IBM WebSphere Business Integration
Adapters

Adapter for JDBC ユーザーズ・ガイド

V 2.4.x

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、 143 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、新しい版で明記されていない限り、Adapter for JDBC バージョン 2.4.x および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。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原 典: IBM WebSphere Business Integration Adapters
Adapter for JDBC User Guide
V 2.4.x

発 行: 日本アイ・ビー・エム株式会社

担 当: ナショナル・ランゲージ・サポート

第 1 刷 2004.1

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

図

本書について

IBM® WebSphere® Business Integration Adapter ポートフォリオは、先進の e-business テクノロジー、エンタープライズ・アプリケーション、およびレガシー/メインフレーム・システムを統合的に接続する機能を提供します。本製品には、コンポーネントをカスタマイズ、作成、および管理するためのツールとテンプレートが含まれており、これにより、ビジネス・プロセスの統合を実現します。

本書では、JDBC のアダプターのインストール、構成、およびビジネス・オブジェクト開発について説明します。

対象読者

本書は、コネクターをお客様のサイトで使用するコンサルタント、デベロッパー、およびシステム管理者を対象としています。

関連文書

この製品に付属する資料の完全セットで、すべての WebSphere Business Integration Adapters のインストールに共通な機能とコンポーネントについて説明します。また、特定のコンポーネントに関する参考資料も含まれています。

以下のサイトから、関連資料をインストールすることができます。

- アダプターの一般情報、WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) でのアダプターの使用、WebSphere Application Server でのアダプターの使用については、次の IBM WebSphere Business Integration Adapters InfoCenter をご覧ください。
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- WebSphere InterChange Server でのアダプターの使用については、次の IBM WebSphere InterChange Server InfoCenter をご覧ください。
  http://www.ibm.com/websphere/integration/wicserver/infocenter
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- WebSphere Message Brokers の詳細については、以下をご覧ください。
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- WebSphere Application Server の詳細については、以下をご覧ください。
  http://www.ibm.com/software/webservers/appserv/library.html

これらのサイトでは、資料のダウンロード、インストール、および表示方法を簡単に説明しています。

表記上の規則

本書では、以下のような規則を使用しています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
イタリック	変数名または相互参照を示します。
青のアウトライン	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
ProductDir	IBM WebSphere Business Integration Adapters 製品のインストール・ディレクトリーを表します。デフォルトの製品ディレクトリーは、WebSphereAdapters です。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
|	構文の記述行の場合、パイプ記号 | は、選択対象のオプションの区切りです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って入力できることを意味します。
< >	不等号括弧は名前の個々の要素を囲み、各要素を区別します (例: `<server_name><`*`connector_name`*`>tmp.log`)。
/、¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。すべての製品のパス名は、使用システムで JDBC 用コネクターがインストールされたディレクトリーを基準とした相対パス名です。
UNIX: および Windows:	これらのいずれかで始まるパラグラフは、オペレーティング・システム間の相違を列挙した注記です。
%text% および *$text*	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、UNIX 環境変数 *text* の値を示します。

本リリースの新機能

リリース 2.4.x の新機能

2003 年 12 月更新。

バージョン 2.1.0 から、Adapter for JDBC は Microsoft Windows NT 上ではサポートされなくなりました。

アダプターのバージョン 2.4.x に対応した本書のリリースでは、次の新規情報または訂正情報が追加されました。

- 第 2 章に掲載されていたコネクターのインストールに関する情報が除去されました。該当の情報の新しい掲載先については、この章を参照してください。
- 第 2 章の『コネクターの構成』に、トラステッド認証を使用する場合はコネクター固有の構成プロパティー ApplicationPassword と ApplicationUserName が不要であることが追記されました。
- 第 3 章の『ビジネス・オブジェクトの動詞の処理』で、DeltaUpdate 操作に関する説明が追加され、Delete 操作に関する説明が変更されました。
- 第 3 章の『ビジネス・オブジェクトのアプリケーション固有の情報』に、CLOB データ型の定義に関する説明が追加されました。
- 第 3 章の上記と同じセクションの『ビジネス・オブジェクトの固有 ID の生成』に、IBM DB2 に関する情報が追加されました。
- 第 4 章の『ノードの展開と表およびビューの選択、ビューおよびストアード・プロシージャー』に、ストアード・プロシージャーに関する情報が追加されました。また、『追加情報の入力』に、ストアード・プロシージャーの属性に関する詳細情報が追加されました。

リリース 2.3.x の新機能

2003 年 7 月更新。アダプターのバージョン 2.3.x に対応した本書のリリースでは、次の新規情報または訂正情報が追加されました。

- アダプターは、WebSphere Application Server を統合ブローカーとして使用できるようになりました。
- アダプターは、以下のプラットフォーム上で実行されるようになりました。
  - HP-UX11i
  - AIX 5.x
  - Solaris 7 および 8
- Oracle のストアード・プロシージャーからの結果セットの戻りをサポートするようになりました。
- CLOB データ型がサポートされるようになりました。
- コピー属性に対する親の親 (祖父母) からのアクセスがサポートされるようになりました。コピー属性に対して親からのアクセスが可能になりました。この結果、ビジネス・オブジェクト階層の下方向に属性を伝播できます。

- eventid は数値データ型でなければならないという制限が取り除かれました。

リリース 2.2.x の新機能

2003 年 3 月更新。今後、「CrossWorlds」という名前は、システム全体を表したり、コンポーネント名やツール名を修飾する目的では使用されません。それ以外の点では、従来とほぼ同じです。例えば、「CrossWorlds System Manager」は今後「System Manager」に、「CrossWorlds InterChange Server」は「WebSphere InterChange Server」になります。

コネクターのバージョン 2.2.x では、次の新規情報または訂正情報が本書に追加されました。

- 以下に対するサポートが追加されました。
  - ビジネス・オブジェクトの最上位にあるラッパー・オブジェクト
  - LIKE 演算子
  - 16 進/バイナリー・データ
  - RetrieveUpdate 動詞のストアード・プロシージャー
  - RetrieveByContent 用の動詞に関するアプリケーション固有情報
  - RetrieveByContent の WHERE 文節の長さが 0 の場合の、WHERE 文節内の動詞に関するアプリケーション固有情報
- ConnectorID プロパティーが int から String に変更されたため、より記述的な名前を使用できるようになりました。
- カスタム JDBC ドライバーによって使用されるネイティブ・ライブラリーを指すため、DRIVERLIB 変数が追加されました。
- オブジェクト処理中のデータベース接続の喪失を検査するための機能が追加されました。
- イベントの検索およびアーカイブ時に、Schema Name プロパティーが使用されるようになりました。
- サンプル・ビジネス・オブジェクト、RetrieveResultSet_SampleBOwithMcardChild および RetrieveResultSet_SampleBOwithScardChild (¥connectors¥JDBC¥Samples ディレクトリーに格納されています) が追加されました。

リリース 2.1.x の新機能

これまでコネクターにインストールされていた orjdbcobjconverter.pl という Perl スクリプト (Oracle オブジェクトを JDBC 固有のオブジェクトに変換するためのスクリプト) はインストールされません。その代わり、このスクリプトは、IBM eCare サポートの Web サイト (http://www.ibm.com/software/integration/cw/support) からダウンロードできるようになっています。

リリース 2.0.x の新機能

コネクターは国際化に対応しています。詳細については、6 ページの「ロケール依存データの処理」、および『付録 A. コネクターの標準構成プロパティー』を参照してください。

リリース 1.9.x の新機能

IBM WebSphere Business Integration Adapter for JDBC は JDBC 用コネクターを含みます。このアダプターは、InterChange Server (ICS) および WebSphere MQ Integrator の 2 つの統合ブローカーをサポートします。統合ブローカーは異機種のアプリケーション間の統合を実現するアプリケーションであり、データ・ルーティングなどのサービスを提供します。

IBM WebSphere Business Integration Adapter for JDBC は以下のものを含みます。

- JDBC に固有のアプリケーション・コンポーネント
- JDBCODA
- サンプル・ビジネス・オブジェクト (¥connectors¥JDBC¥Samples ディレクトリーにあります)
- IBM WebSphere Adapter Framework。以下から構成されます。
  - コネクター・フレームワーク
  - 開発ツール (Business Object Designer、Connector Configurator など)
  - API (ODK、JCDK、CDK など)

本書では、ICS および WebSphere MQ Integrator 統合ブローカーの両方でのアダプターの使用について説明します。

注: コネクターは国際化に対応していないため、ISO Latin-1 データのみが処理されるという保証がない場合には、このコネクターを InterChange Server バージョン 4.1.1 とともに実行しないでください。

リリース 1.8.x の新機能

コネクターのビジネス・オブジェクト定義を生成する JDBCODA が拡張され、その資料も改訂されました。77 ページの『第 4 章 JDBCODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

リリース 1.7.x の新機能

CrossWorlds をインストールすると、IBM ブランドの MS SQL Server 用 JDBC ドライバーが、WebLogic JDBC ドライバーの代わりに使用されるようになりました。Oracle シン・ドライバーも引き続き提供されています。

リリース 1.6.x の新機能

コネクターのバージョン 1.6.x では、次の新規情報または訂正情報が本書に追加されました。

- コネクター用のビジネス・オブジェクトを作成するため、オブジェクト・ディスカバリー・エージェント・ユーティリティーが開発されました。このユーティリティーは、以前のリリースのコネクターで提供されていた、JDBCBORGEN の代わりに使用するものです。77 ページの『第 4 章 JDBCODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

- CheckForEventTableInInit プロパティーの説明が、 19 ページの『コネクター固有のプロパティー』に追加されました。
- CloseDBConnection プロパティーの説明が、 19 ページの『コネクター固有のプロパティー』に追加されました。
- SPBeforePollCall プロパティーの説明が、 19 ページの『コネクター固有のプロパティー』に追加されました。
- 結果セットを戻すストアード・プロシージャーに対するサポートが追加されました。 58 ページの『ビジネス・オブジェクトの Retrieve 操作』および 59 ページの『ビジネス・オブジェクトの RetrieveByContent 操作』を参照してください。
- 固定長のストリング属性に対するサポートが追加されました。 66 ページの表 11 の説明を参照してください。
- カスタム BO ハンドラーのサポートが追加されました。

リリース 1.5.x の新機能

コネクターのバージョン 1.5.x では、次の新規情報または訂正情報が本書に追加されました。

- ArchiveTableName プロパティーの説明が更新されました。詳細については、 22 ページの『ArchiveTableName』を参照してください。
- AutoCommit プロパティーが、 19 ページの『コネクター固有のプロパティー』に追加されました。
- DateFormat プロパティーが、 19 ページの『コネクター固有のプロパティー』に追加されました。
- EventKeyDel プロパティーが更新され、名前と値のペアに対応しました。詳細については、 25 ページの『EventKeyDel』を参照してください。
- EventQueryType プロパティーが、 19 ページの『コネクター固有のプロパティー』に追加されました。
- PingQuery プロパティーが、 19 ページの『コネクター固有のプロパティー』に追加されました。
- PreserveUIDSeq プロパティーが、 19 ページの『コネクター固有のプロパティー』に追加されました。
- SchemaName プロパティーが、 19 ページの『コネクター固有のプロパティー』に追加されました。
- RetrieveByContent 動詞のストアード・プロシージャーがサポートされました。詳細については、 59 ページの『ビジネス・オブジェクトの RetrieveByContent 操作』を参照してください。
- イベント処理用に、非キー値に基づくビジネス・オブジェクト検索に対するサポートが追加されました。詳細については、 5 ページの『イベント処理用ビジネス・オブジェクトの検索』を参照してください。

第 1 章 コネクターの概要

この章では、IBM WebSphere Business Integration Adapter for JDBC のコネクター・コンポーネントについて説明します。

この章には、以下のセクションが含まれています。

- 『コネクター・コンポーネント』
- 2 ページの『コネクターの動作方法』

コネクター・コンポーネント

コネクターは、**コネクター・フレームワーク**と**アプリケーション固有のコンポーネント**の 2 つのパーツで構成されています。

コネクター・フレームワークのコードはすべてのコネクターに共通なので、コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの仲介役の機能を果たします。

アプリケーション固有のコンポーネントには、特定のアプリケーションまたはテクノロジー (この場合は JDBC) に合わせて作成されたコードが含まれます。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間で、以下のサービスを提供します。

- ビジネス・オブジェクトの受信と送信
- 始動メッセージや管理メッセージの交換の管理

JDBC 用コネクターを使用すると、統合ブローカーと、JDBC 2.0 以上の仕様に準拠したドライバーによってサポートされているデータベース上に構築されたアプリケーションとの間で、ビジネス・オブジェクトの交換が可能になります。このセクションでは、コネクターのアーキテクチャー、および コネクターによるさまざまな JDBC ドライバーの使用について、概説します。

コネクターがデータベースへの接続に使用するドライバーの指定については、 17 ページの『マルチドライバー・サポートの使用可能化』を参照してください。

コネクターは、JDBC Connect 機構を使用して、アプリケーションのデータベースに接続します。コネクター固有の構成パラメーターの 1 つ (`DatabaseURL`) を使用して、コネクターの接続先となるデータベース・サーバーの名前を指定できます。構成パラメーターについては、 18 ページの『コネクターの構成』を参照してください。

コネクターは、開始されると、データベースとの間に接続プールを確立します。そして、データベースとの間でのトランザクション処理のすべてに、このプール内の接続を使用します。コネクターの終了時には、プール内の全接続がクローズされます。

コネクター・アーキテクチャー

図 1 に、コネクターのコンポーネントおよび Business Integration システムでの関係を示します。

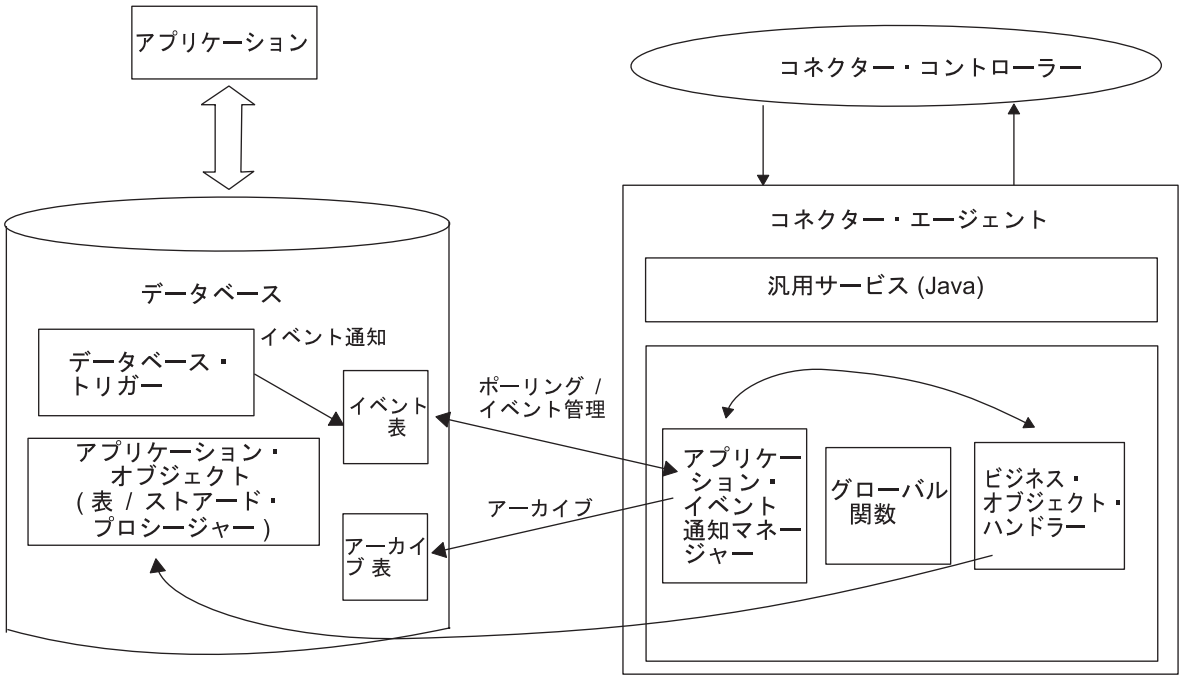


図 1. ビジネス・オブジェクト要求アーキテクチャー

コネクターの動作方法

このセクションでは、メタデータによってコネクターの柔軟性がどのように拡張されるかについて説明し、ビジネス・オブジェクトの処理およびイベント通知について概説します。

コネクターおよびメタデータ

コネクターは、メタデータ主導型です。IBM WebSphere Business Integration Adapter 環境では、**メタデータ**とは、コネクターとアプリケーションの相互作用を支援する、ビジネス・オブジェクトに格納されたアプリケーション固有のデータです。メタデータ主導型コネクターは、サポートする各ビジネス・オブジェクトを処理する際に、コネクター内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義にエンコードされたメタデータに基づいて処理を行います。

ビジネス・オブジェクトのメタデータには、そのビジネス・オブジェクトの構造の他、そのビジネス・オブジェクトにおける属性プロパティーの設定やアプリケーション固有情報の内容が組み込まれています。コネクターはメタデータ主導型であるため、新規の、または変更されたビジネス・オブジェクトを、コネクター・コードの変更を必要とせずに処理することができます。

コネクターは、SQL ステートメントまたはストアード・プロシージャーを実行して、データベースやアプリケーションに含まれるデータを検索または変更します。コネクターでは、動的 SQL ステートメントまたはストアード・プロシージャーの作成のために、アプリケーション固有のメタデータが使用されます。作成された SQL ステートメントおよびストアード・プロシージャーは、コネクターが処理するビジネス・オブジェクトおよび動詞に必要な、データベースやアプリケーションの検索または変更を実行します。アプリケーション固有の情報の使用については、37 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』を参照してください。

ビジネス・オブジェクトの処理

このセクションでは、コネクターが、ビジネス・オブジェクトからの要求やアプリケーション・イベントをどのように処理するかについて概説します。詳細については、43 ページの『ビジネス・オブジェクト動詞の処理』を参照してください。

ビジネス・オブジェクト要求の処理

コネクターは、アプリケーション操作実行の要求を受けると、階層ビジネス・オブジェクトを再帰的に処理します。つまり、すべての個別ビジネス・オブジェクトを処理するまで、子ビジネス・オブジェクトのそれぞれに対し、同じステップを実行します。コネクターが子ビジネス・オブジェクトおよび最上位ビジネス・オブジェクトを処理する順序は、それらの子ビジネス・オブジェクトが所有関係にあるかどうか、および単一カーディナリティーの関係と複数カーディナリティーの関係のどちらにあるかによって異なります。

注: **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、単一のビジネス・オブジェクトを指します。そのビジネス・オブジェクトの子オブジェクトや、そのビジネス・オブジェクトが属する子ビジネス・オブジェクトは含みません。**最上位**ビジネス・オブジェクトという用語は、階層の頂点にある、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。

ビジネス・オブジェクトの検索: コネクターは、統合ブローカーからデータベース内のある階層ビジネス・オブジェクトを検索するよう要求されると、そのビジネス・オブジェクトの現在のデータベース表現に厳密に一致するビジネス・オブジェクトを戻そうとします。つまり、統合ブローカーに戻された各個別ビジネス・オブジェクトの単純属性はすべて、データベース内の対応するフィールドの値に一致します。また、戻されたビジネス・オブジェクトに含まれる各配列での個別ビジネス・オブジェクトの数は、その配列のデータベース内の子の数に一致します。

コネクターは、このような検索を実行する場合には、最上位ビジネス・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データを再帰的に順次検索します。

ビジネス・オブジェクトの内容による検索: コネクターは、統合ブローカーから、最上位ビジネス・オブジェクトの非キー属性に基づいて階層ビジネス・オブジェクトを検索するように要求されると、すべての非ヌル属性の値をデータ検索基準として使用します。

ビジネス・オブジェクトの作成: 統合ブローカーから、データベース内に階層ビジネス・オブジェクトを作成するように要求されると、コネクターは以下のステップを実行します。

1. 所有関係を伴う単一カーディナリティーの子ビジネス・オブジェクトをそれぞれ、データベース内に再帰的に作成します。
2. 所有関係を伴わない単一カーディナリティーの子ビジネス・オブジェクトをそれぞれ処理します。
3. 最上位ビジネス・オブジェクトを、データベース内に作成します。
4. 単一カーディナリティーの子ビジネス・オブジェクトのうち、親/子関係を子に保管するものをそれぞれ作成します。
5. 複数カーディナリティーの子ビジネス・オブジェクトをそれぞれ作成します。

ビジネス・オブジェクトの変更: 統合ブローカーから、データベース内の階層ビジネス・オブジェクトを更新するように要求されると、コネクターは以下のステップを実行します。

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。
2. 最上位ビジネス・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。
3. 関係を親に保管する単一カーディナリティーの子ビジネス・オブジェクトに関しては、親に存在する外部キー値のそれぞれを、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。
4. 検索されたビジネス・オブジェクトの単純属性のすべてを更新します。ただし、ソース・ビジネス・オブジェクト内の対応する属性に値 CxIgnore が含まれるものを除きます。
5. 親/子関係を子に保管する子ビジネス・オブジェクト (複数カーディナリティーであるか、単一カーディナリティーであるかを問いません) のそれぞれにおいて、外部キー値のすべてを、対応する親ビジネス・オブジェクトの基本キー値に設定します。
6. 検索されたビジネス・オブジェクトの配列のすべてを処理します。

ビジネス・オブジェクトの削除: 統合ブローカーから、データベースから階層ビジネス・オブジェクトを削除するように要求されると、コネクターは以下のステップを実行します。

1. 単一カーディナリティーの子を削除します。
2. 複数カーディナリティーの子を削除します。
3. 最上位ビジネス・オブジェクトを削除します。

アプリケーション・イベントの処理

コネクターは、アプリケーションによって生成された Create、Update、および Delete の各イベントを、下記の方法で処理します。

Create 通知: コネクターは、イベント表内に Create イベントを見つけると、そのイベントによって指定されたタイプのビジネス・オブジェクトを作成し、そのビジネス・オブジェクトのキー値を設定して (このとき、イベント表に指定されている

キーが使用されます)、データベース内でそのビジネス・オブジェクトを検索します。目的のビジネス・オブジェクトが検索されると、コネクターは Create 動詞とともに、統合ブローカーに送信します。

Update 通知: コネクターは、イベント表内に Update イベントを見つけると、そのイベントによって指定されたタイプのビジネス・オブジェクトを作成し、そのビジネス・オブジェクトのキー値を設定して (このとき、イベント表に指定されているキーが使用されます)、データベース内でそのビジネス・オブジェクトを検索します。目的のビジネス・オブジェクトが検索されると、Update 動詞とともに、統合ブローカーに送信します。

Delete 通知: コネクターは、イベント表内に Delete イベントを見つけると、そのイベントによって指定されたタイプのビジネス・オブジェクトを作成し、そのビジネス・オブジェクトのキー値を設定して (イベント表に指定されているキーが使用されます)、Delete 動詞とともに統合ブローカーに送信します。キー値以外の値は、すべて CxIgnore に設定されます。非キー・フィールドのいずれかが使用サイトで有意である場合には、必要に応じてそのフィールドの値を変更してください。

コネクターは、アプリケーションによって起動される論理 Delete 操作および物理 Delete 操作を処理します。物理削除の場合、SmartFiltering 機構により、ビジネス・オブジェクトの未処理イベント (Create や Update など) がすべて除去されてから、Delete イベントがイベント表に挿入されます。論理削除の場合、コネクターによって Delete イベントがイベント表に挿入されます。ビジネス・オブジェクトのその他のイベントが除去されることはありません。

イベント処理用ビジネス・オブジェクトの検索: イベント処理のためのビジネス・オブジェクト検索は、2 とおりの方法で実行することができます。第 1 の方法は、ビジネス・オブジェクトのキー属性に基づく検索です。第 2 の方法は、キー属性および非キー属性の両方に基づく検索です。この場合、ビジネス・オブジェクトで `RetrieveByContent` 動詞がサポートされ、オブジェクト・キーに `name_value` ペアが使用されていなければなりません。

注: オブジェクト・キーに `name_value` ペアが使用されていない場合、オブジェクト・キー・フィールド内のキーの順序は、ビジネス・オブジェクト内のキーと同じ順序でなければなりません。

イベント通知

コネクターのイベント検出機構には、イベント表、アーカイブ表、ストアード・プロシージャー、およびデータベース・トリガーが使用されています。イベント管理プロセスは、アーカイブ表へのイベントの挿入を完了するまで、イベント表からイベントを削除しません。これは、イベント処理に関連して、障害が発生する可能性がある点がいくつかあるためです。

データベース・トリガーは、データベース内で特定のイベントが発生すると、イベント表にイベントを格納します。コネクターは、一定間隔 (変更可能) でこの表に対してポーリングを実行し、イベントを検索して処理します。処理は、まず優先順位に従って実行され、次に順次実行されます。コネクターがこのイベント処理を完了すると、イベントの状況が更新されます。

注: インストール手順の一部として、データベースにトリガーを追加する必要があります。

コネクターの ArchiveProcessed プロパティーの設定によって、コネクターが、イベントの状況を更新した後でそのイベントをアーカイブ表にアーカイブするかどうかが決定されます。ArchiveProcessed プロパティーに関する詳細については、18 ページの『コネクターの構成』を参照してください。

表 1 に、ArchiveProcessed プロパティーの設定に応じたアーカイブの振る舞いを示します。

表 1. アーカイブ時の振る舞い

アーカイブ処理済み設定	イベント表から削除される理由	コネクターの振る舞い
true または値なし	処理成功	「Sent to InterChange」状況でアーカイブ済み
	処理失敗	「Error」状況でアーカイブ済み
	ビジネス・オブジェクトに対するサブスクリプションがない	「Unsubscribed」状況でアーカイブ済み
false	処理成功	アーカイブせずにイベント表から削除します。
	処理失敗	状況を Error にしてイベント表に残します。
	ビジネス・オブジェクトに対するサブスクリプションがない	状況を Unsubscribed にしてイベント表に残します。

SmartFiltering は、統合ブローカーおよびコネクターによって実行される処理の量を最小化する、データベース・トリガー内機構です。例えば、コネクターによる前回のイベント・ポーリングの後で、Contract ビジネス・オブジェクトがあるアプリケーションによって 15 回更新されている場合、SmartFiltering は、これらの変更を単一の Update イベントとして保管します。

データベース接続不能の処理

データベース接続は、さまざまな理由で失われます。データベース接続が失われると、コネクターは終了します。JDBC の仕様では、接続の喪失を検出する機構は定められていません。コネクターでは、サポートされているデータベースが多岐にわたるため、データベース接続の喪失に応じたエラー・コード定義は一切用意されていません。

PingQuery プロパティーは、この検出を処理するために提供されています。サービス呼び出し要求中に障害が発生すると、コネクターは、この PingQuery を実行して、データベース接続の喪失が原因で、その障害が発生したのではないことを確認します。PingQuery が失敗した場合に AutoCommit プロパティーが false に設定されていると、コネクターはデータベースへの新規の接続を作成しようとします。データベースへの新規接続の作成に成功した場合、コネクターは処理を続行します。失敗した場合、コネクターは APPRESPONSETIMEOUT を戻します。この結果、コネクターは終了します。

データベースへのアクセス中に障害が発生した場合は、トランザクションのタイプには関係なく PingQuery が実行されます。以下に例を示します。

- イベント表およびアーカイブ表にアクセスしているとき。
- イベントに関連するビジネス・オブジェクトを検索しているとき。
- ビジネス・オブジェクトに関連するレコードを作成または更新しているとき。

ロケール依存データの処理

コネクターは、2 バイト文字セットをサポートして、指定された言語でメッセージ・テキストを送れるように国際化されています。ある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所へデータを転送する場合、コネクターは、そのデータの意味が保持されるように文字変換を実行します。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、最もよく知られた文字コード・セット (単一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。IBM WebSphere Business Integration システムのほとんどのコンポーネントは Java で書かれています。そのため、WebSphere Business Integration システム・コンポーネント間でデータを転送するときは、ほとんどの場合文字変換は必要ありません。

エラー・メッセージや情報メッセージを個々の国や地域に合った適切な言語で記録するには、個々の環境に合わせて Locale 標準構成プロパティーを構成する必要があります。これらのプロパティーの詳細については、103 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

第 2 章 コネクターのインストールと構成

この章では、IBM WebSphere Business Integration Adapter for JDBC のインストール方法および構成方法と、アプリケーションをコネクターとともに動作させるための構成方法について説明します。この章の内容は、次のとおりです。

アダプター環境

アダプターをインストール、構成、使用する前に、環境要件を理解しておく必要があります。環境要件は、以下のセクションでリストされています。

ブローカーの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for JDBC バージョン 2.4.x は、以下のアダプター・フレームワークと統合ブローカーでサポートされています。

- **アダプター・フレームワーク**:
  WebSphere Business Integration Adapter Framework バージョン 2.1、2.2、2.3.x、および 2.4。
- **統合ブローカー:**
  - WebSphere InterChange Server、バージョン 4.1.1、4.2、4.2.1、4.2.2
  - WebSphere MQ Integrator、バージョン 2.1.0
  - WebSphere MQ Integrator Broker、バージョン 2.1.0
  - WebSphere Business Integration Message Broker、バージョン 5.0
  - WebSphere Application Server Enterprise、バージョン 5.0.2 (WebSphere Studio Application Developer Integration Edition バージョン 5.0.1 と併用)

例外については、『*リリース情報*』を参照してください。

注: 統合ブローカーおよびその前提条件のインストールに関する説明については、以下のガイドを参照してください。

WebSphere InterChange Server (ICS) については、「*IBM WebSphere InterChange Server システム・インストール・ガイド (UNIX 版)*」または「*IBM WebSphere InterChange Server システム・インストール・ガイド (Windows 版)*」を参照してください。

WebSphere Message Brokers については、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」を参照してください。

WebSphere Application Server については、「*IBM WebSphere Business Integration Adapters アダプター実装ガイド (WebSphere Application Server)*」を参照してください。

アダプターのプラットフォーム

アダプターは以下のソフトウェアでサポートされています。

オペレーティング・システム:

- AIX 4.3.3、AIX 5.1、AIX 5.2
- Solaris 7.0、Solaris 8.0
- Windows 2000

データベース:

JDBC ドライバーが提供されているデータベースのすべて

サード・パーティー製ソフトウェア:

- JDBC ドライバー

アダプターの依存関係

Adapter for JDBC を使用するには、以下のソフトウェアが必要です。

- JDBC ドライバー・ファイル

グローバリゼーション

このアダプターは DBCS (2 バイト文字セット) で使用可能であり、翻訳されています。

前提条件

コネクターを使用するには、次の操作を実行する必要があります。

- Adapter Development Kit をインストールします。

  コネクターを統合ブローカーとは別のマシンで実行する場合は、その統合ブローカーのバージョンと互換性のある Adapter Development Kit をインストールしてください。
- 使用される JDBC ドライバーをインストールします。
- 必要なベンダー固有のソフトウェア (JDBC ドライバーに必要なソフトウェアなど) がすべてインストール済みであることを確認します。

例えば、Oracle データベース用に JDBC Type 2 ドライバーを使用する場合は、Oracle OCI ライブラリーをインストールする必要があります。

- アプリケーションにユーザー・アカウントが存在することを確認します。

  コネクターは、JDBC の仕様に準拠したドライバーによってサポートされているデータベース上に構築されたアプリケーションのデータを処理します。コネクターは、データベースと直接対話してその内部のデータを処理するため、アプリケーションの有効なユーザー・アカウントとパスワードを使用できなければなりません。使用するユーザー・アカウントには、アプリケーションのデータベースのデータを検索、挿入、更新、および削除する権限が付与されていなければなりません。このようなアカウントが存在しない場合は、作成する必要があります。

- 接続されたデータベースの文字コード・セットを確認します。

  Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、最もよく知られた文字セット (単一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。コネクターは、Java で作成されているため、Unicode を認識します。

アダプターと関連ファイルのインストール

WebSphere Business Integration Adapter 製品のインストールについては、「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。この資料は、次の Web サイトの WebSphere Business Integration Adapters Infocenter にあります。

http://www.ibm.com/websphere/integration/wbiadapters/infocenter

インストール済みファイルの構造

以下のサブセクションでは、UNIX または Windows システムでのアダプターのインストール済みファイルの構造について説明します。

注: 特に指定がない限り、以降のセクションの内容は、コネクターのインストール先が UNIX システムでも、Windows システムでも適用されます。

UNIX システムへのインストール

JDBC アダプターを UNIX システムにインストールする方法については、「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。

12 ページの表 2 に、コネクターが使用する UNIX ファイル構造を示します。

表 2. コネクター用としてインストールされた UNIX ファイル構造

$*ProductDir* のサブディレクトリー	**説明**
connectors/JDBC	コネクターの CWJDBC.jar および start_JDBC.sh ファイルが格納されています。start_JDBC.sh ファイルは、コネクターのシステム始動スクリプトです。これは、汎用のコネクター・マネージャー・スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integrator Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターを ICS とともに使用する場合は、このカスタマイズされたラッパーを使用してコネクターを始動および停止してください。WebSphere MQ Integrator Broker でコネクターを使用する場合、このカスタマイズされたラッパーは、コネクターの始動のみに使用します。コネクターの停止には mqsiremotestopadapter コマンドを使用します。
connectors/JDBC/dependencies	イベント表、アーカイブ表、および固有 ID 表を作成する SQL スクリプトが格納されています。
connectors/messages	JDBCConnector.txt ファイルと、JDBCConnector_*ll_TT*.txt ファイル (言語に固有なメッセージ・ファイル (*ll*) と国/地域に固有なメッセージ・ファイル (*TT*)) が含まれます。
repository/JDBC	CN_JDBC.txt ファイルが格納されています。
connectors/JDBC/Samples	README_Samples.txt、およびさまざまなストアード・プロシージャーとビジネス・オブジェクトを作成するためのサンプル・ファイルが格納されています。
/lib	WBIA.jar ファイルが格納されています。
/bin	CWConnEnv.sh ファイルが格納されています。

コネクター・コンポーネントのインストール方法については、使用している統合ブローカーに応じて、以下のガイドのいずれかを参照してください。

- 「*システム・インストール・ガイド (UNIX 版)*」(ICS を統合ブローカーとして使用する場合)
- 「*IBM WebSphere Business Integration Adapters WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」(WebSphere MQ Integrator Broker を統合ブローカーとして使用する場合)

インストール済みファイル構造 (Windows システムの場合)

JDBC アダプターを Windows システムにインストールする方法については、「*WebSphere Business Integration Adapters インストール・ガイド*」を参照してください。表 3 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 3. コネクター用としてインストールされた Windows ファイル構造

%*ProductDir*% のサブディレクトリー	**説明**
connectors¥JDBC	コネクターの CWJDBC.jar および start_JDBC.bat ファイルが格納されています。

表 3. コネクター用としてインストールされた Windows ファイル構造 (続き)

%*ProductDir*% のサブディレクトリー	説明
connectors¥JDBC¥dependencies	イベント表、アーカイブ表、および固有 ID 表を作成する SQL スクリプトが格納されています。
connectors¥messages	JDBCConnector.txt ファイルと、JDBCConnector_*ll_TT*.txt ファイル (言語に固有なメッセージ・ファイル (*ll*) と国/地域に固有なメッセージ・ファイル (*TT*)) が含まれます。
repository¥JDBC	CN_JDBC.txt ファイルが格納されています。
connectors¥JDBC¥Samples	README_Samples.txt、およびさまざまなストアード・プロシージャーとビジネス・オブジェクトを作成するためのサンプル・ファイルが格納されています。
¥lib	WBIA.jar ファイルが格納されています。
¥bin	CWConnEnv.bat ファイルが格納されています。

インストーラーは、コネクター・ファイルを表すアイコンを「IBM WebSphere Business Integration Adapters」メニューに追加します。このファイルへのショートカットをデスクトップに作成することにより、コネクターの始動を速めることができます。

コネクター・コンポーネントのインストール方法については、使用している統合ブローカーに応じて、以下のガイドのいずれかを参照してください。

- 「*システム・インストール・ガイド (Windows 版)*」(ICS を統合ブローカーとして使用する場合)
- 「*IBM WebSphere Business Integration Adapters WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」(WebSphere MQ Integrator Broker を統合ブローカーとして使用する場合)

コネクター用のアプリケーションの使用可能化

コネクターでイベント引き渡しを処理できるようにするには、データベース内にイベント通知機構をセットアップする必要があります。これを行うには、次の操作を実行する必要があります。

- イベント表およびアーカイブ表をデータベース内に作成します。
- 必要なビジネス・オブジェクトをサポートするため、アプリケーションの表にデータベース・トリガーをインストールします。ユーザーが独自のデータベース・トリガーを開発することが前提となっています。
- カウンター表をインストールします (オプション)。このステップは、ビジネス・オブジェクトの作成時に、固有 ID 生成にコネクターが必要である場合に限り実行してください。固有 ID の生成の詳細については、UID=CW.uidcolumnname[=UseIfMissing] パラメーターを参照してください。

以下のセクションでは、イベント表およびアーカイブ表の作成と構成に関する情報を提供します。

イベント表およびアーカイブ表

コネクターは、イベント表を使用して選出するイベントをキューに入れます。ArchiveProcessed プロパティーを true または値なしに設定した場合、コネクターは、イベント表のイベントの状況を更新した後、アーカイブ表を使用してそれらのイベントを保管します。

コネクターは、イベントごとに、ビジネス・オブジェクト名、動詞、およびキーをイベント表から取得します。コネクターは、これらの情報を使用して、アプリケーション内の完全なエンティティーを検索します。イベントの初回記録後にそのエンティティーが変更された場合、コネクターは、その最初に記録されたイベントとそれ以後の変更のすべてを取得します。つまり、エンティティーの作成後、コネクターがイベント表からそのエンティティーを取得する前に更新が行われた場合には、コネクターは、これらの両方のデータ変更を一度の検索で取得します。

コネクターによって処理されるどのイベントについても、以下の 3 とおりの結果が考えられます。

- イベント処理の正常終了
- イベント処理の失敗
- イベントがサブスクライブされていない (使用している統合ブローカーに固有のサブスクリプション情報については、ブローカーのインプリメンテーション・ガイドを参照してください)

コネクターがイベントを選出した後、それらのイベントがイベント表から削除されなければ、それらのイベントは、イベント表内で不必要にスペースを使用することになります。しかし、これらのイベントが削除されると、未処理のイベントがすべて失われ、イベント処理の監査が不可能になります。このため、アーカイブ表を作成し、ArchiveProcessed プロパティーを true に設定しておいてください。これにより、イベント表からイベントが削除されると、必ず、コネクターによってアーカイブ表にそのイベントが挿入されます。

注: コネクターは、イベント表からイベントを削除しているとき、またはアーカイブ表にイベントを挿入しているときに、アプリケーションのデータベースへのアクセス中に発生した問題のために操作に失敗すると、APPRESPONSETIMEOUT を戻します。

イベント処理およびアーカイブ処理の構成

イベントとアーカイブの処理を構成するには、構成プロパティーを使用して、以下の情報を指定する必要があります。

- イベント表の名前 (EventTableName)。ビジネス・オブジェクトからの要求を処理するためだけにコネクターを使用する場合は、このプロパティーの値を指定する必要はありません。
- 間隔の頻度 (117 ページの『PollFrequency』)。
- ポーリング間隔ごとのイベントの数 (PollQuantity)。
- アーカイブ表の名前 (ArchiveTableName)。

- アンサブスクライブされたイベントおよび未処理のイベントをコネクターがアーカイブするかどうか (ArchiveProcessed)。使用している統合ブローカーに固有のサブスクリプション情報については、ブローカーのインプリメンテーション・ガイドを参照してください。
- コネクターの固有 ID。この ID は、複数のコネクターが同じ表をポーリングするときに重要になります (ConnectorID)。

また、EventOrderBy プロパティーの値を指定して、イベントの処理順序を指定することも可能です。これらのプロパティーおよび他の構成プロパティーに関する詳細については、103 ページの『付録 A. コネクターの標準構成プロパティー』および 19 ページの表 6 を参照してください。

注: イベント表とアーカイブ表の作成は、オプションです。ただし、EventTableName の値を指定しているにもかかわらず、コネクターを使用してイベントをポーリングせず、イベント表も作成しない場合には、コネクターでタイムアウトが発生します。このようなタイムアウトを防ぐには、EventTableName の値を null (ストリング) のままにしておきます。

デフォルトでは、イベント・キュー表の名前は xworlds_events であり、アーカイブ・キュー表の名前は xworlds_archive_events です。

コネクターを要求の処理のみに使用するには、コネクターを始動するときに -fno オプションを使用します。また、EventTableName の値を null (ストリング) に設定します。

使用ドライバーが Java クラス DatabaseMetaData をサポートしておらず、コネクターがイベント表とアーカイブ表の存在を確認しないようにする必要がある場合には、CheckForEventTableInInit の値を false に設定して、CheckForEventTableInInit を使用不可にします。デフォルトでは true です。この値は false に設定しないことをお勧めします。

注: ご使用のサイトで、アーカイブ表へイベントのアーカイブを行わない場合は、ArchiveProcessed の値を false に設定してください。

イベント表およびアーカイブ表をインストールするための SQL スクリプト

DB2 データベース用のイベント表、アーカイブ表、および固有 ID 表をインストールするためのスクリプトには、次のものがあります。

- event_table_db2.sql
- event_package_db2.sql
- archive_table_db2.sql
- uid_table_db2.sql

Oracle データベース用のイベント表、アーカイブ表、および固有 ID 表をインストールするためのスクリプトには、次のものがあります。

- event_table_oracle.sql
- event_package_oracle.sql
- archive_table_oracle.sql

- uid_table_oracle.sql

Microsoft SQL Server データベース用のイベント表、アーカイブ表、および固有 ID 表をインストールするためのスクリプトには、次のものがあります。

- event_table_mssqlserver.sql
- event_package_mssqlserver.sql
- archive_table_mssqlserver.sql
- uid_table_mssqlserver.sql

これらのファイルは、次のディレクトリーにあります。

UNIX:

connectors/JDBC/dependencies/

Windows:

connectors¥JDBC¥dependencies¥

注: これらのスクリプトは、テンプレートとしてのみ提供されています。その目的は、ユーザーがコネクターに必要な表を作成するのを支援することです。その他のデータベースについては、これらのスクリプトをガイドラインとして、独自のスクリプトを作成してください。表列の順序およびデータ型は、非常に重要です。正しい順序および型については、『イベント表およびアーカイブ表のスキーマ』を参照してください。

DBA、またはコネクターのインプリメント担当者には、これらのスクリプトを変更して、インストールおよび照会最適化に関する特定の要件に適合させることを推奨します。例えば、これらのスクリプトでは、表上に索引は作成されません。照会最適化プログラムのパフォーマンスを向上させるための索引作成は、コネクターのインプリメント担当者が行わなければなりません。

イベント表およびアーカイブ表のスキーマ

表 4 に、イベント表とアーカイブ表の列を示します。

表 4. イベント表およびアーカイブ表のスキーマ

名前	説明	型	制約
event_id	イベントの内部 ID	NUMBER	基本キー
connector_id	イベントの宛先コネクターの固有 ID。この値は、複数のコネクターが同一の表に対してポーリングする場合、重要です。	VARCHAR	
object_key	ビジネス・オブジェクトの基本キー。キーは、name_value ペア、またはコロンなどの構成可能な区切り文字で区切られた一連のキー (例: 1000065:10056:2333) として表すことができます。詳細については、25 ページの『EventKeyDel』 プロパティーを参照してください。	VARCHAR	null 以外
object_name	ビジネス・オブジェクト名	VARCHAR	null 以外
object_verb	イベントに関連付けられている動詞	VARCHAR	null 以外

表 4. イベント表およびアーカイブ表のスキーマ (続き)

名前	説明	型	制約
event_priority	コネクターがイベントを優先順位ベースで取得する場合に使用する、イベントの優先順位 (最高は 0、最低は *n*)。コネクターが、この値を使用して、優先順位を上下させることはありません。	NUMBER	null 以外
event_time	イベントの発生日時	DATETIME	デフォルト値は現在の日時 (アーカイブ表では、イベントの実際の発生時刻)
archive_time	イベントがアーカイブされた日時 (アーカイブ表のみ)	DATETIME	アーカイブ日時
event_status	-2 (統合ブローカーへのイベント送信時のエラー) -1 (イベント処理時のエラー) 0 (ポーリング可能) 1 (統合ブローカーに送信) 2 (ビジネス・オブジェクトに対するサブスクリプションなし) 3 (処理中)。この状況はイベント表でのみ使用され、アーカイブ表では使用されません。	NUMBER	null 以外
event_comment	イベント・ストリングまたはエラー・ストリングの説明	VARCHAR	

マルチドライバー・サポートの使用可能化

次の手順を実行することにより、ドライバーを指定することができます。

1. ドライバーをご使用のマシンにインストールします。
2. 製品ディレクトリーの下の connectors/JDBC ディレクトリーに、コネクターで実行時に必要なすべてのダイナミック・ライブラリーを置きます。
3. コネクターの始動ファイルを編集して、JDBCDRIVERPATH 変数に、適切なクラス・パス名をすべて組み込みます (必要に応じ、ライセンス情報も組み込みます)。

   UNIX 上の始動ファイル:

   ```
   $ProductDir/connectors/JDBC/start_JDBC.sh
   ```

   Windows 上の始動ファイル:

   ```
   %ProductDir%¥connectors¥JDBC¥start_JDBC.bat
   ```

4. JDBCDriverClass 構成プロパティーの値を指定します。

注: コネクターがサポートする機能のいずれに関しても、コネクターの動作上、JDBC 2.0 以上の仕様に準拠したドライバーが必要になります。ドライバーがいずれかの機能をサポートしていない場合、コネクターは正常に機能しません。例えば、ドライバーが JDBCODA によって使用されるすべてのメソッド呼び出しをサポートしない場合、JDBCODA ログにドライバーがサポートしないプロセスが示されます。この場合、別のドライバーを使用する必要があります。

カスタム・ビジネス・オブジェクト・ハンドラー・クラスの使用可能化

コネクターは、カスタム・ビジネス・オブジェクト・ハンドラー・クラス (CustomBOH) をサポートします。JDBCBOhandlerInterface インターフェースをインプリメントします。このインターフェースの構文は、次のとおりです。

```
public interface JDBCBOhandlerInterface{
   public int doVerbForCustom(CWConnectorBusObj busObj) throws
     VerbProcessingFailedException, ConnectionFailureException;
}
```

doVerbForCustom メソッドをインプリメントするときには、このメソッドが、次の 2 つの例外を、catch ではなく throw するようにしてください。また、それぞれの例外の throw の前に、例外の状況およびメッセージを設定してください。

- VerbProcessingFailedException: 動詞によって指定された操作が失敗した場合に throw されます。
- ConnectionFailureException: コネクターがアプリケーションとの間の接続を確立できない場合に throw されます。

コネクターで、このビジネス・オブジェクト・ハンドラーをサポート可能にするには、次の操作を実行します。

- 動詞アプリケーション固有の情報で CustomBOH クラス名を指定します。

  コネクターは、カスタム・ビジネス・オブジェクト・ハンドラー・クラス名を、動詞に関するアプリケーション固有情報から取得します。次の構文を使用してください。

  ```
  CustomBOH=customBOhandlerClassName
  ```

  例えば、動詞に関するアプリケーション固有情報が、次のように指定されているとします。

  ```
  CustomBOH=JDBCBOhandlerForOverrideSQL
  ```

  この場合、JDBCBOhandlerForOverrideSQL が、カスタム・ビジネス・オブジェクト・ハンドラー・クラス名です。
- CustomBOH が com.crossworlds.connectors.JDBC に属していることを確認します。

  コネクターは、"CustomBOH=" を動詞アプリケーション固有の情報の中に検出し、com.crossworlds.connectors.JDBC パッケージ内にクラスを検出した場合、カスタム・ビジネス・オブジェクトのハンドラーを実行します。CustomBOH が検出されなければ、指定のクラスを検出できなかったことを通知するエラーを throw します。

コネクターの構成

コネクターを実行するには、コネクターの標準構成プロパティーとコネクター固有の構成プロパティーを設定する必要があります。コネクターの構成プロパティーを設定するには、以下のツールのいずれかを使用します。

- Connector Configurator (ICS が統合ブローカーである場合) － このツールには、System Manager からアクセスします。

- Connector Configurator (WebSphere MQ Integrator Broker が統合ブローカーである場合) － このツールには、IBM WebSphere Business Integration Adapter プログラム・フォルダーからアクセスします。Connector Configurator に関する詳細については、 121 ページの『付録 B. Connector Configurator』を参照してください。

標準コネクター・プロパティー

標準の構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。これらのプロパティーに関する詳細については、 103 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

重要: JDBC 用コネクターは ICS と WebSphere MQ Integrator Broker の両方の統合ブローカーをサポートするため、このコネクターには、両方のブローカーに関する構成プロパティーが関係します。

さらに、IBM WebSphere Business Integration Adapter for JDBC に固有な構成情報については、表 5 を参照してください。この表に示されている情報は、付録に収録されている情報を補足するものです。

表 5. このコネクター固有のプロパティー情報

プロパティー	注
CharacterEncoding	CharacterEncoding プロパティーはこのコネクターによって使用されません。
Locale	このコネクターは国際化されているため、Locale プロパティーの値は変更可能です。 **注:** WebSphere MQ Integrator Broker をブローカーとして使用している場合は、アダプター、ブローカー、およびすべてのアプリケーションで同一のロケールを使用する必要があります。

コネクターを実行するには、`ApplicationName` 構成プロパティーに値を指定する必要があります。

コネクター固有のプロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。また、コネクター固有の構成プロパティーを使用すると、コネクターのコード変更や再ビルドを行わなくても、静的情報またはロジックを変更できます。

表 6 に、コネクターに対するコネクター固有の構成プロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

表 6. コネクター固有の構成プロパティー

名前	指定可能な値	デフォルト値	必須
ApplicationPassword	*コネクター・ユーザー・アカウントのパスワード*		`Yes*`
ApplicationUserName	*コネクター・ユーザー・アカウントの名前*		`Yes*`
ArchiveProcessed	`true` または `false`	`true`	いいえ

表 6. コネクター固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
ArchiveTableName	*アーカイブ・キュー表の名前*	xworlds_archive_events	ArchiveProcessed が true の場合は Yes
AutoCommit	true または false	false	いいえ
CheckforEventTableInInit	true または false	true	いいえ
ChildUpdatePhyDelete	true または false	false	いいえ
CloseDBConnection	true または false	false	いいえ
ConnectorID	*コネクターの固有 ID*	null	いいえ
DatabaseURL	*データベース・サーバーの名前*		はい
DateFormat	*時刻パターン・ストリング*	MM/dd/yyyy HH:mm:ss	いいえ
DriverConnectionProperties	*追加の JDBC ドライバー接続プロパティー*		いいえ
EventKeyDel	*イベント表のオブジェクト・キー列の区切り文字 (複数指定可能)*	semicolon (;)	いいえ
EventOrderBy	none または*列名*, *列名*, ...]		いいえ
EventQueryType	Fixed または Dynamic	Fixed	いいえ
EventTableName	*イベント・キュー表の名前*	xworlds_events	はい (ポーリングが必要な場合)。ポーリングが不要な場合はストリング null
JDBCDriverClass	ドライバー・クラス名		はい
MaximumDatabaseConnections	*同時に存在できるデータベース接続の数*	5	はい
PingQuery	SELECT 1 FROM <*tablename*>		いいえ
PollQuantity	値は 1 から 500	1	いいえ
PreserveUIDSeq	true または false	true	いいえ
RDBMS.initsession	*各データベース・セッションを初期化する SQL ステートメント*		いいえ
RDBMSVendor	MSSQLServer、Oracle、*その他*		はい
ReplaceAllStr	true または false	false	いいえ
ReplaceStrList	1 つの文字、1 つの文字区切り文字、および文字の置換ストリングで構成されたセット。または、このようなセットを、終了区切り文字で区切り複数指定したもの。	Q,DSQ 注: コネクター構成ツールでは、これらの文字は、1 つの単一引用符、1 つのコンマ、および 2 つの単一引用符の並びを表します。	いいえ
RetryCountAndInterval	*カウント, 秒単位の間隔*	3,20	いいえ
SchemaName	イベントが存在するスキーマ		いいえ
SPBeforePollCall	ポーリング呼び出しごとに実行されるストアード・プロシージャーの名前		いいえ

表 6. コネクター固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
StrDelimiter	ReplaceStrList プロパティーに使用される文字区切り文字および終了区切り文字	,:	いいえ
TimingStats	0、1、2	0	いいえ
UniqueIDTableName	*ID 生成に使用される表の名前*	xworlds_uid	いいえ
UseDefaults	true または false	false	はい
UseDefaultsForCreatingChildBOs	true または false	false	いいえ
UseDefaultsForRetrieve	true または false	false	いいえ

* トラステッド認証を使用する場合、ApplicationPassword と ApplicationUserName は不要です。

ApplicationPassword
コネクターのユーザー・アカウントのパスワード。

デフォルト値はありません。

ApplicationUserName
コネクターのユーザー・アカウントの名前。

デフォルト値はありません。

ArchiveProcessed
現行サブスクリプションがないイベントを、コネクターにアーカイブさせるかどうかを指定します。

イベント表からイベントが削除された後でそのイベントをアーカイブ表に挿入させるには、このプロパティーを true に設定します。

コネクターにアーカイブ処理を実行させないようにするには、このプロパティーを false に設定します。この場合、ArchiveTableName プロパティーの値は検査されません。ArchiveProcessed が false に設定されている場合、コネクターは次のように動作します。

- イベントが正常に処理された場合、イベント表からそのイベントを削除しますが、アーカイブは行いません。
- コネクターがそのイベントのビジネス・オブジェクトにサブスクライブしていない場合は、イベントをイベント表に残し、そのイベントの状況を Unsubscribed に変更します。使用している統合ブローカーに固有のサブスクリプション情報については、ブローカーのインプリメンテーション・ガイドを参照してください。
- ビジネス・オブジェクトの処理中に問題が発生した場合、イベントをイベント表に残し、イベントの状況を Error にします。

このプロパティーが false に設定されており、さらに、ポーリング量が少ない場合には、コネクターがイベント表に対してポーリングしているように見えます。しかし、これは、単に同じイベントを繰り返し選出しているだけです。

このプロパティーの値がない場合、コネクターでは、その値が true であると見なします。さらに、ArchiveTableName プロパティーの値もない場合、コネクターでは、アーカイブ表の名前が xworlds_archive_events であると見なします。

デフォルト値は true です。

ArchiveTableName
アーカイブ・キュー表の名前。

ArchiveProcessed プロパティーが false に設定されている場合は、このプロパティーの値を設定する必要はありません。

デフォルトの名前は xworlds_archive_events です。

AutoCommit
このプロパティーは、AutoCommit 設定を構成可能にします。true に設定すると、すべてのトランザクションが自動的にコミットされます。一部のデータベース (Sybase など) は、AutoCommit を true に設定する必要があります。false に設定すると、Sybase 上のストアード・プロシージャーが失敗します。

データベース接続が失われた場合、AutoCommit が false に設定されていれば、コネクターは新規の接続を作成して完全処理を再始動しようとします。新規の接続が無効な場合、または AutoCommit が true に設定されている場合は、コネクターは APPRESPONSETIMEOUT を戻します。この結果、コネクターは終了します。

デフォルト値は false です。

CheckforEventTableInInit
このコネクター・プロパティーを false に設定すると、コネクターは、コネクターの初期化時に、イベント表とアーカイブ表が存在するかどうかの確認を行わなくなります。使用している JDBC ドライバーが JDBC クラス DatabaseMetaData をサポートしていない場合を除き、この値を常時 true に設定することを推奨します。

このプロパティーが false に設定されている場合、コネクターはイベント表とアーカイブ表の存在を確認しません。ただし、コネクターは、初期化プロセスにおいてこれらの表を使用するので、これらの表が常に存在していることが必要です。コネクターがイベント表とアーカイブ表を初期化時に使用しないようにするには、EventTableName プロパティーを null に設定します。

デフォルト値は true です。

ChildUpdatePhyDelete
更新操作時に、子ビジネス・オブジェクトが表現するデータが、データベース内には存在するにもかかわらず、着信したビジネス・オブジェクトからは失われている場合、コネクターにそのデータをどのように処理させるかを指定します。

データベースから該当するデータ・レコードを物理的に削除させるには、このプロパティーを true に設定します。

データベース内の該当するデータ・レコードを、状況列で適切な値に設定することにより論理的に削除させるには、このプロパティーの値を false に設定します。ア

プリケーション固有の情報によって、そのビジネス・オブジェクト・レベルのアプリケーション固有の情報内に指定される StatusColumnValue (SCN) パラメーターから、状況列の名前とその値が取得されます。詳細については、64 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。

デフォルト値は false です。

CloseDBConnection

このプロパティーは、データベース接続のクローズを構成可能にします。true に設定されている場合、サービス呼び出し要求およびポーリング呼び出しごとに、データベース接続がクローズされます。このプロパティーを true に設定するとパフォーマンスが低下するため、お勧めしません。

デフォルト値は false です。

ConnectorID

コネクターの固有 ID です。この ID は、コネクターの特定のインスタンスのためにイベントが検索されるときに役立ちます。

デフォルト値は null です。

DatabaseURL

コネクターの接続先データベース・サーバーの名前です。

WebSphere システム・インテグレーション・システム・ブランドの SQLServer ドライバーを使用する場合、次のような URL が推奨されます。

jdbc:ibm-crossworlds:sqlserver://*MachineName*:*PortNumber*;DatabaseName=*DBname*

重要

AutoCommit が false に設定されている場合は、追加のパラメーター SelectMethod を以下のように設定する必要があります。jdbc:ibm-crossworlds:sqlserver://*MachineName*:*PortNumber*;DatabaseName=*DBname*;SelectMethod=cursor デフォルトでは、SelectMethod は direct に設定されています。詳細については、22 ページの『AutoCommit』を参照してください。

コネクターの処理が正常に行われるようにするには、このプロパティーに値を指定する必要があります。

DateFormat

コネクターで受信および戻すことができる日付形式を指定します。このプロパティーは、24 ページの表 7 に記載されている構文に基づくフォーマットをすべてサポートしています。

24 ページの表 7 の時刻パターン・ストリングを使用して時刻形式の構文を定義します。このパターンに使用されている ASCII 文字は、すべてパターン文字として予約されています。

表 7. 時刻形式構文

シンボル	意味	表示	例
G	紀元	(テキスト)	AD
y	年	(数値)	1996
M	月	(テキスト & 数値)	July & 07
d	日 (月初からの通算)	(数値)	10
h	時 (12 時間制、1 から 12)	(数値)	12
H	時 (24 時間制、0 から 23)	(数値)	0
m	分 (時刻表示用)	(数値)	30
s	秒 (時刻表示用)	(数値)	55
S	ミリ秒	(数値)	978
E	曜日	(テキスト)	Tuesday
D	日 (年初からの通算)	(数値)	189
F	曜日 (月初からの通算)	(数値)	2 (2nd Wed in July)
w	週 (年初からの通算)	(数値)	27
W	週 (月初からの通算)	(数値)	2
a	午前/午後	(テキスト)	PM
k	時 (24 時間制、1 から 24)	(数値)	24
K	時 (12 時間制、0 から 11)	(数値)	0
z	時間帯	(テキスト)	Pacific Standard Time
'	テキストのエスケープ	(区切り文字)	
''	単一引用符	(リテラル)	'

表 8. US ロケールを使用した例

形式パターン	結果
"yyyy.MM.dd G 'at' hh:mm:ss z"	1996.07.10 AD at 15:08:56 PDT
"EEE, MMM d, ''yy"	Wed, July 10, '96
"h:mm a"	12:08 PM
"hh 'o''clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:00 PM, PST
"yyyy.MMMMM.dd GGG hh:mm aaa"	1996.July.10 AD 12:08 PM

DriverConnectionProperties

JDBC ドライバーでは、ユーザー名とパスワードの他にも、追加のプロパティーや情報が必要になる場合があります。DriverConnectionProperties コネクター・プロパティーには、JDBC ドライバーに必要な追加のプロパティーを、名前と値のペアとして指定できます。これらのプロパティーは、次のように指定します。

```
property1=value1[;property2=value2...]
```

これらの追加プロパティーは、セミコロンで区切られた名前と値のペアとして指定されていなければなりません。プロパティーとその値は、等号で区切ります (余分なスペースを入れることはできません)。

例えば、JDBC ドライバーで、ライセンス情報とポート番号が必要になるとします。ライセンス情報として要求されるプロパティー名は MyLicense であり、値は ab23jk5 です。ポート番号として要求されるプロパティー名は PortNumber であり、値は 1200 です。DriverConnectionProperties は値 MyLicense=ab23jk5;PortNumber=1200 に設定しなければなりません。

EventKeyDel

イベント表の object_key 列に複数の属性値が含まれる場合に使用される区切り文字を指定します。

トリガーとなったアプリケーションにおいて作成、更新、または削除されたビジネス・オブジェクトを検索する方法は、2 つあります。

- 最初の方法は、object_key 列に、ビジネス・オブジェクトのキーとなっている属性の値を格納する方法です。EventKeyDel 構成プロパティーには、キー・フィールドの一部となっていない文字を 1 つだけ指定します。**例えば**、区切り文字を "**;**" と指定した場合は、object_key は *xxx*;*123* となります。
- 2 番目の方法は、object_key 列に、ビジネス・オブジェクト内のいずれかの属性の値を格納する方法です。これらの値は name_value ペアとして表されます。最初の区切り文字は name_value の区切りに使用され、2 番目の区切り文字はキーの区切りに使用されます。例えば、区切り文字を "**=;**" と指定した場合は、object_key は *CustomerName=xxx*;*CustomerId=123*; となります。

  区切り文字を "**=:**" と指定した場合は、object_key は *CustomerName=xxx*:*CustomerId=123*: となります。

注: これらのキー値の定義順序は、ビジネス・オブジェクト内のキー属性の順序と同じでなければなりません。

重要: Date 属性のデータを使用する場合は、コロン (:) を区切り文字として使用しないようにしてください。この属性のデータには、コロンが含まれていることがあります。

デフォルト値はセミコロン (;) です。これはキーの区切り文字であり、name_value のペアを扱うことはできません。

EventOrderBy

イベントの順序付けをオフにするかどうかを指定します。または、デフォルトの順序と異なるイベント処理の順序を指定します。

デフォルトでは、ポーリングのたびにコネクターは PollQuantityプロパティーに指定されたイベントの番号のみをプルし、イベント表の event_time 列および event_priority 列内の値でイベント処理を順序付けます。

コネクターによるイベントの順序付けが行われないようにするには、このプロパティーの値を none に設定します。

コネクターに、イベント表の複数の列に基づいて順序付けを行わせるには、それらの列の名前を指定します。列名はコンマ (**,**) で分離します。このプロパティーの値を指定すると、デフォルトの順序が上書きされます。

このプロパティーのデフォルト値はありません。

EventQueryType
EventQueryType プロパティーは、イベント表のイベントの検索の際に、コネクターに照会を動的に生成させるか、または コネクターの組み込みの照会を使用させるかを指定するために使用します。動的に生成された照会に関しては、コネクターはそのイベント構造をイベント表の列にマップします。表列内のデータの順序は、非常に重要です。正しい順序については、16 ページの『イベント表およびアーカイブ表のスキーマ』を参照してください。

EventQueryType の値が Fixed (ストリング) の場合、デフォルトの照会が実行されます。Dynamic (ストリング) に設定されている場合は、『EventTableName』プロパティーに指定されている表から列名を取得して、新規の照会が作成されます。

イベント表列名は変更できます。ただし、列の順序とデータ型は、イベント表の作成のセクションで指定したものと同じでなければなりません。デフォルトの照会または動的に生成された照会には、 25 ページの『EventOrderBy』が追加されます。

EventQueryType プロパティーが追加されていない場合、または含まれていない場合は、デフォルトで Fixed になります。

デフォルト値は Fixed (ストリング) です。

EventTableName
コネクターのポーリング機構によって使用されるイベント・キュー表の名前です。

デフォルトの名前は xworlds_events です。

コネクターのポーリングをオフにする場合は、このプロパティーを null (ストリング) に設定してください。これにより、イベント表とアーカイブ表の存在の確認が行われなくなります。

ユーザー定義イベント表の場合は、event_id が INTEGER、BIGINT、NUMERIC、VARCHAR のいずれかの JDBC 型にマップされるようにしてください。

JDBCDriverClass
ドライバーのクラス名を指定します。特定の JDBC ドライバーを使用する場合は、この構成プロパティーにドライバーのクラス名を指定します。例えば、Oracle シン・ドライバーを指定するには、このプロパティーを oracle.jdbc.driver.OracleDriver に設定します。

詳細については、 17 ページの『マルチドライバー・サポートの使用可能化』および 31 ページの『UseDefaultsForCreatingChildBOs』を参照してください。

デフォルト値はありません。

MaximumDatabaseConnections
同時データベース接続の最大許可数を指定します。実行時には、オープン・データベース接続の最大数はこの値に 1 を加えた数になります。

『PreserveUIDSeq』プロパティーが false に設定されている場合は、この数に 2 を加算した合計が、実行時のオープン・データベース接続の数になります。

デフォルト値は 5 です。

PingQuery
コネクターがデータベース接続をチェックするときに使用する SQL ステータスまたはストアード・プロシージャーを指定します。

次に示すのは、ping 照会として使用される SQL ステータスの一例です。

```
SELECT 1 FROM <tablename>
```

次に示すのは、Oracle または DB2 データベースで ping 照会として使用されるストアード・プロシージャー・コールの一例 (sampleSP) です。

```
call sampleSP( )
```

ストアード・プロシージャー・コールに出力パラメーターを指定することはできません。データベースによって入力パラメーターが必要とされる場合、入力値は、ping 照会の一部として指定する必要があります。以下に例を示します。

```
Call checkproc(2)
```

デフォルト値はありません。詳細については、6 ページの『データベース接続不能の処理』および 99 ページの『アプリケーションへの接続不可』を参照してください。

PollQuantity
コネクターがポーリング間隔ごとに検索するデータベース表の行数です。設定可能な値は、1 から 500 です。

デフォルト値は 1 です。

PreserveUIDSeq
着信した固有 ID シーケンスを固有 ID 表に保存するかどうかを指定します。

true に設定されている場合、固有 ID は、ビジネス・オブジェクトが宛先アプリケーションで正常に処理されるまでコミットされません。固有 ID 表にアクセスしようとしている他のプロセスはすべて、トランザクションがコミットされるまで待機しなければなりません。

false に設定されている場合、固有 ID は、ビジネス・オブジェクトがその ID を要求した時点でコミットされます。ビジネス・オブジェクトの処理と固有 ID の処理は、それぞれ、コネクターの内部に専用のトランザクション・ブロックを持ちます。これは、固有 ID 表に関連するトランザクションに、そのトランザクション専用の接続が用意されている場合に限り可能です。

注: このプロパティーがコネクター構成に追加されていない場合のデフォルトの動作は、このプロパティーが追加され、true に設定されている場合の動作と同じです。また、22 ページの『AutoCommit』が true に設定されている場合は、コネクターは PreserveUIDSeq が false に設定されている場合と同様に振る舞います。

27 ページの『PreserveUIDSeq』プロパティーが false に設定されている場合は、この数に 2 を加算した合計が、実行時のオープン・データベース接続の数になります。

デフォルト値は true です。

RDBMS.initsession

データベースとのセッションのそれぞれを初期化する SQL ステートメントです。コネクターは、始動時に照会を受け付けて実行します。この照会の戻り値はありません。プロパティー名は必要ですが、値は必要ではありません。

デフォルト値はありません。

RDBMSVendor

特殊な処理の際に、コネクターに使用させる RDBMS を指定します。Oracle データベースまたは Microsoft SQL Server データベースを使用している場合は、このプロパティーの値を Oracle または MSSQLServer に設定します。その他のデータベースを使用している場合は、このプロパティーの値をそのデータベース名にするか、ブランクのままにします。

デフォルト以外のデータベースを使用している場合は、適切なドライバーがロードされていることを確認します。このプロパティーが Others に設定されている場合、コネクターは、ドライバーを検索して、使用するデータベースを決定します。

コネクターで処理が正常に行われるようにするには、なんらかの値が必要です。

デフォルト値はありません。

ReplaceAllStr

ReplaceStrListプロパティー内に識別される各文字のすべてのインスタンスを、そのプロパティー内に指定された置換ストリングでコネクターに置換させるかどうかを指定します。コネクターは、各属性の AppSpecificInfo プロパティーの ESC=[true|false] パラメーターに値が含まれていない場合にのみ、ReplaceAllStr を評価します。つまり、ESC パラメーターが指定されている場合、その値が ReplaceAllStr プロパティーに設定されている値に対して優先されます。コネクターに ReplaceAllStr の値を使用させるには、ESC パラメーターが指定されていないことを確認します。

ReplaceAllStr のデフォルト値は false です。

注: ESC パラメーター、ReplaceAllStr プロパティー、および ReplaceStrList プロパティーは、データベース・エスケープ文字機能 (単一引用符のエスケープなど) のサポートも提供します。JDBC ドライバーによって提供される Prepared Statements でも同じ機能が利用できるため、今後リリースされるコネクターでは、こうしたプロパティーのサポートが廃止される予定です。現在、コネクターは JDBC 準備済みステートメントの使用をサポートしています。

ReplaceStrList

1 つの置換対象文字、1 つの文字区切り文字、および 1 つの置換ストリングで構成された置換セットを、1 つ以上指定します。属性の AppSpecificInfo プロパティーの

ESC=[true|false] パラメーターの値、またはコネクターの ReplaceAllStr プロパティーの値が指定されている場合にのみ、コネクターは属性値に対してこの置換を実行します。

注: ESC パラメーター、ReplaceAllStr プロパティー、および ReplaceStrList プロパティーは、データベース・エスケープ文字機能 (単一引用符のエスケープなど) のサポートも提供します。JDBC ドライバーによって提供される Prepared Statements でも同じ機能が利用できるため、今後リリースされるコネクターでは、こうしたプロパティーのサポートが廃止される予定です。現在、コネクターは JDBC 準備済みステートメントの使用をサポートしています。

この属性の構文は、次のとおりです。

single_char1,*substitution_str1*[:*single_char2*,*substitution_str2*[:...]]

ここで、以下のように説明されます。

single_char	置換対象文字。
single_char	コネクターが置換対象文字の置換に使用する置換ストリング。
single_char	置換対象文字と置換ストリングを区切る、文字区切り文字。デフォルトでは、文字区切り文字はコンマ (,) です。この区切り文字を構成するには、StrDelimiterプロパティー内の最初の区切り文字を設定します。
single_char	置換セット (1 つの置換対象文字、1 つの文字区切り文字、および 1 つの置換ストリングで構成されたセット) を区切る終了区切り文字。デフォルトでは、終了区切り文字はコロン (:) です。この区切り文字を変更するには、StrDelimiter プロパティーの 2 番目の区切り文字を設定します。

例えば、単一のパーセント記号 (%) を 2 つのパーセント記号 (%%) で置き換える必要があり、さらに、脱字記号 (^) も、円記号と脱字記号の組み合わせ (¥^) で置き換える必要があるとします。デフォルトでは、StrDelimiter には文字区切り文字としてコンマ (,) が指定されています。また、終了区切り文字としてはコロン (:) が指定されています。デフォルトの区切り文字を変更していない場合は、次のストリングを ReplaceStrList の値として使用してください。

```
%,%%:^,\^
```

注: コネクター構成ツールの制限のため、単一引用符を入力することはできません。したがって、単一引用符は文字 Q で表し、2 つの単一引用符は文字 DSQ で表さなければなりません。上記の例において、1 つの単一引用符 (') を 2 つの単一引用符 ('') で置換する場合は、Q,DSQ:%,%%:^,¥^ と表記します。

RetryCountAndInterval

更新操作中にデータをロックできない場合に、コネクターにロックを試行させる回数と間隔 (秒単位) を指定します。

コネクターは、更新を実行する前に、その更新に関連する行をロックして現在のデータを検索しようとします。行をロックできない場合は、この構成プロパティーに指定されている間隔で、指定されている回数まで、ロックを再試行します。この構成プロパティーに指定されている値に達するまでにロックを達成できなかった場合は、結果としてタイムアウトになります。

値は「カウント、間隔」の形式で秒単位で指定します。例えば、`3,20` という値を使用すると、20 秒間隔で 3 回再試行することが指定されます。

デフォルトは `3,20` です。

SchemaName

このプロパティーは、イベント表およびアーカイブ表の検索を、特定のスキーマ内に限定します。このプロパティーが追加されていない場合、あるいは空になっている場合には、コネクターは、ユーザーがアクセスできるスキーマのすべてを検索します。この SchemaName は、イベント表およびアーカイブ表にアクセスするための照会を作成するときにも使用されます。

Oracle データベースでは、スキーマ名のサポートが提供されます。MSSQL Server または DB2 では、スキーマ名が、データベースの所有者名を指すことがありますが、固有の情報については各 JDBC ドライバーの資料を参照してください。

注: DB2 のスキーマ名は、大文字と小文字が区別されます。スキーマ名は大文字で指定する必要があります。

デフォルト値はありません。

SPBeforePollCall

このプロパティーは、ポーリング呼び出しごとに実行されるストアード・プロシージャーを指定します。`SPBeforePollCall` プロパティーに値 (ストアード・プロシージャー名) が指定されている場合、コネクターは、各ポーリング呼び出しの開始時にそのストアード・プロシージャーを呼び出して、コネクター・プロパティー `ConnectorID` および `PollQuantity` の値を渡します。このプロシージャーは `PollQuantity` 個の行を更新し、connector-id 列を `ConnectorID` に設定します。ここで、status=0 の場合は connector-id は `null` です。これにより、コネクターでのロード・バランシングが可能になります。

注: ポーリング呼び出しが途中で失敗した場合 (データベースがダウンしている場合や、接続が失われた場合) には、コネクター ID が設定されたままになります。これにより、ポーリング時に一部のレコードがスキップされることがあります。このため、イベント表に含まれるレコードのうち、状況値が 0 のものについては、すべて、コネクター ID を定期的に `null` にリセットすることを推奨します。

StrDelimiter

`ReplaceStrList` プロパティー内に使用する文字区切り文字、および終了区切り文字を指定します。

- 文字区切り文字は、置換対象文字と置換ストリングを区切るものです。文字区切り文字は、このプロパティーの値の 1 桁目 (左端) を占めます。デフォルトではコンマです (`,`)。

- 終了区切り文字は、置換セット (1 つの置換対象文字、1 つの文字区切り文字、および 1 つの置換ストリングで構成されたセット) の間を区切るものです。終了区切り文字は、このプロパティーの値の 2 桁目 (右端) を占めます。デフォルトではコロンです (:)。

これらの 2 つの区切り文字には、独自の値を指定することができます。このとき、2 つの値の間に、スペースなどの文字を含めないでください。

デフォルト値は、コンマとその直後に続くコロン (,:) です。

TimingStats

このプロパティーを使用すると、コネクターによる動詞操作のそれぞれについて、タイミングを調べて、問題を見つけ出すことができます。設定可能な値は、次のとおりです。

0 (タイミング統計なし)

1 (階層ビジネス・オブジェクト全体のための動詞操作の開始時および終了時にタイミングを出力)

2 (階層ビジネス・オブジェクトに含まれる各個別ビジネス・オブジェクトのための動詞操作の開始時および終了時にタイミングを出力)

タイミング・メッセージは、トレース・メッセージではなく、ログ・メッセージです。このメッセージの出力オン/オフは、トレース・レベルに関係なく行うことができます。

デフォルト値は 0 です。

UniqueIDTableName

固有 ID の生成に使用された値のうち、最新のものが含まれる表を指定します。デフォルトでは、この表の列は 1 つです (id)。この表をカスタマイズすることにより、UID (固有 ID) の生成を必要とする属性ごとに列を 1 つずつ追加することができます。

デフォルト値は xworlds_uid です。

UseDefaults

UseDefaults が true に設定されている場合や、このプロパティーの設定が行われていない場合には、コネクターは、ビジネス・オブジェクトの必須属性のそれぞれに有効な値またはデフォルト値が与えられているかどうかを確認します。値が与えられている場合は、Create 操作が正常に行われます。与えられていない場合には失敗します。

UseDefaults が false に設定されている場合、コネクターは、ビジネス・オブジェクトの必須属性に有効な値が与えられているかどうかのみを確認します。有効な値が与えられていない場合、Create 操作は失敗します。

デフォルト値は false です。

UseDefaultsForCreatingChildBOs

UseDefaultsForCreatingChildBOs が true に設定されている場合や、このプロパティーの設定が行われていない場合には、コネクターは、ビジネス・オブジェクトの必

須属性のそれぞれに有効な値またはデフォルト値が与えられているかどうかを確認します。値が与えられている場合は、Create 操作が正常に行われます。与えられていない場合には失敗します。

UseDefaultsForCreatingChildBOs が false に設定されている場合、コネクターは、ビジネス・オブジェクトの必須属性に有効な値が与えられているかどうかのみを確認します。有効な値が与えられていない場合、Create 操作は失敗します。

UseDefaultsForRetrieve

ポーリングの場合: UseDefaultsForRetrieve が未定義で true に設定されている場合は、BO がデータベースから検索されてサーバーにディスパッチされる前に BO にデフォルト値が設定されます。

UseDefaultsForRetrieve が定義され、false に設定されている場合は、BO がデータベースから検索されてサーバーにディスパッチされる前に BO にデフォルト値が設定されません。

要求処理の場合: UseDefaultsForRetrieve が未定義で false に設定されている場合は、BO がデータベースから検索されてサーバーにディスパッチされる前に BO にデフォルト値が設定されません。

UseDefaultsForRetrieve が定義され、true に設定されている場合は、BO がデータベースから検索されてサーバーにディスパッチされる前に BO にデフォルト値が設定されます。

コネクターの複数インスタンスの作成

コネクターの複数のインスタンスを作成する作業は、いろいろな意味で、カスタム・コネクターの作成と同じです。以下に示すステップを実行することによって、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。次のようにする必要があります。

- コネクター・インスタンス用に新規ディレクトリーを作成します。
- 必要なビジネス・オブジェクト定義が設定されていることを確認します。
- 新規コネクター定義ファイルを作成します。
- 新規始動スクリプトを作成します。

新規ディレクトリーの作成

それぞれのコネクター・インスタンスごとにコネクター・ディレクトリーを作成する必要があります。このコネクター・ディレクトリーには、次の名前を付けなければなりません。

```
ProductDir¥connectors¥connectorInstance
```

ここで connectorInstance は、コネクター・インスタンスを一意的に示します。

コネクターに、コネクター固有のメタオブジェクトがある場合、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、次のディレクトリーを作成して、ファイルをそこに格納します。

ProductDir¥repository¥connectorInstance

ビジネス・オブジェクト定義の作成

各コネクター・インスタンスのビジネス・オブジェクト定義がプロジェクト内にまだ存在しない場合は、それらを作成する必要があります。

1. 初期コネクターに関連付けられているビジネス・オブジェクト定義を変更する必要がある場合は、適切なファイルをコピーし、Business Object Designer を使用してそれらのファイルをインポートします。初期コネクターの任意のファイルをコピーできます。変更を加えた場合は、名前を変更してください。
2. 初期コネクターのファイルは、次のディレクトリーに入っていなければなりません。

   ProductDir¥repository¥initialConnectorInstance

   作成した追加ファイルは、ProductDir¥repository の適切な connectorInstance サブディレクトリー内に存在している必要があります。

コネクター定義の作成

Connector Configurator 内で、コネクター・インスタンスの構成ファイル (コネクター定義) を作成します。これを行うには、以下のステップを実行します。

1. 初期コネクターの構成ファイル (コネクター定義) をコピーし、名前変更します。
2. 各コネクター・インスタンスが、サポートされるビジネス・オブジェクト (および関連メタオブジェクト) を正しくリストしていることを確認します。
3. 必要に応じて、コネクター・プロパティーをカスタマイズします。

始動スクリプトの作成

始動スクリプトは以下のように作成します。

1. 初期コネクターの始動スクリプトをコピーし、コネクター・ディレクトリーの名前を含む名前を付けます。

   dirname

2. この始動スクリプトを、32 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

これで、ご使用の統合サーバー上でコネクターの両方のインスタンスを同時に実行することができます。

カスタム・コネクター作成の詳細については、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

コネクターの開始

コネクターは、**コネクター始動スクリプト**を使用して明示的に始動する必要があります。始動スクリプトは、次に示すようなコネクターのランタイム・ディレクトリーに存在していなければなりません。

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 9 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 9. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
UNIX ベースのシステム	connector_manager_*connName*
Windows	start_*connName*.bat

コネクター始動スクリプトは、以下に示すいずれかの方法で起動することができます。

- Windows システムで**「スタート」**メニューから。

  「プログラム」>「IBM WebSphere Business Integration Adapters」>「アダプター」>「コネクター」 を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Adapters」となっています。ただし、これはカスタマイズすることができます。あるいは、ご使用のコネクターへのデスクトップ・ショートカットを作成することもできます。
- コマンド行から。
  - Windows システム:

    start_*connName connName brokerName* [-c*configFile*]
  - UNIX ベースのシステム:

    connector_manager_*connName* -start

  ここで、*connName* はコネクターの名前であり、*brokerName* は以下のようにご使用の統合ブローカーを表します。
  - WebSphere InterChange Server の場合は、*brokerName* に ICS インスタンスの名前を指定します。
  - WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、または WebSphere Business Integration Message Broker) または WebSphere Application Server の場合は、*brokerName* にブローカーを示すストリングを指定します。

  注: Windows システム上の WebSphere Message Broker または WebSphere Application Server の場合は、-c オプションに続いてコネクター構成ファイルの名前を指定しなければなりません。ICS の場合は、-c はオプションです。
- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。

- System Monitor から (WebSphere InterChange Server 製品のみ)。

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムがブートしたとき (自動サービスの場合)、または Windows サービス・ウィンドウを通じてサービスを始動したとき (手動サービスの場合) に、コネクターが始動します。

コマンド行の始動オプションなどのコネクターの始動方法の詳細については、以下の資料のいずれかを参照してください。

- WebSphere InterChange Server については、「*システム管理ガイド*」を参照してください。
- WebSphere Message Brokers については、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」を参照してください。
- WebSphere Application Server については、「*アダプター実装ガイド (WebSphere Application Server)*」を参照してください。

コネクターの停止

コネクターを停止する方法は、以下に示すように、コネクターが始動された方法によって異なります。

- コマンド行からコネクターを始動した場合は、コネクター始動スクリプトを用いて、以下の操作を実行します。
  - Windows システムでは、始動スクリプトを起動すると、そのコネクター用の別個の「コンソール」ウィンドウが作成されます。このウィンドウで、「Q」と入力して Enter キーを押すと、コネクターが停止します。
  - UNIX ベースのシステムでは、コネクターはバックグラウンドで実行されるため、別ウィンドウはありません。代わりに、次のコマンドを実行してコネクターを停止します。

    ```
    connector_manager_connName -stop
    ```

    ここで、*connName* はコネクターの名前です。
- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- System Monitor から (WebSphere InterChange Server 製品のみ)

  このツールを使用して、コネクターのロード、アクティブ化、非アクティブ化、休止、シャットダウン、または削除を行うことができます。
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

第 3 章 コネクターのビジネス・オブジェクトについて

この章では、JDBC 用コネクターでのビジネス・オブジェクトの処理方法、およびコネクターによるデータ検索およびデータ変更の際の前提事項について説明します。この章の内容は、次のとおりです。

- 『ビジネス・オブジェクトおよび属性の命名規則』
- 『ビジネス・オブジェクトの構造』
- 43 ページの『ビジネス・オブジェクト動詞の処理』
- 60 ページの『ビジネス・オブジェクトの属性プロパティー』
- 63 ページの『ビジネス・オブジェクトのアプリケーション固有の情報』

この情報は、既存のビジネス・オブジェクトを変更する場合のためのガイドとして、または新規ビジネス・オブジェクトのインプリメントに関する提案事項として役立てることができます。データベース表からの WebSphere Business Integration Adapter ビジネス・オブジェクト定義ファイルの作成を自動化するユーティリティーについては、77 ページの『第 4 章 JDBCODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

コネクターでは、サポートされるビジネス・オブジェクトの構造、親ビジネス・オブジェクトと子ビジネス・オブジェクトの関係、アプリケーション固有の情報の形式、およびビジネス・オブジェクトのデータベース表記に関する前提事項が想定されます。したがって、コネクターによって処理されるビジネス・オブジェクトを作成または変更する際には、コネクターが順守するように設計されているルールに変更内容を準拠させる必要があります。変更がその規則に準拠していない場合、コネクターでは新規ビジネス・オブジェクトまたは変更されたビジネス・オブジェクトを正しく処理することができません。

ビジネス・オブジェクトおよび属性の命名規則

コネクターで使用されるビジネス・オブジェクトの名前には、英数字と下線文字のみを使用できます。ビジネス・オブジェクト属性名にも、英数字と下線文字のみを使用できます。

ビジネス・オブジェクトの構造

多くの場合、コネクターはすべての個別ビジネス・オブジェクトが 1 つのデータベース表またはビューによって表され、オブジェクト内部のそれぞれの**単純属性** (つまり、`String` または `Integer` または `Date` などの単一値を表す属性) はその表またはビュー内の列によって表されると想定します。したがって、同じ個別ビジネス・オブジェクトに含まれる属性を、別々のデータベース表に格納することはできません。ただし、次のような状態は可能です。

- データベース表に、対応する個別ビジネス・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 (つまり、データベース列の一部

が、ビジネス・オブジェクト内に表されていません)。ユーザー設計のビジネス・オブジェクトには、ビジネス・オブジェクトの処理に必要な列のみを組み込んでください。

- 個別ビジネス・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 (つまり、ビジネス・オブジェクト内の属性の一部が、データベース表内に表されていません)。データベース表の列を表していないビジネス・オブジェクトの属性には、アプリケーション固有情報が含まれていないか、あるいはデフォルト値またはストアード・プロシージャーが指定されています。
- 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。コネクターでは、アプリケーション内で起動された Create、Retrieve、Update、および Delete の各イベントを処理するときに、そのようなビジネス・オブジェクトを使用することができます。ただし、ビジネス・オブジェクトからの要求を処理する場合には、Retrieve 要求に対してのみ、そのようなビジネス・オブジェクトを使用できます。
- 個別ビジネス・オブジェクトは、関連のないビジネス・オブジェクトのコンテナーとして使用されるラッパー・オブジェクトを表すことができます。ラッパー・オブジェクトはデータベース表やビューによって表されません。ラッパー・オブジェクトは他のオブジェクトの子として使用することはできません。

注: ビジネス・オブジェクトがストアード・プロシージャーを基にしている場合、各単純属性 (ストアード・プロシージャー用の特殊な SP 属性を除く) には、アプリケーション固有情報が含まれていることも、含まれていないこともあります。詳細については、52 ページの『ストアード・プロシージャー』を参照してください。

WebSphere Business Integration Adapter ビジネス・オブジェクトは、フラットなものと階層のものがあります。**フラット**・ビジネス・オブジェクトの属性は、すべて単純属性であり、単一の値を表します。

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単一の子ビジネス・オブジェクトを表すときに発生します。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。

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。この場合、この属性は子ビジネス・オブジェクトと同じタイプの配列です。

注: **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、単一のビジネス・オブジェクトを指します。そのビジネス・オブジェクトの子オブジェクトや、そのビジネス・オブジェクトが属する子ビジネス・オブジェクトは含みません。**最上位**ビジネス・オブジェクトという用語は、階層の頂点にある、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。

コネクターでは、ビジネス・オブジェクト間での以下の関係がサポートされます。

- 『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』
- 41 ページの『複数カーディナリティー関係』
- 41 ページの『関係を子に格納する単一カーディナリティー関係』
- 42 ページの『ラッパー・オブジェクト』

カーディナリティーのタイプを問わず、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、その関係が保管されるビジネス・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記述されています。このアプリケーション固有情報の詳細は、 66 ページの『FK=[fk_object_name.]fk_attribute_name』を参照してください。

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。一方の属性のタイプは、子ビジネス・オブジェクトのタイプと同じになります。もう一方の属性は、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。

関係を設定する外部キーが親に保管されるため、各親には特定のタイプの単一カーディナリティーの子を 1 つだけ格納できます。

図 2 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、`fk1` が子ビジネス・オブジェクトの基本キーを格納するための単純属性であり、`child[1]` が子ビジネス・オブジェクトを表す属性です。

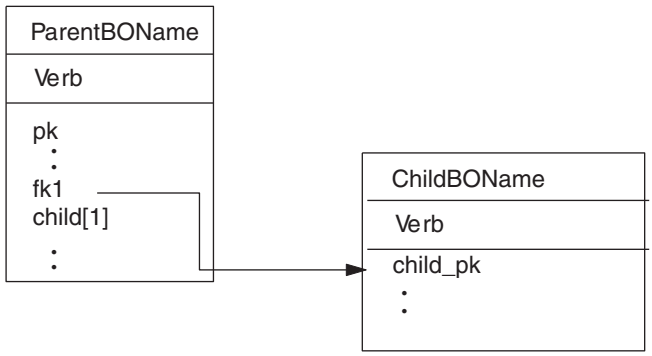


図 2. 典型的な単一カーディナリティー関係

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

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、Customer ビジネス・オブジェクトのそれぞれに Address ビジネス・オブジェクトが 1 つずつ含まれる場合、新しいカスタマーが作成されると、Customer 表と Address 表の両方に

新しい行が挿入されます。挿入された新しい住所は、その新しいカスタマーに固有です。同様に、Customer 表からカスタマーを削除すると、そのカスタマーの住所も Address 表から削除されます。

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、そのデータがそれらのビジネス・オブジェクトのいずれにも所有されていない場合があります。例えば、Address ビジネス・オブジェクトに StateProvince[1] 属性があり、単一カーディナリティーの StateProvince 索引表を表しているとします。この索引表は、ほとんど更新されることがないものであり、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、この索引表内のデータが影響を受けることはありません。コネクターは、既存の州名を検出するか、検出に失敗するかのいずれかです。この索引表内の値を追加または変更することはありません。

複数のビジネス・オブジェクトに同一の単一カーディナリティーの子ビジネス・オブジェクトが含まれている場合、各親ビジネス・オブジェクトの外部キー属性では、関係が NO_OWNERSHIP に設定されていなければなりません。統合ブローカーからコネクターに、階層ビジネス・オブジェクトが Create、Delete、または Update 要求とともに送信された場合、コネクターは所有関係にない単一カーディナリティーの子を無視します。コネクターは、それらのビジネス・オブジェクトに対しては、検索のみを実行します。そ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、コネクターはエラーを戻して処理を停止します。

所有権なしの関係を指定する方法については、72 ページの『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属性』を参照してください。外部キーの関係の指定については、68 ページの『属性の外部キーの指定』を参照してください。

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は、静的索引表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の同期も可能にします。

正規化データから非正規化データへの同期化: 関係を NO_OWNERSHIP に設定すると、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・アプリケーションで、A と B という 2 つの表にデータが格納されているとします。また、非正規化されている宛先アプリケーションでは、単一の表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ているとします。

この例では、表 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するには、表 B のデータが変更されるたびに表 A のイベントを起動する必要があります。さらに、表 B のデータは表 A に重複して格納されているので、表 A の行ごとに、表 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。

非正規化データから正規化データへの同期化: 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する場合、コネクターは、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、または更新しません。

正規化されているアプリケーションにデータを同期する場合、コネクターは、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

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、関係を表すための属性が 1 つだけ含まれています。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ことができるようにするため、関係を設定する外部キーはそれぞれの子に保管されます。

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保管されるので、親は、それぞれ、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 3 に複数カーディナリティー関係を示します。この例では、`parentId` が親ビジネス・オブジェクトの基本キーを格納するための単純属性であり、`child[n]` が子ビジネス・オブジェクトを表す属性です。

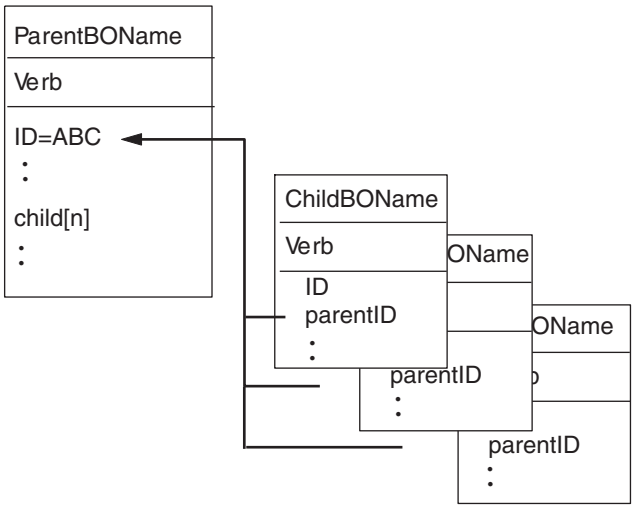


図 3. 複数カーディナリティー・ビジネス・オブジェクト関係

関係を子に格納する単一カーディナリティー関係

一部のアプリケーションでは、子エンティティーが 1 つだけ格納されていて、関係が親ではなくこの子エンティティーに保管されます。つまり、子には、親の基本キーに格納されている値と同一の値の外部キーが格納されます。

図 4 に、特別なタイプの単一カーディナリティー関係を示します。

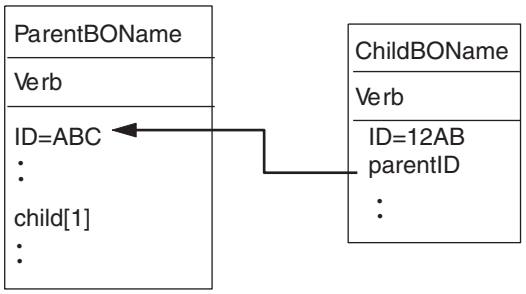


図 4. 関係を子に格納している単一カーディナリティー・ビジネス・オブジェクト

このタイプの単一カーディナリティー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は、複数の親によって所有されることはありません。また、子とその外部キー値の作成は、親とその基本キー値があらかじめ存在していなければ実行できません。

このようなアプリケーションに対応するため、コネクターでは、単一カーディナリティーの関係で子を含んでいるにもかかわらず (親ではなく) 子に関係を保管する階層ビジネス・オブジェクトも、サポートしています。

このような特殊な方法で親ビジネス・オブジェクトに単一カーディナリティーの子を含めるには、子が格納される属性のアプリケーション固有情報を指定するときに、CONTAINMENT パラメーターを含めないようにします。詳細については、72 ページの『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属性』を参照してください。

ラッパー・オブジェクト

ラッパー・オブジェクトは、どのデータベース表またはビューにも対応しない最上位のビジネス・オブジェクトです。ラッパー・オブジェクトは、true の値を持つ最上位ビジネス・オブジェクト・プロパティー WRAPPER によって示されます。ラッパー・オブジェクトは関連のない子のコンテナーとして使用されるダミーの親です。ラッパー・オブジェクトの処理中、コネクターは最上位ビジネス・オブジェクトを無視し、子のみを処理します。ラッパー・オブジェクトには N のカーディナリティーを持つエンティティーまたは N-1 のカーディナリティーを持つエンティティー、あるいはその両方を含めることができます。

N のカーディナリティーを持つエンティティーは、最低でも 1 つの固有属性が基本キーとしてマークされ、最低でも 1 つの属性が外部キーとしてマークされている必要があります。この外部キーは、次に基本キーとしてラッパー・オブジェクトに追加されます。エンティティーの外部キーは、ここで追加されたラッパー・オブジェクトの基本キーを参照します。

N-1 のカーディナリティーを持つエンティティーの場合、基本キーは基本キーとしてマークされると同時に、ラッパーの基本キーを参照する外部キー (N-1 のエンティティーの基本キーと同じ) としてマークされる必要があります。

ビジネス・オブジェクト動詞の処理

このセクションでは、ビジネス・オブジェクトの動詞の処理における以下の点について説明します。

- 『動詞の判別』では、コネクターがそれぞれのソース・ビジネス・オブジェクトごとに使用する動詞を決定する方法を説明します。
- 『変更後イメージと差分』では、用語を定義し、コネクターが変更後イメージを扱う方法を説明します。
- 45 ページの『動詞の処理』では、ビジネス・オブジェクトを作成、検索、更新および削除を行う際に、コネクター が実行するステップを説明します。
- 52 ページの『SQL ステートメント』では、コネクターが単純な SQL ステートメントを使用して選択、更新、検索、または削除の操作を行う方法を説明します。
- 52 ページの『ストアード・プロシージャー』では、コネクターがストアード・プロシージャーを使用する方法を説明します。
- 60 ページの『トランザクション・コミットとロールバック』では、コネクターがトランザクション・ブロックを使用する方法を簡単に説明します。

動詞の判別

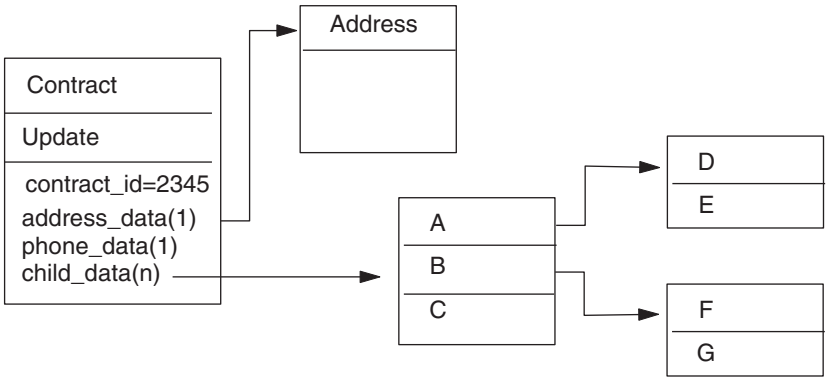
最上位ビジネス・オブジェクト、およびその子にあたる個別ビジネス・オブジェクトには、それぞれ別々に動詞を格納することができます。したがって、親ビジネス・オブジェクトと子ビジネス・オブジェクトで動詞が異なるビジネス・オブジェクトが、統合ブローカーから コネクターに渡される場合があります。この場合、コネクターでは、最上位の親ビジネス・オブジェクトの動詞を参照して、ビジネス・オブジェクト全体をどのように処理するかを決定します。詳細については、45 ページの『動詞の処理』を参照してください。

変更後イメージと差分

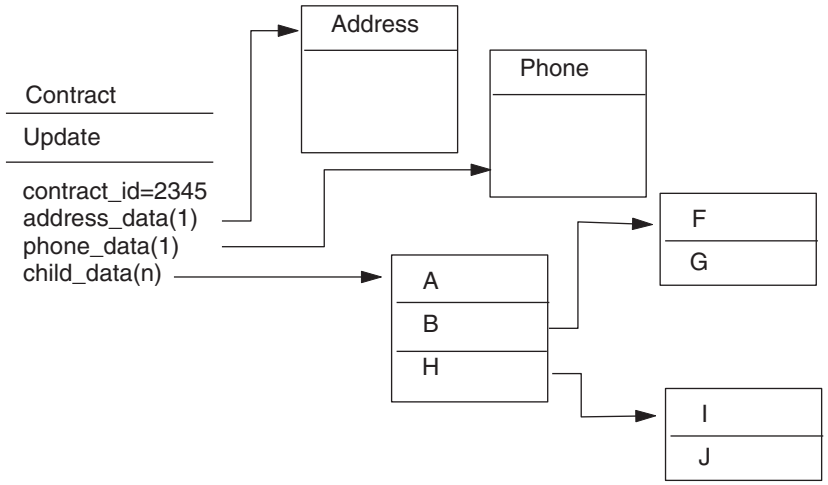
変更後イメージとは、すべての変更が行われた後のビジネス・オブジェクトの状態のことです。差分とは、更新操作で使用される、キー値および変更対象のデータのみを含むビジネス・オブジェクトのことです。コネクターは変更後イメージのみをサポートしているため、更新用のビジネス・オブジェクトを受信した場合には、そのビジネス・オブジェクトが、データの更新後のあるべき状態を表していると見なします。

したがって、統合ブローカーからビジネス・オブジェクトが Update 動詞とともに送信された場合、コネクターは、そのビジネス・オブジェクトのデータベースにおける現在の表現を変更して、ソース・ビジネス・オブジェクトに厳密に一致させます。これを行うため、コネクターは、単純属性の値の変更や、子ビジネス・オブジェクトの追加または除去を実行します。

例えば、データベース内の Contract 2345 の現在の状態が次のとおりである場合を想定します。



さらに、統合ブローカーが、次に示すビビネス・オブジェクトをコネクターに受け渡すと想定します。



コネクターは、更新処理を行うため、次の変更をデータベースに適用します。

- Contract ビジネス・オブジェクト (最上位) および Address ビジネス・オブジェクトの単純属性の更新
- Phone ビジネス・オブジェクトの作成
- 子ビジネス・オブジェクト A、B、F、および G の単純属性の更新
- 子ビジネス・オブジェクト C、D、および E の削除
- 子ビジネス・オブジェクト H、I、および J の作成

コネクターでは、統合ブローカーから受信した各ビジネス・オブジェクトについても、変更後イメージを表していると見なされます。このため、このコネクターに送信されるどの更新用ビジネス・オブジェクトにも、既存の有効な子ビジネス・オブジェクトのすべてが含まれるよう注意する必要があります。子ビジネス・オブジェクトの中に単純属性がまったく変更されていないものがある場合にも、その子ビジネス・オブジェクトがソース・ビジネス・オブジェクトに含まれていなければなりません。

ただし、一部のコネクターについては、更新操作時に欠落している子ビジネス・オブジェクトが削除されるのを防ぐ方法があります。子 (または子の配列) を表す属性のアプリケーション固有情報を使用して、コネクターに対し、ソース・ビジネス・オブジェクトに含まれない子ビジネス・オブジェクトを保持するよう指示することができます。これを行うには、KEEP_RELATIONSHIP を true に設定します。詳細については、68 ページの『属性の外部キーの指定』を参照してください。

動詞の処理

このセクションでは、コネクターが統合ブローカーから受信したビジネス・オブジェクトを作成、検索、更新、または削除するときに実行するステップについて、概説します。コネクターでは、階層ビジネス・オブジェクトを再帰的に処理します。つまり、個別ビジネス・オブジェクトがすべて処理されるまで、同じステップを子ビジネス・オブジェクトごとに実行します。

注: ラッパーである最上位ビジネス・オブジェクトは、create、retrieve、update、および delete 動詞をサポートします。ラッパー・オブジェクトの処理で唯一異なる点は、ラッパー・オブジェクトが処理されずにラッパー・オブジェクトに含まれるオブジェクトのみが処理されるということです。

ビジネス・オブジェクトの比較

以下に概説する処理のさまざまなポイントで、コネクターは 2 つのビジネス・オブジェクトを比較し、それらが同一であるかどうかを確認します。例えば、更新操作時には、ビジネス・オブジェクトの配列内に、特定のビジネス・オブジェクトが存在するかどうかを判定します。コネクターは、この検査を行うため、その特定のビジネス・オブジェクトを配列内のビジネス・オブジェクトのそれぞれと比較します。2 つのビジネス・オブジェクトが同一であるのは、次の 2 つの条件が満たされている場合です。

- 比較されている 2 つのビジネス・オブジェクトのタイプが一致していること。例えば、Customer ビジネス・オブジェクトと Contact ビジネス・オブジェクトの属性がすべて一致している場合でも、これらのビジネス・オブジェクトが同一であると見なされることはありません。
- 2 つのビジネス・オブジェクトの、対応するキー属性のすべてに、同一の値が格納されていること。あるキー属性が両方のビジネス・オブジェクトで CxIgnore に設定されている場合、コネクターでは、それらのキー属性を同一と見なします。しかし、あるキー属性が一方のビジネス・オブジェクトでは CxIgnore に設定されており、もう一方のビジネス・オブジェクトではこの値に設定されていない場合には、それらのビジネス・オブジェクトは同一ではありません。

Create 操作

コネクターは、ビジネス・オブジェクトの作成時に、2 つの状況のいずれかを戻します。戻される状況は、操作が正常に終了した場合 (操作によってビジネス・オブジェクトの変更が生じたかどうかを問いません) は VALCHANGE、操作が失敗した場合は FAIL です。

コネクターでは、階層ビジネス・オブジェクトの作成時に、以下のステップを実行します。

1. 所有関係にある単一カーディナリティーの子ビジネス・オブジェクトのすべてを、データベース内に再帰的に挿入します。つまり、コネクターは、子ビジネス・オブジェクトおよびその子孫にあたるビジネス・オブジェクトのすべてを作成します。

   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関係の子ビジネス・オブジェクトを表すものとされている場合に、その属性が空になっていると、コネクターはその属性を無視します。ただし、ビジネス・オブジェクト定義上、その属性が子を表すことが必須であるにもかかわらず、子を表していない場合には、コネクターはエラーを戻して処理を停止します。
2. 所有関係にない単一カーディナリティーの子ビジネス・オブジェクトを、次のようにしてすべて処理します。
   a. 統合ブローカーから渡されたキー値を使用して、データベース内で子の検索を再帰的に試行します。
   b. 検索に失敗した場合 (子がデータベースに現存しないことを意味します)、コネクターはエラーを戻して処理を停止します。検索に成功した場合、コネクターは子ビジネス・オブジェクトを再帰的に更新します。

   注: アプリケーションのデータベースに子ビジネス・オブジェクトがすでに存在する場合、このアプローチが正しく機能するようにするため、子ビジネス・オブジェクトの基本キー属性が作成操作時に正しく相互参照されるようにしなければなりません。アプリケーションのデータベースに子ビジネス・オブジェクトが存在していない場合は、基本キー属性を `CxBlank` に設定してください。
3. 最上位ビジネス・オブジェクトを、次のようにしてデータベース内に挿入します。
   a. 最上位ビジネス・オブジェクトの外部キー値を、対応する単一カーディナリティーの関係にある子ビジネス・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、子ビジネス・オブジェクト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、コネク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ています。属性にデータベース・シーケンスまたはカウンターが関連付けられている場合、コネクターによって生成された値により、統合ブローカーから渡された値が上書きされます。データベース・シーケンスまたはカウンターの指定については、 66 ページの『単純属性のアプリケーション固有情報』の `UID=AUTO` を参照してください。
   c. 属性のアプリケーション固有情報に含まれる `CA` (CopyAttribute) パラメーターの指定に従って、属性間で値をコピーします。`CA` パラメーターの使用については、 66 ページの『単純属性のアプリケーション固有情報』の `CA=set_attr_name` を参照してください。
   d. 最上位ビジネス・オブジェクトをデータベース内に挿入します。

注: ラッパーである最上位ビジネス・オブジェクトは、データベース内に挿入されません。

4. 単一カーディナリティーの子ビジネス・オブジェクトのうち、親/子関係を子に保管するものすべてを、次のようにして処理します。
   a. 子の外部キー値を、親に含まれる対応する基本キー属性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ります。そのため、ここでは、コネクターが子をデータベースに挿入する前に、それぞれの子の外部キー値を正しいものにします。
   b. 子をデータベースに挿入します。
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、次のようにして処理します。
   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ります。そのため、ここでは、コネクターが子をデータベースに挿入する前に、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。

Retrieve 操作

コネクターでは、階層ビジネス・オブジェクトの検索時に、以下のステップを実行します。

1. 統合ブローカーで受信した最上位ビジネス・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。
2. 最上位ビジネス・オブジェクトを、データベース内で検索します。
   - 検索の結果戻された行が 1 つの場合、コネクターは処理を継続します。
   - 検索の結果戻された行がない場合 (目的の最上位ビジネス・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)、コネクターは `BO_DOES_NOT_EXIST` を戻します。
   - 検索の結果戻された行が複数ある場合、コネクターは `FAIL` を戻します。

   注: ビジネス・オブジェクトには、どのデータベース列にも対応しない属性 (プレースホルダー属性など) が含まれていることがあります。コネクターが、検索時に最上位ビジネス・オブジェクトのそのような属性を変更することはありません (それらの属性は、統合ブローカーから受信した値に設定されたまま保持されます)。子ビジネス・オブジェクトのそのような属性については、検索時にコネクターによってデフォルト値に設定されます。

   注: ラッパーである最上位ビジネス・オブジェクトには、ラッパー・オブジェクトの直下にあるオブジェクトのすべての属性値が含まれている必要があります。この値はキーおよびプレースホルダー属性などのオブジェクトの検索に必要となります。ラッパー・オブジェクトにはすべてのキーおよびプレースホルダー属性が取り込まれる必要があります。ラッパーの 1 レベル下のオブジェクトで外部キーとして使用されるラッパー・オブジェクトの単純属性は、ラッパー・オブジェクトのキーとしてマークされる必要があります。
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: コネクターは、ビジネス・オブジェクトの配列を検索するときに、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、コネクターは、それらの重複する子を戻します。

4. 単一カーディナリティーの子を、所有関係にあるかどうかを問わず、すべて再帰的に検索します。

   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、すべて、親ビジネス・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る前に処理されます。子ビジネス・オブジェクトが所有されているかいないかは、処理シーケンスの決定には関係がありません。ただし、処理のタイプの決定には関係があります。

RetrieveByContent 操作

RetrieveByContent 動詞は、最上位ビジネス・オブジェクトに対してのみ適用できます。これは、最上位ビジネス・オブジェクトのみに含まれる属性を基に、コネクターによる検索が実行されるからです。

最上位ビジネス・オブジェクトに RetrieveByContent 動詞が使用されている場合は、非ヌル属性のすべて (非キー属性を含みます) が検索基準として使用されます。

複数の行が戻された場合、コネクターは最初の行を結果行として使用します。また、MULTIPLE_HITS を戻します。

注: RetrieveByContent 動詞はラッパーである最上位ビジネス・オブジェクトには適用されません。

Update 操作

コネクターは、ビジネス・オブジェクトの更新時に、2 つの状況のいずれかを戻します。戻される状況は、操作が正常に終了した場合 (操作によってビジネス・オブジェクトの変更が生じたかどうかを問いません) は VALCHANGE、操作が失敗した場合は FAIL です。コネクターは、Oracle データベースを操作する場合、データを検索している間、データ保全性を確保するためにデータをロックします。

コネクターでは、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。

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 (目的の最上位ビジネス・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)、コネクターは BO_DOES_NOT_EXIST を戻します。この場合、更新は失敗します。

     注: ラッパーである最上位ビジネス・オブジェクトはデータベース内に存在する必要はありません。ただし、ラッパー・オブジェクトの直下にあるオブジェクトのすべての属性値が含まれている必要があります。この値はキーおよびプレースホルダー属性などのオブジェクトの検索に必要となります。ラッパー・オブジェクトにはすべてのキーおよびプレースホルダー属性が取り込まれる必要があります。ラッパーの 1 レベル下のオブジェク

トで外部キーとして使用されるラッパー・オブジェクトの単純属性は、ラッパー・オブジェクトのキーとしてマークされる必要があります。

- 検索に成功した場合、コネクターは、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、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の比較は行いません。コネクターは、非キーの単純属性のすべてで、値を更新します。

  最上位ビジネス・オブジェクトの単純属性がすべてキーを表している場合、コネクターはその最上位ビジネス・オブジェクト用の更新照会を生成できません。この場合、コネクターは、警告を記録してからステップ 2 に進みます。

2. 最上位ビジネス・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、更新は失敗し、コネクターはエラーを戻します。

   コネクターでは、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、コネクターは、データベース内の既存の子を更新するのではなく、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は、データベース内のその子を再帰的に削除します。削除タイプが物理的であるか、論理的であるかは `ChildUpdatePhyDelete` プロパティーの値に依存します。

   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、コネクターは、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。コネクターは、目的の子ビジネス・オブジェクトの検索に成功すると、その子ビジネス・オブジェクトを移植します。しかし、所有関係にない単一カーディナリティーの子ビジネス・オブジェクトは、コネクターによって変更されないものであるため、更新は行いません。

3. 関係を親に保管する単一カーディナリティーの子ビジネス・オブジェクトに関しては、親に存在する外部キー値のそれぞれを、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。このステップが必要なのは、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、新しい固有 ID が生成されている可能性があるためです。
4. 検索されたビジネス・オブジェクトの単純属性のすべてを更新します。ただし、ソース・ビジネス・オブジェクト内の対応する属性に値 `CxIgnore` が含まれるものを除きます。

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、コネクターは、結果として 1 行のみが処理されることを確認します。1 つ以上の行が戻されている場合、コネクターはエラーを戻します。

5. 親/子関係を子に保管する子ビジネス・オブジェクト (複数カーディナリティーであるか、単一カーディナリティーであるかを問いません) のそれぞれにおいて、外部キー値のすべてを、対応する親ビジネス・オブジェクトの基本キー値に設定します。(ICS を統合ブローカーとして使用する場合は、通常、これらの値はデータ・マッピング時に相互参照されます。) ただし、関係を子に保存する新しい子の外部キー値を、コネクターがそれらの子を更新する前に正しいものにするためには、このステップが重要です。
6. 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、`KEEP_RELATIONSHIP` が `true` に設定されている場合を除きます。この場合、コネクターは、データベースからその子を削除しません。詳細については、68 ページの『属性の外部キーの指定』を参照してください。削除タイプが物理的であるか、論理的であるかは `ChildUpdatePhyDelete` プロパティーの値に依存します。

注: 統合ブローカーでは、ソース・ビジネス・オブジェクト内の複数カーディナリティーの関係にあるビジネス・オブジェクトがそれぞれ一意であること (つまり、ある配列に同一のビジネス・オブジェクトが複数含まれていないこと) を確認する必要があります。受信ソース・ビジネス・オブジェクトにおいて、1 つの配列に同じビジネス・オブジェクトが重複して存在していると、コネクターはそのビジネス・オブジェクトを 2 回処理します。これにより、予期しない結果が発生する可能性があります。

DeltaUpdate 操作

DeltaUpdate 動詞の処理は、Update 動詞の処理と以下の点で異なります。

1. Update 動詞が処理される際には更新の前に検索が実行されますが、DeltaUpdate が処理される際には実行されません。
2. 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。
3. どの子も、各子オブジェクトに設定されている動詞セットに基づいて処理されます。子に動詞セットが設定されていない場合、コネクターはエラーを戻します。

コネクターは、ビジネス・オブジェクトの差分更新時に、VALCHANGE と FAIL のいずれかの状況を戻します。操作が正常に終了した場合 (操作によってビジネス・オブジェクトの変更が生じたかどうかを問いません) は VALCHANGE、操作が失敗した場合は FAIL です。

コネクターでは、階層ビジネス・オブジェクトの差分更新時に、以下のステップを実行します。

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・オブジェクト定義で IsRequired が true に設定されている子は、インバウンド・オブジェクトに必ず存在していなければなりません。存在しない場合、差分更新は失敗し、コネク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。ただし、インバウンド・ビジネス・オブジェクトで IsIgnore に設定されている属性を除きます。コネクターでは、インバウンド・オブジェクトと現在のオブジェクトを属性レベルで比較して、UPDATE ステートメントに追加する必要がある属性を決定することはありません。つまり、属性はすべて更新されます。更新されるオブジェクトは一意である必要があるため、コネクターは、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子のうち、カーディナリティーが N のものすべてで、親の属性を参照する外部キー値のすべてを、それぞれ対応する親の値に設定します。通常、これらの値はデータ・マッピング時に相互参照されます。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ことがあります。ここでの処理により、カーディナリティーが N の子のすべてで、これらの子が更新される前に外部キー値を確実に正しい値にすることができます。
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、それぞれの子の動詞が取得されて適切な操作が実行されます。DeltaUpdate が処理される際に許可される子の動詞は、Create、Delete、および DeltaUpdate です。

- 子で Create 動詞が検出された場合、その子が所有関係にある子であれば、データベースにその子が作成されます。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。
- 子で Delete 動詞が検出された場合、その子は削除されます。
- 子で DeltaUpdate 動詞が検出された場合、データベースでその子が更新されます。

Delete 操作

コネクターは、ビジネス・オブジェクトの削除時には、操作に成功すると状況 SUCCESS を戻し、失敗すると状況 FAIL を戻します。アダプターは、まず親ビジネ

ス・オブジェクトを検索します。次に、親から見て所有関係にある単一カーディナリティーの子のすべてを再帰的に削除してから、親ビジネス・オブジェクト自体を削除します。最後に、カーディナリティーが N の子をすべて削除します。所有関係にない単一カーディナリティーの子は削除されません。操作対象のビジネス・オブジェクトが存在しない場合、コネクターは FAIL を戻します。

コネクターはオブジェクトのアプリケーション固有情報にある 状況列名 (SCN) 値によって、論理的な削除も物理的な削除もサポートします。SCN 値が定義されている場合は、論理削除を実行します。SCN 値が定義されていない場合は、物理削除を実行します。

物理削除: コネクターでは、階層ビジネス・オブジェクトの物理削除時に、以下のステップを実行します。

1. 所有権付きで含まれてい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。
2. 最上位ビジネス・オブジェクトを削除します。
3. 複数カーディナリティーの子ビジネス・オブジェクトすべてを再帰的に削除します。

注: ラッパーである最上位ビジネス・オブジェクトは対応するデータベース表を持たないため、データベースから削除されません。ラッパーの単純属性値はすべて無視されます。

論理削除: ビジネス・オブジェクトの論理削除時には、コネクターは以下のステップを実行します。

1. UPDATE を発行して、ビジネス・オブジェクトの状況属性を、ビジネス・オブジェクトのアプリケーション固有情報に指定されている値に設定します。コネクターでは、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。
2. 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。コネクターは、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。

SQL ステートメント

コネクターでは、単純な SQL ステートメントを使用して、選択、更新、検索、または削除の操作を行うことができます。SQL ステートメント用の列名は、属性の AppSpecificInfo プロパティーから取得されます。各照会は複数の表にまたがることができません。ただし、ビューに追加することはできます。

ストアード・プロシージャー

ストアード・プロシージャーとは、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、コネク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。

コネクターは、次の目的でストアード・プロシージャーを呼び出します。

- 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、操作後処理を行う。
- 単純な INSERT、RETRIEVE、UPDATE、または DELETE 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。

コネクターでは、階層ビジネス・オブジェクトを処理するときに、ストアード・プロシージャーを使用して、最上位ビジネス・オブジェクトまたは任意の子ビジネス・オブジェクトを処理することができます。ただし、ビジネス・オブジェクト (またはビジネス・オブジェクトの配列) には、ストアード・プロシージャーが個別に用意されていなければなりません。

ストアード・プロシージャーの指定

このセクションでは、コネクターからビジネス・オブジェクトに対してストアード・プロシージャーを使用する場合に、実行する必要があるステップについて説明します。このセクションの内容は、次のとおりです。

- 『ビジネス・オブジェクトへの属性の追加』
- 54 ページの『ストアード・プロシージャーの構文』
- 55 ページの『ストアード・プロシージャーの例』
- 55 ページの『ストアード・プロシージャーの指定』

ビジネス・オブジェクトへの属性の追加: コネクターが処理されるストアード・プロシージャーがどのタイプでも、ビジネス・オブジェクトには特殊な種類の属性を追加しなければなりません。この属性は、ストアード・プロシージャーのタイプと、ストアード・プロシージャーを定義するアプリケーション固有情報のみを表します。標準的な単純属性で使用できるアプリケーション固有情報用のパラメーターは、これらの属性では使用しません。

使用されるストアード・プロシージャーのタイプに応じて属性を指定してください。例えば、コネクターに AfterUpdate および BeforeRetrieve ストアード・プロシージャーを使用させる場合には、AfterUpdateSP および BeforeRetrieveSP 属性を追加します。

コネクターでは、以下のビジネス・オブジェクト属性名が認識されます。

```
BeforeCreateSP
AfterCreateSP
CreateSP
BeforeUpdateSP
AfterUpdateSP
UpdateSP
BeforeDeleteSP
AfterDeleteSP
DeleteSP
BeforeRetrieveSP
AfterRetrieveSP
RetrieveSP
BeforeRetrieveByContentSP
AfterRetrieveByContentSP
RetrieveByContentSP
BeforeRetrieveUpdateSP
AfterRetrieveUpdateSP
RetrieveUpdateSP
```

注: コネクターに実行させるストアード・プロシージャーについてのみ、属性を作成してください。コネクターにビジネス・オブジェクトが送信される前に、こ

れらの属性の値を指定するには、アプリケーション固有情報またはマッピング (ICS を統合ブローカーとして使用する場合のみ) を使用します。これらの値に変更が加えられた場合に、それ以後のビジネス・オブジェクトに対するストアード・プロシージャーの呼び出しのためにコネクターにその変更を認識させるには、コネクターを再始動する必要があります。

ストアード・プロシージャーの構文: ストアード・プロシージャーを指定するための構文

```
SPN=StoredProcedureName;RS=true|false[;IP=Attribute_Name1[:Attribute_Name2[:...]]]
[;OP=Attribute_Name1| RS[:Attribute_Name2| RS[:...]]]
[;IO=Attribute_Name1[:Attribute_Name2[:...]]]
```

ここで、以下のように説明されます。

StoredProcedureName
: ストアード・プロシージャーの名前。

RS
: ストアード・プロシージャーが結果セットを戻す場合は true、それ以外の場合は false。デフォルトは false です。この値が true である場合、属性のアプリケーション固有情報に含まれる ColumnName プロパティーは、結果セットの該当するカラムを指します。RS が出力パラメーター・リストの一部である場合は、その特定のパラメーターが結果セットを戻します。1 つの結果セット OUT パラメーターのみがサポートされます。複数の結果セットが OUT パラメーターとして戻された場合は、最初の結果セットのみが戻され、その他の結果セットはすべて無視されます。現在、この機能は Oracle 8i 以上、および Oracle JDBC ドライバーを使用するストアード・プロシージャーについてのみサポートされます。データベース内のストアード・プロシージャーの場合、対応するパラメーターは REFCURSOR タイプを戻します。

IP
: 入力パラメーター。コネクターがストアード・プロシージャーの処理時に入力値として使用する値が格納されているビジネス・オブジェクト属性のリスト。

OP
: 出力パラメーター。コネクターがストアード・プロシージャーの実行後に戻り値を格納するビジネス・オブジェクト属性のリスト。結果セットの記述については、RS を参照してください。

IO
: 入出力パラメーター。コネクターが入力値として使用し、コネクターがストアード・プロシージャーの実行後に戻り値を格納するビジネス・オブジェクト属性のリスト。

注: StoredProcedureName、RS、およびパラメーターの間での順序は重要ですが、パラメーター同士の間での順序は重要ではありません。つまり、ストアード・プロシージャーのパラメーターがタイプ別にまとめて並べられていても、タイプによる区別なく並べられていても、コネクターの動作に違いは生じません。

複数の同じタイプのパラメーターがまとめて並べられている場合は、その値をコロンで区切ります。パラメーター名をそれぞれの値の前に繰り返す必要はありません。タイプの異なるパラメーターの間は、セミコロンで区切ります。パラメーターの値を指定するときには、等号 (=) の前後どちらにも、空白を入れません。

ストアード・プロシージャーの例: 以下の例では、CustomerInsert および VendorInsert というストアード・プロシージャーを示します。これらのストアード・プロシージャーは、2 つの入力属性から値を取得して、4 つの出力属性に値を戻します。これらの例では、ストアード・プロシージャーの構造が異なっています。

- 同じタイプのパラメーターがまとめて並べられているものは、次のとおりです (IP、IP、OP、OP、OP、OP、IO)。

  ```
  SPN=CustomerInsert;RS=false;IP=LastName:FirstName;OP=CustomerName:
  CustomerID:ErrorStatus:ErrorMessage;IO=VendorID
  ```

- 同じタイプのパラメーターがまとめて並べられていないものは、次のとおりです (IP、OP、OP、OP、IP、IO、OP)。

  ```
  SPN=VendorInsert;RS=false;IP=LastName;OP=CustomerName:
  CustomerID:ErrorStatus;IP=FirstName;IO=VendorID;OP=ErrorMessage
  ```

コネクターは JDBC ドライバーがサポートする単純データ型のみをサポートします。

ストアード・プロシージャーの指定: ストアード・プロシージャー名とパラメーター値を指定するには、2 つの方法があります。

- 属性の AppSpecificInfo プロパティー

  ストアード・プロシージャーを指定するテキストの長さが 4000 バイト以下である場合は、属性の AppSpecificInfo プロパティーにその値を指定できます。このプロパティーを使用すると、コネクターがビジネス・オブジェクトのポーリングを実行済みである (つまり、ビジネス・オブジェクトがアプリケーション・イベントを表している) か、あるいはビジネス・オブジェクトを統合ブローカーからの要求として受信済みであるかに関係なく、ストアード・プロシージャーを指定することができます。

  次の例では、アプリケーション固有情報を使用したストアード・プロシージャーの指定を示します。この場合、MaxLength プロパティーに指定されている値は、ストアード・プロシージャーにとって重要ではありません。

  ```
     [Attribute]
     Name = BeforeCreateSP
     Type = String
     MaxLength = 15
     IsKey = false
     IsRequired = false
     AppSpecificInfo =SPN=ContactInsert;IP=LastName:FirstName;OP=CustomerName:
  CustomerID:ErrorStatus:ErrorMessage

  [End]
  ```

- 属性の値 (ICS を統合ブローカーとして使用する場合にのみ関係)

  ストアード・プロシージャーを指定するテキストの長さが 4000 バイトを超える場合は、ストアード・プロシージャーの指定にマッピングを使用する必要があります。マッピングを使用したストアード・プロシージャーの指定は、ビジネス・オブジェクトが統合ブローカーからの要求を表している場合に限り可能です。つ

まり、コネクターがイベントをポーリングしている場合は、ストアード・プロシージャーの指定に属性の値を使用できません。

ストアード・プロシージャーを指定するテキストの長さが 4000 バイトを超えているため、ストアード・プロシージャーの指定にマッピングを使用する場合は、MaxLength プロパティーの値をテキスト全体の長さに合わせて必ず拡張してください。

注: 作成、更新、または削除操作を処理するストアード・プロシージャーが、子ビジネス・オブジェクトの配列が含まれる階層ビジネス・オブジェクトに対して実行されると、コネクターは各子ビジネス・オブジェクトを個別に処理します。例えば、コネクターは、BeforeCreate ストアード・プロシージャーを実行する場合、子ビジネス・オブジェクトの配列をまとめて処理せずに、その配列に含まれるメンバーをそれぞれ処理します。BeforeRetrieve ストアード・プロシージャーを処理する場合には、単一のビジネス・オブジェクトを操作します。AfterRetrieve ストアード・プロシージャーを処理する場合には、検索によって戻されたビジネス・オブジェクトのすべてを操作します。

ストアード・プロシージャーまたは単純な SQL ステートメントを使用したビジネス・オブジェクトの処理

以下のセクションでは、コネクターでのストアード・プロシージャーの処理方法について説明します。

- 『ビジネス・オブジェクトの Create 操作』
- 57 ページの『ビジネス・オブジェクトの Update 操作』
- 57 ページの『ビジネス・オブジェクトの Delete 操作』
- 58 ページの『ビジネス・オブジェクトの Retrieve 操作』
- 59 ページの『ビジネス・オブジェクトの RetrieveByContent 操作』
- 59 ページの『ビジネス・オブジェクトの Retrieve-for-Update 操作』

ビジネス・オブジェクトの Create 操作: Create ストアード・プロシージャーは、通常、コネクターが最上位ビジネス・オブジェクトの単純属性を設定するために使用する値を戻します。コネクターは、Create ストアード・プロシージャー (BeforeCreate、Create、AfterCreate) の処理時に以下のステップを実行します。

1. ビジネス・オブジェクトが BeforeCreateSP 属性を含むかどうかをチェックします。含まれている場合、BeforeCreate ストアード・プロシージャーを呼び出します。
2. ストアード・プロシージャーから出力パラメーターを介して値が戻されれば、その値をビジネス・オブジェクトの単純属性の値の設定に使用します。
3. 単一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。
4. 最上位ビジネス・オブジェクトの外部キー値のそれぞれを、単一カーディナリティーの子オブジェクトのそれぞれの基本キー値に設定します。
5. ビジネス・オブジェクトが CreateSP 属性を含むかどうかをチェックします。含まれている場合、Create ストアード・プロシージャーを呼び出して、最上位ビジネス・オブジェクトを作成します。含まれていない場合は、INSERT ステートメントを作成して実行することにより、最上位ビジネス・オブジェクトを作成します。

6. Create ストアード・プロシージャーから出力パラメーターを介して値が戻されれば、その値をビジネス・オブジェクトの単純属性の値の設定に使用します。
7. 複数カーディナリティーの子のそれぞれの外部キー値を、それらの親の基本キー属性の値に設定します。
8. 複数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。
9. ビジネス・オブジェクトが AfterCreateSP 属性を含むかどうかをチェックします。含まれている場合、AfterCreate ストアード・プロシージャーを呼び出します。
10. ストアード・プロシージャーから出力パラメーターを介して値が戻されれば、その値を、ビジネス・オブジェクトの単純属性の値の設定に使用します。

コネクターはステップ 10 で戻された値を使用して、ステップ 3 またはステップ 5 で作成したビジネス・オブジェクトの値を変更できます。

ビジネス・オブジェクトの Update 操作: Update ストアード・プロシージャーは、通常、コネクターが最上位ビジネス・オブジェクトの単純属性を設定するために使用する値を戻します。コネクターは、Update ストアード・プロシージャー (BeforeUpdate、Update、AfterUpdate) の処理時に以下のステップを実行します。

1. ビジネス・オブジェクトが BeforeUpdateSP 属性を含むかどうかをチェックします。含まれている場合、BeforeUpdate ストアード・プロシージャーを呼び出します。
2. BeforeUpdate ストアード・プロシージャーから出力パラメーターを介して値が戻されれば、その値をビジネス・オブジェクトの単純属性の値の設定に使用します。
3. 単一カーディナリティーの子ビジネス・オブジェクトを更新します。
4. 最上位ビジネス・オブジェクトの外部キー値のそれぞれを、単一カーディナリティーの子オブジェクトのそれぞれの基本キー値に設定します。
5. ビジネス・オブジェクトが UpdateSP 属性を含むかどうかをチェックします。含まれている場合、Update ストアード・プロシージャーを呼び出して、最上位ビジネス・オブジェクトを更新します。含まれていない場合は、UPDATE ステートメントを作成して実行することにより、最上位ビジネス・オブジェクトを更新します。
6. Update ストアード・プロシージャーから出力パラメーターを介して値が戻されれば、その値をビジネス・オブジェクトの単純属性の値の設定に使用します。
7. 複数カーディナリティー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値を参照するように設定します。
8. 複数カーディナリティーの子ビジネス・オブジェクトを更新します。
9. ビジネス・オブジェクトが AfterUpdateSP 属性を含むかどうかをチェックします。含まれている場合、AfterUpdate ストアード・プロシージャーを呼び出します。
10. ストアード・プロシージャーから出力パラメーターを介して値が戻されれば、その値をビジネス・オブジェクトの単純属性の値の設定に使用します。

ビジネス・オブジェクトの Delete 操作: Delete ストアード・プロシージャーは、コネクターに値を戻しません。コネクターは、Delete ストアード・プロシージャー (BeforeDelete、Delete、AfterDelete) の処理時に以下のステップを実行します。

1. ビジネス・オブジェクトが BeforeDeleteSP 属性を含むかどうかをチェックします。含まれている場合、BeforeDelete ストアード・プロシージャーを呼び出します。
2. 単一カーディナリティーの子ビジネス・オブジェクトを削除します。
3. 複数カーディナリティーの子ビジネス・オブジェクトを削除します。
4. ビジネス・オブジェクトが DeleteSP 属性を含むかどうかをチェックします。含まれている場合、Delete ストアード・プロシージャーを呼び出して、最上位ビジネス・オブジェクトを削除します。含まれていない場合、DELETE ステートメントを作成して実行します。
5. ビジネス・オブジェクトが AfterDeleteSP 属性を含むかどうかをチェックします。含まれている場合、AfterDelete ストアード・プロシージャーを呼び出します。

ビジネス・オブジェクトの Retrieve 操作: 単純な検索操作を行う場合には、最上位ビジネス・オブジェクトや単一カーディナリティーの子の他、複数カーディナリティーの子に対しても、ストアード・プロシージャーを使用することができます。ストアード・プロシージャーの順序は、次のとおりです。

- BeforeRetrieve
- Retrieve
- AfterRetrieve

コネクターは、単一カーディナリティーの子ビジネス・オブジェクトや複数カーディナリティーの子ビジネス・オブジェクトの検索の際に、一時オブジェクトを作成します。コネクターは、BeforeRetrieve ストアード・プロシージャーを一時ビジネス・オブジェクトに適用します。また、このコンテナー用に検索された子オブジェクトのそれぞれには、AfterRetrieve ストアード・プロシージャーが適用されます。

AfterRetrieve ストアード・プロシージャーが実行されるのは、ビジネス・オブジェクトのメタデータから動的に生成された Retrieve 照会または同名のストアード・プロシージャーが、ビジネス・オブジェクトに対して実行された後です。

JDBC の仕様によると、StoredProcedure 呼び出しには、次の 3 つのタイプがあります。

- {call <spName>(?,?,?)}
- {call <spName>}
- {?= call <spName>(?,?,?)}

コネクターでは、最初の 2 つのタイプがサポートされています。StoredProcedure から戻される ResultSet を処理します。

ストアード・プロシージャーの構文に RS=true と指定されている場合は、ストアード・プロシージャーから戻された結果セットが処理されます。RS=false の場合は、結果セットは処理されません。デフォルトでは、RS の値は false です。結果セットの値の処理が終了されてから、ストアード・プロシージャーの出力変数が処理されます。RS=true と指定されている場合、複数カーディナリティーの子では、関連するストアード・プロシージャーの出力変数を指定できません。

注: 結果セットの処理のサポートは、Retrieve 動詞操作および RetrieveSP に対してのみ提供されています。

Retrieve ストアード・プロシージャー (RetrieveSP) から戻された結果セットの処理: Retrieve ストアード・プロシージャーから戻された結果セットに対し、ResultSetMetaData が取得されます。結果セット内のすべての列の値が取得され、ビジネス・オブジェクト内の対応する属性に格納されます。属性のアプリケーション固有情報の ColumnName プロパティーには、属性を列と突き合わせる ResultSet 列名が含まれている必要があります。

単一カーディナリティーのオブジェクトに関しては、対応する結果セットは 1 行のみで構成されています。複数の行が結果セット内に含まれて戻された場合、エラーが報告されます。

複数カーディナリティーの子に関しては、結果セットを介して複数の行が戻される場合があります。戻された行ごとに新しいオブジェクトが作成され、コンテナーに追加されます。このコンテナーは、その後親オブジェクトの必須属性索引に追加されます。

ビジネス・オブジェクトの RetrieveByContent 操作: 単純な「RetrieveByContent」操作を行う場合には、最上位ビジネス・オブジェクトとその単一カーディナリティーの子に対してのみ、ストアード・プロシージャーを使用することができます。つまり、結果セットまたは複数の行を戻すためにストアード・プロシージャーを使用することはできません。ストアード・プロシージャーの順序は、次のとおりです。

- BeforeRetrieveByContent
- RetrieveByContent
- AfterRetrieveByContent

コネクターは、単一カーディナリティーの子ビジネス・オブジェクトや複数カーディナリティーの子ビジネス・オブジェクトの検索の際に、一時オブジェクトを作成します。複数カーディナリティーのビジネス・オブジェクトに関しては、BeforeRetrieveByContent ストアード・プロシージャーが一時ビジネス・オブジェクトに適用されます。また、このコンテナー用に検索された子オブジェクトのそれぞれには、AfterRetrieveByContent ストアード・プロシージャーが適用されます。

AfterRetrieveByContent ストアード・プロシージャーが実行されるのは、ビジネス・オブジェクトのメタデータから動的に生成された RetrieveByContent 照会または同名のストアード・プロシージャーが、ビジネス・オブジェクトに対して実行された後です。このとき、階層ビジネス・オブジェクトの検索でもそのビジネス・オブジェクトの子ビジネス・オブジェクトが検索されるにもかかわらず、コネクターは、配列内のすべてのビジネス・オブジェクトに対して AfterRetrieveByContent ストアード・プロシージャーを実行します。

ビジネス・オブジェクトの Retrieve-for-Update 操作: 以下のストアード・プロシージャーが最上位ビジネス・オブジェクトに対して呼び出され、単純な Retrieve と同じ方法で、子ビジネス・オブジェクトのすべてを検索します。

ストアード・プロシージャーの順序は、次のとおりです。

- BeforeRetrieveUpdate

- RetrieveUpdate
- AfterRetrieveUpdate

これらのストアード・プロシージャーは、BeforeRetrieve および AfterRetrieve と同じ操作を実行します。これらの名前は異なっているため、別個の属性を作成して、コネクターに BeforeRetrieve 操作および BeforeRetrieveUpdate 操作を実行させるとともに、AfterRetrieve 操作および AfterRetrieveUpdate 操作を実行させることができます。

コネクターは、単一カーディナリティーの子ビジネス・オブジェクトや複数カーディナリティーの子ビジネス・オブジェクトの検索の際に、一時オブジェクトを作成します。複数カーディナリティーのビジネス・オブジェクトに関しては、BeforeRetrieveUpdate ストアード・プロシージャーが一時ビジネス・オブジェクトに適用されます。また、このコンテナー用に検索された子オブジェクトのそれぞれには、AfterRetrieveUpdate ストアード・プロシージャーが適用されます。

AfterRetrieveUpdate ストアード・プロシージャーが実行されるのは、ビジネス・オブジェクトのメタデータから動的に生成された RETRIEVE 照会または同名のストアード・プロシージャーが、ビジネス・オブジェクトに対して実行された後です。このとき、階層ビジネス・オブジェクトの検索でもそのビジネス・オブジェクトの子ビジネス・オブジェクトが検索されるにもかかわらず、コネクターは、配列内のすべてのビジネス・オブジェクトに対して AfterRetrieveUpdate ストアード・プロシージャーを実行します。

トランザクション・コミットとロールバック

コネクターは、処理すべきビジネス・オブジェクトを受信すると、必ずトランザクション・ブロックを開始します。コネクターがそのビジネス・オブジェクトを処理するときに実行する SQL ステートメントのすべてが、そのトランザクション・ブロック内にカプセル化されます。コネクターは、そのビジネス・オブジェクトの処理に成功した場合には、処理の終了後、そのトランザクション・ブロックをコミットします。エラーが発生した場合は、トランザクションをロールバックします。

ビジネス・オブジェクトの属性プロパティー

ビジネス・オブジェクトのアーキテクチャーでは、属性に適用されるさまざまなプロパティーが定義されています。このセクションでは、これらのプロパティーの一部について、コネクターでどのように解釈されるかを説明します。また、ビジネス・オブジェクトを変更する場合に、どのように設定すればよいかについても説明します。

Name プロパティー

どのビジネス・オブジェクト属性にも、固有の名前が含まれていなければなりません。

Type プロパティー

どのビジネス・オブジェクト属性にも、Integer や String などの型か、子ビジネス・オブジェクトのタイプが含まれていなければなりません。コネクターでは、Date、Long Text、または String 型の属性を検出すると、その値を引用符で囲み、文字データとして取り扱います。

Cardinality プロパティー

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) を表すビジネス・オブジェクト属性では、いずれもこのプロパティーに値 1 または n が設定されています。子ビジネス・オブジェクトを表す属性はすべて、ContainedObjectVersion プロパティー (子のバージョン番号を指定) と Relationship プロパティー (値の Containment を指定) を持ちます。

Max length プロパティー

String 型の属性では、このプロパティーにより、その属性の値に許可される最大長が指定されます。

Key プロパティー

どのビジネス・オブジェクトでも、1 つ以上の単純属性がキーに指定されなければなりません。属性をキーとして定義するには、このプロパティーを Yes に設定します。ビジネス・オブジェクト属性の型が String の場合は、データベースのデータ型は型 char ではなく、Varchar にしてください。

注: コネクターでは、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) を表す属性をキー属性に指定することについては、サポートしていません。

単純属性のキー・プロパティーを true に設定すると、コネクターは、ビジネス・オブジェクトの処理中に生成する SELECT、UPDATE、RETRIEVE、および DELETE の各 SQL ステートメントの WHERE 文節にその属性を追加します。

親/子関係を子に格納する子 (複数カーディナリティーであるか、単一カーディナリティーであるかを問いません) に含まれるある属性で、キー・プロパティーが true に設定されている場合、コネクターは、親の基本キーを SELECT ステートメントの WHERE 文節に使用します。Key プロパティーは使用しません。子の外部キー属性をセットするために値が使用されるビジネス・オブジェクト属性の、名前を指定する方法は、65 ページの『属性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。

Foreign key プロパティー

コネクターでは、このプロパティーを使用して、属性が外部キーであるかどうかを判別します。

Required プロパティー

Required プロパティーは、属性が値を必要とするかどうかを指定します。

このプロパティーが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属性に対して指定されている場合、その親ビジネス・オブジェクト内には、この属性の子ビジネス・オブジェクトが含まれている必要があります。

コネクターでは、Create 要求を伴うビジネス・オブジェクトを受信しても、以下の条件の両方が true である場合には、作成操作を失敗させます。

- 受信したビジネス・オブジェクトの必須属性に、有効な値またはデフォルト値が含まれていない場合。
- アプリケーション固有情報に、コネクターでの固有 ID の生成が指定されていない場合。

また、Retrieve 要求を伴うビジネス・オブジェクトを受信しても、そのビジネス・オブジェクトの必須属性に有効な値またはデフォルト値が含まれていない場合には、検索操作を失敗させます。

コネクターは、子ビジネス・オブジェクトの配列を含む属性に関しては、このプロパティーを使用しません。

注: キー属性がシーケンス、またはカウンターを使用する場合、あるいはデータベースから取り込まれる場合 (UID=AUTO) は、Required にしないでください。

AppSpecificInfo

このプロパティーに関する詳細については、65 ページの『属性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。

Default value プロパティー

このプロパティーは、データベース表の値が格納されない単純属性に値を設定するために、コネクターで使用される値 (デフォルト値) を指定します。コネクターは、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) を表す属性に関しては、このプロパティーを評価しません。

UseDefaults 構成プロパティーが true に設定されている場合にのみ、コネクターはこのプロパティーを評価します。詳細については、19 ページの表 6 を参照してください。

特殊属性値

ビジネス・オブジェクトの単純属性には、特殊値 CxIgnore を格納することができます。コネクターでは、統合ブローカーからビジネス・オブジェクトを受信したとき、値が CxIgnore の属性をすべて無視します。それらの属性は、不可視として取り扱われます。

コネクターでは、データベースからのデータの受信時に、SELECT ステートメントからある属性に対して null 値が戻された場合には、デフォルトでその属性の値を CxIgnore に設定します。その属性のアプリケーション固有情報の UNVL パラメーターに値がすでに指定されている場合には、コネクターはその値を使用して null を表します。

コネクターの要件として、すべてのビジネス・オブジェクトに 1 つ以上の基本キー属性が含まれていなければなりません。したがって、開発者は、コネクターに渡さ

れる WebSphere Business Integration Adapter ビジネス・オブジェクトに、CxIgnore に設定されていない基本キーが、必ず 1 つ以上含まれるようにしてください。この要件の例外は、コネクター がカウンターまたはシーケンスを使用して、あるいはデータベースによって基本キーを生成するビジネス・オブジェクトの場合のみです。

コネクターは、データベースへのデータの挿入時に、値が指定されていないビジネス・オブジェクト属性があると、その属性の UseNullValue プロパティーに指定されている値を使用します。UseNullValue に関する詳細については、 66 ページの表 11 の UNVL=value を参照してください。

ビジネス・オブジェクトのアプリケーション固有の情報

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、コネクターに対し、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。コネクターでは、ビジネス・オブジェクトの属性または動詞、あるいはビジネス・オブジェクト自体から取得したアプリケーション固有情報を解析して、作成、更新、検索、および削除操作のための照会を生成します。

コネクターは、ビジネス・オブジェクトのアプリケーション固有情報の一部については、キャッシュに保管し、その情報をすべての動詞の照会をビルドするために使用します。

アプリケーション固有のビジネス・オブジェクトを拡張、または変更する場合は、ビジネス・オブジェクト定義のアプリケーション固有情報が、コネクターの予期する構文に必ず合致するよう確認してください。

このセクションでは、コネクターがサポートするビジネス・オブジェクトのための、オブジェクト・レベル、属性、および動詞に関するアプリケーション固有情報について、その形式に関する情報を提供します。

表 10 に、ビジネス・オブジェクトのアプリケーション固有情報で使用可能な機能の概要を示します。

表 10. サポートされているビジネス・オブジェクトのアプリケーション固有情報の概説

アプリケーション固有情報の有効範囲	機能
ビジネス・オブジェクト全体	以下のものを指定します。 • 対応するデータベース表の名前。 • コネクターが論理 (ソフト) 削除実行のために WHERE 文節内に使用する値が含まれる列を定義します。 • 最上位ビジネス・オブジェクトがラッパーであることを指定します。

表 10. サポートされているビジネス・オブジェクトのアプリケーション固有情報の概説 (続き)

アプリケーション固有情報の有効範囲	機能
単純属性	以下のものを指定します。 • 属性に対応するデータベース列名。 • 現在のビジネス・オブジェクトの属性と親 (または子) ビジネス・オブジェクトの間の外部キー関係。 • 固有 ID 値の自動生成。 • コネクターが現在の属性の値を設定するために必要とする値が含まれる、同一ビジネス・オブジェクト内の別の属性の名前。 • 検索のソート時に現在の属性を使用するかどうか。 • 現在の属性の値が null の場合に使用する値。 • ストリング置換時の動作。 • ストリングの比較時に LIKE 演算子または = 演算子のどちらを使用するか。 • LIKE 演算子の使用時に、ワイルドカード位置として使用する値。
子または子ビジネス・オブジェクト配列を含む属性	単一カーディナリティーの子が親に所有されているかどうかを指定します。更新操作において、子データがソース・ビジネス・オブジェクトにない場合、コネクターがその子データを削除するかどうかを指定します。
ビジネス・オブジェクト動詞	動詞 Retrieve に対してのみ使用されます。テキストを使用して、検索時に WHERE 文節に組み込む属性を指定します。演算子や属性値を指定することもできます。

以下のセクションでは、この機能をより詳細に解説します。

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

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り、次のことが可能になります。

- 対応するデータベース表の名前の指定
- 物理削除または論理削除の実行に必要な情報の提供
- 最上位ビジネス・オブジェクトがラッパー・オブジェクトであることを指定

ビジネス・オブジェクト・レベルでは、アプリケーション固有情報の形式は、コロン (:) またはセミコロン (;) によって区切られた複数のパラメーターで構成されています。

```
TN=TableName; SCN=StatusColumnName:StatusValue
```

では、TableName はデータベース表を示します。また、*StatusColumnName* は論理削除の実行に使用されるデータベース列の名前であり、*StatusValue* はビジネス・オブジェクトが非アクティブまたは削除済みであることを示す値です。

例えば、Customer ビジネス・オブジェクトで、そのアプリケーション固有情報に、以下の値が指定されているとします。

```
TN=CUSTOMER; SCN=CUSTSTATUS:DELETED
```

また、コネクターで、カスタマー削除要求を受信したとします。上記のような値が指定されている場合、コネクターは次の SQL ステートメントを発行します。

```
UPDATE CUSTOMER SET CUSTSTATUS = 'DELETED' WHERE CUSTOMER_ID = 2345
```

コネクターは、SCN パラメーターが含まれていない場合や、このパラメーターに値が指定されていない場合には、ビジネス・オブジェクトをデータベースから物理的に削除します。つまり、Delete 動詞を伴うビジネス・オブジェクトで、アプリケーション固有情報に SCN パラメーターが含まれている場合、コネクターは論理削除を実行します。Delete 動詞を持つビジネス・オブジェクトが SCN パラメーターをアプリケーション固有情報内に含まない場合、コネクターは物理削除を実行します。

SCN プロパティーの値は、更新操作と削除操作の両方で使用できます。

- 更新を実行するとき、コネクターは `ChildUpdatePhyDelete` プロパティーの値を使用して、欠落している子データを物理的に削除するか、論理的に削除するかを判断します。子のデータを論理的に削除する場合には、SCN パラメーターの値を使用して、状況列の名前と状況値を示すテキストを取得します。詳細については、48 ページの『Update 操作』を参照してください。
- コネクターは、削除の実行時には、SCN パラメーターの値に基づいて、ビジネス・オブジェクト全体を物理的に削除するか、論理的に削除するかを決定します。SCN パラメーターに値が含まれている場合は、論理削除を実行します。SCN パラメーターに値が含まれていない場合は、物理削除を実行します。詳細については、51 ページの『Delete 操作』を参照してください。

ビジネス・オブジェクト・レベルでは、アプリケーション固有情報はラッパーの指定に使用される場合があります。

```
WRAPPER=true|false
```

wrapper パラメーターが true に設定されている場合、最上位ビジネス・オブジェクトはラッパー・オブジェクトです。ラッパー・オブジェクトはデータベース表やビューによって表されません。ラッパーは関連のないビジネス・オブジェクトのコンテナーとして使用されます。コネクターは最上位オブジェクトを無視し、子のみを処理します。ラッパー・オブジェクトには N のカーディナリティーを持つエンティティーまたは N-1 のカーディナリティーを持つエンティティー、あるいはその両方を含めることができます。

属性レベルのアプリケーション固有情報

属性のアプリケーション固有情報は、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。さらに、子を表す属性のアプリケーション固有情報は、親/子関係が子に保管されるか、親に保管されるかによって異なります。子または子ビジネス・オブジェクト配列を表す属性のアプリケーション固有情報については、68 ページの『属性の外部キーの指定』を参照してください。

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、名前と値のペアを表す 11 個のパラメーターで構成されています。どのパラメーターにも、パラメーター名とその値が含まれます。各パラメーター・セットはコロンで次のものと区切られます。

属性のアプリケーション固有情報の形式は、次のとおりです。大括弧 ([ ]) で囲まれた部分は、オプションのパラメーターです。縦線 (|) は、選択可能なオプションの区切りです。コロンは、区切り文字として予約されています。

```
CN=col_name:[FK=[fk_object_name.]fk_attribute_name]:
[UID=[AUTO|uid_name| schema_name.uid_name[=UseIfMissing]|CW.uidcolumnname
[=UseIfMissing]]]:
[PH=true|false]:[CA=set_attr_name| ..set_attr_name]:[OB=[ASC|DESC]]:[UNVL=value]:
[ESC=true|false]:[FIXEDCHAR=true|false]:
[BYTEARRAY=true|false]:[USE_LIKE=true|false:
[WILDCARD_POSITION=non-negative number|NONE|BEGIN|END|BOTH]]:
[CLOB=true]
```

コネクターで処理する単純属性に必須のパラメーターは、列名だけです。例えば、列名だけを指定するには、次の形式を使用します。

```
CN=customer_id
```

表 11 に、それぞれの名前と値のペアを表すパラメーターを示します。

表 11. 属性アプリケーション固有情報内の名前と値のペアを表すパラメーター

パラメーター	説明
CN=*col_name*	このパラメーターが指定されている属性に対応するデータベース列の名前。
FK=[*fk_object_name*.]*fk_attribute_name*	このプロパティーの値は、親子関係が親ビジネス・オブジェクトまたは子に格納されているかどうかによって異なります。属性が外部キーでない場合は、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。詳細については、68 ページの『属性の外部キーの指定』を参照してください。
UID=AUTO UID=*uid_name*| schema_name.uid_name [=UseIfMissing] UID=CW.*uidcolumnname*[=UseIfMissing]	コネクターでは、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。属性で固有 ID の生成が必要とされていない場合には、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。ビジネス・オブジェクトの処理中に固有の ID を保存する方法の詳細は、PreserveUIDSeq プロパティー記述を参照してください。詳細については、71 ページの『ビジネス・オブジェクトの固有 ID の生成』を参照してください。 **注:** CW は、UID の型を示すために使用されるキーワードであり、表名を示すものではありません。
PH=true|false	PH=true の場合、対応する単純属性はプレースホルダー属性です。単純属性は、アプリケーション固有の情報 (ASI) がブランクまたはヌルの場合も、プレースホルダーとなります。

表 11. 属性アプリケーション固有情報内の名前と値のペアを表すパラメーター (続き)

パラメーター	説明
CA=*set_attr_name*| ..set_attr_name	set_attr_name が、現在の個別ビジネス・オブジェクトに含まれる別の属性の名前に設定されている場合、コネクターは、作成操作の際に、その指定されている属性の値を使用して、このパラメーターが指定されている属性の値を設定します。この属性値の設定は、ビジネス・オブジェクトがデータベースに追加される前に行われます。set_attr_name の値は子ビジネス・オブジェクトの属性を参照できませんが、set_attr_name の前にピリオドが 2 つある場合は親ビジネス・オブジェクトの属性を参照できます。このパラメーターがアプリケーション固有情報に含まれていない場合、コネクターは、別の属性の属性値をコピー (CA) せずに、現在の属性の値を使用します。
OB=[ASC|DESC]	このパラメーターに値が指定されている場合、このパラメーターが指定されている属性が子ビジネス・オブジェクト内に存在するものであれば、コネクターでは、検索照会の ORDER BY 文節に、その属性の値を使用します。コネクターは、子ビジネス・オブジェクトを昇順または降順で検索することができます。昇順での検索を指定するには ASC を使用します。降順での検索を指定するには DESC を使用します。このパラメーターがアプリケーション固有情報に含まれていない場合、コネクターは、検索順序を指定するときに、このパラメーターが指定されている属性を使用しません。
UNVL=*value*	値が null の属性を含むビジネス・オブジェクトが検索された場合に、null 表現用としてコネクターに使用させる値を指定します。このパラメーターがアプリケーション固有情報に含まれていない場合、コネクターは、その属性の値として CxIgnore を挿入します。
ESC=[true|false]	コネクターが、ReplaceAllStr プロパティーで特定された各文字のすべてのインスタンスを、ReplaceStrList プロパティーで指定された置換ストリングに置き換えるかどうかを決定します。このパラメーターが値を含んでいなければ、コネクターは ReplaceStrList プロパティーの値を使用して決定します。 **注:** ESC パラメーター、ReplaceAllStr プロパティー、および ReplaceStrList プロパティーは、データベース・エスケープ文字機能 (単一引用符のエスケープなど) のサポートも提供します。JDBC ドライバーによって提供される Prepared Statements でも同じ機能が利用できるため、今後リリースされるコネクターでは、こうしたプロパティーのサポートが廃止される予定です。現在、コネクターは JDBC 準備済みステートメントの使用をサポートしています。
FIXEDCHAR=true|false	表内の列が (VARCHAR 型ではなく) CHAR 型である場合に、このパラメーターが指定されている属性を固定長とするかどうかを指定します。例えば、ある特定の属性が CHAR 型の列にリンクされている場合は、その属性のアプリケーション固有の情報では FIXEDCHAR=true が指定されるため、コネクターはその属性の値を**固定長**と見なします。このパラメーターが指定されている属性の MaxLength プロパティーの指定値は、データベース内に指定されている CHAR の長さと一致するようにしてください。デフォルトでは FIXEDCHAR=false です。
BYTEARRAY=true|false	BYTEARRAY=true の場合、コネクターはデータベースに対するバイナリー・データの読み取りおよび書き込みを実行し、そのデータをストリングとして ICS または WebSphere Integrator Broker に送信します。 BYTEARRAY=false がデフォルトです。詳細については、73 ページの『バイナリー・データを使用した作業』を参照してください。

表 11. 属性アプリケーション固有情報内の名前と値のペアを表すパラメーター (続き)

パラメーター	説明
USE_LIKE=true|false	コネクターがストリングを比較する時に = 演算子または LIKE 演算子のどちらを使用するかを指定します。USE_LIKE が true に設定されている場合、ワイルドカード照会を実行するには WILDCARD_POSITION を設定します。USE_LIKE が false に設定されている場合は、= 演算子が使用されます。
WILDCARD_POSITION=non-negative number|NONE|BEGIN|END|BOTH	USE_LIKE が true の場合、ワイルドカードの位置を指定するために WILDCARD_POSITION が使用されます。この値は負以外の任意の数値、NONE、BEGIN、END、または BOTH に設定できます。例えば、BEGIN を使用すると、ワイルドカード文字がストリングの先頭に置かれます (%string)。END を使用すると、ワイルドカード文字がストリングの末尾に置かれます (string%)。BOTH を使用すると、ワイルドカード文字がストリングの先頭と末尾の両方に置かれます (%string%)。
CLOB=true	String 属性タイプにのみ適用可能。この属性に対応するデータベース列が CLOB データ型であることを指定します。 **注:** CLOB データ型については、以下のように定義されています。 • CLOB に対応する属性では Type が String に設定されており、長さを示す値は CLOB の長さを規定するために使用されています。 • CLOB に対応する属性では、AppSpecificInfo=CN=xyz; CLOB=true と指定されています。 • その他のタイプの属性の ASI で CLOB を使用すると、エラーが発生します。 • CLOB=false と指定すると、エラーが発生します。 通常の String 型の属性は CLOB 対応の属性とほぼ同じですが、ASI に CLOB が使用されていません。CLOB データ型を使用する場合、4 KB 以上のサイズのデータを挿入または更新することができます。ただし、このデータ型を使用できるのは Oracle に限られており、また、Oracle でこのデータ型を使用するためには CLOB をサポートするシン・ドライバーが必要です。それ以外のドライバーを使用すると、エラーが発生する可能性があります。

注: ビジネス・オブジェクトのどの属性にも、コネクターに照会を作成または実行させるアプリケーション固有情報が含まれない場合、コネクターは警告を記録して、動作を継続します。例外を throw することや、失敗を戻すことはありません。

属性の外部キーの指定: このプロパティーの値は、親/子関係が親ビジネス・オブジェクトに保管されるか、子ビジネス・オブジェクトに保管されるかによって異なります。

- 親に保管される場合: 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含まれるように値を設定します。
- 子に保管される場合: 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。

fk_object_name の値が子ビジネス・オブジェクトのタイプに一致しない場合や、*fk_attribute_name* の値が親または子 (いずれか該当する方) の属性の名前に一致し

ない場合には、コネクターでは指定されている属性を外部キーとして処理することができません。ビジネス・オブジェクト名、および属性名は大文字小文字を区別します。

例えば、Customer ビジネス・オブジェクトに、Address 子ビジネス・オブジェクトを表す Addr[1] 属性と、この子ビジネス・オブジェクトの基本キーが外部キーとして格納される AID 属性が含まれているとします。この場合、親の外部キー属性のアプリケーション固有情報には、子ビジネス・オブジェクトのタイプ (Address) と基本キー属性の名前 (ID) が含まれていなければなりません。この例では、AID 属性のアプリケーション固有の情報に FK=Address.ID が含まれます。

外部キー属性の命名: 複数の親ビジネス・オブジェクトに、同一の子ビジネス・オブジェクトを含めることができます。このとき、子ビジネス・オブジェクトは、単一カーディナリティーの関係にあっても、複数カーディナリティーの関係にあってもかまいません、また、親/子関係は、親に保管しても、子に保管してもかまいません。ただし、親/子関係が保管される親ビジネス・オブジェクトは、いずれも同じ名前の属性を使用して、子の基本キーを格納しなければなりません。さらに、親/子関係が保管される子ビジネス・オブジェクトは、いずれも同じ名前の属性を使用して、親の基本キーを格納しなければなりません。図 5 にこれらの関係を示します。

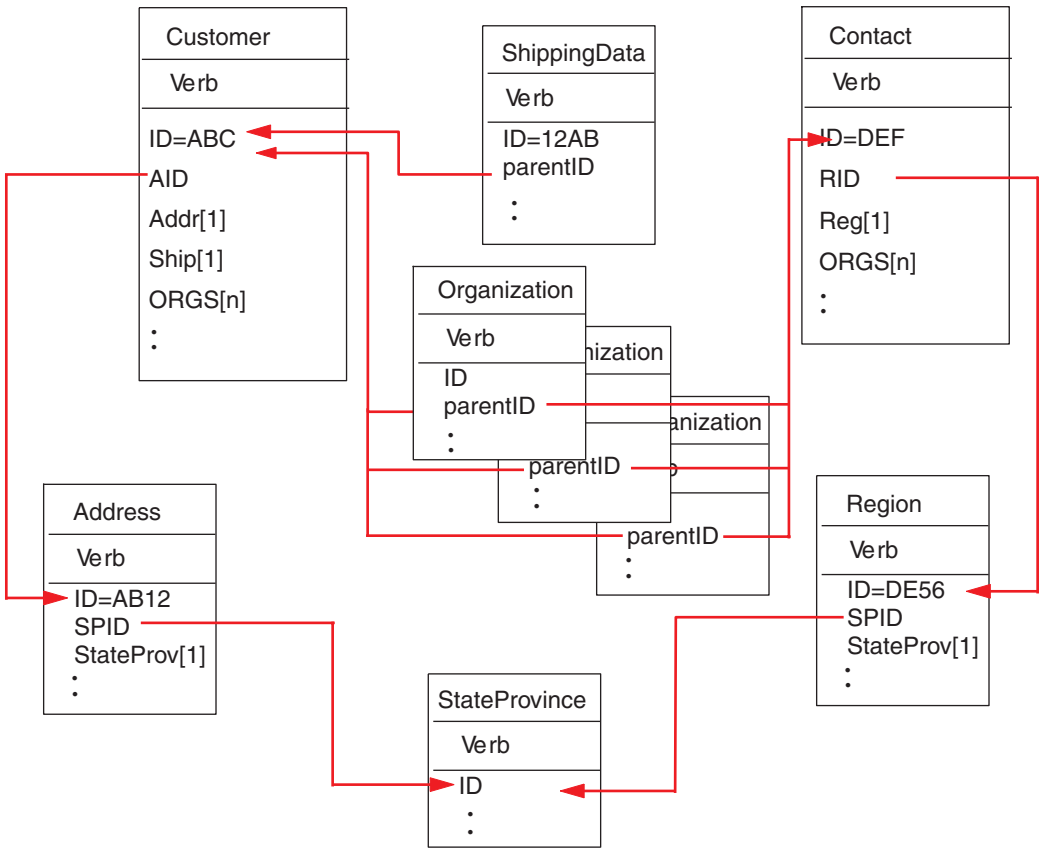


図 5. ビジネス・オブジェクト間の関係の例

図 5 に以下の関係を示します。

- Customer (ID=ABC) および Contact (ID=DEF) の ORGS[n] 属性は、Organization ビジネス・オブジェクトの配列を表しています。Organization ビジネス・オブジ

ェクトの配列に含まれる各ビジネス・オブジェクトの外部キー値は、Customer ビジネス・オブジェクトおよび Contact ビジネス・オブジェクトに含まれる ID 属性の基本キー値に対応しています。この場合、配列内の各ビジネス・オブジェクトは、複数の親に含まれています。

ORGS 属性のアプリケーション固有の情報は以下のようになっています。

```
KEEP_RELATIONSHIP=true
```

KEEP_RELATIONSHIP パラメーターに関する詳細については、72 ページの『子を表す属性のアプリケーション固有情報』を参照してください。

Organization ビジネス・オブジェクトの配列内の各子ビジネス・オブジェクトに含まれる parentID 属性のアプリケーション固有情報には、この属性に対応するデータベース内の列の名前が含まれています。また、この属性の外部キーとなる、親の基本キー属性の名前も含まれています。これらは、次のような形式で指定されています。

```
CN=ORG_ID:FK=ID
```

注: 親/子関係を子に保管する手法で同一の子を複数のビジネス・オブジェクトに含める場合、すべての親ビジネス・オブジェクトにおいて、子の外部キーが格納される属性の名前が同一である必要があります。子ビジネス・オブジェクトのアプリケーション固有情報の外部キー・パラメーター (FK) には、この属性の名前のみを指定します。親ビジネス・オブジェクトのタイプは指定しません。コネクターでは、どの子についても、その直接の親が所有者であると見なされます。

- Customer の Addr[1] 属性は、所有関係にある Address ビジネス・オブジェクトを表します。Customer の AID 属性では、Address ビジネス・オブジェクトの基本キーが、親の外部キーに指定されています。この場合、親の外部キー属性には、子ビジネス・オブジェクトの基本キー属性の名前だけでなく、子ビジネス・オブジェクトのタイプも含まれていなければなりません。単一カーディナリティーの子である Address を含む親は 1 つだけです。

  Addr 属性のアプリケーション固有の情報は、次のとおりです。

  ```
  CONTAINMENT=OWNERSHIP
  ```

  AID 属性のアプリケーション固有情報には、この属性に対応するデータベース列の名前が含まれています。また、この属性の外部キーが、子ビジネス・オブジェクトのタイプと子ビジネス・オブジェクトの基本キー属性名を使用して指定されています。これらは、次のような形式で指定されています。

  ```
  CN=FK_AD:FK=Address.ID
  ```

  子ビジネス・オブジェクトの基本キー属性のアプリケーション固有情報は、次のとおりです。

  ```
  CN=pk
  ```

- Address ビジネス・オブジェクトおよび Region ビジネス・オブジェクトの StateProv [1] 属性は、所有関係にない StateProvince ビジネス・オブジェクトを表しています。Address ビジネス・オブジェクトおよび Region ビジネス・オブジェクトの SPID 属性には、子ビジネス・オブジェクトのタイプ (StateProvince)

と、この子ビジネス・オブジェクトの基本キー属性 (親の外部キー) の名前が含まれています。このようにして、複数の親に、同じ単一カーディナリティーの子 (StateProvince) が含まれています。

SPID 属性のアプリケーション固有の情報は、次のとおりです。

```
CONTAINMENT=NO_OWNERSHIP
```

CONTAINMENT パラメーターに関する詳細については、72 ページの『子を表す属性のアプリケーション固有情報』を参照してください。

Address ビジネス・オブジェクトに含まれる Address SPID 属性のアプリケーション固有情報には、この属性に対応するデータベース列の名前が含まれています。また、この属性の外部キーが、子ビジネス・オブジェクトのタイプと子ビジネス・オブジェクトの基本キー属性名を使用して指定されています。これらは、次のような形式で指定されています。

```
CN=FK_SP:FK=StateProvince.ID
```

子ビジネス・オブジェクトの基本キー属性のアプリケーション固有情報は、次のとおりです。

```
CN=SP_ID
```

注: 親/子関係を親に保管する手法で同一の子を複数のビジネス・オブジェクトに含める場合、すべての子ビジネス・オブジェクトにおいて、親の外部キーが格納される属性の名前が同一である必要があります。

- Customer の Ship [1] 属性は、カスタマー向け出荷情報が格納されている、ShippingData ビジネス・オブジェクトを表しています。Customer の ID 属性は、この出荷データの外部キーとして機能します。この場合、ShippingData はその親から独立して存在できず、親が作成された後でなければ作成されないものであるため、親/子関係は子に保管されます。

  この子の parentID 属性のアプリケーション固有情報には、この属性に対応するデータベース列の名前が含まれています。また、この属性の外部キーが、親の基本キー属性名を使用して指定されています。これらは次のような形式で指定されています。

  ```
  CN=SD_ID:FK=ID
  ```

ビジネス・オブジェクトの固有 ID の生成: コネクターでは、UID パラメーターを参照して、ビジネス・オブジェクトの固有 ID を生成します。コネクターは、Oracle 用のシーケンスまたは表構造のカウンターを使用して固有 ID を生成した後、INSERT ステートメントを発行します。

IBM DB2 および Microsoft SQL Server では、INSERT ステートメントで ID を渡す必要はありません。その代わりに、作成時に ID を生成します。ビジネス・オブジェクトの作成が正常に完了すると、コネクターでこの値を検索および使用することができます。

コネクターは、シーケンスまたはカウンターを使用して ID 値を生成し、その後で INSERT ステートメントを発行します。

- UID = AUTO の場合、データベースで ID が生成され、コネクターでは生成された ID を検索しなければなりません。この設定は、IBM DB2 および Microsoft SQL Server データベースに対してのみ使用できます。

- UID =*uid_name* の場合、*uid_name* の値は、このパラメーターが指定されている属性の固有 ID を生成するためにコネクターが使用する Oracle シーケンスの名前を示します。コネクターは、このシーケンス値を取り出すと、キー属性に値を格納して、INSERT ステートメントを発行します。この構文は、現在は Oracle データベース用にのみ使用されています。
- UID = *uid_name*=UseIfMissing であり、属性の値が CxIgnore でない場合は、コネクターは、固有 ID を生成せずに属性の値を使用します。=UseIfMissing パラメーターにはブランクを入れることはできず、大文字と小文字は区別されません。このオプションは、Oracle データベースに対してのみ使用できます。
- UID=CW.*uidcolumnname* の場合は、コネクターはカウンター表を使用して属性の固有 ID を生成します。この表の作成時には、id 列のみが含まれています。この表の名前は変更することができます。この表をカスタマイズすることにより、UID (固有 ID) の生成を必要とする属性ごとに列を 1 つずつ追加することができます。固有 ID の生成時にコネクターに使用させる列の名前を指定するには、*uidcolumnname* パラメーターを使用します。コネクターでは、UID の生成を必要とする列に関しては数値データ型のみがサポートされていることに注意してください。

  表の名前の構成方法については、UniqueIDTableName を参照してください。この表をインストールするためのスクリプトは、次のとおりです。

  ¥connectors¥JDBC¥dependencies¥uid_table_oracle.sql

  ¥connectors¥JDBC¥dependencies¥uid_table_mssqlserver.sql

  ¥connectors¥JDBC¥dependencies¥uid_table_db2.sql
- UID=CW.*uidcolumnname*=UseIfMissing であり、属性の値が CxIgnore でない場合は、コネクターは、固有 ID を生成せずに属性の値を使用します。=UseIfMissing パラメーターにはブランクを入れることはできず、大文字と小文字は区別されません。

処理中に固有 ID シーケンスを保持する方法については、27 ページの『PreserveUIDSeq』プロパティーを参照してください。

子を表す属性のアプリケーション固有情報

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属性では、その子が親に所有されるか、または複数の親の間で共用されるかを指定することができます。

単一カーディナリティーの子、または子ビジネス・オブジェクト配列を表す属性は、親および子のサブセットを更新するときのコネクターの動作を指定できます。

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属性: 単一カーディナリティーの子を表す属性の、アプリケーション固有情報の形式は、次のとおりです。

```
CONTAINMENT= [OWNERSHIP|NO_OWNERSHIP]
```

親ビジネス・オブジェクトが子ビジネス・オブジェクトを所有する単一カーディナリティーの関係を表すには、CONTAINMENT を OWNERSHIP に設定します。親ビジネス・オブジェクトが子ビジネス・オブジェクトを共用する単一カーディナリティー

の関係を表すには、CONTAINMENT を NO_OWNERSHIP に設定します。関係が親ではなく子に保管される単一カーディナリティーの関係を表す場合は、CONTAINMENT パラメーターを含めないでください。

詳細については、 39 ページの『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』および 41 ページの『関係を子に格納する単一カーディナリティー関係』を参照してください。

親のキーを保管する子を表す属性: 親/子関係を子に保管するビジネス・オブジェクトの配列に対する更新操作に関しては、その子を表す属性でのみ使用できる、KEEP_RELATIONSHIPが用意されています。これを true に設定すると、既存の子のデータがソース・ビジネス・オブジェクト内に表されていない場合に、コネクターがそのデータを削除するのを防ぐことができます。

例えば、ある既存の契約が、既存サイトである New York に関連付けられているとします。また、コネクターで Contract ビジネス・オブジェクトの更新要求を受信し、その要求には San Francisco をサイトとして関連付けるための子ビジネス・オブジェクトが 1 つだけ含まれているとします。サイト・データを表す属性の KEEP_RELATIONSHIP が true である場合、コネクターは、既存の契約を更新してその契約に San Francisco との関連付けを追加しますが、その契約の New York との関連付けは削除しません。

しかし、KEEP_RELATIONSHIP が false である場合には、コネクターは、既存の子のデータのうち、ソース・ビジネス・オブジェクトに含まれないものすべてを削除します。この場合、更新対象の契約は、San Francisco のみに関連付けられることになります。

このアプリケーション固有情報の形式は、次のとおりです。

```
KEEP_RELATIONSHIP=[true|false]
```

このアプリケーション固有情報の検査の際には、大文字小文字は区別されません。

バイナリー・データを使用した作業: BYTEARRAY=true の場合、コネクターはデータベースに対するバイナリー・データの読み取りおよび書き込みを実行します。WebSphere Business Integration システムの現行バージョンではバイナリー・データがサポートされないため、バイナリー・データは String に変換されてから統合ブローカーに送信されます。このストリングの形式は、1 バイトにつき 2 文字を使用した 16 進数になります。例えば、データベース内のバイナリー・データが 3 バイトで (10 進数の) 値が (1, 65, 255) の場合、ストリングは ″0141ff″ となります。

動詞のアプリケーション固有情報形式

コネクターでは、Retrieve および RetrieveByContent 動詞の場合に、動詞に関するアプリケーション固有情報を使用します。テキストで記述されるこの情報を使用して、検索時に WHERE 文節に組み込まれる属性を指定することができます。演算子や属性値を指定することもできます。

Retrieve および RetrieveByContent 動詞用のアプリケーション固有情報の構文を以下に示します。

```
[condition_variable conditional_operator @ [...]:[..]attribute_name [, ...]]
```

ここで、以下のように説明されます。

condition_variable	データベース列の名前。
conditonal_operator	データベースによってサポートされている演算子 (例: =、>、OR、AND、IN (*value1*,*value2*)。
@	getAttrValue(*attribute_name*) によって取得した値で置換される変数。置換は定位置形式です。このため、コネクターは、: 区切り文字の後に指定された最初の *attribute_name* 変数 の値で最初の @ を置換します。
..	attribute_name 変数に指定されている属性は、直接の親にあたるビジネス・オブジェクトに属すると見なされます。この値が欠落している場合は、現在のビジネス・オブジェクトに属すると見なされます。
attribute_name	@ を置換するための値が含まれる属性の名前。

このプロパティーの構文を理解するため、まず、Item ビジネス・オブジェクトに、値が XY45 の item_id 属性と、値が RED の Color 属性が含まれていると考えてみます。さらに、Retrieve 動詞の AppSpecificInfo プロパティーに、次のように指定したとします。

```
Color='RED'
```

このようにアプリケーション固有情報の値が指定されている場合、コネクターは次の WHERE 文節を検索用に作成します。

```
where item_id=XY45 and Color = 'RED'
```

さらに複雑な例としては、Customer ビジネス・オブジェクトに、値が 1234 の customer_id 属性と、値が 01/01/90 の creation_date 属性が含まれていると考えてみます。また、このビジネス・オブジェクトの親には、値が 20 の quantity 属性が含まれているとします。

さらに、Retrieve 動詞の AppSpecificInfo プロパティーに、次のように指定したとします。

```
creation_date > @ OR quantity = @ AND customer_status IN
('GOLD', 'PLATINUM') : creation_date, ..quantity
```

このようにアプリケーション固有情報の値が指定されている場合、コネクターは次の WHERE 文節を検索用に作成します。

```
where customer_id=1234 and creation_date > '01/01/90'
OR quantity = 20 AND customer_status IN ('GOLD', 'PLATINUM')
```

コネクターは、現在のビジネス・オブジェクトの creation_date 属性から日付値 ('01/01/90') を取得します。また、アプリケーション固有情報に ..quantity と指定されているので、親ビジネス・オブジェクトの quantity 属性から数量値 (20) を取得します。

コネクターは、Retrieve 動詞用のアプリケーション固有情報の解析を完了すると、ビジネス・オブジェクトの基本キーまたは外部キーに基づいて構成した RETRIEVE ステートメントの WHERE 文節に、解析によって得られたテキストを追加します。コネクターは、先行する AND を WHERE 文節に追加します。アプリケーション固有情報の値は、有効な SQL 構文である必要があります。RetrieveByContent の場合、ア

プリケーション固有情報は、値が取り込まれたビジネス・オブジェクトの属性に基づいて構成した RETRIEVE ステートメントの WHERE 文節に追加されます。

また、WHERE 文節では、実際の属性に代えて、親ビジネス・オブジェクト内のプレースホルダー属性を参照することもできます。このプレースホルダー属性には、アプリケーション固有情報は含まれません。属性が ASI について以下のいずれかの条件を満たしている場合は、属性をプレースホルダーにすることができます。

1. ASI=null or '' を持つ単純属性
2. ASI=PH=TRUE を持つ単純属性

例: Order ビジネス・オブジェクトに複数カーディナリティーの品目用ビジネス・オブジェクトが含まれています。このうち、特定の品目のみを検索する必要があります。この検索は、Order ビジネス・オブジェクト内のプレースホルダー属性を使用して処理することができます。子オブジェクトはすべて除去されるので、このプレースホルダーは親オブジェクトに含まれていなければなりません。プレースホルダー属性には、統合ブローカーを使用して、コンマ (,) で区切られた特定品目のリストを実行時に格納することができます。

この例では、子にあたる品目ビジネス・オブジェクトに対する Retrieve 動詞の WHERE 文節に、次の情報を追加します。

```
line_item_id in(@,@,@):..placeholder1,..placeholder2,..placeholder3
```

ここで、line_item_id は子ビジネス・オブジェクトの ID であり、placeholder は親のプレースホルダー属性です。placeholder が値 12,13,14 を含む場合は、照会で WHERE 文節から以下のものが選択されます。

```
line_item_id in(12,13,14)
```

ここで、SELECT:..FROM:..WHERE*x* in (1,2,3) は標準のデータベース SQL 構文です。

RetrieveByContent 動詞で、WHERE 文節の長さが 0 の場合、コネクターは RETRIEVE ステートメントの WHERE 文節内のアプリケーション固有情報を使用します。この機能を使用すると、ユーザーは属性値が取り込まれていないビジネス・オブジェクトを送信し、RetrieveByContent に動詞に関するアプリケーション固有情報を指定できます。また、コネクターは動詞に関するアプリケーション固有情報のみに指定された情報に基づいて WHERE 文節を作成できます。

第 4 章 JDBCODA を使用したビジネス・オブジェクト定義の生成

この章では、JDBC 用コネクター用のビジネス・オブジェクトを生成するオブジェクト・ディスカバリー・エージェント (ODA) である、JDBCODA について説明します。コネクターは表ベースまたはビュー・ベースのオブジェクトとともに機能するので、JDBCODA では、データベース表およびビューを使用して、JDBCODA の JDBC データ・ソースに固有のビジネス・オブジェクト要件を発見します。

注: データベース概念および JDBC ドライバーに関し、JDBCODA を構成するのに十分な知識があれば、この ODA の動作を理解するうえで役立ちます。

この章には、以下のセクションが含まれています。

インストールと使用法

このセクションでは、以下について説明します。

JDBCODA のインストール

JDBCODA をインストールするには、IBM WebSphere Business Integration Adapter 用インストーラーを使用します。「*システム・インストール・ガイド (UNIX 版 または Windows 版)*」に記載されている手順に従ってください。インストールの完了後、使用システム上の製品をインストールしたディレクトリーを調べると、次のファイルがインストールされています。

- `ODA¥JDBCODA.jar`
- `ODA¥JDBCODAAgent.txt`
- `ODA¥messages¥JDBCODAAgent_ll_TT.txt` (言語 (*ll*) および国または地域 (*TT*) に固有なメッセージ・ファイル)
- `ODA¥start_JDBCODA.bat` (Windows のみ)
- `ODA/JDBC/start_JDBCODA.sh` (UNIX のみ)

- bin¥CWODAEnv.bat (Windows のみ)
- bin/CWODAEnv.sh (UNIX のみ)

注: 特に指定がない限り、本書ではディレクトリー・パスの記述に円記号 (¥) を使用します。UNIX システムの場合には、円記号 (¥) はスラッシュ (/) に置き換えてください。すべての製品のパス名は、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

JDBCODA を使用する前に

JDBCODA を使用するには、以下を実行する必要があります。

- 適切な JDBC ドライバーをインストールします。

  重要: JDBCODA は、JDBC 2.0 以上をサポートする JDBC ドライバーを使用して、データベースに接続します。

- JDBCODA は、対応するデータベース表およびカラムの名前からビジネス・オブジェクトの名前と属性名を生成し、かつ、ビジネス・オブジェクト名と属性名は ISO Latin-1 で指定されていなければならないため、それぞれのデータベース・コンポーネントに Latin-1 の名前が付けられているかどうかを確認してください。Latin-1 の名前が付けられていない場合には、次のいずれかの方法で名前を付けることができます。
  - Business Object Designer において手動でビジネス・オブジェクト定義を作成する。
  - JDBCODA によって生成された定義を編集して、すべてのビジネス・オブジェクト名および属性名を Latin-1 で指定する。
- シェルまたはバッチ・ファイルを編集用に開いて、表 12 に記載されている値を構成します。

表 12. シェルおよびバッチ・ファイルの構成変数

変数	説明	例
AGENTNAME	ODA の名前。	**UNIX:** AGENTNAME=JDBCODA **Windows:** set AGENTNAME=JDBCODA
AGENT	ODA の JAR ファイルの名前。	**UNIX:** AGENT=$CROSSWORLDS/ODA/JDBC/JDBCODA.jar **Windows:** set AGENT= %CROSSWORLDS%¥ODA¥JDBC¥JDBCODA.jar
DRIVERPATH	JDBC ドライバー・ライブラリーのパス。JDBCODA は、ドライバー・クラスを使用して、特定のデータベースへの接続を確立します。	**UNIX:** DRIVERPATH=$CROSSWORLDS/lib/ ¥ xwutil.jar:$CROSSWORLDS/lib/ ¥ xwbase.jar:$CROSSWORLDS/lib/ ¥ xwsqlserver.jar:$CROSSWORLDS/lib/ ¥ spy/lib/spy.jar **Windows:** set DRIVERPATH=%CROSSWORLDS%¥ / lib¥xwutil.jar;%CROSSWORLDS%¥lib¥ / xwbase.jar;%CROSSWORLDS%¥lib¥ / xwsqlserver.jar;%CROSSWORLDS%¥lib¥ / spy¥lib¥spy.jar

表 12. シェルおよびバッチ・ファイルの構成変数 (続き)

変数	説明	例
DRIVERLIB	JDBC ドライバーによって使用されるネイティブ・ライブラリーのパス。	**UNIX:**DRIVERLIB=$CROSSWORLDS/bin/db2jdbc.dll **Windows:** DRIVERLIB=%CROSSWORLDS%¥bin¥db2jdbc.dll

JDBC ドライバーのインストールと、シェル・ファイルまたはバッチ・ファイルの構成値の設定を完了したら、以下の手順を実行して、ビジネス・オブジェクトを生成します。

1. ODA を起動します。
2. Business Object Designer を起動します。
3. Business Object Designer の 6 つのステップの処理を実行して、ODA を構成し、実行します。

このステップについては、以下のセクションで詳しく説明します。

JDBCODA の起動

JDBCODA を起動するには、ご使用のオペレーティング・システムに応じた始動スクリプトを使用します。

UNIX:

```
start_JDBCODA.sh
```

Windows:

```
start_JDBCODA.bat
```

JDBCODA を構成して実行するには、Business Object Designer を使用します。Business Object Designer は、各スクリプト・ファイルまたはバッチ・ファイルの AGENTNAME 変数に指定された名前に基づいて、各 ODA を探し出します。このコネクターのデフォルト ODA の名前は、JDBCODA です。

複数の JDBCODA インスタンスの実行

ODA のインスタンスを複数実行する場合には、ODA の名前を変更することを推奨します。一意的に命名された追加の JDBCODA インスタンスを作成するには、次のようにします。

- インスタンスごとに、別個のスクリプト・ファイルまたはバッチ・ファイルを作成します。
- 各スクリプト・ファイルまたはバッチ・ファイルの AGENTNAME 変数に、固有の名前を指定します。

複数の ODA インスタンスを複数の異なるマシン上で実行する場合には、それらの名前の先頭にホスト・マシン名を付けることを推奨します。

82 ページの図 6 は、Business Object Designer のウィンドウで、実行する ODA を選択する様子を示したものです。

エラーおよびトレース・メッセージ・ファイルの処理

エラー・メッセージおよびトレース・メッセージ用のファイル (デフォルトは JDBCODAAgent.txt) は ¥ODA¥messages¥ にあります。このディレクトリーは製品ディレクトリーの下にあります。このファイルには、次の命名規則が適用されます。

*AgentName*Agent.txt

ODA スクリプト・ファイルまたはバッチ・ファイルのインスタンスを複数作成し、表現される ODA のそれぞれに固有の名前を与える場合、ODA インスタンスのそれぞれにメッセージ・ファイルを 1 つずつ用意することができます。あるいは、名前の異なる複数の ODA に、同一のメッセージ・ファイルを使用させることもできます。有効なメッセージ・ファイルを指定する方法は、次の 2 つです。

- ODA の名前を変更し、それに対応するメッセージ・ファイルを作成しない場合には、ODA 構成の一部として、Business Object Designer でメッセージ・ファイルの名前を変更する必要があります。Business Object Designer はメッセージ・ファイルに対して、名前を提供しますが、実際にファイルを作成するわけではありません。ODA 構成の際に表示されたファイルが存在しないものである場合は、その値を既存のファイルを指すように変更します。
- 特定の ODA 用の既存のメッセージ・ファイルをコピーして、必要に応じて変更することができます。Business Object Designer では、ユーザーは命名規則に従って各ファイルに名前を付けると想定されています。例えば、AGENTNAME 変数に JDBCODA1 と指定されている場合、このツールでは、関連するメッセージ・ファイルは JDBCODA1Agent.txt であると考えられます。したがって、Business Object Designer が確認のため ODA 構成の一部としてファイル名を提供するとき、このファイル名は ODA 名に基づいています。デフォルトのメッセージ・ファイルが正しく命名されていることを確認し、必要ならば訂正してください。

重要: ODA の構成時にメッセージ・ファイルの名前を正しく指定できなかった場合には、ODA はメッセージなしに稼働します。メッセージ・ファイル名の指定の詳細については、82 ページの『初期化プロパティーの構成』を参照してください。

構成処理では、次のものを指定します。

- JDBCODA がエラー情報とトレース情報を書き込むファイルの名前
- トレースのレベル (0 から 5)

表 13 にこれらの値を示します。

表 13. トレース・レベル

トレース・レベル	説明
0	すべてのエラーを記録します。
1	すべてのメソッド開始/終了メッセージをトレースします。
2	ODA のプロパティーとその値をトレースします。
3	すべてのビジネス・オブジェクトの名前をトレースします。
4	作成されたスレッドすべての詳細をトレースします。
5	• すべてのプロパティーの ODA 初期化値を示します。• JDBCODA が作成した各スレッドの詳細な状況をトレースします。• ビジネス・オブジェクト定義ダンプをトレースします。

これらの値をどこで構成するかについては、82 ページの『初期化プロパティーの構成』を参照してください。

Business Object Designer での JDBCODA の使用

このセクションでは、Business Object Designer で、JDBCODA を使用してビジネス・オブジェクト定義を生成する方法について説明します。Business Object Designer の起動については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

ODA の起動後、Business Object Designer を起動させ、ODA を構成し、実行します。Business Object Designer での、ODA を使用したビジネス・オブジェクト定義の生成は、6 つのステップで構成されています。Business Object Designer には、これらのステップを順次案内するウィザードが用意されています。

ODA の起動後、このウィザードを起動するには、次の手順を実行します。

1. Business Object Designer を開きます。
2. 「ファイル」メニューから、「New Using ODA...」サブメニューを選択します。

   Business Object Designer は、ウィザードの最初のウィンドウ (「エージェントの選択」という名前) を表示します。82 ページの図 6 にこのウィンドウを示します。

ODA を選択、構成、および実行するには、以下のステップを実行してください。

1. 『ODA の選択』
2. 82 ページの『初期化プロパティーの構成』
3. 84 ページの『ノードの展開と表、ビューおよびストアード・プロシージャーの選択』
4. 85 ページの『データベース・オブジェクト選択の確認』
5. 86 ページの『定義の生成』およびオプションで 86 ページの『追加情報の入力』
6. 88 ページの『定義の保管』

ODA の選択

図 6 に、Business Object Designer の 6 段階のウィザードの最初のダイアログ・ボックスを示します。このウィンドウでは、実行する ODA を選択します。

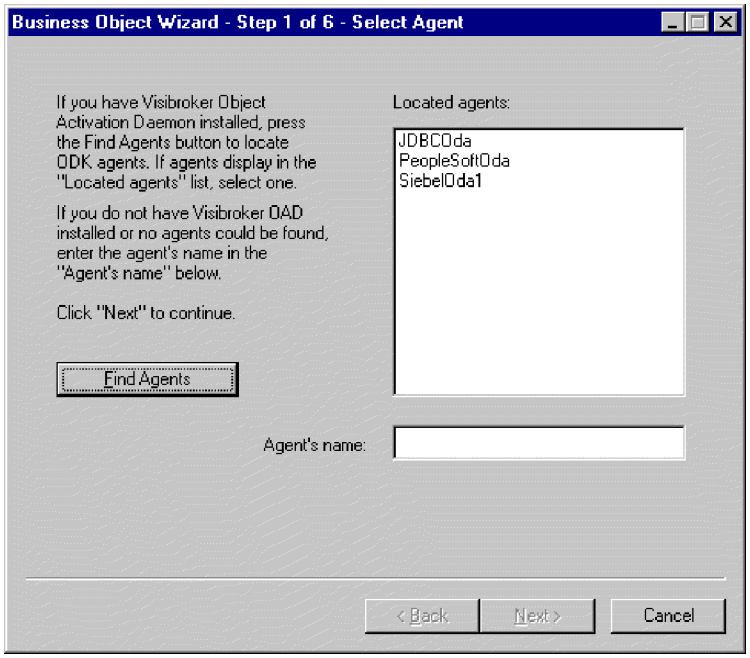


図 6. ODA の選択

ODA を選択するには、次の手順を実行します。

1. 「エージェントの検索」ボタンをクリックすることにより、登録済みまたは現在実行中の ODA のすべてを「検索されたエージェント」フィールドに表示します。

   注: Business Object Designer が目的の ODA を見つけられない場合には、ODA の設定をチェックしてください。

2. 表示リストから、目的の ODA を選択します。

   Business Object Designer の「エージェント名」フィールドに、選択した ODA が表示されます。

初期化プロパティーの構成

Business Object Designer で最初に JDBCODA とやり取りするときに、一連の初期設定プロパティーの入力プロンプトが出されます (図 7 を参照)。これらのプロパティーは、JDBCODA を使用するたびに再入力しなくても済むよう、名前つきプロファイルに保存することができます。ODA プロファイルの指定については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

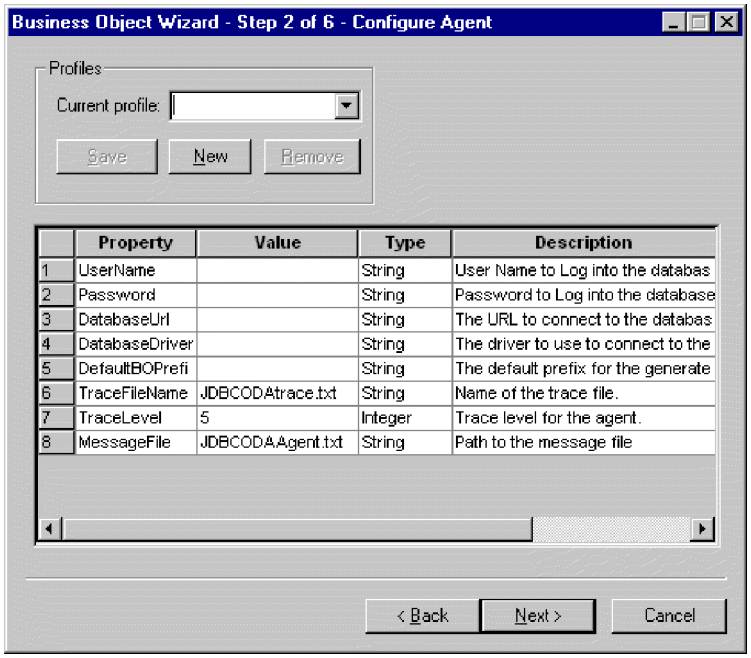


図 7. エージェント初期化プロパティーの構成

JDBCODA プロパティーの構成を表 14 に示します。

表 14. JDBCODA プロパティー

行番号	プロパティー名	プロパティー・タイプ	説明
1	UserName	String	データベースに接続する権限を持つユーザーの名前。
2	Password	String	データベースに接続する権限を持つユーザーのパスワード。
3	DatabaseUrl	String	データベースに接続するための URL。例: jdbc:oracle:thin:@MACHINENAME:1521:SIDNAME
4	DatabaseDriver	String	接続の確立に使用されるドライバーの名前。例: oracle.jdbc.driver.OracleDriver
5	DefaultBOPrefix	String	ビジネス・オブジェクト名を固有の名前にするために、その先頭に付けられるテキスト。これは、後に、Business Object Designer でビジネス・オブジェクトのプロパティーの入力を要求されたときに、必要に応じて変更することができます。詳細については、86 ページの『追加情報の入力』を参照してください。
6	TraceFileName	String	JDBCODA がトレース情報を書き込むファイル。このファイルが存在しない場合、JDBCODA はディレクトリーにそのファイルを作成します。このファイルがすでに存在する場合、JDBCODA はそのファイルに追記します。 JDBCODA は、命名規則に従って、ファイルに名前を付けます。例えば、エージェントの名前が JDBCODA の場合は、JDBCODAtrace.txt という名前のトレース・ファイルが生成されます。このプロパティーを使用して、このファイルとは異なる名前を指定します。
7	TraceLevel	Integer	JDBCODA のトレース・レベル

表 14. JDBCODA プロパティー (続き)

行番号	プロパティー名	プロパティー・タイプ	説明
8	MessageFile	String	エラーおよびメッセージ用のファイルの名前。JDBCODA は、命名規則に従ってファイル名を表示します。例えば、エージェント名が JDBCODA である場合、このメッセージ・ファイル・プロパティーの値は、JDBCODAAgent.txt と表示されます。**重要**: このエラーおよびメッセージ・ファイルは ¥ODA¥messages ディレクトリーになければなりません。既存のファイルを確認または指定する場合に、このプロパティーを使用します。

重要

Business Object Designer に表示されたデフォルトのメッセージ・ファイル名が、存在しないファイルを示している場合は、その名前を修正します。このダイアログ・ボックスから移動したときに、名前が不正であった場合には、Business Object Designer は、ODA の起動元となったウィンドウにエラー・メッセージを表示します。このメッセージは、Business Object Designer 内にポップアップ表示されません。有効なメッセージ・ファイルを指定できない場合、ODA は実行時にメッセージを出力しません。

ノードの展開と表、ビューおよびストアード・プロシージャーの選択

JDBCODA のすべての初期化プロパティーの構成を完了すると、Business Object Designer は指定されたデータベースに接続し、そのデータベース内のスキーマ名をすべて含むツリーを表示します。これらの名前は、ツリーのノードとして表示され、展開することができます。スキーマ名をクリックすると、各スキーマ内にある表、ビュー、およびストアード・プロシージャーがすべて表示されます。図 8 にこのダイアログ・ボックスでいくつかのスキーマを展開した様子を示します。

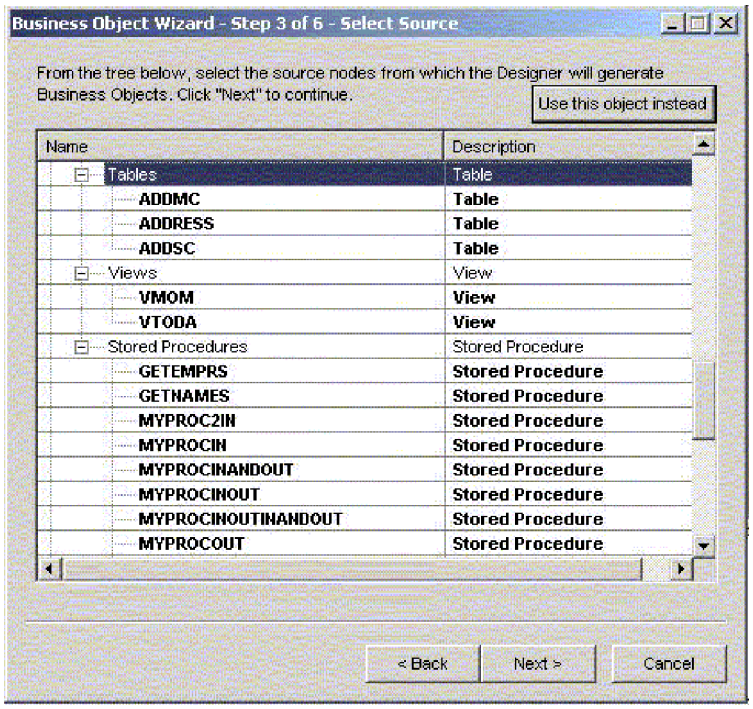


図 8. 一部のノードが展開された状態のスキーマ・ツリー

生成されたビジネス・オブジェクト定義のデータを格納するデータベース・オブジェクトをすべて識別するには、必要な表、ビューおよびをストアード・プロシージャーを選択して、「次へ」をクリックします。戻されるオブジェクトのフィルター方法の詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

データベース・オブジェクト選択の確認

生成されるビジネス・オブジェクト定義に関連付けられるデータベース・オブジェクトの特定をすべて完了すると、Business Object Designer には、選択された表とビューのみを示すダイアログ・ボックスが表示されます。図 9 にこのダイアログ・ボックスを示します。

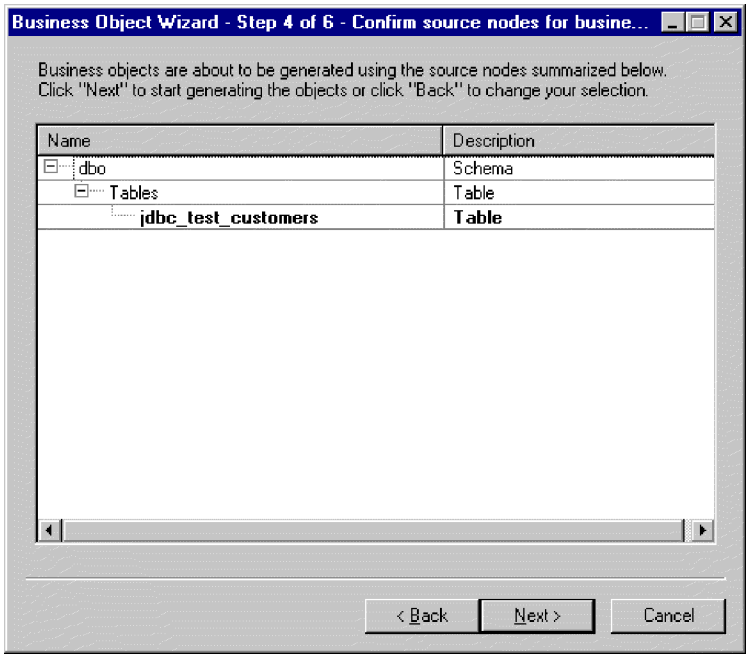


図 9. データベース・オブジェクトの選択の確認

このウィンドウには、次のオプションが用意されています。

- 選択内容を確認するには、「次へ」をクリックします。
- 選択内容が正しくない場合は、「戻る」をクリックして直前のウィンドウに戻り、必要な変更を行います。選択内容が訂正されたら、「次へ」をクリックします。

定義の生成

選択したデータベース・オブジェクトを確認すると、次のダイアログ・ボックスが開き、Business Object Designer によって定義が生成されていることを通知します。

追加情報の入力

JDBCODA に追加情報が必要な場合、Business Object Designer では、「BO プロパティー」ウィンドウを表示し、ユーザーにその情報の入力を要求します。

「BO プロパティー」ウィンドウで、次の情報を入力または変更します。

- *Prefix:* ビジネス・オブジェクト名を固有の名前にするために、その先頭に付けられるテキスト。「エージェントの構成」ウィンドウ (図 7) で *DefaultBOPrefix* プロパティーの値として入力した値を使用しても不都合がない場合は、ここで値を変更する必要はありません。
- *Verbs:*「*値*」フィールドをクリックし、ポップアップ・メニューから 1 つ以上の動詞を選択します。これらは、ビジネス・オブジェクトでサポートされる動詞になります。
- *Add Stored Procedure:*「値」フィールドで「`Yes`」または「`No`」をクリックします。

– 「Yes」を選択して「OK」をクリックすると、ストアード・プロシージャーの属性をすべて示すリストを含むウィンドウが表示されます。ビジネス・オブジェクトに追加するストアード・プロシージャーの属性を選択してください。
– 生成されたビジネス・オブジェクト定義に、ストアード・プロシージャーの属性が追加されないようにするには、「No」を選択します。

デフォルトは Yes です。

注: 「BO プロパティー」ダイアログ・ボックスに複数の値を含むフィールドがある場合、そのフィールドは、このダイアログ・ボックスが最初に開いた時点では、空であるかのように表示されます。フィールド内をクリックすると、含まれる値を示すドロップダウン・リストが表示されます。

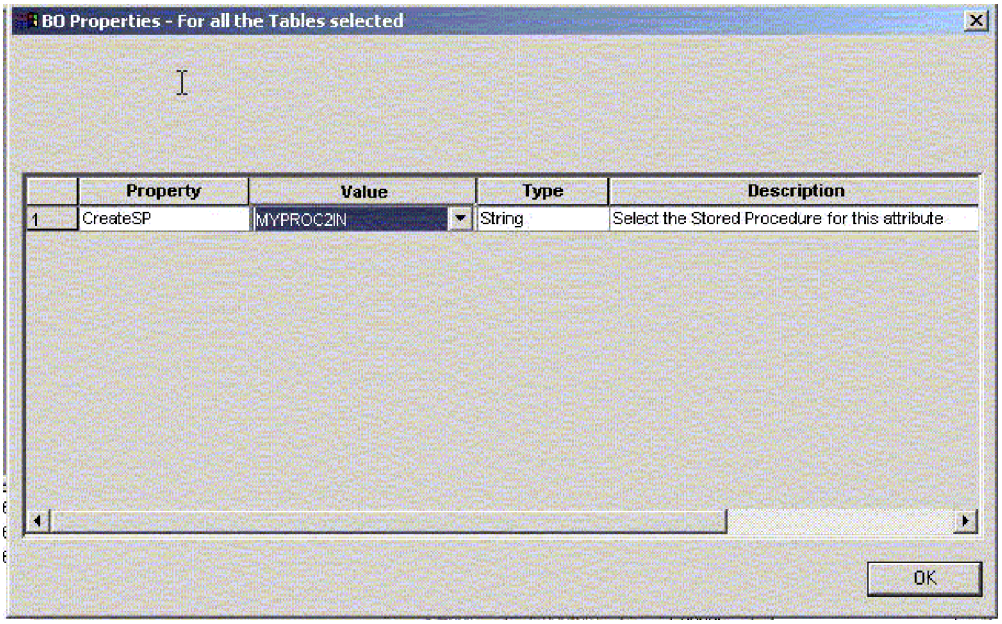


図 10. ストアード・プロシージャーとストアード・プロシージャー属性の関連付け

ビジネス・オブジェクトに追加されるストアード・プロシージャーの属性は、接続先データベースの特定のスキーマに含まれるストアード・プロシージャーのいずれかと関連付けることができます。ストアード・プロシージャーは、ドロップダウン・リストを使用して、ストアード・プロシージャーの属性ごとに選択することができます。ドロップダウン・リストには、接続先データベースの特定のスキーマに含まれるストアード・プロシージャーがすべて表示されます。ここで指定した情報に基づいて、各属性に必要な ASI 情報が生成されます。

オブジェクト・レベルの ASI (アプリケーション固有の情報) は、TN=tableName のようになります。

また、属性レベルでは、ASI は CN=ColumnName のようになります。

ストアード・プロシージャーからビジネス・オブジェクトを生成しようとしているときに、Adapter for JDBC の SPForCreate などのストアード・プロシージャー属性をビジネス・オブジェクトに関連付けると、特定のスキーマに含まれるすべてのストアード・プロシージャーの名前を示すリストがそれらのストアード・プロシージ

ャー属性に対して表示され、必要なストアード・プロシージャーをビジネス・オブジェクトに関連付けることができます。その結果、Adapter for JDBC のストアード・プロシージャー属性の ASI として、次の情報が生成されます。

```
SPN=stored procedure Name; IN=a1:a2; OUT=b1:b2; IO=c1:c2
```

ここで、IN はその後に続くストアード・プロシージャー用のパラメーターが入力タイプであることを意味し、OUT はその後に続くパラメーターが出力タイプであることを意味し、IO はその後に続くパラメーターが入出力タイプであることを意味しています。ODA は、ASI を生成するとき、RS を (true または false に) 設定しません。したがって、この情報は手動で設定する必要があります。

ビジネス・オブジェクトに追加される動詞は標準の動詞です。これは、基本的には Retrieve、RetrieveByContent、Create、Update、および Delete です。

ストアード・プロシージャーの戻りパラメーターのタイプが ResultSet である場合、ODA は、結果セットを分析し、結果セットの列がビジネス・オブジェクトの属性になるようにビジネス・オブジェクトを作成します。ストアード・プロシージャーによって戻される列に対応する ASI には、`CN=StoredProcedureColumnName` が設定されます。ODA では、ドライバーから戻される JDBC メタデータ情報を基にキー属性を設定します。この情報が戻されない場合、ODA は、デフォルトではどの属性もキー属性としてマークしません。その他の属性 (長さや型など) については、いずれも、表から生成された属性に設定される場合と同様に設定されます。

定義の保管

「BO プロパティー」ダイアログ・ボックスに必要な情報をすべて入力して「OK」をクリックすると、Business Object Designer には、ウィザードの最後のウィンドウが表示されます。ここで、定義をサーバーまたはファイルに保管することができます。あるいは、Business Object Designer 内で定義を開き、編集することができます。詳細な変更の方法などの詳細情報については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

生成される定義の内容

JDBCODA によって生成されたビジネス・オブジェクト定義には、次のものが含まれます。

- 指定されたデータベース表およびビューの列に対応する属性 (1 列につき 1 属性)
- 「BO プロパティー」ウィンドウで指定された動詞
- アプリケーション固有情報
  - ビジネス・オブジェクト・レベルの情報
  - 属性ごとの情報
  - 動詞ごとの情報

このセクションで説明する内容は次のとおりです。

- 89 ページの『ビジネス・オブジェクト・レベルのプロパティー』
- 89 ページの『属性プロパティー』
- 91 ページの『動詞』

ビジネス・オブジェクト・レベルのプロパティー

JDBCODA は、ビジネス・オブジェクト・レベルでは、次の情報を生成します。

- ビジネス・オブジェクト名
- バージョン: デフォルトで 1.0.0
- 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により、次のことが可能になります。

- 対応するデータベース表の名前の指定
- 物理削除または論理削除の実行に必要な情報の提供

ビジネス・オブジェクト・レベルでは、アプリケーション固有情報の形式は、セミコロン (;) によって区切られた複数のパラメーターで構成されています。パラメーター名とその値は、コロン (:) で区切られます。次の構文、

TN=*TableName*; SCN=*StatusColumnName*:*StatusValue*

では、TableName はデータベース表を示します。また、*StatusColumnName* は論理削除の実行に使用されるデータベース列の名前であり、*StatusValue* はビジネス・オブジェクトが非アクティブまたは削除済みであることを示す値です。

JDBCODA がこのレベルで生成する AppSpecificInfo には、データベース表名またはビュー名に専用の値も含まれています。状況列の値の指定については、64 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。

属性プロパティー

このセクションでは、JDBCODA が属性ごとに生成するプロパティーについて説明します。属性の詳細については、60 ページの『ビジネス・オブジェクトの属性プロパティー』を参照してください。

Name プロパティー

JDBCODA は、データベース表またはビュー内の列名に基づいて、属性名の値を取得します。

Data Type プロパティー

属性の型の設定時に、JDBCODA は表またはビューの列のデータ型を、対応する IBM WebSphere Business Integration Adapter ビジネス・オブジェクト・タイプに変換します。この変換は 2 つのステップで実行されます。まず、データベース内のデータ型が JDBC 型に変換されます。次に、JDBC 型が IBM WebSphere Business Integration Adapter ビジネス・オブジェクト・タイプに変換されます。最初の変換はご使用の JDBC ドライバーによって実行されます。JDBC 型への個々のデータベース・タイプのマッピングの詳細については、JDBC 仕様 (2.0 以上) を参照してください。表 14 は、JDBC 型から対応する IBM WebSphere Business Integration Adapter ビジネス・オブジェクト・タイプへの変換を示しています。

表 15. データ型の対応関係

JDBC 型	WebSphere Business Integration Adapter ビジネス・オブジェクト・タイプ
BIT	BOOLEAN
CHAR	STRING
VARCHAR	STRING
LONGVARCHAR	STRING
INTEGER	INTEGER
NUMERIC	INTEGER
SMALLINT	INTEGER
TINYINT	INTEGER
BIGINT	INTEGER
DATE	DATE
TIME	DATE
TIMESTAMP	DATE
DECIMAL	STRING
DOUBLE	DOUBLE
FLOAT	DOUBLE
REAL	FLOAT
BINARY	STRING, BYTEARRAY=TRUE を以下に追加 AppSpecificInfo
VARBINARY	STRING, BYTEARRAY=TRUE を以下に追加 AppSpecificInfo

注: 列のデータ型が、表 15 に含まれるデータ型以外のものである場合、JDBCODA はその列をスキップし、その列を処理できないというメッセージを表示します。

Cardinality プロパティー

JDBCODA は、どの単純属性についても、カーディナリティーを 1 に設定します。

MaxLength プロパティー

JDBCODA は、varchar、char、または text データ型で指定された長さからストリングの長さを取得します。

IsKey プロパティー

列が表の基本キーである場合、JDBCODA はその列をキー属性としてマークします。ただし、Business Objects を生成するソース・ノードとして表ではなくビューが選択されている場合は、JDBCODA はその列をキー属性としてマークしません。この場合、キー属性を手動で設定する必要があります。

IsForeignKey プロパティー

JDBCODA は IsForeignKey プロパティーを設定しません。このプロパティーは、Business Object Designer で設定することができます。

IsRequired プロパティー

表またはビューに非 null に指定されているフィールドがある場合、JDBCODA は、そのフィールドを必須属性としてマークします。ただし、JDBCODA はキー・フィールドを必須属性としてマークしません。これは、そのフィールドに関連付けられているシーケンスが存在する場合や、そのフィールドが ID 列である場合があるためです。

AppSpecificInfo プロパティー

JDBCODA によって属性レベルで組み込まれる AppSpecificInfo プロパティーのパラメーターは、2 つあります。指定パラメーターの構文は、次のとおりです。

CN=*ColumnName*

では、ColumnName は、この構文が使用されている属性に関連付けられている、データベース表またはビューの列名です。

BYTEARRAY=*true*|*false*

JDBCODA はバイナリー・データを含む列を認識し、AppSpecificInfo プロパティーが BYTEARRAY=true の String 型の属性を作成します。

注: 追加の AppSpecificInfo パラメーターは、Business Object Designer で設定することができます。これらのパラメーターに関する詳細については、65 ページの『属性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。

動詞

JDBCODA は、「BO プロパティー」ウィンドウで指定された動詞を生成します。各動詞の AppSpecificInfo プロパティーを作成しますが、設定は行いません。詳細については、73 ページの『動詞のアプリケーション固有情報形式』を参照してください。

ビジネス・オブジェクト定義ファイルのサンプル

ビジネス・オブジェクト定義ファイルのサンプルを、以下に示します。

```
[BusinessObjectDefinition]
Name = CUSTOMER
Version = 1.0.0
AppSpecificInfo = TN=ra_customers;SCN=

            [Attribute]
               Name = customer_id
               Type = Integer
               Cardinality = 1
```

```
                MaxLength = 0
                IsKey = true
                IsForeignKey = false
                IsRequired = false
                AppSpecificInfo = CN=customer_id
                DefaultValue =
                [End]

                *******Other attributes *********

[Attribute]
                Name = ObjectEventId
                Type = String
                Cardinality = 1
                MaxLength = 0
                IsKey = false
                IsForeignKey = false
                IsRequired = false
                AppSpecificInfo =
                DefaultValue =
                [End]

                [Verb]
                Name = Delete
                AppSpecificInfo =
                [End]

                [Verb]
                Name = Update
                AppSpecificInfo =
                [End]

                [Verb]
                Name = Create
                AppSpecificInfo =
                [End]

                [Verb]
                Name = Retrieve
                AppSpecificInfo =
                [End]
[End]
```

子ビジネス・オブジェクトを含む属性の挿入

単一カーディナリティーまたは複数カーディナリティーの子ビジネス・オブジェクトを表現する属性を挿入するには、Business Object Designer を使用します。詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

ビジネス・オブジェクト定義への情報の追加

ビジネス・オブジェクト定義に必要な情報の一部が、データベース表およびビューに含まれていない場合があるため、JDBCODA によって作成されたビジネス・オブジェクト定義に情報を追加しなければならないことがあります。詳細については、37 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』を参照してください。

ビジネス・オブジェクト定義の確認、または情報の追加には、Business Object Designer またはテキスト・エディターを使用できます。更新した定義を IBM

WebSphere Business Integration Adapter リポジトリーに再ロードするには、Business Object Designer を使用するか、ICS が統合ブローカーの場合は `repos_copy` コマンドを使用します。

第 5 章 トラブルシューティングとエラー処理

この章では、JDBC 用コネクターの始動時および実行時に発生する可能性がある問題について説明します。この章の内容は、次のとおりです。

始動時の問題

コネクターの始動時に問題が発生した場合は、統合ブローカーが稼働中であることを確認してください。

イベント処理

イベント表にイベントが存在するにもかかわらず、コネクターの実行中にそれらが処理されない場合は、次のことを確認してください。

- 関係するビジネス・プロセスが実行中であること。
- イベント表内のビジネス・オブジェクト名が、ビジネス・プロセスのポートに指定されているビジネス・オブジェクト名と一致していること。

マッピング (ICS 統合ブローカーのみ)

このセクションでは、以下について説明します。

マッピングの問題

ビジネス・オブジェクトがマップされていない場合、またはマッピングが行われない場合には、マップが正しいディレクトリーにインストールされていることを確認します。

日付型変換

注: この日付変換手順は、バージョン 1.5.0 よりも前のバージョンの コネクターのみに適用できます。

データベースに Date 形式で格納されているデータを WebSphere Business Integration Adapter ビジネス・オブジェクトで使用する String 形式に変換するには、マップを使用します。

例えば、Oracle データベースに格納されている次のような日付

```
Sun Jan 01 00:00:00 CEST 1999
```

が存在するとします。この日付を、次のようなストリングに変換してから、WebSphere Business Integration Adapter for JDBC ビジネス・オブジェクト内で処理する必要があるとします。

```
Jan 01 1999 00:00:00
```

この変換を行うには、マッピングでデータ変換用に定義されている DtpDate() コンストラクターおよび DtpSplitString() コンストラクターを使用します。これらのコンストラクターの構文と説明、およびこれらのコンストラクターによってオブジェクトが作成されるクラスについては、「*マップ開発ガイド*」を参照してください。

マップを使用して Date 値を String に変換するには、以下のステップを行います。

1. DtpSplitString() を使用して、スペースを区切り文字としてストリングを 6 つの部分に分け、DtpDate で使用できる順序に並べ替えます。上の例の日付の場合、次のように使用します。

   ```
   DtpSplitString OurSplitString = new DtpSplitString
      ("Sun Jan 01 00:00:00 CEST 1999"," ");
   ```

   このステートメントでは、OurSplitString は DtpSplitString 型のユーザー定義変数です。また、スペースが区切り文字に指定されています。

2. DtpSplitString クラスの nextElement() メソッドを使用して、新規に作成した OurSplitString 変数の中をループし、その変数に 6 個ある要素それぞれを要素の型が String の配列に格納します。以下の例では、出力配列として OurStringPieces を指定しています。

   ```
   String[] OurStringPieces = new String[6];
   for (i=0;i<=5;i=i+1){
      OurStringPieces[i]=OurSplitString.nextElement();
   }
   ```

   このループ処理により、次の配列要素が生成されます。

   ```
   OurStringPieces[0] = Sun
   OurStringPieces[1] = Jan
   OurStringPieces[2] = 01
   OurStringPieces[3] = 00:00:00
   OurStringPieces[4] = CEST
   OurStringPieces[5] = 1999
   ```

3. これらの断片的なストリングのうち、DtpDate に入力する必要があるものを連結します。ここでの変換例では、DtpDate への入力のフォーマットとして「M D Y

h:m:s」を使用します。これにより、変換後のストリングは、「Jan 01 1999 00:00:00」と表示されます。下記の例では、String は OurStringPieces 配列の要素 1、2、5、および 3 を使用します。

```
OurConcatenatedString =
OurStringPieces[1]+OurStringPieces[2]+OurStringPieces[5]+OurStringPieces[3];
```

4. 連結された新しいストリングを、DtpDate への入力として使用します。

```
DtpDate OurDtpDate = new DtpDate(OurConcatenatedString,"M D Y h:m:s");
```

Date 値を DtpDate 形式に変換した後は、マップを使用してその日付を操作することができます。

エラー処理とロギング

コネクターは、現在実行中のビジネス・オブジェクトおよび動詞の処理に失敗する条件が発生した場合には、必ずエラー・メッセージを記録します。また、そのようなエラーが発生した場合、コネクターは、処理に失敗したビジネス・オブジェクトが、受信時点でどのような状態であったかを示すテキスト表現も出力します。出力されたテキストは、コネクターの構成に応じて、コネクターのログ・ファイルまたは標準出力ストリームに書き込まれます。このテキストは、エラーの原因を判別するための補助情報として使用できます。

エラー・タイプ

表 16 では、コネクターが各トレース・レベルで出力するトレース・メッセージのタイプについて説明します。これらのメッセージは、Java コネクター実行ラッパーおよび WebSphere MQ メッセージ・インターフェースなどの IBM WebSphere Business Integration Adapter アーキテクチャーによるトレース・メッセージ出力に追加されます。

表 16. コネクター・トレース・メッセージ

トレース・レベル	トレース・メッセージ
レベル 0	コネクターのバージョンを示すメッセージ。このレベルでは、その他のトレースは行われません。これはデフォルト値です。
レベル 1	• 状況メッセージ。 • 処理されたビジネス・オブジェクトごとの識別 (キー) 情報を示すメッセージ。 • pollForEvents メソッドが実行されるたびにデリバリーされるメッセージ。
レベル 2	• コネクターがビジネス・オブジェクトの処理中に検出または検索した情報 (配列や子ビジネス・オブジェクトなど) を含む、ビジネス・オブジェクト・ハンドラー・メッセージ。 • ビジネス・オブジェクトが gotApplEvent() または executeCollaboration() のいずれかから統合ブローカーに通知されるたびにログに記録されるメッセージ。 • ビジネス・オブジェクトが統合ブローカーからの要求として受信されたことを示すメッセージ。

表 16. コネクター・トレース・メッセージ (続き)

トレース・レベル	トレース・メッセージ
レベル 3	• コネクターがビジネス・オブジェクトの外部キーをいつ検出または設定したかなどの情報を含む、外部キー処理メッセージ。 • ビジネス・オブジェクトの処理についての情報を示すメッセージ。例えば、このメッセージは、コネクターがビジネス・オブジェクト間で一致を検出した場合や、子ビジネス・オブジェクトの配列の中にビジネス・オブジェクトを検出した場合にデリバリーされます。
レベル 4	• アプリケーション固有情報に関するメッセージ (ビジネス・オブジェクトのアプリケーション固有情報フィールドを解析する関数によって戻された値を示すメッセージなど)。 • コネクターによる関数の開始または終了のタイミングを示すメッセージ。このメッセージは、コネクターの処理フローのトレースに役立ちます。 • スレッドに固有のメッセージのすべて。コネクターによって複数のスレッドが作成される場合は、新しいスレッドが作成されるたびにメッセージが出力されます。
レベル 5	• コネクターの初期化を示すメッセージ (統合ブローカーから検索された、各構成プロパティーの値を示すメッセージなど)。 • アプリケーションで実行されたステートメントを含むメッセージ。このトレース・レベルでは、宛先アプリケーションで実行されたステートメントのすべてと、置換された変数の値がコネクターのログ・ファイルに含まれます。 • コネクターでビジネス・オブジェクトの処理が開始される以前のビジネス・オブジェクトの表現 (コネクターがビジネス・オブジェクトを受信したときのビジネス・オブジェクトの状態を示すもの)、および、ビジネス・オブジェクトの処理を終了した後のビジネス・オブジェクトの表現 (コネクターからビジネス・オブジェクトを戻したときのビジネス・オブジェクトの状態を示すもの) を含むメッセージ。 • ビジネス・オブジェクト・ダンプを含むメッセージ。 • コネクターが実行時に作成したスレッドのそれぞれの状況を示すメッセージ。

エラー・メッセージ

コネクター・メッセージ・ファイル

コネクターが生成するすべてのエラー・メッセージは、JDBCConnector.txt または JDBCConnector_*ll*_*TT*.txt というメッセージ・ファイルに保管されます (*ll* の部分には言語、*TT* の部分には国または地域を示す文字が入ります)。それぞれのエラー・メッセージの前にはエラー番号が付けられています。以下に例を示します。

```
20017
Connector Infrastructure version does not match.

20018
Connection from {1} to the Application is lost! Please enter 'q'
to stop the connector, then restart it after the problem is fixed.

20019
Error: ev_id is NULL in pollForEvent().
```

アプリケーションへの接続不可

コネクターは、接続の確立に失敗した場合、統合ブローカーに FAIL を送信して終了します。

AutoCommit を false に設定している場合に PingQuery が失敗すると、コネクターはデータベースへの新規の接続を作成しようとします。データベースへの新規接続の作成に成功した場合、コネクターは処理を続行します。失敗した場合、コネクターは APPRESPONSETIMEOUT を戻します。この結果、コネクターは終了します。

fetch out of sequence エラー

Oracle データベースのバージョン 8.0 および 8.1 を Sun Solaris または Oracle 8.1 とともに Windows 2000 で使用する場合は、AutoCommit プロパティーを false に設定する必要があります。false に設定しないと、「ORA-01002 (フェッチ順序が無効です (fetch out of sequence))」というエラー・メッセージが表示されます。Oracle データベースの以前のバージョンでは、このエラーは発生しません。AutoCommit を false に設定すると、パフォーマンスが向上します。

DB2 を使用する際のイベントまたはアーカイブ表の位置指定の不可

始動時に、コネクターは、SchemaName 構成プロパティーによって指定されたデータベース内のイベントおよびアーカイブ表を見つけ出そうとします。ユーザーがデータベースとして DB2 を使用している場合、コネクターは、イベント表およびアーカイブ表の検索に失敗して次のようなエラー・メッセージを戻すことがあります。

Event/Archive table *table_name* does not exist in the database.

この問題を回避するには、コネクターの SchemaName 構成プロパティーで指定する DB2 のスキーマ名を常に大文字 (例えば、SUSER というような名前) にしてください。

DB2 データベースと連動するコネクターの使用可能化

DB2 データベースでコネクターを使用するには、以下のステップを実行する必要があります。

1. db2java.zip というファイルを、DB2 ホストから、コネクターを実行するマシンの $*ProductDir*¥lib ディレクトリーへコピーする。
2. db2jdbc.dll というファイルを、DB2 ホストから、コネクターを実行するマシンの $*ProductDir*¥bin ディレクトリーへコピーする。
3. ご使用のオペレーティング・システムに応じて、コネクターのスタートアップ・ファイル (start_JDBC.sh or start_JDBC.bat) で以下を変更する。

   UNIX: JDBCDRIVERPATH=$*ProductDir*/lib/db2java.zip

   Windows: set JDBCDRIVERPATH=%*ProductDir*%¥lib¥db2java.zip
4. DB2 ホスト・マシンで、DB2/bin/db2jstrt プロセスを始動する。使用するポート番号 (例えば、DB2/bin/db2jstrt 50000) を指定します。

5. コネクターの JDBCDriverClass プロパティーの値を COM.ibm.db2.jdbc.net.DB2Driver (または、DB2 データベースがコネクターを実行するマシンと同じマシン上にある場合は COM.ibm.db2.jdbc.app.DB2Driver) に設定する。
6. コネクターの DatabaseURL プロパティーの値を jdbc:db2://*MachineName*:*PortNumber*/*DBname* (または、DB2 データベースがコネクターを実行するマシンと同じマシン上にある場合は *jdbc:db2:DBname*) に設定する。

resource busy エラー

注: コネクターでこのエラーが発生するのは、コネクターが Oracle データベース上で実行中の場合に限られます。

コネクターがアプリケーション内のデータを検索または変更しているときに、以下のようなエラーが発生することがあります。

```
[Time: 2001/05/29 16:30:07.356] [System: ConnectorAgent] [SS: SOVTConnector]
[Type: Trace] [Mesg: Select CLIENT,COUNTRY,STRT_CODE,CITY_CODE,CITYP_CODE,
STRTYPEAB,COMMU_CODE,REGIOGROUP,TAXJURCODE from ADRSTREET where CLIENT='100'
and COUNTRY='DE' and STRT_CODE='000001114136' FOR UPDATE NOWAIT]
[Time: 2001/05/29 16:30:07.526] [System: ConnectorAgent] [SS: SOVTConnector]
[Type: Trace ] [Mesg: :logMsg]
[Time: 2001/05/29 16:30:07.536] [System: ConnectorAgent] [SS: SOVTConnector]
 [Type: Error ] [MsgID: 37002]
[Mesg: Execution of Retrieve statement failed : java.
sql.SQLException: ORA-00054: Versuch, mit NOWAIT eine bereits
belegte Ressourceanzufordern.]
```

このエラーは、コネクターが現在ロックされているレコードを更新しようとしたときに発生します。レコードは他のプロセスによってロックされているか、またはコネクターがマルチスレッドである場合にはコネクター自体によってロックされている可能性があります。

更新処理の際にレコードはロックされていなければなりません。コネクターは、統合ブローカーの受信したオブジェクトの変更後イメージを探し出そうとし、その際にデータの保全性を保持するためにデータベース内のオブジェクト全体をロックします。

この問題を解決するには、コネクターによるレコードのロックを妨げているプロセスを停止するか、あるいは、コネクターの RetryCountInterval 構成プロパティーを調整します。

JDBC ドライバーがサポートされていないため、JDBCODA が正常に動作しません

JDBC ドライバーが JDBCODA の機能をサポートしない場合、オブジェクト・ディスカバリー・エージェントは正常に機能しません。例えば、ドライバーが JDBCODA によって使用されるすべてのメソッド呼び出しをサポートしない場合、JDBCODA ログに失敗したプロセスが示されます。例えば、次のようにログに記録されます。

```
[Time: 2002/05/15 17:00:55.147] [System: Object Discovery Agent] [SS: null]
[Type: 6] [Mesg: A SQL Error occurred in getting Schema Names from Database.
Reason [ProductName][ODBC ProductName Driver]Optional feature not
implemented]
```

この場合、別の JDBC ドライバーを使用する必要があります。

付録 A. コネクターの標準構成プロパティー

この付録では、WebSphere Business Integration Adapter のコネクター・コンポーネントの標準構成プロパティーについて説明します。この付録の内容は、以下の統合ブローカーで実行されるコネクターを対象としています。

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker (WebSphere Message Brokers (WMQI) と総称)
- WebSphere Application Server (WAS)

コネクターによっては、一部の標準プロパティーが使用されないことがあります。Connector Configurator から統合ブローカーを選択するときには、そのブローカーで実行されるアダプターについて構成する必要のある標準プロパティーのリストが表示されます。

コネクター固有のプロパティーの詳細については、該当するアダプターのユーザーズ・ガイドを参照してください。

注: 本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムを使用している場合は、円記号をスラッシュ (/) に置き換えてください。また、各オペレーティング・システムの規則に従ってください。

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有の構成プロパティー

このセクションでは、標準構成プロパティーについて説明します。コネクター固有の構成プロパティーについては、該当するアダプターのユーザーズ・ガイドを参照してください。

Connector Configurator の使用

Connector Configurator からコネクター・プロパティーを構成します。Connector Configurator には、System Manager からアクセスします。Connector Configurator の使用法の詳細については、付録の『Connector Configurator』を参照してください。

注: Connector Configurator と System Manager は、Windows システム上でのみ動作します。コネクターを UNIX システム上で稼働している場合でも、これらのツールがインストールされた Windows マシンが必要です。UNIX 上で動作するコネクターのコネクター・プロパティーを設定する場合は、Windows マシン上で System Manager を起動し、UNIX の統合ブローカーに接続してから、コネクター用の Connector Configurator を開く必要があります。

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

コネクターは、以下の順序に従ってプロパティーの値を決定します (最も番号の大きい項目が他の項目よりも優先されます)。

1. デフォルト
2. リポジトリー (WebSphere InterChange Server が統合ブローカーである場合のみ)
3. ローカル構成ファイル
4. コマンド行

コネクターは、始動時に構成値を取得します。実行時セッション中に 1 つ以上のコネクター・プロパティーの値を変更する場合は、プロパティーの**更新メソッド**によって、変更を有効にする方法が決定されます。標準コネクター・プロパティーには、以下の 4 種類の更新メソッドがあります。

- **動的**
  変更を System Manager に保管すると、変更が即時に有効になります。コネクターが System Manager から独立してスタンドアロン・モードで稼働している場合 (例えば、いずれかの WebSphere Message Brokers と連携している場合) は、構成ファイルでのみプロパティーを変更できます。この場合、動的更新は実行できません。
- コンポーネント再始動
  System Manager でコネクターを停止してから再始動しなければ、変更が有効になりません。アプリケーション固有コンポーネントまたは統合ブローカーを停止、再始動する必要はありません。
- サーバー再始動
  アプリケーション固有のコンポーネントおよび統合ブローカーを停止して再始動しなければ、変更が有効になりません。
- エージェント再始動 (ICS のみ)
  アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

特定のプロパティーの更新方法を確認するには、「Connector Configurator」ウィンドウ内の**「更新メソッド」**列を参照するか、次に示すプロパティーの要約の表の「更新メソッド」列を参照してください。

標準プロパティーの要約

表 17 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

コネクターを実行する前に、これらのプロパティーの一部の値を設定する必要があります。各プロパティーの詳細については、次のセクションを参照してください。

表 17. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	*CONNECTORNAME* /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	*CONNECTORNAME*/ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は MQ および IDL: Repository Directory は <REMOTE>
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクター・アプリケーション名として指定された値	コンポーネント再始動	
BrokerType	ICS、WMQI、WAS			
CharacterEncoding	ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437 **注:** これは、サポートされる値の一部です。	ascii7	コンポーネント再始動	
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE>
ContainerManagedEvents	値なしまたは JMS	値なし	コンポーネント再始動	Delivery Transport は JMS
ControllerStoreAndForwardMode	true または false	True	動的	Repository Directory は <REMOTE>
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ

表 17. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ、Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonicの場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない

表 17. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は True でなければならない
OADAutoRestartAgent	True または False	False	動的	Repository Directory は <REMOTE> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	HH:MM (HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合: C:¥crossworlds¥repository に設定する

表 17. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*/REQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/RESPONSEQUEUE	コンポーネント再始動	Delivery Transport が JMS の場合: Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分) : 1 から 2147483547	1	動的	
RHF2MessageDomain	mrm、xml	mrm	コンポーネント再始動	DeliveryTransport Transport が JMS であり、かつ WireFormat が CwXML である。
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

統合ブローカーからコネクターへ管理メッセージが送信されるときに使用されるキューです。

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

コネクターから統合ブローカーへ管理メッセージが送信されるときに使用されるキューです。

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

AgentConnections プロパティーは、orb.init[] により開かれる ORB 接続の数を制御します。

デフォルトでは、このプロパティーの値は 1 に設定されます。このデフォルト値を変更する必要はありません。

AgentTraceLevel

アプリケーション固有のコンポーネントのトレース・メッセージのレベルです。デフォルトは 0 です。コネクターは、設定されたトレース・レベル以下の該当するトレース・メッセージをすべてデリバリーします。

ApplicationName

コネクターのアプリケーションを一意的に特定する名前です。この名前は、システム管理者が WebSphere Business Integration システム環境をモニターするために使用されます。コネクターを実行する前に、このプロパティーに値を指定する必要があります。

BrokerType

使用する統合ブローカー・タイプを指定します。オプションは ICS、WebSphere Message Brokers (WMQI、WMQIB または WBIMB) または WAS です。

CharacterEncoding

文字 (アルファベットの文字、数値表現、句読記号など) から数値へのマッピングに使用する文字コード・セットを指定します。

注: Java ベースのコネクターでは、このプロパティーは使用しません。C++ ベースのコネクターでは、現在、このプロパティーに ascii7 という値が使用されています。

デフォルトでは、ドロップ・リストには、サポートされる文字エンコードの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、

製品ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、Connector Configurator に関する付録を参照してください。

ConcurrentEventTriggeredFlows

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

コネクターがイベントのデリバリー時に並行処理できるビジネス・オブジェクトの数を決定します。この属性の値を、並行してマップおよびデリバリーできるビジネス・オブジェクトの数に設定します。例えば、この属性の値を 5 に設定すると、5 個のビジネス・オブジェクトが並行して処理されます。デフォルト値は 1 です。

このプロパティーを 1 よりも大きい値に設定すると、ソース・アプリケーションのコネクターが、複数のイベント・ビジネス・オブジェクトを同時にマップして、複数のコラボレーション・インスタンスにそれらのビジネス・オブジェクトを同時にデリバリーすることができます。これにより、統合ブローカーへのビジネス・オブジェクトのデリバリーにかかる時間、特にビジネス・オブジェクトが複雑なマップを使用している場合のデリバリー時間が短縮されます。ビジネス・オブジェクトのコラボレーションに到達する速度を増大させると、システム全体のパフォーマンスを向上させることができます。

ソース・アプリケーションから宛先アプリケーションまでのフロー全体に並行処理を実装するには、次のようにする必要があります。

- Maximum number of concurrent events プロパティーの値を増加して、コラボレーションが複数のスレッドを使用できるように構成します。
- 宛先アプリケーションのアプリケーション固有コンポーネントが複数の要求を並行して実行できることを確認します。つまり、このコンポーネントがマルチスレッド化されているか、またはコネクター・エージェント並列処理を使用でき、複数プロセスに対応するよう構成されている必要があります。Parallel Process Degree 構成プロパティーに、1 より大きい値を設定します。

ConcurrentEventTriggeredFlows プロパティーは、順次に実行される単一スレッド処理であるコネクターのポーリングでは無効です。

ContainerManagedEvents

このプロパティーにより、JMS イベント・ストアを使用する JMS 対応コネクターが、保証付きイベント・デリバリーを提供できるようになります。保証付きイベント・デリバリーでは、イベントはソース・キューから除去され、単一 JMS トランザクションとして宛先キューに配置されます。

デフォルト値は No value です。

ContainerManagedEvents を JMS に設定した場合には、保証付きイベント・デリバリーを使用できるように次のプロパティーも構成する必要があります。

- PollQuantity = 1 から 500
- SourceQueue = CONNECTORNAME/SOURCEQUEUE

また、MimeType、DHClass、および DataHandlerConfigMOName (オプショナル) プロパティーを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティーの値を設定するには、Connector Configurator の**「データ・ハンドラー」**タブを使用します。「データ・ハンドラー」タブの値のフィールドは、ContainerManagedEvents を JMS に設定した場合にのみ表示されます。

注: ContainerManagedEvents を JMS に設定した場合、コネクターはその pollForEvents() メソッドを*呼び出さ*なくなる ため、そのメソッドの機能は使用できなくなります。

このプロパティーは、DeliveryTransport プロパティーが値 JMS に設定されている場合にのみ表示されます。

ControllerStoreAndForwardMode

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

宛先側のアプリケーション固有のコンポーネントが使用不可であることをコネクター・コントローラーが検出した場合に、コネクター・コントローラーが実行する動作を設定します。

このプロパティーを true に設定した場合、イベントが ICS に到達したときに宛先側のアプリケーション固有のコンポーネントが使用不可であれば、コネクター・コントローラーはそのアプリケーション固有のコンポーネントへの要求をブロックします。アプリケーション固有のコンポーネントが作動可能になると、コネクター・コントローラーはアプリケーション固有のコンポーネントにその要求を転送します。

ただし、コネクター・コントローラーが宛先側のアプリケーション固有のコンポーネントにサービス呼び出し要求を**転送した後で**このコンポーネントが使用不可になった場合、コネクター・コントローラーはその要求を失敗させます。

このプロパティーを false に設定した場合、コネクター・コントローラーは、宛先側のアプリケーション固有のコンポーネントが使用不可であることを検出すると、ただちにすべてのサービス呼び出し要求を失敗させます。

デフォルト値は true です。

ControllerTraceLevel

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

コネクター・コントローラーのトレース・メッセージのレベルです。デフォルトは 0 です。

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

コネクターから統合ブローカーへビジネス・オブジェクトが送信されるときに使用されるキューです。

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

イベントのデリバリーのためのトランスポート機構を指定します。指定可能な値は、WebSphere MQ の MQ、CORBA IIOP の IDL、Java Messaging Service の JMS です。

- ICS がブローカー・タイプの場合は、DeliveryTransport プロパティーの指定可能な値は MQ、IDL、または JMS であり、デフォルトは IDL になります。
- RepositoryDirectory がローカル・ディレクトリーの場合は、指定可能な値は JMS のみです。

DeliveryTransport プロパティーに指定されている値が、MQ または IDL である場合、コネクターは、CORBA IIOP を使用してサービス呼び出し要求と管理メッセージを送信します。

WebSphere MQ および IDL

イベントのデリバリー・トランスポートには、IDL ではなく WebSphere MQ を使用してください (1 種類の製品だけを使用する必要がある場合を除きます)。
WebSphere MQ が IDL よりも優れている点は以下のとおりです。

- 非同期 (ASYNC) 通信:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネントは、サーバーが利用不能である場合でも、イベントをポーリングして永続的に格納することができます。
- サーバー・サイド・パフォーマンス:
  WebSphere MQ を使用すると、サーバー・サイドのパフォーマンスが向上します。最適化モードでは、WebSphere MQ はイベントへのポインターのみをリポジトリー・データベースに格納するので、実際のイベントは WebSphere MQ キュー内に残ります。これにより、サイズが大きい可能性のあるイベントをリポジトリー・データベースに書き込む必要がありません。
- エージェント・サイド・パフォーマンス:
  WebSphere MQ を使用すると、アプリケーション固有のコンポーネント側のパフォーマンスが向上します。WebSphere MQ を使用すると、コネクターのポーリング・スレッドは、イベントを選出した後、コネクターのキューにそのイベントを入れ、次のイベントを選出します。この方法は IDL よりも高速で、IDL の場合、コネクターのポーリング・スレッドは、イベントを選出した後、ネットワーク経由でサーバー・プロセスにアクセスしてそのイベントをリポジトリー・データベースに永続的に格納してから、次のイベントを選出する必要があります。

JMS

Java Messaging Service (JMS) を使用しての、コネクターとクライアント・コネクター・フレームワークとの間の通信を可能にします。

JMS をデリバリー・トランスポートとして選択した場合は、
jms.MessageBrokerName、jms.FactoryClassName、jms.Password、jms.UserName などの追加の JMS プロパティーが Connector Configurator 内に表示されます。このうち最初の 2 つは、このトランスポートの必須プロパティーです。

重要: 以下の環境では、コネクターに JMS トランスポート機構を使用すると、メモリー制限が発生することもあります。

- AIX 5.0
- WebSphere MQ 5.3.0.1
- ICS が統合ブローカーの場合

この環境では、WebSphere MQ クライアント内でメモリーが使用されるため、(サーバー側の) コネクター・コントローラーと (クライアント側の) コネクターの両方を始動するのは困難な場合があります。ご使用のシステムのプロセス・ヒープ・サイズが 768M 未満である場合には、次のように設定することをお勧めします。

- CWSharedEnv.sh スクリプト内で LDR_CNTRL 環境変数を設定する。

  このスクリプトは、製品ディレクトリー配下の ¥bin ディレクトリーにあります。テキスト・エディターを使用して、CWSharedEnv.sh スクリプトの最初の行として次の行を追加します。

  ```
  export LDR_CNTRL=MAXDATA=0x30000000
  ```

  この行は、ヒープ・メモリーの使用量を最大 768 MB (3 セグメント * 256 MB) に制限します。プロセス・メモリーがこの制限値を超えると、ページ・スワッピングが発生し、システムのパフォーマンスに悪影響を与える場合があります。
- IPCCBaseAddress プロパティーの値を 11 または 12 に設定する。このプロパティーの詳細については、「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

DuplicateEventElimination

このプロパティーを true に設定すると、JMS 対応コネクターによるデリバリー・キューへの重複イベントのデリバリーが防止されます。この機能を使用するには、コネクターに対し、アプリケーション固有のコード内でビジネス・オブジェクトの **ObjectEventId** 属性として一意のイベント ID が設定されている必要があります。これはコネクター開発時に設定されます。

このプロパティーは、false に設定することもできます。

注: DuplicateEventElimination を true に設定する際は、MonitorQueue プロパティーを構成して保証付きイベント・デリバリーを使用可能にする必要があります。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、そのメッセージを状況表示および問題説明とともにこのプロパティーに指定されているキューに移動します。

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128M です。

JvmMaxNativeStackSize

エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128K です。

JvmMinHeapSize

エージェントの最小ヒープ・サイズ (メガバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 1M です。

jms.FactoryClassName

JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクター・プロパティーを必ず 設定してください。

デフォルト値は CxCommon.Messaging.jms.IBMMQSeriesFactory です。

jms.MessageBrokerName

JMS プロバイダーのために使用するブローカー名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクター・プロパティーを必ず 設定してください。

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

コネクターに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出し要求はブロックされ、既存のいずれかの要求が完了した後で処理されます。

デフォルト値は 10 です。

jms.Password

JMS プロバイダーのためのパスワードを指定します。このプロパティーの値はオプションです。

デフォルトはありません。

jms.UserName

JMS プロバイダーのためのユーザー名を指定します。このプロパティーの値はオプションです。

デフォルトはありません。

ListenerConcurrency

このプロパティーは、統合ブローカーとして ICS を使用する場合の MQ Listener でのマルチスレッド化をサポートしています。このプロパティーにより、データベースへの複数イベントの書き込み操作をバッチ処理できるので、システム・パフォーマンスが向上します。デフォルト値は 1 です。

このプロパティーは、MQ トランスポートを使用するコネクターにのみ適用されます。DeliveryTransport プロパティーには MQ を設定してください。

Locale

言語コード、国または地域、および、希望する場合には、関連した文字コード・セットを指定します。このプロパティーの値は、データの照合やソート順、日付と時刻の形式、通貨記号などの国/地域別情報を決定します。

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

ll	2 文字の言語コード (普通は小文字)
TT	2 文字の国または地域コード (普通は大文字)
codeset	関連文字コード・セットの名前。名前のこの部分は、通常、オプションです。

デフォルトでは、ドロップ・リストには、サポートされるロケールの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、Connector Configurator に関する付録を参照してください。

デフォルト値は en_US です。コネクターがグローバル化に対応していない場合、このプロパティーの有効な値は en_US のみです。特定のコネクターがグローバル化に対応しているかどうかを判別するには、以下の Web サイトにあるコネクターのバージョン・リストを参照してください。

http://www.ibm.com/software/websphere/wbiadapters/infocenter、または
http://www.ibm.com/websphere/integration/wicserver/infocenter

LogAtInterchangeEnd

RepositoryDirectory が <REMOTE> の場合のみ適用可能です。

統合ブローカーのログ宛先にエラーを記録するかどうかを指定します。ブローカーのログ宛先にログを記録すると、電子メール通知もオンになります。これにより、エラーまたは致命的エラーが発生すると、InterchangeSystem.cfg ファイルに指定された MESSAGE_RECIPIENT に対する電子メール・メッセージが生成されます。

例えば、LogAtInterChangeEnd を true に設定した場合にコネクターからアプリケーションへの接続が失われると、指定されたメッセージ宛先に、電子メール・メッセージが送信されます。デフォルト値は false です。

MaxEventCapacity

コントローラー・バッファー内のイベントの最大数。このプロパティーはフロー制御が使用し、RepositoryDirectory プロパティーの値が <REMOTE> の場合にのみ適用されます。

値は 1 から 2147483647 の間の正整数です。デフォルト値は 2147483647 です。

MessageFileName

コネクター・メッセージ・ファイルの名前です。メッセージ・ファイルの標準位置は ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクター・メッセージ・ファイルが存在しない場合は、コネクターは InterchangeSystem.txt をメッセージ・ファイルとして使用します。このファイルは、製品ディレクトリーに格納されています。

注: 特定のコネクターについて、コネクター独自のメッセージ・ファイルがあるかどうかを判別するには、該当するアダプターのユーザーズ・ガイドを参照してください。

MonitorQueue

コネクターが重複イベントをモニターするために使用する論理キューです。このプロパティーは、DeliveryTransport プロパティー値が JMS であり、かつ DuplicateEventElimination が TRUE に設定されている場合にのみ使用されます。

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

RepositoryDirectory が <REMOTE> の場合のみ有効です。

コネクターが自動再始動およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用して、異常シャットダウン後にコネクターを再始動したり、System Monitor からリモート・コネクターを始動したりします。

自動再始動機能およびリモート再始動機能を使用可能にするには、このプロパティーを true に設定する必要があります。MQ により起動される OAD 機能の構成方法については、「*システム・インストール・ガイド (Windows 版)*」または「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

RepositoryDirectory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後で MQ により起動される OAD がコネクターの再始動を自動的に試行する回数の最大数を指定します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

RepositoryDirectory が <REMOTE> の場合のみ有効です。

MQ により起動される OAD の再試行時間間隔の分数を指定します。コネクター・エージェントがこの再試行時間間隔内に再始動しない場合は、コネクター・コントローラーはコネクター・エージェントを再び再始動するように OAD に要求します。OAD はこの再試行プロセスを OADMaxNumRetry プロパティーで指定された回数だけ繰り返します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルトは 10 です。

PollEndTime

イベント・キューのポーリングを停止する時刻です。形式は HH:MM です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

PollFrequency

ポーリング・アクション間の時間の長さです。PollFrequency は以下の値のいずれかに設定します。

- ポーリング・アクション間のミリ秒数。
- ワード key。コネクターは、コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力されたときにのみポーリングを実行します。このワードは小文字で入力します。
- ワード no。コネクターはポーリングを実行しません。このワードは小文字で入力します。

デフォルト値は 10000 です。

重要: 一部のコネクターでは、このプロパティーの使用が制限されています。このプロパティーが使用されるかどうかを特定のコネクターについて判別するには、該当するアダプター・ガイドのインストールと構成についての章を参照してください。

PollQuantity

コネクターがアプリケーションからポーリングする項目の数を指定します。アダプターにコネクター固有のポーリング数設定プロパティーがある場合、標準プロパティーの値は、このコネクター固有のプロパティーの設定値によりオーバーライドされます。

PollStartTime

イベント・キューのポーリングを開始する時刻です。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

RequestQueue

統合ブローカーが、ビジネス・オブジェクトをコネクターに送信するときに使用されるキューです。

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

コネクターが XML スキーマ文書を読み取るリポジトリーの場所です。この XML スキーマ文書には、ビジネス・オブジェクト定義のメタデータが含まれています。

統合ブローカーが ICS の場合はこの値を <REMOTE> に設定する必要があります。これは、コネクターが InterChange Server リポジトリーからこの情報を取得するためです。

統合ブローカーが WebSphere Message Broker または WAS の場合は、この値を <*local directory*> に設定する必要があります。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用可能で、RepositoryDirectory が <REMOTE> の場合のみ必須です。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・フレームワークから統合ブローカーへデリバリーします。統合ブローカーが ICS の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartRetryCount

コネクターによるコネクター自体の再始動の試行回数を指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する回数が指定されます。

デフォルト値は 3 です。

RestartRetryInterval

コネクターによるコネクター自体の再始動の試行間隔を分単位で指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する間隔が指定されます。指定可能な値の範囲は 1 から 2147483647 です。

デフォルトは 1 です。

RHF2MessageDomain

WebSphere Message Brokers および WAS でのみ使用されます。

このプロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WMQI に送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 mrm が書き込まれます。この構成可能なドメイン名により、ユーザーは WMQI ブローカーによるメッセージ・データの処理方法を追跡できます。

サンプル・ヘッダーを以下に示します。

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

デフォルト値は mrm ですが、このプロパティーには xml も設定できます。このプロパティーは、DeliveryTransport が JMS に設定されており、かつ WireFormat が CwXML に設定されている場合にのみ表示されます。

SourceQueue

DeliveryTransport が JMS で、ContainerManagedEvents が指定されている場合のみ適用されます。

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、110 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期応答を要求する要求メッセージを、コネクター・フレームワークからブローカーにデリバリーします。このキューは、コネクターが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクター・フレームワークは、SynchronousRequestQueue にメッセージを送信し、SynchronousResponseQueue でブローカーから戻される応答を待機します。コネクターに送信される応答メッセージには、元のメッセージの ID を指定する 相関 ID が含まれています。

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期要求に対する応答として送信される応答メッセージを、ブローカーからコネクター・フレームワークにデリバリーします。このキューは、コネクターが同期実行を使用する場合にのみ必要です。

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

- RepositoryDirectory がローカル・ディレクトリーの場合は、設定は CwXML になります。
- RepositoryDirectory の値が <REMOTE> の場合には、設定値は CwBO です。

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

XMLNameSpaceFormat

WebSphere Message Brokers および WAS 統合ブローカーでのみ使用されます。

ビジネス・オブジェクト定義の XML 形式でネーム・スペースを short と long のどちらにするかをユーザーが指定できるようにするための、強力なプロパティーです。

デフォルト値は short です。

付録 B. Connector Configurator

この付録では、Connector Configurator を使用してアダプターの構成プロパティー値を設定する方法について説明します。

Connector Configurator を使用して次の作業を行います。

- コネクターを構成するためのコネクター固有のプロパティー・テンプレートを作成する
- 構成ファイルを作成する
- 構成ファイル内のプロパティーを設定する

注:

本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムを使用している場合は、円記号をスラッシュ (/) に置き換えてください。また、各オペレーティング・システムの規則に従ってください。

この付録では、次のトピックについて説明します。

Connector Configurator の概要

Connector Configurator では、次の統合ブローカーで使用するアダプターのコネクター・コンポーネントを構成できます。

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker (WebSphere Message Brokers (WMQI) と総称)
- WebSphere Application Server (WAS)

Connector Configurator を使用して次の作業を行います。

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定する。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、ICS の場合はコラボレーションとともに使用するマップを

指定し、必要に応じてメッセージング、ロギング、トレース、およびデータ・ハンドラー・パラメーターを指定する必要があります。

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なります。例えば、使用している統合ブローカーが WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (122 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

コネクター構成プロパティーには、標準の構成プロパティー (すべてのコネクターがもつプロパティー) と、コネクター固有のプロパティー (特定のアプリケーションまたはテクノロジーのためにコネクターで必要なプロパティー) とが含まれます。

標準プロパティーはすべてのコネクターにより使用されるので、標準プロパティーを新規に定義する必要はありません。ファイルを作成すると、Connector Configurator により標準プロパティーがこの構成ファイルに挿入されます。ただし、Connector Configurator で各標準プロパティーの値を設定する必要があります。

標準プロパティーの範囲は、ブローカーと構成によって異なる可能性があります。特定のプロパティーに特定の値が設定されている場合にのみ使用できるプロパティーがあります。Connector Configurator の「標準のプロパティー」ウィンドウには、特定の構成で設定可能なプロパティーが表示されます。

ただし、**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内ですでにテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、123 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

注: Connector Configurator は、Windows 環境内でのみ実行されます。UNIX 環境でコネクターを実行する場合には、Windows で Connector Configurator を使用して構成ファイルを変更し、このファイルを UNIX 環境へコピーします。

Connector Configurator の始動

以下の 2 種類のモードで Connector Configurator を開始および実行できます。

- スタンドアロン・モードで個別に実行
- System Manager から

スタンドアロン・モードでの Configurator の実行

どのブローカーを実行している場合にも、Connector Configurator を個別に実行し、コネクター構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Connector Configurator」**をクリックします。
- **「ファイル」>「新規」>「構成ファイル」**を選択します。

- **「システム接続: Integration Broker」**の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (128 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

System Manager から Connector Configurator を実行できます。

Connector Configurator を実行するには、以下のステップを実行します。

1. System Manager を開きます。
2. 「System Manager」ウィンドウで、**「統合コンポーネント・ライブラリー」**アイコンを展開し、**「コネクター」**を強調表示します。
3. System Manager メニュー・バーから、**「ツール」>「Connector Configurator」**をクリックします。「Connector Configurator」ウィンドウが開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。
4. **「システム接続: Integration Broker」**の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーでいずれかの構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
2. 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

コネクターの構成ファイルを作成するには、コネクター固有プロパティーのテンプレートとシステム提供の標準プロパティーが必要です。

コネクター固有プロパティーのテンプレートを新規に作成するか、または既存のファイルをテンプレートとして使用します。

- テンプレートの新規作成については、123 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティーを作成し、プロパティーの一般特性および値を定義し、プロパティー間の依存関係を指定する方法について説明します。次にそのテンプレートを保管し、新規コネクター構成ファイルを作成するためのベースとして使用します。

テンプレートは以下のように作成します。

1. **「ファイル」>「新規」>「コネクター固有プロパティー・テンプレート」**をクリックします。
2. 以下のフィールドを含む**「コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。
   - **「テンプレート」、「名前」**

     このテンプレートが使用されるコネクター (またはコネクターのタイプ) を表す固有の名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - **「旧テンプレート」、「変更する既存のテンプレートを選択してください」**

     「テンプレート名」表示に、現在使用可能なすべてのテンプレートの名前が表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、**「テンプレート名」**表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが**「テンプレートのプレビュー」**表示に表示されます。テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。
3. **「テンプレート名」**表示からテンプレートを選択し、その名前を**「名前の検索」**フィールドに入力し (または**「テンプレート名」**で自分の選択項目を強調表示し)、**「次へ」**をクリックします。

ご使用のコネクターで使用するコネクター固有のプロパティーが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。

一般特性の指定

「次へ」をクリックしてテンプレートを選択すると、**「プロパティー: コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスには、定義済みプロパティーの「一般」特性のタブと「値」の制限のタブがあります。「一般」表示には以下のフィールドがあります。

- **一般:**
  プロパティー・タイプ
  更新されたメソッド
  説明
- **フラグ**
  標準フラグ
- **カスタム・フラグ**
  フラグ

プロパティーの一般特性の選択を終えたら、**「値」**タブをクリックします。

値の指定

「値」タブを使用すると、プロパティーの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。編集可能な値も許可されます。これを行うには、以下のステップを実行します。

1. **「値」**タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドで、変更を行います。次のステップで説明するように、プロパティーの**「プロパティー値」**ダイアログ・ボックスを開かない限り、そのプロパティーの変更内容は受け入れられませんので、注意してください。
4. 値テーブルの左上の隅にあるボックスを右マウス・ボタンでクリックしてから、**「追加」**をクリックします。**「プロパティー値」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスではプロパティーのタイプに応じて、値だけを入力できる場合と、値と範囲の両方を入力できる場合があります。適切な値または範囲を入力し、**「OK」**をクリックします。
5. **「値」**パネルが最新表示され、**「最大長」**および**「最大複数値」**で行った変更が表示されます。以下のような 3 つの列があるテーブルが表示されます。

   「値」の列には、**「プロパティー値」**ダイアログ・ボックスで入力した値と、以前に作成した値が表示されます。

   「デフォルト値」の列では、値のいずれかをデフォルトとして指定することができます。

   「値の範囲」の列には、**「プロパティー値」**ダイアログ・ボックスで入力した範囲が表示されます。

   値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。テーブルにある既存の値の変更を行うには、その行の行番号をクリックして行全体を選択します。次に**「値」**フィールドを右マウス・ボタンでクリックし、**「値の編集 (Edit Value)」**をクリックします。

依存関係の設定

「一般」タブと**「値」**タブで変更を行ったら、**「次へ」**をクリックします。**「依存関係: コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。

依存プロパティーは、別のプロパティーの値が特定の条件に合致する場合にのみ、テンプレートに組み込まれて、構成ファイルで使用されるプロパティーです。例えば、テンプレートに `PollQuantity` が表示されるのは、トランスポート機構が JMS であり、`DuplicateEventElimination` が `True` に設定されている場合のみです。プロパティーを依存プロパティーとして指定し、依存する条件を設定するには、以下のステップを実行します。

1. **「使用可能なプロパティー」**表示で、依存プロパティーとして指定するプロパティーを選択します。
2. **「プロパティーを選択」**フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティーを選択します。
3. **「条件演算子」**フィールドで以下のいずれかを選択します。

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

>= (より大か等しい)

<= (より小か等しい)

4. **「条件値」**フィールドで、依存プロパティーをテンプレートに組み込むために必要な値を入力します。
5. **「使用可能なプロパティー」**表示で依存プロパティーを強調表示させて矢印をクリックし、**「依存プロパティー」**表示に移動させます。
6. **「完了」**をクリックします。Connector Configurator により、XML 文書として入力した情報が、Connector Configurator がインストールされている ¥bin ディレクトリーの ¥data¥app の下に保管されます。

新しい構成ファイルを作成

構成ファイルを新規に作成するには、最初に統合ブローカーを選択します。選択したブローカーによって、構成ファイルに記述されるプロパティーが決まります。

ブローカーを選択するには、以下のステップを実行します。

- Connector Configurator のホーム・メニューで、**「ファイル」>「新規」>「コネクター構成」**をクリックします。**「新規コネクター」**ダイアログ・ボックスが表示されます。
- **「Integration Broker」**フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って**「新規コネクター」**ウィンドウの残りのフィールドに入力します。

また、以下の作業も実行できます。

- 「System Manager」ウィンドウで**「コネクター」**フォルダーを右クリックし、**「新規コネクターの作成」**を選択します。Connector Configurator が開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。

コネクター固有のテンプレートからの構成ファイルの作成

コネクター固有のテンプレートを作成すると、テンプレートを使用して構成ファイルを作成できます。

1. **「ファイル」>「新規」>「コネクター構成」**をクリックします。
2. 以下のフィールドを含む**「新規コネクター」**ダイアログ・ボックス表示されます。
   - **名前**

     コネクターの名前を入力します。名前では大文字と小文字が区別されます。入力する名前は、システムにインストールされているコネクターのファイル名に対応した一意の名前でなければなりません。

     重要: Connector Configurator では、入力された名前のスペルはチェックされません。名前が正しいことを確認してください。
   - **システム接続**

     ICS 接続、WebSphere Message Brokers 接続、WAS のいずれかをクリックします。

- **「コネクター固有プロパティー・テンプレート」を選択します。**

  ご使用のコネクター用に設計したテンプレートの名前を入力します。**「テンプレート名」**表示に、使用可能なテンプレートが表示されます。「テンプレート名」表示で名前を選択すると、**「プロパティー・テンプレートのプレビュー」**表示に、そのテンプレートで定義されているコネクター固有のプロパティーが表示されます。

  使用するテンプレートを選択し、**「OK」**をクリックします。

3. 構成しているコネクターの構成画面が表示されます。タイトル・バーに統合ブローカーとコネクターの名前が表示されます。ここですべてのフィールドに値を入力して定義を完了するか、ファイルを保管して後でフィールドに値を入力するかを選択できます。
4. ファイルを保管するには、**「ファイル」>「保管」>「ファイルに」**をクリックするか、**「ファイル」>「保管」>「プロジェクトに」**をクリックします。プロジェクトに保管するには、System Manager が実行中でなければなりません。
   ファイルとして保管する場合は、**「ファイル・コネクターを保管」**ダイアログ・ボックスが表示されます。`*.cfg` をファイル・タイプとして選択し、「ファイル名」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、**「保管」**をクリックします。Connector Configurator のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

   重要: ここで設定するディレクトリー・パスおよび名前は、コネクターの始動ファイルで指定するコネクター構成ファイルのパスおよび名前に一致している必要があります。
5. この章で後述する手順に従って、「Connector Configurator」ウィンドウの各タブにあるフィールドに値を入力し、コネクター定義を完了します。

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

- コネクター定義ファイル。
  コネクター定義ファイルは、特定のコネクターのプロパティーと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクターの配布パッケージの `¥repository` ディレクトリー内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は `.txt` です。例えば、XML コネクターの場合は CN_XML.txt です)。
- ICS リポジトリー・ファイル。
  コネクターの以前の ICS インプリメンテーションで使用した定義は、そのコネクターの構成で使用されたリポジトリー・ファイルで使用可能になります。そのようなファイルの拡張子は、通常 `.in` または `.out` です。
- コネクターの以前の構成ファイル。
  これらのファイルの拡張子は、通常 *.cfg です。

これらのいずれのファイル・ソースにも、コネクターのコネクター固有プロパティーのほとんど、あるいはすべてが含まれますが、この章内の後で説明するように、コネクター構成ファイルは、ファイルを開いて、プロパティーを設定しない限り完成しません。

既存ファイルを使用してコネクターを構成するには、Connector Configurator でそのファイルを開き、構成を修正し、そのファイルを再度保管する必要があります。

以下のステップを実行して、ディレクトリーから *.txt、*.cfg、または *.in ファイルを開きます。

1. Connector Configurator 内で、**「ファイル」>「開く」>「ファイルから」**をクリックします。
2. **「ファイル・コネクターを開く」**ダイアログ・ボックス内で、以下のいずれかのファイル・タイプを選択して、使用可能なファイルを調べます。
   - 構成 (*.cfg)
   - ICS リポジトリー (*.in、*.out)

     ICS 環境でのコネクターの構成にリポジトリー・ファイルが使用された場合には、このオプションを選択します。リポジトリー・ファイルに複数のコネクター定義が含まれている場合は、ファイルを開くとすべての定義が表示されます。
   - すべてのファイル (*.*)

     コネクターのアダプター・パッケージに *.txt ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。
3. ディレクトリー表示内で、適切なコネクター定義ファイルへ移動し、ファイルを選択し、**「開く」**をクリックします。

System Manager プロジェクトからコネクター構成を開くには、以下のステップを実行します。

1. System Manager を始動します。System Manager が開始されている場合にのみ、構成を System Manager から開いたり、System Manager に保管したりできます。
2. Connector Configurator を始動します。
3. **「ファイル」>「開く」>「プロジェクトから」**をクリックします。

構成ファイルの完成

構成ファイルを開くか、プロジェクトからコネクターを開くと、「Connector Configurator」ウィンドウに構成画面が表示されます。この画面には、現在の属性と値が表示されます。

構成画面のタイトルには、ファイル内で指定された統合ブローカーとコネクターの名前が表示されます。正しいブローカーが設定されていることを確認してください。正しいブローカーが設定されていない場合、コネクターを構成する前にブローカー値を変更してください。これを行うには、以下のステップを実行します。

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 131 ページの

『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。

3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

   ファイルに保管する場合は、`*.cfg` を拡張子として選択し、ファイルの正しい格納場所を選択して、**「保管」**をクリックします。

   複数のコネクター構成を開いている場合、構成をすべてファイルに保管するには**「すべてファイルに保管」**を選択し、コネクター構成をすべて System Manager プロジェクトに保管するには**「すべてプロジェクトに保管」**をクリックします。

   Connector Configurator では、ファイルを保管する前に、必須の標準プロパティーすべてに値が設定されているかどうかが確認されます。必須の標準プロパティーに値が設定されていない場合、Connector Configurator は、検証が失敗したというメッセージを表示します。構成ファイルを保管するには、そのプロパティーの値を指定する必要があります。

構成ファイル・プロパティーの設定

新規のコネクター構成ファイルを作成して名前を付けるとき、または既存のコネクター構成ファイルを開くときには、Connector Configurator によって構成画面が表示されます。構成画面には、必要な構成値のカテゴリーに対応する複数のタブがあります。

Connector Configurator では、すべてのブローカーで実行されているコネクターで、以下のカテゴリーのプロパティーに値が設定されている必要があります。

- 標準のプロパティー
- コネクター固有のプロパティー
- サポートされるビジネス・オブジェクト
- トレース/ログ・ファイルの値
- データ・ハンドラー (保証付きイベント・デリバリーで JMS メッセージングを使用するコネクターの場合に該当する)

注: JMS メッセージングを使用するコネクターの場合は、データをビジネス・オブジェクトに変換するデータ・ハンドラーの構成に関して追加のカテゴリーが表示される場合があります。

ICS で実行されているコネクターの場合、以下のプロパティーの値も設定されている必要があります。

- 関連付けられたマップ
- リソース
- メッセージング (該当する場合)

重要: Connector Configurator では、英語文字セットまたは英語以外の文字セットのいずれのプロパティー値も設定可能です。ただし、標準のプロパティーおよびコネクター固有プロパティー、およびサポートされるビジネス・オブジェクトの名前では、英語文字セットのみを使用する必要があります。

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

- コネクターの標準プロパティーは、コネクターのアプリケーション固有のコンポーネントとブローカー・コンポーネントの両方によって共用されます。すべてのコネクターが同じ標準プロパティーのセットを使用します。これらのプロパティーの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。
- アプリケーション固有のプロパティーは、コネクターのアプリケーション固有コンポーネント (アプリケーションと直接対話するコンポーネント) のみに適用されます。各コネクターには、そのコネクターのアプリケーションだけで使用されるアプリケーション固有のプロパティーがあります。これらのプロパティーには、デフォルト値が用意されているものもあれば、そうでないものもあります。また、一部のデフォルト値は変更することができます。各アダプター・ガイドのインストールおよび構成の章に、アプリケーション固有のプロパティーおよび推奨値が記述されています。

「標準プロパティー」と**「コネクター固有プロパティー」**のフィールドは、どのフィールドが構成可能であるかを示すために色分けされています。

- 背景がグレーのフィールドは、標準のプロパティーを表します。値を変更することはできますが、名前の変更およびプロパティーの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティーを表します。これらのプロパティーは、アプリケーションまたはコネクターの特定のニーズによって異なります。値の変更も、これらのプロパティーの除去も可能です。
- 「値」フィールドは構成できます。
- **「更新メソッド」**フィールドは通知用であり、構成できません。このフィールドは、値が変更されたプロパティーをアクティブにするために必要なアクションを示します。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。
3. 標準のプロパティーの値をすべて入力後、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator を終了するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「いいえ」**をクリックします。
   - Connector Configurator 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。**「標準のプロパティー」**(またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
   - 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

アプリケーション固有の構成プロパティーの場合、プロパティー名の追加または変更、値の構成、プロパティーの削除、およびプロパティーの暗号化が可能です。プロパティーのデフォルトの長さは 255 文字です。

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 130 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

各プロパティーごとに表示される「更新メソッド」は、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。

重要: 事前設定のアプリケーション固有のコネクター・プロパティー名を変更すると、コネクターに障害が発生する可能性があります。コネクターをアプリケーションに接続したり正常に実行したりするために、特定のプロパティー名が必要である場合があります。

コネクター・プロパティーの暗号化

「プロパティーを編集」ウィンドウの**「暗号化」**チェック・ボックスにチェックマークを付けると、アプリケーション固有のプロパティーを暗号化することができます。値の暗号化を解除するには、**「暗号化」**チェック・ボックスをクリックしてチェックマークを外し、**「検証」**ダイアログ・ボックスに正しい値を入力し、**「OK」**をクリックします。入力された値が正しい場合は、暗号化解除された値が表示されます。

各プロパティーとそのデフォルト値のリストおよび説明は、各コネクターのアダプター・ユーザーズ・ガイドにあります。

プロパティーに複数の値がある場合には、プロパティーの最初の値に**「暗号化」**チェック・ボックスが表示されます。**「暗号化」**を選択すると、そのプロパティーのすべての値が暗号化されます。プロパティーの複数の値を暗号化解除するには、そのプロパティーの最初の値の**「暗号化」**チェック・ボックスをクリックしてチェックマークを外してから、**「検証」**ダイアログ・ボックスで新規の値を入力します。入力値が一致すれば、すべての複数値が暗号化解除されます。

更新メソッド

付録『コネクターの*標準構成プロパティー*』の 104 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

コネクターで使用するビジネス・オブジェクトを指定するには、Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブを使用します。

汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

注: コネクターによっては、アプリケーションでイベント通知や (メタオブジェクトを使用した) 追加の構成を実行するために、特定のビジネス・オブジェクトをサポートされているものとして指定することが必要な場合もあります。詳細は、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

ご使用のブローカーが ICS の場合

ビジネス・オブジェクト定義がコネクターでサポートされることを指定する場合や、既存のビジネス・オブジェクト定義のサポート設定を変更する場合は、**「サポートされているビジネス・オブジェクト」**タブをクリックし、以下のフィールドを使用してください。

ビジネス・オブジェクト名: ビジネス・オブジェクト定義がコネクターによってサポートされることを指定するには、System Manager を実行し、以下の手順を実行します。

1. **「ビジネス・オブジェクト名」**リストで空のフィールドをクリックします。System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの**「エージェント・サポート」**(以下で説明) を設定します。
4. 「Connector Configurator」ウィンドウの「ファイル」メニューで、**「プロジェクトに保管」**をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager のプロジェクトに保管されます。

サポートされるリストからビジネス・オブジェクトを削除する場合は、以下の手順を実行します。

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator」ウィンドウの**「編集」**メニューから、**「行を削除」**をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. **「ファイル」**メニューから、**「プロジェクトに保管」**をクリックします。

サポートされるリストからビジネス・オブジェクトを削除すると、コネクター定義が変更され、削除されたビジネス・オブジェクトはコネクターのこのインプリメンテーションで使用不可になります。コネクターのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

エージェント・サポート: ビジネス・オブジェクトがエージェント・サポートを備えている場合、システムは、コネクター・エージェントを介してアプリケーションにデータを配布する際にそのビジネス・オブジェクトの使用を試みます。

一般に、コネクターのアプリケーション固有ビジネス・オブジェクトは、そのコネクターのエージェントによってサポートされますが、汎用ビジネス・オブジェクトはサポートされません。

ビジネス・オブジェクトがコネクター・エージェントによってサポートされるよう指定するには、**「エージェント・サポート」**ボックスにチェックマークを付けます。「Connector Configurator」ウィンドウでは「エージェント・サポート」の選択の妥当性は検査されません。

最大トランザクション・レベル: コネクターの最大トランザクション・レベルは、そのコネクターがサポートする最大のトランザクション・レベルです。

ほとんどのコネクターの場合、選択可能な項目は「最大限の努力」のみです。

トランザクション・レベルの変更を有効にするには、サーバーを再始動する必要があります。

ご使用のブローカーが WebSphere Message Broker の場合

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス名を入力する必要があります。

System Manager を実行している場合、**「サポートされているビジネス・オブジェクト」**タブの**「ビジネス・オブジェクト名」**列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。リストから必要なビジネス・オブジェクトを選択します。

「メッセージ・セット ID」は、WebSphere Business Integration Message Broker 5.0 のオプションのフィールドです。この ID が提供される場合、一意である必要はありません。ただし、WebSphere MQ Integrator および Integrator Broker 2.1 の場合は、一意の **ID** を提供する必要があります。

ご使用のブローカーが WAS の場合

使用するブローカー・タイプとして WebSphere Application Server を選択した場合、Connector Configurator にメッセージ・セット ID は必要ありません。**「サポートされているビジネス・オブジェクト」**タブには、サポートされるビジネス・オブジェクトの**「ビジネス・オブジェクト名」**列のみが表示されます。

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス・オブジェクト名を入力する必要があります。

System Manager を実行している場合、「サポートされているビジネス・オブジェクト」タブの「ビジネス・オブジェクト名」列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。このリストから必要なビジネス・オブジェクトを選択します。

関係付けられたマップ (ICS のみ)

各コネクターは、現在 WebSphere InterChange Server でアクティブなビジネス・オブジェクト定義、およびそれらの関連付けられたマップのリストをサポートします。このリストは、**「関連付けられたマップ」**タブを選択すると表示されます。

ビジネス・オブジェクトのリストには、エージェントでサポートされるアプリケーション固有のビジネス・オブジェクトと、コントローラーがサブスクライブ・コラボレーションに送信する、対応する汎用オブジェクトが含まれます。マップの関連によって、アプリケーション固有のビジネス・オブジェクトを汎用ビジネス・オブジェクトに変換したり、汎用ビジネス・オブジェクトをアプリケーション固有のビジネス・オブジェクトに変換したりするときに、どのマップを使用するかが決定されます。

特定のソースおよび宛先ビジネス・オブジェクトについて一意的に定義されたマップを使用する場合、表示を開くと、マップは常にそれらの該当するビジネス・オブジェクトに関連付けられます。ユーザーがそれらを変更する必要はありません (変更できません)。

サポートされるビジネス・オブジェクトで使用可能なマップが複数ある場合は、そのビジネス・オブジェクトを、使用する必要のあるマップに明示的にバインドすることが必要になります。

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

  これらは、**「サポートされているビジネス・オブジェクト」**タブで指定した、このコネクターでサポートされるビジネス・オブジェクトです。「サポートされているビジネス・オブジェクト」タブでビジネス・オブジェクトを追加指定した場合、その内容は、「Connector Configurator」ウィンドウの**「ファイル」**メニューから**「プロジェクトに保管」**を選択して、変更を保管した後に、このリストに反映されます。

- **関連付けられたマップ**

  この表示には、コネクターの、サポートされるビジネス・オブジェクトでの使用のためにシステムにインストールされたすべてのマップが示されます。各マップのソース・ビジネス・オブジェクトは、**「ビジネス・オブジェクト名」**表示でマップ名の左側に表示されます。

- **明示的**

  場合によっては、関連マップを明示的にバインドすることが必要になります。

  明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバインドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見つけて、バインドしようとします。

  他のマップのスーパーセットであるマップがないと、サーバーは、ビジネス・オブジェクトを単一のマップにバインドすることができないため、バインディングを明示的に設定することが必要になります。

  以下の手順を実行して、マップを明示的にバインドします。

1. **「明示的 (Explicit)」**列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
2. ビジネス・オブジェクトに関連付けるマップを選択します。
3. 「Connector Configurator」ウィンドウの**「ファイル」**メニューで、**「プロジェクトに保管」**をクリックします。
4. プロジェクトを ICS に配置します。
5. 変更を有効にするため、サーバーをリブートします。

リソース (ICS)

「リソース」タブでは、コネクター・エージェントが、コネクター・エージェント並列処理を使用して同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定できます。
すべてのコネクターがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

メッセージング・プロパティーは、`DeliveryTransport` 標準プロパティーの値として MQ を設定し、ブローカー・タイプとして ICS を設定した場合にのみ、使用可能です。これらのプロパティーは、コネクターによるキューの使用方法に影響します。

トレース/ログ・ファイル値の設定

コネクター構成ファイルまたはコネクター定義ファイルを開くと、Connector Configurator は、そのファイルのログおよびトレースの値をデフォルト値として使用します。Connector Configurator 内でこれらの値を変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

1. **「トレース/ログ・ファイル」**タブをクリックします。
2. ログとトレースのどちらでも、以下のいずれかまたは両方へのメッセージの書き込みを選択できます。
   - コンソールに (STDOUT):
     ログ・メッセージまたはトレース・メッセージを STDOUT ディスプレイに書き込みます。

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの**「トレース/ログ・ファイル」**タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、**「保管」**をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

注: ログ・ファイルとトレース・ファイルはどちらも単純なテキスト・ファイルです。任意のファイル拡張子を使用してこれらのファイル名を設定できます。ただし、トレース・ファイルの場合、拡張子として .trc ではなく .trace を使用することをお勧めします。これは、システム内に存在する可能性がある他のファイルとの混同を避けるためです。ログ・ファイルの場合、通常使用されるファイル拡張子は .log および .txt です。

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。すべてのアダプターでデータ・ハンドラーを使用できるわけではありません。

これらのプロパティーに使用する値については、付録 A の『コネクターの標準構成プロパティー』の ContainerManagedEvents の下の説明を参照してください。その他の詳細は、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

構成ファイルの保管

コネクターの構成が完了したら、コネクター構成ファイルを保管します。Connector Configurator では、構成中に選択したブローカー・モードでファイルを保管します。Connector Configurator のタイトル・バーには現在のブローカー・モード (ICS、WMQI、または WAS) が常に表示されます。

ファイルは XML 文書として保管されます。XML 文書は次の 3 通りの方法で保管できます。

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。

System Manager でのプロジェクトの使用法、および配置の詳細については、以下のインプリメンテーション・ガイドを参照してください。

- ICS: 「*WebSphere InterChange Server インプリメンテーション・ガイド*」
- WebSphere Message Brokers: 「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」
- WAS: 「*アダプター実装ガイド (WebSphere Application Server)*」

構成ファイルの変更

既存の構成ファイルの統合ブローカー設定を変更できます。これにより、他のブローカーで使用する構成ファイルを新規に作成するときに、このファイルをテンプレートとして使用できます。

注: 統合ブローカーを切り替える場合には、ブローカー・モード・プロパティーと同様に他の構成プロパティーも変更する必要があります。

既存の構成ファイルでのブローカーの選択を変更するには、以下の手順を実行します (オプション)。

- Connector Configurator で既存の構成ファイルを開きます。
- **「標準のプロパティー」**タブを選択します。
- 「標準のプロパティー」タブの**「BrokerType」**フィールドで、ご使用のブローカーに合った値を選択します。
  現行値を変更すると、プロパティー画面の利用可能なタブおよびフィールド選択がただちに変更され、選択した新規ブローカーに適したタブとフィールドのみが表示されます。

構成の完了

コネクターの構成ファイルを作成し、そのファイルを変更した後で、コネクターの始動時にコネクターが構成ファイルの位置を特定できるかどうかを確認してください。

これを行うには、コネクターが使用する始動ファイルを開き、コネクター構成ファイルに使用されている格納場所とファイル名が、ファイルに対して指定した名前およびファイルを格納したディレクトリーまたはパスと正確に一致しているかどうかを検証します。

グローバル化環境における Connector Configurator の使用

Connector Configurator はグローバル化され、構成ファイルと統合ブローカー間の文字変換を処理できます。 Connector Configurator では、ネイティブなエンコード方式を使用しています。構成ファイルに書き込む場合は UTF-8 エンコード方式を使用します。

Connector Configurator は、以下の場所で英語以外の文字をサポートします。

- すべての値のフィールド
- ログ・ファイルおよびトレース・ファイル・パス (**「トレース/ログ・ファイル」**タブで指定)

CharacterEncoding および Locale 標準構成プロパティーのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーの ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。

例えば、Locale プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太文字で示した行を追加してください。

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
                <ValidType>String</ValidType>
            <ValidValues>
                                <Value>ja_JP</Value>
                                <Value>ko_KR</Value>
                                <Value>zh_CN</Value>
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
                                <Value>it_IT</Value>
```

```
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>
                        <DefaultValue>en_US</DefaultValue>
                </ValidValues>
        </Property>
```

付録 C. ビジネス・オブジェクトのサンプル

この付録では、JDBC 用コネクターに組み込まれているサンプル・ビジネス・オブジェクトについて詳しく説明します。JDBC コネクターには、以下のビジネス・オブジェクト・サンプルが含まれています。

- AfterUpdateSPSampleBO.txt
- BeforeCreateSPSampleBO.txt
- BOwithDifferentParameterOrder.txt
- BOwithIOandOPParams.txt
- BOwithFewerSPParamsthanBOAttribs.txt
- CreateSPUpdateSPSampleBO.txt

AfterUpdateSPSampleBO.txt

属性名: AfterUpdateSP

Jdbctest_Customer ビジネス・オブジェクトが含まれています。この属性のアプリケーション固有テキストには、SPN=UpdateAllColumns;IP=fid:CustomerName:CustomerNumber:CustomerDesc という情報があります。ここで、UpdateAllColumns は、4 つのビジネス・オブジェクト属性 (fid、CustomerName、CustomerNumber および CustomerDesc) すべてを入力パラメーターとして使用するストアード・プロシージャーの名前です。このストアード・プロシージャーは、Update 操作の完了後に実行されます。

BeforeCreateSPSampleBO.txt

属性名: BeforeCreateSP

Jdbctest_Customer ビジネス・オブジェクトが含まれています。この属性のアプリケーション固有テキストには、SPN=GetCustomerID;OP=fid という情報があります。ここで、GetCustomerID は、fid ビジネス・オブジェクト属性を出力パラメーターとして使用する (通常は、MasterID テーブルから ID 値を取得するため) ストアード・プロシージャーの名前です。このストアード・プロシージャーは、Create 操作の完了前に実行されます。

BOwithDifferentParameterOrder.txt

属性名: AfterRetrieveSP

Jdbctest_Address ビジネス・オブジェクトが含まれています。この属性のアプリケーション固有テキストには、SPN=UpdateAddress;IP=addressid;IP=zipcode:city:street という情報があります。ここで、UpdateAddress は、すべてのビジネス・オブジェクト属性を入力パラメーターとして使用する (通常は、Jdbctest_Address 以外のテーブル内のアドレスを更新するため) ストアード・プロシージャーの名前です。パラメーターの順序が

ビジネス・オブジェクト属性の順序とは異なること、入力パラメーターで指定される name:value のペアは複数あることに注目してください。ストアード・プロシージャーは、Retrieve 操作の完了後に実行されます。

BOwithIOandOPParams.txt

属性名: RetireveSP

Jdbctest_Address ビジネス・オブジェクトが含まれています。この属性のアプリケーション固有テキストには、SPN=RetrieveAddress;IO=addressid;OP=street:city:zipcode という情報があります。ここで、RetrieveAddress は、ビジネス・オブジェクト属性 addressid を入出力パラメーターとして使用するストアード・プロシージャーの名前です。また、残りのビジネス・オブジェクト属性 zipcode、city、street も出力パラメーターとして使用します。ストアード・プロシージャーは Retrieve 操作の代わりに実行されます。

BOwithFewerSPParamsthanBOAttribs.txt

属性名: AfterUpdateSP

Jdbctest_Address ビジネス・オブジェクトが含まれています。この属性のアプリケーション固有テキストには、SPN=UpdateZipOnly;IP=addressid:zipcode という情報があります。ここで、UpdateZipOnly は、ビジネス・オブジェクト属性 addressid および zipcode を入力パラメーターとして使用するストアード・プロシージャーの名前です。ストアード・プロシージャーのパラメーター総数がビジネス・オブジェクト属性の総数よりも小さいことに注目してください。

CreateSPUpdateSPSampleBO.txt

属性名: CreateSP

Jdbctest_Address ビジネス・オブジェクトが含まれています。この属性のアプリケーション固有テキストには、SPN=CreateAddress;IP=addressid;IP=street:city:zipcode という情報があります。ここで、CreateAddress は、4 つのビジネス・オブジェクト属性すべてを入力パラメーターとして使用するストアード・プロシージャーの名前です。入力パラメーターで指定される name:value のペアは複数あることに注目してください。ストアード・プロシージャーは Create 操作の代わりに実行されます。また、このビジネス・オブジェクトは UpdateSP 属性を持っています。属性には SPN=UpdateCity;IP=addressid:city という情報があります。ここで、UpdateCity は、addressid および city を入力パラメーターとして使用するストアード・プロシージャーの名前です。ストアード・プロシージャーのパラメーター総数がビジネス・オブジェクト属性の総数よりも小さいことに注目してください。ストアード・プロシージャーは Update 操作の代わりに実行されます。

付録 D. ヌル値およびブランク値のサポート

この付録では、ビジネス・オブジェクトのキー値がブランクまたはヌルの場合のさまざまな合格/不合格シナリオについて詳しく説明します。この付録では、ブランクまたはヌルのビジネス・オブジェクト値を持つために必要な機能的変更についても説明します。

合格/不合格シナリオ

ビジネス・オブジェクトのキー値がデータベース内にブランク値またはヌル値を持っている場合は、「=」演算子タイプではなく「is null」タイプの where 文節を構築してください。

IBM では、ビジネス・オブジェクトに対して、ブランク値を持たないキー属性を 1 つ以上定義することをお勧めします。

以下は、1 つのキーを持ち、このキーがヌル値を持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは不合格です。

表 18. Customer

属性	型
cid	Integer (キー)
name	String
comments	String

以下は、2 つのキーを持ち、そのうちの 1 つがヌル値を持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは合格です。

表 19. Customer

属性	型
cid	Integer (キー)
name	String
comments	String

2 つ目の例では、cid=1000 かつ name がヌルに設定されているという条件で customer から cid、name、および comments を選択することにより、retrieve 照会を構築します。

以下は、コンテナー・オブジェクト内に 1 つの外部キー参照を持つ子オブジェクトを 1 つ持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは不合格です。

表 20. Customer

属性	型
cid	Integer (キー)

表 20. Customer (続き)

属性	型
name	String (キー)
comments	String
Address	Address
Aid	Integer (キー) ASI:FK=cid
Acity	String
Azip	String

cid にヌル値が含まれている場合は、address から Aid、Acity、および Azip を選択することにより、retrieve 照会を構築します。Aid の値はヌルに設定してください。

以下は、コンテナー・オブジェクト内に 2 つの外部キー参照を持つ子オブジェクトを 1 つ持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは合格です。

表 21. Customer

属性	型
cid	Integer (キー)
name	String
comments	String
Address	Address
Aid	Integer (キー) ASI:FK=cid
Acity	String (キー) ASI:FK=name
Azip	String

name にヌル値が含まれている場合は、Aid=Cid と Acity がヌル値を持っているという条件で address から Aid、Acity、および Azip を選択することにより、Retrieve 照会を構築します。

機能性

コネクターは、キーにブランク値を見つけると、その値を属性の UseNull 値と比較します。この結果の値が true の場合、コネクターはヌル値を照会に追加します。これにより、以下の動詞の操作が影響を受けます。

- Retrieve
- RetrieveBy Content
- Update
- Delete

特記事項

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。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

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 Burlingame Laboratory Director

IBM Burlingame Laboratory
577 Airport Blvd., Suite 800
Burlingame, CA 94010
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。

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。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

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

以下は、IBM Corporation の商標です。

IBM
IBM ロゴ
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

MMX、Pentium および ProShare は、Intel 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、Sun Microsystems, Inc. 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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

IBM WebSphere Business Integration Adapter Framework V2.4