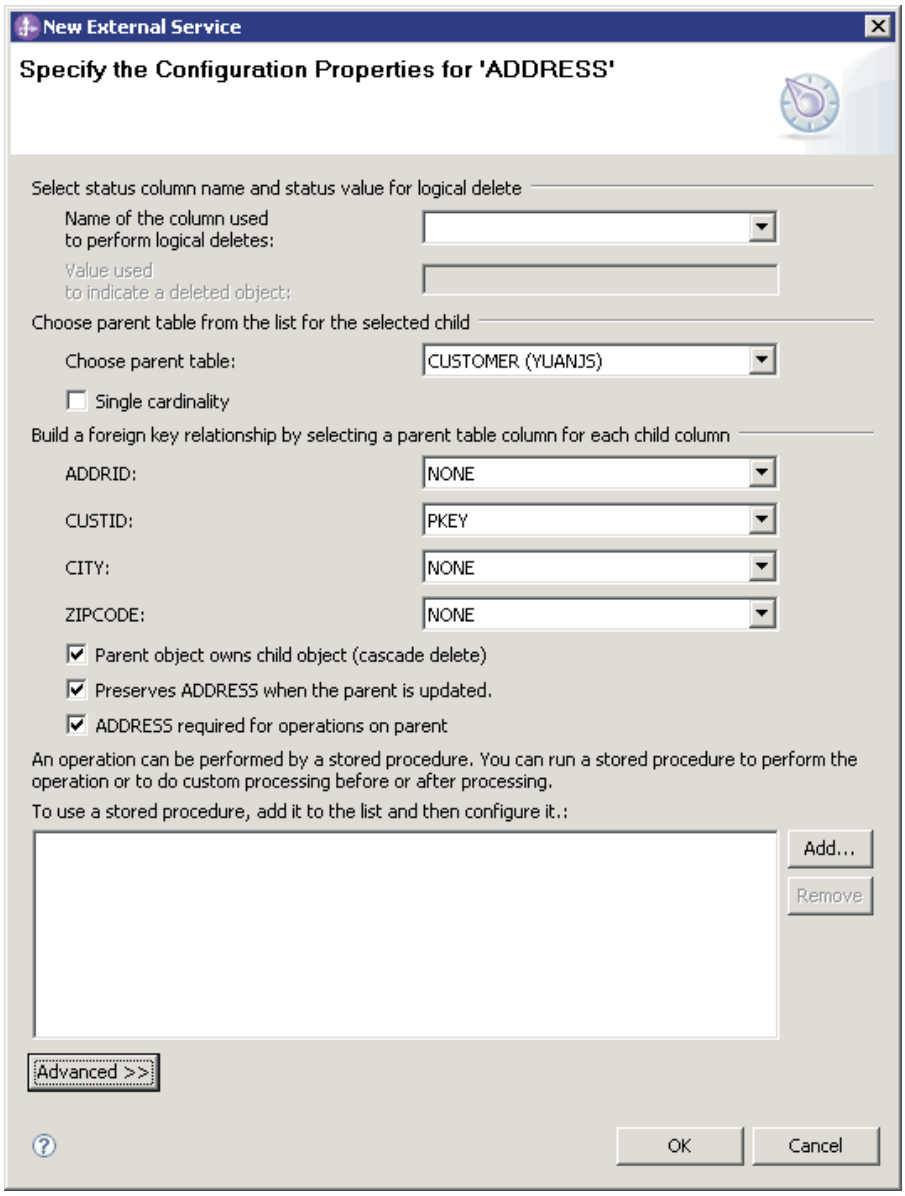
1. 以下のプロセスを使用して、親ビジネス・オブジェクト (CUSTOMER) と子ビジネス・オブジェクト (ADDRESS) との間に複数カーディナリティー関係を作成します。
   - 以下のプロセスを使用して、複数カーディナリティー関係を生成するために外部サービス・ディスカバリーを実行します。
     - 「新規外部サービス」ウィンドウで、「**ビジネス・オブジェクトの追加時に追加構成設定のプロンプトを出す**」チェック・ボックスを選択します。



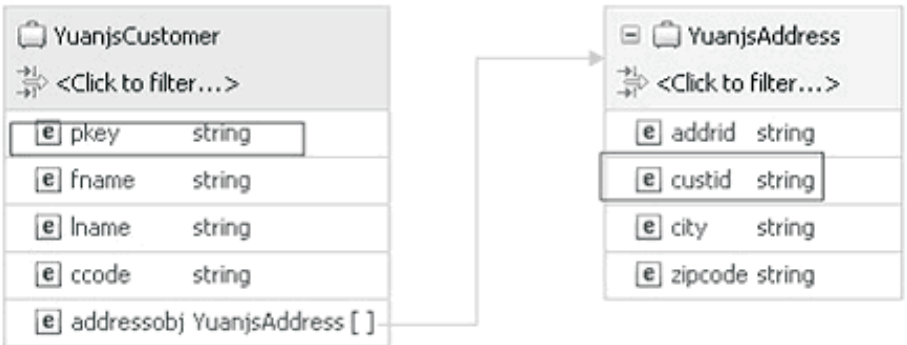
– 下の表のように、CUSTOMER テーブルを、親ビジネス・オブジェクトとして、「**選択されたオブジェクト**」領域に追加します。



– 「選択されたオブジェクト」リストに ADDRESS を追加すると、構成プロパティーのパネルが表示されます。構成プロパティーのパネルから、ADDRESS の親テーブルとして CUSTOMER を選択し、CUSTOMER テーブルから PKEY 列を選択し、ADDRESS テーブルから CUSTID 列を選択します。「**単一カーディナリティー**」チェック・ボックスをクリアします。



- 生成される成果物 - 例えば、下の図は、 YuanjsCustomer ビジネス・オブジェクトと YuanjsAddress ビジネス・オブジェクトの間の複数カーディナリティー関係を示します。



YuanjsCustomer ビジネス・オブジェクトの ccode 属性は、次のように xsd を定義します。

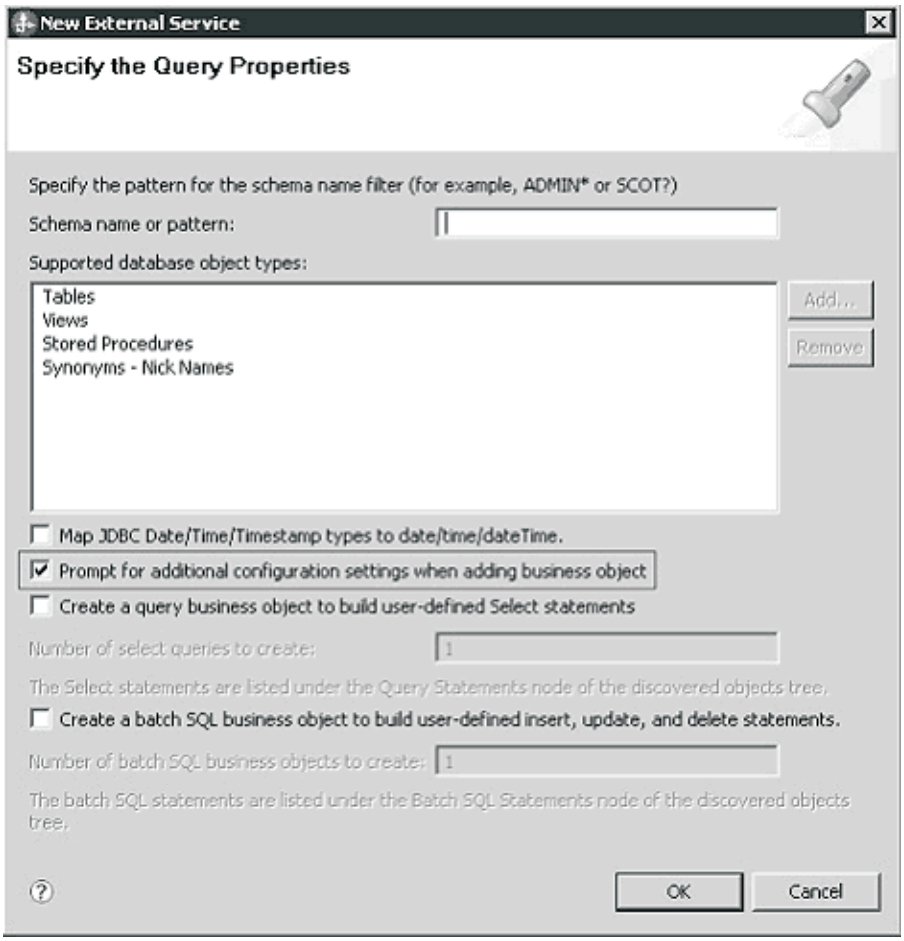
```
<jdbcasi:ColumnName>PKEY</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
```

YuanjsAddress ビジネス・オブジェクトの ccode 属性は、次のように xsd を定義します。

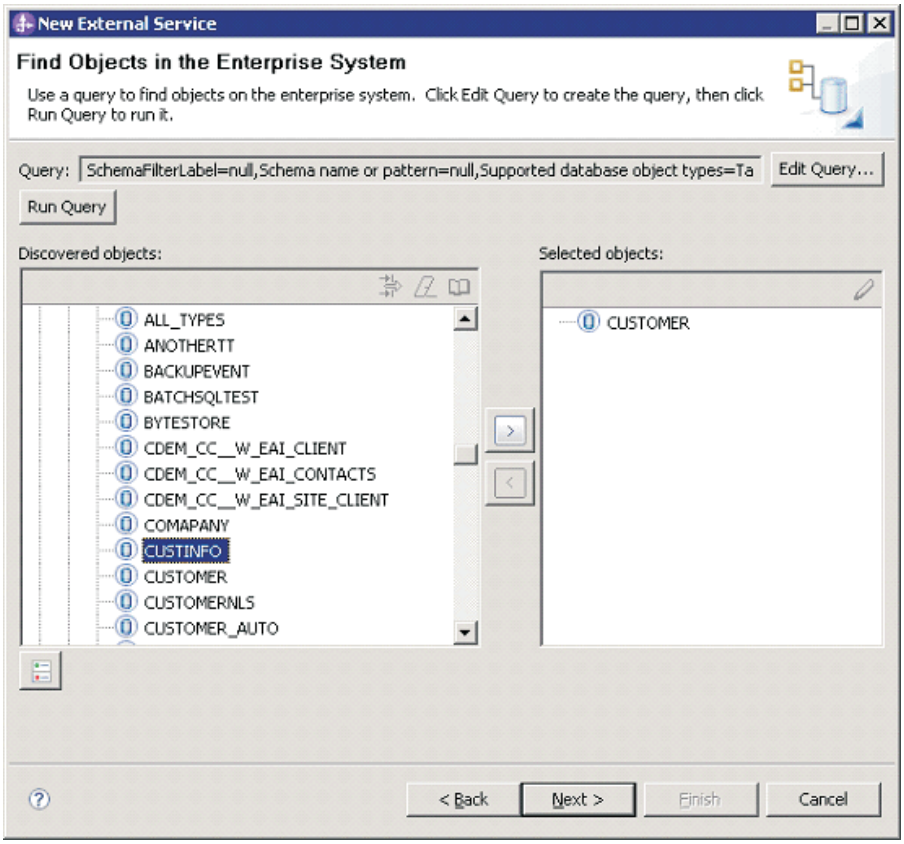
```
<jdbcasi:ColumnName>CUSTID</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
<jdbcasi:ForeignKey>pkey</jdbcasi:ForeignKey>
```

注: 単一カーディナリティー関係の場合、`<jdbcasi:ForeignKey>` アノテーション・メッセージは親ビジネス・オブジェクトに保管されます。

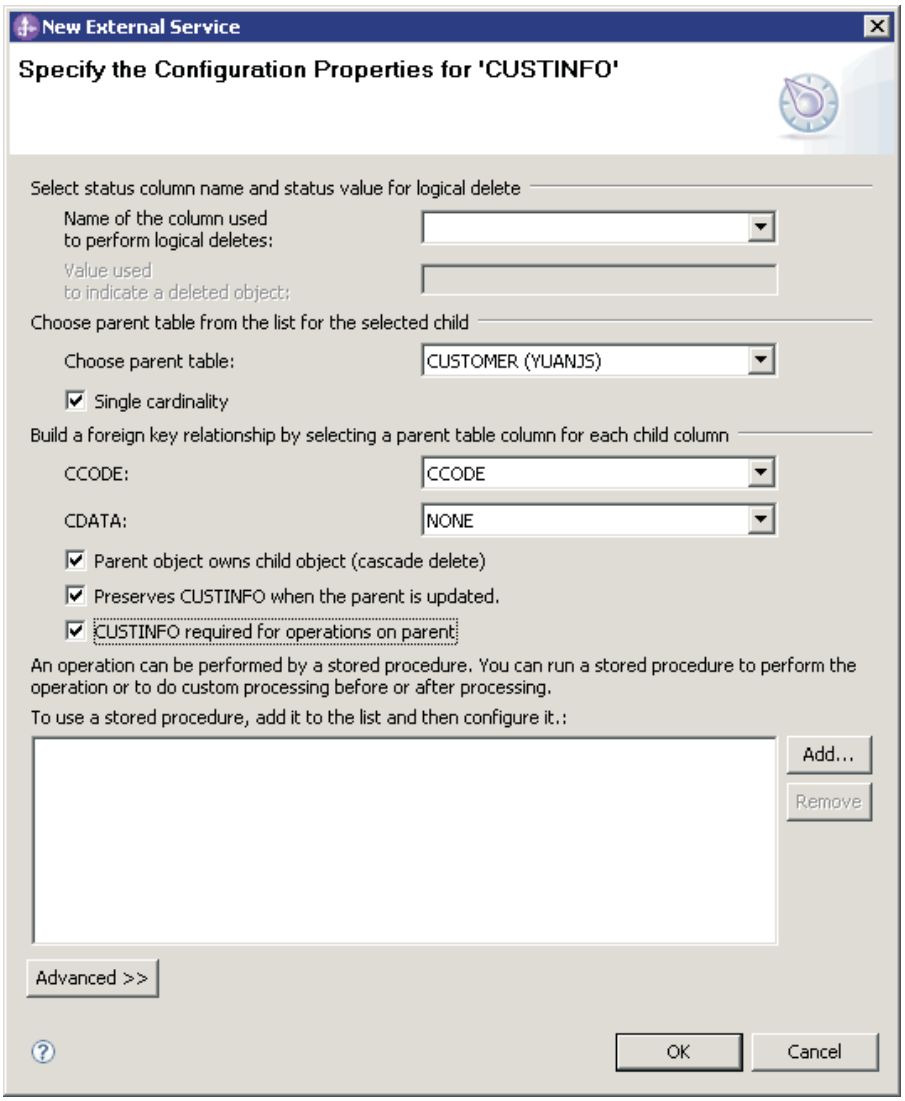
2. 以下のプロセスを使用して、親ビジネス・オブジェクト (CUSTOMER) と子ビジネス・オブジェクト (CUSTINFO) との間に単一カーディナリティー関係を作成します。
   - 以下のプロセスを使用して、単一カーディナリティー関係を生成するために外部サービス・ディスカバリーを実行します。
     – 「新規外部サービス」ウィンドウで、「**ビジネス・オブジェクトの追加時に追加構成設定のプロンプトを出す**」チェック・ボックスを選択します。



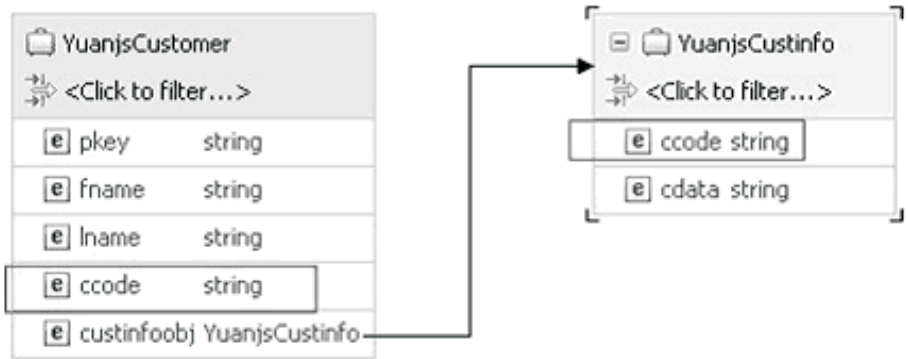
     – 下の表のように、CUSTOMER テーブルを、親ビジネス・オブジェクトとして、「**選択されたオブジェクト**」領域に追加します。



– 「選択されたオブジェクト」リストに CUSTINFO を追加すると、構成プロパティーのパネルが表示されます。構成プロパティーのパネルから、CUSTINFO の親テーブルとして CUSTOMER を選択し、CUSTOMER テーブルから CCODE 列を選択し、CUSTINFO テーブルから CCODE 列を選択します。次に、「**単一カーディナリティー**」チェック・ボックスを選択します。



- 生成される成果物 - 例えば、下の図は、 YuanjsCustomer ビジネス・オブジェクトと YuanjsCustinfo ビジネス・オブジェクトの間の単一カーディナリティー関係を示します。



YuanjsCustomer ビジネス・オブジェクトの ccode 属性は、次のように xsd を定義します。

```
<jdbcasi:ColumnName>CCODE</jdbcasi:ColumnName>CCODE
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
<jdbcasi:ForeignKey>custinfoobj/ccode</jdbcasi:ForeignKey>
```

YuanjsCustinfo ビジネス・オブジェクトの ccode 属性は、次のように xsd を定義します。

```
<jdbcasi:ColumnName>CCODE</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
```

注: 単一カーディナリティー関係の場合、<jdbcasi:ForeignKey> アノテーション・メッセージは親ビジネス・オブジェクトに保管されます。

親ビジネス・オブジェクトと子ビジネス・オブジェクトの間に単一または複数カーディナリティー関係を作成するときには、まず最初に、すべての親テーブルを「選択されたオブジェクト」リストに追加し、次に、親および子のテーブル列の基本キーと外部キーの関係を構成する必要があります。

カーディナリティー関係が「単一」の場合、外部キーのアノテーション情報は親ビジネス・オブジェクトに保管されます。カーディナリティー関係が「複数」の場合、外部キーのアノテーション情報は子ビジネス・オブジェクトに保管されます。

外部キーのアノテーション情報の場所が正しくない場合、親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は生成されません。

入力が大量の場合、バッチ操作のパフォーマンスが低下する

問題

入力が大量の場合、バッチ操作のパフォーマンスが低下します。

原因

Oracle などのデータベース・サーバーでは、以下のような場合にはバッチ操作中に多数のエラーが発生し、パフォーマンスが低下します。

- 入力のほとんどが無効である。
- **skipErrorsInBatch** プロパティーが True に設定されている。

外部サービス・ディスカバリー・ウィザードの使用時にエラーが表示される

問題

外部サービス・ディスカバリー・ウィザードの使用時にエラーが表示されます。

1. 「必要なファイルおよびライブラリーの位置指定」ページで、 JDBC ドライバー・ファイルを追加します。
2. 「**次へ**」をクリックして、「ディスカバリー・プロパティーの指定」パネルに移動します。
3. 「**戻る**」をクリックして、「必要なファイルおよびライブラリーの位置指定」ページに移動し、別の JDBC ドライバー・ファイルを追加します。

4. 「**次へ**」をクリックして、「ディスカバリー・プロパティーの指定」パネルに移動します。

IBM WebSphere Adapter for JDBC によって次のエラーが返されます。「EIS への接続が失敗しました。ドライバー com.ibm.db2.jcc.DB2Driver のロードは失敗しました。JDBC ドライバー・クラスがクラスパス内にあることを確認してください。」

原因

これは、クラス・ロードに関する Eclipse の制約です。

解決策

コネクター・プロジェクトをいったん閉じてから再度開いて、 JDBC ドライバー・ファイルを再度追加してください。

バッチ操作ドライバーに関する問題

問題 1

Oracle データベースの場合 - データベース操作が正常に実行された場合でも、ドライバーは、すべての個別ビジネス・オブジェクトに対して常に -2 を返します。これは、個別ビジネス・オブジェクトの実行が成功だったことを暗黙に示しますが、各ビジネス・オブジェクトごとの影響を受けた行数は不明です。BatchUpdate 操作および BatchDelete 操作中に、入力ビジネス・オブジェクトの対応するレコードに一致する既存レコードがデータベース内にない場合、またはデータベース内の複数のレコードが一致する場合、アダプターは正しいエラー・メッセージや例外を生成しません。

原因

これは、 Oracle データベース・ドライバーの制約です。

問題 2

MS SQL Server データベースの場合 - 入力レコードが複数ある操作が 1 バッチ・サイズ内で失敗した場合、アダプターは、失敗した最初のレコードについてのみ正しいエラー・メッセージを戻し、失敗した他のレコードに関しては不明のエラー・メッセージを戻すことがあります。

原因

これは、 MS SQL Server データベース・ドライバーの制約です。

EMD ウィザードの「追加」および「除去」ボタンが正しく機能しない

問題

EMD が同じ JDBC RAR ファイルを使用して連続して実行され、インポートされた JDBC ドライバー・ライブラリーが除去対象として選択されると、選択されていないライブラリーも含めてすべてのライブラリーが除去されます。例えば、DB2 ドライバー・ライブラリーは、ドライバー JAR ファイルとライセンス JAR ファイルで

構成されています。2 回目の実行時に、いずれかの JAR ファイルを除去しようとすると、もう一方の JAR ファイルも一緒に除去されてしまいます。

原因

これは、 Rational® Application Developer for WebSphere Software の問題により発生します。

Informix データベースに接続できない

問題

EMD ウィザードの「**サービス生成およびデプロイメント・プロパティーの指定**」ページで、「**データベース接続情報**」フィールドから「**ローカル・データベース接続情報の指定**」オプションを選択し、次に、「**データベース・システム接続情報**」フィールドに Informix データベース接続詳細を指定すると、アプリケーションは、デプロイされて実行された後、以下の例外を生成します。

com.ibm.db2.jcc.am.SqlSyntaxErrorException: [jcc][10165][10051][4.11.69] Invalid database URL syntax:

jdbc:informix-sqli://localhost:9090/sysmaster:INFORMIXSERVER=ol_ids_1150. ERRORCODE=-4461, SQLSTATE=42815.

原因

現在は、アダプターはデータ・ソースを経由してのみ Informix サーバーに接続します。

解決策

一般的なローカル接続プロパティーを指定する代わりに、Informix サーバー用に対してデータ・ソース接続を使用します。

EMD ウィザードの「**サービス生成およびデプロイメント・プロパティーの指定**」ページで、「**データベース接続情報**」フィールドから「**事前定義済み接続プール・データ・ソースの指定**」オプションを選択し、次に、既に構成済みの Informix データ・ソース名を指定します。

ヌル・データ - よくある質問

よくある質問 (FAQ) は、IBM WebSphere Adapter for JDBC が、ヌル、ヌル・ストリング (空ストリング)、ブランク・ストリング、ヌル値、デフォルト値をどのように処理するのかについて、また、一部のフィールドをどのようにスキップするのかについての照会に対処します。

質問

アダプターは、ヌル、ヌル・ストリング (空ストリング)、ブランク・ストリング、ヌル値、デフォルト値をどのように処理し、一部のフィールドをスキップするのですか? アダプターのユニバーサル・テスト環境 (Universal Test Environment (UTE)) で、`value`、`unset`、`null`、および `default` は、どう違うのですか? テーブル内の一部のフィールドにアダプターはどのように `Null` を設定するのですか? テーブル内

の一部のフィールドにアダプターはどのように空ストリングを設定するのですか? アダプターはどのようにしてテーブル内のフィールドでの操作をスキップするのですか?

上記の質問に対処するため、アダプターは各操作に関して以下のアクションを実行します。

1. **Create 操作** - UTE では、図 1 に示すようなさまざまな値を入力できます。

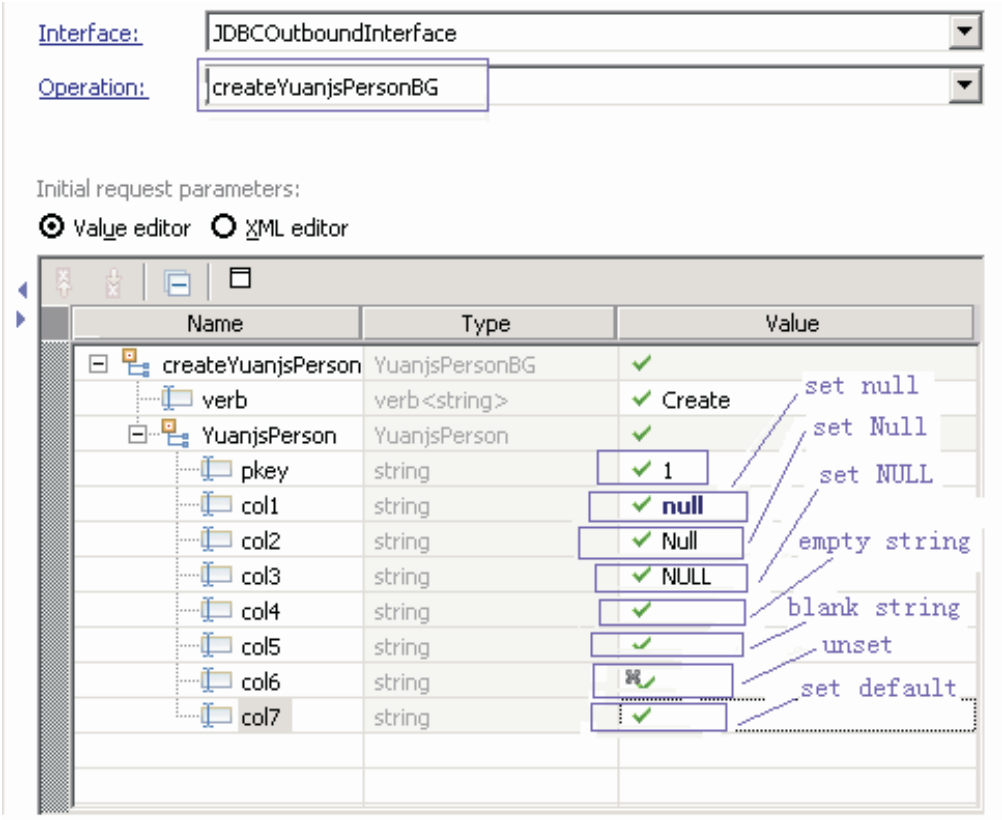


図 37.

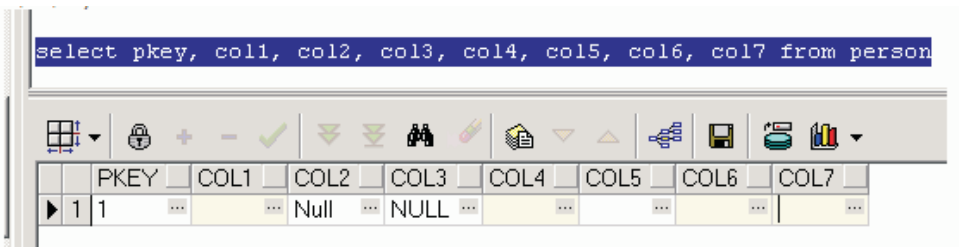
図 1 では、異なる属性に対して異なる値が設定されています。これを実行すると、以下のビジネス・オブジェクト・データが生成されます。

```
[7/16/10 15:09:18:390 CST] 00000047 JDBCRA001 3 JDBCInteraction execute
Execute Interaction for the function Create
[7/16/10 15:09:18:390 CST] 00000047 JDBCRA001 3 JDBCInteraction execute
<?xml version="1.0" encoding="UTF-8"?>
<p:YuanjsPerson
xsi:type="p:YuanjsPerson"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjsperson">
<pkey>1</pkey>
<col1 xsi:nil="true"/>
<col2>Null</col2>
<col3>NULL</col3>
<col4></col4>
<col5></col5>
<col7></col7>
</p:YuanjsPerson>
```

ここで unset は、エレメントがビジネス・オブジェクト・データから消失するか、削除されることを暗黙に示し、値 set default は空ストリングを暗黙に示しています。

以下の SQL ステートメントが生成されます。

```
[7/16/10 15:09:18:750 CST] 00000047 JDBCRA001 1
com.ibm.j2ca.dbadapter.core.runtime.DBSQLBuilder
composeCreateSQL Create SQL is :
INSERT INTO YUANJS.PERSON ( PKEY, COL2, COL3, COL4, COL5, COL7 )
VALUES ( ? , ? , ? , ? , ? , ? )
```



- set null の場合、SQL ステートメントは insert into person(pkey,col1) values(1,null); です
- set Null の場合、SQL ステートメントは insert into person(pkey,col2) values(1,'Null'); です。
- set NULL の場合、SQL ステートメントは insert into person(pkey,col3) values(1,'NULL'); です。
- empty string の場合、SQL ステートメントは insert into person(pkey,col4) values(1,''); です。
- 3 個のブランクを伴う blank string の場合、SQL ステートメントは insert into person(pkey,col5) values(1,' '); です。
- unset の場合、SQL ステートメントには unset 属性の対応する列が含まれず、この列に対する操作はありません。
- set default は空ストリングと同じです。

2. **Update 操作** - UTE で、図 1 に示すようなさまざまな値を入力することができ、Create 操作の場合と同じビジネス・オブジェクトを生成できます。

   ここで、unset は、エレメントがビジネス・オブジェクト・データから消失するか、削除されることを暗黙に示し、 set default は空ストリングを暗黙に示しています。

   以下の SQL ステートメントが生成されます。

```
[7/16/10 15:43:29:687 CST] 00000048 JDBCRA001
1 com.ibm.j2ca.dbadapter.core.runtime.
DBSQLBuilder composeUpdateSQL Update SQL is :
UPDATE YUANJS.PERSON SET COL1 = ? , COL2 = ? , COL3 = ? , COL4 = ? , COL5 = ? , COL7 = ?
WHERE (PKEY = ?)
```

   ここで、

   - set null の場合、SQL ステートメントは update person set col1= null where (pkey='1') ; です。

- set Null の場合、SQL ステートメントは update person set col2= 'Null' where (pkey='1') ; です。
- set NULL の場合、SQL ステートメントは update person set col3= 'NULL' where (pkey='1') ; です。
- empty string の場合、SQL ステートメントは update person set col4= '' where (pkey='1') ; です。
- 3 個のブランクを伴う blank string の場合、SQL ステートメントは update person set col5= ' ' where (pkey='1') ; です。
- unset の場合、SQL ステートメントには unset 属性の対応する列が含まれず、この列に対する操作はありません。
- set default は空ストリングと同じです。

3. **Delete 操作** - UTE で、図 1 に示すようなさまざまな値を入力することができ、Create 操作の場合と同じビジネス・オブジェクトを生成できます。

   ここで、unset は、エレメントがビジネス・オブジェクト・データから消失するか、削除されることを暗黙に示し、値 set default は空ストリングを暗黙に示しています。

   以下の SQL ステートメントが生成されます。

   ```
   [7/16/10 15:57:33:250 CST] 00000049 JDBCRA001 1
   com.ibm.j2ca.dbadapter.core.runtime.DBSQLBuilder
   composeDeleteSQL Delete SQL is : DELETE FROM YUANJS.PERSON WHERE (PKEY = ?)
   ```

   Delete 操作の場合、基本キーを持つ属性のみが WHERE 節ステートメントに追加され、他の属性は無視されます。上記の SQL ステートメントでは、属性 (PKEY) が WHERE 節ステートメントに含まれる基準です。属性のいずれかと PKEY 属性が複合キーであると見なされた場合、次の SQL ステートメントが生成されます。

   - set null の場合、SQL ステートメントは delete from person where (pkey='1') and (col1= is null); です。
   - set Null の場合、SQL ステートメントは delete from person where (pkey='1') and (col2 = 'Null'); です。
   - set NULL の場合、SQL ステートメントは delete from person where (pkey='1') and (col3 = 'NULL'); です。
   - empty string の場合、SQL ステートメントは delete from person where (pkey='1') and (col4 = ''); です。
   - 3 個のブランクを伴う blank string の場合、SQL ステートメントは delete from person where (pkey='1') and (col5 = ' '); です。
   - unset の場合、WHERE 節ステートメントには unset 属性の対応する列が含まれず、この列に対する操作はありません。SQL ステートメントは delete from person where (pkey='1'); です。
   - set default は空ストリングと同じです。

4. **Retrieve 操作** - UTE で、図 1 に示すようなさまざまな値を入力することができ、Create 操作の場合と同じビジネス・オブジェクトを生成できます。

ここで、unset は、エレメントがビジネス・オブジェクト・データから消失するか、削除されることを暗黙に示し、値 set default は空ストリングを暗黙に示しています。

以下の SQL ステートメントが生成されます。

```
[7/16/10 18:12:26:703 CST] 00000048 JDBCRA001 1
com.ibm.j2ca.dbadapter.core.runtime.DBSQLBuilder
composeRetrieveSQL Retrieve SQL is : SELECT PKEY, COL1, COL2, COL3, COL4,
COL5, COL6, COL7
FROM YUANJS.PERSON WHERE (PKEY = ?)
```

Retrieve 操作の場合、基本キーを持つ属性のみが WHERE 節ステートメントに追加され、他の属性は無視されます。上記の SQL ステートメントでは、属性 (PKEY) が WHERE 節ステートメントに含まれる基準です。属性のいずれかと PKEY 属性が複合キーであると見なされた場合、次の SQL ステートメントが生成されます。

- set null の場合、SQL ステートメントは select ... from person where (pkey='1') and (col1 is null); です。
- set Null の場合、SQL ステートメントは select ... from person where (pkey='1') and (col2 = 'Null'); です。
- set NULL の場合、SQL ステートメントは select ... from person where (pkey='1') and (col3 = 'NULL'); です。
- empty string の場合、SQL ステートメントは select ... from person where (pkey='1') and (col4 = ''); です。
- 3 個のブランクを伴う blank string の場合、SQL ステートメントは select ... from person where (pkey='1') and (col5= ' '); です。
- unset の場合、WHERE 節ステートメントには unset 属性の対応する列が含まれず、この列に対する操作はありません。SQL ステートメントは select ... from person where (pkey='1'); です。
- set default は空ストリングと同じです。

5. **RetrieveAll 操作** - UTE で、図 1 に示すようなさまざまな値を入力することができ、Create 操作の場合と同じビジネス・オブジェクトを生成できます。

   ここで、unset は、エレメントがビジネス・オブジェクト・データから消失するか、削除されることを暗黙に示し、 set default は空ストリングを暗黙に示しています。

   以下の SQL ステートメントが生成されます。

   ```
   [7/16/10 18:16:51:078 CST] 00000048 JDBCRA001 1
   com.ibm.j2ca.dbadapter.core.runtime.DBSQLBuilder buildSQLForExistsNRetrieveAll
   RetrieveAll SQL is :
   SELECT PKEY, COL1, COL2, COL3, COL4, COL5, COL6, COL7
   FROM YUANJS.PERSON WHERE (PKEY = ?)
   AND (COL1 IS NULL) AND (COL2 = ?) AND (COL3 = ?) AND (COL4 = ?)
   AND (COL5 = ?) AND (COL7 = ?)
   ```

   RetrieveAll 操作の場合、どの属性も WHERE 節ステートメントに追加されます。

   - set null の場合、SQL ステートメントは select ... from person where ... and (col1 is null); です。

- set Null の場合、SQL ステートメントは select ... from person where ... and (col2 = 'Null'); です。
- set NULL の場合、SQL ステートメントは select ... from person where ... and (col3 = 'NULL'); です。
- empty string の場合、SQL ステートメントは select ... from person where ... and (col4 = ''); です。
- 3 個のブランクを伴う blank string の場合、SQL ステートメントは select ... from person where ... and (col5 = ' '); です。
- unset の場合、WHERE 節ステートメントには unset 属性の対応する列が含まれません。したがって、基準に含まれる列はありません。
- set default は空ストリングと同じです。

注:

- Retrieve 操作は、基本キーに従って 1 つの行のみを戻し、複数の行を戻しません。トップレベルのビジネス・オブジェクトの基本キーのみが、 SQL ステートメントの WHERE 節を組み立てるのに使用されます。
- RetrieveAll 操作は、属性値のうちの一部の値に従って、複数の行を戻します。Retrieve 操作および RetrieveAll 操作について詳しくは、http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/topic/com.ibm.wsadapters.620.jca.jdbc.doc/doc/cjdb_retrieveoperate.html および :http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/topic/com.ibm.wsadapters.620.jca.jdbc.doc/doc/cjdb_retrievealloperate.html を参照してください。

6. **Exists 操作** - UTE で、図 1 に示すようなさまざまな値を入力することができ、Create 操作の場合と同じビジネス・オブジェクトを生成できます。

   ここで、unset は、エレメントがビジネス・オブジェクト・データから消失するか、削除されることを暗黙に示し、値 set default は空ストリングを暗黙に示しています。

   以下の SQL ステートメントが生成されます。

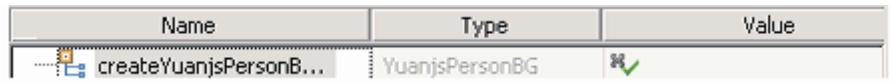
   ```
   [7/16/10 18:20:28:812 CST] 00000048 JDBCRA001 1
   com.ibm.j2ca.dbadapter.core.runtime.DBSQLBuilder buildSQLForExistsNRetrieveAll
   RetrieveAll SQL is :
   SELECT count(*) as numRecords FROM YUANJS.PERSON
   WHERE (PKEY = ?)
   AND (COL1 IS NULL) AND (COL2 = ?) AND (COL3 = ?) AND (COL4 = ?)
   AND (COL5 = ?) AND (COL7 = ?)
   ```

   Exists 操作の場合、どの属性も WHERE 節ステートメントに追加されます。これは RetrieveAll 操作に似ています。

7. **重要ポイント**:

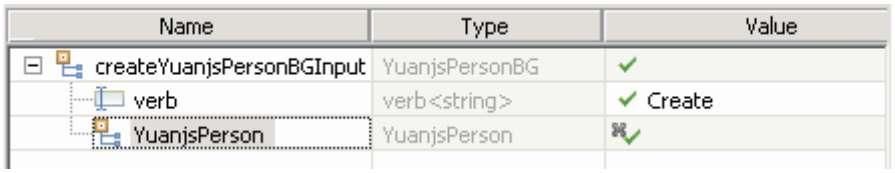
   各種の操作を実行するときに、以下のような正しくないビジネス・オブジェクト・データを避けるように注意してください。

   - **ビジネス・オブジェクト内にビジネス・グラフがある場合** Create、Update、Delete、および Retrieve 各操作では、以下のビジネス・オブジェクト・データは無効です。
     a. ビジネス・グラフが空:



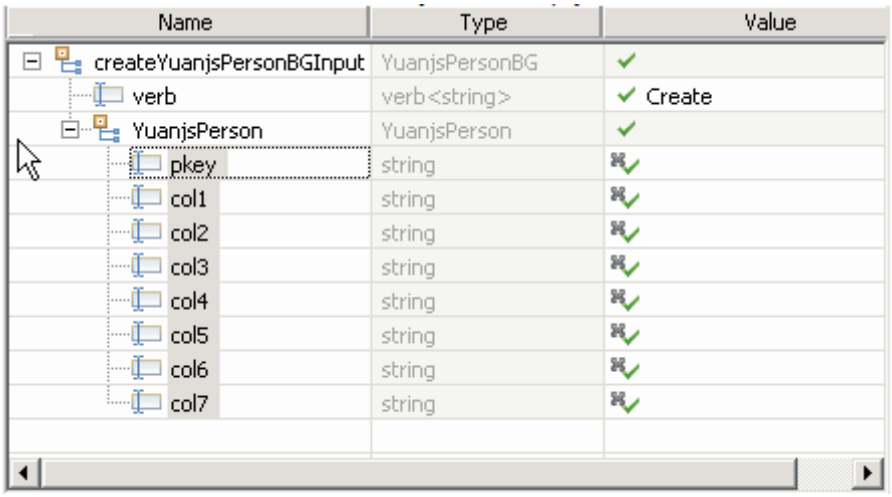

```
<?xml version="1.0" encoding="UTF-8" ?>
```

b. ビジネス・オブジェクトが空:




```
<?xml version="1.0" encoding="UTF-8"?>
<createYuanjsPersonBGInput
xmlns:ns2="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjspersonbg"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:YuanjsPersonBG">
<verb>Create<verb>
</createYuanjsPersonBGInput>
```

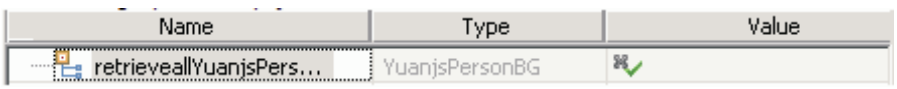
c. ビジネス・オブジェクトの下のすべての属性が空:




```
<?xml version="1.0" encoding="UTF-8"?>
<createYuanjsPersonBGInput
xmlns:ns2="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjspersonbg"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:YuanjsPersonBG">
<verb>Create</Verb>
<YuanjsPerson/>
</createYuanjsPersonBGInput>
```

RetrieveAll 操作および Exists 操作では、以下のビジネス・オブジェクト・データは無効です。

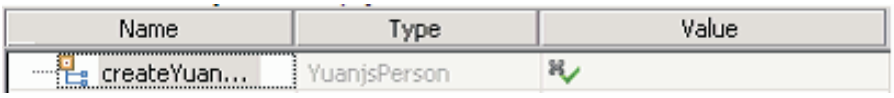
ビジネス・グラフが空:



- **ビジネス・オブジェクトにビジネス・グラフがない場合**

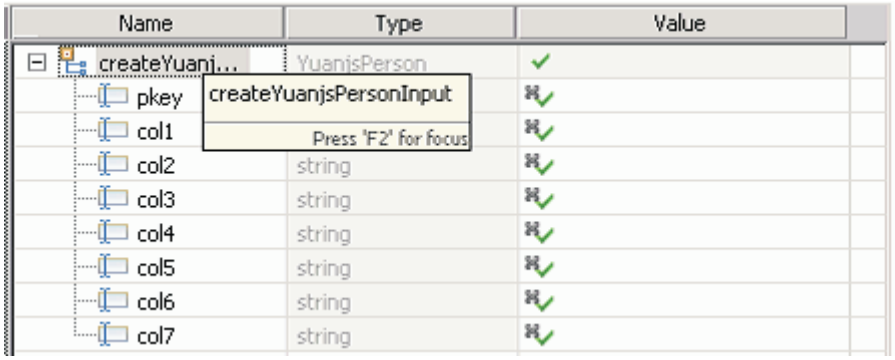
Create、 Update、Delete、および Retrieve の各操作では、以下のビジネス・オブジェクト・データは無効です。

a. **ビジネス・オブジェクトが空**:




```
<?xml version="1.0" encoding="UTF-8"?>
```

b. **ビジネス・オブジェクトの下のすべての属性が空**:




```
<?xml version="1.0" encoding="UTF-8"?>
<createYuanjsPersonInput
xmlns:ns2="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjsperson"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:YuanjsPerson"/>
```

RetrieveAll 操作および Exists 操作では、以下のビジネス・オブジェクト・データは無効です。

ビジネス・オブジェクトが空:


```
<?xml version="1.0" encoding="UTF-8"?>
```

- **Java で、設定解除、ヌルの設定、空ストリングの設定、およびブランク・ストリングの設定を、BO API を使用して行うにはどうすればいいでしょうか?**

  Java コードでビジネス・オブジェクトに値を割り当てるのに役立つ例を以下に示します。

  a. **Java で、ビジネス・オブジェクト属性を設定解除するか、または、ビジネス・オブジェクト属性をヌルに設定する**

  ビジネス・オブジェクト・データが次のようであるとします。

```
DataObject customer = bg.createDataObject("YuanjsCustomer");
customer.set("pkey", "0001");
customer.unset("fname");
customer.set("lname", null);
```

  以下のビジネス・オブジェクト・データが生成されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<p:YuanjsCustomer xmlns:p="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc
/yuanjscustomer]"
xmlns:ns0="[http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjscustomer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="p:YuanjsCustomer">
<pkey>0001</pkey>
<lname xsi:nil="true"/>
</p:YuanjsCustomer>
```

b. **Java でビジネス・オブジェクトをヌルに設定する**

ビジネス・オブジェクト・データが次のようであるとします。

```
DataObject bg = (DataObject) factory.create(
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjscustomebg]",
"YuanjsCustomerBG");
DataObject customer = bg.createDataObject("YuanjsCustomer");
```

以下のビジネス・オブジェクト・データが生成されます。

```
xmlns:ns0="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjscustomer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="p:YuanjsCustomer"/>
```

c. **Java で、ビジネス・オブジェクト属性を空ストリング、ブランク・ストリング、ヌル・ストリングに設定する**

ビジネス・オブジェクト・データが次のようであるとします。

```
DataObject customer = bg.createDataObject("YuanjsCustomer");
customer.set("pkey", "0001");
customer.setString("fname","");
customer.setString("lname", null);
customer.setString("ccode", " ");
```

以下のビジネス・オブジェクト・データが生成されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<p:YuanjsCustomer
xmlns:p="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjscustomer"
xmlns:ns0="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/yuanjscustomer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="p:YuanjsCustomer">
<pkey>0001</pkey>
<fname></fname>
<lname xsi:nil="true"/>
<ccode> </ccode>
</p:YuanjsCustomer>
```

セルフ・ヘルプ・リソース

IBM ソフトウェア・サポートのリソースは、最新のサポート情報やテクニカル文書を入手したり、サポート・ツールやフィックスをダウンロードしたり、WebSphere Adapters の問題を回避したりするために使用することができます。また、セルフ・ヘルプ・リソースは、アダプターに関連する問題を診断するのに役立ち、IBM ソフトウェア・サポートへの連絡方法についての情報を提供します。

サポート Web サイト

WebSphere Adapters ソフトウェアのサポート Web サイト (http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Family) では、WebSphere Adapters の学習、使用、およびトラブルシューティングに役立つ多数のリソースへのリンクを提供しています。以下のリソースがあります。

- フラッシュ (製品に関する警告)
- 製品のインフォメーション・センター、マニュアル、IBM Redbooks®、およびホワイト・ペーパーなどの技術情報。

- 研修関連
- 技術情報

推奨フィックス

適用する必要のある推奨フィックスのリストは、http://www.ibm.com/support/docview.wss?fdoc=aimadp&rs=695&uid=swg27010397 にあります。

技術情報

技術情報は、WebSphere Adapter for JDBC に関する最新の資料を提供します。以下のトピックがあります。
- 問題とそれに対する現在使用可能な解決策
- よくある質問に対する答え
- アダプターのインストール、構成、使用法、トラブルシューティングに関する手引きとなる情報
- *IBM ソフトウェア・サポート・ハンドブック*

WebSphere Adapters の技術情報のリストについては、以下のアドレスにアクセスしてください。

http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm

IBM Support Assistant のプラグイン

WebSphere Adapter for JDBC では、IBM Support Assistant のプラグインを提供します。これは、無料の保守容易性ローカル・ソフトウェア・ワークベンチです。プラグインは、動的トレース・フィーチャーをサポートします。 IBM Support Assistant のインストールおよび使用については、以下のアドレスにアクセスしてください。

http://www.ibm.com/software/support/isa/

参照情報

ビジネス・オブジェクト、アダプター・プロパティー (エンタープライズ・サービス・ディスカバリー・プロパティー、リソース・アダプター・プロパティー、管理 (J2C) 接続ファクトリー・プロパティー、活動化仕様プロパティー、および対話仕様プロパティー)、メッセージ、および関連製品情報に関する詳細情報は参照用に提供されます。

ビジネス・オブジェクト情報

ビジネス・オブジェクトは、アダプターによるビジネス・オブジェクトの処理方法、およびビジネス・オブジェクトに対して実行される操作に関するアプリケーション固有情報 (メタデータ) を格納する構造です。ビジネス・オブジェクトの名前は、アダプターの命名規則に従って、外部サービス・ウィザードが生成します。

ビジネス・オブジェクト属性

ビジネス・オブジェクト属性は、ビジネス・オブジェクトの内容を定義するものであり、データベース・オブジェクトの列リストから作成されます。各属性は、名

前、型、カーディナリティーなどのプロパティーを持ちます。外部サービス・ウィザードで、列名に属性名が設定されます。アダプターでは、属性のカーディナリティー、型、およびアプリケーション固有情報が追加されます。

ビジネス・オブジェクトは、属性で指定されるデータのコンテナーです。データベースのデータの構造はビジネス・オブジェクトによって定義されますが、データベースのデータはビジネス・オブジェクト属性内にあります。

表 22 に、ビジネス・オブジェクト属性のプロパティーをリストし、それらの解釈および設定値について説明します。

表 22. 属性プロパティー

プロパティー	**解釈と設定値**
Cardinality	ビジネス・オブジェクトのカーディナリティーを示す整数。1 つの子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す各ビジネス・オブジェクト属性は、カーディナリティーの値が単一 (1) または複数 (制限のない整数) になります。 単一カーディナリティー関係および複数カーディナリティー関係の両方で、親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は、その関係を保管するビジネス・オブジェクトのキー属性に含まれるアプリケーション固有情報によって記述されます。
Foreign Key	カーディナリティーが *n* の子ビジネス・オブジェクトの配列が検索されると、SELECT ステートメントの WHERE 文節で外部キーが使用されます。 RetrieveAll 操作は、キーおよび外部キーの使用を指定変更します。 **注:** アダプターでは、子ビジネス・オブジェクトを表す属性を外部キーとして指定することについては、サポートしていません。
Name	このプロパティーは、属性が単純属性の場合は、属性の固有の名前。属性が子ビジネス・オブジェクトの場合は、ビジネス・オブジェクトの名前を表します。
MinOccurs MaxOccurs	列が基本キーではなく、ヌル可能でない場合、MinOccurs および MaxOccurs 属性は必須であり、値は 1 に設定されます。
Primary Key	この属性が基本キーかどうかを示します。各ビジネス・オブジェクトで少なくとも 1 つの単純属性を基本キーとして指定する必要があります。 単純属性の基本キー・プロパティーを `true` に設定すると、アダプターは、ビジネス・オブジェクトの処理中に生成する SELECT および SQL UPDATE の各ステートメントの WHERE 文節にその属性を追加します。RetrieveAll 操作は、基本キーおよび外部キーの使用を指定変更します。 **注:** アダプターでは、子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す属性を基本キー属性として指定することについてはサポートしていません。

表 22. 属性プロパティー (続き)

プロパティー	解釈と設定値
必須	属性が値を含む必要があるかどうかを指定します。カーディナリティーが単一 (1) のコンテナーに対して、このプロパティーが `true` に設定されている場合、アダプターでは、その親ビジネス・オブジェクトが、この属性に対応する子ビジネス・オブジェクトを含んでいる必要があります。Create、Update、および Delete 操作でアダプターに渡されるビジネス・オブジェクトは、子ビジネス・オブジェクトも含んでいなければなりません。単純属性のカーディナリティーは単一 (1) で、コンテナー属性のカーディナリティーは複数 (n) です。必須属性に対してビジネス・オブジェクトが有効な値またはデフォルト値を持っていない場合、Create 操作は失敗します。また、データベースから取り出すときにこのオブジェクトに対する使用可能なデータがない場合も操作は失敗します。
Type	単純属性の場合、このプロパティーは属性の型 (Integer、String、Date、Timestamp、Boolean、Double、Float など) を指定します。サポートされる単純属性の型と、それらがマップされるデータベース・オブジェクトの JDBC タイプを表 23 に示します。 子ビジネス・オブジェクトを指定する属性の場合、このプロパティーはビジネス・オブジェクトの名前を指定します。

JDBC メタデータとして戻される各データベース・オブジェクトのタイプは、表 23 のリストにあるようにビジネス・オブジェクト属性タイプにマップされます。リストされている JDBC タイプのみがアダプターでサポートされます。リストされていないタイプの列は、ビジネス・オブジェクトに追加されません。その場合は、問題を説明する通知メッセージが生成されます (例: テーブル *yyyy* の列名 *xxxx* のタイプはサポートされていません。ビジネス・オブジェクトには追加されません)。

表 23. JDBC メタデータ列タイプとビジネス・オブジェクト属性タイプ

JDBC メタデータの列タイプ	ビジネス・オブジェクト属性のタイプ
BIT	Boolean
CHAR LONGVARCHAR VARCHAR	String
NUMERIC	Decimal Int
INTEGER SMALLINT TINYINT	Int
BIGINT	Long Int
TIME	String Time
TIMESTAMP	String DateTime

表 23. JDBC メタデータ列タイプとビジネス・オブジェクト属性タイプ (続き)

JDBC メタデータの列タイプ	ビジネス・オブジェクト属性のタイプ
DATE	String Date Datetime
DECIMAL	Decimal
DOUBLE FLOAT	Double
REAL	Float
BLOB	hexBinary
CLOB	String
BINARY VARBINARY LONGBINARY	hexBinary
NCHAR NVARCHAR NTEXT	String
TEXT	String
RAW	hexBinary
MONEY SMALLMONEY	Decimal
STRUCT または ARRAY	アダプターは、これらのデータ型をテーブル・ビジネス・オブジェクトまたは照会ビジネス・オブジェクトの子ビジネス・オブジェクトとして処理します。 **注:** アダプターが複合型をサポートするのは、Oracle テーブル・ビジネス・オブジェクトおよび照会ビジネス・オブジェクトの場合のみです。テーブルに配列、構造体、ネストされた構造体、またはテーブルなどの複合データ型が含まれている場合は、型名およびサブ属性の詳細も自動的にディスカバーされて表示されます。 **注:** アダプターは空の複合列をヌルとして処理します。それがヌルに設定されているかどうかは関係ありません。

表 23. JDBC メタデータ列タイプとビジネス・オブジェクト属性タイプ (続き)

JDBC メタデータの列タイプ	ビジネス・オブジェクト属性のタイプ

XML	String
	
	Oracle の場合、ランタイム環境で XML データ型を扱うには、さらに以下のライブラリーが必要です。
	xdb.jar
	xmlparserv2.jar
	
	また、JNDI データ・ソースを使用してデータベースに接続している場合は、データ・ソースの作成時に必ず xdb.jar と xmlparserv2.jar をクラスパスに追加してください。
	注: XML 型は、テーブル・ビジネス・オブジェクト、照会ビジネス・オブジェクト、ストアード・プロシージャー・ビジネス・オブジェクト、およびバッチ照会ビジネス・オブジェクト内でサポートされます。
	
	XML コンテンツの例:
	`<customer>`
	`<fname>John</fname>`
	`<lname>Smith</lname>`
	`</customer>`

属性に関するアプリケーション固有情報

ビジネス・オブジェクト属性のアプリケーション固有情報 (ASI) は、属性が単純属性であるか、子ビジネス・オブジェクト (または子ビジネス・オブジェクトの配列) を表す属性であるかによって異なります。子を表す属性のアプリケーション固有情報は、親子関係が子に格納されるか親に格納されるかによっても異なります。

単純属性のアプリケーション固有情報

単純属性では、アプリケーション固有情報の形式は、いくつかのパラメーターとその値で構成されています。単純属性に必要な唯一のパラメーターは列名です。単純属性のアプリケーション固有情報については、表 24で説明します。

表 24. 単純属性のアプリケーション固有情報

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法

BLOB	Boolean	この属性に対応するデータベース列が BLOB データ型であるかどうかを示します。BLOB データの表示中、アダプターはバイト数を 16 進値で表示します。属性の型は hexBinary です。
		
		`True` に設定されている場合、列のデータ型は BLOB です。
			なし	True、 False	`<jdbcasi:`
					`BLOB>`
					`true<`
					`/jdbcasi:BLOB>`

ByteArray	Boolean	列がバイナリー・データ型であるかどうかを示します。 `True` の場合、アダプターはデータベースでバイナリー・データを読み取りおよび書き込みし、そのデータをストリングとしてアプリケーション・サーバーに送信します。アダプターは、ビジネス・オブジェクトにバイナリー・データを設定します。属性の型は hexBinary です。	`False`	True、 False	`<jdbcasi:`
					`ByteArray>`
					`false`
					`</jdbcasi:`
					`ByteArray>`

表 24. 単純属性のアプリケーション固有情報 (続き)

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法
ChildBOType	String	属性が複合データ型の場合は、このアプリケーション固有情報を使用して実際の型を指定します。 • Struct • Array • ResultSet	なし	Struct、 Array、 ResultSet	<jdbcasi: ChildBOType> Array </jdbcasi: ChildBOType>
ChildBOTypeName	String	ChildBOType アプリケーション固有情報の値が Struct または Array の場合、このパラメーターはユーザー定義タイプの名前を表します。この値は、大/小文字の区別があります。		<ユーザー定義の struct または array 型の名前>	Oracle には CUSTOMER_STRUCT_TYPE という struct 型があります。 <jdbcasi: ChildBOTypeName> CUSTOMER_STRUCT _TYPE< /jdbcasi: ChildBOTypeName>
CLOB	Boolean	この属性に対応するデータベース列が CLOB データ型であるかどうかを示します。この値は、String 型の属性にのみ適用されます。 True の場合は、列のデータ型は CLOB です。 CLOB 属性は String 型で、その長さを使用して CLOB の長さを定義します。	なし	True、 False	<jdbcasi: CLOB> True< /jdbcasi:CLOB>
ColumnName	String	この属性に対応するデータベース列の名前。 唯一の必須パラメーターです。	なし	<列名>	<jdbcasi: ColumnName> pkey< /jdbcasi: ColumnName>
CopyAttribute	String	同一ビジネス・オブジェクトまたは親ビジネス・オブジェクト内から別の属性名を参照するユーザー指定値。 アプリケーション固有情報で設定されている値が、同一ビジネス・オブジェクト内の別の属性の名前を参照している場合、アダプターは、Create 操作でビジネス・オブジェクトをデータベースに追加する前に、その別の属性の値を使用してこの属性の値 (アプリケーション固有情報が定義されている属性) を設定します。 例えば、テーブルの新しい行の contact 列に、email 列と同じ値を組み込むには、contact 属性の CopyAttribute パラメーターに email を設定します。 値で子ビジネス・オブジェクトの属性を参照することはできませんが、属性名の前に 2 つのピリオドを付加することによって、親ビジネス・オブジェクト内の属性を参照できます。例えば、親ビジネス・オブジェクト内の ccode 属性を参照するには、..ccode と指定します。 アプリケーション固有情報にこのパラメーターを指定しないと、アダプターは、別の属性から値をコピーせずに現行属性の値を使用します。	なし	<属性名> または ..<属性名>	<jdbcasi: CopyAttribute> contact< /jdbcasi: CopyAttribute> または <jdbcasi: CopyAttribute> ..code< /jdbcasi: CopyAttribute>

表 24. 単純属性のアプリケーション固有情報 (続き)

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法

DateType	String	対応するエレメントが日付、時刻、またはタイム・スタンプであることを指定します。次の値のいずれかを指定してください。
		• Date
		• Time
		• Timestamp
		
		DateType 型の属性の値を設定するときには、次の形式で設定します。
		• Date には、*yyyy-MM-dd* を使用します。
		• Time には、*hh:mm:ss* を使用します。
		• Timestamp には、*yyyy-MM-dd hh:mm:ss.fffffffff* を使用します。
		
		注: java.sql.Timestamp.valueOf () メソッドにより、JDBC タイム・スタンプ形式が Timestamp 値に変換され、その値がデータベースに挿入されます。 java.sql.Timestamp.toString() メソッドは、データベースにある Timestamp 値をストリングに変換します。アダプターは、データベースに含まれる Timestamp 値を使用します。 Timestamp メソッドについての詳しい説明は、Sun の Web サイト (http://java.sun.com/j2se/1.5.0/docs/api/) で、*Timestamp* を検索し、参照してください。
			なし	Date、Time、Timestamp	`<jdbcasi:`
					`DateType>`
					`Timestamp<`
					`/jdbcasi:`
					`DateType>`

DateFormat	String	Date、Time、および Timestamp の各データ型の形式をカスタマイズできます。アダプターは、Date、Time、または Timestamp の SQL データ型をストリングに変換する (およびその逆の変換を行う) 必要があるときに、このパラメーターを使用します。	なし	Date、Time、Timestamp	`<jdbcasi:`
					`DateFormat>`
					`Time<`
					`/jdbcasi:`
					`DateFormat>`

DecimalScale	Int	10 進データ型の位取りを指定します。例えば、`unscaledVal` × 10^{-scale} です。	なし	<整数値>	`<jdbcasi:`
					`Decimal>`
					`3<`
					`/jdbcasi:`
					`Decimal>`
					
					元の値が 6.34444、変更後の値が 6.344 であることを示します。

Dummy	Boolean	ダミー列を示します。`True` の場合、ダミー列の値は更新されず、データベースにも挿入されません。このアプリケーション固有情報を使用するのは、 1 つの列に複数の ForeignKey 値を構成したいときです。	なし	True、 False	`<jdbcasi:`
					`Dummy>`
					`True<`
					`/jdbcasi:`
					`Dummy>`

表 24. 単純属性のアプリケーション固有情報 (続き)

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法

FixedChar	Boolean	テーブル内の列が VARCHAR 型ではなく CHAR 型である場合に、属性を固定長とするかどうかを指定します。例えば `true` に設定すると、ある特定の属性が CHAR 型の列にリンクされている場合、アダプターはデータベースを照会するときに、属性値をその属性の最大長までブランクで埋めます。
		
		ビジネス・オブジェクトの XSD ファイルでは、このパラメーターは手動で更新する必要があります。XML またはテキスト・エディターを使用してビジネス・オブジェクトを開いて XSD ファイルを編集し、以下の 2 つの変更を行ってください。
		1. オブジェクト属性の `<element>` タグにデフォルトで追加された `type="string"` を削除します。
		2. 次の例に示すように、`</element>` の前に新規に `<simpletype>` セクションを追加します。
		`<xsd:simpleType>`
		`<xsd:restriction`
		`base="xsd:string">`
		`<xsd:maxLength value="10"/>`
		`</xsd:restriction>`
		`</xsd:simpleType>`
		
		オブジェクト定義を保存して、更新後に XSD ファイル内で検証エラーが発生しないことを確認してください。
		
		この表の下にある、『ビジネス・オブジェクトの XSD ファイル内の FixedChar パラメーターの例』のセクションを参照してください。
			`false`	True、 False	`<jdbcasi:`
					`FixedChar>`
					`True<`
					`/jdbcasi:`
					`FixedChar>`

ForeignKey	String	このプロパティーの値は、親子関係が親ビジネス・オブジェクトに格納されるか、子ビジネス・オブジェクトに格納されるかによって異なります。
		
		関係が親に格納される場合、子ビジネス・オブジェクトのタイプと、外部キー *(Child_BO_name/Child_Property_Name)* として使用される子ビジネス・オブジェクト内の属性の名前の両方がこの値に含まれます。
		
		この関係が子に保管される場合は、外部キーとして使用される親ビジネス・オブジェクト内の属性の名前のみを含むように値を設定します。
		
		属性が外部キーではない場合は、アプリケーション固有情報にこのパラメーターを含めないでください。
			なし	`<`属性名`>` または `<childboname>` `/<childbo attributename>`	`<jdbcasi:`
					`ForeignKey>`
					`custinfoObj`
					`/custCode<`
					`/jdbcasi:`
					`ForeignKey>,`
					`<jdbcasi:`
					`ForeignKey>`
					`custCode<`
					`/jdbcasi:`
					`ForeignKey>`

OrderBy	String	値が指定されている場合、アダプターは RetrieveAll 操作の ORDER BY 節に指定された値を使用します。アダプターは、指定された順序どおりに、ビジネス・オブジェクトを取り出します。このパラメーターをアプリケーション固有情報に含めないと、アダプターは RetrieveAll 操作のための取り出し順序を指定しません。	なし	DESC、ASC	`<jdbcasi:`
					`OrderBy>`
					`ASC<`
					`/jdbcasi:`
					`OrderBy>`

PrimaryKey	Boolean	この属性に関連付けられている列が、データベース内の対応するテーブルの基本キーである場合、PrimaryKey パラメーターは `True` に設定されます。	なし	True、 False	`<jdbcasi:`
					`PrimaryKey>`
					`True<`
					`/jdbcasi:`
					`PrimaryKey>`

表 24. 単純属性のアプリケーション固有情報 (続き)

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法
SPParameterType	String	ストアード・プロシージャーのタイプを指定します。 使用可能な値は、以下のとおりです。 • IP (入力のみ) • OP (出力のみ) • IO (入出力) • RS (結果セット)	なし	IP、OP、 IO、RS	<jdbcasi: SPParameterType> IO< /jdbcasi: SPParameterType>
UniqueIdentifier (UID)	String	アダプターは、ビジネス・オブジェクトの固有 ID の生成に、このパラメーターを使用します。シーケンスと ID 列の生成がサポートされます (ID 列は、Informix では、*シリアル列* と呼ばれています)。DB2 は、シーケンスと ID 列の両方をサポートします。 ID 列は、データベースでテーブルの各行に自動的に固有の数値を生成する方法を提供します。 ID 列は、DB2 および Microsoft SQL Server の場合に定義でき、Informix の場合はシリアル列が定義できます。 このパラメーターの形式を以下に示します。 UID=AUTO|*Sequence_Name* DB2 または Microsoft SQL Server データベースのいずれかのテーブルに対してディスカバリー・プロセスを実行する場合、UID (固有 ID) 属性を手動で AUTO に設定する必要があります (例えば、<UID>AUTO</UID>)。 **注:** UID (固有 ID) 属性を手動で AUTO に設定する要件は、DB2 および Microsoft SQL Server の ID 列に特有のものです。この要件は、Informix のシリアル列には当てはまりません。Informix の場合、シリアル列の UID 属性は自動生成され、<UID>SERIAL</UID> または <UID>SERIAL8</UID> のいずれかになります。 ID 列と同様、シーケンスも数値の自動生成に使用されます。データベースによるシーケンスおよび ID 列の使用について詳しくは、ご使用のデータベースの資料を参照してください。 シーケンスの場合、UID 属性にシーケンス名を設定します。シーケンスは、DB2 および Oracle データベースで定義できます。 属性で固有 ID が必要とされない場合は、このパラメーターをアプリケーション固有情報に含めないでください。	なし	AUTO、 SERIAL、 SERIAL8	<jdbcasi: UID> AUTO< /jdbcasi:UID>
XML	Boolean	データベース内のテーブル列が XML 型のものである場合、XML パラメーターは True に設定されます。	なし	True、 False	<jdbcasi: XML> True< /jdbcasi: XML>

表 24. 単純属性のアプリケーション固有情報 (続き)

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法
UpdateAllCriteria	String	UpdateAll 操作のためのユーザー定義の照会基準。	なし	<組み込み SQL>	<jdbcasi: UpdateAllCriteria> where company.id = (select company.id from company, customer where company.id = customer.pkey and customer.pkey = :customerobj: pkey)< /jdbcasi: UpdateAllCriteria>
RetrieveAllCriteria	String	RetrieveAll 操作のためのユーザー定義の照会基準。	なし	<組み込み SQL>	<jdbcasi: UpdateAllCriteria> where company.id = (select company.id from company, customer where company.id = customer.pkey and customer.pkey = :customerobj: pkey)< /jdbcasi: Retrieve AllCriteria>
DeleteAllCriteria	String	DeleteAll 操作のためのユーザー定義の照会基準。	なし	<組み込み SQL>	<jdbcasi: DeleteAllCriteria> where company.id = (select company.id from company, customer where company.id = customer.pkey and customer.pkey = :customerobj: pkey)< /jdbcasi: DeleteAllCriteria>
ExistsCriteria	String	Exists 操作のためのユーザー定義の照会基準。	なし	<組み込み SQL>	<jdbcasi: ExistsCriteria> where company.id = (select company.id from company, customer where company.id= customer.pkey and customer.pkey = :customerobj :pkey)< /jdbcasi: ExistsCriteria>

属性のアプリケーション固有情報の形式は、XSD ファイルの以下の実例セクションに示します。

XSD ファイルのセクション例

```
      <jdbcasi:ColumnName>pkey</jdbcasi:ColumnName>
      <jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
      <jdbcasi:FixedChar>true</jdbcasi:FixedChar>
   </jdbcasi:JDBCAttributeTypeMetadata>
   </appinfo>
   </annotation>
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
      <xsd:maxLength value="10"/>
      </xsd:restriction>
```

```
</xsd:simpleType>
</element>
<element name="custCode" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>ccode</jdbcasi:ColumnName>
   <jdbcasi:ForeignKey>custinfoObj/custCode</jdbcasi:ForeignKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="firstName" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>fname</jdbcasi:ColumnName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="lastName" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>lname</jdbcasi:ColumnName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
```

ビジネス・オブジェクトの XSD ファイル内の FixedChar パラメーターの例

```
<element name="primaryKey">
<annotation>
<appinfo source="WBI">
        <jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
                <jdbcasi:ColumnName>pkey</jdbcasi:ColumnName>
                <jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
                <jdbcasi:FixedChar>true</jdbcasi:FixedChar>
        </jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
<xsd:simpleType>
        <xsd:restriction base="xsd:string">
                <xsd:maxLength value="10"/>
        </xsd:restriction>
</xsd:simpleType>
</element>
```

子ビジネス・オブジェクトを参照する属性のアプリケーション固有情報

子ビジネス・オブジェクトを参照する属性には、2 つのアプリケーション固有情報パラメーターが使用されます (単純属性ではなく複合属性)。このアプリケーション固有情報を設定する場合は、表 25に示すパラメーターを指定します。

表 25. 型が子ビジネス・オブジェクトの属性のアプリケーション固有情報

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法
KeepRelationship	Boolean	True の場合、このパラメーターにより Update 操作中に子ビジネス・オブジェクトは削除されません。	なし	True、False	<jdbcasi:KeepRelationship>True<</jdbcasi:KeepRelationship>

表 25. 型が子ビジネス・オブジェクトの属性のアプリケーション固有情報 (続き)

パラメーター	タイプ	説明	デフォルト値	有効な値	使用法
Ownership	Boolean	このパラメーターは、子ビジネス・オブジェクトが親によって所有されることを指定します。True の場合、子ビジネス・オブジェクトに対して Create、Update、および Delete 操作が許可されます。False の場合、どの更新も子ビジネス・オブジェクトには適用できません。親が作成されると、データベース内で関係の整合性が保持されるように、子が存在するかどうかが検証されます。	なし	True、False	<jdbcasi: Ownership> True< /jdbcasi: Ownership>

ビジネス・オブジェクトの XSD ファイル内の ownership の例

```
<element minOccurs="0" name="addressObj" type="bons0:OutboundRtasserAddress"
 maxOccurs="unbounded">
  <annotation>
    <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
    <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <jdbcasi:Ownership>true</jdbcasi:Ownership>
    </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
  </annotation>
</element>


<element minOccurs="0" name="custinfoObj" type="bons1:OutboundRtasserCustinfo"
maxOccurs="1">
 <annotation>
  <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
    <jdbcasi:Ownership>false</jdbcasi:Ownership>
   </jdbcasi:JDBCAttributeTypeMetadata>
  </appinfo>
 </annotation>
</element>
```

単一および複数カーディナリティー子ビジネス・オブジェクトの XSD ファイルのもう 1 つの例

単一または複数カーディナリティー子ビジネス・オブジェクトの XSD 定義ファイルの例をここに示します。エレメント custInfoObj は単一カーディナリティー子ビジネス・オブジェクトで、addressObj は複数カーディナリティー子ビジネス・オブジェクトです。

```
<element name="addressObj" minOccurs="1" type="Address:Address"
maxOccurs="unbounded">
                        <annotation>
                        <appinfo source="WBI">
                        <pasi:JDBCAttributeTypeMetadata xmlns:pasi=
```

```
"urn:app:jdbc:asi">
                                <pasi:Ownership>true</pasi:Ownership>
                        </pasi:JDBCAttributeTypeMetadata>
                        </appinfo>
                        </annotation>
                        </element>
                        <element name="custInfoObj" minOccurs="0" type=
"CustInfo:CustInfo" maxOccurs="1">
                        <annotation>
                        <appinfo source="WBI">
                        <pasi:JDBCAttributeTypeMetadata xmlns:pasi=
"urn:app:jdbc:asi">
                                <pasi:Ownership>false</pasi:Ownership>
                        </pasi:JDBCAttributeTypeMetadata>
                        </appinfo>
                        </annotation>
                        </element>
```

操作のアプリケーション固有情報

アダプターは、操作レベルのアプリケーション固有情報を使用してデータベース内の情報の取得や更新などの操作を実行します。アダプターは、ビジネス・オブジェクトでの指定に従って、SQL 照会、ストアード・プロシージャー、またはストアード関数を使用してデータベース表を取得および更新します。

ビジネス・オブジェクトにストアード・プロシージャーまたはストアード関数を追加する場合、操作レベルのアプリケーション固有情報 (ASI) を、表 26に指定されているとおりに設定します。

表 26. 操作に関するアプリケーション固有情報

操作 ASI の StoredProcedure のパラメーター・エレメント	**ウィザードによって設定**	**説明**
Parameters	はい	ストアード・プロシージャーのパラメーターをリストします。
PropertyName	はい	選択したビジネス・オブジェクト属性の名前に設定します。
ResultSet	いいえ	ストアード・プロシージャーから結果セットが返される場合は、ビジネス・オブジェクト定義でこのパラメーターを `True` に設定します。

表 26. 操作に関するアプリケーション固有情報 (続き)

操作 ASI の StoredProcedure のパラメーター・エレメント	ウィザードによって設定	説明
ReturnValue	はい	ストアード・プロシージャーに戻り値がある場合は、このパラメーターには次のいずれかの値が設定されます。 • ストリング RS。この値は、プロシージャーから結果セットが戻され、この結果セットを使用して、このビジネス・オブジェクトに対応する複数カーディナリティー・コンテナーが作成されることを示します。 • ビジネス・オブジェクト属性の名前。この値は、プロシージャーから戻される値が、実行時にビジネス・オブジェクトの特定の属性に割り当てられることを示します。 属性が別の子ビジネス・オブジェクトである場合、アダプターはエラーを返します。
StoredProcedure	はい	ストアード・プロシージャー名に設定します。
StoredProcedureType	はい	タイプのリストから選択します。 有効なストアード・プロシージャー・タイプについては、 67 ページの『ストアード・プロシージャー・タイプ』を参照してください。 有効なストアード・プロシージャー・タイプについては、ストアード・プロシージャー・タイプを参照してください。
タイプ	はい	ストアード・プロシージャーのパラメーターのタイプを設定します。使用可能な値は、以下のとおりです。 • IP (入力のみ) • OP (出力のみ) • IO (入出力) • RS (結果セット)

ビジネス・オブジェクト・レベルのアプリケーション固有情報

ビジネス・オブジェクト定義内のアプリケーション固有情報は、アダプターに対し、ビジネス・オブジェクトの処理方法に関するアプリケーション依存の指示を与えるものです。アダプターでは、ビジネス・オブジェクト、またはビジネス・オブジェクトの属性あるいは操作から取得したアプリケーション固有情報を解析して、Create、Update、Retrieve、および Delete 操作のための照会を生成します。

テーブルおよびビュー・ビジネス・オブジェクトのアプリケーション固有情報

ビジネス・オブジェクト・レベルのアプリケーション固有情報は、対応するデータベース表の名前を指定するとき、および物理削除または論理削除操作の実行に必要な情報を指定するときに使用されます。

外部サービス・ウィザードは、アプリケーション固有情報属性 TableName に、*SchemaName.TableName* という形式の値を設定します。物理/論理 Delete 操作の実行に必要な情報の入力を求めるプロンプトが出され、表 27 に示すビジネス・オブジェクト・レベルのアプリケーション固有情報が設定されます。

表 27. テーブル・ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報 (ASI)

アプリケーション固有情報	タイプ	説明
TableName	String	このビジネス・オブジェクトに対応するデータベース表の名前。
ステータス列名	String	アダプターがテーブルのデータを論理的に削除するか、または物理的に削除するかを示します。StatusColumnName パラメーターが設定されていない場合、データは物理的に削除されます。このパラメーターが設定されている場合は、論理的に削除された行を示す列の名前がパラメーターにより指定されます。外部サービス・ウィザードでテーブル・オブジェクトを選択するときに、このパラメーターを指定します。 このパラメーターは、Update 操作と Delete 操作の両方に適用されます。
ステータス値	String	列が論理的に削除されることを示す値。外部サービス・ウィザードでテーブル・オブジェクトを選択するときに、この値を指定します。

Update 操作または Delete 操作に対応して、アダプターが論理削除と物理削除操作のどちらを実行するかを判別する方法を示すため、Customer ビジネス・オブジェクトに、表 28に示すビジネス・オブジェクトのアプリケーション固有情報が含まれているとします。

表 28. テーブル・ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報のパラメーターの例

アプリケーション固有情報	値
TableName	customer
ステータス列名	status
ステータス値	deleted

アダプターが、顧客の削除要求を受信したとします。ビジネス・オブジェクトのアプリケーション固有情報に StatusColumnName パラメーターが含まれるため、アダプターは論理削除操作を実行します。このため、StatusValue パラメーターに指定されているストリング「deleted」を、StatusColumnName パラメーターで指定されている列である status 列に挿入します。

このような要求により、アダプターは次の SQL ステートメントを発行します。

```
UPDATE customer set status = 'deleted' where pkey = . . . .
```

ただし、 StatusColumnName パラメーターが設定されていない場合は、カスタマー・レコードは物理的に削除されます。アダプターは次の SQL ステートメントを実行します。

```
DELETE from customer where pkey = . . . .
```

ストアード・プロシージャー・ビジネス・オブジェクトのアプリケーション固有情報

ストアード・プロシージャーに基づくビジネス・オブジェクトの場合、外部サービス・ウィザードでビジネス・オブジェクト・レベルのアプリケーション固有情報 SPName が *SchemaName* + *SPName* の形の値に設定されます。設定されるビジネス・オブジェクト・レベルのアプリケーション固有情報を表 29に示します。ビジネス・オブジェクトの属性は、ストアード・プロシージャーの入出力パラメーターに基づいて作成されます。ストアード・プロシージャーが 1 つの戻り値を持つ場合は、対応するビジネス・オブジェクト属性が作成されます。戻り値または入出力パラメーターに複合データ型がある場合、ウィザードによってそれらの子ビジネス・オブジェクトが作成されます。

外部サービス・ウィザードでのデータベース・オブジェクトのディスカバリーは、ネストされた構造体および配列をサポートできます。返される結果セットからこれらの子ビジネス・オブジェクトが生成される場合、それらの子ビジネス・オブジェクトの名前の形式は *Prefix* + *SchemaName* + *SPName* + `RetRS` + *Number* になります。例えば、あるストアード・プロシージャーが 2 つの結果セットを返す場合、ウィザードはそれらに対応する 2 つの子ビジネス・オブジェクトを作成します。それぞれの名前は、*Prefix* + *SchemaName* + *SPName* + `RetRS1` と *Prefix* + *SchemaName* + *SPName* + `RetRS2` です。

子ビジネス・オブジェクトが、複合データ型の ResultSet、Struct、または Array を持つ入出力パラメーターから生成される場合、これらの子ビジネス・オブジェクト名の形式は *Prefix+SchemaName+SPName+ParameterName* になります。ネストされた構造体および配列に対応する子ビジネス・オブジェクトの場合、ビジネス・オブジェクト名の形式は *Prefix+SchemaName+SPName+ParameterName+ColumnName* になります。

表 29. ストアード・プロシージャーに基づくビジネス・オブジェクトの場合のビジネス・オブジェクトのアプリケーション固有情報 (ASI)

アプリケーション固有情報	タイプ	説明
SPName	String	ストアード・プロシージャーまたはストアード関数の名前
ResultSet	Boolean	ストアード・プロシージャーまたはストアード関数が結果セットを戻すかどうかを示します。`true` の場合は、ストアード・プロシージャーが 1 つ以上の結果セットを戻します。`false` の場合は、ストアード・プロシージャーまたはストアード関数は結果セットを戻しません。
MaxNumberOfRetRS	String	アダプター・ランタイムによって処理される、返される結果セットの最大数。

表 29. ストアード・プロシージャーに基づくビジネス・オブジェクトの場合のビジネス・オブジェクトのアプリケーション固有情報 (ASI) (続き)

アプリケーション固有情報	タイプ	説明
ReturnValue	String	ストアード・プロシージャーに戻り値がある場合は、対応するビジネス・オブジェクト属性の名前に設定されます。戻り値が単純データ型の場合は、属性も単純データ型です。戻り値が結果セットの場合、この属性は子ビジネス・オブジェクトを指します。

照会ビジネス・オブジェクトのアプリケーション固有情報

照会ビジネス・オブジェクトには、ビジネス・オブジェクト・レベルのアプリケーション固有情報が 1 つあります。表 30 にこの情報を示します。

表 30. 照会ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報 (ASI)

アプリケーション固有情報	タイプ	説明
SelectStatement	String	照会を実行する完全な SELECT ステートメント。外部サービス・ウィザードでこのステートメントを指定します。

また、外部サービス・ウィザードでは、すべてのビジネス・オブジェクトが最上位であるため、すべてのビジネス・オブジェクトのビジネス・グラフも生成されます。ビジネス・グラフの名前は、ビジネス・オブジェクト名に「BG」が付いたものです。例えば、JDBCSchema1Customer という名前のビジネス・オブジェクトのビジネス・グラフの名前は JDBCSchema1CustomerBG となります。ビジネス・オブジェクトに設定された操作はビジネス・グラフにも設定されます。

ウィザードは、ストアード・プロシージャー・ビジネス・オブジェクトを生成する場合に、必要であれば ResultSet、Struct、Array などの子ビジネス・オブジェクトを作成します。テーブル・ビジネス・オブジェクト間の親子関係は、Business Object Editor を使用して手動で作成します。

ウィザードは、同義語がストアード・プロシージャーのものであっても、同義語/ニックネームに基づくビジネス・オブジェクトもテーブルおよびビューに基づくオブジェクトのように処理します。

バッチ SQL ビジネス・オブジェクトのアプリケーション固有情報

バッチ SQL ビジネス・オブジェクトには、以下のアプリケーション固有情報が含まれています。

表 31. バッチ SQL ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報 (ASI)

アプリケーション固有情報	タイプ	説明
BatchSQLIndex	String	SQL ステートメントの順序を指定します。例えば、ユーザーが 1 つのバッチ SQL ビジネス・オブジェクトに 3 つのステートメントを指定する場合 (セミコロン区切り)、最初のステートメントの索引が 1、2 番目の索引が 2、最後の索引が 3 となります。

SQLStatement	String	ユーザーに指定された 1 つの INSERT、UPDATE、および DELETE SQL ステートメントを含みます。ユーザーが 1 つのビジネス・オブジェクトに複数の SQL ステートメントを指定した場合、それぞれが別の SQLStatement エレメントに格納されます。例えば、次のようになります。
		
		`Delete From Customer where pkey=?`
		
		`Insert into customer (pkey,ccode,fname,lname values(?,?,?,?)`

ラッパー・ビジネス・オブジェクトのアプリケーション固有情報

ラッパー・ビジネス・オブジェクトの場合、ラッパー・アプリケーション固有情報が追加され、`True` に設定されます。ラッパー・ビジネス・オブジェクトのビジネス・オブジェクト・レベルでは、その他のアプリケーション固有情報は不要です。

アプリケーション固有情報	タイプ	説明
Wrapper	Boolean	ビジネス・オブジェクトがラッパー・ビジネス・オブジェクトであるかどうかを示します。 Wrapper が `True` の場合、その他のビジネス・オブジェクト・レベル ASI は不要です。

命名規則

外部サービス・ウィザードでは、ビジネス・オブジェクトの生成時に、アダプターの命名規則に従った名前がビジネス・オブジェクトに指定されます。通常、そのビジネス・オブジェクト名はビジネス・オブジェクトの構造を示します。

外部サービス・ウィザードは、ビジネス・オブジェクト名の作成時に、ビジネス・オブジェクト名のアンダースコアー (_) 以外の特殊文字を、`U` とその後にその文字の Unicode 番号を続けたストリングに置き換えます。例えば、データベースの Order_Item テーブルのビジネス・オブジェクト名は Order_Item です。Shipping-Address テーブルのビジネス・オブジェクト名は ShippingU45Address です。

ビジネス・オブジェクト名には、アダプターまたはデータベースに対して意味を持つ値は含まれません。すなわち、ビジネス・オブジェクト名から情報や意味が派生することはありません。名前が別の名前で置換された場合でも、アダプターの動作は同じです。

ビジネス・オブジェクト名はデータベース固有のメタデータを扱います。名前のプレフィックスに JDBC や *%AppName%* のようなストリングを使用すると、アプリケーション固有と汎用の 2 つのタイプのビジネス・オブジェクトを区別するのに役立ちます。名前の残りの部分で、ビジネス・オブジェクトが表すテーブルまたはストアード・プロシージャーを説明することができます。例えば、Human Resources (HR) のようなデータベース・アプリケーションの Employee テーブル用のビジネス・オブジェクト定義を生成する場合、相当するビジネス・オブジェクト名は HREmployee です。

データベース照会、バッチ SQL ステートメント、およびラッパーのビジネス・オブジェクトなど、データベース・オブジェクトに対応しないビジネス・オブジェクトにテーブルやストアード・プロシージャーのビジネス・オブジェクトと同じ名前を付けると、その名前にウィザードによって固有の番号が付加されて、名前が重複しないようになります。

ビジネス・オブジェクト名ではグローバル化文字がサポートされています。

IBM Integration Designer のリファクタリング機能を使用して、ビジネス・オブジェクトの名前を変更することができます。詳細については、IBM Integration Designer の資料を参照してください。

ウィザードでビジネス・オブジェクトに対して使用される命名規則を以下の表に示します。

表 32. ビジネス・オブジェクトの命名規則

エレメント	命名規則
ビジネス・グラフ	親ビジネス・オブジェクトが含まれているビジネス・グラフの名前は、含まれるビジネス・オブジェクトの名前の後にストリング「BG」を付加したものになります。例えば、SalesCustomer ビジネス・オブジェクトを含むビジネス・グラフの名前は SalesCustomerBG になります。

表 32. ビジネス・オブジェクトの命名規則 (続き)

エレメント	命名規則
以下の項目に対応するビジネス・オブジェクト: • テーブル • ビュー • ストアード・プロシージャー • ストアード関数 • シノニムとニックネーム	テーブル、ビュー、ストアード・プロシージャー、シノニムおよびニックネームに基づくビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *SchemaName* + *ObjectName* という形式の名前を生成します。ここで、 • *Prefix* はプレフィックスという名前の外部サービス接続プロパティーで指定された値です。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *SchemaName* は、オブジェクトが属するスキーマの名前です。 • *ObjectName* はテーブル、ビュー、ストアード・プロシージャー、ストアード関数、またはシノニム/ニックネームの名前です。 同じ名前を持つ別のビジネス・オブジェクトと区別するために、必要に応じて、ビジネス・オブジェクトの名前に番号が付加されます。 例えば、Sales スキーマで Customer 表の Campaign12 プレフィックスを使用する場合、ビジネス・オブジェクト名は Campaign12SalesCustomer となります。
照会ビジネス・オブジェクト	照会ビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *QueryBOName* という形式の名前を生成します。ここで、 • *Prefix* は、ウィザードで指定するプレフィックスです。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *QueryBOName* は、ウィザードでビジネス・オブジェクトを構成したときに指定した値です。 同じ名前を持つ別のビジネス・オブジェクトと区別するために、必要に応じて、ビジネス・オブジェクトの名前に番号が付加されます。
BatchSQL ビジネス・オブジェクト	batchSQL ビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *BatchSQLBOName* という形式の名前を生成します。ここで、 • *Prefix* は、ウィザードで指定するプレフィックスです。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *BatchSQLBOName* は、ウィザードでビジネス・オブジェクトを構成したときに指定した名前です。 同じ名前を持つ別のビジネス・オブジェクトと区別するために、必要に応じて、ビジネス・オブジェクトの名前に番号が付加されます。

表 32. ビジネス・オブジェクトの命名規則 (続き)

エレメント	命名規則
ラッパー・ビジネス・オブジェクト	ラッパー・ビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *WrapperBOName* という形式の名前を生成します。ここで、 • *Prefix* は、ウィザードで指定するプレフィックスです。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *WrapperBOName* は、ウィザードでビジネス・オブジェクトを構成したときに指定した名前です。 同じ名前を持つ別のビジネス・オブジェクトと区別するために、必要に応じて、ビジネス・オブジェクトの名前に番号が付加されます。

構成プロパティー

IBM WebSphere Adapter for JDBC には、オブジェクトやサービスを生成したり作成したりするときに、外部サービス・ウィザードを使用して設定する、いくつかの種類の構成プロパティーがあります。IBM Business Process Manager または WebSphere Enterprise Service Bus にアプリケーションをデプロイした後に、リソース・アダプター、管理接続ファクトリー、活動化仕様のプロパティーを変更することができます。

Outbound 構成プロパティー

IBM WebSphere Adapter for JDBC には、オブジェクトやサービスを生成したり作成したりするときに、外部サービス・ウィザードを使用して設定する、いくつかの種類の Outbound 接続構成プロパティーがあります。リソース・アダプターおよび管理接続ファクトリーのプロパティーは、IBM Business Process Manager または WebSphere Enterprise Service Bus にモジュールをデプロイした後に、IBM Integration Designer または管理コンソールを使用して変更できますが、外部サービス・ウィザードの接続プロパティーは、デプロイメント後に変更することはできません。

プロパティーの詳細についてのガイド:

WebSphere Adapter for JDBC を構成するときに使用されるプロパティーは、リソース・アダプター・プロパティーや管理接続ファクトリー・プロパティーなど、それぞれの構成プロパティーのトピックに記載されている表で詳細に説明されています。 これらの表を使用しやすくするため、参照する各行の情報を以下に説明します。

次の表では、構成プロパティーの表に表示される場合がある各行の意味を説明します。

行	説明
必須	アダプターが動作するためには、必須フィールド (プロパティー) に値が必要です。必須プロパティーに対しては、外部サービス・ウィザードがデフォルト値を提供する場合があります。 外部サービス・ウィザードの*必須フィールド*から*デフォルト値を除去*しても、*デフォルト値*は*変更されません*。必須フィールドに値がまったく入っていない場合、外部サービス・ウィザード はそのフィールドに割り当てられたデフォルト値を使用してフィールドを処理し、そのデフォルト値は管理コンソールに表示されます。 可能な値は「**はい**」および「**いいえ**」です。 プロパティーは、他のプロパティーが特定の値の場合のみ必須となることがあります。その場合は、表にこの依存関係が記載されます。以下に例を示します。 • EventQueryType プロパティーが Dynamic に設定された場合は「はい」 • Oracle データベースの場合は「はい」
使用可能な値	プロパティーで選択可能な値をリストして説明します。
デフォルト	外部サービス・ウィザードによって設定される事前定義値。プロパティーが必須の場合は、デフォルト値を受け入れるか、ユーザーが値を指定する必要があります。プロパティーにデフォルト値がない場合、表には「デフォルト値なし」と記載されます。 `None` という語は、受け入れ可能なデフォルト値です。デフォルト値がないという意味ではありません。
計測単位	プロパティーの計測単位を指定します (例: キロバイト、秒)。
プロパティー・タイプ	プロパティー・タイプを示します。有効なプロパティー・タイプは以下のとおりです。 • Boolean • String • Integer

行	説明
使用法	プロパティーに適用される場合がある使用の条件または制限について記述します。制限の記載例を以下に示します。 Rational Application Developer for WebSphere Software バージョン 6.40 またはそれ以前では、パスワードに以下の制限があります。 • 大文字である必要があります • 長さが 8 文字である必要があります Rational Application Developer for WebSphere Software バージョン 6.40 よりも後のバージョンでは、パスワードの制限が以下のように変更されました。 • 大文字小文字を区別しません • 長さが 40 文字まで可能です このセクションでは、このプロパティーに影響を及ぼす他のプロパティー、またはこのプロパティーによって影響を受けるプロパティーをリストし、その条件付き関係の内容を説明します。
例	次のようなサンプル・プロパティー値が示されます。 「言語が JA (日本語) に設定された場合、コード・ページ番号は `8000` に設定されます。」
グローバル化	グローバル化される場合、プロパティーには各国語サポートがあるので、自国の言語に設定できます。 有効な値は「**はい**」および「**いいえ**」です。
BIDI 対応	プロパティーが双方向 (bidi) 処理でサポートされているかどうかを示します。双方向処理とは、同一ファイルに右から左 (ヘブライ語やアラビア語など) と左から右 (URL やファイル・パスなど) の両方の意味内容を含むデータを処理するタスクを指します。 有効な値は「**はい**」および「**いいえ**」です。

ウィザードの接続プロパティー:

外部サービス接続プロパティーは、外部サービス・ウィザード (ビジネス・オブジェクト作成ツール) とデータベース間の接続を確立するために使用されます。これらのプロパティーにより、接続構成、双方向変換プロパティー、およびウィザードのログ記録オプションなどが指定されます。接続の確立後に、ウィザードは、ビジネス・オブジェクトの作成に必要なメタデータをデータベース内でディスカバーできます。

データベース内のオブジェクトをディスカバーするためにウィザードで指定したプロパティーの一部は、ウィザードで後で指定するランタイム・プロパティーの初期値として使用されます。これらには、リソース・アダプター、管理接続ファクトリー、および活動化仕様のプロパティーが含まれます。

外部サービス・ウィザードの接続プロパティーとその目的を以下の表に示します。各プロパティーの完全な説明は、表に続くセクションで説明します。後続セクションのプロパティー詳細表の見方について詳しくは、 308 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 33. 外部サービス・ウィザードの接続プロパティー

ウィザードのプロパティー名	説明
追加の JDBC ドライバー接続プロパティー	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
データベース	データベースの名前を指定します。
データベース・ソフトウェア	アダプターが使用するデータベース管理ソフトウェアの名前およびバージョン
データベース URL	データベースへの接続に使用されるデータベース URL
ホスト名	データベース・サーバーのホスト名または IP アドレス。
JDBC ドライバー・クラス名	JDBC ドライバー・クラスの名前
JDBC ドライバー・タイプ	使用する JDBC ドライバーのタイプ
パスワード	対応するユーザー名のパスワード
ポート番号	データベース・インスタンスに接続する際のポート番号。
ビジネス・オブジェクト名のプレフィックス	ビジネス・オブジェクト名に追加されるプレフィックス
サーバー名	アダプターの接続先の Informix データベース・サーバーの名前
データベース接続時に自動コミットを設定	基礎となるデータベース接続が自動コミット・モードかどうかを指定します。
ユーザー名	データベース接続に使用するデータベース・ユーザー名を指定します。

外部サービス・ウィザードは、双方向接続プロパティーを使用して、エンタープライズ情報システムに渡すデータに適切な双方向変換を適用します。

追加の JDBC ドライバー接続プロパティー

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 34. 追加の JDBC ドライバー接続プロパティーの詳細

行	説明
必須	いいえ
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 34. 追加の JDBC ドライバー接続プロパティーの詳細 (続き)

行	説明
使用法	これらの接続プロパティーを、UserName および Password プロパティーと共に使用して、アダプターが使用するデータベース接続をカスタマイズします。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。

例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。
	
	`loginTimeout:20;readOnly:true;`
	`securityMechanism:USER_ONLY_SECURITY`
	
	高可用性環境では、High Availability and Disaster Recovery (HADR) データベースへの信頼できる接続のために DB2 ドライバーが必要とするプロパティー (retryIntervalForClientReroute、 maxRetriesForClientReroute、clientRerouteAlternateServerName、clientRerouteAlternatePortNumber) を取得するため、正しいプロパティー・ストリングが生成されるようこのプロパティーを構成する必要があります。以下に例を示します。
	
	`retryIntervalForClientReroute:15;`
	`maxRetriesForClientReroute:5;`
	`clientRerouteAlternateServerName:WLOXS01B.svl.ibm.com;`
	`clientRerouteAlternatePortNumber:50000`
	
	WebSphere 高可用性環境での WebSphere Adapter の使用について詳しくは、 https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=sw-app&S_PKG=wpswasadapt&S_TACT=105AGX10&S_CMP=LC を参照してください。

グローバル化	はい
BIDI 対応	いいえ

データベース

このプロパティーは、データベースの名前を指定します。

表 35. 「データベース名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値はデータベースによって異なります。
プロパティー・タイプ	String
使用法	これは、アクセスするデータベースの名前です。Oracle データベースの場合、これはデータベースを識別するシステム ID (SID) です。
グローバル化	はい
BIDI 対応	はい

データベース・ソフトウェア

このプロパティーは、アダプターがアクセスするデータベースを管理するデータベース管理ソフトウェアを指定します。

表 36. 「データベース・ソフトウェア」の詳細

行	説明
必須	はい
使用可能な値	このプロパティーは、一般的なデータベース・ソフトウェアを名前およびバージョン番号ごとにリストします。ご使用のソフトウェアがリストされていない場合は、「汎用 JDBC (Generic JDBC)」を選択してください。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードは、このプロパティーの値を使用して他のプロパティーのデフォルト値を設定し、データベース固有の選択リストを生成します。例えば、「DB2 USB Version 9.1」を選択すると、ウィザードの JDBC ドライバー・クラス・フィールドには、そのバージョンの DB2 UDB によってサポートされる JDBC ドライバーのみが表示されます。「Oracle 10」を選択すると、異なる JDBC ドライバーのセットが表示されます。
グローバル化	はい
BIDI 対応	はい

データベース URL

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 37. 「データベース URL」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これは、使用するデータベース・ソフトウェアと JDBC ドライバーに固有の値です。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。IP アドレスを大括弧 ([]) で囲んでください。

表 37. 「データベース URL」の詳細 (続き)

行	説明

例	一般的なデータベース・サーバーの標準的な値を以下に示します。
	DB2 Universal (タイプ 4) JDBC ドライバー
	`jdbc:db2://<Host_Name>/DB`
	DB2 Universal JDBC ドライバー (IPv6 アドレス)
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	DB2 Universal Database タイプ 2 ドライバー (ローカル接続用) `jdbc:db2:TEST`
	DB2 Universal Database タイプ 2 ドライバー (リモート接続用) `jdbc:db2://<Host_Name>/TEST`
	Informix V10
	`jdbc:informix-sql://<Host_Name>/symaster:INFORMIXSERVER=server`
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`
	Derby JDBC ドライバー (非リモート)
	`jdbc:derby://<runtime home>/runtimes/bi_v6/derby/databases/JDBCTEST`
	z/OS のリモート・テスト環境を使用している場合、Derby データベース URL に次の値を使用します。
	Derby JDBC ドライバー (リモート z/OS テスト環境)
	`jdbc:db2j:net://<HOST_NAME>:1527//<remote_derbydb_path>/JDBCTEST</remote_derbydb_path></HOST_NAME>`

グローバル化	はい
BIDI 対応	はい

ホスト名

このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。

表 38. 「ホスト名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。データベース・サーバーにより IPv6 がサポートされている場合は、ホスト名を IPv6 形式で指定できます。
グローバル化	はい
BIDI 対応	はい

JDBC ドライバー・クラス名

このプロパティーは、JDBC ドライバー・クラスの名前を指定します。

表 39. 「JDBC ドライバー・クラス名」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。
プロパティー・タイプ	String
使用法	ウィザードには、選択した JDBC ドライバー・タイプのデフォルト・クラス名が表示されますが、必要に応じて別のクラス名を入力できます。JDBC ドライバーの値として「**その他**」を選択すると、デフォルトは提供されず、クラス名の入力が必要になります。クラス名は、ウィザードの開始時に指定した JDBC ドライバー・ファイルに記述されている必要があります。
グローバル化	はい
BIDI 対応	いいえ

JDBC ドライバー・タイプ

このプロパティーは、使用する JDBC ドライバーのタイプを指定します。

表 40. 「JDBC ドライバー・タイプ」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。
プロパティー・タイプ	String
使用法	これは、使用する JDBC ドライバーのタイプです。根本的な点は使用するドライバーがタイプ 2 とタイプ 4 (ユニバーサル) のどちらであるかですが、各データベース・システムでは、ドライバーにデータベース・システム固有の名前が使用されています。各データベース・システムの既知のドライバーのリストがウィザードに表示されます。使用するドライバーがリストにない場合は、「その他」を選択します。このフィールドの情報は、ウィザードの開始時に指定した JDBC ドライバー・ファイルと一致している必要があります。
グローバル化	はい
BIDI 対応	いいえ

パスワード (Password)

対応するユーザー名のパスワード

表 41. パスワードの詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	オブジェクトをディスカバーする目的でデータベースに接続する際に入力されたユーザー名に関連付けられたパスワード。
グローバル化	はい
BIDI 対応	はい

ポート番号

このプロパティーは、データベース・インスタンスのポート番号を指定します。

表 42. ポート番号の詳細

行	説明
必須	はい
デフォルト	デフォルト値はデータベースによって異なります。また、JDBC ドライバー・タイプに特定のドライバーを選択した場合、ウィザードによってデフォルト値があらかじめ設定されます。
プロパティー・タイプ	String
使用法	これは、データベース・インスタンスへ接続するポートのポート番号です。 JDBC ドライバー・タイプに、その他を選択する場合は、このプロパティーは使用できません。
グローバル化	はい
BIDI 対応	いいえ

ビジネス・オブジェクト名のプレフィックス

ビジネス・オブジェクトの名前に追加されるプレフィックス。

表 43. 「プレフィックス」の詳細

行	説明
必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	プレフィックスを使用して、ビジネス・オブジェクトのタイプを容易に区別できるようにします。
例	汎用ビジネス・オブジェクトにプレフィックス `JDBC` を指定し、アプリケーション固有のビジネス・オブジェクトに `%AppName%` を指定する場合があります。

表 43. 「プレフィックス」の詳細 (続き)

行	説明
グローバル化	はい
BIDI 対応	いいえ

サーバー名

アダプターの接続先 Informix データベース・サーバーのデフォルトの名前を指定します。

表 44. 「サーバー名」の詳細

行	説明
必須	はい
デフォルト	サーバー
プロパティー・タイプ	String
使用法	サーバー名の値は、ローカル・サーバーでもリモート・サーバーでも構いませんが、アプリケーションを実行しているコンピューターの `$INFORMIXDIR/etc/sqlhosts` ファイルにある有効な `dbservername` 項目に対応している必要があります。 `dbservername` は、小文字で始まらなければなりません。また、128 バイトを超過してはいけません。大文字、フィールド区切り文字 (空白スペースまたはタブ)、改行文字、およびハイフン (マイナス) 記号を除く任意の印刷可能文字を含むことができます。
グローバル化	いいえ
BIDI 対応	いいえ

データベース接続時に自動コミットを設定

基礎となるデータベース接続が自動コミット・モードかどうかを指定します。ストアード・プロシージャーのトランザクション・モードについての Sybase データベースのデフォルト設定を指定変更します。

表 45. 「データベース接続時に自動コミットを設定」の詳細

行	説明
必須	はい (特定の JDBC ドライバーの場合、および特定のストアード・プロシージャー構成の場合)。
デフォルト	False
プロパティー・タイプ	Boolean

表 45. 「データベース接続時に自動コミットを設定」の詳細 (続き)

行	説明
使用法	この値を設定して、基礎となるデータベース接続が自動コミット・モードかどうかを指定します。このプロパティーを True に設定した場合、基礎となるデータベースに対する変更は、ウィザードの完了後はロールバックしません。 Sybase データベースで JConnect ドライバーを使用してディスカバリーを実行している場合で、なおかつ Sybase データベース上のストアード・プロシージャーのトランザクション・モードの設定が「非チェーン・モード」または「Transact-SQL モード」になっている場合、このプロパティーの設定は必須です。 このプロパティーを True に設定することで、アダプターが結果セットのディスカバリーをできない Sybase トランザクション・モードの構成を指定変更します。
グローバル化	はい
BIDI 対応	いいえ

ユーザー名 (UserName)

このプロパティーは、データベース接続に使用するデータベース・ユーザー名を指定します。

表 46. 「ユーザー名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	ユーザー名は、オブジェクトをディスカバーする目的でデータベースに接続する場合に入力する名前です。
グローバル化	はい
BIDI 対応	はい

リソース・アダプター・プロパティー:

リソース・アダプター・プロパティーは、アダプターの一般的な操作 (ビジネス・オブジェクトの名前空間の指定など) を制御します。リソース・アダプター・プロパティーは、アダプターの構成時に外部サービス・ウィザードを使用して設定します。アダプターのデプロイ後に、これらのプロパティーを変更するには、管理コンソールを使用します。

ロギングおよびトレースに関する次のプロパティーは、非推奨になっています。

- ログ・ファイル最大サイズ
- ログ・ファイル名
- ログ・ファイル数
- トレース・ファイル最大サイズ

- トレース・ファイル名
- トレース・ファイル数

以下のリソース・アダプター・プロパティーは、 Inbound 処理用の活動化仕様または Outbound 処理用の管理接続ファクトリーに同じプロパティーが設定されている場合には指定変更されます。

- DatabaseVendor
- ping 照会
- 照会タイムアウト
- ReturnDummyBOForSP

BONameSpaceプロパティーは、活動化仕様プロパティーに移動しました。

以下の表に、リソース・アダプター・プロパティーとその目的のリストを示します。各プロパティーの完全な説明は、表に続くセクションで説明します。プロパティー詳細表の見方について詳しくは、 308 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 47. Adapter for JDBC 用のリソース・アダプター・プロパティー

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
アダプター ID	AdapterID	PMI イベントのアダプター・インスタンス、ロギングおよびトレースのアダプター・インスタンスを識別する場合に使用します。
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する	HideConfidentialTrace	ログおよびトレース・ファイルにユーザー・データではなく X ストリングを書き込み、潜在的な機密情報を隠すようにするかどうかを指定します。
照会タイムアウト (秒)	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
322 ページの『高可用性 サポートを使用可能にする (enableHASupport)』	enableHASupport	WebSphere Adapter for JDBC の構成モード (Active-Active または Active-Passive) を指定します。
(なし)	LogFileSize	非推奨
(なし)	ログ・ファイル名	非推奨
(なし)	ログ・ファイル数	非推奨
接続を検証するための SQL 照会	PingQuery	データベースへの接続の信頼性をテストするのに使用する SQL 照会
(なし)	TraceFileSize	非推奨
(なし)	トレース・ファイル名	非推奨
(なし)	トレース・ファイル数	非推奨

アダプター ID (AdapterID)

このプロパティーは、アダプターの特定のデプロイメントまたはインスタンスを識別します。

表 48. 「アダプター ID」の詳細

必須	はい
デフォルト	001
プロパティー・タイプ	String
使用法	このプロパティーは、ログおよびトレース・ファイル内のアダプター・インスタンスを識別します。また、アダプターのモニター時にアダプター・インスタンスを識別する場合に役立ちます。アダプター ID は、アダプター固有の ID、JDBCRA と共に使用され、 Log and Trace Analyzer ツールによって使用されるコンポーネント名を構成します。例えば、アダプター ID プロパティーが、001 に設定されている場合、コンポーネント ID は、JDBCRA001 となります。 同じアダプターの複数のインスタンスを実行する場合、アダプター ID プロパティーの最初の 7 文字は、必ずインスタンスごとに固有のものにし、ログおよびトレース情報を特定のアダプター・インスタンスに相互に関連付けられるようにしてください。アダプター ID プロパティーの最初の 7 文字を固有のものにすることにより、そのアダプターの複数インスタンスのコンポーネント ID も固有のものになり、アダプターの特定インスタンスにログおよびトレース情報を相互に関連付けることができるようになります。 例えば、WebSphere Adapter for JDBC の 2 つのインスタンスのアダプター ID プロパティーを 001 および 002 に設定するとします。これらのインスタンスのコンポーネント ID、JDBCRA001 および JDBCRA002 は、短いので固有性を保つことができ、別のアダプター・インスタンスとして区別することができます。しかし、もっと長いアダプター ID プロパティーのインスタンスの場合、互いを区別できなくなります。2 つのインスタンスのアダプター ID プロパティーを Instance01 と Instance02 に設定した場合、各アダプター・インスタンスのログおよびトレース情報を調べることはできなくなります。これは、両方のインスタンスのコンポーネント ID が JDBCRAInstanc に切り捨てられるためです。 Inbound 処理の場合、このプロパティーの値は、リソース・アダプター・レベルで設定されます。Outbound 処理の場合、この値は、リソース・アダプター・レベルと管理接続ファクトリー・レベルの両方で設定できます。外部サービス・ウィザードを使用してアダプターを Outbound 処理用に構成した後、リソース・アダプター・プロパティーおよび管理接続ファクトリー・プロパティーを個別に設定できます。IBM Integration Designer アセンブリー・エディターまたは管理コンソールを使用してこれらのプロパティーを再設定する場合は、ログおよびトレース・エントリーのマーキングが不整合にならないように、矛盾がない設定になっていることを確認してください。
グローバル化	はい
BIDI 対応	いいえ

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 49. 「データベース・ベンダー」の詳細

必須	はい
使用可能な値	DB2 Informix MSSQLServer Oracle Others
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 その他のデータベースの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する (HideConfidentialTrace)

このプロパティーは、ログおよびトレース・ファイル中のユーザー・データを「X」のストリングに置換し、潜在的な機密データが許可なく外部に漏れないようにします。

表 50. ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述するの詳細

必須	いいえ
使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean

表 50. ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述するの詳細 (続き)

使用法	このプロパティーを True に設定すると、アダプターでは、ログおよびトレース・ファイルに書き込む時に、ユーザー・データを「X」のストリングに置換します。 Inbound 処理の場合、このプロパティーの値は、リソース・アダプター・レベルで設定されます。Outbound 処理の場合、この値は、リソース・アダプター・レベルと管理接続ファクトリー・レベルの両方で設定できます。外部サービス・ウィザードを使用してアダプターを Outbound 処理用に構成した後、リソース・アダプター・プロパティーおよび管理接続ファクトリー・プロパティーを個別に設定できます。IBM Integration Designer アセンブリー・エディターまたは管理コンソールを使用してこれらのプロパティーを再設定する場合は、ログおよびトレース・エントリーのマーキングが不整合にならないように、矛盾がない設定になっていることを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

高可用性 サポートを使用可能にする (enableHASupport)

このプロパティーは、 WebSphere Adapter for JDBC の構成モード (Active-Active または Active-Passive) を指定するために使用します。

表 51. 「高可用性 サポートを使用可能にする」プロパティーの特性

必須	いいえ
使用可能な値	True False
デフォルト	True (管理コンソールの enableHASupport プロパティーの値が true に設定されており、アダプターが Active-Passive モードであることを示します。)
プロパティー・タイプ	Boolean
使用法	このプロパティーを false に設定すると、アダプターはクラスター環境で Active-Active モードになります。これにより、複数のアダプター・インスタンスが異なるサーバー・ノードでアクティブになります。各アダプター・インスタンスが別々のイベントを並行して処理します。その結果、各アダプター・インスタンスはそれぞれ別の固有イベントをポーリングし、イベントは重複することなくエンドポイントに送達されます。
グローバル化	いいえ
BIDI 対応	いいえ

照会タイムアウト (秒) (QueryTimeOut)

このプロパティーは、1 つの照会ですべての SQL ステートメントの実行に費やすことのできる最大時間を秒数で指定します。

表 52. 「照会タイムアウト」の詳細

必須	いいえ

表 52. 「照会タイムアウト」の詳細 (続き)

デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会を処理する際に、指定された秒数より長い時間がかかると、キャプチャーされた SQL 例外がデータベースから戻されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	いいえ
BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 53. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean
使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが `False` に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP プロパティーが `True` に設定されている場合は、ダミーのビジネス・オブジェクトが作成され、ストアード・プロシージャーから返されるパラメーター (出力パラメーターと入出力パラメーターを含む) が、対応する属性に取り込まれます。
グローバル化	いいえ
BIDI 対応	いいえ

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の信頼性をテストするために使用される SQL 照会を指定します。

表 54. 「ping 照会」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし
使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。 アダプターは、Outbound 操作の実行時に SQLException 例外を受け取るたびに、ping 照会を実行します。 アダプターは、接続のリカバリーを試行しません。 ping 照会により、データベースの接続が有効でなくなったことがわかると、アダプターはコンテナーに通知します。失効した接続のプールからの削除は、接続プール・マネージャーが行います。こうすることで、後続の Outbound 要求の処理が可能になります。
グローバル化	いいえ
BIDI 対応	いいえ

管理接続ファクトリー・プロパティー:

管理接続ファクトリー・プロパティーは、データベースとの Outbound 接続インスタンスを作成するために、アダプターが実行時に使用します。

アダプターの構成時に、外部サービス・ウィザードで管理接続ファクトリー・プロパティーを設定します。プロパティーは、デプロイメント前に IBM Integration Designer アセンブリー・エディターを使用して、またはデプロイメント後に IBM Business Process Manager または WebSphere Enterprise Service Bus の管理コンソールを使用して変更できます。

以下の表に管理接続ファクトリー・プロパティーの説明を示します。各プロパティーの完全な説明は、表に続くセクションで説明します。表に続くセクションについては、 308 ページの『プロパティーの詳細についてのガイド』を参照してください。

注: 外部サービス・ウィザードは、これらのプロパティーを管理接続ファクトリー・プロパティーとして参照し、管理コンソールは J2C 接続ファクトリー・プロパティーとして参照します。

表 55. Adapter for JDBC の管理接続ファクトリー・プロパティー

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
アダプター ID	AdapterID	PMI イベントのアダプター・インスタンス、ロギングおよびトレースのアダプター・インスタンスを識別する場合に使用します。

表 55. *Adapter for JDBC の管理接続ファクトリー・プロパティー (続き)*

プロパティー名		説明
ウィザード内	**管理コンソール内**	
追加の JDBC ドライバー接続プロパティー [name:value;name:value]	JDBCDriverConnectionProperties	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
自動コミット (Auto commit)	自動コミット	接続に使用する AutoCommit 値。
データベース接続情報	ConnectionType	アダプターがデータベースへの接続を確立する方法を指定します。
データ・ソース JNDI 名	DataSourceJNDIName	非推奨
データベース URL	DatabaseURL	データベースへの接続に使用されるデータベース URL
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
(なし)	ErrorOnEmptyResultSet	レコードが検出されない場合に例外を生成するかどうかを指定します。
ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する	HideConfidentialTrace	ログおよびトレース・ファイルにユーザー・データではなく X ストリングを書き込み、潜在的な機密情報を隠すようにするかどうかを指定します。
JDBC ドライバー・クラス (JDBC driver class)	JDBCDriverClass	データベース接続に使用される JDBC ドライバーのクラス名
接続に失敗した場合の最大再試行回数	ConnectionRetryLimit	アダプターが データベース への Outbound 接続の再確立を試行する最大回数を指定します。
パスワード	Password	対応するユーザー名のパスワード
接続プール・データ・ソースの JNDI 名	PoolDataSourceJNDIName	データベースへの接続の確立に使用される接続プール・データ・ソースの JNDI 名。
照会タイムアウト (秒)	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
接続再試行間隔 (ミリ秒単位)	ConnectionRetryInterval	接続に失敗した場合に データベース への再接続を試行する時間間隔を指定します。
ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
「接続を検証するための SQL 照会」	PingQuery	データベースへの接続の信頼性をテストするのに使用する SQL 照会
ユーザー名	UserName	データベース・ユーザー名
XA DataSource 名	XADataSourceName	このプロパティーでは、XA (分散) トランザクションを実行するためにデータベース接続を確立するときに使用される XA データ・ソースの名前を指定します。
XA データベース名	XADatabaseName	XA 接続に使用されるデータベース名
XA DataSource JNDI 名	XADataSourceJNDIName	データベースへの接続の確立に使用される XA データ・ソースの JNDI 名。

アダプター ID (AdapterID)

このプロパティーは、アダプターの特定のデプロイメントまたはインスタンスを識別します。

表 56. 「アダプター ID」の詳細

必須	はい
デフォルト	001
プロパティー・タイプ	String
使用法	このプロパティーは、ログおよびトレース・ファイル内のアダプター・インスタンスを識別します。また、アダプターのモニター時にアダプター・インスタンスを識別する場合に役立ちます。アダプター ID は、アダプター固有の ID、JDBCRA と共に使用され、 Log and Trace Analyzer ツールによって使用されるコンポーネント名を構成します。例えば、アダプター ID プロパティーが、001 に設定されている場合、コンポーネント ID は、JDBCRA001 となります。 同じアダプターの複数のインスタンスを実行する場合、アダプター ID プロパティーの最初の 7 文字は、必ずインスタンスごとに固有のものにし、ログおよびトレース情報を特定のアダプター・インスタンスに相互に関連付けられるようにしてください。アダプター ID プロパティーの最初の 7 文字を固有のものにすることにより、そのアダプターの複数インスタンスのコンポーネント ID も固有のものになり、アダプターの特定インスタンスにログおよびトレース情報を相互に関連付けることができるようになります。 例えば、WebSphere Adapter for JDBC の 2 つのインスタンスのアダプター ID プロパティーを 001 および 002 に設定するとします。これらのインスタンスのコンポーネント ID、JDBCRA001 および JDBCRA002 は、短いので固有性を保つことができ、別のアダプター・インスタンスとして区別することができます。しかし、もっと長いアダプター ID プロパティーのインスタンスの場合、互いを区別できなくなります。2 つのインスタンスのアダプター ID プロパティーを Instance01 と Instance02 に設定した場合、各アダプター・インスタンスのログおよびトレース情報を調べることはできなくなります。これは、両方のインスタンスのコンポーネント ID が JDBCRAInstanc に切り捨てられるためです。 Inbound 処理の場合、このプロパティーの値は、リソース・アダプター・レベルで設定されます。Outbound 処理の場合、この値は、リソース・アダプター・レベルと管理接続ファクトリー・レベルの両方で設定できます。外部サービス・ウィザードを使用してアダプターを Outbound 処理用に構成した後、リソース・アダプター・プロパティーおよび管理接続ファクトリー・プロパティーを個別に設定できます。IBM Integration Designer アセンブリー・エディターまたは管理コンソールを使用してこれらのプロパティーを再設定する場合は、ログおよびトレース・エントリーのマーキングが不整合にならないように、矛盾がない設定になっていることを確認してください。
グローバル化	はい
BIDI 対応	いいえ

追加の JDBC ドライバー接続プロパティー [name:value;name:value] (JDBCDriverConnectionProperties)

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 57. 追加の JDBC ドライバー接続プロパティーの詳細

行	説明
必須	いいえ
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これらの接続プロパティーを、UserName および Password プロパティーと共に使用して、アダプターが使用するデータベース接続をカスタマイズします。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。

例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。
	
	`loginTimeout:20;readOnly:true;`
	`securityMechanism:USER_ONLY_SECURITY`
	
	高可用性環境では、High Availability and Disaster Recovery (HADR) データベースへの信頼できる接続のために DB2 ドライバーが必要とするプロパティー (retryIntervalForClientReroute、 maxRetriesForClientReroute、clientRerouteAlternateServerName、clientRerouteAlternatePortNumber) を取得するため、正しいプロパティー・ストリングが生成されるようこのプロパティーを構成する必要があります。以下に例を示します。
	
	`retryIntervalForClientReroute:15;`
	`maxRetriesForClientReroute:5;`
	`clientRerouteAlternateServerName:WLOXS01B.svl.ibm.com;`
	`clientRerouteAlternatePortNumber:50000`
	
	WebSphere 高可用性環境での WebSphere Adapter の使用について詳しくは、 https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=sw-app&S_PKG=wpswasadapt&S_TACT=105AGX10&S_CMP=LC を参照してください。

グローバル化	はい
BIDI 対応	いいえ

自動コミット (Auto commit) (AutoCommit)

このプロパティーは、接続に AutoCommit を設定するかどうかを指定します。

表 58. 「自動コミット (Auto commit)」の詳細

必須	いいえ

表 58. 「自動コミット (Auto commit)」の詳細 (続き)

使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean
使用法	このプロパティーは、XA (分散) トランザクションでは無視されます。
グローバル化	いいえ
BIDI 対応	いいえ

データベース URL (DatabaseURL)

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 59. 「データベース URL」の詳細

必須	接続タイプが LocalConnectionProps または ConnectionProps の場合は、はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードのデータベース固有のフィールドに情報を入力し、データベース URL を作成します。例えば、DB2 データベースのデータベース URL は、データベース名、サーバー・ホスト名、およびデータベース・ポート番号で構成されています。管理コンソールで、データベース URL 値全体を入力します。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。 ホスト名を IPv6 形式の IP アドレスとして指定する場合は、その IP アドレスを大括弧 ([]) で囲んでください。

例	一般的なデータベース・サーバーの標準的な値を以下に示します。
	
	DB2 Universal (タイプ 4) JDBC ドライバー
	`jdbc:db2://www.example.com:50000/DB`
	
	DB2 Universal JDBC ドライバー (IPv6 アドレス)
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	
	DB2 Universal Database タイプ 2 ドライバー (ローカル接続用) jdbc:db2:TEST
	
	DB2 Universal Database タイプ 2 ドライバー (リモート接続用) jdbc:db2://www.example.com:50000/TEST
	
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

グローバル化	はい
BIDI 対応	はい

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 60. 「データベース・ベンダー」の詳細

必須	はい
使用可能な値	DB2 Informix MSSQLServer Oracle Others
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 その他のデータベースの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する (HideConfidentialTrace)

このプロパティーは、ログおよびトレース・ファイル中のユーザー・データを「X」のストリングに置換し、潜在的な機密データが許可なく外部に漏れないようにします。

表 61. ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述するの詳細

必須	いいえ
使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean

表 61. ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述するの詳細 (続き)

使用法	このプロパティーを `True` に設定すると、アダプターでは、ログおよびトレース・ファイルに書き込む時に、ユーザー・データを「`X`」のストリングに置換します。 Inbound 処理の場合、このプロパティーの値は、リソース・アダプター・レベルで設定されます。Outbound 処理の場合、この値は、リソース・アダプター・レベルと管理接続ファクトリー・レベルの両方で設定できます。外部サービス・ウィザードを使用してアダプターを Outbound 処理用に構成した後、リソース・アダプター・プロパティーおよび管理接続ファクトリー・プロパティーを個別に設定できます。IBM Integration Designer アセンブリー・エディターまたは管理コンソールを使用してこれらのプロパティーを再設定する場合は、ログおよびトレース・エントリーのマーキングが不整合にならないように、矛盾がない設定になっていることを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

レコードが一切検出されない (ErrorOnEmptyResultSet) 場合は例外をスローします。

このプロパティーは、RetrieveAll 操作の RecordNotFoundException プロパティーに対して ErrorOnEmptyResultSet プロパティーが設定されるかどうかを指定します。

表 62. 「レコードが一切検出されない場合の例外のスロー」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`True`
プロパティー・タイプ	Boolean
使用法	このプロパティーを `False` に設定すると、RetrieveAll 操作は、レコードが一切検出されないときに RecordNotFoundException 例外を生成しません。
グローバル化	いいえ
BIDI 対応	いいえ

JDBC ドライバー・クラス (JDBC driver class) (JDBCDriverClass)

このプロパティーは、データベース接続に使用される JDBC ドライバーのクラス名を指定します。

表 63. 「JDBC ドライバー・クラス (JDBC driver class)」の詳細

行	説明
必須	接続タイプが LocalConnectionProps または ConnectionProps の場合は、はい
使用可能な値	値はデータベースによって異なります。
デフォルト	デフォルト値なし

表 63. 「JDBC ドライバー・クラス (JDBC driver class)」の詳細 (続き)

プロパティー・タイプ	String
使用法	外部サービス・ウィザードでは、共通データベース・ソフトウェアとドライバーの組み合わせ (IBM DB2、Oracle、および Microsoft SQL の最新バージョンのタイプ 4 ドライバーなど) を選択すると、JDBC ドライバー・クラスが表示されます。ほとんどのデータベース・ソフトウェアのほとんどのタイプ 2 ドライバーでは、データベース・クラス名を入力する必要があります。 例えば、DB2 Universal Database タイプ 2 ドライバーの場合、クラス名は `COM.ibm.db2.jdbc.app.DB2Driver` です。 管理コンソールで、ドライバーのデータベース固有名を入力してください。

例	外部サービス・ウィザードおよび管理コンソールの両方の JDBC ドライバー・クラス表示画面の値。次の例は、外部サービス・ウィザード と管理コンソールの両方の JDBC ドライバーのクラス・プロパティーを示しています。**外部サービス・ウィザード の場合:**
	• ユニバーサル (タイプ 4) JDBC ドライバーを使用して DB2 データベースに接続するには、「`IBM DB2 Universal`」を選択します。
	• DB2 ユニバーサルのタイプ 2 ドライバーを使用して DB2 データベースに接続するには、「`Other`」を選択します。
	• タイプ 4 ドライバーを使用して Oracle 10 データベースに接続するには、「`Oracle Thin Driver`」を選択します。
	
	管理コンソール内
	
	DB2 Universal Database タイプ 2 ドライバー
	`COM.ibm.db2.jdbc.app.DB2Driver`
	
	DB2 Universal Database タイプ 4 ドライバー
	`com.ibm.db2.jcc.DB2Driver`
	
	Oracle Thin JDBC ドライバー
	`oracle.jdbc.driver.OracleDriver`
	
	IBM Toolkit for Java リモート・ドライバー (IBM i 用)
	`com.ibm.as400.access.AS400JDBCDriver`
	
	IBM WebSphere Connect JDBC ドライバー (Microsoft SQL Server 用)
	`com.ibm.websphere.jdbc.sqlserver.SQLServerDriver`

グローバル化	いいえ
BIDI 対応	いいえ

パスワード (Password)

このプロパティーは、データベース・ユーザーのユーザー名に対するパスワードを指定します。

表 64. パスワードの詳細

必須	いいえ。Inbound 処理では、認証別名または DataSourceJNDIName を設定する場合は、パスワードは必須ではありません。ただし、DataSourceJNDIName、および「パスワード」フィールドを設定した場合、パスワードに指定した値が優先されます。 Outbound 処理では、認証別名、XADataSourceJNDIName プロパティー、または PoolDataSourceJNDIName プロパティーを設定した場合、パスワードは必須ではありません。ただし、XADataSourceJNDIName または PoolDataSourceJNDIName、および「パスワード」フィールドを設定した場合、パスワードに指定した値が優先されます。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	Inbound 処理の場合、このプロパティーを設定すると、認証別名または DataSourceJNDIName プロパティーを使用してサーバーのデータ・ソースに指定されたパスワードが指定変更されます。 Outbound 処理の場合、このプロパティーを設定すると、認証別名または、XADataSourceJNDIName または PoolDataSourceJNDIName プロパティーを使用してサーバーのデータ・ソースに指定されているパスワードが上書きされます。 JAAS をセキュリティー資格情報として指定すると、このプロパティーは認証別名によって指定変更されます。
グローバル化	はい
BIDI 対応	はい

照会タイムアウト (秒) (QueryTimeOut)

このプロパティーは、1 つの照会ですべての SQL ステートメントの実行に費やすことのできる最大時間を秒数で指定します。

表 65. 「照会タイムアウト」の詳細

必須	いいえ
デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会を処理する際に、指定された秒数より長い時間がかかると、キャプチャーされた SQL 例外がデータベースから戻されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	いいえ

表 65. 「照会タイムアウト」の詳細 (続き)

BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 66. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean
使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが `False` に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP プロパティーが `True` に設定されている場合は、ダミーのビジネス・オブジェクトが作成され、ストアード・プロシージャーから返されるパラメーター (出力パラメーターと入出力パラメーターを含む) が、対応する属性に取り込まれます。
グローバル化	いいえ
BIDI 対応	いいえ

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の信頼性をテストするために使用される SQL 照会を指定します。

表 67. 「ping 照会」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし

表 67. 「ping 照会」の詳細 (続き)

使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。 アダプターは、Outbound 操作の実行時に SQLException 例外を受け取るたびに、ping 照会を実行します。 アダプターは、接続のリカバリーを試行しません。 ping 照会により、データベースの接続が有効でなくなったことがわかると、アダプターはコンテナーに通知します。失効した接続のプールからの削除は、接続プール・マネージャーが行います。こうすることで、後続の Outbound 要求の処理が可能になります。
グローバル化	いいえ
BIDI 対応	いいえ

ユーザー名 (UserName)

このプロパティーは、データベースのアクセスに使用されるデータベース・ユーザー名を指定します。

表 68. 「ユーザー名」の詳細

必須	いいえ。Inbound 処理では、認証別名または DataSourceJNDIName を設定した場合、ユーザー名プロパティーは必須ではありません。ただし、DataSourceJNDIName、および「ユーザー名」フィールドを設定した場合、ユーザー名に指定した値が優先されます。 Outbound 処理では、認証別名または、XADataSourceJNDIName または PoolDataSourceJNDIName を設定した場合、ユーザー名は必須ではありません。ただし、XADataSourceJNDIName または PoolDataSourceJNDIName、および「ユーザー名」フィールドを設定した場合、ユーザー名に指定した値が優先されます。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	Inbound 処理の場合、このプロパティーを設定すると、DataSourceJNDIName プロパティーまたは認証別名を使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。 Outbound 処理の場合、このプロパティーを設定すると、XADataSourceJNDIName プロパティー、PoolDataSourceJNDIName プロパティーまたは認証別名を使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。 JAAS をセキュリティー資格情報として指定すると、このプロパティーは認証別名によって指定変更されます。
グローバル化	はい
BIDI 対応	はい

XA DataSource 名 (XADataSourceName)

このプロパティーでは、XA (分散) トランザクションを実行するためにデータベース接続を確立するときに使用される XA データ・ソースの名前を指定します。

表 69. 「XA データ・ソース名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	DB2 データベースへの XA 接続を確立する場合は、このプロパティーが XADatabaseName プロパティーと組み合わせて使用されます。 Oracle データベースへの XA 接続を確立する場合、このプロパティーが使用されますが、XADatabaseName プロパティーは使用されません。

例	Oracle データベースの標準的な値:
	
	`oracle.jdbc.xa.client.OracleXADataSource`
	
	タイプ 2 JDBC ドライバー (db2java.zip) を備えた DB2 データベースの標準的な値:
	
	`COM.ibm.db2.jdbc.DB2XADataSource`
	
	タイプ 4 JDBC ドライバー (db2jcc.jar) を備えた DB2 データベースの標準的な値:
	
	`com.ibm.db2.jcc.DB2XADataSource`

グローバル化	いいえ
BIDI 対応	いいえ

XA データベース名 (XADatabaseName)

このプロパティーは、XA 接続に使用するデータベースの名前を指定します。

表 70. 「XA データベース名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	DB2 データベースへの XA 接続を確立するため、このプロパティーと XADataSourceName プロパティーが組み合わせて使用されます。Oracle データベースの場合は、このプロパティーは不要です。
グローバル化	はい
BIDI 対応	はい

接続に失敗した場合の最大再試行回数 (connectionRetryLimit)

このプロパティーでは、アダプターで Outbound 接続関連のエラーが発生したときに、アダプターが データベース への接続の再確立を試行する回数を指定します。

表 71. 「システム接続に失敗した場合の最大再試行回数」の詳細

必須	いいえ
使用可能な値	ゼロより大きいか等しい整数
デフォルト	0
プロパティー・タイプ	Integer
使用法	アダプターで Outbound 接続関連のエラーが発生すると、アダプターは、プロパティー (『接続の再試行間隔 (ミリ秒単位) (ConnectionRetryInterval)』) で指定された間隔で、このプロパティーで指定された回数だけ (物理接続が確立されるまで) 物理接続の再確立を試行します。 値が 0 に設定されている場合は、アダプターは データベース への接続を検証せずに、Outbound 操作を実行します。データベース 接続が無効な場合、Outbound 操作は失敗します。アダプターは、データベース への再接続を試行しません。 値が 0 より大きい値に設定されている場合は、各要求時に、アダプターは、データベース への接続がアクティブかどうかを検証します。 • 接続が有効である場合、操作が実行されます。 • 接続が無効な場合、アダプターは現在の管理接続を終了し、新しい管理接続 (新規物理接続) が作成されます。アダプターが データベース への接続の再確立に成功した場合は、Outbound 操作が実行されます。それ以外の場合は、アダプターは、指定された回数だけ再接続を試行した後、ResourceException を生成します。
グローバル化	いいえ
BIDI 対応	いいえ

接続の再試行間隔 (ミリ秒単位) (ConnectionRetryInterval)

このプロパティーでは、接続に失敗した場合に データベース への再接続を試行する時間間隔を指定します。

表 72. 「接続が失敗した場合の再試行間隔」の詳細

必須	いいえ
使用可能な値	ゼロより大きいか等しい整数
デフォルト	60000
計測単位	ミリ秒
プロパティー・タイプ	Integer

表 72. 「接続が失敗した場合の再試行間隔」の詳細 (続き)

使用法	このプロパティーでは、データベース への接続の確立中にアダプターでエラーが発生したときに、アダプターが接続の再確立を試行するまで待機する時間間隔を指定します。 デフォルトでは、このプロパティーは無効です。プロパティーの値 (336 ページの『接続に失敗した場合の最大再試行回数 (connectionRetryLimit)』) がゼロより大きい値に設定されている場合にのみ有効になります。
グローバル化	いいえ
BIDI 対応	いいえ

XA DataSource JNDI 名 (XADataSourceJNDIName)

このプロパティーは、データベースへの接続を確立するときに使用される XA データ・ソースの JNDI 名を指定します。

表 73. 「XA データ・ソース JNDI 名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、ターゲット・データベースの接続情報を指定する IBM Business Process Manager または WebSphere Enterprise Service Bus の XA データ・ソースの JNDI 名を指定します。『データベース接続情報 (ConnectionType)』 が「XADataSourceJNDI」に設定されていると、アダプターはこのプロパティーを使用して、データベースへの接続を確立します。 Outbound 操作のパフォーマンスを改善するには、準備済みステートメントのキャッシュ用に使用可能にされているデータ・ソースの名前を指定します。その他の有効な認証プロパティーも設定されている場合は、それらのプロパティーによってデータ・ソース内の認証プロパティーが指定変更されます。
グローバル化	はい
BIDI 対応	いいえ

データベース接続情報 (ConnectionType)

このプロパティーでは、アダプターがデータベースへの接続を確立する方法を指定します。

表 74. データベース接続情報

必須	はい
使用可能な値	XADataSourceJNDI、XAConnectionProps、PoolDataSourceJNDI、または LocalConnectionProps
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 74. データベース接続情報 (続き)

使用法	このプロパティーは、アダプターが実行時にデータベース接続を確立する方法を指定します。このプロパティーの有効な値は、以下のとおりです。 • XADataSourceJNDI - データベース接続が、定義済みの XA データ・ソースに対応する XADataSourceJNDIName プロパティーを使用して確立されることを示します。 • XAConnectionProps - XADataSourceName プロパティーと、DatabaseURL または XADatabaseName のプロパティーを使用してデータベース接続が確立されることを示します。 • PoolDataSourceJNDI - データベース接続が、定義済みのデータ・ソースに対応する poolDataSourceJNDIName プロパティーを使用して確立されることを示します。 • LocalConnectionProps - DatabaseURL プロパティーおよび JDBCDriverClass プロパティーを使用してデータベース接続が確立されることを示します。 新規アプリケーションの場合、このプロパティーは、外部のサービス・ウィザードによって自動的に設定されます。マイグレーションされたアプリケーションについては、このプロパティーは、マイグレーション・プロセス中に ManagedConnectionFactory プロパティーに従って設定されます。 • DataSourceJNDIName プロパティーが設定されている場合、このプロパティーの値は XADataSourceJNDI に設定されます。 • XADataSourceName が設定されている場合、このプロパティーの値は XAConnectionProps に設定されます。 • それ以外の場合、このプロパティーの値は LocalConnectionProps に設定されます。 このプロパティーが設定されない場合、アダプターは後方互換モードを使用して、データベース接続を確立します。後方互換性モードでは、データベース接続プロパティーは、次の順序で使用されます。 1. DataSourceJNDIName プロパティーが設定されている場合、アダプターはこのプロパティーを使用してデータベースへの接続を確立します。 2. DataSourceJNDIName プロパティーが設定されていないときに、XADataSourceName プロパティーと XADatabaseName プロパティーが設定されている場合、アダプターはそれらのプロパティーを使用して接続を確立します。DataSourceJNDIName プロパティーは、XA データ・ソースまたは接続プール・データ・ソースを表します。 XA トランザクションをサポートするサーバーで JNDI データ・ソースを定義し、その後アダプターの構成時にそのデータ・ソースを指定する場合は、XA トランザクションをサポートするすべてのタイプのデータベースに接続できます。XA データ・ソースとデータベースを使用する場合は、アダプターは DB2 および Oracle データベースでのみ XA トランザクションをサポートします。 3. DataSourceJNDIName、XADataSourceName、および XADatabaseName プロパティーが設定されていない場合、アダプターは DatabaseURL、JDBCDriverClass、 UserName、および Password プロパティーを使用して接続を確立します。
グローバル化	いいえ
BIDI 対応	いいえ

接続プール・データ・ソースの JNDI 名 (PoolDataSourceJNDIName)

データベースへの接続の確立に使用される接続プール・データ・ソースの JNDI 名。

表 75. 接続プール・データ・ソース

必須	いいえ
デフォルト	デフォルト値なし

表 75. 接続プール・データ・ソース (続き)

プロパティー・タイプ	String
使用法	このプロパティーを使用して、ターゲット・データベースの接続情報を指定する、IBM Business Process Manager または WebSphere Enterprise Service Bus 内の接続プール・データ・ソースの、JNDI 名を指定します。 337 ページの『データベース接続情報 (ConnectionType)』 のプロパティーが「PoolDataSourceJNDI」に設定されていると、アダプターはこのプロパティーを使用して、データベースへの接続を確立します。 Outbound 操作のパフォーマンスを改善するには、準備済みステートメントのキャッシュ用に使用可能にされているデータ・ソースの名前を指定します。その他の有効な認証プロパティーも設定されている場合は、それらのプロパティーによってデータ・ソース内の認証プロパティーが指定変更されます。
グローバル化	いいえ
BIDI 対応	いいえ

対話仕様プロパティー:

対話仕様プロパティー、ないし InteractionSpec は、操作の対話処理を制御します。アダプターの構成時には、外部サービス・ウィザードによって対話仕様プロパティーを設定します。一般に、このプロパティーを変更する必要はありません。ただし、Outbound 操作に関する一部のプロパティーはユーザーが変更できます。例えば、RetrieveAll 操作が返す情報が十分でない場合は、RetrieveAll によって返されるレコードの最大数を指定する対話仕様プロパティーの値を大きくすることができます。アプリケーションのデプロイ後にこれらのプロパティーを変更する場合、IBM Integration Designer のアセンブリー・エディターを使用します。これらのプロパティーは、インポートのメソッド・バインディングの中にあります。

表 76 に、ユーザーが設定する対話仕様プロパティーをリストし、説明します。後続セクションのプロパティー詳細表の見方について詳しくは、 308 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 76. WebSphere Adapter for JDBC 用の対話仕様プロパティー

プロパティー名		**説明**
ウィザード内	**対話仕様クラス**	
340 ページの『パターン・ストリング 』	customFaultPattern	フォールト・パターンを指定します
341 ページの『カスタム・フォールト名』	customFaultName	対応するフォールト名を指定します

表 77. WebSphere Adapter for JDBC での RetrieveAll 操作用の対話仕様プロパティー

プロパティー名		**説明**
ウィザード内	**対話仕様クラス**	
341 ページの『返されるレコードの最大数』	maxRecords	RetrieveAll 操作で返すレコードの最大数

表 77. WebSphere Adapter for JDBC での RetrieveAll 操作用の対話仕様プロパティー (続き)

プロパティー名		説明
ウィザード内	**対話仕様クラス**	
なし	342 ページの『enablePaging』	RetrieveAll 操作で開始索引からレコードを返すかどうかを指定します
343 ページの『レコードを返す開始索引』	startIndex	RetrieveAll 操作でレコードを返す開始索引
343 ページの『ページ当たりのレコード数』	pageSize	RetrieveAll 操作で開始索引から返されるレコードの数

表 78. WebSphere Adapter for JDBC での UpdateAll 操作および DeleteAll 操作用の対話仕様プロパティー

プロパティー名		説明
ウィザード内	**対話仕様クラス**	
344 ページの『UpdateAll または DeleteAll 操作中に影響を受けるレコードがない場合は例外を返す』	errorOnNoRecordAffect	UpdateAll または DeleteAll 操作中にデータベース内のレコードが影響を受けない場合に例外を返すかどうかを指定します

表 79. WebSphere Adapter for JDBC での BatchCreate、BatchUpdate、および BatchDelete の各操作用の対話仕様プロパティー

プロパティー名		説明
ウィザード内	**対話仕様クラス**	
N/A	344 ページの『batchSize』	1 回のバッチ対話でのビジネス・オブジェクトの最大数を指定します
N/A	345 ページの『skipErrorsInBatch』	アダプターが BatchCreate、BatchUpdate、または BatchDelete の各操作を実行するときにエラーをどのように扱うのかを指示します
N/A	345 ページの『returnBOsInBatch』	バッチでビジネス・オブジェクトが戻されるかどうかを指示します。

パターン・ストリング

このプロパティーは、カスタム・フォールト・パターンを指定します。

表 80. パターン・ストリング

必須	いいえ
デフォルト	デフォルト値なし
使用法	正規表現を入力して、フォールト・パターンを定義します。例えば、「.*Record is locked.*」です。実行時に、アダプターはユーザーが定義した正規表現に基づいてカスタム・フォールトを返します。
プロパティー・タイプ	String

表 80. パターン・ストリング (続き)

グローバル化	いいえ
BIDI 対応	いいえ

カスタム・フォールト名

このプロパティーは、対応するフォールト名を指定します。

表 81. カスタム・フォールト名

必須	いいえ
デフォルト	`False`
使用法	.xsd ファイルを生成するのに使用されるフォールト名を指定します。カスタム・フォールト・パターン・ストリングが、事前定義されたフォールト例外メッセージに一致すると、事前定義されたフォールトに優先して、カスタム・フォールトが戻されます。
プロパティー・タイプ	Boolean
グローバル化	いいえ
BIDI 対応	いいえ

返されるレコードの最大数

このプロパティーは、RetrieveAll 操作で返されるレコードの最大数を指定します。

表 82. 返されるレコードの最大数の詳細

必須	いいえ
デフォルト	100

表 82. 返されるレコードの最大数の詳細 (続き)

使用法	このプロパティーを使用して、RetrieveAll 操作から返されるレコードの数を制御します。 • 値が -1 の場合、RetrieveAll 操作は、照会と一致するすべてのレコードを返します。「**すべてのレコードを返す**」を選択している場合、このプロパティーの値は、内部的に -1 に設定されます。 **注:** このプロパティーの値が -1 に設定されている場合、「**レコードを返す開始索引**」プロパティーに指定された値に関係なく、実行時にレコードのページングは有効になりません。 • -1 以外のゼロ以下の値の場合、アダプターは InvalidMaxRecords メッセージを伴う ResourceException を返します。 • ゼロより大きい値の場合、データベース中の一致数がこのプロパティーの値を超えると、アダプターは MatchesExceededLimitException を返します。RetrieveAll 操作ですべてのレコードが返されない場合は、この値を大きくします。例えば、値を 50 に設定し、テーブルに 100 レコードが含まれている場合、アダプターは例外 MatchesExceededLimitException を返します。 • 値がゼロより大きく、かつデータベース内の一致数がこのプロパティーの値より小さい場合、RetrieveAll 操作はすべてのレコードを返します。例えば、値を 50 に設定したときに、テーブル内のレコード数が 25 の場合、RetrieveAll 操作は 25 のレコードをすべて返します。
プロパティー・タイプ	Integer
グローバル化	いいえ
BIDI 対応	いいえ

enablePaging

このプロパティーは、RetrieveAll 操作で、開始索引から始めてレコードを返すかどうかを指定します

表 83. RetrieveAll 操作でのレコードのページングを許可の詳細

必須	いいえ
デフォルト	`False`

表 83. RetrieveAll 操作でのレコードのページングを許可の詳細 (続き)

使用法	このプロパティーを `True` に設定すると、アダプターは指定された開始索引からレコードを返します。指定した最大レコード数の値を超える数のレコードがデータベースから取り出された場合、指定範囲内のレコードのみが返され、例外は生成されません。 **注:** このプロパティーを `True` に設定すると、アダプターは RetrieveAll 操作用に照会を最適化します。ただし、アダプターは、RetrieveAllSP を使用して構成されたテーブル・ビジネス・オブジェクトでの RetrieveAll 操作用の照会を最適化することはできません。これを行うと RetrieveAll 操作のパフォーマンスが低下するためです。したがって、テーブルに対する RetrieveAllSP を構成し、このプロパティーを `True` に設定することはお勧めしません。 このプロパティーが `False` の場合、アダプターは、「**返されるレコードの最大数**」プロパティーに指定された最大数のレコードを返します。ただし、指定された最大レコード数の値を超える数のレコードがデータベースから取り出された場合、アダプターは MatchesExceededLimitException を戻します。
プロパティー・タイプ	Boolean
グローバル化	いいえ
BIDI 対応	いいえ

レコードを返す開始索引

このプロパティーは、RetrieveAll 操作で返すレコードの開始索引を指定します。

表 84. レコードを返す開始索引の詳細

必須	いいえ
デフォルト	0
使用法	例: 「**ページ当たりのレコード数**」プロパティーが 50 に設定され、「**レコードを返す開始索引**」プロパティーが 20 に設定されている場合、アダプターは 21 番目の行から開始して 70 番目の行までのすべてのレコードを返します。 値が 0 より小さい場合、アダプターは例外を戻します。
プロパティー・タイプ	Integer
グローバル化	いいえ
BIDI 対応	いいえ

ページ当たりのレコード数

このプロパティーは、RetrieveAll 操作で開始索引から返されるレコードの数を指定します。

表 85. ページ当たりのレコード数の詳細

必須	いいえ
デフォルト	100

表 85. ページ当たりのレコード数の詳細 (続き)

使用法	例: 「**ページ当たりのレコード数**」プロパティーが 50 に設定され、「**レコードを返す開始索引**」プロパティーが 20 に設定されている場合、アダプターは 21 番目の行から開始して 70 番目の行までのすべてのレコードを返します。 値が 1 より小さい場合、アダプターは例外を戻します。
プロパティー・タイプ	Integer
グローバル化	いいえ
BIDI 対応	いいえ

UpdateAll または DeleteAll 操作中に影響を受けるレコードがない場合は例外を返す

このプロパティーは、UpdateAll または DeleteAll 操作中にデータベース内のレコードが影響を受けない場合に例外を返すかどうかを指定します。

表 86. *UpdateAll または DeleteAll 操作中に影響を受けるレコードがない場合は例外を返す*

必須	いいえ
デフォルト	`False`
使用法	このプロパティーを `True` に設定すると、アダプターは、UpdateAll 操作または DeleteAll 操作中にデータベース内のレコードが影響を受けなければ RecordNotFoundException を返します。 **注:** このプロパティーは、UpdateAll および DeleteAll 操作に適用されます。
プロパティー・タイプ	Boolean
グローバル化	いいえ
BIDI 対応	いいえ

batchSize

このプロパティーは、1 回のバッチ対話でのビジネス・オブジェクトの最大数を指定します。

表 87. *バッチ・サイズの詳細*

必須	いいえ
デフォルト	100
使用法	このプロパティーは、1 回のバッチ対話でのビジネス・オブジェクトの最大数を指定します。このプロパティーの最適値は、データベースおよび環境によって異なります。場合によっては、あまり高い値にすると悪影響が出ることがあります。環境に合わせてパフォーマンスを調整するには、各データベースの資料を参照するか、別の値を試してみてください。
プロパティー・タイプ	Integer
グローバル化	いいえ
BIDI 対応	いいえ

skipErrorsInBatch

このプロパティーは、アダプターが BatchCreate、BatchUpdate、または BatchDelete の各操作を実行するときにエラーをどのように扱うのかを指示します。

表 88. 「バッチでのエラーをスキップする」の詳細

必須	いいえ
デフォルト	`True`
使用法	• skipErrorsInBatch プロパティーが `True` に設定されている場合、個別ビジネス・オブジェクトの処理中に例外が発生すると、アダプターはエラー・メッセージを、対応するエラー・メッセージ項目に設定します。その後、アダプターは残りのビジネス・オブジェクトの処理を続行します。戻される結果を分析して、例外が起こったビジネス・オブジェクトに関して必要なアクションを実行することができます。 • skipErrorsInBatch が `false` に設定されている場合、個別ビジネス・オブジェクトの処理中に例外が発生すると、最初の例外がスローされ、returnBOsInBatch オプションに値を設定していても、結果のビジネス・オブジェクト・リストは戻されません。
プロパティー・タイプ	Boolean
グローバル化	いいえ
BIDI 対応	いいえ

returnBOsInBatch

このプロパティーは、バッチでビジネス・オブジェクトが戻されるかどうかを指示します。

表 89. 「バッチでビジネス・オブジェクトを戻す」の詳細

必須	いいえ
デフォルト	`False`
使用法	• returnBOsInBatch が `True` に設定されている場合、バッチ操作は、操作が完了した後、結果ビジネス・オブジェクトを戻します。入力ビジネス・オブジェクトのプロパティーに構成されている一連の UID または CopyAttributes ASI がある場合、更新されたビジネス・オブジェクトが戻されます。 • returnBOsInBatch が `False` に設定されている場合、操作が完了した後、結果ビジネス・オブジェクトは戻されません。 **注:** skipErrorsInBatch が `False` に設定されている場合、個別ビジネス・オブジェクトの処理中に例外が発生すると、returnBOsInBatch オプションに値を設定していても、結果のビジネス・オブジェクト・リストは戻されません。
プロパティー・タイプ	Boolean
グローバル化	いいえ
BIDI 対応	いいえ

Inbound 構成プロパティー

WebSphere Adapter for JDBC には、オブジェクトやサービスを生成したり作成したりするときに、外部サービス・ウィザードを使用して設定する、いくつかの種類の Inbound 接続構成プロパティーがあります。リソース・アダプターおよび活動化仕様のプロパティーは、モジュールをデプロイした後に IBM Integration Designer 管理コンソールまたは 管理コンソール を使用して変更できますが、外部サービス・ウィザードの接続プロパティーは、デプロイメント後に変更することはできません。

プロパティーの詳細についてのガイド:

WebSphere Adapter for JDBC を構成するときに使用されるプロパティーは、リソース・アダプター・プロパティーや管理接続ファクトリー・プロパティーなど、それぞれの構成プロパティーのトピックに記載されている表で詳細に説明されています。 これらの表を使用しやすくするため、参照する各行の情報を以下に説明します。

次の表では、構成プロパティーの表に表示される場合がある各行の意味を説明します。

行	説明
必須	アダプターが動作するためには、必須フィールド (プロパティー) に値が必要です。必須プロパティーに対しては、外部サービス・ウィザードがデフォルト値を提供する場合があります。 外部サービス・ウィザードの*必須フィールドからデフォルト値を除去*しても、*デフォルト値は変更されません*。必須フィールドに値がまったく入っていない場合、外部サービス・ウィザード はそのフィールドに割り当てられたデフォルト値を使用してフィールドを処理し、そのデフォルト値は管理コンソールに表示されます。 可能な値は「**はい**」および「**いいえ**」です。 プロパティーは、他のプロパティーが特定の値の場合のみ必須となることがあります。その場合は、表にこの依存関係が記載されます。以下に例を示します。 • EventQueryType プロパティーが Dynamic に設定された場合は「はい」 • Oracle データベースの場合は「はい」
使用可能な値	プロパティーで選択可能な値をリストして説明します。
デフォルト	外部サービス・ウィザードによって設定される事前定義値。プロパティーが必須の場合は、デフォルト値を受け入れるか、ユーザーが値を指定する必要があります。プロパティーにデフォルト値がない場合、表には「デフォルト値なし」と記載されます。 `None` という語は、受け入れ可能なデフォルト値です。デフォルト値がないという意味ではありません。
計測単位	プロパティーの計測単位を指定します (例: キロバイト、秒)。

行	説明
プロパティー・タイプ	プロパティー・タイプを示します。有効なプロパティー・タイプは以下のとおりです。 • Boolean • String • Integer
使用法	プロパティーに適用される場合がある使用の条件または制限について記述します。制限の記載例を以下に示します。 Rational Application Developer for WebSphere Software バージョン 6.40 またはそれ以前では、パスワードに以下の制限があります。 • 大文字である必要があります • 長さが 8 文字である必要があります Rational Application Developer for WebSphere Software バージョン 6.40 よりも後のバージョンでは、パスワードの制限が以下のように変更されました。 • 大文字小文字を区別しません • 長さが 40 文字まで可能です このセクションでは、このプロパティーに影響を及ぼす他のプロパティー、またはこのプロパティーによって影響を受けるプロパティーをリストし、その条件付き関係の内容を説明します。
例	次のようなサンプル・プロパティー値が示されます。 「言語が JA (日本語) に設定された場合、コード・ページ番号は `8000` に設定されます。」
グローバル化	グローバル化される場合、プロパティーには各国語サポートがあるので、自国の言語に設定できます。 有効な値は「**はい**」および「**いいえ**」です。
BIDI 対応	プロパティーが双方向 (bidi) 処理でサポートされているかどうかを示します。双方向処理とは、同一ファイルに右から左 (ヘブライ語やアラビア語など) と左から右 (URL やファイル・パスなど) の両方の意味内容を含むデータを処理するタスクを指します。 有効な値は「**はい**」および「**いいえ**」です。

ウィザードの接続プロパティー:

外部サービス接続プロパティーは、外部サービス・ウィザード (ビジネス・オブジェクト作成ツール) とデータベース間の接続を確立するために使用されます。これらのプロパティーにより、接続構成、双方向変換プロパティー、およびウィザードのログ記録オプションなどが指定されます。接続の確立後に、ウィザードは、ビジネス・オブジェクトの作成に必要なメタデータをデータベース内でディスカバーできます。

データベース内のオブジェクトをディスカバーするためにウィザードで指定したプロパティーの一部は、ウィザードで後で指定するランタイム・プロパティーの初期値として使用されます。これらには、リソース・アダプター、管理接続ファクトリー、および活動化仕様のプロパティーが含まれます。

外部サービス・ウィザードの接続プロパティーとその目的を以下の表に示します。各プロパティーの完全な説明は、表に続くセクションで説明します。後続セクションのプロパティー詳細表の見方について詳しくは、 308 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 90. 外部サービス・ウィザードの接続プロパティー

ウィザードのプロパティー名	説明
追加の JDBC ドライバー接続プロパティー	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
データベース	データベースの名前を指定します。
データベース・ソフトウェア	アダプターが使用するデータベース管理ソフトウェアの名前およびバージョン
データベース URL	データベースへの接続に使用されるデータベース URL
ホスト名	データベース・サーバーのホスト名または IP アドレス。
JDBC ドライバー・クラス名	JDBC ドライバー・クラスの名前
JDBC ドライバー・タイプ	使用する JDBC ドライバーのタイプ
パスワード	対応するユーザー名のパスワード
ポート番号	データベース・インスタンスに接続する際のポート番号。
ビジネス・オブジェクト名のプレフィックス	ビジネス・オブジェクト名に追加されるプレフィックス
サーバー名	アダプターの接続先の Informix データベース・サーバーの名前
データベース接続時に自動コミットを設定	基礎となるデータベース接続が自動コミット・モードかどうかを指定します。
ユーザー名	データベース接続に使用するデータベース・ユーザー名を指定します。

外部サービス・ウィザードは、双方向接続プロパティーを使用して、エンタープライズ情報システムに渡すデータに適切な双方向変換を適用します。

追加の JDBC ドライバー接続プロパティー

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 91. 追加の JDBC ドライバー接続プロパティーの詳細

行	説明
必須	いいえ

表 91. 追加の JDBC ドライバー接続プロパティーの詳細 (続き)

行	説明
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これらの接続プロパティーを、UserName および Password プロパティーと共に使用して、アダプターが使用するデータベース接続をカスタマイズします。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。

例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。
	
	`loginTimeout:20;readOnly:true;`
	`securityMechanism:USER_ONLY_SECURITY`
	
	高可用性環境では、High Availability and Disaster Recovery (HADR) データベースへの信頼できる接続のために DB2 ドライバーが必要とするプロパティー (retryIntervalForClientReroute、 maxRetriesForClientReroute、clientRerouteAlternateServerName、clientRerouteAlternatePortNumber) を取得するため、正しいプロパティー・ストリングが生成されるようこのプロパティーを構成する必要があります。以下に例を示します。
	
	`retryIntervalForClientReroute:15;`
	`maxRetriesForClientReroute:5;`
	`clientRerouteAlternateServerName:WLOXS01B.svl.ibm.com;`
	`clientRerouteAlternatePortNumber:50000`
	
	WebSphere 高可用性環境での WebSphere Adapter の使用について詳しくは、 https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=sw-app&S_PKG=wpswasadapt&S_TACT=105AGX10&S_CMP=LC を参照してください。

グローバル化	はい
BIDI 対応	いいえ

データベース

このプロパティーは、データベースの名前を指定します。

表 92. 「データベース名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値はデータベースによって異なります。
プロパティー・タイプ	String
使用法	これは、アクセスするデータベースの名前です。Oracle データベースの場合、これはデータベースを識別するシステム ID (SID) です。

表 92. 「データベース名」の詳細 (続き)

行	説明
グローバル化	はい
BIDI 対応	はい

データベース・ソフトウェア

このプロパティーは、アダプターがアクセスするデータベースを管理するデータベース管理ソフトウェアを指定します。

表 93. 「データベース・ソフトウェア」の詳細

行	説明
必須	はい
使用可能な値	このプロパティーは、一般的なデータベース・ソフトウェアを名前およびバージョン番号ごとにリストします。ご使用のソフトウェアがリストされていない場合は、「汎用 JDBC (Generic JDBC)」を選択してください。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードは、このプロパティーの値を使用して他のプロパティーのデフォルト値を設定し、データベース固有の選択リストを生成します。例えば、「DB2 USB Version 9.1」を選択すると、ウィザードの JDBC ドライバー・クラス・フィールドには、そのバージョンの DB2 UDB によってサポートされる JDBC ドライバーのみが表示されます。「Oracle 10」を選択すると、異なる JDBC ドライバーのセットが表示されます。
グローバル化	はい
BIDI 対応	はい

データベース URL

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 94. 「データベース URL」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これは、使用するデータベース・ソフトウェアと JDBC ドライバーに固有の値です。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。IP アドレスを大括弧 ([]) で囲んでください。

表 94. 「データベース URL」の詳細 (続き)

行	説明
例	一般的なデータベース・サーバーの標準的な値を以下に示します。 **DB2 Universal (タイプ 4) JDBC ドライバー** jdbc:db2://<Host_Name>/DB **DB2 Universal JDBC ドライバー (IPv6 アドレス)** jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB **DB2 Universal Database タイプ 2 ドライバー (ローカル接続用)** jdbc:db2:TEST **DB2 Universal Database タイプ 2 ドライバー (リモート接続用)** jdbc:db2://<Host_Name>/TEST **Informix V10** jdbc:informix-sql://<Host_Name>/ symaster:INFORMIXSERVER=server **Oracle V10** jdbc:oracle:thin:@9.26.248.148:1521:dev **Derby JDBC ドライバー (非リモート)** jdbc:derby://<runtime home>/runtimes/bi_v6/derby/ databases/JDBCTEST z/OS のリモート・テスト環境を使用している場合、Derby データベース URL に次の値を使用します。 **Derby JDBC ドライバー (リモート z/OS テスト環境)** jdbc:db2j:net://<HOST_NAME>:1527// <remote_derbydb_path>/JDBCTEST</ remote_derbydb_path></HOST_NAME>
グローバル化	はい
BIDI 対応	はい

ホスト名

このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。

表 95. 「ホスト名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。データベース・サーバーにより IPv6 がサポートされている場合は、ホスト名を IPv6 形式で指定できます。
グローバル化	はい
BIDI 対応	はい

JDBC ドライバー・クラス名

このプロパティーは、JDBC ドライバー・クラスの名前を指定します。

表 96. 「JDBC ドライバー・クラス名」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。
プロパティー・タイプ	String
使用法	ウィザードには、選択した JDBC ドライバー・タイプのデフォルト・クラス名が表示されますが、必要に応じて別のクラス名を入力できます。JDBC ドライバーの値として「**その他**」を選択すると、デフォルトは提供されず、クラス名の入力が必要になります。クラス名は、ウィザードの開始時に指定した JDBC ドライバー・ファイルに記述されている必要があります。
グローバル化	はい
BIDI 対応	いいえ

JDBC ドライバー・タイプ

このプロパティーは、使用する JDBC ドライバーのタイプを指定します。

表 97. 「JDBC ドライバー・タイプ」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。
プロパティー・タイプ	String
使用法	これは、使用する JDBC ドライバーのタイプです。根本的な点は使用するドライバーがタイプ 2 とタイプ 4 (ユニバーサル) のどちらであるかですが、各データベース・システムでは、ドライバーにデータベース・システム固有の名前が使用されています。各データベース・システムの既知のドライバーのリストがウィザードに表示されます。使用するドライバーがリストにない場合は、「その他」を選択します。このフィールドの情報は、ウィザードの開始時に指定した JDBC ドライバー・ファイルと一致している必要があります。
グローバル化	はい
BIDI 対応	いいえ

パスワード (Password)

対応するユーザー名のパスワード

表 98. パスワードの詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	オブジェクトをディスカバーする目的でデータベースに接続する際に入力されたユーザー名に関連付けられたパスワード。
グローバル化	はい
BIDI 対応	はい

ポート番号

このプロパティーは、データベース・インスタンスのポート番号を指定します。

表 99. ポート番号の詳細

行	説明
必須	はい
デフォルト	デフォルト値はデータベースによって異なります。また、JDBC ドライバー・タイプに特定のドライバーを選択した場合、ウィザードによってデフォルト値があらかじめ設定されます。
プロパティー・タイプ	String
使用法	これは、データベース・インスタンスへ接続するポートのポート番号です。 JDBC ドライバー・タイプに、その他を選択する場合は、このプロパティーは使用できません。
グローバル化	はい
BIDI 対応	いいえ

ビジネス・オブジェクト名のプレフィックス

ビジネス・オブジェクトの名前に追加されるプレフィックス。

表 100. 「プレフィックス」の詳細

行	説明
必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	プレフィックスを使用して、ビジネス・オブジェクトのタイプを容易に区別できるようにします。
例	汎用ビジネス・オブジェクトにプレフィックス `JDBC` を指定し、アプリケーション固有のビジネス・オブジェクトに `%AppName%` を指定する場合があります。

表 100. 「プレフィックス」の詳細 (続き)

行	説明
グローバル化	はい
BIDI 対応	いいえ

サーバー名

アダプターの接続先 Informix データベース・サーバーのデフォルトの名前を指定します。

表 101. 「サーバー名」の詳細

行	説明
必須	はい
デフォルト	サーバー
プロパティー・タイプ	String
使用法	サーバー名の値は、ローカル・サーバーでもリモート・サーバーでも構いませんが、アプリケーションを実行しているコンピューターの `$INFORMIXDIR/etc/sqlhosts` ファイルにある有効な `dbservername` 項目に対応している必要があります。 `dbservername` は、小文字で始まらなければなりません。また、128 バイトを超過してはいけません。大文字、フィールド区切り文字 (空白スペースまたはタブ)、改行文字、およびハイフン (マイナス) 記号を除く任意の印刷可能文字を含むことができます。
グローバル化	いいえ
BIDI 対応	いいえ

データベース接続時に自動コミットを設定

基礎となるデータベース接続が自動コミット・モードかどうかを指定します。ストアード・プロシージャーのトランザクション・モードについての Sybase データベースのデフォルト設定を指定変更します。

表 102. 「データベース接続時に自動コミットを設定」の詳細

行	説明
必須	はい (特定の JDBC ドライバーの場合、および特定のストアード・プロシージャー構成の場合)。
デフォルト	False
プロパティー・タイプ	Boolean

表 102. 「データベース接続時に自動コミットを設定」の詳細 (続き)

行	説明
使用法	この値を設定して、基礎となるデータベース接続が自動コミット・モードかどうかを指定します。このプロパティーを True に設定した場合、基礎となるデータベースに対する変更は、ウィザードの完了後はロールバックしません。 Sybase データベースで JConnect ドライバーを使用してディスカバリーを実行している場合で、なおかつ Sybase データベース上のストアード・プロシージャーのトランザクション・モードの設定が「非チェーン・モード」または「Transact-SQL モード」になっている場合、このプロパティーの設定は必須です。 このプロパティーを True に設定することで、アダプターが結果セットのディスカバリーをできない Sybase トランザクション・モードの構成を指定変更します。
グローバル化	はい
BIDI 対応	いいえ

ユーザー名 (UserName)

このプロパティーは、データベース接続に使用するデータベース・ユーザー名を指定します。

表 103. 「ユーザー名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	ユーザー名は、オブジェクトをディスカバーする目的でデータベースに接続する場合に入力する名前です。
グローバル化	はい
BIDI 対応	はい

リソース・アダプター・プロパティー:

リソース・アダプター・プロパティーは、アダプターの一般的な操作 (ビジネス・オブジェクトの名前空間の指定など) を制御します。リソース・アダプター・プロパティーは、アダプターの構成時に外部サービス・ウィザードを使用して設定します。アダプターのデプロイ後に、これらのプロパティーを変更するには、管理コンソールを使用します。

ロギングおよびトレースに関する次のプロパティーは、非推奨になっています。

- ログ・ファイル最大サイズ
- ログ・ファイル名
- ログ・ファイル数
- トレース・ファイル最大サイズ

- トレース・ファイル名
- トレース・ファイル数

以下のリソース・アダプター・プロパティーは、 Inbound 処理用の活動化仕様または Outbound 処理用の管理接続ファクトリーに同じプロパティーが設定されている場合には指定変更されます。

- DatabaseVendor
- ping 照会
- 照会タイムアウト
- ReturnDummyBOForSP

BONameSpaceプロパティーは、活動化仕様プロパティーに移動しました。

以下の表に、リソース・アダプター・プロパティーとその目的のリストを示します。各プロパティーの完全な説明は、表に続くセクションで説明します。プロパティー詳細表の見方について詳しくは、 308 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 104. Adapter for JDBC 用のリソース・アダプター・プロパティー

プロパティー名		説明
ウィザード内	**管理コンソール内**	
アダプター ID	AdapterID	PMI イベントのアダプター・インスタンス、ロギングおよびトレースのアダプター・インスタンスを識別する場合に使用します。
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する	HideConfidentialTrace	ログおよびトレース・ファイルにユーザー・データではなく X ストリングを書き込み、潜在的な機密情報を隠すようにするかどうかを指定します。
照会タイムアウト (秒)	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
359 ページの『高可用性 サポートを使用可能にする (enableHASupport)』	enableHASupport	WebSphere Adapter for JDBC の構成モード (Active-Active または Active-Passive) を指定します。
(なし)	LogFileSize	非推奨
(なし)	ログ・ファイル名	非推奨
(なし)	ログ・ファイル数	非推奨
接続を検証するための SQL 照会	PingQuery	データベースへの接続の信頼性をテストするのに使用する SQL 照会
(なし)	TraceFileSize	非推奨
(なし)	トレース・ファイル名	非推奨
(なし)	トレース・ファイル数	非推奨

アダプター ID (AdapterID)

このプロパティーは、アダプターの特定のデプロイメントまたはインスタンスを識別します。

表 105. 「アダプター ID」の詳細

必須	はい
デフォルト	001
プロパティー・タイプ	String
使用法	このプロパティーは、ログおよびトレース・ファイル内のアダプター・インスタンスを識別します。また、アダプターのモニター時にアダプター・インスタンスを識別する場合に役立ちます。アダプター ID は、アダプター固有の ID、JDBCRA と共に使用され、 Log and Trace Analyzer ツールによって使用されるコンポーネント名を構成します。例えば、アダプター ID プロパティーが、001 に設定されている場合、コンポーネント ID は、JDBCRA001 となります。 同じアダプターの複数のインスタンスを実行する場合、アダプター ID プロパティーの最初の 7 文字は、必ずインスタンスごとに固有のものにし、ログおよびトレース情報を特定のアダプター・インスタンスに相互に関連付けられるようにしてください。アダプター ID プロパティーの最初の 7 文字を固有のものにすることにより、そのアダプターの複数インスタンスのコンポーネント ID も固有のものになり、アダプターの特定インスタンスにログおよびトレース情報を相互に関連付けることができるようになります。 例えば、WebSphere Adapter for JDBC の 2 つのインスタンスのアダプター ID プロパティーを 001 および 002 に設定するとします。これらのインスタンスのコンポーネント ID、JDBCRA001 および JDBCRA002 は、短いので固有性を保つことができ、別のアダプター・インスタンスとして区別することができます。しかし、もっと長いアダプター ID プロパティーのインスタンスの場合、互いを区別できなくなります。2 つのインスタンスのアダプター ID プロパティーを Instance01 と Instance02 に設定した場合、各アダプター・インスタンスのログおよびトレース情報を調べることはできなくなります。これは、両方のインスタンスのコンポーネント ID が JDBCRAInstanc に切り捨てられるためです。 Inbound 処理の場合、このプロパティーの値は、リソース・アダプター・レベルで設定されます。Outbound 処理の場合、この値は、リソース・アダプター・レベルと管理接続ファクトリー・レベルの両方で設定できます。外部サービス・ウィザードを使用してアダプターを Outbound 処理用に構成した後、リソース・アダプター・プロパティーおよび管理接続ファクトリー・プロパティーを個別に設定できます。IBM Integration Designer アセンブリー・エディターまたは管理コンソールを使用してこれらのプロパティーを再設定する場合は、ログおよびトレース・エントリーのマーキングが不整合にならないように、矛盾がない設定になっていることを確認してください。
グローバル化	はい
BIDI 対応	いいえ

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 106. 「データベース・ベンダー」の詳細

必須	はい
使用可能な値	DB2 Informix MSSQLServer Oracle Others
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 その他のデータベースの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述する (HideConfidentialTrace)

このプロパティーは、ログおよびトレース・ファイル中のユーザー・データを「X」のストリングに置換し、潜在的な機密データが許可なく外部に漏れないようにします。

表 107. ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述するの詳細

必須	いいえ
使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean

表 107. ログ・ファイルおよびトレース・ファイルで、ユーザー・データを「XXX」と記述するの詳細 (続き)

使用法	このプロパティーを True に設定すると、アダプターでは、ログおよびトレース・ファイルに書き込む時に、ユーザー・データを「X」のストリングに置換します。 Inbound 処理の場合、このプロパティーの値は、リソース・アダプター・レベルで設定されます。Outbound 処理の場合、この値は、リソース・アダプター・レベルと管理接続ファクトリー・レベルの両方で設定できます。外部サービス・ウィザードを使用してアダプターを Outbound 処理用に構成した後、リソース・アダプター・プロパティーおよび管理接続ファクトリー・プロパティーを個別に設定できます。IBM Integration Designer アセンブリー・エディターまたは管理コンソールを使用してこれらのプロパティーを再設定する場合は、ログおよびトレース・エントリーのマーキングが不整合にならないように、矛盾がない設定になっていることを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

高可用性 サポートを使用可能にする (enableHASupport)

このプロパティーは、 WebSphere Adapter for JDBC の構成モード (Active-Active または Active-Passive) を指定するために使用します。

表 108. 「高可用性 サポートを使用可能にする」プロパティーの特性

必須	いいえ
使用可能な値	True False
デフォルト	True (管理コンソールの enableHASupport プロパティーの値が true に設定されており、アダプターが Active-Passive モードであることを示します。)
プロパティー・タイプ	Boolean
使用法	このプロパティーを false に設定すると、アダプターはクラスター環境で Active-Active モードになります。これにより、複数のアダプター・インスタンスが異なるサーバー・ノードでアクティブになります。各アダプター・インスタンスが別々のイベントを並行して処理します。その結果、各アダプター・インスタンスはそれぞれ別の固有イベントをポーリングし、イベントは重複することなくエンドポイントに送達されます。
グローバル化	いいえ
BIDI 対応	いいえ

照会タイムアウト (秒) (QueryTimeOut)

このプロパティーは、1 つの照会ですべての SQL ステートメントの実行に費やすことのできる最大時間を秒数で指定します。

表 109. 「照会タイムアウト」の詳細

必須	いいえ

表 109. 「照会タイムアウト」の詳細 (続き)

デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会を処理する際に、指定された秒数より長い時間がかかると、キャプチャーされた SQL 例外がデータベースから戻されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	いいえ
BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 110. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean
使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが `False` に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP プロパティーが `True` に設定されている場合は、ダミーのビジネス・オブジェクトが作成され、ストアード・プロシージャーから返されるパラメーター (出力パラメーターと入出力パラメーターを含む) が、対応する属性に取り込まれます。
グローバル化	いいえ
BIDI 対応	いいえ

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の信頼性をテストするために使用される SQL 照会を指定します。

表 111. 「ping 照会」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし
使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。 アダプターは、Outbound 操作の実行時に SQLException 例外を受け取るたびに、ping 照会を実行します。 アダプターは、接続のリカバリーを試行しません。 ping 照会により、データベースの接続が有効でなくなったことがわかると、アダプターはコンテナーに通知します。失効した接続のプールからの削除は、接続プール・マネージャーが行います。こうすることで、後続の Outbound 要求の処理が可能になります。
グローバル化	いいえ
BIDI 対応	いいえ

活動化仕様プロパティー:

活動化仕様プロパティーは、エクスポート用の Inbound イベント処理の構成情報を保持するプロパティーです。

活動化仕様プロパティーは、アダプターの構成時に外部サービス・ウィザードを使用して設定します。また、IBM Integration Designer アセンブリー・エディターを使用して、またはデプロイメント後に IBM Business Process Manager または WebSphere Enterprise Service Bus の管理コンソールを使用して変更できます。

以下の表に、活動化仕様プロパティーとその説明を示します。各プロパティーの完全な説明は、表に続くセクションで説明します。後続セクションのプロパティー詳細表の見方について詳しくは、 308 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 112. IBM WebSphere Adapter for JDBC のアクティブ化仕様プロパティー

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
イベント・フィルター用のアダプター・インスタンス	AdapterInstanceEventFilter	このアダプター・インスタンスがイベント・ストア内の特定のイベントを処理するかどうかを決定する ID
追加の JDBC ドライバー接続プロパティー	JDBCDriverConnectionProperties	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
ビジネス・オブジェクト名前空間	BONameSpace	ビジネス・オブジェクト定義の名前空間

表 112. IBM WebSphere Adapter for JDBC のアクティブ化仕様プロパティー (続き)

プロパティー名		説明
ウィザード内	管理コンソール内	
ユーザー定義削除照会	CustomDeleteQuery	各イベントの処理後に実行され、イベントの送達後に削除可能なレコードを削除する照会、ストアード・プロシージャー、またはストアード関数の名前
ユーザー定義イベント照会	CustomEventQuery	イベントのポーリングを実行する照会、ストアード・プロシージャー、またはストアード関数の名前
ユーザー定義の更新照会	CustomUpdateQuery	各イベントの処理後に実行され、後続のイベント・サイクルで処理対象としてイベントが選択されることを防ぐ照会、ストアード・プロシージャー、またはストアード関数の名前
失敗したイベント送達のためのユーザー定義の更新照会	CustomUpdateQueryForFailedEvent	イベントが正常に送達されなかった場合に実行される、照会、ストアード・プロシージャー、またはストアード関数の名前
データベース接続情報	ConnectionType	アダプターがデータベースへの接続を確立する方法を指定します。
DataSource JNDI 名	DataSourceJNDIName	データベースへの接続の確立に使用される JNDI データ・ソースの名前。
データベース URL	DatabaseURL	データベースへの接続に使用されるデータベース URL
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
将来のタイム・スタンプを持つイベントを処理しない	FilterFutureEvents	アダプターが各イベントのタイム・スタンプをシステム時刻と比較することによって、将来のイベントをフィルターで除去するかどうかを指定します。
イベントを一度のみ送達する	AssuredOnceDelivery	アダプターにより、1 回のイベント送達を確保する機能が提供されるかどうかを指定します。
イベント順序	イベント順序	イベントが取得および処理される順序。
イベント照会タイプ (Event query type)	EventQueryType	標準イベント・ストアまたはカスタム照会のいずれを使用するかを決定します
イベント・テーブル名	イベント・テーブル名	データベースが Inbound 処理のために生成するイベントを格納するデータベース・テーブルの名前
処理するイベント・タイプ	EventTypeFilter	どのイベントをアダプターが配信するかをアダプターに示す、区切り文字で区切られているイベント・タイプのリスト。
失敗したイベントの再試行制限	FailedEventRetryLimit	アダプターの再送信の試行回数で、この回数に達すると失敗とマークされます。
ポーリング期間の間隔	ポーリング間隔	ポーリング期間中にアダプターが待機する時間の長さ

表 112. IBM WebSphere Adapter for JDBC のアクティブ化仕様プロパティー (続き)

プロパティー名		説明
ウィザード内	管理コンソール内	
JDBC ドライバー・クラス (JDBC driver class)	JDBCDriverClass	データベース接続に使用される JDBC ドライバーのクラス名
最大接続数	MaximumConnections	アダプターが Inbound イベント送達に使用できる接続の最大数
最小接続数	MinimumConnections	アダプターが Inbound イベント送達に使用できる接続の最小数
システム接続に失敗した場合の最大再試行回数	RetryLimit	エラーの発生後に、アダプターが Inbound 接続の再確立を試行する回数。
パスワード	Password	データベースからイベントを検索する際にユーザーを認可するためのパスワード。
ポーリング数量	ポーリング数量	各ポーリング期間中にアダプターがエクスポートに配信するイベント数
照会タイムアウト	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
開始時に EIS 接続を再試行する	RetryConnectionOnStartup	アダプターが始動時に データベース に接続できない場合に、接続を再試行するかどうかを指定します。
システム接続に失敗した場合の再試行間隔 (ミリ秒)	RetryInterval	Inbound 操作時にエラーが発生した後、接続を再確立する各試行間にアダプターが待機する時間の長さ。
RetrieveSP に対しダミー・ビジネス・オブジェクトを戻す (Return dummy business object for RetrieveSP)	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
「接続を検証するための SQL 照会」	ping 照会	データベースへの接続の信頼性をテストするのに使用する SQL 照会
ポーリング時にエラーが検出された場合はアダプターを停止する	StopPollingOnError	ポーリング時にアダプターがエラーを検出した場合、アダプターがイベントのポーリングを停止するかどうかを指定します。
ポーリング後に実行するストアード・プロシージャー (Stored procedure to run after polling)	SPAfterPoll	各ポーリング周期の終了ごとに実行するストアード・プロシージャーの名前
ポーリング前に実行するストアード・プロシージャー (Stored procedure to run before polling)	SPBeforePoll	実際のポーリング照会の呼び出しの前に実行するストアード・プロシージャーの名前
送達のタイプ	DeliveryType	イベントがアダプターによってエクスポートに配信される順序を指定します。
ユーザー名	UserName	Inbound イベントに使用するデータベース・ユーザー名
(なし)	performTrimOnObjectKeyValue	キーと値のペアのスペースを切り取るかどうかを指定します。

表 112. IBM WebSphere Adapter for JDBC のアクティブ化仕様プロパティー (続き)

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
(なし)	nameValuePairDelimiter	キーと値のペアを区切るための区切り文字を指定します。
(なし)	valueDelimeter	キーと値のペアの中でキーと値を分離するための区切り文字を指定します。
(なし)	unsetValueKeyWord	object_key フィールドに unset 値を指定します。
(なし)	nullValueKeyWord	object_key フィールドにヌル値を指定します。
390 ページの『HA Active-Active イベント 処理のタイムアウト期間 (秒)』	eventTimeout	高可用性 Active-Active サポートにおいて、ポーリングされたイベントが処理されなければならない期限を示すタイムアウト期間を指定します。タイムアウト期間が終わると、未処理イベントは新規イベントとして処理されるようになります。 **注:** この値を十分に大きい値にするようにしてください。そうでないと、アダプターがイベント処理フェーズで予期しない例外を生成する可能性があります。

イベント・フィルター用のアダプター・インスタンス (AdapterInstanceEventFilter)

このプロパティーは、このアダプター・インスタンスがイベント・ストア内の特定のイベントを処理するかどうかを制御します。

表 113. 「イベント・フィルター用のアダプター・インスタンス」の詳細

必須	いいえ
デフォルト	NULL
プロパティー・タイプ	String

表 113. 「イベント・フィルター用のアダプター・インスタンス」の詳細 (続き)

使用法	このプロパティーは、WebSphere Business Integration Adapter for JDBC から WebSphere Adapter for JDBC にマイグレーションできるようにします。WebSphere Business Integration Adapter for JDBC では、複数のアダプター・インスタンスで同タイプのイベントを処理できるようにして、ボリュームの大きなイベント・タイプのロード・バランシングを行うことができます。ロード・バランシングが不要な場合は、単独のアダプター・インスタンスで特定タイプのすべてのイベントを処理します。このプロパティーを利用すれば、現在、connectorID フィルター処理を使用している WBIA のお客様が円滑に JCA にマイグレーションを行うことができます。 WebSphere Adapter for JDBC では、通常、このようなロード・バランシングは必要ありませんが、これをサポートすることによって、イベントをイベント・ストアに書き込むデータベース・トリガーその他の機構を変更せずにマイグレーションが行えるようにしています。 AdapterInstanceEventFilter プロパティーは、 WebSphere Business Integration Adapter for JDBC の ConnectorID プロパティーに相当します。 この機能を使用するには、イベント・ストアにイベントを作成するデータベース・トリガーやその他の機構で、適切な値を connector_ID 列に割り当てる必要があります。 366 ページの表 114 は、AdapterInstanceEventFilter プロパティーと、イベント・ストアの connector_ID 列の値との相互関係を示しています。 このプロパティーは、標準のイベント処理に対してのみ適用されます。カスタムのイベント処理の場合は、カスタムのイベント照会により、必要なフィルタリングを実行する必要があります。 EventTypeFilter および AdapterInstanceEventFilter プロパティーの両方が設定された場合、アダプターは、両方の基準を満たすイベントだけを処理します。すなわち、EventTypeFilter プロパティーにタイプが指定されており、connector_ID 列が AdapterInstanceEventFilter プロパティーに一致しているイベントだけが処理されます。 **注:** カスタムのイベント処理を使用している場合、このプロパティーはサポートされません。
例	366 ページの表 114 を参照してください。
グローバル化	はい
BIDI 対応	はい

表 114. AdapterInstanceEventFilter プロパティーと、イベント・ストアの connector_ID 列との相互関係

AdapterInstanceEventFilter プロパティー	イベントの connector_ID 列	結果
NULL	NULL	アダプターはイベントを処理します。
NULL	Instance1	connector_ID 列がチェックされていないため、アダプターはイベントを処理します。
Instance1	Instance1	アダプターはイベントを処理します。
Instance1	Instance2	インスタンス ID が一致しないため、アダプターはイベントを処理しません。
Instance1	NULL	インスタンス ID が一致しないため、アダプターはイベントを処理しません。

追加の JDBC ドライバー接続プロパティー [name:value;name:value] (JDBCDriverConnectionProperties)

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 115. 追加の JDBC ドライバー接続プロパティーの詳細

行	説明
必須	いいえ
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これらの接続プロパティーを、UserName および Password プロパティーと共に使用して、アダプターが使用するデータベース接続をカスタマイズします。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。

表 115. 追加の JDBC ドライバー接続プロパティーの詳細 (続き)

行	説明

例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。
	`loginTimeout:20;readOnly:true;`
	`securityMechanism:USER_ONLY_SECURITY`
	
	高可用性環境では、High Availability and Disaster Recovery (HADR) データベースへの信頼できる接続のために DB2 ドライバーが必要とするプロパティー (retryIntervalForClientReroute、 maxRetriesForClientReroute、clientRerouteAlternateServerName、clientRerouteAlternatePortNumber) を取得するため、正しいプロパティー・ストリングが生成されるようこのプロパティーを構成する必要があります。以下に例を示します。
	`retryIntervalForClientReroute:15;`
	`maxRetriesForClientReroute:5;`
	`clientRerouteAlternateServerName:WLOXS01B.svl.ibm.com;`
	`clientRerouteAlternatePortNumber:50000`
	
	WebSphere 高可用性環境での WebSphere Adapter の使用について詳しくは、 https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=sw-app&S_PKG=wpswasadapt&S_TACT=105AGX10&S_CMP=LC を参照してください。

グローバル化	はい
BIDI 対応	いいえ

ビジネス・オブジェクト名前空間 (BONameSpace)

このプロパティーは、ビジネス・オブジェクト定義の名前空間を指定します。

表 116. 「ビジネス・オブジェクト名前空間」プロパティーの特性

必須	いいえ
デフォルト	`http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc`
プロパティー・タイプ	String
使用法	この値は、ビジネス・オブジェクト名を論理的に分離するため、ビジネス・オブジェクト名の前に追加されます。
例	以下の例は、デフォルト名前空間を使用した `Schema1Customer` ビジネス・オブジェクトを示します。 `http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/Schema1Customer`
BIDI 対応	いいえ

ユーザー定義削除照会 (CustomDeleteQuery)

このプロパティーを使用して、イベントの送達後に削除可能なレコードを削除するために各イベントが処理された後に実行する、SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 117. 「カスタム削除照会 (Custom delete query)」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、EventQueryType プロパティーが Dynamic に設定されている場合に実行する SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。 **注:** 条件を使用した照会の場合、BIDI 変換中に、WHERE 節の後の引用符内のストリングのみが変換されます。例えば、照会 `select * from customer where customerid='john'` では、ストリング 'john' のみが変換されます。
例	`<customEventQuery>select event_id, object_key, object_name, object_function,connector_id from wbia_jdbc_eventstore_dynamic where event_status = 0</customEventQuery> <customUpdateQuery>update wbia_jdbc_eventstore_dynamic set event_status= 1 where event_id = ?</customUpdateQuery> <customUpdateQueryForFailedEvent>update wbia_jdbc_eventstore_dynamic set event_status=-1 where event_id = ?</customUpdateQueryForFailedEvent> <customDeleteQuery>delete from wbia_jdbc_eventstore_dynamic where event_id= ?</customDeleteQuery>`
グローバル化	はい
BIDI 対応	はい

ユーザー定義イベント照会 (CustomEventQuery)

このプロパティーを使用して、カスタム・イベント処理においてイベントをポーリングするために実行する、SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 118. 「カスタム・イベント照会 (Custom event query)」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、EventQueryType プロパティーが Dynamic に設定されている場合の各ポーリング・サイクル中に実行する、SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 118. 「カスタム・イベント照会 (Custom event query)」の詳細 (続き)

例	次の例において、カスタム・イベント照会は、状況列の値が 0 になっている MY_EVENT_TABLE イベント・ストアにあるすべてのレコードのイベント ID、オブジェクト・キー、およびオブジェクト名を返す SQL ステートメントを実行します。
	`select event_id, object_key, object_name from MY_EVENT_TABLE where status = 0`
	次の例では、Oracle データベースの場合に限り、返されるイベント・レコードを PollQuantity プロパティーの値に制限します。
	`select event_id, object_key, object_name from MY_EVENT_TABLE where status = 0 and rownum < POLL QUANTITY`
	次の例では、2 つのパラメーターを持つストアード・プロシージャーを実行します。
	`CALL MY_EVENT_STORED_PROC (?,?)`
	以下の例では、1 つのパラメーターと 1 つの戻り値を持つストアード関数を実行します。
	`? = CALL MY_EVENT_FUNCTION(?)`
	条件を使用した照会の場合、BIDI 変換中に、WHERE 節の後の引用符内のストリングのみが変換されます。例えば、照会 `select * from customer where customerid='john'` では、ストリング 'john' のみが変換されます。もっと例を見るには、367 ページの『ユーザー定義削除照会 (CustomDeleteQuery)』 にある例を参照してください。

グローバル化	はい
BIDI 対応	はい

ユーザー定義の更新照会 (CustomUpdateQuery)

このプロパティーを使用して、同じイベントが後続のイベント周期で処理対象として取り出されないように、各イベントの処理後に実行する SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 119. 「カスタム更新照会 (Custom update query)」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、EventQueryType プロパティーが `Dynamic` に設定されている場合に実行する SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。 **注:** 条件を使用した照会の場合、BIDI 変換中に、WHERE 節の後の引用符内のストリングのみが変換されます。例えば、照会 `select * from customer where customerid='john'` では、ストリング 'john' のみが変換されます。

表 119. 「カスタム更新照会 (Custom update query)」の詳細 (続き)

例	367 ページの『ユーザー定義削除照会 (CustomDeleteQuery)』にリストされている例を参照してください。
グローバル化	はい
BIDI 対応	はい

失敗したイベント送達のためのユーザー定義の更新照会 (CustomUpdateQueryForFailedEvent)

このプロパティーを使用して、イベントが正常に送達されなかったときにイベント状況を更新するために実行する、SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 120. 失敗したイベント送達のためのカスタム更新照会の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、EventQueryType プロパティーが `Dynamic` に設定されている場合に実行する SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。 イベントが正常に送達された場合、イベント状況を更新するために、 369 ページの『ユーザー定義の更新照会 (CustomUpdateQuery)』 プロパティーが使用されます。イベントが正常に送達されない場合、イベント状況を更新するために、『失敗したイベント送達のためのユーザー定義の更新照会 (CustomUpdateQueryForFailedEvent)』 プロパティーが使用されます。
例	367 ページの『ユーザー定義削除照会 (CustomDeleteQuery)』にリストされている例を参照してください。
グローバル化	はい
BIDI 対応	はい

データ・ソース JNDI 名 (DataSourceJNDIName)

このプロパティーは、データベース接続を確立するときに使用される JNDI データ・ソースの名前を指定します。

表 121. 「データ・ソース JNDI 名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 121. 「データ・ソース JNDI 名」の詳細 (続き)

使用法	このプロパティーを使用して、ターゲット・データベースの接続情報を指定する IBM Business Process Manager または WebSphere Enterprise Service Bus のデータ・ソースの JNDI 名を指定します。 Inbound 操作および Outbound 操作のパフォーマンスを改善するには、準備済みステートメントをキャッシュするために使用可能に設定されているデータ・ソースの名前を指定します。 UserName プロパティーと Password プロパティーも設定されている場合は、データ・ソース内のユーザー名とパスワードはこれらのプロパティーにより指定変更されます。 データ・ソース JNDI 名前プロパティーを、サーバーの管理接続ファクトリーまたは活動化仕様の JNDI 名と混同しないでください。以下のリストに、JNDI 名のタイプ間での重要な違いを示します。 • データ・ソース JNDI 名 – データベースへの接続を指定する – ユーザー名とパスワードをアダプターのプロパティーに保存する代わりに使用される – アダプター・プロパティーとして保存される • 管理接続ファクトリーまたは活動化仕様の JNDI 名 – サーバーの管理接続ファクトリーまたは活動化仕様への接続を指定する – ウィザードで各管理接続ファクトリーまたは活動化仕様プロパティーの値を指定する代わりに使用される – インポート・ファイルの接続ターゲットとして保存される
グローバル化	はい
BIDI 対応	いいえ

データベース URL (DatabaseURL)

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 122. 「データベース URL」の詳細

必須	接続タイプが LocalConnectionProps または ConnectionProps の場合は、はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 122. 「データベース URL」の詳細 (続き)

使用法	外部サービス・ウィザードのデータベース固有のフィールドに情報を入力し、データベース URL を作成します。例えば、DB2 データベースのデータベース URL は、データベース名、サーバー・ホスト名、およびデータベース・ポート番号で構成されています。管理コンソールで、データベース URL 値全体を入力します。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。 ホスト名を IPv6 形式の IP アドレスとして指定する場合は、その IP アドレスを大括弧 ([]) で囲んでください。

例	一般的なデータベース・サーバーの標準的な値を以下に示します。
	
	DB2 Universal (タイプ 4) JDBC ドライバー
	`jdbc:db2://www.example.com:50000/DB`
	
	DB2 Universal JDBC ドライバー (IPv6 アドレス)
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	
	DB2 Universal Database タイプ 2 ドライバー (ローカル接続用) jdbc:db2:TEST
	
	DB2 Universal Database タイプ 2 ドライバー (リモート接続用) jdbc:db2://www.example.com:50000/TEST
	
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

グローバル化	はい
BIDI 対応	はい

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 123. 「データベース・ベンダー」の詳細

必須	はい

使用可能な値	`DB2`
	`Informix`
	`MSSQLServer`
	`Oracle`
	`Others`

デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 123. 「データベース・ベンダー」の詳細 (続き)

使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 その他のデータベースの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

送達タイプ (DeliveryType)

このプロパティーでは、イベントがアダプターによってエクスポートに配信される順序を指定します。

表 124. 送達タイプの詳細

必須	いいえ
使用可能な値	ORDERED UNORDERED
デフォルト	ORDERED
プロパティー・タイプ	String
使用法	以下の値がサポートされています。 • ORDERED: アダプターは、一度に 1 つのイベントをエクスポートに配信します。 • UNORDERED: アダプターは、一度にすべてのイベントをエクスポートに配信します。UNORDERED に設定する場合、イベント送達でアダプターがエンドポイントにイベントを正しく配信できるようにするには、複数の接続を指定する必要があります。複数の接続を指定するには、エクスポート・ファイルを編集して、「 380 ページの『最小接続数 (Minimum connections) (MinimumConnections)』」プロパティーおよび「 379 ページの『最大接続数 (Maximum connections) (MaximumConnections)』」プロパティーに適切な値を指定します。
グローバル化	いいえ
BIDI 対応	いいえ

将来のタイム・スタンプを持つイベントを処理しない (FilterFutureEvents)

このプロパティーでは、アダプターが各イベントのタイム・スタンプをシステム時刻と比較することによって、将来のイベントをフィルターで除去するかどうかを指定します。

表 125. 「将来のタイム・スタンプを持つイベントを処理しない」の詳細

必須	はい

表 125. 「将来のタイム・スタンプを持つイベントを処理しない」の詳細 (続き)

使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean
使用法	`True` に設定すると、アダプターは各イベントの時刻をシステム時刻と比較します。イベント時刻がシステム時刻より後の時刻である場合、そのイベントは配信されません。 `False` に設定すると、アダプターはすべてのイベントを配信します。 **注:** カスタムのイベント処理を使用している場合、このプロパティーはサポートされません。
グローバル化	いいえ
BIDI 対応	いいえ

イベントを一度のみ送達する (AssuredOnceDelivery)

このプロパティーでは、Inbound イベントに対して、「イベントを一度のみ送達する」の機能を提供するかどうかを指定します。

表 126. 「イベントを一度のみ送達する」の詳細

必須	はい
使用可能な値	`True` `False`
デフォルト	`True`
プロパティー・タイプ	Boolean
使用法	このプロパティーを `True` に設定すると、アダプターにより、1 回のイベント送達を確保する機能が提供されます。つまり、各イベントは 1 回のみ配信されます。値を `False` にすると、1 回のイベント送達を確保する機能は提供されませんが、パフォーマンスは向上します。 このプロパティーを `True` に設定すると、アダプターにより、トランザクション (XID) 情報のイベント・ストアへの保管が試行されます。このプロパティーを `False` に設定した場合は、アダプターではこの情報の保管は行われません。 このプロパティーは、エクスポート・コンポーネントがトランザクションの対象である場合にのみ使用されます。そうでない場合は、このプロパティーの値に関係なく、トランザクションを使用することはできません。
グローバル化	いいえ
BIDI 対応	いいえ

イベント順序 (EventOrderBy)

イベントが取得および処理される順序。

表 127. 「イベント順序」の詳細

必須	いいえ
使用可能な値	イベント・ストアの列名をコンマ (,) で区切ったリストと、順序属性 `asc` および `desc`
デフォルト	`event_time, event_priority`
プロパティー・タイプ	String
使用法	イベント・ストアの列名をコンマで区切って記述したリストと、オプションの属性 (昇順または降順) を指定します。

例	並べ替えの第 1 条件として時刻、第 2 条件として優先順位を使用して並べ替えられたイベントを表示するには、次のように指定します。
	`event_time, event_priority`
	並べ替えの第 1 条件としてオブジェクト名 (昇順)、第 2 条件として時間 (降順) を使用して並べ替えられたイベントを表示するには、次のように指定します。
	`object_name asc, event_time desc`

グローバル化	はい
BIDI 対応	はい

イベントを処理するためのイベント照会タイプ (EventQueryType)

このプロパティーは、標準照会処理とカスタム照会処理のどちらを使用するかを指定します。

表 128. 「イベント照会タイプ (Event query type)」の詳細

必須	はい
使用可能な値	`Standard` `User-Defined(Dynamic)`
デフォルト	`Standard`
プロパティー・タイプ	String
使用法	有効な値は、標準イベント処理用の `Standard` とカスタム・イベント処理用の `Dynamic` です。 このプロパティーが `User-Defined(Dynamic)` に設定されていると、 CustomEventQuery、 CustomUpdateQuery、 CustomUpdateQueryForFailedEvent、および CustomDeleteQuery プロパティーが使用されます。このプロパティーが `Standard` に設定されていると、これらのプロパティーは無視されます。
グローバル化	いいえ
BIDI 対応	いいえ

イベント・テーブル名 (EventTableName)

このプロパティーは、Inbound 処理に使用されるイベント・ストアが格納されているターゲット・データベース内のテーブルの名前を指定します。

表 129. 「イベント・テーブル名」の詳細

必須	はい
デフォルト	`WBIA_JDBC_EventStore`
プロパティー・タイプ	String
使用法	アダプターの構成を開始する前に、イベント・ストアを作成してください。 標準イベント処理では、イベントはトリガーまたはその他のメカニズムを介してデータベースにより生成されます。カスタム照会処理では、アダプターはカスタム照会の結果を受信すると、イベントをイベント・ストアに保存します。
グローバル化	はい
BIDI 対応	はい

処理するイベント・タイプ (EventTypeFilter)

このプロパティーには、どのイベントをアダプターが配信するかをアダプターに示す、区切り文字で区切られているイベント・タイプのリストが入っています。

表 130. 「処理するイベント・タイプ」の詳細

必須	いいえ
使用可能な値	のコンマ区切り (,) のリスト。 • 「,」は、各フィルターを分離する際に使用します (後方互換)。 • 「:」は、ビジネス・オブジェクト名と操作名を分離する際に使用します。 • 「|」は、サポートされる各操作を分離する際に使用します。
デフォルト	`NULL`
プロパティー・タイプ	String

表 130. 「処理するイベント・タイプ」の詳細 (続き)

使用法	イベントは、ビジネス・オブジェクト名または機能/操作フィルター・タイプによってフィルタリングされます。によってフィルタリングされます。このプロパティーを設定すると、アダプターは、リスト内のパターンに一致するイベントのみを送達します。値が `null` の場合は、フィルターを適用せずにすべてのイベントをエクスポートに送達することを示します。 **注:** カスタムのイベント処理を使用している場合、このプロパティーはサポートされません。 ポーリング・フィルターは、反復モードをサポートしません。操作を反復モードで変更した場合は、値を手動で更新し、フィルター値の操作と EMD ウィザードの操作リストの操作の整合性を保つ必要があります。 `*` 記号は、すべての値を表し、フィルターを適用しないことを示します。例えば、`,` は論理演算子 OR を表すため、値 `*:*,BOName1:Create` は、フィルターを適用しないことを示します。
例	このフィルター内のビジネス名は、イベント・テーブル内の object_name 列と同じで、操作は Create、Update、および Delete です。例えば、次のようになります。 • Customer ビジネス・オブジェクトおよび Order ビジネス・オブジェクトに関連するイベントのみを受信するには、値 `SchemaNameCustomerBG,SchemaNameOrderBG` を指定します。 • Customer ビジネス・オブジェクトに関連する Create 操作のみのイベント、および Order ビジネス・オブジェクトに関連する Create または Delete 操作のイベントを受信するには、値 `SchemaNameCustomerBG:Create, SchemaNameOrderBG:Create|Delete` を指定します。 **注:** `SchemaName` を実際のスキーマ名に置き換えて使用してください。 このプロパティーは、標準のイベント処理に対してのみ適用されます。カスタムのイベント処理の場合は、カスタムのイベント照会により、必要なフィルタリングを実行する必要があります。 EventTypeFilter および AdapterInstanceEventFilter プロパティーの両方が設定された場合、アダプターは、両方の基準を満たすイベントだけを処理します。すなわち、EventTypeFilter プロパティーにタイプが指定されており、connector_ID 列が AdapterInstanceEventFilter プロパティーに一致しているイベントだけが処理されます。
グローバル化	いいえ
BIDI 対応	いいえ

失敗したイベントの再試行制限 (FailedEventRetryLimit)

このプロパティーは、アダプターがイベントの再送信を試行する回数を指定します。この回数に達するとイベントは失敗とマーキングされます。

表 131. 「失敗したイベントの再試行制限」の詳細

必須	いいえ
使用可能な値	整数
デフォルト	5
プロパティー・タイプ	Integer
使用法	このプロパティーは、この回数以上は失敗とマークする場合の、アダプターがイベントを送信する回数を指定する時に使用します。以下のいずれかの値を取ります。 **デフォルト** このプロパティーが設定されない場合、アダプターは、イベント送信失敗の後、さらに 5 回イベントの送信を試み、それでも送達できない場合にイベントを失敗とマーク付けします。 **0** アダプターは、回数無制限でイベントの送信を試行します。このプロパティーが 0 に設定されると、イベントはイベント・ストアに残されたままになり、イベントが失敗とマークされることはなくなります。 **> 0** 正の整数の場合、アダプターは、指定した回数再試行を行った後、イベントを失敗とマークします。 **< 0** 負の整数の場合、アダプターは失敗したイベントの送信を再試行しません。
グローバル化	いいえ
BIDI 対応	いいえ

JDBC ドライバー・クラス (JDBC driver class) (JDBCDriverClass)

このプロパティーは、データベース接続に使用される JDBC ドライバーのクラス名を指定します。

表 132. 「JDBC ドライバー・クラス (JDBC driver class)」の詳細

行	説明
必須	接続タイプが LocalConnectionProps または ConnectionProps の場合は、はい
使用可能な値	値はデータベースによって異なります。
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 132. 「JDBC ドライバー・クラス (JDBC driver class)」の詳細 (続き)

使用法	外部サービス・ウィザードでは、共通データベース・ソフトウェアとドライバーの組み合わせ (IBM DB2、Oracle、および Microsoft SQL の最新バージョンのタイプ 4 ドライバーなど) を選択すると、JDBC ドライバー・クラスが表示されます。ほとんどのデータベース・ソフトウェアのほとんどのタイプ 2 ドライバーでは、データベース・クラス名を入力する必要があります。 例えば、DB2 Universal Database タイプ 2 ドライバーの場合、クラス名は `COM.ibm.db2.jdbc.app.DB2Driver` です。 管理コンソールで、ドライバーのデータベース固有名を入力してください。

例	外部サービス・ウィザードおよび管理コンソールの両方の JDBC ドライバー・クラス表示画面の値。次の例は、外部サービス・ウィザード と管理コンソールの両方の JDBC ドライバーのクラス・プロパティーを示しています。**外部サービス・ウィザード の場合:**
	• ユニバーサル (タイプ 4) JDBC ドライバーを使用して DB2 データベースに接続するには、「`IBM DB2 Universal`」を選択します。
	• DB2 ユニバーサルのタイプ 2 ドライバーを使用して DB2 データベースに接続するには、「`Other`」を選択します。
	• タイプ 4 ドライバーを使用して Oracle 10 データベースに接続するには、「`Oracle Thin Driver`」を選択します。
	
	管理コンソール内
	
	DB2 Universal Database タイプ 2 ドライバー
	`COM.ibm.db2.jdbc.app.DB2Driver`
	
	DB2 Universal Database タイプ 4 ドライバー
	`com.ibm.db2.jcc.DB2Driver`
	
	Oracle Thin JDBC ドライバー
	`oracle.jdbc.driver.OracleDriver`
	
	IBM Toolkit for Java リモート・ドライバー (IBM i 用)
	`com.ibm.as400.access.AS400JDBCDriver`
	
	IBM WebSphere Connect JDBC ドライバー (Microsoft SQL Server 用)
	`com.ibm.websphere.jdbc.sqlserver.SQLServerDriver`

グローバル化	いいえ
BIDI 対応	いいえ

最大接続数 (Maximum connections) (MaximumConnections)

このプロパティーでは、アダプターが Inbound イベント送達に使用できる接続の最大数を指定します。

表 133. 「最大接続数 (Maximum connections)」の詳細

必須	いいえ
デフォルト	1
プロパティー・タイプ	Integer
使用法	正の値のみが有効です。アダプターは、1 より小さい正の入力値を 1 であるとみなします。このプロパティーに対して負の値を入力すると、ランタイム・エラーが発生することがあります。
グローバル化	いいえ
BIDI 対応	いいえ

最小接続数 (Minimum connections) (MinimumConnections)

このプロパティーでは、アダプターが Inbound イベント送達に使用できる接続の最小数を指定します。

表 134. 「最小接続数 (Minimum connections)」の詳細

必須	いいえ
デフォルト	1
プロパティー・タイプ	Integer
使用法	正の値のみが有効です。1 より小さい値は、アダプターによって 1 として処理されます。 このプロパティーに対して負の値または 1 を入力すると、実行時エラーが発生することがあります。
グローバル化	いいえ
BIDI 対応	いいえ

パスワード (Password)

このプロパティーは、データベース・ユーザーのユーザー名に対するパスワードを指定します。

表 135. パスワードの詳細

必須	いいえ。Inbound 処理では、認証別名または DataSourceJNDIName を設定する場合は、パスワードは必須ではありません。ただし、DataSourceJNDIName、および「パスワード」フィールドを設定した場合、パスワードに指定した値が優先されます。 Outbound 処理では、認証別名、XADataSourceJNDIName プロパティー、または PoolDataSourceJNDIName プロパティーを設定した場合、パスワードは必須ではありません。ただし、XADataSourceJNDIName または PoolDataSourceJNDIName、および「パスワード」フィールドを設定した場合、パスワードに指定した値が優先されます。
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 135. パスワードの詳細 (続き)

使用法	Inbound 処理の場合、このプロパティーを設定すると、認証別名または DataSourceJNDIName プロパティーを使用してサーバーのデータ・ソースに指定されたパスワードが指定変更されます。 Outbound 処理の場合、このプロパティーを設定すると、認証別名または、XADataSourceJNDIName または PoolDataSourceJNDIName プロパティーを使用してサーバーのデータ・ソースに指定されているパスワードが上書きされます。 JAAS をセキュリティー資格情報として指定すると、このプロパティーは認証別名によって指定変更されます。
グローバル化	はい
BIDI 対応	はい

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の信頼性をテストするために使用される SQL 照会を指定します。

表 136. 「ping 照会」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし
使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。 インバウンド・イベント処理の場合、ping 照会が無効な接続を指摘すると、アダプターが例外をスローします。
グローバル化	いいえ
BIDI 対応	いいえ

ポーリング期間の間隔 (PollPeriod)

このプロパティーでは、ポーリング期間中にアダプターが待機する時間の長さを指定します。

表 137. 「ポーリング期間の間隔」の詳細

必須	はい
使用可能な値	0 以上の整数
デフォルト	2000
計測単位	ミリ秒
プロパティー・タイプ	Integer

表 137. 「ポーリング期間の間隔」の詳細 (続き)

使用法	ポーリング期間は一定の割合で確立されます。つまり、ポーリング周期の実行が何らかの理由で遅延すると (例えば、前のポーリング周期が完了するまでに予想より時間がかかった場合)、遅延によって失った時間を取り戻すために次のポーリング周期がすぐに開始されます。
グローバル化	いいえ
BIDI 対応	いいえ

ポーリング期間内の最大イベント数 (ポーリング数量)

このプロパティーでは、各ポーリング期間中にアダプターがエクスポートに配信するイベント数を指定します。

表 138. 「ポーリング期間内の最大イベント数」の詳細

必須	はい
デフォルト	10
プロパティー・タイプ	Integer
使用法	値は 0 より大きくする必要があります。この値を大きくすると、ポーリング期間ごとに処理されるイベントの数が増加し、アダプターのパフォーマンス効率が低下する場合があります。この値を小さくすると、ポーリング期間ごとに処理されるイベントの数が減少し、アダプターのパフォーマンスが若干向上することがあります。
グローバル化	いいえ
BIDI 対応	いいえ

照会タイムアウト (秒) (QueryTimeOut)

このプロパティーは、1 つの照会ですべての SQL ステートメントの実行に費やすことのできる最大時間を秒数で指定します。

表 139. 「照会タイムアウト」の詳細

必須	いいえ
デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会を処理する際に、指定された秒数より長い時間がかかると、キャプチャーされた SQL 例外がデータベースから戻されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	いいえ
BIDI 対応	いいえ

接続が失敗した場合の再試行間隔 (RetryInterval)

このプロパティーでは、アダプターが Inbound 接続に関連したエラーを検出した場合に、アダプターが接続を再確立しようとするまでの待機時間の長さを指定します。

表 140. 再試行間隔の詳細

必須	はい
デフォルト	2000
計測単位	ミリ秒
プロパティー・タイプ	Integer
使用法	正の値のみが有効です。このプロパティーでは、アダプターが、Inbound 接続に関連したエラーを検出した場合に新規接続の確立を試行するまでの待機時間の長さを指定します。
グローバル化	いいえ
BIDI 対応	いいえ

システム接続を再試行する回数 (RetryLimit)

このプロパティーでは、アダプターが Inbound 接続の再確立を試行する回数を指定します。

表 141. 「システム接続を再試行する回数」の詳細

必須	いいえ
使用可能な値	0 および正の整数
デフォルト	0
プロパティー・タイプ	Integer
使用法	このプロパティーは、アダプターが データベース に接続して Inbound 処理を実行できない場合に、接続を再試行する回数を指定します。値が 0 の場合は、再試行回数が無制限になることを指定します。 アダプターの当初始動時に、データベース に接続できない場合、アダプターが再試行するかどうかを制御するには、RetryConnectionOnStartup プロパティーを使用します。
グローバル化	いいえ
BIDI 対応	いいえ

開始時に EIS 接続を再試行する (RetryConnectionOnStartup)

このプロパティーは、アダプターが始動時に データベース に接続できない場合に、再度接続を試みるかどうかを指定します。

表 142. 「開始時に EIS 接続を再試行する」の詳細

必須	いいえ
使用可能な値	True False
デフォルト	False

表 142. 「開始時に EIS 接続を再試行する」の詳細 (続き)

プロパティー・タイプ	Boolean
使用法	このプロパティーは、アダプターの始動時に、データベース に接続できない場合に、接続を再試行するかどうかを指定します。 • アダプターが、データベース に対する接続を確立できたかどうかに関するフィードバックを直ちに必要とする場合、例えば、アダプターからのイベントを受信するアプリケーションを作成し、テストしている場合は、このプロパティーを `False` に設定します。アダプターが接続できない場合、アダプターは、ログおよびトレース情報を書き込んで、停止します。管理コンソールは、アプリケーション状況を `Stopped` と表示します。この場合、接続の問題を解決後、手動でアダプターを始動してください。 • 接続に関するフィードバックをすぐに必要としない場合は、このプロパティーは `True` に設定します。アダプターが始動時に接続できない場合、アダプターはログおよびトレース情報を書き込んでから、RetryInterval プロパティーで再試行の頻度を判別して再接続を試み、RetryLimit プロパティーの値で指定された値に達するまで、再試行を複数回行います。管理コンソールは、アプリケーション状況を `Started` と表示します。
グローバル化	いいえ
BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 143. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean

表 143. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細 (続き)

使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが `False` に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP プロパティーが `True` に設定されている場合は、ダミーのビジネス・オブジェクトが作成され、ストアード・プロシージャーから返されるパラメーター (出力パラメーターと入出力パラメーターを含む) が、対応する属性に取り込まれます。
グローバル化	いいえ
BIDI 対応	いいえ

ポーリング時にエラーが検出された場合はアダプターを停止する (StopPollingOnError)

このプロパティーでは、ポーリング時にアダプターがエラーを検出した場合、アダプターがイベントのポーリングを停止するかどうかを指定します。

表 144. 「ポーリング時にエラーが検出された場合はアダプターを停止する」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean
使用法	このプロパティーを `True` に設定した場合、アダプターはエラーを検出するとポーリングを停止します。 このプロパティーを `False` に設定した場合、アダプターはポーリング時にエラーを検出すると例外をログに記録し、ポーリングを続行します。
グローバル化	いいえ
BIDI 対応	いいえ

ポーリング後に実行するストアード・プロシージャー (SPAfterPoll)

このプロパティーは、各ポーリング周期の完了後に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。

表 145. 「ポーリング後に実行するストアード・プロシージャー」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 145. 「ポーリング後に実行するストアード・プロシージャー」の詳細 (続き)

使用法	ストアード・プロシージャーは、ポーリング回数を表す整数タイプの 1 つの入力パラメーターを使用します。これは、ランタイム時にアダプターによって設定されます。構文は次のとおりです。
	`call <spname> (?)`
	例: `SPAfterPoll>call afterpool(?)</SPAfterPoll>`
	ストアード・プロシージャーでは、追加の入力パラメーターを使用することもできます。その場合には、値を call ステートメント自体で指定する必要があります。追加の入力パラメーターを使用するストアード・プロシージャーの構文を以下に示します。
	`call <spname> (25, ?, 'test')`

グローバル化	はい
BIDI 対応	はい

ポーリング前に実行するストアード・プロシージャー (SPBeforePoll)

このプロパティーは、実際のポーリング照会の呼び出し前に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。

表 146. 「ポーリング前に実行するストアード・プロシージャー」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String

使用法	ストアード・プロシージャーは、ポーリング回数を表す整数タイプの 1 つの入力パラメーターを使用します。これは、ランタイム時にアダプターによって設定されます。構文は次のとおりです。
	`call <spname> (?)`
	例: `<SPBeforePoll>call beforepool(?)</SPBeforePoll>`
	ストアード・プロシージャーでは、追加の入力パラメーターを使用することもできます。その場合には、値を call ステートメント自体で指定する必要があります。追加の入力パラメーターを使用するストアード・プロシージャーの構文を以下に示します。
	`call <spname> (25, ?, 'test')`

グローバル化	はい
BIDI 対応	はい

ユーザー名 (UserName)

このプロパティーは、データベースのアクセスに使用されるデータベース・ユーザー名を指定します。

表 147. 「ユーザー名」の詳細

必須	いいえ。Inbound 処理では、認証別名または DataSourceJNDIName を設定した場合、ユーザー名プロパティーは必須ではありません。ただし、DataSourceJNDIName、および「ユーザー名」フィールドを設定した場合、ユーザー名に指定した値が優先されます。 Outbound 処理では、認証別名または、XADataSourceJNDIName または PoolDataSourceJNDIName を設定した場合、ユーザー名は必須ではありません。ただし、XADataSourceJNDIName または PoolDataSourceJNDIName、および「ユーザー名」フィールドを設定した場合、ユーザー名に指定した値が優先されます。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	Inbound 処理の場合、このプロパティーを設定すると、DataSourceJNDIName プロパティーまたは認証別名を使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。 Outbound 処理の場合、このプロパティーを設定すると、XADataSourceJNDIName プロパティー、PoolDataSourceJNDIName プロパティーまたは認証別名を使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。 JAAS をセキュリティー資格情報として指定すると、このプロパティーは認証別名によって指定変更されます。
グローバル化	はい
BIDI 対応	はい

キー値ペアのスペースの切り取り (performTrimOnObjectKeyValue)

このプロパティーは、Inbound 操作でキー値ペアの前後にあるスペースを切り取る必要があるかどうかを指定します。

表 148. 「キー値ペアのスペースの切り取り」の詳細

必須	いいえ。
使用可能な値	`True` `False`
デフォルト	`True`
プロパティー・タイプ	Boolean
使用法	このプロパティーを `True` に設定すると、inbound 操作はキー値ペアの前後のスペースを切り取ります。このプロパティーを `False` に設定すると、スペースは切り取られません。
グローバル化	いいえ
BIDI 対応	いいえ

キーと値のペアの分離 (nameValuePairDelimiter)

このプロパティーでは、キーと値のペアを区切るための区切り文字を指定します。

表 149. 「名前と値のペアの分離」の詳細

必須	いいえ。
使用可能な値	任意のストリング
デフォルト	なし
プロパティー・タイプ	String
使用法	キーと値のペアを区切るための区切り文字。
例	key1=value1;key2=value2;key3=value3 または value1;value2;value3
グローバル化	いいえ
BIDI 対応	いいえ

キーと値の分離 (valueDelimeter)

このプロパティーは、キー値ペアのキーと値を分離するための区切り文字を指定します。

表 150. 「キーと値の分離」の詳細

必須	いいえ。
使用可能な値	任意のストリング
デフォルト	=
プロパティー・タイプ	String
使用法	キーと値を区切るための区切り文字。
例	key1=value1 または value1
グローバル化	いいえ
BIDI 対応	いいえ

未設定値 (unsetValueKeyWord)

このプロパティーは object_key フィールド内の未設定値を表します。

表 151. 未設定値のキーワード詳細

必須	いいえ
使用可能な値	JCA_JDBC_UNSET
デフォルト	ブランク
プロパティー・タイプ	String

表 151. 未設定値のキーワード詳細 (続き)

使用法	このプロパティーがブランク以外に設定されている場合、アダプターは object_key フィールド内の値を検査します。object_key フィールドに値が存在する場合には、アダプターはこのフィールドを未設定として扱います。例えば、このプロパティーが「JCA_JDBC_UNSET」に設定され、object_key フィールド値が「1; JCA_JDBC_UNSET;2」の場合、アダプターは実行時に、値「JCA_JDBC_UNSET」に対応するフィールドを unset として扱い、オブジェクトを取得するときにそのフィールドを WHERE 節で使用しません。
グローバル化	いいえ
BIDI 対応	いいえ

ヌル値 (nullValueKeyWord)

このプロパティーは object_key フィールド内のヌル値を表します。

表 152. ヌル値のキーワード詳細

必須	いいえ
使用可能な値	JCA_JDBC_NULL
デフォルト	ブランク
プロパティー・タイプ	String
使用法	このプロパティーがブランク以外に設定されている場合、アダプターは object_key フィールド内の値を検査します。object_key フィールドに値が存在する場合には、アダプターはこのフィールドをヌルとして扱います。例えば、このプロパティーが「JCA_JDBC_NULL」に設定され、object_key フィールド値が「1; JCA_JDBC_NULL;2」の場合、アダプターは実行時に、値「JCA_JDBC_NULL」に対応するフィールドをヌルとして扱い、オブジェクトを取得するときに WHERE 節でそのフィールド値を「is Null」に設定します。
グローバル化	いいえ
BIDI 対応	いいえ

データベース接続情報 (ConnectionType)

このプロパティーは、アダプターがデータベースへの接続を確立する方法を指定します。

表 153. データベース接続情報

必須	はい
使用可能な値	ConnectionProps または DataSourceJNDI
デフォルト	ConnectionProps
プロパティー・タイプ	String

表 153. データベース接続情報 (続き)

使用法	このプロパティーは、アダプターが実行時にデータベース接続を確立する方法を指定します。このプロパティーの有効な値は、以下のとおりです。 • ConnectionProps - データベース接続が DatabaseURL プロパティーおよび JDBCDriverClass プロパティーを使用して確立されることを示します。 • DataSourceJNDI - データベース接続が、定義済みデータ・ソースに対応する DataSourceJNDIName プロパティーを使用して確立されることを示します。 新規アプリケーションの場合、このプロパティーは、外部のサービス・ウィザードによって自動的に設定されます。マイグレーション済みアプリケーションの場合、このプロパティーは、マイグレーション・プロセス中に ActivationSpecification プロパティーに応じて設定されます。 • DataSourceJNDIName プロパティーが設定されている場合、このプロパティーの値は DataSourceJNDI に設定されます。 • DataSourceName プロパティーが設定されている場合、このプロパティーの値は ConnectionProps に設定されます。 このプロパティーが設定されない場合、アダプターは後方互換モードを使用して、データベース接続を確立します。後方互換性モードでは、データベース接続プロパティーは、次の順序で使用されます。 1. DataSourceJNDIName プロパティーが設定されている場合、アダプターはこのプロパティーを使用してデータベースへの接続を確立します。 2. DataSourceJNDIName プロパティーが設定されていない場合、アダプターは DatabaseURL、JDBCDriverClass、UserName、および Password プロパティーを使用して接続を確立します。
グローバル化	いいえ
BIDI 対応	いいえ

HA Active-Active イベント処理のタイムアウト期間 (秒)

HA Active-Active イベントを処理する時間を秒数で指定します。この時間が終わった時点で処理されていないイベントは新規イベントとして処理されます。

表 154. HA Active-Active イベント処理のタイムアウト期間 (秒) プロパティーの特性

必須	はい
デフォルト	300
計測単位	秒
プロパティー・タイプ	Integer
使用法	このプロパティーは、ポーリングされたイベントをアダプターが処理する時間を秒単位で指定するのに使用されます。クラスター環境で、活動状態のアダプター・インスタンスが、ポーリングされたイベントを何らかの理由で指定時間内にすべて処理することができない場合 (例えば、クラスター内のノードが異常終了になった場合)、処理されなかったイベントは他の活動状態のアダプター・インスタンスによって新規イベントとしてリストアされます。 **注:** このプロパティーを使用できるのは、高可用性 Active-Active 構成が有効になっている場合です。Inbound 高可用性 Active-Active モードを有効にするには、 322 ページの『高可用性 サポートを使用可能にする (enableHASupport)』 を false に設定します。詳しくは、 84 ページの『クラスター環境での WebSphere Adapters』を参照してください。
グローバル化	いいえ

表 154. HA Active-Active イベント処理のタイムアウト期間 (秒) プロパティーの特性 (続き)

BIDI 対応	いいえ

グローバリゼーション

WebSphere Adapter for JDBC は、複数の言語および国/地域別環境で使用することができる、グローバル化されたアプリケーションです。アダプターは、文字セット・サポートおよびホスト・サーバーのロケールに基づいて、メッセージ・テキストを適切な言語で送信します。アダプターは、統合コンポーネント間の双方向スクリプト・データの変換をサポートします。

グローバリゼーションおよび双方向変換

アダプターは、1 バイト文字セットとマルチバイト文字セットをサポートし、メッセージ・テキストを指定された言語で配信できるようにグローバル化されています。このアダプターは、双方向変換も行います。双方向変換とは、同じファイル内に右から左の意味内容 (ヘブライ語やアラビア語など) と左から右の意味構造 (URL またはファイル・パスなど) の両方を含むデータを処理するタスクのことです。

グローバリゼーション

グローバル化されたソフトウェア・アプリケーションは、言語環境や国/地域別環境が単一ではなく複数の環境で使用することを目的として設計され、開発されています。IBM WebSphere Adapters、IBM Integration Designer、IBM Business Process Manager、および IBM WebSphere Enterprise Service Bus は、Java で作成されています。Java 仮想マシン (JVM) 内の Java ランタイム環境は、 Unicode 文字コード・セットでデータを表現します。Unicode には、知られているほとんどの文字コード・セット (1 バイトとマルチバイトの両方) の文字エンコードが含まれています。そのため、これらの統合システム・コンポーネント間でデータを転送するときに文字を変換する必要はありません。

エラー・メッセージや通知メッセージを適切な言語や個々の国や地域に合った形でログに記録するために、アダプターは稼働先システムのロケールを使用します。

双方向変換

アラビア語やヘブライ語などの言語は右から左に記述されますが、内部に、左から右に記述されるテキストのセグメントが組み込まれているため、双方向スクリプトとなります。ソフトウェア・アプリケーションで双方向スクリプトを表示および処理する方法は複数あります。 IBM Business Process Manager と IBM WebSphere Enterprise Service Bus では Windows 標準形式が使用されますが、IBM Business Process Manager または WebSphere Enterprise Service Bus とデータを交換するエンタープライズ情報システムでは他の形式を使用できます。IBM WebSphere Adapters は、2 つのシステム間でやり取りされる双方向スクリプト・データの変換を行うことによって、トランザクションの両側でデータが正確に処理および表示されるようにします。

双方向形式

IBM Business Process Manager および IBM WebSphere Enterprise Service Bus は、ILYNN (暗黙、左から右、オン、オフ、公称) の双方向形式を使用します。これは、

Windows によって使用される形式です。エンタープライズ情報システムが別の形式を使用する場合、アダプターは、データを IBM Business Process Manager または WebSphere Enterprise Service Bus に導入する前に形式を変換します。

双方向形式は、5 つの属性で構成されます。双方向プロパティーを設定する場合、これらの各属性に値を割り当てます。属性と設定値を次の表に示します。

表 155. 双方向形式の属性

文字の位置	目的	値	説明	デフォルト設定
1	スキーマの配列	I	暗黙 (論理的)	I
		V	表示	
2	方向	L	左から右	L
		R	右から左	
		C	コンテキスト上の左から右	
		D	コンテキスト上の右から左	
3	対称スワッピング	Y	対称スワッピングのオン	Y
		N	対称スワッピングのオフ	
4	テキストの形状指定	S	テキストの形状を指定する	N
		N	テキストの形状を指定しない (名目)	
		I	語頭形の指定	
		M	語中形の指定	
		F	語尾形の指定	
		B	独立形の指定	
5	数字の形状指定	H	各国 (ヒンディ語)	N
		C	コンテキストによる形状指定	
		N	数字の形状を指定しない (名目)	

アダプターは、データを左から右の論理形式に変換してから IBM Business Process Manager または WebSphere Enterprise Service Bus に送信します。

双方向プロパティーの使用

複数の双方向プロパティーを使用して、コンテンツ・データとメタデータの両方の変換を制御できます。特別な双方向プロパティーを設定してコンテンツ・データまたはメタデータのいずれかを双方向変換から除外するか、変換中に特別な処理が必要なデータを識別できます。

以下の表で、双方向プロパティーのタイプについて説明します。

表 156. 双方向プロパティーのタイプ

プロパティー・タイプ	データ変換
EIS	コンテンツ・データ (エンタープライズ情報システム、すなわちデータベースによって送信されるデータ) の形式を制御します。

表 156. 双方向プロパティーのタイプ (続き)

プロパティー・タイプ	データ変換
メタデータ	メタデータ (コンテンツ・データに関する情報を提供するデータ) の形式を制御します。
スキップ	変換から除外するコンテンツまたはメタデータを識別します。
特殊形式	変換処理中に異なる処理を必要とするファイル・パスまたは URL などの特定のテキストを識別する。コンテンツ・データまたはメタデータのいずれかに設定できる。

以下の領域で双方向変換を制御するプロパティーを設定できます。

- **リソース・アダプター・プロパティー:** これらのプロパティーは、TurnBiDiOff プロパティー (アダプター・インスタンスが双方向変換を実行するかどうかを制御します) などのデフォルト構成設定を格納します。これらのプロパティーを構成するには、サーバーの管理コンソールを使用します。
- **管理接続ファクトリー・プロパティー:** これらのプロパティーは、エンタープライズ情報システムとの Outbound 接続インスタンスを作成するために実行時に使用されます。管理接続ファクトリー・プロパティーは作成された後、デプロイメント記述子に格納されます。
- **活動化仕様プロパティー:** これらのプロパティーは、メッセージ・エンドポイントに対する Inbound イベント処理構成情報を保持します。外部サービス・ウィザードを使用するか、またはサーバーの管理コンソールを使用して設定します。

プロパティーのスコープと検索機構

アダプターの双方向プロパティーに値を設定すると、アダプターは双方向変換を実行します。実行時には、プロパティー設定の階層の継承と、検索機構に依存するロジックを使用します。

リソース・アダプター内で定義されたプロパティーは階層のトップにありますが、他の領域で定義されたプロパティーまたはビジネス・オブジェクト内で注釈が付けられたプロパティーは下位の階層にあります。このため、例えば、リソース・アダプターの EIS タイプの双方向プロパティーのみに値を設定すると、Inbound (活動化仕様) トランザクションと Outbound (Managed Connection Factory) トランザクションのいずれで発生するかにかかわらず、定義された EIS タイプの双方向プロパティーを必要とする変換によってそれらの値が継承および使用されます。

しかし、リソース・アダプターと活動化仕様の両方の EIS タイプの双方向プロパティーに値を設定した場合、Inbound トランザクションから発生した変換は、活動化仕様に設定された値を使用します。

処理ロジックでは、変換時に使用する双方向プロパティーの値を、検索機構を使用して検索します。検索機構は、変換が生じるレベルから検索を開始し、適切なプロパティー・タイプを持つ定義済みの値を対象に、階層の上位に向かって検索を進めます。検出された最初の有効値が使用されます。階層の検索は、子から親の方向にのみ進行します。兄弟は検索の対象になりません。

双方向データ変換で使用可能なプロパティー

IBM WebSphere Adapter for JDBC には、双方向データ変換で使用可能な各種構成プロパティーがあります。

アダプターは、クライアント・アプリケーションとデータベースの間での双方向データの交換を有効にします。これは、データベースのデータの双方向形式が、ランタイム環境で使用される双方向形式と異なる場合でも有効になります。アダプターの構成時、およびビジネス・オブジェクトのアプリケーション固有情報では、双方向文字を使用できます。双方向サポートで使用可能なプロパティーとアプリケーション固有情報を以下に示します。

- 構成プロパティー
  - 活動化仕様プロパティー
  - 外部サービス・ウィザードの接続プロパティー
  - 管理接続ファクトリー・プロパティー
- アプリケーション固有情報
  - ビジネス・オブジェクト・レベル ASI
  - 操作レベル ASI
  - 属性レベル ASI

以降のセクションでは、双方向変換で使用可能な特定の構成プロパティーとアプリケーション固有情報をリストします。

ウィザードで使用される接続プロパティー

双方向スクリプト・データ変換で使用可能な、外部サービス・ウィザードの接続プロパティーを以下に示します。

- ユーザー名
- パスワード

管理接続ファクトリー・プロパティー

双方向スクリプト・データ変換で使用可能な管理接続プロパティーを以下に示します。

- 追加の JDBC ドライバー接続プロパティー
- データベース URL
- パスワード
- ユーザー名
- XA データベース名

活動化仕様プロパティー

双方向スクリプト・データ変換で使用可能な活動化仕様プロパティーを以下に示します。

- カスタム削除照会 (Custom delete query)
- カスタム・イベント照会 (Custom event query)
- カスタム更新照会 (Custom update query)

- 追加の JDBC ドライバー接続プロパティー
- データベース URL
- イベント順序
- イベント・テーブル名
- パスワード
- ポーリング前に実行するストアード・プロシージャー (Stored procedure to run before polling)
- ポーリング後に実行するストアード・プロシージャー (Stored procedure to run after polling)
- ユーザー名

ビジネス・オブジェクトのアプリケーション固有情報

双方向スクリプト・データ変換で使用可能なビジネス・オブジェクトのアプリケーション固有情報パラメーターを以下に示します。

- TableName
- StatusColumnName
- SPName
- SelectStatement

操作に関するアプリケーション固有情報

双方向スクリプト・データ変換で使用可能な操作のアプリケーション固有情報パラメーターを以下に示します。

- StoredProcedureName
- Parameters の PropertyName

属性に関するアプリケーション固有情報

双方向スクリプト・データ変換で使用可能な属性のアプリケーション固有情報パラメーターを以下に示します。

- ColumnName

フォールト・ビジネス・オブジェクト

アダプターは、予想される例外で Outbound サービス記述で宣言されている例外であるビジネス・フォールトか、インポートをサポートします。ビジネス・フォールトは、ビジネス・ルールの違反または制約違反が原因で、ビジネス・プロセスの予測可能なポイントに発生します。

IBM WebSphere Adapter for JDBCにより、フォールトが使用可能になります。フォールトを手動で構成する必要はありません。WBIFault ビジネス・オブジェクトには、フォールトの処理に必要な情報が含まれています。

アダプターには、ウィザードが作成する以下のフォールト・ビジネス・オブジェクトがあります。

- IntegrityConstraintFault

アダプターは、Create、 Update、UpdateAll、DeleteAll、Execute、および Merge の各操作を処理するときに、データベース が保全性制約違反に関する SQLException 例外を返すと、このフォールトを戻します。例えば外部キーが見つからない場合、アダプターはこのフォールトを返します。

- MatchesExceededLimitFault

  RetrieveAll 操作を処理する場合に、データベース照会から返されたレコード数が、対話仕様内の MaxRecords プロパティーの値を超えると、アダプターはこのフォールトを返します。

  返されるレコード数を増やすには、RetrieveAll 操作の対話仕様プロパティー内の MaxRecords プロパティーの値を増やします。

  このフォールトのビジネス・オブジェクトには 1 つのプロパティー matchCount のみがあり、このプロパティーは一致した数が含まれているストリングです。

- MissingDataFault

  Outbound 操作に渡されたビジネス・オブジェクトに必要なすべての属性が揃っていない場合、アダプターはこのフォールトを返します。このフォールトは、Create、Delete、Update、Retrieve、ApplyChanges、Merge および Exists の各操作で発生する可能性があります。

- MultipleMatchingRecordsFault

  Retrieve、Merge、ApplyChanges、または Update 操作の処理時に、照会で指定されたキーのレコードが複数返された場合、アダプターによりこのフォールトが返されます。このフォールトのビジネス・オブジェクトには 1 つのプロパティー matchCount があり、このプロパティーは一致した数が含まれるストリングです。

- ObjectNotFoundFault

  アダプターは、Create、Update、 ApplyChanges、および Merge 操作を処理するときに、単一カーディナリティーの子オブジェクトを、子オブジェクトの所有権が false であれば取得します。その取得処理でオブジェクトが見つからない場合、アダプターはこのフォールトを戻します。

- RecordNotFoundFault

  データ検索時に、指定されたキーのレコードがデータベース内で見つからなかった場合、アダプターはこのフォールトを返します。このフォールトは、Retrieve、RetrieveAll、Delete、Merge、 ApplyChanges、Execute、および Update の各操作で発生する可能性があります。

- UniqueConstraintFault

  Create、Merge、ApplyChanges、 Execute、UpdateAll、または Update の各操作の処理時に、固有制約違反のために データベースから SQLException 例外を受け取った場合、アダプターによりこのフォールトが返されます。

アダプター・メッセージ

WebSphere Adapter for JDBC によって送出されたメッセージを以下の場所で表示します。

メッセージのリンク先は http://bidoc.torolab.ibm.com:750/help/topic/com.ibm.wbpm.ref.doc/topics/welc_ref_msg_wbpm.html です。

表示される Web ページには、メッセージ・プレフィックスのリストがあります。メッセージ・プレフィックスをクリックすると、以下に示すように、そのプレフィックスがあるすべてのメッセージを参照できます。

- プレフィックス CWYBC があるメッセージの送出元は WebSphere Adapter for JDBC です。
- プレフィックス CWYDB を持つメッセージは、WebSphere Adapter for JDBC と WebSphere Adapter for Oracle E-Business Suite で共用される共通コンポーネントによって発行されます。
- プレフィックス CWYBS があるメッセージの送出元はアダプター・ファウンデーション・クラスで、これらのクラスはすべてのアダプターによって使用されます。

関連情報

以下のインフォメーション・センター、IBM Redbooks、および Web ページには、WebSphere Adapter for JDBC の関連情報が記載されています。

情報リソース

- WebSphere Business Process Management の情報リソース Web ページ (http://www14.software.ibm.com/webapp/wsbroker/redirect?version=pix&product=wps-dist&topic=bpmroadmaps) には、各種の記事、Redbooks、資料、および研修用資料へのリンクが掲載されており、WebSphere Adapters を習得するのに役立ちます。
- WebSphere Adapters ライブラリーのページ (http://www.ibm.com/software/integration/wbiadapters/library/infocenter/) には、資料の全バージョンへのリンクが組み込まれています。

関連製品の情報

- IBM Business Process Manager、バージョン 7.5、インフォメーション・センター (http://bidoc.torolab.ibm.com:7500/help/index.jsp)。ここには、IBM Business Process Manager、IBM WebSphere Enterprise Service Bus、および IBM Integration Designer の情報が記載されています。
- IBM Business Process Manager、バージョン 7.0、インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp)。ここには、IBM Business Process Manager、IBM WebSphere Enterprise Service Bus、および IBM Integration Designer の情報が記載されています。
- WebSphere Adapters、バージョン 6.2.x、インフォメーション・センター: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp
- WebSphere Application Server バージョン 8.0 への IBM WebSphere Adapters バージョン 7.5 のインストールに関する情報: http://www-01.ibm.com/support/docview.wss?rs=695&uid=swg27011040

developerWorks® リソース

- WebSphere Adapter Toolkit
- WebSphere Business Integration ゾーン (WebSphere business integration zone)

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒242-8502
神奈川県大和市下鶴間1623番14号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Department 2Z4A/SOM1
294 Route 100
Somers, NY 10589-0100
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向性および指針に関するすべての記述は、予告なく変更または撤回される場合があります。これらは目標および目的を提示するものにすぎません。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを

経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物にも、次のように、著作権表示を入れていただく必要があります。(c) (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 (c) Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告:

診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。 現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

この製品には、Eclipse Project (http://www.eclipse.org) により開発されたソフトウェアが含まれています。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

A

B

C

D

E

F

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21