

**IBM WebSphere Business Integration
Adapters**



Adapter for JDBC ユーザーズ・ガイド

バージョン 2.5.x

**IBM WebSphere Business Integration
Adapters**



Adapter for JDBC ユーザーズ・ガイド

バージョン 2.5.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、157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、Adapter for JDBC (5724-H05) バージョン 2.5.x 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
Version 2.5.x

発 行： 日本アイ・ピー・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 2004.7

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

© Copyright International Business Machines Corporation 2000, 2004. All rights reserved.

© Copyright IBM Japan 2004

目次

図	vii
本書について	ix
対象読者	ix
関連文書	ix
表記上の規則	x
本リリースの新機能	xi
リリース 2.5.x の新機能	xi
リリース 2.4.x の新機能	xi
リリース 2.3.x の新機能	xii
リリース 2.2.x の新機能	xii
リリース 2.1.x の新機能	xiii
リリース 2.0.x の新機能	xiii
リリース 1.9.x の新機能	xiii
リリース 1.8.x の新機能	xiv
リリース 1.7.x の新機能	xiv
リリース 1.6.x の新機能	xiv
リリース 1.5.x の新機能	xiv
第 1 章 コネクターの概要	1
コネクタ・コンポーネント	1
コネクターの動作方法	2
第 2 章 コネクターのインストール	9
アダプター環境	9
前提条件	10
アダプターおよび関連ファイルのインストール	11
インストール済みファイルの構造	11
第 3 章 ビジネス・オブジェクトの理解	15
ビジネス・オブジェクトおよび属性の命名規則	15
ビジネス・オブジェクトの構造	15
ビジネス・オブジェクト動詞の処理	21
ビジネス・オブジェクト属性のプロパティ	39
ビジネス・オブジェクトのアプリケーション固有の情報	42
第 4 章 JDBC ODA を使用したビジネス・オブジェクト定義の生成	55
JDBC ODA のインストールと実行	55
Business Object Designer 内での JDBC ODA の使用	59
生成される定義の内容	69
ビジネス・オブジェクト定義ファイルのサンプル	72
子ビジネス・オブジェクトを含む属性の挿入	73
ビジネス・オブジェクト定義への情報の追加	73
第 5 章 コネクターの構成	75
マルチドライバー・サポートの使用可能化	75
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	75
コネクターの構成プロパティの設定	76
Connector Configurator の使用	77

Connector Configurator の概要	78
Connector Configurator の始動	79
System Manager からの Configurator の実行	80
コネクタ固有のプロパティ・テンプレートの作成	80
新しい構成ファイルを作成	83
既存ファイルの使用	84
構成ファイルの完成	85
構成ファイル・プロパティの設定	86
構成ファイルの保管	93
構成ファイルの変更	94
構成の完了	94
グローバル化環境における Connector Configurator の使用	94
第 6 章 コネクタの実行	97
コネクタの開始	97
コネクタの停止	98
コネクタの複数インスタンスの作成	99
第 7 章 アプリケーションの構成	101
コネクタ用のアプリケーションの使用可能化	101
第 8 章 トラブルシューティングとエラー処理	105
始動時の問題	105
イベント処理	105
マッピング (InterChange Server 統合ブローカーのみ)	105
エラー処理とロギング	107
アプリケーションへの接続不可	109
fetch out of sequence エラー	109
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	109
DB2 データベースと連動するコネクタの使用可能化	109
resource busy エラー	110
JDBC ドライバがサポートされていないため、JDBC ODA が正常に動作しません	110
付録 A. コネクタの標準構成プロパティ	113
新規プロパティと削除されたプロパティ	113
標準コネクタ・プロパティの構成	113
標準プロパティの要約	115
標準構成プロパティ	119
付録 B. コネクタ固有の構成プロパティ	133
プロパティの詳細	134
付録 C. ビジネス・オブジェクトのサンプル	149
AfterUpdateSPSampleBO.txt	149
BeforeCreateSPSampleBO.txt	149
BOwithDifferentParameterOrder.txt	149
BOwithIOandOPParams.txt	150
BOwithFewerSPParamsthanBOAttribs.txt	150
CreateSPUpdateSPSampleBO.txt	150
付録 D. ヌル値およびブランク値のサポート	151
合格/不合格シナリオ	151
機能性	152
付録 E. メインフレーム・データ・アクセス	153
メインフレーム・データ・アクセスに対する JDBC Connector のサポート	153

メインフレーム・データ・アクセスの使用	153
メインフレーム・データ・アクセスのコンポーネント	154
索引	155
特記事項.	157
プログラミング・インターフェース情報	158
商標	159



1. ビジネス・オブジェクト要求に対するコネクタ ー・アーキテクチャー	2	9. オブジェクトを検索するための ALL SCHEMAS の展開	64
2. 典型的な単一カーディナリティー関係	17	10. データベース・オブジェクトの選択の確認	65
3. 複数カーディナリティー・ビジネス・オブジェ クト関係	19	11. データベース・オブジェクトに関する追加情報 の指定	66
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	20	12. ストアード・プロシージャ属性の選択	67
5. ビジネス・オブジェクト間の関係の例	48	13. ストアード・プロシージャとストアード・プ ロシージャ属性の関連付け	68
6. ODA の選択	60	14. メインフレーム・データ・アクセスのサポー ト	153
7. エージェント初期化プロパティーの構成	61		
8. ノードが展開されたスキーマ・ツリー	63		

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、Adapter for JDBC のインストール、構成、ビジネス・オブジェクト開発、およびトラブルシューティングについて説明します。

対象読者

本書は、アダプターをお客様のサイトで使用するコンサルタント、開発者、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/wicsserver/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。

注: 本書の発行後に公開されたテクニカル・サポートの技術情報や速報に、本書の対象製品に関する重要な情報が記載されている場合があります。これらの情報は、WebSphere Business Integration Support Web サイト (<http://www.ibm.com/software/integration/websphere/support/>) にあります。

関心のあるコンポーネント・エリアを選択し、「Technotes」セクションと「Flashes」セクションを参照してください。

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。
<i>ProductDir</i>	製品がインストールされているディレクトリーを表します。IBM WebSphere InterChange Server 環境の場合、デフォルトの製品ディレクトリーは『IBM¥WebSphereICS』です。IBM WebSphere Business Integration Adapters 環境の場合、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 ^(R) システムの場合、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 Connector for JDBC 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: ^(TM)	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 2.5.x の新機能

2004 年 6 月更新。アダプターのバージョン 2.5.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。

- 新規のコネクター固有の構成プロパティには、QueryTimeout、ReturnDummyBOForSP、SelectiveCommitForPoll、および (UseDefaultsForRetrieve に代わって) UseDefaultsWhenPolling があります。既存のプロパティの DriverSupportForLong について記述されています。
- コネクターは、IBM DB2 のシーケンスを使用する固有 ID の生成をサポートします。
- アダプターは、IBM DB2 Information Integration Classic Federation for z/OS^(R) V 8.1 を使用するメインフレーム・データ・アクセスをサポートします。このサポートは、付録 E に説明されています。
- アダプターは、DATE 型の属性に対するアプリケーション固有情報用の新規のパラメーターをサポートします。
- コネクターは、DeltaUpdate 動詞の BeforeDeltaUpdateSP、AfterDeltaUpdateSP、DeltaUpdateSP のストアド・プロシージャ属性名を認識します。
- スキーマのツリーには、生成されたビジネス・オブジェクト定義に関連するデータベース・オブジェクトを示す同義語/ニックネームという名前の追加ノードが含まれます。
- スキーマのツリーには、オブジェクトに関連付けられたスキーマを持たないデータベースからオブジェクトを容易に検索するため、スキーマ名 ALL SCHEMAS が含まれます。
- バージョン 2.5.x の時点で、Adapter for JDBC は Solaris 7.0 ではサポートされていないため、同プラットフォームに対する記述は、本書から削除されています。

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^(TM) 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。インストール情報の新しい記載先については、そ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 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^(R) 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』に、ストアード・プロシージャ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^(R) 5.x
 - Solaris 7 および 8
- Oracle のストアード・プロシージャ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。「CrossWorlds^(R)」という名前は、システム全体について述べたり、コンポーネント名やツール名を修飾したりする目的では使用されなくなりました。名前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
 - 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。その代わりに、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』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 Connector for JDBC 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
- JDBC ODA
- 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、InterChange Server および WebSphere MQ Integrator の両方の統合ブローカーでのアダプターの使用について説明します。

注: コネクタ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

コネクタのビジネス・オブジェクト定義を生成する JDBC ODA が拡張され、その資料も改訂されました。55 ページの『第 4 章 JDBC ODA 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 Microsoft 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で提供されていた、JDBCBOGEN の代わりに使用するものです。55 ページの『第 4 章 JDBC ODA 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 プロパティの説明が、77 ページの『コネクタ固有の構成プロパティ』に追加されました。
- CloseDBConnection プロパティの説明が、77 ページの『コネクタ固有の構成プロパティ』に追加されました。
- SPBeforePollCall プロパティの説明が、77 ページの『コネクタ固有の構成プロパティ』に追加されました。
- 結果セットを戻すストアード・プロシージャに対するサポートが追加されました。36 ページの『ビジネス・オブジェクトの Retrieve 操作』および 38 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。45 ページの表 5 の説明を参照してください。
- カスタム・ビジネス・オブジェクト・ハンドラーに対する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` プロパティの説明が更新されました。詳細については、135ページの『`ArchiveTableName`』を参照してください。
- `AutoCommit` プロパティが、77ページの『コネクター固有の構成プロパティ』に追加されました。
- `DateFormat` プロパティが、77ページの『コネクター固有の構成プロパティ』に追加されました。
- `EventKeyDel` プロパティが更新され、名前と値のペアに対応しました。詳細については、139ページの『`EventKeyDel`』を参照してください。
- `EventQueryType` プロパティが、77ページの『コネクター固有の構成プロパティ』に追加されました。
- `PingQuery` プロパティが、77ページの『コネクター固有の構成プロパティ』に追加されました。
- `PreserveUIDSeq` プロパティが、77ページの『コネクター固有の構成プロパティ』に追加されました。
- `SchemaName` プロパティが、77ページの『コネクター固有の構成プロパティ』に追加されました。
- `RetrieveByContent` 動詞のストアード・プロシージャがサポートされました。詳細については、38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 コネクタの概要

この章では、IBM WebSphere Business Integration Adapter for JDBC のコネクタ・コンポーネントについて説明します。WebSphere Business Integration システムでは、アプリケーションまたは (JDBC のような) テクノロジーのいずれか一方と統合ブローカーとの間の通信をサポートするためのコンポーネントを、アダプターが提供します。アダプターには、コネクタ、メッセージ・ファイル、および構成ツールが必ず含まれています。また、Object Discovery Agent (ODA) またはデータ・ハンドラーが含まれる場合もあります。

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・コンポーネント』
- 2 ページの『コネクタの動作方法』

コネクタ・コンポーネント

コネクタはアダプターのコンポーネントであり、ビジネス・オブジェクトを使用して、統合ブローカーへのイベント情報の送信 (イベント通知)、および統合ブローカーからの要求情報の受信 (要求処理) を行います。コネクタは、**コネクタ・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。

そのコードがすべてのコネクタに共通なコネクタ・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コンポーネントの間の対話を管理します。

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・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

Connector for JDBC を使用すると、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統合ブローカーとの間で、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、コネクタのアーキテクチャーの全体像について説明します。

コネクタ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、75 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。

コネクタは、JDBC Connect 機構を使用して、アプリケーション・データベースにリンクされます。コネクタ固有の構成パラメーターの 1 つ (DatabaseURL) を使用すると、コネクタの接続先のデータベース・サーバーの名前を指定することができます。構成パラメーターについては、76 ページの『コネクタの構成プロパティの設定』を参照してください。

コネクタは、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。コネクタが停止すると、プール内の全接続がクローズされます。

コネクタ・アーキテクチャ

図1に、コネクタ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。

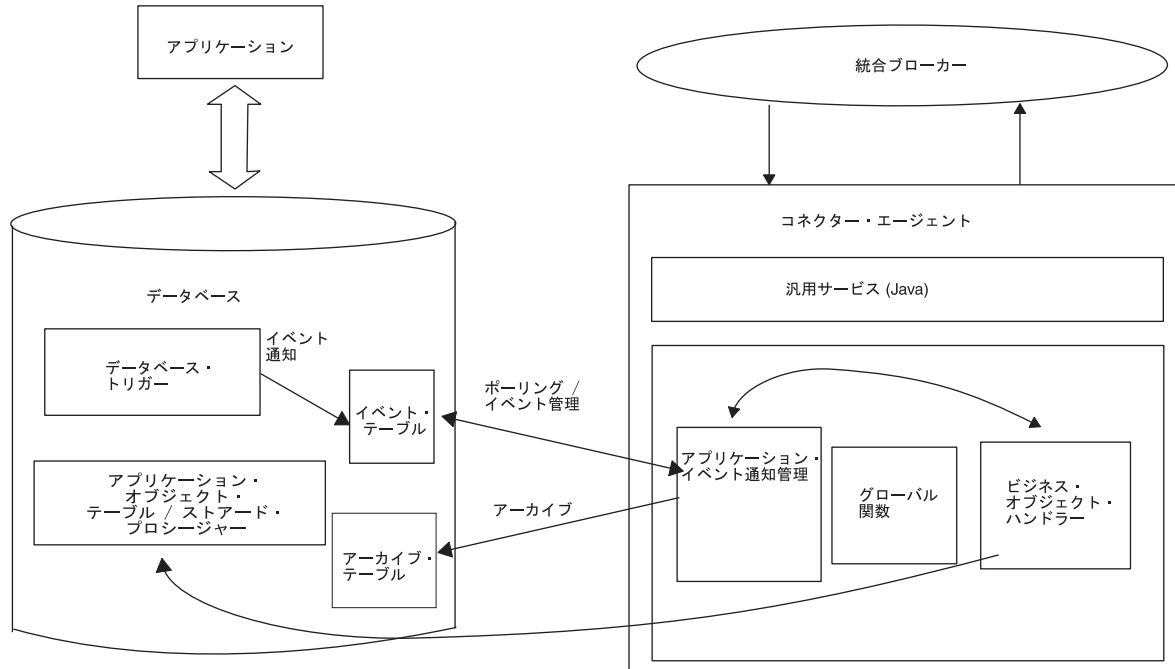


図1. ビジネス・オブジェクト要求に対するコネクタ・アーキテクチャ

コネクタの動作方法

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。

コネクタおよびメタデータ

コネクタは、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は、コネクタでハードコーディングされた命令でなくビジネス・オブジェクト定義でエンコードされたメタデータに基づいて、サポートする各ビジネス・オブジェクトを処理します。

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の設定やアプリケーシ

オン固有情報の内容が組み込まれています。コネクタはメタデータ主導型のため、コネクタのコーディングを変更しなくても、新規ビジネス・オブジェクトや変更されたビジネス・オブジェクトを処理できます。

コネクタは、SQL ステートメントまたはストアド・プロシージャを実行して、データベースやアプリケーション内のデータを検索または変更します。コネクタでは、動的 SQL ステートメントまたはストアド・プロシージャの作成のために、アプリケーション固有のメタデータが使用されます。これらの SQL ステートメントおよびストアド・プロシージャは、ビジネス・オブジェクトおよびコネクタが処理する動詞に必要な、データベースやアプリケーションに対する検索または変更を実行します。アプリケーション固有情報については、15 ページの『第 3 章 ビジネス・オブジェクトの理解』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、21 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、階層の頂点に存在して、自身は親ビジネス・オブジェクトを持たない個別ビジネス・オブジェクトを指します。

ビジネス・オブジェクトの検索: コネクタは、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. 親子関係を子に格納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. 親子関係を子に格納する各子ビジネス・オブジェクト (複数カーディナリティーと単一カーディナリティーの両方) 内の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、ご使用のサイトで重要な場合、そのフィールドの値は、必要に応じて変更することができます。

コネクタは、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。SmartFiltering は、統合ブローカーおよびコネクタ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。論理削除の場合、コネクタによって 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。

オブジェクト・イベント ID の設定: オブジェクト・イベント ID は、アプリケーション・ブローカーからのロギング重複イベントを避けるために使用される固有 ID です。例えば、あるイベントが進行中で、統合ブローカーに送信された後、アダプターに障害が起きたとします。アダプターは、再始動されると、イベントの再処理と再送信を行います。その後、統合ブローカーはイベント ID を比較して、重複イベントがある場合はそれを廃棄します。これは、各イベント ID は固有なものであるためです。

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。第 2 の方法では、ビジネス・オブジェクトは、RetrieveByContent 動詞をサポートし、オブジェクト・キーに名前と値のペアを使用する必要があります。

注: オブジェクト・キーに名前と値のペアを使用しない場合、オブジェクト・キー・フィールドのキーの順序は、ビジネス・オブジェクト内のキーの順序と同じにする必要があります。

イベント通知

コネクタ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、およびデータベース・トリガーが使用されています。潜在的な障害ポイントがイベント処理には伴うため、イベント管理プロセスは、イベントがアーカイブ表に挿入されるまで、イベント表からイベントを削除することはありません。

アダプターのインストール手順の一部としてデータベースに追加されるデータベース・トリガーは、所定のイベントがデータベースに発生すると、イベント表にデータを読み込みます。コネクタは、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。

`ArchiveProcessed` プロパティの値によって、イベントの状況の更新後に、コネクタがアーカイブ表にイベントをアーカイブするかどうかが決まります。

`ArchiveProcessed` プロパティに関する詳細については、76 ページの『コネクタの構成プロパティの設定』を参照してください。

表 1 は、`ArchiveProcessed` プロパティの値に応じたアーカイブの振る舞いを示しています。

表 1. `ArchiveProcessed` プロパティの値に基づくアーカイブの振る舞い

<code>ArchiveProcessed</code> 値	イベント表から削除される理由	コネクタの振る舞い
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・オブジェクトのサブスクリプションなし	「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 <code>Error</code> にしてイベント表に残します。
	ビジネス・オブジェクトのサブスクリプションなし	状況を <code>Unsubscribed</code> 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・トリガー内の機構です。例えば、コネクタによる前回のイベント・ポーリングの後で、`Contract` ビジネス・オブジェクトがアプリケーションによって 15 回更新された場合でも、`SmartFiltering` 機構は、これらの変更を単一の `Update` イベントとして格納します。

データベース接続の喪失

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。このコネクタが、複数のデータベースをサポートしているため、データベース接続の喪失に対して、単一のエラー・コード定義は用意されていません。

この検出を処理するため、141 ページの『PingQuery』プロパティが用意されています。サービス呼び出し要求時に障害が発生すると、コネクタは、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、135 ページの『AutoCommit』プロパティが false に設定されていると、コネクタは、データベースとの新規接続を確立しようとします。コネクタは、データベースへの新規接続の確立に成功すると処理を続行し、失敗すると、APPRESPONSETIMEOUT を戻し、結果として終了します。

コネクタが、任意のタイプのトランザクションのためにデータベースにアクセスしている間に障害が発生すると、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

コネクタは、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。

JavaTM 仮想マシン (JVM) 内のランタイム環境では、データが Unicode 文字コード・セットで表現されます。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を構成する必要があります。これらのプロパティの詳細については、113 ページの『付録 A. コネクタの標準構成プロパティ』を参照してください。

第 2 章 コネクタのインストール

この章では、IBM WebSphere Business Integration Adapter for JDBC のインストール方法を説明します。この章の内容は、次のとおりです。

- 『アダプター環境』
- 10 ページの『前提条件』
- 11 ページの『アダプターおよび関連ファイルのインストール』
- 11 ページの『インストール済みファイルの構造』

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』
- 10 ページの『アダプターのプラットフォーム』
- 10 ページの『アダプターの依存関係』
- 10 ページの『グローバリゼーション』

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。アダプター・フレームワークは、使用されている統合ブローカーに応じて、統合ブローカーがインストールされるか、アダプターがインストールされるときにインストールされます。Adapter for JDBC バージョン 2.5.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、および 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、「*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
- HP-UX11i
- 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

グローバル化

このアダプターは、2 バイト文字セット (DBCS) 対応であり、変換されています。

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 は、アダプターの機能を拡張または変更する場合に限り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。コネクタは、Java で作成されているため、Unicode を認識します。

アダプターおよび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、コネクタのシステム始動スクリプトです。これは、汎用のコネクタ・マネージャー・スクリプトから呼び出されます。Connector Configurator (統合ブローカーとしては WebSphere MQ Integrator Broker) または System Manager の「コネクタ構成 (Connector Configuration)」画面 (統合ブローカーとしては InterChange Server) から Install をクリックすると、IBM WebSphere Business Integration Adapter 用インストーラーが、このコネクタ・マネージャー・スクリプト用のカスタマイズされたラッパーを作成します。コネクタを InterChange Server 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を始動および停止してください。WebSphere MQ Integrator Broker でコネクタを使用する場合、このカスタマイズされたラッパーは、コネクタの始動のみに使用します。コネクタの停止には 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、JDBCCConnector_II_TT.txt ファイル (言語に固有なメッセージ・ファイル (II) 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	異なるストアード・プロシージャとビジネス・オブジェクトを作成する場合に使用されるサンプル・ファイルが格納されています。それらのサンプル・ファイルは、付録 C に説明されています。
/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・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)」 (InterChange Server 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	コネクタの CWJDBC.jar および start_JDBC.bat ファイルが格納されています。
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。

表 3. コネクタ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¥messages	JDBCCConnector.txt ファイルと、JDBCCConnector_II_TT.txt ファイル (言語に固有なメッセージ・ファイル (II) 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	異なるストアード・プロシージャとビジネス・オブジェクトを作成する場合に使用されるサンプル・ファイルが格納されています。それらのサンプル・ファイルは、付録 C に説明されています。
¥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。

IBM WebSphere Business Integration Adapter for JDBC 用インストーラーは、コネクタ・ファイル用のアイコンを「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、コネクタの始動を速めることができます。

コネクタ・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)」 (InterChange Server 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)

第 3 章 ビジネス・オブジェクトの理解

この章では、Connector for JDBC でのビジネス・オブジェクトの処理方法、およびコネクタ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』
- 21 ページの『ビジネス・オブジェクト動詞の処理』
- 39 ページの『ビジネス・オブジェクト属性のプロパティ』
- 42 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・オブジェクトの変更または新規のビジネス・オブジェクトのインプリメントを行う場合のガイドとして使用することができます。データベース表からのビジネス・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については、55 ページの『第 4 章 JDBC ODA 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。コネクタの実行前に、ビジネス・オブジェクトを生成しておく必要がありますが、ビジネス・オブジェクトの作成前にコネクタの構成と実行の試行を優先する場合には、149 ページの『付録 C. ビジネス・オブジェクトのサンプル』のビジネス・オブジェクトのサンプルを使用することができます。

コネクタでは、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、コネクタは、各ビジネス・オブジェクトがすべて 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各単純属性 (つまり、String、Integer、または Date などの単一値を表す属性) はデータベース表またはビュー内の列によって表されることを前提にしています。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)。ビジネス・オブジェクトの処理にとって必要な列のみを実際的设计に含めるようにします。
- 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)。データベース内に表されていない属性は、アプリケーション固有情報を持たないか、デフォルト値で設定されているか、ストアード・プロシージャを指定しているかのいずれかです。
- 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 (SP) に基づいている場合は、各単純属性 (特殊な SP 属性とは異なる属性) に、アプリケーション固有情報が存在するとは限りません。詳細については、30 ページの『ストアード・プロシージャ』を参照してください。

ビジネス・オブジェクトは、フラットまたは階層です。フラット・ビジネス・オブジェクトの属性は、すべて単純属性であり、1 つの値を表します。**階層**ビジネス・オブジェクトという用語は、あらゆるレベルの子ビジネス・オブジェクトをすべて含む、完全なビジネス・オブジェクトを指します。**個別**ビジネス・オブジェクトという用語は、1 つのビジネス・オブジェクトであり、それが含む子オブジェクトや、それを含む子ビジネス・オブジェクトから独立しているものを指します。**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。

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 1 つの子ビジネス・オブジェクトを表すときに発生します。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。

コネクタでは、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。

- 17 ページの『単一カーディナリティー関係』

- 『単一カーディナリティー関係および所有権のないデータ』
- 19 ページの『複数カーディナリティー関係』
- 19 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』
- 20 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、45 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、ParentBOName ボックス内の fk1 は、子の基本キーを含む単純属性であり、同様に、ParentBOName ボックス内にある child[1] は、子ビジネス・オブジェクトを表す属性です。

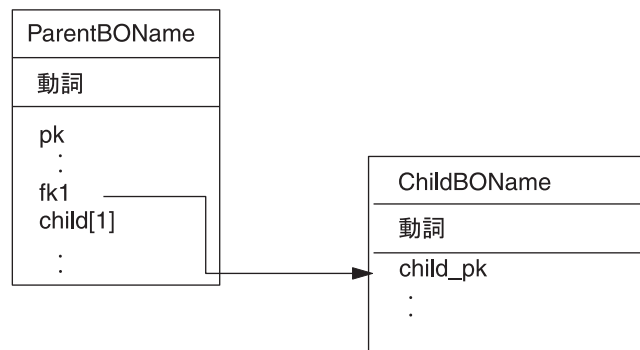


図 2. 典型的な単一カーディナリティー関係

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、各 Customer ビジネス・オブジェクトが Address ビジネス・オブジェクトを 1 つ含んでいる場合に、新しい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 1 行挿入されます。挿入された新しい住所は、その新しい顧客に固有です。同様に、Customer 表から顧客を削除すると、その顧客の住所も 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。コネクタは、それらのビジネス・オブジェクトに対しては、Retrieve 操作のみを実行します。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、51 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、47 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には、静的参照テーブルの使用を容易にするだけでなく、正規化データと非正規化データを同期するという別の機能があります。

非正規化データへの正規化データの同期化: 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・アプリケーションが、A と B という 2 つの表にデータを格納するものとします。また、非正規化されている宛先アプリケーションは、各エンティティ A が B のデータを重複して格納する形式で、1 つの表にすべてのデータを格納するものとします。

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。

正規化データへの非正規化データの同期化: 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場合、関係を設定する外部キーはそれぞれの子に格納されます。

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

関係を設定する外部キーが子に保管されるので、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、3 つの ChildBOName ボックス内の parentID は、親の基本キーを含む単純属性であり、ParentBOName ボックス内にある child[n] は、子ビジネス・オブジェクトの配列を表す属性です。

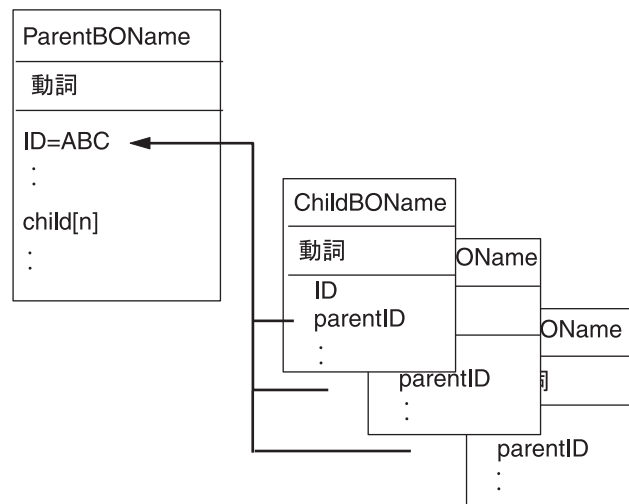


図 3. 複数カーディナリティー・ビジネス・オブジェクト関係

関係を子に格納する単一カーディナリティー関係

アプリケーションによっては、親子関係を親ではなく子に格納するように、子エンティティーを 1 つ格納するものがあります。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。

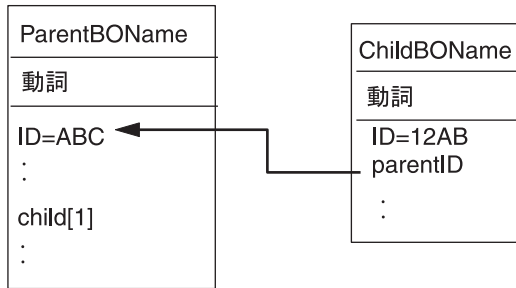


図4. 関係を子に格納している単一カーディナリティー・ビジネス・オブジェクト

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・オブジェクトが単一カーディナリティーの子を含むように指定するには、子を格納す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 パラメータを含めないようにします。詳細については、51 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、いかなる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 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・オブジェクトは、n のカーディナリティーのエンティティまたは n-1 のカーディナリティーのエンティティ、あるいはその両方を含む場合があります。

n のカーディナリティーを持つエンティティでは、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、基本キーとしてラッパー・オブジェクトに追加されます。エンティティの外部キーは、この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、コネクタが各ソース・ビジネス・オブジェクト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、コネクタが変更後イメージを扱う方法を説明します。
- 23 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 が実行するステップを説明します。
- 30 ページの『SQL ステートメント』では、コネクタが単純な SQL ステートメントを使用してビジネス・オブジェクトの選択、更新、検索、または削除を行う方法を説明します。
- 30 ページの『ストアード・プロシージャ』では、コネクタがストアード・プロシージャを使用する方法を説明します。
- 39 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

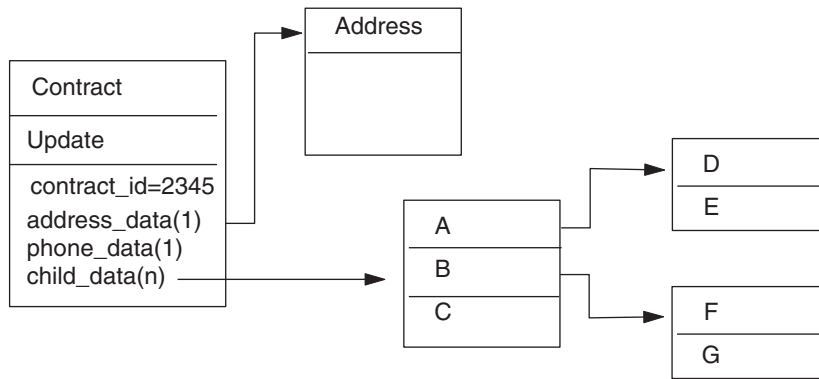
最上位ビジネス・オブジェクト、およびその子である各個別ビジネス・オブジェクトは、それ自身の動詞を格納することができます。したがって、親ビジネス・オブジェクトと子ビジネス・オブジェクトで動詞が異なるビジネス・オブジェクトが、統合ブローカーからコネクタに渡される場合があります。この場合、コネクタでは、最上位の親ビジネス・オブジェクトの動詞を参照して、ビジネス・オブジェクト全体をどのように処理するかを決定します。詳細については、23 ページの『動詞の処理』を参照してください。

変更後イメージと差分

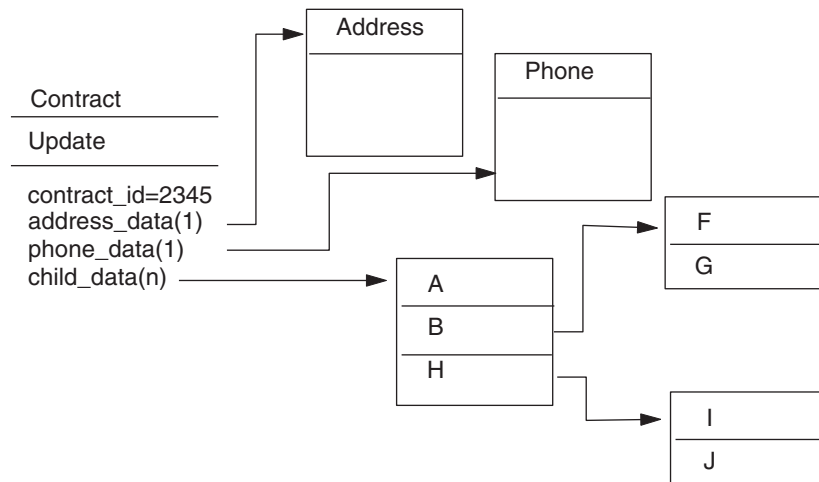
変更後イメージとは、ビジネス・オブジェクトに対するすべての変更が行われた後の、ビジネス・オブジェクトの状態です。差分とは、Update 操作で使用される、キー値および変更対象のデータのみを含むビジネス・オブジェクトのことです。コネクタは変更後イメージしかサポートしないため、更新用のビジネス・オブジェクトを受信すると、コネクタはそのビジネス・オブジェクトがデータの更新後の正しい状態を表しているものと想定します。

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、最上位に Contract と Address の両ビジネス・オブジェクト、属性に contract_id、address_data、phone_data、child_data、および A から G の子ビジネス・オブジェクトを持つ以下の図に示すようなものとします。



さらに、統合ブローカーが、以下に示すビジネス・オブジェクトをコネクタに受け渡し、次に説明する変更を引き起こすものとします。



Update 操作を処理するため、コネクタは、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 (トップレベル) 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思えます。このため、このコネクタ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、Update 操作時に欠落している子ビジネス・オブジェクトを一部のコネクタに削除させないようにする方法があります。子または子の配列を表す属性のアップ

リケーション固有情報を使用して、コネクターに対して、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に指示することができます。これは、KEEP_RELATIONSHIP を true に設定して行います。詳細については、47 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: 最上位ラッパー・ビジネス・オブジェクトは、create、retrieve、update、および delete の各動詞をサポートします。ラッパー・オブジェクト自体は処理されません。ラッパー・オブジェクトが含むオブジェクトのみが処理されます。

ビジネス・オブジェクトの比較

以下に概説するビジネス・オブジェクトの動詞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、Update 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. コネクターは、所有関係を伴う単一カーディナリティーの各子ビジネス・オブジェクトを、データベース内に再帰的に挿入します。つまり、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、コネクタはその属性を無視します。ただし、ビジネス・オブジェクト定義により、その属性が子を表すことが必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. 子がデータベース内に存在しないことを示し、Retrieve 操作が失敗した場合、コネクタはエラーを戻して処理を停止します。Retrieve 操作が成功した場合、コネクタは子ビジネス・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・オブジェクトが存在する場合に、このアプローチが正しく機能するには、子ビジネス・オブジェクト内の基本キー属性の相互参照が、Create 操作時に正しく行われる必要があります。アプリケーションのデータベースに子ビジネス・オブジェクトが存在しない場合は、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、45 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 パラメータの使用については、45 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: 最上位ビジネス・オブジェクトがラッパーである場合は、データベース内に挿入されません。

4. コネクタは、親子関係を子に格納している単一カーディナリティーの子ビジネス・オブジェクトを、すべて次のように処理します。

- 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 (目的的最上位ビジネス・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:

- a. 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。
 - b. 最上位ラッパー・ビジネス・オブジェクトは、ラッパー・オブジェクトの直下にオブジェクトのすべての属性値を含む必要があります。これらはキー属性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時に必要となります。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 レベル下にあるオブジェクト内で外部キーとして使用されるラッパー・オブジェクト内の単純属性は、ラッパー・オブジェクト内で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。

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、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。子オブジェクトに対する所有関係の有無は、処理シーケンスを決定しませんが、処理のタイプは決定します。

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、コネクタは最初の行を結果行として使用し、MULTIPLE_HITS メッセージを戻します。

注: RetrieveByContent 動詞は、最上位ラッパー・ビジネス・オブジェクトには適用できません。

Update 操作

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

注: 最上位ラッパー・ビジネス・オブジェクトは、データベースに存在する必要はありません。ただし、最上位ラッパー・ビジネス・オブジェクトは、ラッパー・オブジェクトの直下にオブジェクトの属性値を含む必要があります。これらは、キー属性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時に必要となります。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・オブジェクト内の単純属性は、ラッパー・オブジェクト内で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。コネクタは、非キーの単純属性すべての値を更新します。

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をすべてを再帰的に更新します。

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

コネクタでは、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。削除操作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、コネクタは、ソース・ビジネス・オブジェクトに存在する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。コネクタは、子の検索に成功すると、その子ビジネス・オブジェクトにデータを読み込みますが、更新は行いません。これは、所有関係にない単一カーディナリティーの子はコネクタによって変更されることがないためです。

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・オブジェクトの場合には、コネクタが、親に存在する各外部キー値を、対応する単一カーディナリティーの子ビジネス・オブジェクトの基本キー値に設定します。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、コネクタはエラーを戻します。

5. 親子関係を子に格納するそれぞれの子 (複数カーディナリティーと単一カーディナリティーの両方) の外部キー値をすべて、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。(InterChange Server を統合ブローカーとして使用する場合は、これらの値は、通常、データ・マッピング時に相互参照されます。) 関係を子に格納する新しい子の外部キー値を正しい値にした後に、コネクタにそれらの子を更新させるために、このステップは重要です。
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、47 ページの『属性の外部キーの指定』を参照してください。削除操作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。コネクタが、ソース配列に重複したビジネス・オブジェクトを受信すると、そのビジネス・オブジェクトを 2 回処理するため、予期しない結果が発生します。

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。

- DeltaUpdate 操作では、Update 動詞処理の場合とは異なり、更新の前に Retrieve 操作が実行されません。
- 着信ビジネス・オブジェクトとデータベース内のビジネス・オブジェクトの比較が行われません。
- 子はすべて、各子オブジェクトに設定されている動詞に基づいて処理されます。動詞が設定されていない子の場合、コネクタはエラーを戻します。

DeltaUpdate によるビジネス・オブジェクトの更新時、コネクタは、操作が成功した場合 (操作によりビジネス・オブジェクトが変更されたかどうかには関係なく) VALCHANGE、または操作が失敗した場合 FAIL のいずれかの状況を戻します。

コネクタは、DeltaUpdate による階層ビジネス・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. 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、DeltaUpdate 操作は失敗し、コネクタ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ド・プロシージャ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 `Ignore` に設定されている属性を除きます。コネクタでは、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、コネクタは、結果として 1 行のみが確実に処理されることをチェックします。複数の行が処理される場合、エラーが戻されます。
4. 現在のオブジェクトのカーディナリティー `N` のすべての子にある、親の属性を参照する外部キー値をすべて、対応する親の値に設定します。通常、これらの値はデータ・マッピング時にすでに相互参照されています。ただし、これはカーディナリティーが `N` のコンテナに含まれる新しい子には該当しない場合があります。このステップにより、カーディナリティーが `N` の子すべての外部キー値が正しい値になってから、それらの子の更新が行われることが徹底されます。
5. 現在のオブジェクトの、カーディナリティーが `N` のコンテナ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 操作の子で許可される動詞は、Create、Delete、および DeltaUpdate です。

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

Delete 操作

コネクタは、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。親ビジネス・オブジェクトの検索後、アダプターは、検索された親と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、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、コネクタは FAIL を戻します。

コネクタ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、コネクタは、論理的削除操作を実行します。SCN 値が定義されていない場合、コネクタは、物理的削除操作を実行します。

物理的削除操作: コネクタでは、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: 最上位ラッパー・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ることはありません。ラッパーの単純属性値はすべて無視されます。

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

1. Update 操作を発行して、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よって指定されている値に設定します。コネクタでは、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 プロパティから取得されます。各照会は複数の表にまたがることはできません。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

ストアード・プロシージャは、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・プロシージャは、コネクタ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。

コネクタは、次の目的でストアード・プロシージャを呼び出します。

- 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

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・プロシージャの構文と例

このセクションでは、ビジネス・オブジェクトのストアード・プロシージャの構文について説明します。また、例も記載しています。この章の内容は、次のとおりです。

- 『ビジネス・オブジェクトへの属性の追加』
- 32 ページの『ストアード・プロシージャの構文』
- 33 ページの『結果セットを戻さないストアード・プロシージャの例』
- 33 ページの『ストアード・プロシージャの指定』

ビジネス・オブジェクトへの属性の追加: コネクターが処理するストアード・プロシージャのタイプごとに、ビジネス・オブジェクトに特種な属性を追加する必要があります。この属性は、ストアード・プロシージャのタイプと、ストアード・プロシージャ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。

属性は、使用されるストアード・プロシージャのタイプに従って命名されています。例えば、AfterUpdate および BeforeRetrieve ストアード・プロシージャをコネクターが使用するようにセットアップする場合は、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
BeforeDeltaUpdateSP
AfterDeltaUpdateSP
DeltaUpdateSP
```

注: コネクターに実行させるストアード・プロシージャについてのみ、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 (InterChange Server 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。コネクターを再始動して、ビジネス・オブジェクトに対する以降の呼び出し時に、コネクターがそれらの属性値に対する変更を認識するようにします。

ストアード・プロシージャの構文: ストアード・プロシージャ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。

<i>StoredProcedureName</i>	ストアード・プロシージャの名前。
RS	ストアード・プロシージャが結果セットを戻す場合は true、それ以外の場合は false。デフォルト値は 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 <i>ColumnName</i> プロパティは、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・プロシージャに対してサポートされています。データベース内のストアード・プロシージャの場合、対応するパラメーターは REF CURSOR 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・プロシージャの実行時にその値を入力値として使用するビジネス・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・プロシージャの実行後に値を戻す先のビジネス・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: コネクターがその値を入力値として使用し、ストアード・プロシージャの実行後に値を戻す先のビジネス・オブジェクト属性のリスト。

プロパティ *ReturnDummyBOForSP* は、結果セットが true でありながら、空の場合にも出力パラメーターを戻します。RetrieveSP の場合、結果のセットが戻されず。結果のセットが空の場合は、ビジネス・オブジェクトが生成されず、プロシージャ呼び出しの戻す出力パラメーターをリトリートする方法も存在しません。*ReturnDummyBOForSP* が true の場合は、対応する属性に読み込まれた出力パラメーターと入出力パラメーターの値を持つダミーのビジネス・オブジェクトが戻されます。このプロパティのデフォルト値は false です。

StoredProcedureName、RS、およびパラメーターの順序は重要ですが、パラメーター同士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グループ化されている場合は、それらの値はコロンで区切られ、値ごとにパラメーターの名前は繰り返されません。異なるタイプのパラメーターは、セミコロンで区切られます。パラメーター値の等号 (=) の両側には、スペースを含めません。

結果セットを戻すストアード・プロシージャの例: 以下に示す例は、ストアード・プロシージャ `CustomerAddressRetrieve` および

`CustomerAddressRetrieveForOracleDB` を使用して、複数のアドレスが含まれ、カーディナリティーが N の子ビジネス・オブジェクトの作成に使用される結果セットを戻します。

注: 結果セットは、属性 `RetrieveSP` が指定されている場合に限り処理され、カーディナリティーが N の子ビジネス・オブジェクトの作成に使用されます。

データベースが Oracle の場合、結果セットは出力パラメーターとして戻され、アダプターで処理されます。それ以外のデータベースの場合、結果セットはストアード・プロシージャの戻り値です。

- `CustomerAddressRetrieve` (Oracle 以外のデータベース用)

```
Attribute : RetrieveSP
ASI : SPN=CustomerAddressRetrieve;RS=true;
IP=CustomerName;IP=CustomerId;
OP=ErrorStatus;OP=ErrorMsg
```

- `CustomerAddressRetrieveForOracleDB` (Oracle データベース用)

```
Attribute : RetrieveSP
ASI : SPN=CustomerAddressRetrieveForOracleDB;
RS=true;IP=CustomerName;IP=CustomerId;
OP=RS;OP=ErrorStatus;OP=ErrorMsg
```

(`OP=RS` は、最初の出力パラメーターが結果セット用であることを示しています。)

結果セットを戻さないストアード・プロシージャの例: 以下の例では、`CustomerInsert` および `VendorInsert` というストアード・プロシージャを示します。これらのストアード・プロシージャは、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・プロシージャの構造が異なります。

- 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;IP=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・プロシージャの指定: ストアード・プロシージャ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

ストアード・プロシージャ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 プロパティにその値を指定できます。このプロパティを使用すると、コネクタがビジネス・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・プロシージャの指定を示します。この場合、MaxLength プロパティに指定されている値は、ストアード・プロシージャ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 (InterChange Server 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。マッピングは、ストアード・プロシージャを指定するために使用しますが、その使用は、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です。つまり、コネクタがイベントをポーリングしている場合は、ストアード・プロシージャの指定に属性の値を使用することはできません。

ストアード・プロシージャ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・プロシージャの指定にマッピングが使用される場合は、テキスト全体が収まるよう、MaxLength プロパティの値を大きくする必要があります。

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、コネクタは、BeforeCreate ストアード・プロシージャを実行する場合、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・プロシージャを処理する場合、コネクタは、1 つのビジネス・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、コネクタによるストアード・プロシージャと単純な SQL ステートメント処理方法について説明します。

- 35 ページの『ビジネス・オブジェクトの Create 操作』

- 『ビジネス・オブジェクトの Update 操作』
- 36 ページの『ビジネス・オブジェクトの Delete 操作』
- 36 ページの『ビジネス・オブジェクトの Retrieve 操作』
- 37 ページの『Retrieve ストアード・プロシージャ (RetrieveSP) 操作から戻された結果セット』
- 38 ページの『ビジネス・オブジェクトの RetrieveByContent 操作』
- 38 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・プロシージャは、通常、コネクタが最上位ビジネス・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。コネクタは、Create ストアード・プロシージャ (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・プロシージャを呼び出します。
2. ストアード・プロシージャが出力パラメーターを介して値を戻す場合は、その値を使用してビジネス・オブジェクトの単純属性の値を設定します。
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。トップレベル・ビジネス・オブジェクトの外部キー値それぞれを、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セットします。
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・プロシージャが出力パラメーターを通じて値を戻した場合、その値をビジネス・オブジェクトの単純属性にセットします。
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・プロシージャを呼び出します。
10. ストアード・プロシージャが出力パラメーターを介して値を戻す場合は、その値を使用してビジネス・オブジェクトの単純属性の値の設定します。

コネクタはステップ 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・プロシージャは、通常、コネクタが最上位ビジネス・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。コネクタは、Update ストアード・プロシージャ (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・プロシージャを呼び出します。
2. BeforeUpdate ストアード・プロシージャ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・プロシージャが出力パラメーターを通じて値を戻した場合、その値をビジネス・オブジェクトの単純属性にセットします。
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・プロシージャを呼び出します。
10. ストアード・プロシージャが出力パラメーターを介して値を戻す場合は、その値を使用してビジネス・オブジェクトの単純属性の値を設定します。

ビジネス・オブジェクトの Delete 操作: Delete ストアード・プロシージャは、コネクタに値を戻しません。コネクタは、Delete ストアード・プロシージャ (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・プロシージャを呼び出します。

ビジネス・オブジェクトの Retrieve 操作: 単純な Retrieve 操作の場合は、最上位ビジネス・オブジェクト、単一カーディナリティーの子、および複数カーディナリティーの子に対して、ストアード・プロシージャを使用することができます。ストアード・プロシージャの順序は、次のとおりです。

- BeforeRetrieve

- Retrieve
- AfterRetrieve

コネクタは、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時に、一時オブジェクトを作成します。コネクタは、BeforeRetrieve ストアード・プロシージャを一時ビジネス・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・プロシージャが適用されます。

コネクタにより AfterRetrieve ストアード・プロシージャが実行されるのは、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・プロシージャが、ビジネス・オブジェクトに対して実行された後です。

JDBC 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>(?,?,?)}

ここで、spName は、ストアード・プロシージャの名前です。

コネクタでは、最初の 2 つのタイプがサポートされています。コネクタは、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 に対して提供されています。

Retrieve ストアード・プロシージャ (RetrieveSP) 操作から戻された結果セット

: Retrieve ストアード・プロシージャ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の場合、対応する結果セットは 1 行のみで構成される必要があります。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の場合、結果セットを介して複数の行が戻されることがあります。戻された行ごとに新しいオブジェクトが作成され、コンテナに追加されます。このコンテナは、その後親オブジェクトの必須属性索引に追加されません。

ラッパー・ビジネス・オブジェクトの n カーディナリティーの子は、ストアード・プロシージャ属性を持ち、それらの属性は、入力パラメーターと結果セット列を表します。WRAPPER=true は、ビジネス・オブジェクトのアプリケーション固有情報レベルで設定されます。子ビジネス・オブジェクトのアプリケーション固有情報は、TN=dummy を持ちます。

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・プロシージャを使用することはできません。ストアード・プロシージャの順序は、次のとおりです。

- BeforeRetrieveByContent
- RetrieveByContent
- AfterRetrieveByContent

コネクタは、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時に、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・プロシージャが一時ビジネス・オブジェクトに適用されます。コンテナ用に検索された各子オブジェクトには、AfterRetrieveByContent ストアード・プロシージャが適用されます。

AfterRetrieveByContent ストアード・プロシージャが実行されるのは、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 操作と同じ方法で、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・プロシージャは、BeforeRetrieve および AfterRetrieve と同じ操作を実行します。コネクタに BeforeRetrieve 操作と

BeforeRetrieveUpdate 操作、および AfterRetrieve 操作と AfterRetrieveUpdate 操作を実行させる属性に対応するために、それらには、異なる名前が付けられています。

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の終了後、処理が成功するとトランザクション・ブロックをコミットし、エラーが発生するとトランザクションをロールバックします。

ビジネス・オブジェクト属性のプロパティー

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、コネクターが解釈する方法とその設定値について説明します。

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各ビジネス・オブジェクト属性は、この属性に対して、それぞれ、値 1 または n を持ちま

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 (子のバージョン番号を指定) と `Relationship` プロパティ (値の `Containment` を指定) を持ちます。

Max length プロパティ

`String` 型の属性では、このプロパティにより、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

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。値 `Yes` は、属性をキーとして定義します。ビジネス・オブジェクト属性の型が `String` の場合は、データベースでのデータ型は、`char` 型ではなく、`Varchar` 型にします。

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と単一カーディナリティーの両方) 内の属性に対して、キー・プロパティが `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、`Key` プロパティは使用しません。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、44 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

コネクターでは、このプロパティを使用して、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

`Required` プロパティは、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、コネクターは、その親ビジネス・オブジェクトが、この属性に対する子ビジネス・オブジェクトを含むことを要求します。

コネクターが、`Create` 要求を伴うビジネス・オブジェクトを受信したときに、以下の条件の両方が `true` であると、`Create` 操作は失敗します。

- 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。

コネクターが、Retrieve 要求を伴うビジネス・オブジェクトを受信したときに、そのビジネス・オブジェクトが必須属性に対して有効な値またはデフォルト値を持っていないと、検索操作は失敗します。

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、44 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、133 ページの表 15、および 146 ページの『UseDefaults』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は特殊値 CxIgnore を持つことがあります。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 値を表します。

コネクターの要件として、すべてのビジネス・オブジェクトが基本キー属性を最低でも 1 つ持つ必要があるため、コネクターに渡されるビジネス・オブジェクトには、CxIgnore に設定されていない基本キーを少なくとも 1 つ持つ必要があります。この要件に対するただ一つの例外は、その基本キーが、コネクターによってカウンターまたはシーケンスで生成されたビジネス・オブジェクトまたはデータベースによって生成されたビジネス・オブジェクトです。

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、45 ページの表 5 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

このセクションでは、コネクタ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。

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。

拡張または変更されたアプリケーション固有のビジネス・オブジェクトでは、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、コネクタの予期する構文に一致する必要があります。

注: InterChange Server 環境におけるパフォーマンスは、アプリケーション固有ビジネス・オブジェクトが 1 MB 未満のときに最高になります。いかなる場合でも 5 MB を超えないようにします。ビジネス・オブジェクトを大きくすると、InterChange Server が実行されている Java 仮想マシンに対する制限のため、パフォーマンス上の問題が発生します。

表 4 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 4. 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	<ul style="list-style-type: none">• 対応するデータベース表の名前を指定します• コネクタが、WHERE 文節内でその値を使用して論理的 (ソフト) 削除を実行するための列を定義します• 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 4. 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	<ul style="list-style-type: none"> 属性のデータベース列名を指定します 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を指定します 固有 ID 値の自動生成を指定します コネクタが現在の属性の値を設定するために、その値を使用する必要のある同一ビジネス・オブジェクト内の別の属性の名前を指定します 検索時にデータの順序付けを行う場合に現在の属性を使用するかどうかを指定します 現在の属性の値が null の場合に使用する値を指定します ストリング置換の振る舞いを指定します ストリングの比較時に LIKE 演算子または = 演算子のいずれを使用するかを指定します 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を指定します
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含む属性	<p>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、および Update 操作時に、子データがソース・ビジネス・オブジェクト内で表されていない場合に、コネクタがその子データを削除するかどうかを指定します</p>
ビジネス・オブジェクト動詞	<p>動詞 Retrieve に対してのみ使用されます。このテキストは、検索時に WHERE 文節に組み込む属性を指定します。また、演算子および属性値も指定します</p>

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は、次の作業に使用されます。

- 対応するデータベース表の名前を指定します。
- 物理的削除または論理的削除の操作の実行に必要な情報を提供します。
- 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します。

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; WRAPPER=true|false
```

ここで、TableName はデータベース表を示し、StatusColumnName は論理的削除操作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されていることを示す値であり、true|false は、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かどうかを示します。

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、コネクタで、カスタマー削除要求を受信したとします。このような要求により、コネクタは、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・オブジェクトが、そのアプリケーション固有情報に SCN パラメーターを含む場合、コネクタは論理的削除操作を実行します。Delete 動詞を伴うビジネス・オブジェクトが、そのアプリケーション固有情報に SCN パラメーターを含まない場合、コネクタは物理的削除操作を実行します。

更新と削除の両操作では、SCN プロパティの値を以下のように使用できます。

- Update 操作を実行するとき、コネクタは ChildUpdatePhyDelete プロパティの値を使用して、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。欠落している子データを論理的に削除するため、コネクタは、その SCN パラメーターの値を使用して、状況列の名前と状況値のテキストを取得します。詳細については、26 ページの『Update 操作』を参照してください。
- 削除操作を実行するとき、コネクタは、その SCN パラメーターの値を使用して、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判断します。SCN パラメーターに値が含まれている場合、コネクタは、論理的削除操作を実行します。SCN パラメーターに値が含まれていない場合、コネクタは、物理的削除操作を実行します。詳細については、29 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として使用されます。コネクタ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・オブジェクトは、n のカーディナリティーのエンティティーまたは n-1 のカーディナリティーのエンティティー、あるいはその両方を含む場合があります。

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。子を表す属性のアプリケーション固有情報は、親子関係が子に格納され

るか、親に格納されるかによっても異なります。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、47 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、それぞれがパラメーター名とその値を含む、複数の名前と値のペアのパラメーターで構成されています。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]
[TS=true|false]]
```

コネクターが処理する単純属性に必須のパラメーターは、列名のみです。例えば、以下は、列名のみを指定する形式です。

```
CN=customer_id
```

表 5 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 5. 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 FK=[fk_object_name.]fk_attribute_name	このパラメーターの値は、この属性に対するデータベース列の名前です。 このパラメーターの値は、親子関係が親ビジネス・オブジェクトに格納されるか、子ビジネス・オブジェクトに格納されるかによって異なります。属性が外部キーでない場合は、アプリケーション固有情報にこのパラメーターを含めないでください。詳細については、47 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 が必要とされない場合は、このパラメーターをアプリケーション固有情報に含めないでください。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 プロパティ記述を参照してください。詳細については、50 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ブレースホルダー属性です。単純属性は、そのアプリケーション固有情報がブランクまたはヌルの場合も、ブレースホルダーとなります。

表 5. 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= <i>set_attr_name</i> <i>..set_attr_name</i>	<p><i>set_attr_name</i> が、現在の個別ビジネス・オブジェクト内部の別の属性の名前に設定されている場合、コネクターは Create 操作時に、その別の属性の値を使用してこの属性値を設定してから、ビジネス・オブジェクトをデータベースに追加します。<i>set_attr_name</i> の値は、子ビジネス・オブジェクト内の属性を参照できませんが、<i>set_attr_name</i> の前にピリオドが 2 つある場合は親ビジネス・オブジェクト内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、別の属性から CA 値をコピーしないで、現在の属性の値を使用します。</p>
OB=[ASC DESC]	<p>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 (ASC) または降順 (DESC) で検索することができます。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、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。</p>
UNVL= <i>value</i>	<p>このパラメーターは、null 値の属性を含むビジネス・オブジェクトを検索した場合に、コネクターが、null 表現用に使用する値を指定します。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。</p>
ESC=[true false]	<p>このパラメーターは、コネクターが、ReplaceAllStr プロパティで示された各文字のすべてのインスタンスを、ReplaceStrList プロパティ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 プロパティの値を使用して決定します。</p>
FIXEDCHAR=true false	<p>注: ESC パラメーターおよび ReplaceAllStr と ReplaceStrList の両プロパティにより、データベース・エスケープ文字機能 (例えば、単一引用符をエスケープするなど) がサポートされます。</p> <p>このパラメーターは、表内の列が VARCHAR 型ではなく CHAR 型である場合に、属性を固定長とするかどうかを指定します。例えば、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 の長さとも一致するようにしてください。デフォルトでは FIXEDCHAR=false です。</p>
BYTEARRAY=true false	<p>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 InterChange Server 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、52 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。</p>
USE_LIKE=true false	<p>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せん。</p>

表 5. 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
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。 <ul style="list-style-type: none"> • 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。 • CLOB に関する他の属性の型をアプリケーション固有情報 (ASI) 内で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。 <p>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。</p>
TS=true false	型 DATE の属性の場合、その属性のアプリケーション固有情報内で TS=false を指定すると、属性は、DATE 型として処理されます。TS=true の場合、属性は TIMESTAMP 型として処理されます。TS のデフォルト値は true です。

注: ビジネス・オブジェクトのどの属性にも、コネクタ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: このプロパティの値は、親子関係が親ビジネス・オブジェクトに格納されるか、子ビジネス・オブジェクトに格納されるかによって異なります。

- 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・オブジェクトが同一の子ビジネス・オブジェクトを含むことができます。その場合に、その子ビジネス・オブジェクトが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で格納されているか、および親子関係が親または子に格納されているかは関係ありません。ただし、親子関係を格納する親ビジネス・オブジェクトはすべて、同じ名前の属性を使用して、子の基本キーを格納する必要があります。さらに、親子関係を格納する子ビジネス・オブジェクトはすべて、同じ名前の属性を使用して、親の基本キーを格納する必要があります。図 5 にこれらの関係を示します。

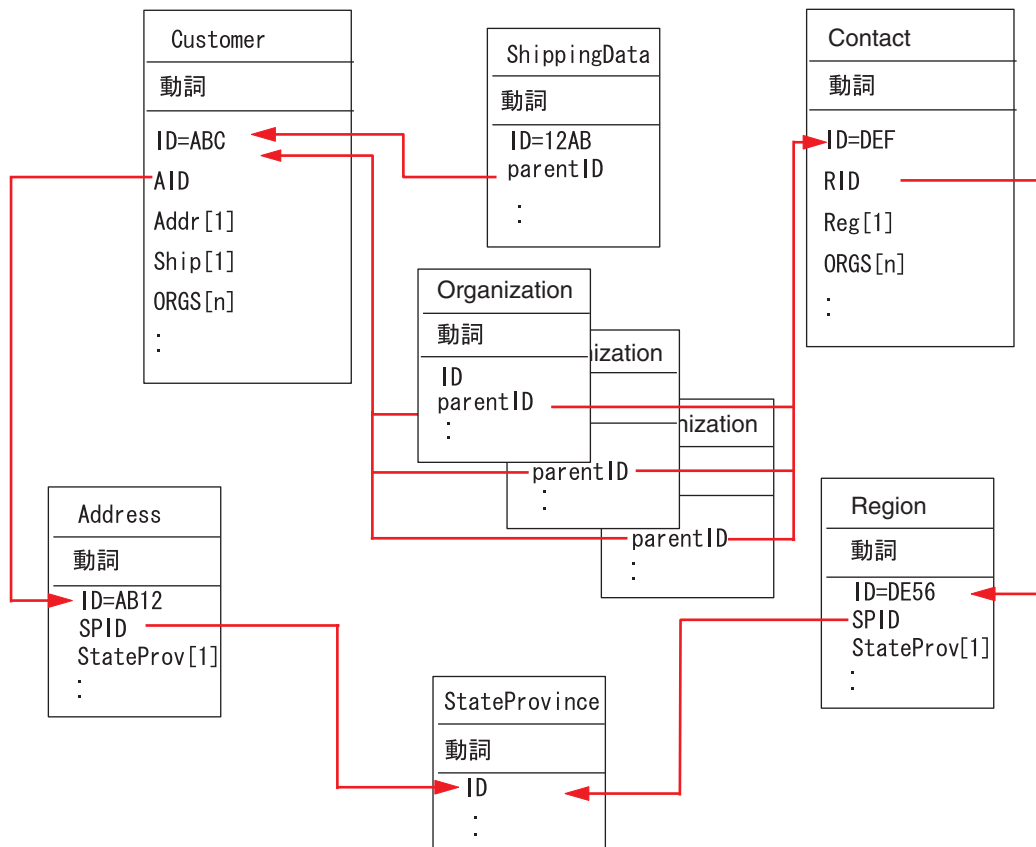


図 5. ビジネス・オブジェクト間の関係の例

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

エ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、51 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: 複数のビジネス・オブジェクトに同一の子を含めるためには (親子関係が子に格納されている場合)、親ビジネス・オブジェクトはすべて、同一の名前の属性を使用して、子の外部キーを格納する必要があります。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、所有関係のない 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、51 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: (親子関係を親に格納している) 複数のビジネス・オブジェクトが同一の子を含むためには、子ビジネス・オブジェクトはすべて、同一の名前の属性を使用して、親の外部キーを格納する必要があります。

- 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、親が作成された後に限り作成されるものであるため、親子関係は子に格納されます。

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、シーケンス (DB2 と Oracle の処理の場合) またはカウンター (表構造) 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が使用する DB2 または Oracle シーケンスの名前を示します。コネクタは、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 DB2 および Oracle データベース用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 パラメーターには空白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、DB2 および 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、大文字小文字の区別がなく、空白を含めることもできません。

処理中の固有 ID シーケンスを保持する方法については、141 ページの『PreserveUIDSeq』プロパティ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有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、17 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 19 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: 親子関係を子に格納するビジネス・オブジェクトの配列に対する Update 操作の場合は、その子を表す属性に対する特別な値が存在します。KEEP_RELATIONSHIP を true に設定すると、ソース・ビジネス・オブジェクト内に表されていない既存の子データを、コネクタに削除させないようにできます。

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 システムではバイナリー・データがサポートされていないため、バイナリー・データは、String に変換された後、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。

<i>condition_variable</i>	データベース列の名前。
<i>conditonal_operator</i>	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 (<i>value1,value2</i>)).
@	getAttrValue(<i>attribute_name</i>) 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 <i>attribute_name</i> 変数 の値で最初の @ を置換します。
..	<i>attribute_name</i> 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。
<i>attribute_name</i>	@ 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を考えてみます。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 in は子ビジネス・オブジェクトの ID であり、placeholder は親に存在する属性です。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:..WHEREx 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 JDBC ODA 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、Object Discovery Agent (ODA) の JDBC ODA を使用して Connector for JDBC のビジネス・オブジェクト定義を生成する方法について説明します。JDBC ODA は、データベース表、ビュー、ストアド・プロシージャ、および同義語/ニックネームを使用して、その JDBC データ・ソースに特定のビジネス・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。

- 『JDBC ODA のインストールと実行』
- 59 ページの『Business Object Designer 内での JDBC ODA の使用』
- 69 ページの『生成される定義の内容』
- 72 ページの『ビジネス・オブジェクト定義ファイルのサンプル』
- 73 ページの『子ビジネス・オブジェクトを含む属性の挿入』
- 73 ページの『ビジネス・オブジェクト定義への情報の追加』

JDBC ODA のインストールと実行

このセクションでは、以下について説明します。

- 『JDBC ODA のインストール』
- 56 ページの『JDBC ODA を使用する前に』
- 57 ページの『JDBC ODA の起動』
- 57 ページの『JDBC ODA のインスタンスの複数実行』
- 58 ページの『エラーおよびトレース・メッセージ・ファイルの処理』

JDBC ODA のインストール

JDBC ODA をインストールするには、IBM WebSphere Business Integration Adapter for JDBC 用の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「システム・インストール・ガイド (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_11_11.txt (言語 (11) および国または地域 (11) 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。

JDBC ODA を使用する前に

JDBC ODA を使用するには、以下を実行する必要があります。

1. 適切な JDBC ドライバーをインストールします。「システム・インストール・ガイド (UNIX 版)」または「システム・インストール・ガイド (Windows 版)」に記載されている手順に従ってください。

重要: JDBC ODA は、JDBC 2.0 以上をサポートする JDBC ドライバーを使用して、任意のデータベースに接続することができます。

2. JDBC ODA 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。
 - JDBC ODA 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。
3. UNIX シェルまたは Windows バッチ・ファイルを編集するために開き、表 6 に説明されている値を設定します。

表 6. 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

表 6. 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
DRIVERPATH	JDBC ドライバー・ライブラリーのパス。JDBC ODA 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。	<p>UNIX: DRIVERPATH=\$CROSSWORLDS/lib/ ¥ xwutil.jar:\$CROSSWORLDS/lib/ ¥ xwbase.jar:\$CROSSWORLDS/lib/ ¥ xwsqserver.jar:\$CROSSWORLDS/lib/ ¥ spy/lib/spy.jar</p> <p>Windows: set DRIVERPATH=%CROSSWORLDS%¥ / lib¥xwutil.jar;%CROSSWORLDS%¥lib¥ / xwbase.jar;%CROSSWORLDS%¥lib¥ / xwsqserver.jar;%CROSSWORLDS%¥lib¥ / spy¥lib¥spy.jar</p>
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。	<p>UNIX: DRIVERLIB=\$CROSSWORLDS/lib/db2jdbc.so</p> <p>Windows: DRIVERLIB=%CROSSWORLDS%¥bin¥db2jdbc.dll</p>

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。

JDBC ODA の起動

JDBC ODA 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
```

JDBC ODA 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。このコネクタのデフォルト ODA の名前は、JDBCODA です。

JDBC ODA のインスタンスの複数実行

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 JDBC ODA 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。

60 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。

AgentNameAgent.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、60 ページの『初期化プロパティの構成』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。

- JDBC ODA 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7 にこれらの値を示します。

表7. 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	<ul style="list-style-type: none">• ODA のすべてのプロパティの初期化値を示します。• JDBC ODA が作成したスレッドのそれぞれの詳細な状況をトレースします。• ビジネス・オブジェクト定義の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、60 ページの『初期化プロパティの構成』を参照してください。

Business Object Designer 内での JDBC ODA の使用

このセクションでは、Business Object Designer で JDBC ODA 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。本書は、IBM WebSphere Business Integration Adapters InfoCenter: <http://www.ibm.com/websphere/integration/wbiadapters/infocenter> からダウンロードできます。

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。60 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. 60 ページの『ODA の選択』
2. 60 ページの『初期化プロパティの構成』
3. 62 ページの『ノードの展開とデータベース・オブジェクトの選択』
4. 64 ページの『データベース・オブジェクト選択の確認』
5. 65 ページの『定義の生成』およびオプションで 65 ページの『追加情報の入力』
6. 69 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。

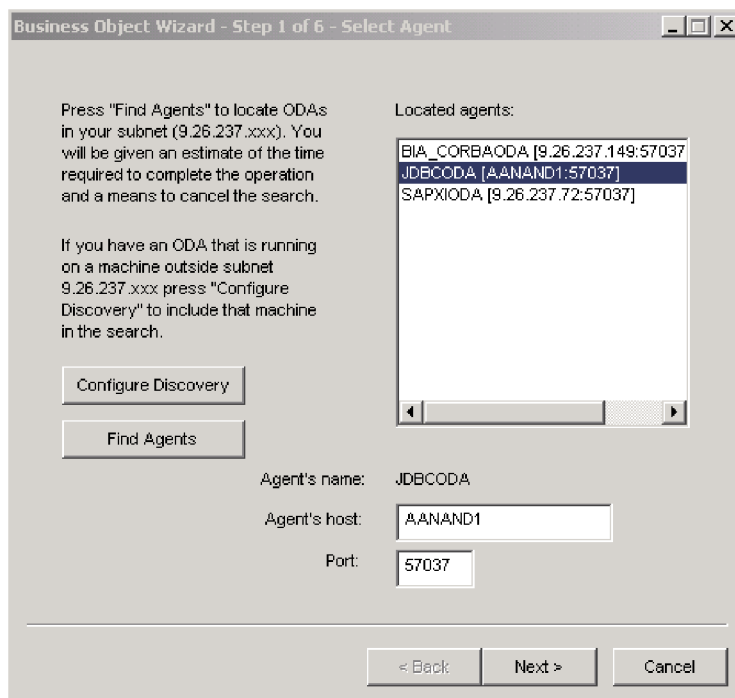


図 6. ODA の選択

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 を見つけれられない場合には、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。

3. 「次へ」をクリックします。

初期化プロパティの構成

Business Object Designer は、指定された JDBC ODA と初めてやり取りするときに、図 7 に示す一連の初期化プロパティの入力をユーザーに求めます。これらのプロパティは、その JDBC ODA を使用するたびに再入力しなくても済むように、名前付きプロファイルに保管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。本書は、IBM WebSphere Business Integration Adapters InfoCenter からダウンロードできます。

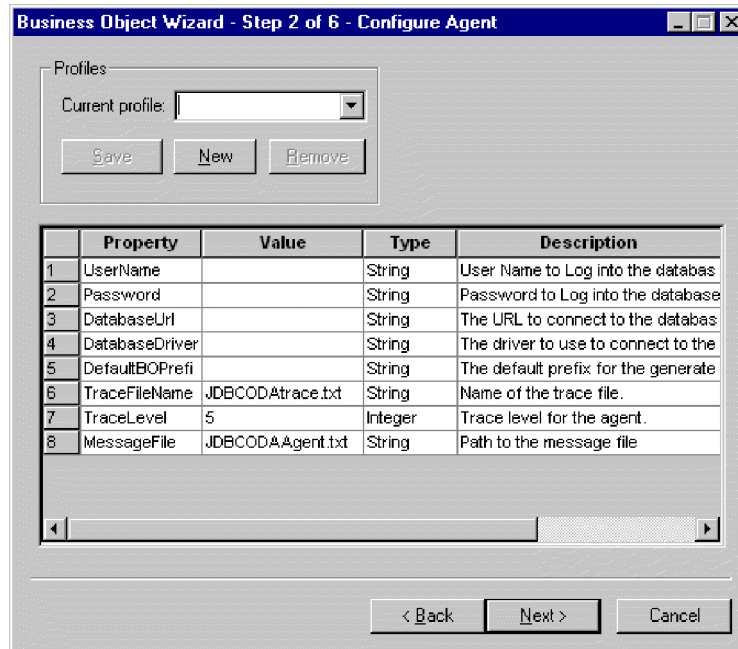


図 7. エージェント初期化プロパティの構成

JDBC ODA プロパティの構成を表 8 に示します。

表 8. JDBC ODA プロパティ

行番号	プロパティ名	プロパティ・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:@MACHINE:1521:SIDNAME
4	DatabaseDriver	String	接続の確立に使用されるドライバの名前。例: oracle.jdbc.driver.OracleDriver
5	DefaultBOPrefi	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、65 ページの『追加情報の入力』を参照してください。
6	TraceFileName	String	JDBC ODA 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、JDBC ODA は %ODA%\JDBC ディレクトリにそのファイルを作成します。既に存在している場合、JDBC ODA はそのファイルに追記します。JDBC ODA 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、JDBCODAttrace.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	JDBC ODA で使用可能なトレースのレベル。59 ページの表 7 を参照してください。

表 8. JDBC ODA プロパティ (続き)

行番号	プロパティ名	プロパティ・ 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。JDBC ODA 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。

ノードの展開とデータベース・オブジェクトの選択

JDBC ODA のすべての初期化プロパティ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。クリックすると、各スキーマに含まれる表、ビュー、ストアド・プロシージャ、および同義語/ニックネームのすべてが表示されます。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。

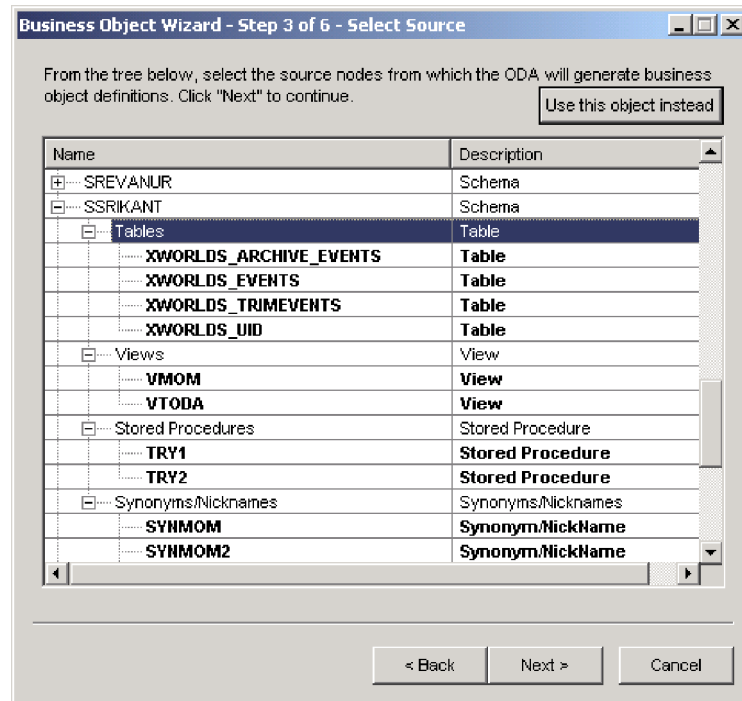


図 8. ノードが展開されたスキーマ・ツリー

ビジネス・オブジェクト定義の生成に必要なデータが格納されているデータベース・オブジェクトをすべて特定するため、必要な表、ビュー、ストアド・プロシージャ、および同義語/ニックネームをすべて選択し、「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。本書は、IBM WebSphere Business Integration Adapters InfoCenter からダウンロードできます。

スキーマ名 ALL SCHEMAS は、スキーマが関連付けられていないオブジェクト (表やビューなど) があるデータベースのオブジェクトの検索を容易にするために使用されています。「ALL SCHEMAS」を展開すると、表、ビュー、ストアド・プロシージャ、および同義語/ニックネームを含むツリーが表示されます。これらのノードのいずれかを展開すると、そのノードに分類されるデータベース・オブジェクトが、所属するスキーマに関係なくすべて表示されます。

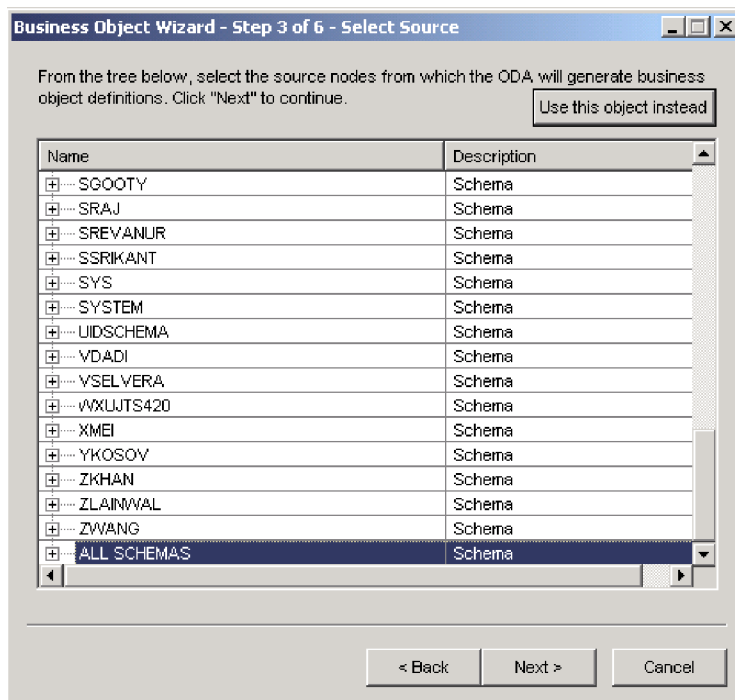


図9. オブジェクトを検索するための ALL SCHEMAS の展開

データベース・オブジェクト選択の確認

生成するビジネス・オブジェクト定義に関連付けるデータベース・オブジェクトの特定を完了すると、Business Object Designer に、選択された表、ビュー、ストアド・プロシージャ、および同義語/ニックネームのみを示すダイアログ・ボックスが表示されます。図 10 にこのダイアログ・ボックスを示します。

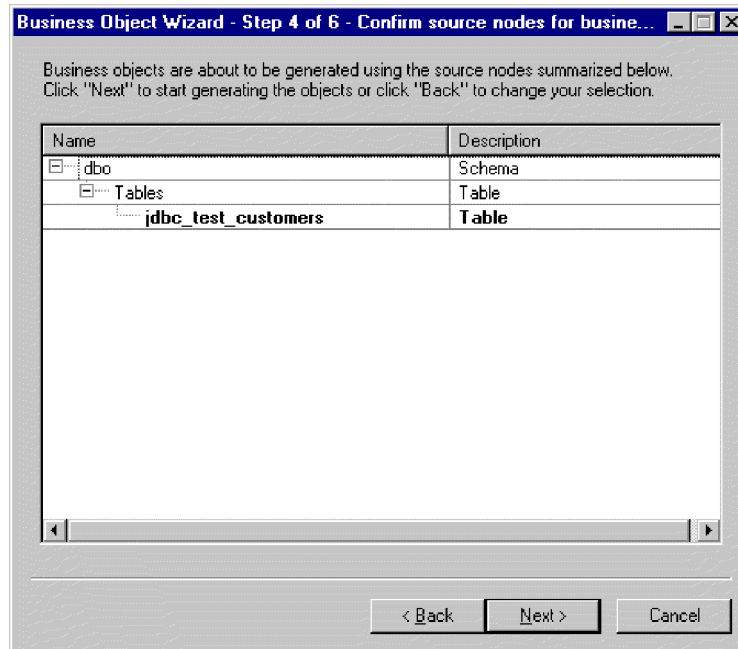


図 10. データベース・オブジェクトの選択の確認

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

データベース・オブジェクトを確認すると、Business Object Designer が定義を生成していることを通知するダイアログ・ボックスが表示されます。

追加情報の入力

JDBC ODA が追加情報を必要とする場合は、Business Object Designer が、プロンプトを出してユーザーに情報の入力を要求する、66 ページの図 11 に示す「BO プロパティ」ウィンドウを表示します。

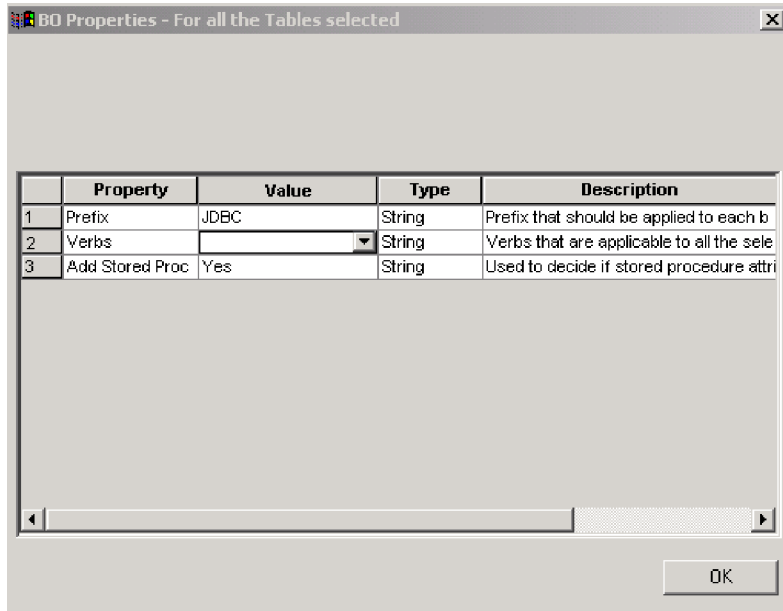


図 11. データベース・オブジェクトに関する追加情報の指定

「BO プロパティ」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、JDBC ODA が、67 ページの図 12 に示すような、すべてのストアード・プロシージャの属性のリストを示すウィンドウを表示します。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。

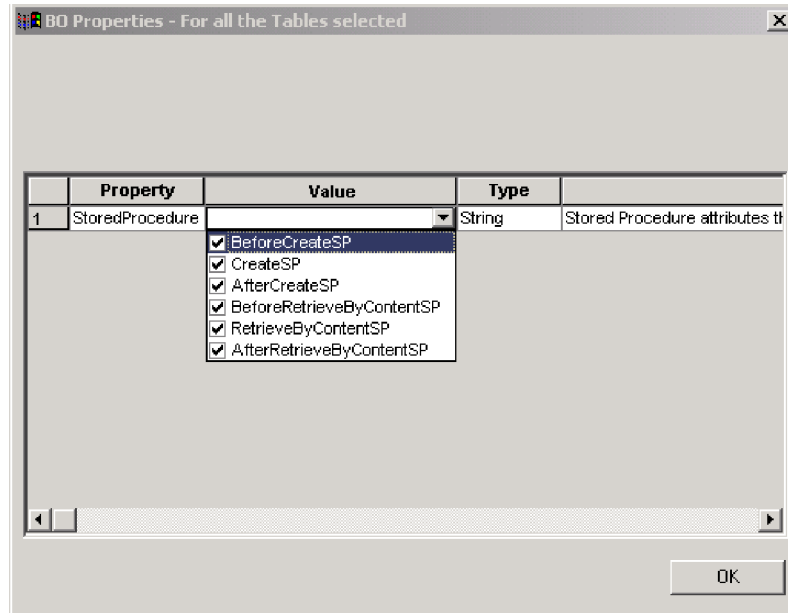


図 12. ストアド・プロシージャ属性の選択

注: 「BO プロパティ」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。

ビジネス・オブジェクトに追加されるストアド・プロシージャの属性は、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ド・プロシージャのいずれかと関連付けることができます。68 ページの図 13 に示すように、各ストアド・プロシージャの属性に対して、そのスキーマ内のデータベースにあるすべてのストアド・プロシージャのドロップダウン・リストから、ストアド・プロシージャを選択できます。ここで指定した情報に基づいて、各属性に必要なアプリケーション固有情報が生成されます。

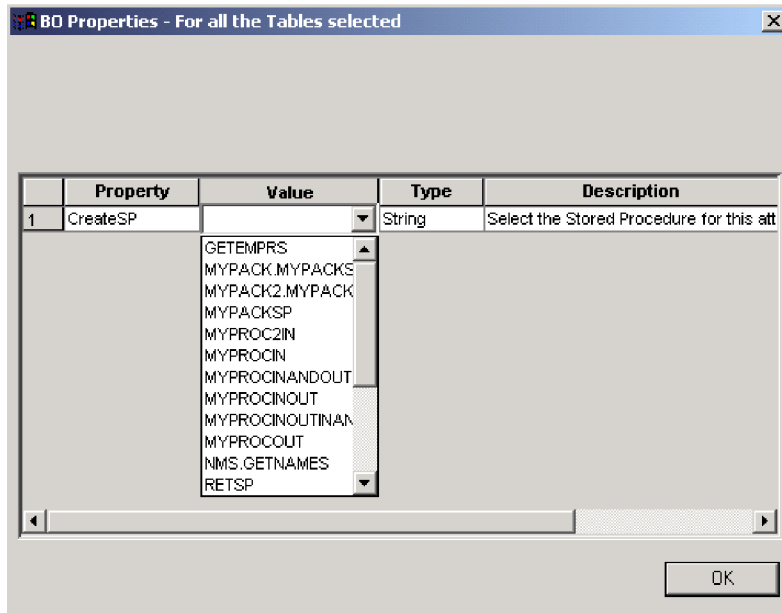


図 13. ストアド・プロシージャとストアド・プロシージャ属性の関連付け

オブジェクト・レベルのアプリケーション固有情報 (ASI) 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。

ストアド・プロシージャからビジネス・オブジェクトが生成され、(SPForCreate などの) JDBC コネクター・ストアド・プロシージャの属性が、それに関連付けられている場合は、ODA が、ストアド・プロシージャの属性に対して、そのスキーマ内のすべてのストアド・プロシージャ名のリストを提供します。ODA により、必要なストアド・プロシージャをビジネス・オブジェクトに関連付けることが可能になります。これにより、以下のように、JDBC コネクター・ストアド・プロシージャの属性に対する ASI が生成されます。

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

ここで、IN はその後続くストアド・プロシージャ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。

ストアド・プロシージャの戻りパラメーターが ResultSet Type である場合、ODA は結果セットを分析し、ビジネス・オブジェクトを生成し、そのビジネス・オブジェクトの結果セット属性の列を作成します。ストアド・プロシージャ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

一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」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。本書は、IBM WebSphere Business Integration Adapters Infocenter からダウンロードできます。

生成される定義の内容

JDBC ODA 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、ビュー、ストアド・プロシージャ、および同義語/ニックネームが指す対象の列に対応する属性 (1 列につき 1 属性)
- 「BO プロパティ」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。

- 『ビジネス・オブジェクト・レベルのプロパティ』
- 70 ページの『属性プロパティ』
- 72 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

JDBC ODA 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。

JDBC ODA がこのレベルで生成する AppSpecificInfo には、指定されたデータベース表、ビュー、ストアド・プロシージャ、または同義語/ニックネームの名前に対応する値が 1 つだけ含まれます。状況列の値の指定については、43 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

このセクションでは、JDBC ODA が属性ごとに生成するプロパティについて説明します。属性の詳細については、39 ページの『ビジネス・オブジェクト属性のプロパティ』を参照してください。

Name プロパティ

JDBC ODA は、指定されたデータベース表、ビュー、ストアド・プロシージャ、または同義語/ニックネームが指す対象に含まれる列の名前に基づいて、属性名の値を取得します。

Data Type プロパティ

JDBC ODA は、属性の型を設定するときに、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 型にマッピングされる個々のデータベース・タイプに関する詳細については、JDBC 仕様 (2.0 以上) を参照してください。表 9 に、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ます。

表 9. 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

表 9. データ型の対応関係 (続き)

JDBC 型	WebSphere Business Integration Adapter ビジネス・オブジェクト・タイプ
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 70 ページの表 9 に示す型のいずれでもない場合、JDBC ODA はその列をスキップし、列が処理できない由のメッセージを表示します。

Cardinality プロパティ

JDBC ODA 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

JDBC ODA は、varchar、char、または text データ型の所定の長さから、ストリングの長さを取得します。

IsKey プロパティ

列が表の基本キーである場合、JDBC ODA はその列をキー属性としてマークします。ただし、ビジネス・オブジェクトの生成のためのソース・ノードとして表ではなくビュー、ストアド・プロシージャ、または同義語/ニックネームが選択されている場合、JDBC ODA 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

JDBC ODA は、IsForeignKey プロパティを設定しません。このプロパティは、Business Object Designer で設定することができます。

IsRequired プロパティ

表、ビュー、ストアド・プロシージャ、または同義語/ニックネームが指す対象に非ヌルに指定されているフィールドがある場合、JDBC ODA 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、JDBC ODA 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、そのフィールドにシーケンスが関連付けられている可能性や、そのフィールドが ID 列である可能性があるためです。

AppSpecificInfo プロパティ

JDBC ODA が属性レベルの AppSpecificInfo プロパティに組み込むパラメータは 2 つあります。指定パラメータの構文は、次のとおりです。

- `CN=ColumnName`

ここで、ColumnName は、この構文が使用されている属性に関係付けられている、データベース表、ビュー、ストアド・プロシージャ、または同義語/ニックネームの列の名前です。

- `BYTEARRAY=true|false`

JDBC ODA は、バイナリー・データを含む列を認識し、AppSpecificInfo プロパティに `BYTEARRAY=true` が含まれる String 型の属性を作成します。

注: 追加の AppSpecificInfo パラメータは、Business Object Designer で設定することができます。これらのパラメータに関する詳細については、44 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

JDBC ODA は、「BO プロパティ」ウィンドウで指定された動詞を生成します。各動詞の AppSpecificInfo プロパティを作成しますが、設定は行いません。詳細については、52 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。本書は、**IBM WebSphere Business Integration Adapters InfoCenter** からダウンロードできます。

ビジネス・オブジェクト定義への情報の追加

データベース表、ビュー、ストアド・プロシージャ、および同義語/ニックネームには、ビジネス・オブジェクト定義にとって必要な情報が欠けている可能性があるため、**JDBC ODA** が作成したビジネス・オブジェクト定義に情報を追加することが必要になる場合があります。詳細については、15 ページの『**第 3 章 ビジネス・オブジェクトの理解**』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。改訂した定義を **IBM WebSphere Business Integration Adapter** リポジトリに再ロードするには、**Business Object Designer** を使用します (**InterChange Server** が統合ブローカーである場合は、**repos_copy** コマンドを使用します)。

第 5 章 コネクターの構成

この章では、IBM WebSphere Business Integration Adapter for JDBC の構成方法を説明します。この章の内容は、次のとおりです。

- 『マルチドライバー・サポートの使用可能化』
- 『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』
- 76 ページの『コネクターの構成プロパティーの設定』
- 77 ページの『Connector Configurator の使用』

マルチドライバー・サポートの使用可能化

JDBC ドライバーは、次の手順を実行して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、`JDBC_DRIVER_PATH` 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. `JDBC_DRIVER_CLASS` 構成プロパティーの値を指定します。

注: JDBC 2.0 以上の仕様に準拠する任意のドライバーがすべての機能をサポートする場合、コネクターは、そのドライバーで稼動することができます。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 JDBC ODA 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、JDBC ODA 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。

コネクターの構成プロパティの設定

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とコネクター固有の構成プロパティ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を設定するには、Connector Configurator ツールを使用します。

- InterChange Server が統合ブローカーである場合は、System Manager から Connector Configurator ツールにアクセスします。
- WebSphere MQ Integrator Broker が統合ブローカーである場合は、IBM WebSphere Business Integration Adapter プログラム・フォルダーから Connector Configurator ツールにアクセスします。

Connector Configurator に関する詳細については、77 ページの『Connector Configurator の使用』を参照してください。

標準構成プロパティ

標準の構成プロパティにより、すべてのコネクタ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に関する詳細については、113 ページの『付録 A. コネクタの標準構成プロパティ』を参照してください。

重要: Connector for JDBC が InterChange Server と WebSphere MQ Integrator Broker をともにサポートしているため、両統合ブローカーの構成プロパティは、Connector for JDBC に関連します。

さらに、IBM WebSphere Business Integration Adapter for JDBC に固有な構成情報については、表 10 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 10. このコネクタ固有のプロパティ情報

プロパティ	注
CharacterEncoding	CharacterEncoding プロパティはこのコネクタによって使用されません。
Locale	このコネクタは国際化されているため、Locale プロパティ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。

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

コネクタ固有の構成プロパティ

コネクタ固有の構成プロパティは、コネクタが実行時に必要とする情報を提供します。また、コネクタ固有の構成プロパティを使用すると、コネクタ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。

これらのプロパティに関する詳細については、133 ページの『付録 B. コネクタ固有の構成プロパティ』を参照してください。

Connector Configurator の使用

このセクションでは、Connector Configurator を使用してアダプターの構成プロパティ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。

- コネクタを構成するためのコネクタ固有のプロパティ・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。

このセクションでは、次のトピックについて説明します。

- 『Connector Configurator の概要』
- 79 ページの『Connector Configurator の始動』
- 80 ページの『System Manager からの Configurator の実行』
- 80 ページの『コネクタ固有のプロパティ・テンプレートの作成』
- 83 ページの『新しい構成ファイルを作成』
- 84 ページの『既存ファイルの使用』
- 85 ページの『構成ファイルの完成』
- 86 ページの『構成ファイル・プロパティの設定』
- 93 ページの『構成ファイルの保管』
- 94 ページの『構成ファイルの変更』
- 94 ページの『構成の完了』
- 94 ページの『グローバル化環境における Connector Configurator の使用』

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・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。

- コネクタを構成するための**コネクタ固有のプロパティ・テンプレート**を作成する。
- **コネクタ構成ファイル**を作成します。インストールするコネクタ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を設定する。
場合によっては、コネクタ・テンプレートでプロパティ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 (79 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

コネクタ構成プロパティには、標準の構成プロパティ (すべてのコネクタがもつプロパティ) と、コネクタ固有のプロパティ (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に必要なプロパティ) とが含まれます。

標準プロパティはすべてのコネクタにより使用されるので、標準プロパティ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の値を設定する必要があります。

標準プロパティの範囲は、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があります。Connector Configurator の「標準のプロパティ」ウィンドウには、特定の構成で設定可能なプロパティが表示されます。

ただし、**コネクタ固有プロパティ**の場合は、最初にプロパティを定義し、その値を設定する必要があります。このため、特定のアダプタのコネクタ固有プロパティ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、80 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 Tools」>「Connector Configurator」をクリックします。
- 「ファイル」>「新規」>「コネクタ構成」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 (85 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」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**」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。

- 「System Manager」ウィンドウの「コネクタ」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。
- Connector Configurator で「ファイル」>「開く」を選択します。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクタ構成ファイルを選択します。
- 「標準のプロパティ」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を確認します。

コネクタ固有のプロパティ・テンプレートの作成

コネクタの構成ファイルを作成するには、コネクタ固有プロパティのテンプレートとシステム提供の標準プロパティが必要です。

コネクタ固有プロパティのテンプレートを新規に作成するか、または既存のコネクタ定義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、80 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、新しい名前でのこのテンプレートを保管します。既存のテンプレートは `¥WebSphereAdapters¥bin¥Data¥App` ディレクトリーにあります。

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を作成し、プロパティの一般特性および値を定義し、プロパティ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。

Connector Configurator でテンプレートを作成するには、以下のステップを実行します。

1. 「ファイル」>「新規」>「コネクタ固有プロパティ・テンプレート」をクリックします。
2. 「コネクタ固有プロパティ・テンプレート」 ダイアログ・ボックスが表示されます。
 - 「新規テンプレート名を入力してください」の下の「名前」フィールドに、新規テンプレート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。
 - テンプレートに含まれているコネクタ固有のプロパティ定義を調べるには、「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。
3. テンプレートを作成するときには、ご使用のコネクタに必要なプロパティ定義に類似したプロパティ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。ご使用のコネクタで使用するコネクタ固有のプロパティ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。
 - 既存のテンプレートを変更する場合には、「変更する既存のテンプレートを選択してください: 検索テンプレート」の下の「テンプレート名」テーブルのリストから、テンプレート名を選択します。
 - このテーブルには、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。テンプレートを検索することもできます。

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: コネクタ固有プロパティ・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・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. 「プロパティを編集」表示でプロパティの名前を選択します。
3. 「最大長」および「最大複数値」のフィールドに値を入力します。

新規プロパティ値を作成するには、以下のステップを実行します。

1. 「**プロパティを編集**」リストでプロパティを選択し、右マウス・ボタンでクリックします。
2. ダイアログ・ボックスから「**追加**」を選択します。
3. 新規プロパティ値の名前を入力し、「**OK**」をクリックします。右側の「**値**」パネルに値が表示されます。

「**値**」パネルには、3つの列からなるテーブルが表示されます。

「**値**」の列には、「**プロパティ値**」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・テンプレート**」ダイアログ・ボックスが表示されます。

依存プロパティは、別のプロパティ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。プロパティを依存プロパティとして指定し、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**」表示で、依存プロパティとして指定するプロパティを選択します。
2. 「**プロパティを選択**」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」表示で依存プロパティを強調表示させて矢印をクリックし、「依存プロパティ」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、構成ファイルの名前を指定し、統合ブローカーを選択する必要があります。

- 「System Manager」ウィンドウで「コネクタ」フォルダーを右クリックし、「新規コネクタの作成」を選択します。Connector Configurator が開き、「新規コネクタ」ダイアログ・ボックスが表示されます。
- スタンドアロン・モードの場合は、Connector Configurator で「ファイル」>「新規」>「コネクタ構成」を選択します。「新規コネクタ」ウィンドウで、新規コネクタの名前を入力します。

また、統合ブローカーも選択する必要があります。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。

- 「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」ウィンドウの残りのフィールドに入力します。

コネクタ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」ダイアログ・ボックス表示されます。

- **名前**

コネクタ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。

- 「コネクタ固有プロパティ・テンプレート」を選択します。

ご使用のコネクタ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・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固有のプロパティが表示されます。

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・パスおよび名前は、コネクタの始動ファイルで指定するコネクタ構成ファイルのパスおよび名前に一致する必要があります。

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。

- コネクタ定義ファイル。
コネクタ定義ファイルは、特定のコネクタのプロパティと、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。コネクタの配布パッケージの `¥repository` ディレクトリ内には、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 コネクタの場合は CN_XML.txt です)。
- ICS リポジトリ・ファイル。
コネクタの以前の ICS インプリメンテーションで使用した定義は、そのコネクタの構成で使用されたりポジトリ・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、コネクタのコネクタ固有プロパティのほとんど、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、コネクタ構成ファイルは、ファイルを開いて、プロパティ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」タブで、BrokerType プロパティ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が「標準のプロパティ」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 89 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すべてに値が設定されているかどうかを確認されます。必須の標準プロパティ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・プロパティの設定

新規のコネクタ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で、以下のカテゴリのプロパティ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
- コネクタ固有のプロパティ
- 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の場合、以下のプロパティ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値も設定可能です。ただし、標準のプロパティおよびコネクタ固有プロパティ、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。

標準プロパティとコネクタ固有プロパティの違いは、以下のとおりです。

- コネクタの標準プロパティは、コネクタ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有されます。すべてのコネクタが同じ標準プロパティのセットを使用します。これらのプロパティの説明は、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は、コネクタ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。各コネクタには、そのコネクタ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があります。これらのプロパティには、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および推奨値が記述されています。

「標準プロパティ」と「コネクタ固有プロパティ」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を表します。これらのプロパティは、アプリケーションまたはコネクタ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の除去も可能です。
- 「値」フィールドは構成できます。
- プロパティごとに「更新メソッド」フィールドが表示されます。これは、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。この設定を構成することはできません。

標準コネクタ・プロパティの設定

標準のプロパティ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」(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の設定

アプリケーション固有の構成プロパティの場合、プロパティ名の追加または変更、値の構成、プロパティの削除、およびプロパティの暗号化が可能です。プロパティ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。プロパティを追加するときは「追加」をクリックします。子プロパティを追加するには、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. プロパティまたは子プロパティの値を入力します。
3. プロパティを暗号化するには、「暗号化」ボックスを選択します。
4. 87 ページの『標準コネクタ・プロパティの設定』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・プロパティ名を変更すると、コネクタに障害が発生する可能性があります。コネクタ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、特定のプロパティ名が必要である場合があります。

コネクタ・プロパティの暗号化

「コネクタ固有プロパティ」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。

各プロパティとそのデフォルト値のリストおよび説明は、各コネクタのアダプター・ユーザーズ・ガイドにあります。

プロパティに複数の値がある場合には、プロパティの最初の値に「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、そのプロパティのすべての値が暗号化されます。プロパティの複数の値を暗号化解除するには、そのプロパティの最初の値の「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、すべての複数值が暗号化解除されます。

更新メソッド

付録 A『コネクターの標準構成プロパティ』の 114 ページの『プロパティ値の設定と更新』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 ICL (Integration Component Library) 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

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・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。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ています。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・プロパティは、DeliveryTransport 標準プロパティの値として 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は、コネクターによるキューの使用方法に影響しません。

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、指定する格納場所へ移動し、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『コネクターの標準構成プロパティー』にある `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。デフォルトでは、このファイルは `¥WebSphereAdapters¥bin¥Data¥App` に保管されます。
- WebSphere Application Server プロジェクトをセットアップしている場合には、このファイルを WebSphere Application Server プロジェクトに保管することもできます。

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 システム・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變更され、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>
  </ValidValues>
  <DefaultValue>en_US</DefaultValue>
</Property>
```

第 6 章 コネクタの実行

この章では、コネクタの開始方法と停止方法、およびコネクタの複数のインスタンスの作成方法を説明します。この章の内容は、次のとおりです。

- 『コネクタの開始』
- 98 ページの『コネクタの停止』
- 99 ページの『コネクタの複数インスタンスの作成』

コネクタの開始

コネクタは、**コネクタ始動スクリプト**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のランタイム・ディレクトリに存在していなければなりません。

```
ProductDir¥connectors¥connName
```

ここで、*connName* はコネクタを示します。始動スクリプトの名前は、表 11 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 11. コネクタの始動スクリプト

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」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 [-cconfigFile ]
```

– UNIX ベースのシステム:

```
connector_manager_connName -start
```

ここで、*connName* はコネクタの名前であり、*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。

コネクタの停止

コネクタを停止する方法は、以下に示すように、コネクタ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を始動した場合は、コネクタ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、コネクタが停止します。
 - UNIX ベースのシステムでは、コネクタ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を停止します。

```
connector_manager_connName -stop
```

ここで、*connName* はコネクタの名前です。

- 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。

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いと必要ありません。

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。

コネクタ定義の作成

Connector Configurator 内で、コネクタ・インスタンスの構成ファイル (コネクタ定義) 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. 初期コネクタの構成ファイル (コネクタ定義) をコピーし、名前変更します。
2. 各コネクタ・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・プロパティ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. 初期コネクタの始動スクリプトをコピーし、コネクタ・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、99 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・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. 初期コネクタのショートカット・テキストをコピーし、新規コネクタ・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・コネクタ作成の詳細については、「コネクタ開発ガイド (C++ 用)」または「コネクタ開発ガイド (Java 用)」を参照してください。

第 7 章 アプリケーションの構成

この章では、アプリケーションによるイベントのポーリングを使用可能にする方法を説明します。この章は、セクション 『コネクター用のアプリケーションの使用可能化』 から構成されています。

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` プロパティが `true` または値なしに設定されている場合、コネクターは、イベント表で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 プロパティを true に設定しておいてください。イベント表からイベントを削除する前に、コネクタは、アーカイブ表にそのイベントを挿入します。

注: コネクタは、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。

イベント処理と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 (140 ページの『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を使用する場合は、このプロパティの値を指定する必要はありません。
- 間隔の頻度 (128 ページの『PollFrequency』)。
- ポーリング間隔ごとのイベントの数 (141 ページの『PollQuantity』)。
- アーカイブ表の名前 (135 ページの『ArchiveTableName』)。
- サブスクライブされていないイベントおよび未処理のイベントをコネクタがアーカイブするかどうか (135 ページの『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。
- コネクタの固有 ID。この ID は、複数のコネクタが同じ表をポーリングするときに重要になります (136 ページの『ConnectorID』)。

また、EventOrderBy プロパティの値を指定して、処理するイベントの順序を指示することも可能です。これらのプロパティおよび他の構成プロパティに関する詳細については、113 ページの『付録 A. コネクタの標準構成プロパティ』および 133 ページの表 15 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。

コネクタを要求の処理のみに使用するには、コネクタを始動するときに -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場合には、
CheckForEventTableOnInit の値を false に設定して、CheckForEventTableOnInit 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

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に必要な表を作成するのを支援することです。IBM
DB2、Oracle、および Microsoft SQL Server 以外のデータベースについては、ここ
のスクリプトを指針に使用して、ユーザーのスクリプトを作成してください。

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: 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、『イベントおよびアーカイブ表スキーマ』を参照してください。

データベース管理者またはコネクタのインプリメント担当者が、特定のインストールおよび照会の最適化要件に適合するように、これらのスクリプトを変更す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。

イベントおよびアーカイブ表スキーマ

表 12 に、イベント表とアーカイブ表の列を示します。

表 12. 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の固有 ID。この値は、複数のコネクタ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、139 ページの『EventKeyDel』プロパティ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
event_priority	コネクタ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	

第 8 章 トラブルシューティングとエラー処理

この章では、Connector for JDBC 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』
- 『イベント処理』
- 『マッピング (InterChange Server 統合ブローカーのみ)』
- 107 ページの『エラー処理とロギング』
- 109 ページの『アプリケーションへの接続不可』
- 109 ページの『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』
- 110 ページの『resource busy エラー』
- 110 ページの『JDBC ドライバーがサポートされていないため、JDBC ODA が正常に動作しません』

始動時の問題

コネクタ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 (InterChange Server 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』
- 106 ページの『日付型変換』

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、コネクタの構成に応じて、コネクタのログ・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 13 では、コネクタが各トレース・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 アーキテクチャによるトレース・メッセージ出力に追加されます。

表 13. コネクタ・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	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• コネクタがビジネス・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 gotAppEvent() 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 13. 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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、JDBCCConnector.txt または JDBCCConnector_II_TT.txt というメッセージ・ファイルに保管されます (II 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、コネクタは処理を続行します。失敗した場合、コネクタは APPRESPONSE_TIMEOUT を戻します。この結果、コネクタは終了します。

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、コネクタは、SchemaName 構成プロパティ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、コネクタの SchemaName 構成プロパティで指定する 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の使用可能化

DB2 データベースでコネクタを使用するには、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、コネクタを実行するマシンの `ProductDir\lib` ディレクトリへコピーする。
2. db2jdbc.dll というファイルを、DB2 ホストから、コネクタを実行するマシンの `ProductDir\bin` ディレクトリへコピーする。
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: JDBC_DRIVER_PATH=`ProductDir/lib/db2java.zip`

Windows: set JDBC_DRIVER_PATH=`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] [Msg: Select CLIENT,COUNTRY,STRT_CODE,CITY_CODE,CITY_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 ] [Msg: :logMsg]
[Time: 2001/05/29 16:30:07.536] [System: ConnectorAgent] [SS: SOVTConnector]
[Type: Error ] [MsgID: 37002]
[Msg: 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、JDBC ODA が正常に動作しません

JDBC ドライバーが JDBC ODA 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 JDBC ODA 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、JDBC ODA 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

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のリストが表示されます。

コネクター固有のプロパティの詳細については、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。

新規プロパティと削除されたプロパティ

以下の標準プロパティは、本リリースで追加されました。

新規プロパティ

- XMLNamespaceFormat

削除されたプロパティ

- RestartCount

標準コネクター・プロパティの構成

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があります。

- 標準構成プロパティ
- コネクター固有の構成プロパティ

このセクションでは、標準構成プロパティについて説明します。コネクター固有の構成プロパティについては、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・プロパティを構成します。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『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のコネクタ・プロパティを設定する場合は、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、コネクタ用の Connector Configurator を開く必要があります。

プロパティ値の設定と更新

プロパティ・フィールドのデフォルトの長さは 255 文字です。

コネクタは、以下の順序に従ってプロパティの値を決定します (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 (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・プロパティの値を変更する場合は、プロパティの**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・プロパティには、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を変更できます。この場合、動的更新は実行できません。

• 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。

• コンポーネント再始動

System Manager でコネクタ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。

特定のプロパティ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 115 ページの表 14 の「更新メソッド」列を参照してください。

標準プロパティの要約

表 14 は、標準コネクタ構成プロパティの早見表です。標準プロパティの依存関係は RepositoryDirectory に基づいているため、コネクタによっては使用されないプロパティがあり、使用する統合ブローカーによってプロパティの設定が異なる可能性があります。

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

注: 表 14 の「注」列にある「Repository Directory は REMOTE」という句は、ブローカーが InterChange Server であることを示します。ブローカーが WMQI または WAS の場合には、リポジトリ・ディレクトリは LOCAL に設定されます。

表 14. 標準構成プロパティの要約

プロパティ名	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> (ブローカーは ICS)
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクタ・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		コンポーネント再始動	
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> (ブローカーは ICS)
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> (ブローカーは ICS)

表 14. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
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
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 の場合は 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> (ブローカーは ICS)
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> (ブローカーは ICS)
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> (ブローカーは ICS)

表 14. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならぬ
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> でなければならぬ (ブローカーは ICS)
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
MessageFileName	パスまたはファイル名	CONNECTORNAMEConnector.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 でなければならぬ
OADAutoRestartAgent	true または false	false	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならぬ (ブローカーは ICS)
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	

表 14. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
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
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	Delivery 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

表 14. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
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
WsifSynchronousRequestTimeout	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

このセクションでは、各標準コネクタ構成プロパティの定義を示します。

AdminInQueue

統合ブローカーからコネクタ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 プロパティは、orb.init[] により開かれる ORB (オブジェクト・リクエスト・ブローカー) 接続の数を制御します。

このプロパティのデフォルト値は 1 に設定されます。必要に応じてこの値を変更できます。

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。コネクタを実行する前に、このプロパティ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 ベースのコネクタでは、このプロパティは使用しません。C++ ベースのコネクタでは、現在、このプロパティに `ascii7` という値が使用されています。

デフォルトでは、ドロップダウン・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。ドロップダウン・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・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` 構成プロパティに、1 より大きい値を設定します。

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

ContainerManagedEvents

このプロパティにより、JMS イベント・ストアを使用する JMS 対応コネクタが、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。

デフォルト値はありません。

`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も構成する必要があります。

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

また、`MimeType`、`DHClass` (データ・ハンドラー・クラス)、および `DataHandlerConfigMOName` (オプションのメタオブジェクト名) プロパティを設定したデータ・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。

これらのプロパティはアダプター固有ですが、例の値は次のようになります。

- `MimeType` = `text/xml`
- `DHClass` = `com.crossworlds.DataHandlers.text.xml`
- `DataHandlerConfigMOName` = `MO_DataHandler_Default`

「データ・ハンドラー」タブのこれら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。

このプロパティは、`DeliveryTransport` プロパティが値 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・コントローラーが検出した場合に、コネクタ・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・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・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・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・コントローラーはその要求を失敗させます。

このプロパティを `false` に設定した場合、コネクタ・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・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。

- RepositoryDirectory がリモートの場合は、DeliveryTransport プロパティの指定可能な値は 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 プロパティに指定されている値が、MQ または IDL である場合、コネクタは、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・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・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・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、コネクタとクライアント・コネクタ・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です。

重要: 以下の環境では、コネクタに JMS トランスポート機構を使用すると、メモリ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が使用されるため、(サーバー側の) コネクタ・コントローラーと (クライアント側の) コネクタ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` プロパティの値を 11 または 12 に設定する。このプロパティの詳細については、「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

このプロパティを `true` に設定すると、JMS 対応コネクタによるデリバリー・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。これはコネクタ開発時に設定されます。

このプロパティは、`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

コネクタでメッセージを処理中にエラーが発生すると、コネクタは、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・プロパティ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・プロパティを必ず設定してください。

デフォルト値は `crossworlds.queue.manager` です。ローカル・メッセージ・ブローカーに接続する場合は、デフォルト値を使用します。

リモート・メッセージ・ブローカーに接続すると、このプロパティは次の (必須) 値をとります。

`QueueMgrName:<Channel>:<HostName>:<PortNumber>`

各変数の意味は以下のとおりです。

QueueMgrName: キュー・マネージャー名です。

Channel: クライアントが使用するチャンネルです。

HostName: キュー・マネージャーの配置先のマシン名です。

PortNumber: キュー・マネージャーが `listen` に使用するポートの番号です。

以下に例を示します。

```
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456
```

jms.NumConcurrentRequests

コネクタ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。

このプロパティは、MQ トランSPORTを使用するコネクタにのみ適用されます。DeliveryTransport プロパティには 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。

<i>ll</i>	2 文字の言語コード (普通は小文字)
<i>TT</i>	2 文字の国または地域コード (普通は大文字)
<i>codeset</i>	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、ドロップダウン・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。ドロップダウン・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。コネクタがグローバル化に対応していない場合、このプロパティの有効な値は `en_US` のみです。特定のコネクタ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/wicsserver/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・バッファ内のイベントの最大数。このプロパティはフロー制御が使用し、RepositoryDirectory プロパティの値が <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・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、製品ディレクトリーの %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・メッセージ・ファイルが存在しない場合は、コネクタは 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: 特定のコネクタについて、コネクタ独自のメッセージ・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ズ・ガイドを参照してください。

MonitorQueue

コネクタ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。このプロパティは、DeliveryTransport プロパティ値が 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を再始動したり、System Monitor からリモート・コネクタ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。このプロパティを有効にするためには、OADAutoRestartAgent プロパティを 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・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・コントローラーはコネクタ・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 プロパティで指定された回数だけ繰り返します。このプロパティを有効にするためには、OADAutoRestartAgent プロパティを 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

これは、前回のポーリングの終了から次のポーリングの開始までの間の間隔です。PollFrequency は、あるポーリング・アクションの終了から次のポーリング・アクションの開始までの時間をミリ秒単位で指定します。これはポーリング・アクション間の間隔ではありません。この論理を次に説明します。

- ポーリングし、PollQuantity の値により指定される数のオブジェクトを取得します。
- これらのオブジェクトを処理します。一部のアダプターでは、これは個別のスレッドで部分的に実行されます。これにより、次のポーリング・アクションまで処理が非同期に実行されます。
- PollFrequency で指定された間隔にわたって遅延します。
- このサイクルを繰り返します。

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 (整数)。
- ワード key。コネクタは、コネクタのコマンド・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、このプロパティの使用が制限されています。このようなコネクターが存在する場合には、アダプターのインストールと構成に関する章で制約事項が説明されています。

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がある場合、標準プロパティの値は、このコネクター固有のプロパティの設定値によりオーバーライドされます。

電子メール・メッセージもイベントと見なされます。コネクターは、電子メールに関するポーリングを受けたときには次のように動作します。

コネクターは、1 回目のポーリングを受けると、メッセージの本文を選出します。これは、本文が添付とも見なされるからです。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターは本文を無視します。

コネクターは PO の最初の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。

2 回目のポーリングを受けると、コネクターは PO の 2 番目の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。これが受け入れられると、PO の 3 番目の添付が届きます。

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・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

コネクタによるコネクタ自体の再始動の試行回数を指定します。このプロパティを並列コネクタに対して使用する場合、コネクタ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

コネクタによるコネクタ自体の再始動の試行間隔を分単位で指定します。このプロパティを並列コネクタに対して使用する場合、コネクタ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・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、このプロパティには xml も設定できます。このプロパティは、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・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、121 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・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・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 を指定する関連 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・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。

WsifSynchronousRequestTimeout

WAS 統合ブローカーでのみ使用されます。

コネクタ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、強力なプロパティです。

デフォルト値は short です。

付録 B. コネクタ固有の構成プロパティ

コネクタ固有の構成プロパティは、コネクタが実行時に必要とする情報を提供します。また、コネクタ固有の構成プロパティを使用すると、コネクタ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 15 に、コネクタに対するコネクタ固有の構成プロパティを示します。プロパティの説明については、以下の各セクションを参照してください。

表 15. コネクタ固有の構成プロパティ

名前	指定可能な値	デフォルト値	必須
ApplicationPassword	コネクタ・ユーザー・アカウントのパスワード		はい*
ApplicationUserName	コネクタ・ユーザー・アカウントの名前		はい*
ArchiveProcessed	true または false	true	いいえ
ArchiveTableName	アーカイブ・キュー表の名前	xworlds_archive_events	ArchiveProcessed が true の場合ははい
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		いいえ
DriverSupportForLong	true または false	true	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	なし、ColumnName、ColumnName, ...]		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

表 15. コネクタ固有の構成プロパティ (続き)

名前	指定可能な値	デフォルト値	必須
QueryTimeout	秒単位の整数値		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	DB2、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
ReturnDummyBOForSP	true または false	false	いいえ
SchemaName	イベントが存在するスキーマ		いいえ
SelectiveCommitForPoll	true または false	false	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・プロシージャの名前		いいえ
StrDelimiter	ReplaceStrList プロパティ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
UseDefaultsWhenPolling	true または false	true	いいえ

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

プロパティの詳細

コネクタ固有の構成プロパティを以下に説明します。

ApplicationPassword

コネクタのユーザー・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。

イベントをアーカイブ表に挿入した後にイベント表から削除させるには、このプロパティを `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、このプロパティを `false` に設定します。この場合、`ArchiveTableName` プロパティ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` プロパティが `false` に設定されている場合は、このプロパティ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 上のストアド・プロシージャが失敗します。

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・プロパティを `false` に設定すると、コネクタは、コネクタの初期化時に、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。

このプロパティが `false` に設定されている場合、コネクタ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、43 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

コネクタの固有 ID です。この ID は、コネクタ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

コネクタの接続先データベース・サーバーの名前です。

WebSphere Business Integration システム・ブランドの SQLServer ドライバーを使用する場合に推奨される URL は次のとおりです。

```
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、135 ページの『AutoCommit』を参照してください。

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

DateFormat

コネクタで受信および戻すことができる日付形式を指定します。このプロパティは、138 ページの表 16 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。

138 ページの表 16 は、時刻パターン・ストリングを使用して DateFormat 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 16. *DateFormat* 構文

シンボル	意味	表示	例
G	紀元	(Text)	AD
y	年	(Number)	1996
M	月	(Text & Number)	July & 07
d	日 (月初からの通算)	(Number)	10
h	時 (12 時間制、1 から 12)	(Number)	12
H	時 (24 時間制、0 から 23)	(Number)	0
m	分 (時刻表示用)	(Number)	30
s	秒 (時刻表示用)	(Number)	55
S	ミリ秒	(Number)	978
E	曜日	(Text)	Tuesday
D	日 (年初からの通算)	(Number)	189
F	曜日 (月初からの通算)	(Number)	2 (2nd Wed in July)
w	週 (年初からの通算)	(Number)	27
W	週 (月初からの通算)	(Number)	2
a	午前/午後	(Text)	PM
k	時 (24 時間制、1 から 24)	(Number)	24
K	時 (12 時間制、0 から 11)	(Number)	0
z	時間帯	(Text)	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 17. *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。

DriverSupportForLong

`PreparedStatements` における整数パラメーターの取り扱い方法を指定します。このプロパティが `true` の場合は、整数パラメーターを設定するために `setLong` が使用されます。`false` の場合は、整数パラメーターを設定するために `setInt` が使用されます。

デフォルト値は `true` です。

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: 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くします。

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は `PollQuantity` プロパティ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。

コネクタによるイベントの順序付けが行われなくするには、このプロパティの値を `none` に設定します。

コネクタに、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。このプロパティの値を指定すると、デフォルトの振る舞いが無効になります。

このプロパティのデフォルト値はありません。

EventQueryType

`EventQueryType` プロパティは、イベント表のイベントの検索の際に、コネクタに照会を動的に生成させるか、またはコネクタ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、104 ページの『イベントおよびアーカイブ表スキーマ』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。この値が `Dynamic` (ストリング) に設定されている場合は、

『`EventTableName`』プロパティ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、139 ページの『`EventOrderBy`』が追加されます。

`EventQueryType` プロパティ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。

コネクタのポーリングをオフにする場合は、このプロパティを `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、このプロパティを `oracle.jdbc.driver.OracleDriver` に設定します。

詳細については、75 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 146 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』プロパティが `false` に設定されている場合、実行時のオープン・データベースの接続数は、この値に 2 を加算した和になります。

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 ステータスマたはストアード・プロシージャを指定します。

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・コールの一例 (sampleSP) です。

```
call sampleSP( )
```

ストアード・プロシージャ・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『データベース接続の喪失』および 109 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: このプロパティがコネクタ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、135 ページの『AutoCommit』が true に設定されている場合は、コネクタは PreserveUIDSeq が false に設定されている場合と同様に振る舞います。

141 ページの『PreserveUIDSeq』プロパティが false に設定されている場合、実行時のオープン・データベースの接続数は、この値に 2 を加算した和になります。

デフォルト値は true です。

QueryTimeout

このプロパティの値は秒単位の整数値であり、指定された秒数にすべての検索の QueryTimeout を設定します。値を指定しない場合は、照会にタイムアウトを設定しないことが暗黙指定されます。照会の処理が、指定された秒数より長引く場合は、データベースにより、キャプチャーされる SQL 例外が生成されます。関連メッセージが、ログ・ファイルに記録されます。

デフォルト値はありません。

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、コネクタに使用させる RDBMS を指定します。IBM DB2、Oracle、または Microsoft SQL Server を使用する場合は、このプロパティの値を DB2、Oracle、または MSSQLServer に設定します。別のデータベースを使用する場合は、そのデータベースの名前または Others に値を設定します。

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。このプロパティが Others に設定されている場合、コネクタは、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。

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。

ReturnDummyBOForSP

このプロパティーは、結果のセットが空の場合でも出力パラメーターを戻すために使用されます。

`RetrieveSP` の場合、結果のセットが戻されます。結果のセットが空の場合は、ビジネス・オブジェクトが生成されず、プロシーチャー呼び出しの戻す出力パラメーターもリトリブできません。ただし、`ReturnDummyBOForSP` が `true` の場合は、対応する属性に読み込まれた出力パラメーターと入出力パラメーターの値を持つダミーのビジネス・オブジェクトが戻されます。

デフォルト値は `false` 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。Microsoft SQL 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。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。

SelectiveCommitForPoll

データベースへのコミットをいつ行うかを指定します。true の場合、イベントの処理終了後、コミットは 1 回のみ行われます。false の場合は、データベースのコミットがイベント処理の各段階で行われるという標準的な振る舞いとなります。

デフォルト値は false です。

SPBeforePollCall

このプロパティは、ポーリング呼び出しごとに実行されるストアード・プロシージャを指定します。SPBeforePollCall プロパティに値 (ストアード・プロシージャ名) が指定されている場合、コネクタは、各ポーリング呼び出しの開始時にそのストアード・プロシージャを呼び出して、コネクタ・プロパティ ConnectorID および PollQuantity の値を渡します。このプロシージャは 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 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 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

このプロパティを使用すると、コネクタ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、このプロパティの設定が行われていない場合には、コネクタは、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、このプロパティの設定が行われていない場合には、コネクタは、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。

UseDefaultsWhenPolling

`UseDefaultsWhenPolling` が `true` に設定されている場合、ビジネス・オブジェクトは、デフォルト値が設定された後、データベースから検索され、サーバーにディスパッチされます。

`UseDefaultsWhenPolling` が `false` に設定されている場合、ビジネス・オブジェクトは、デフォルト値が設定されずに、データベースから検索され、サーバーにディスパッチされます。

デフォルト値は `true` です。

注: このコネクタ固有の構成プロパティは、`UseDefaultsForRetrieve` に代わります。

付録 C. ビジネス・オブジェクトのサンプル

この付録では、Connector for JDBC 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・プロシージャの名前です。このストアード・プロシージャは、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・プロシージャの名前です。このストアード・プロシージャは、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・プロシージャは Retrieve 操作の代わりに実行されます。

BOwithFewerSPPParamsthanBOAttribs.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・プロシージャの名前です。ストアード・プロシージャ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・プロシージャの名前です。ストアード・プロシージャ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・プロシージャは Update 操作の代わりに実行されます。

付録 D. ナ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ナ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、ブランクまたはナルのビジネス・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ナ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。

ビジネス・オブジェクトには、ブランク値を持たないキー属性が、少なくとも 1 つ、必ず存在するようにします。

以下は、1 つのキーを持ち、このキーがナ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 つがナ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 がナ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・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・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

付録 E. メインフレーム・データ・アクセス

Adapter for JDBC は、IBM DB2 Information Integrator Classic Federation for z/OS, V 8.1 を使用してメインフレーム・データ・アクセスをサポートします。この DB2 製品は、Web アプリケーションおよび分散アプリケーションとメインフレーム・データベースとの間の読み取り/書き込み接続を確立します。メインフレーム・データ・ソースに対する高性能の SQL 駆動アクセスとフェデレーションも実現します。ユーザーは、自ら選択したインターネット・ツールとアプリケーションを持つデスクトップを使用して、メインフレームの主幹業務情報に透過的にアクセスすることができます。JDBC Adapter は、JDBC 2.1 仕様に準拠します。

メインフレーム・データ・アクセスに対する JDBC Connector のサポート

JDBC Connector は、DB2 II Classic Federation Server に対して、SQL データ・アクセス要求を行います。JDBC Connector は、DB2 II Classic Federation JDBC Connector と呼ばれる JDBC ドライバーを使用します。



図 14. メインフレーム・データ・アクセスのサポート

Adapter for JDBC は、DB2 II Classic Federation Server を使用して、以下に説明する機能を提供します。

- **サブスクリプション・デリバリー:** JDBC Connector は、DB2 II Classic Federation Server がイベント送達をサポートしていないため、サブスクリプション・デリバリーには使用できません。
- **要求処理:** Adapter for JDBC は、以下のメインフレーム・データ・ソースをサポートします。
 - Software AG ADABAS
 - VSAM
 - CA IDMS
 - IBM IMS Transaction Manager と Database
 - IBM DB2 データベース

メインフレーム・データ・アクセスの使用

メインフレーム・データ・アクセスの使用準備は、以下のようになります。

1. DB2 II Classic Federation for z/OS V 8.1 クライアント・ソフトウェアをインストールします。

2. JDBC Connector の始動スクリプト (97 ページの『コネクターの開始』を参照) 内の JDBCDRIVERPATH に、DB2 JDBC Driver (cacjdbc21.jar) を組み込みます。

注: JDBC ドライバーは、クライアント・ソフトウェアに同梱のものを使用する必要があります。

DB2 II Classic Federation Server に接続するには、以下のようになります。

1. 以下の URL に進み `VLAN: jdbc:cac:DATASOURCENAME:tcp/host name` または `IP address/port number` にログインします。

以下に例を示します。

```
jdbc:cac:CACWBI:tcp/9.30.136.99/8094
```

2. プロンプトが表示されたら、ご使用の VLAN のユーザー ID とパスワードを入力します。ログインが終了すると、ポップアップ・ウィンドウが表示され、認証が成功したことが示されます。このウィンドウを閉じますが、ボタンは一切クリックしないでください。接続は、固定時間が経過するとタイムアウトします。エラー・メッセージが表示された場合は、VLAN に再度ログインする必要があります。

メインフレーム・データ・アクセスのコンポーネント

以下のコンポーネントが相互に連動して、メインフレーム・データに対する SQL データ・アクセスを使用可能にします。

- **DB2 II Classic Federation ODBC と JDBC Connector:** 関数呼び出しを処理し、特定のデータ・ソースに SQL 要求を実行依頼し、Adapter for JDBC に結果を戻します。
- **データ・ソース定義:** アクセスするデータの名前とロケーションから構成されています。必要なデータ・ソース定義には、データ・ソース名および TCP/IP または WebSphere MQ 用の通信パラメーターが含まれます。データ・ソース名は、データ・アクセス要求に対応するために使用される特定のアプリケーション・サーバーまたはエンタープライズ・サーバーを示します。
- **呼び出しレベル・インターフェース:** DB2 II Classic Federation ODBC または JDBC Connector からサーバーで実行されている Query Processor タスクへの橋渡しを行うために使用されます。
- **DB2 II Classic Federation Connection Handler:** アプリケーション・サーバーまたはエンタープライズ・サーバーと通信するために使用されます。DB2 II Classic Federation は、Winsock Interface と WebSphere MQ を使用する TCP/IP インプリメンテーションをサポートする Connection Handler を提供します。

Adapter for JDBC は、DB2 II Classic Federation Server に対して、SQL データ・アクセス要求を行い、(JDBC ODA によって使用される) 標準的な JDBC メタデータ呼び出しによるメタデータ・アクセスをサポートします。このアダプターは、DB2 II Classic Federation ODBC と JDBC Connector を使用します。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

アーカイブ時の振る舞い 6
アーカイブ表 101
アクセス、メインフレーム情報へのアダプター 153
アダプター
依存関係 10
インストール済みファイルの構造 11
環境要件 9
サポートされているオペレーティング・システム 10
サポートされているサード・パーティー製ソフトウェア 10
サポートされているデータベース 10
アダプター・フレームワーク 9
アプリケーション 固有のコンポーネント 1
アプリケーション固有情報
子を表す属性 51
属性レベル 44
名前/値ペア・パラメーター 45
動詞の形式 52
ビジネス・オブジェクト・レベル 43
イベントおよびアーカイブ処理 102
イベントおよびアーカイブ処理の構成 102
イベントおよびアーカイブ表
インストール
SQL スクリプト 103
スキーマ 104
イベント通知 6
イベント表 101
イベント・デリバリー 101
インストール
アダプター
情報 11
統合ブローカー 9
インストール済みファイルの構造 11
インストールの検証 11
エラー処理とロギング 107
エラー・タイプ 107
エラー・メッセージ 108
エラー・メッセージ・ファイルとトレース・メッセージ・ファイル 58

オペレーティング・システム、サポートされている 10

[カ行]

カスタム・ビジネス・オブジェクト・ハン
ドラー・クラス 75
キー値のブランクまたはヌル 151
合格/不合格シナリオ
キー値のブランクまたはヌル 151
構成プロパティ
コネクタ固有 76, 133
プロパティの詳細 134
標準 76
コネクタ
構成プロパティ 76
定義済み 1
コネクタ・アーキテクチャー 2
コネクタ・フレームワーク 1
コネクタ・マネージャー・スクリプト 12
コミット 39

[サ行]

サード・パーティー製ソフトウェア、サポートされている 10
サブスクリプション・デリバリー使用不可 153
サンプル・ビジネス・オブジェクト 149
スクリプト
コネクタ・マネージャー 12
ストアド・プロシージャ
概要 3
処理 34
属性の選択 67
定義済み 30
例 33
Create 操作 35
Delete 操作 36
RetrieveByContent 操作 38
Update 操作 35
属性プロパティ 39
属性名 15

[タ行]

単一カーディナリティー関係
子に関係を格納する 19
所有権のない 17

単一カーディナリティー関係 (続き)
定義済み 16

単純属性 15
データベース、サポートされている 10
データベース接続の喪失 6
データベース・トリガー 6
動詞の処理 21, 23
トランザクション・コミットとロールバック 39
トリガー 6
トレース・メッセージ
タイプ 107
トレース・レベル 59

[ハ行]

バイナリー・データ 52
ビジネス・オブジェクト
イベント ID 5
階層 3, 16
キー値のブランクまたはヌル 151
構造 15
個別 3, 16
最上位 3, 16
差分 21
サンプル 149
処理 3
属性プロパティ 39
属性名 15
単一カーディナリティー関係 16
動詞の処理 21, 23
比較 23
複数カーディナリティー関係 16
フラット 16
変更後イメージ 21
命名規則 15
メタデータ 3
Create 操作 4, 23
Delete 操作 4, 29
DeltaUpdate 操作 28
RetrieveByContent 操作 4
Retrieve 操作 3, 25
RetrieveByContent 操作 26
Retrieve-for-update 操作 38
Update 操作 4, 22, 26
ビジネス・オブジェクト定義
JDBC ODA による生成 55
ビジネス・オブジェクト定義ファイルのサンプル 72
日付型変換 106
複数カーディナリティー関係 16, 19

ポーリング、イベントの 101

[マ行]

メインフレーム・データ・アクセス

使用の準備 153

要求処理 153

DB2 II Classic Federation for z/OS の
使用 153

SQL データ・アクセス要求 153

メタデータ 2

メッセージ・ファイル、エラーとトレース
58

問題

始動時 105

ビジネス・オブジェクトのマッピング
105

[ラ行]

ラッパー・オブジェクト 16, 20, 30

ロールバック 39

[数字]

2 バイト文字セット 7

A

Adapter Development Kit (ADK) 10

ALL SCHEMAS 63

B

Business Object Designer 59

データベース・オブジェクトの選択

62

「BO プロパティ」ウィンドウ 65

C

Connector Configurator 76

Create ストアード・プロシージャ 35

Create 操作 23

D

DB2 II Classic Federation for z/OS 153

DB2 II Classic Federation Server

イベント送達なし 153

接続先 154

Delete ストアード・プロシージャ 36

Delete 操作 29

物理的 30

Delete 操作 (続き)

論理 30

DeltaUpdate 操作 28

J

Java 仮想マシン (JVM) 11

JDBC

仕様 1

ドライバー 10

マルチドライバー・サポート 75

JDBC ODA 55

インストール 55

起動 57

初期化プロパティ 60

ビジネス・オブジェクト定義 69

ファイルのサンプル 72

複数のインスタンスの実行 57

O

Object Discovery Agent (ODA) 55

R

Retrieve 操作 25

RetrieveByContent ストアード・プロシ
ージャ 38

RetrieveByContent 操作 26

Retrieve-for-Update ストアード・プロシ
ージャ 38

S

SmartFiltering 機構 5

SQL ステートメント

概要 3

使用 30

処理 34

U

Unicode 文字コード・セット 7, 11

Update ストアード・プロシージャ 35

Update 操作 22, 26

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を提供し、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せんが、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認できません。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をお願いします。

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ます、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
Lotus
Lotus Domino
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



Printed in Japan