IBM WebSphere Business Integration
Adapters

Mainframe Adapter Suite ユーザーズ・ガイド

Adapter バージョン 2.4.x

お願い

本書および本書で紹介する製品をご使用になる前に、 207 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、Mainframe Adapter Suite バージョン 2.4.x、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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
Mainframe Adapter Suite User Guide
Adapter Version 2.4.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.1

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

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

目次

図

本書について

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

本書では、IBM WebSphere Business Integration Mainframe Adapter Suite のインストール、構成、ビジネス・オブジェクトの開発、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、顧客サイトで WebSphere Business Integration システムを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。

本書の前提条件

以下について理解している必要があります。

- IBM WebSphere Business Integration システム (WebSphere InterChange Server を統合ブローカーとして使用する場合)
- IBM WebSphere MQ Integrator Broker (MQ Integrator Broker を統合ブローカーとして使用する場合)
- MAS が WebSphere Business Integration Adapter を使用して統合する対象の、OS/390 ベースのメインフレーム・レガシー・アプリケーション:
  CICS/TS、DB2、VSAM、ADABAS、IMS/TM、IMS/DB、Natural、または IDMS Database

関連文書

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

以下のサイトから、関連資料をインストールすることができます。

- アダプターの一般情報、WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) でのアダプターの使用、WebSphere Application Server でのアダプターの使用については、次の IBM WebSphere Business Integration Adapters InfoCenter をご覧ください。
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- WebSphere InterChange Server でのアダプターの使用については、次の IBM WebSphere InterChange Server InfoCenter をご覧ください。

http://www.ibm.com/websphere/integration/wicserver/infocenter
http://www.ibm.com/websphere/integration/wbicollaborations/infocenter

- WebSphere Message Brokers の詳細については、以下をご覧ください。
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- WebSphere Application Server の詳細については、以下をご覧ください。
  http://www.ibm.com/software/webservers/appserv/library.html

上記のサイトには資料のダウンロード、インストール、および表示に関する簡単な説明が記載されています。

表記上の規則

本書では、以下のような規則を使用しています。

`courier font`	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、*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	オンラインで表示したときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
ProductDir	IBM WebSphere Business Integration Adapters 製品がインストールされるディレクトリーを表します。デフォルトの製品ディレクトリーは、WebSphereAdapters です。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、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指定できることを意味します。
< >	命名規則では、不等号括弧は名前の個々の要素を囲み、各要素を区別します。(例: *<server_name><connector_name>*`tmp.log`)
/, ¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての WebSphere Business Integration システム製品のパス名は、ご使用の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。
UNIX: および Windows:	これらのいずれかで始まる段落は、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。
u	この記号は、**UNIX/Windows** 段落の終わりを示します。また、複数の段落にわたる注の終わりを示す場合もあります。
%text% および *$text*	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、UNIX 環境変数 *text* の値を示します。

本リリースの新機能

リリース 2.4.x の新機能

2003 年 12 月更新。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。新たな入手先については、第 2 章を参照してください。
- 第 2 章ではまた、表 3 および表 4 で、サブディレクトリー名が変更されました。
- 第 3 章の 『CICS/TS および IMS/TM の CALL ステートメントの構文』で、CICS/TS の CALL の構文および IMS/TM の CALL の構文が変更されました。また、IMS に SQL 要求を発行するステートメントが変更されました。
- 第 4 章の『コネクターの構成』に、トラステッド認証を使用する場合はコネクター固有の構成プロパティー ApplicationPassword と ApplicationUserName が不要であることが追記されました。
- 第 5 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 5 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。
- 付録 C で、『コネクター機能リスト』が除去されました。
- 付録 E で、ビジネス・オブジェクトのサンプルが新しいサンプルに置き換えられました。

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。詳細については、 9 ページの『ブローカーの互換性』を参照してください。
- アダプターは、以下のプラットフォーム上で実行されるようになりました。
  - HP-UX11i
  - AIX 5.x
  - Solaris 8
- Oracle のストアード・プロシージャーからの結果セットの戻りをサポートするようになりました。
- CLOB データ型がサポートされるようになりました。
- コピー属性に対する親の親 (祖父母) からのアクセスがサポートされるようになりました。コピー属性に対して親からのアクセスが可能になりました。この結果、ビジネス・オブジェクト階層の下方向に属性を伝播できます。
- eventid は数値データ型でなければならないという制限が取り除かれました。

リリース 2.2.x の新機能

2003 年 3 月更新。「CrossWorlds」という名前は、現在ではシステム全体を表したり、コンポーネント名やツール名を修飾するためには使用されなくなりました。コンポーネント名およびツール名自体は、以前とほとんど変わりません。例えば、「CrossWorlds System Manager」は現在では「System Manager」となり、「CrossWorlds InterChange Server」は「WebSphere InterChange Server」となっています。

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・オブジェクトのトップレベルにあるラッパー・オブジェクト
  - LIKE 演算子
  - 16 進/バイナリー・データ
  - RetrieveUpdate 動詞のストアード・プロシージャー
  - RetrieveByContent 用の動詞に関するアプリケーション固有情報
  - RetrieveByContent の WHERE 文節の長さが 0 の場合の、WHERE 文節内の動詞に関するアプリケーション固有情報
- ConnectorID プロパティーが int から String に変更されたため、より記述的な名前を使用できるようになりました。
- カスタム JDBC ドライバーによって使用されるネイティブ・ライブラリーを指すため、DRIVERLIB 変数が追加されました。
- オブジェクト処理中のデータベース接続の喪失を検査するための機能が追加されました。

リリース 2.1.x の新機能

Mainframe Connector Suite に含まれる IBM WebSphere Business Integration Adapters には、以前のリリースと同じ機能が備わっています。

リリース 1.9.x の新機能

Mainframe Connector Suite に含まれる IBM WebSphere Business Integration Adapters には、以下のアダプター用のコネクターが含まれています。

- WebSphere Business Integration Adapter for IMS Database Manager
- WebSphere Business Integration Adapter for IMS Transaction Manager
- WebSphere Business Integration Adapter for DB2 Databases
- WebSphere Business Integration Adapter for CICS
- WebSphere Business Integration Adapter for VSAM
- WebSphere Business Integration Adapter for ADABAS

これらのアダプターは、InterChange Server (ICS) および WebSphere MQ Integrator 統合ブローカーとともに動作します。統合ブローカーとは、異種のアプリケーショ

ン・セット間の統合を実行するアプリケーションです。統合ブローカーは、データ・ルーティングなどのサービスを提供します。アダプターには以下のものが組み込まれています。

- アダプターに固有のアプリケーション・コンポーネント
- サンプル・ビジネス・オブジェクト
- IBM WebSphere Adapter フレームワーク。コンポーネントは以下のとおりです。
  - コネクター・フレームワーク
  - 開発ツール (Business Object Designer と Connector Configurator を含む)
  - API (CDK を含む)

本書では、このアダプターを ICS と MQIntegrator の両方の統合ブローカーと共に使用するための情報を提供します。

重要: コネクターは国際化に対応していないため、ISO Latin-1 データのみが処理されることが確実である場合を除いて、コネクターと ICS バージョン 4.1.1 を併用しないでください。

リリース 1.8.x の新機能

Mainframe Connector Suite が IBM CrossWorlds 4.1.x で使用可能になりました。

リリース 1.7.x の新機能

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

リリース 1.6.x の新機能

バージョン 1.6.x が、CrossWorlds Mainframe Connectivity Suite の最初の公開リリースになります。コネクターには以下の機能があります。

- UNIX および Windows 上でのデータ・ソース構成のフル・サポート。 29 ページの『第 3 章 ドライバー・データ・ソースの構成』を参照してください。
- MCS ドライバー接続をテストするユーティリティー。 45 ページの『MAS ドライバー接続のテスト』を参照してください。
- ストアード・プロシージャーのサポート。 90 ページの『ストアード・プロシージャーの使用』を参照してください。

第 1 章 概要

本章では、Mainframe Adapter Suite (MAS) に含まれている WebSphere Business Integration Adapters コンポーネントについて記述し、コネクターの動作方法の概要を示します。本章の内容は、次のとおりです。

Connectors for MAS は、MAS に含まれている WebSphere Business Integration Adapters のランタイム・コンポーネントです。コネクターにより、 WebSphere 統合ブローカーが、MAS のレガシーで使用可能なビジネス・プロセスとビジネス・オブジェクトをやりとりすることが可能です。

用語

Mainframe Adapter Suite はサード・パーティーのソフトウェアおよび資料を取り込んで、バンドルされています。サード・パーティーの文書で使用されている用語は、表 1 に示すように、WebSphere の資料とは異なっています。

表 1. MainFrame Adapter Suite の用語

WebSphere での用語	サード・パーティーでの用語
Mainframe Agent	Shadow Server
MAS ドライバー	Neon Client -JDBC/ODBC ドライバー

注: *レガシー・アプリケーション* および *アプリケーション* は同じ意味で使用されています。

Mainframe Adapter Suite

Mainframe Adapter Suite (MAS) は、8 つのコネクター、MAS ドライバー・コンポーネント、および Mainframe Agent の集合体です。MAS は、以下の IBM OS/390 ベースのメインフレーム・レガシー・アプリケーションを WebSphere Business Integration 環境と統合します。

- CICS/TS
- DB2
- VSAM
- ADABAS
- IMS/TM
- IMS/DB
- Natural
- IDMS Database

MAS ドライバー・コンポーネントおよび Mainframe Agent は、すべての MAS コネクターに単一のテクノロジー・インターフェースを提供します。ほかの MAS コネクターと同様、統合するレガシー・アプリケーションをサポートするように MAS ドライバーを構成する必要があります。ドライバーにデータ・ソースを構成する詳細については、29 ページの『第 3 章 ドライバー・データ・ソースの構成』を参照してください。

コネクター・コンポーネント

コネクターは、アプリケーション固有のコンポーネントとコネクター・フレームワークから成り立っています。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に応じて調整されたコードが含まれます。コネクター・フレームワークは統合ブローカーとアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはどのコネクターにも共通です。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のような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
- 始動メッセージおよび管理メッセージの交換の管理

コネクターにより、統合ブローカーが IBM OS/390 メインフレーム・システム上で構築されたレガシー・アプリケーションとビジネス・オブジェクトをやりとりできるようになります。このセクションでは、コネクターのアーキテクチャーの概要を説明します。

コネクターは、始動時にレガシー・アプリケーションとの接続プールを確立します。そして、データベース/アプリケーションとの間でのトランザクション処理のすべてに、このプール内の接続を使用します。コネクターの終了時には、プール内の全接続がクローズされます。

コネクター・アーキテクチャー

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

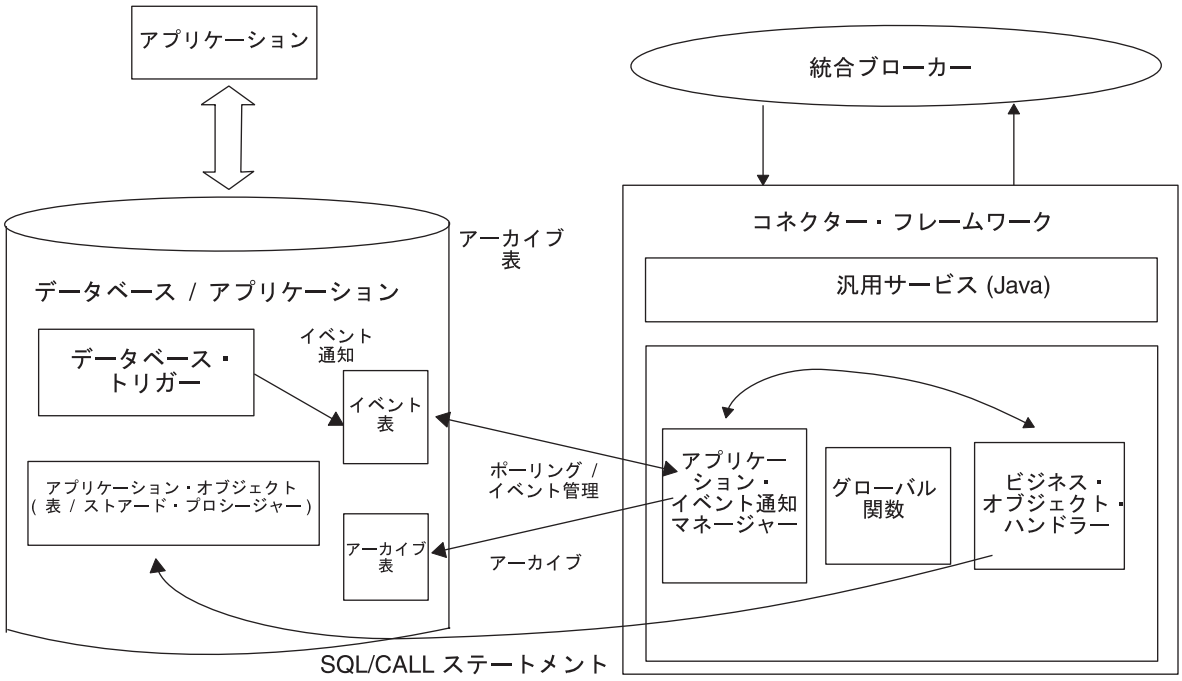


図 1. ビジネス・オブジェクト要求アーキテクチャー

コネクターの動作方法

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

コネクターおよびメタデータ

コネクターは、メタデータ主導型です。WebSphere 環境では、**メタデータ**とは、コネクターとアプリケーションの相互作用を支援する、ビジネス・オブジェクトに格納されたアプリケーション固有のデータです。メタデータ主導型コネクターは、コネクター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ーの設定やアプリケーション固有情報の内容が組み込まれています。コネクターはメタデータ主導型であるため、新規の、または変更されたビジネス・オブジェクトを、コネクター・コードの変更を必要とせずに処理することができます。

コネクターは、SQL ステートメントまたはストアード・プロシージャーを実行して、レガシー・アプリケーションのデータを検索または変更します。コネクターでは、動的 SQL ステートメントまたはストアード・プロシージャーの作成のために、アプリケーション固有のメタデータが使用されます。これらの SQL ステートメントおよびストアード・プロシージャーは、ビジネス・オブジェクトが必要と

し、コネクターが処理している動詞が必要とする、レガシー・アプリケーションからの検索または変更を実行します。アプリケーション固有の情報の使用については、75 ページの『第 5 章 Mainframe Adapter Suite のビジネス・オブジェクトの開発』を参照してください。

ビジネス・オブジェクトの処理

このセクションでは、コネクターが、ビジネス・オブジェクトからの要求やアプリケーション・イベントをどのように処理するかについて概説します。詳細については、81 ページの『ビジネス・オブジェクト動詞の処理』を参照してください。

ビジネス・オブジェクト要求の処理

コネクターは、アプリケーション操作実行の要求を受けると、階層ビジネス・オブジェクトを再帰的に処理します。つまり、すべての個別ビジネス・オブジェクトを処理するまで、子ビジネス・オブジェクトのそれぞれに対し、同じステップを実行します。コネクターが子ビジネス・オブジェクトおよびトップレベル・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: **階層型**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。**トップレベル**・ビジネス・オブジェクトという用語は、それ自身は親ビジネス・オブジェクトを持たない階層構造のトップの個々のビジネス・オブジェクトを指します。

ビジネス・オブジェクトの検索: ビジネス・オブジェクト要求が、レガシー・アプリケーションから階層ビジネス・オブジェクトを検索するようにコネクターに要求すると、コネクターは、そのビジネス・オブジェクトの現在のアプリケーション表記と正確に一致するビジネス・オブジェクトを戻すことを試みます。言い換えると、戻される個々のビジネス・オブジェクトのすべての基本属性が、アプリケーション内の対応するフィールドの値と一致しているということです。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、その配列のアプリケーション内の子の数に一致します。

コネクターはそのような検索を実行する場合、統合ブローカーから受け取ったトップレベル・ビジネス・オブジェクト内の基本キーの値を使用して、アプリケーション内の対応するデータ全体を再帰的に降りて行きます。

ビジネス・オブジェクトの内容による検索: コネクターは、ビジネス・オブジェクト要求から、トップレベル・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: ビジネス・オブジェクト要求から、アプリケーション内に階層ビジネス・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、アプリケーション内に再帰的に作成します。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。

   注: 単一カーディナリティーの子ビジネス・オブジェクトは、すべて、親ビジネス・オブジェクト内での出現順序に従って、親ビジネス・オブジェクトが処理される前に処理されます。子ビジネス・オブジェクトが所有されているかいないかは、処理シーケンスの決定には関係がありません。ただし、処理のタイプの決定には関係があります。
2. トップレベルのビジネス・オブジェクトをアプリケーションに作成します。
3. 単一カーディナリティーの子ビジネス・オブジェクトのうち、親/子関係を子に保管するものをそれぞれ作成します。
4. 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: ビジネス・オブジェクト要求から、アプリケーション内に階層ビジネス・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、アプリケーション内の対応するエンティティーを検索します。
2. トップレベル・ビジネス・オブジェクトの単一カーディナリティーの子をすべて再帰的に更新します。
3. 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。
4. 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: ビジネス・オブジェクト要求から、階層ビジネス・オブジェクトをアプリケーションから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. トップレベル・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

イベント表は、ビジネス・オブジェクト処理に関連したイベントを保存するために使用されます。コネクターには、作成、更新、および削除などのアプリケーション・イベントが、データベース・トリガーなどの適切なメカニズムを通じて通知されます。イベント表にはこれらのアプリケーション・イベントのデータを取り込む必要があります。イベント表は、アプリケーション・データベース内に常駐します。

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。

注: CICS/TS、Natural、IDMS Database、VSAM、および IMS/TM のレガシー・アプリケーションは、以下に記述するすべての機能をサポートするとは限りません。

Create 通知: コネクターは、イベント表内に Create イベントを見つけると、そのイベントが指定したタイプのビジネス・オブジェクトを作成し、ビジネス・オブジェクトのキー値を設定し (イベント表に指定されているキーが使用されます)、アプリケーション内でそのビジネス・オブジェクトを検索します。目的のビジネス・オブジェクトが検索されると、コネクターは Create 動詞とともに、統合ブローカーに送信します。

Update 通知: コネクターは、イベント表内に Update イベントを見つけると、そのイベントによって指定されたタイプのビジネス・オブジェクトを作成し、そのビジネス・オブジェクトのキー値を設定して (このとき、イベント表に指定されているキーが使用されます)、アプリケーション内でそのビジネス・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 動詞とともに送信します。

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

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

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

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

イベント処理

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

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

のイベント処理を完了すると、イベントの状況が更新されます。インストール手順の一部として、アプリケーション・データベースにトリガーを追加する必要があります。

コネクターの ArchiveProcessed プロパティーの設定によって、コネクターが、イベントの状況を更新した後でそのイベントをアーカイブ表にアーカイブするかどうかが決定されます。ArchiveProcessed プロパティーの詳細については、 55 ページの『第 4 章 コネクターの構成』を参照してください。

表 2 に、ArchiveProcessed プロパティーの設定に応じたアーカイブの振る舞いを示します。

表 2. アーカイブ時の振る舞い

アーカイブ処理済み設定	イベント表から削除される理由	コネクターの振る舞い
true または値なし	処理成功	「Sent to InterChange」状況でアーカイブ済み
	処理失敗	「Error」状況でアーカイブ済み
	ビジネス・オブジェクト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 s 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 Unsubscribed にしてイベント表に残します。

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

データベース接続不能の処理

アプリケーションまたはデータベースへの接続が切断されるのには、数多くの理由があります。データベース接続が失われると、コネクターは終了します。JDBC の仕様では、接続の喪失を検出する機構は定められていません。 66 ページの『PingQuery』 プロパティーは、この検出を処理するために提供されています。サービス呼び出し要求中に障害が発生すると、コネクターは、この PingQuery を実行して、アプリケーション/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 がエラーになり、AutoCommit プロパティーが false に設定されている場合、コネクターはデータベースへの新規接続の作成を試みます。データベースへの新規接続の作成に成功した場合、コネクターは処理を続行します。失敗した場合は APPRESPONSETIMEOUT を戻します。この結果、コネクターは終了します。

アプリケーションへのアクセス中に障害が発生した場合は、トランザクションのタイプには関係なく 66 ページの『PingQuery』 が実行されます。例えば、次のようになります。

- 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

第 2 章 コネクターのインストールと構成

本章では、MAS コネクターのインストール方法について説明します。以降の章では、ドライバーおよびコネクターの構成方法を説明します。

この章は、以下のセクションから構成されています。

アダプター環境

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

ブローカーの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for MAS のアダプターのバージョン 2.3.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 2.3.x および 2.4
- **統合ブローカー:**
  - WebSphere InterChange Server、バージョン 4.1.1、4.2、4.2.1、4.2.2
  - WebSphere MQ Integrator、バージョン 2.1.0
  - WebSphere MQ Integrator Broker、バージョン 2.1.0
  - WebSphere Business Integration Message Broker、バージョン 5.0
  - WebSphere Application Server Enterprise、バージョン 5.0.2

  (WebSphere Studio Application Developer Integration Edition、バージョン 5.0.1 と併用)

例外については、「*リリース情報*」を参照してください。

注: 統合ブローカーおよびその前提条件のインストールに関する説明については、以下のガイドを参照してください。
WebSphere InterChange Server (ICS) については、「*IBM WebSphere InterChange Server システム・インストール・ガイド (UNIX 版)*」または「*IBM WebSphere InterChange Server システム・インストール・ガイド (Windows 版)*」を参照してください。
WebSphere Message Brokers については、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」を参照してください。
WebSphere Application Server については、「*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 5.1、AIX 5.2
- Solaris 7.0、Solaris 8.0
- HP UX 11.0、HP UX 11i
- Windows 2000

グローバリゼーション

このアダプターは DBCS (2 バイト文字セット) で使用可能です。

前提条件

MAS 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。

- WebSphere Business Integration システム・ソフトウェア・バージョン 4.2.0 以降、または WebSphere Business Integration Adapter Framework バージョン 2.0 以降を Adapter Development Kit (ADK) と共にインストールします。

  WebSphere InterChangeServer (ICS) または WebSphere MQ Integrator Broker とは異なるマシンでコネクターを実行する場合は、コネクターを実行する前に、ICS または WebSphere MQ Integrator Broker のバージョンと互換性がある ADK をインストールします。
- JDK 1.3.1 ソフトウェアをインストールします。「*IBM WebSphere InterChange Server システム・インストール・ガイド (Windows 版)*」または「*IBM WebSphere InterChange Server システム・インストール・ガイド (UNIX 版)*」を参照してください。
- Neon Client バージョン 3.8.788 をインストールします。
- Mainframe Agent[1] がインストール済みであることを確認します。
- アプリケーションにユーザー・アカウントが存在していることを確認します。

1. Shadow Server

コネクターは、そのコネクターが直接対話しているアプリケーションのデータを処理する際、その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。

MAS ドライバーの前提条件

Mainframe Agent がインストール済みで、現在 MAS ODBC ドライバーおよび JDBC ドライバーにサポートされている以下のプラットフォームを使用していることが必要です。

- Windows
  - Windows 3.1
  - Windows 95
  - Windows 98
  - Windows 2000
- UNIX
  - Solaris 2.4 以降
  - HP-UX 11i
  - AIX 5.2

アダプターと関連ファイルのインストール

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

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

MAS ドライバーのインストール

MAS ドライバーはすべての MAS コネクターに必要です。以下のサブセクションで、UNIX システムおよび Windows システム上の MAS ドライバーの内容について解説します。

UNIX システム上の MAS ドライバー

表 3 に、UNIX システム上の MAS ドライバーのファイル構造を示します。

表 3. MAS ドライバー用としてインストールされた UNIX ファイル構造

***$ProductDir* のサブディレクトリー**	**説明**
`IBM Shadow Client/Shadow`	ファイル場所、クラス・パス環境変数、その他の情報の変更点をすべて記述する、`Readme12` という名前のテキスト・ファイルが格納されます。

表 3. MAS ドライバー用としてインストールされた UNIX ファイル構造 (続き)

***$ProductDir* のサブディレクトリー**	**説明**
IBM Shadow Client/Shadow	MCS ドライバーである dsa、neontrace.set、odbc (ドライバー)、scjdlog、および scodbcdm の各ファイルが格納されます。
IBM Shadow Client/Shadow	jdbc2_0-stdext.jar、scjd12.jar ファイルと scjd12ts.jar ファイル、および Java クラス・ファイルである scjdsm01 および scjdsm02 が格納されます。
IBM Shadow Client/Shadow	libsccyp.so、libscjd12.so、libscjd12ts.so、libscodbc_r.so.1、libscodbcts_r.so.1、および libscssl.so の各ファイルが格納されます。

Windows システム上の MAS ドライバー

表 4 に、Windows システム上の MAS ドライバーのファイル構造を示します。

表 4. MAS ドライバー用としてインストールされた Windows ファイル構造

***%ProductDir%* のサブディレクトリー**	**説明**
IBM Shadow Client/Shadow	Dsa32.exe、scjdlog.exe、scod32dm.exe、scod32dm.map、および Vbdemo.32.exe が格納されます。
IBM Shadow Client/Shadow	scjd12.jar ファイルと scjdts12.jar ファイル、および Java クラス・ファイルである scjdsm01 と、ファイル場所、クラス・パス環境変数、その他の情報の変更点をすべて記述する、Readme11 という名前のテキスト・ファイルが格納されます。scjd12.dll ファイル、および scjd12ts.dll ファイルも追加されました。
IBM Shadow Client/Shadow	.dll ファイルである Scod32.dll、Scod32r.dll、Scod32tr.dll、および Scod32ts.dll が格納されます。

ADABAS コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの ADABAS コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 5 に、コネクターが使用する UNIX ファイル構造を示します。

表 5. MAS Adapter for ADABAS 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/ADABAS	コネクターの CWADABAS.jar ファイルと start_ADABAS.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	ADABASConnector.txt ファイルが格納されます。
repository/ADABAS	CN_ADABAS.txt ファイルが格納されます。

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

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

コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

14 ページの表 6 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 6. *MAS Connector for ADABAS* 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\ADABAS	コネクターの CWADABAS.jar ファイルと start_ADABAS.bat ファイルが格納されます。
connectors\messages	ADABASConnector.txt ファイルが格納されます。
repository\ADABAS	CN_ADABAS.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

CICS/TS コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの CICS/TS コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 7 に、コネクターが使用する UNIX ファイル構造を示します。

表 7. MAS Connector for CICS/TS 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/Cics	コネクターの CWCICs.jar ファイルおよび start_CICS.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	CICSConnector.txt ファイルが格納されます。
repository/Cics	CN_CICS.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

16 ページの表 8 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 8. *MAS Connector for CICS/TS 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\Cics	コネクターの CWCICS.jar ファイルおよび start_CICS.bat ファイルが格納されます。
connectors\messages	CICSConnector.txt ファイルが格納されます。
repository\Cics	CN_CICS.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

DB2 コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの DB2 コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 9 に、コネクターが使用する UNIX ファイル構造を示します。

表 9. MAS Connector for DB2 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/Db2	コネクターの CWDB2.jar ファイルおよび start_DB2.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	DB2Connector.txt ファイルが格納されます。
repository/Db2	CN_DB2.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

18 ページの表 10 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 10. MAS Connector for DB2 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\Db2	コネクターの CWDB2.jar ファイルおよび start_DB2.bat ファイルが格納されます。
connectors\messages	DB2Connector.txt ファイルが格納されます。
repository\Db2	CN_DB2.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

IMS/DB コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの IMS/DB コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 11 に、コネクターが使用する UNIX ファイル構造を示します。

表 11. MAS Connector for IMS/DB 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/Ims_db	コネクターの CWIMS_DB.jar ファイルおよび start_IMS_DB.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	IMS_DBConnector.txt ファイルが格納されます。
repository/Ims_db	CN_IMS_DB.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

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

表 12. MAS Connector for IMS/DB 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\Ims_db	コネクターの CWIMS_DB.jar ファイルおよび start_IMS_DB.bat ファイルが格納されます。
connectors\messages	IMS_DBConnector.txt ファイルが格納されます。
repository\Ims_db	CN_IMS_DB.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

IMS/TM コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの IMS/TM コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 13 に、コネクターが使用する UNIX ファイル構造を示します。

表 13. MAS Connector for IMS/TM 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/Ims_tm	コネクターの CWIMS_TM.jar ファイルおよび start_IMS_TM.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	IMS_TMConnector.txt ファイルが格納されます。
repository/Ims_tm	CN_IMS_TM.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

Windows システムにコネクターをインストールするには、IBM WebSphere InterChange Server インストーラーを実行してコネクターを選択してください。IBM WebSphere InterChange Server インストーラーがコネクターに関連した標準ファイルをインストールします。 22 ページの表 14 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 14. MAS Connector for IMS/TM 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\Ims_tm	コネクターの CWIMS_TM.jar ファイルおよび start_IMS_TM.bat ファイルが格納されます。
connectors\messages	IMS_TMConnector.txt ファイルが格納されます。
repository\Ims_tm	CN_IMS_TM.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。55 ページの『第 4 章 コネクターの構成』を参照してください。

VSAM コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの VSAM コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 15 に、コネクターが使用する UNIX ファイル構造を示します。

表 15. MAS Connector for VSAM 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/Vsam	コネクターの CWVSAM.jar ファイルおよび start_VSAM.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	VSAMConnector.txt ファイルが格納されます。
repository/Vsam	CN_VSAM.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

24 ページの表 16 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 16. MAS Connector for VSAM 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\Vsam	コネクターの CWVSAM.jar ファイルおよび start_VSAM.bat ファイルが格納されます。
connectors\messages	VSAMConnector.txt ファイルが格納されます。
repository\Vsam	CN_VSAM.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

Natural コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの Natural コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 17 に、コネクターが使用する UNIX ファイル構造を示します。

表 17. MAS Connector for Natural 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/Natural	コネクターの BIA_NATURAL.jar ファイルおよび start_NATURAL.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	BIA_NATURALConnector.txt ファイルが格納されます。
repository/Natural	BIA_CN_NATURAL.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

26 ページの表 18 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 18. MAS Connector for Natural 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\Natural	コネクターの BIA_NATURAL.jar ファイルおよび start_NATURAL.bat ファイルが格納されます。
connectors\messages	BIA_NATURALConnector.txt ファイルが格納されます。
repository\NATURAL	BIA_CN_NATURAL.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

IDMS Database コネクターのインストール済みファイル構造

以下のサブセクションでは、UNIX または Windows システムでの IDMS Database コネク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。

インストール済みファイル構造 (UNIX システムの場合)

表 19 に、コネクターが使用する UNIX ファイル構造を示します。

表 19. MAS Connector for IDMS Database 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/IDMS_DB	コネクターの BIA_IDMS_DB.jar ファイルおよび start_IDMS_DB.sh ファイルが格納されます。コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integration Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクター・コンフィグレーション (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ 統合ブローカーと連動する場合、このカスタマイズされたラッパーはコネクターを始動するためのみに使用します。コネクターを停止するには、mqsiremotestopadapter を使用します。
connectors/messages	BIA_IDMS_DBConnector.txt ファイルが格納されます。
repository/Idms_db	BIA_CN_IDMS_DB.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

インストール済みファイル構造 (Windows システムの場合)

28 ページの表 20 に、コネクターが使用する Windows ファイルのファイル構造を示します。

表 20. MAS Connector for IDMS Database 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\Idms_db	コネクターの BIA_IDMS_DB.jar ファイルおよび start_IDMS_DB.bat ファイルが格納されます。
connectors\messages	BIA_IDMS_DBConnector.txt ファイルが格納されます。
repository\Idms_db	BIA_CN_IDMS_DB.txt ファイルが格納されます。

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

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

注: コネクターを使用する前に、構成する必要があります。 55 ページの『第 4 章 コネクターの構成』を参照してください。

第 3 章 ドライバー・データ・ソースの構成

本章では、Mainframe Adapter Suite ドライバーのデータ・ソースの構成方法について説明します。本章の内容は、次のとおりです。

MAS ドライバーの概要

MAS ドライバーは、ODBC (Open DataBase Connectivity) API および JDBC (Java Database Connectivity) API のいずれかを使用するメインフレーム・システムとも接続できます。ODBC ドライバーと JDBC ドライバーには、以下の機能があります。

- DB2、CICS、IMS/TM、VSAM、ADABAS、Natural、IDMS Database、およびその他のソースへのアクセス
- 1 つの接続で複数のデータ・ソースに対応
- 認証およびメッセージ処理を行う OS/390 セキュリティー (RACF、ACF2、および Top Secret) との緊密な統合性
- TCP/IP、SNA、または WebSphere MQ 上で稼働
- DBCS サポート

DB2 UDB for OS/390 にアクセスする Web ベース・アプリケーションの多くは、DB2 のストアード・プロシージャーのサポートが必要です。MAS Mainframe Agent および MAS ドライバーは、DB2 のストアード・プロシージャーを完全にサポートし、そのため、CICS/TS 、IMS/TM、IMS/DB、ADABAS、VSAM、Natural、IDMS Database に加え DB2 にアクセスする COBOL、C/C++、PL/I、またはアセンブラーの各プログラムを JDBC アプリケーションから呼び出すことができます。

Windows 環境と UNIX 環境

Windows 環境では、ODBC アプリケーションは Microsoft から無償で提供されるドライバー・マネージャーと直接インターフェースします。このドライバー・マネージャーが実際に ODBC ドライバーのロードおよび ODBC API の起動を行います。このアプリケーションは、ドライバーの静的参照は行いません。

UNIX 環境では、サード・パーティー・ベンダーから購入しないと ODBC ドライバー・マネージャーは使用できません。UNIX アプリケーションは大容量スコープを持ち複雑なため、DBMS を切り替えて標準化された ODBC API インターフェースを使用する必要があることはめったにありません。たいていの場合、アプリケーションは 1 つの DBMS タイプに接続するように設計されています。そのため、ドライバー・マネージャーは不要で、すでに複雑なクライアント/サーバー設計に、さらに別の障害ポイントが展開されることがあります。

MAS ODBC ドライバーは、アプリケーションがドライバー・マネージャーをバイパスし、直接ドライバー・ライブラリーにリンクできるように設計されています。ただし、まれにドライバー・マネージャーが必要となるケースもあり、その場合ドライバー・マネージャーが稼働するための簡単なステップを実行します。

MAS ODBC ドライバー

MAS ODBC ドライバーには ODBC 仕様のレベル 2 が実装されているため、現行のレベル 3 アプリケーションの多くをサポートします。また、既存の IMS トランザクション、CICS プログラム、および VSAM および ADABAS データにアクセスしたり、顧客が記述した OS/390 常駐プログラムによってアクセスするような、非リレーショナル OS/390 データ・ソースへの ODBC アクセスも提供します。 MAS ODBC ドライバーが使用されるのは、クライアント/サーバー・アプリケーション、3 層 Web サーバー、EIS、随時照会ツール、エンタープライズ・リソース・プランニング (ERP) ソリューション、EAI、および ODBC に準拠したデータウェアハウスなどです。

MAS JDBC ドライバー

MAS JDBC ドライバーにより、Java アプリケーションは、JDBC API を使用して多重仮想記憶 (MVS) データとトランザクション・ソースを統合できます。JDBC は、Java データベース・アプリケーションで使用されるように設計されています。

JDBC ドライバーは JDBC 2.0 に準拠しており、JDK 1.2.3 (J2EE) および Java サーブレットをサポートします。標準 JDBC API を使用して DB2 Universal Database (UDB) for OS/390、ADABAS、VSAM、IMS/DB、IMS/TM、CICS/TS などのデータにアクセスするために、OS/390 常駐のデータおよびトランザクションのアクセスと統合をサポートします。

ODBC ドライバーの構成

MAS ODBC Client には、MAS ODBC Direct という 1 つの基本接続オプションがあります。このオプションおよび機能については、以下のセクションで説明します。

MAS ODBC Direct

これは推奨オプションです。このオプションによって、Windows NT および 2000、OS/2、UNIX、および Linux の各クライアント・システムは、DB2-OS/390、IMS/DB、IMS/TM、CICS/TS、ADABAS、VSAM、PDS、フラット・ファイルなどを含むさまざまなタイプのデータにアクセスできます。TCP/IP ユーザーおよび LU 6.2 ユーザーが使用可能なドライバーがあります。MAS Mainframe Agent は、ODBC API を使用するアプリケーションと対話します。

クライアント接続に ODBC を選択している場合、DBMS タイプを知っておく必要があります。DBMS タイプとはデータにアクセスする方法です。最初に、ODBC ドライバー・データ・ソースを構成するのに使用するツール、ODBC データ ソース アドミニストレータに DB2 タイプが表示されます。 33 ページの『ODBC データ ソース アドミニストレータの使用』を参照してください。

ODBC ドライバーの制御

MAS ODBC ドライバーは、構成が完了するとドライバー・キーワードを使用して制御できます。これらのキーワードには、使用するコネクターによって変わるデフォルト値が自動的に指定されます。詳細については、155 ページの『付録 C. MAS ドライバーのキーワード』を参照してください。

データ・ソース

定義するデータ・ソースには、指定されたデータ・プロバイダーへの接続方法についての情報が保管されています。

自動的にインストールされた ODBC アドミニストレータでデータ・ソースを構成します。ODBC アドミニストレータにより、要件に従って、同じワークステーション上に任意のタイプの複数のデータ・ソースを構成できます。

MAS のデータ・ソースはデスクトップです。すべての接続情報は、各ユーザーの ODBC.INI ファイルにローカルに保管されています。接続情報を変更する場合、新規 ODBC.INI ファイルを全利用者の PC に波及させる必要があります。データ・ソースを別のコンピューターに移動させる場合、すべてを再構成する必要があります。

マルチスレッド・アプリケーション・サポートの使用

MAS ODBC ドライバーは、マルチスレッド・アプリケーションで稼働するように機能拡張されています。マルチスレッド・アプリケーションには、並行スレッドまたはスレッド・ベース接続プール・モデルを使用して ODBC 操作を起動するさまざまな汎用的サーバー製品が組み込まれています。MAS ODBC ドライバーは完全にスレッド・セーフなため、スレッド親和性は必要ありません。

デフォルトでは、MAS ODBC ドライバーはスレッド・セーフ・モードでは稼働しません。スレッド・セーフ・モードを使用可能に設定する必要があります。このデフォルトは、並行スレッドによる共有リソースへのアクセスを制御する要求や、マルチスレッド・プログラミング・モデルを使用するアプリケーションがほとんどないために生じるパフォーマンス・ペナルティーを除去するために設定します。

マルチスレッド・モードによるドライバーの実行

マルチスレッド・アプリケーションによる呼び出し時にドライバーをスレッド・セーフ・モードで実行するには、以下のいずれかの方法で NEONTHREADSAFE 変数を YES または 1 に設定します。

- レジストリー・キーを変更する (Windows の場合のみ)。ドライバーをスレッド・セーフ・モードで実行するには、以下のようにレジストリー・キー NEONTHREADSAFE を YES または 1 に設定します。

  ```
  NEONTHREADSAFE = YES
  ```

  レジストリーが変更されると、ドライバーはどのアプリケーションに呼び出されても、シングルスレッド設計かマルチスレッド設計かに関わらずスレッド・セーフ・モードで稼働します。 レジストリー・キーによる NEONTHREADSAFE 変数のスレッド・セーフ・モードへの設定は、Microsoft Windows プラットフォームの場合のみ使用可能なマシン全体の設定になります。

- 環境変数の設定 (Windows または UNIX の場合)。環境変数 NEONTHREADSAFE を設定してドライバーをスレッド・セーフ・モードで実行するには、アプリケーションのスタートアップ・ファイル (.bat または .sh) に以下の行を追加してください。

  ```
  NEONTHREADSAFE = YES
  ```

  環境変数を使用してスレッド・セーフ・モードを使用可能にすると、起動時に環境設定を引き継ぐアプリケーションにのみ反映されます。

  注: ODBC.INI レジストリー・キーまたは ODBC.INI ファイルを手動で編集してこのキーワードを追加する、または接続ストリングにこのキーワードを追加する場合、スレッド・セーフにはなりません。

マルチスレッド・モードによるデバッグ・トレースの使用可能化

環境変数 NEONTRACE を使用する場合、その値によって該当セッションのトレース設定が制御されます。これは、NEONTRACE ODBC キーワードによって設定された値は無視されることを意味します。デバッグ・トレースとスレッド・セーフを使用可能にするには、NEONTRACE 環境変数を使用してすべてのキーワードを設定してください。

例えば、スレッド・セーフを強制し、INFO レベルのトレース・ファイルを Windows システムの C:¥NEONLOG.TXT に生成するとします。

- Windows 95 以上の場合: AUTOEXEC.BAT に

  ```
  NEONTRACE=LOCKTHREAD INFO LOG c:¥neonlog.txt
  ```

  という行を追加します。
- Windows NTの場合: 「**コントロールパネル**」を開き、「**システム**」アイコンを選択して、ご使用の環境変数設定に

  ```
  NEONTRACE=LOCKTHREAD INFO LOG=C:¥neonlog
  ```

  という行を追加します。

UNIX システムの場合:

- MAS Connector のデバッグ・バージョンをご使用の場合、トレース機能が使用できます。デバッグを使用可能にするには、odbc.ini ファイルを編集して NEONTRACE 変数の値とトレース・ログ・ファイルのパスを指定してください。odbc.ini ファイルのサンプルを挙げます。

  ```
  NEONTRACE=INFO BUFFERTRACE LOG=/home/ai38aas/shadow/bin/neonlog.txt
  ```

注: トレースの詳細については、48 ページの『トレース』を参照してください。

Windows 用 ODBC ドライバー・データ・ソースの構成

Windows システムでは、MAS ODBC データ ソース アドミニストレータというグラフィカル・ユーザー・インターフェース (GUI) を使用してデータ・ソースを構成します。以下のセクションでその方法について説明します。

ODBC データ ソース アドミニストレータの使用

MAS をインストールすると、PC の「**コントロール パネル**」に MAS ODBC データ ソース アドミニストレータのアイコンが自動的に表示されます。ドライバーへのアクセスは、インストール処理中に指定したパスを使用して行うこともできますが、「**コントロール パネル**」を使用するのが最も簡単な方法です。

MAS ODBC ドライバー・データ・ソースを構成するには、以下の処理を行います。

1. 「**スタート**」メニューを開いて、Windows NT 3.0 以上の場合は「**設定**」/「**コントロールパネル**」を、Windows 95 以上の場合は「**設定**」/「**コントロール パネル**」/「**管理ツール**」を選択します。
2. 「**ODBC データ ソース**」 アイコン (「一覧」または「詳細」で表示している場合はアイテム) をダブルクリックします。

   「**ODBC データ ソース アドミニストレータ**」ダイアログ・ボックスが表示されます。

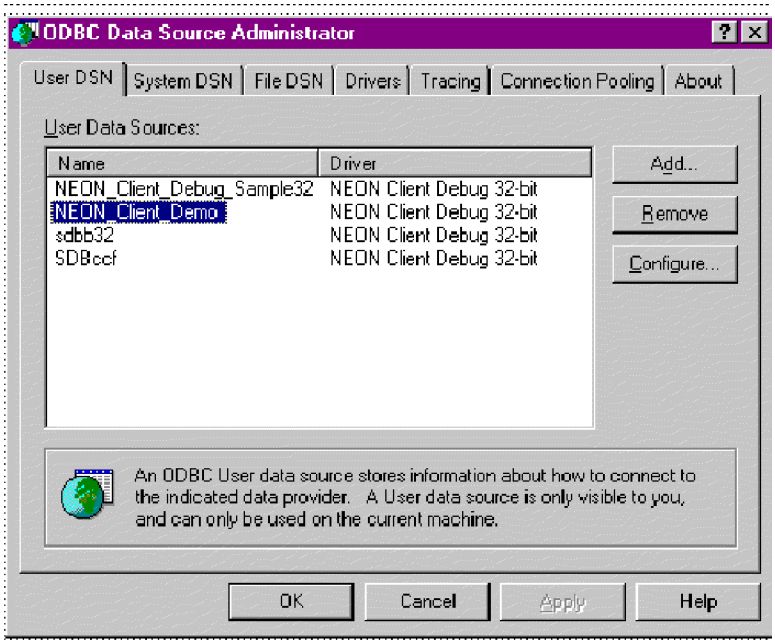


図 2. 「ODBC データ ソース アドミニストレータ」ダイアログ・ボックス

3. 「**User DSN**」タブを選択します。
4. 「**追加**」をクリックします。

   図 3 に示しているように、「データ ソースの新規作成」ボックスが表示されます。

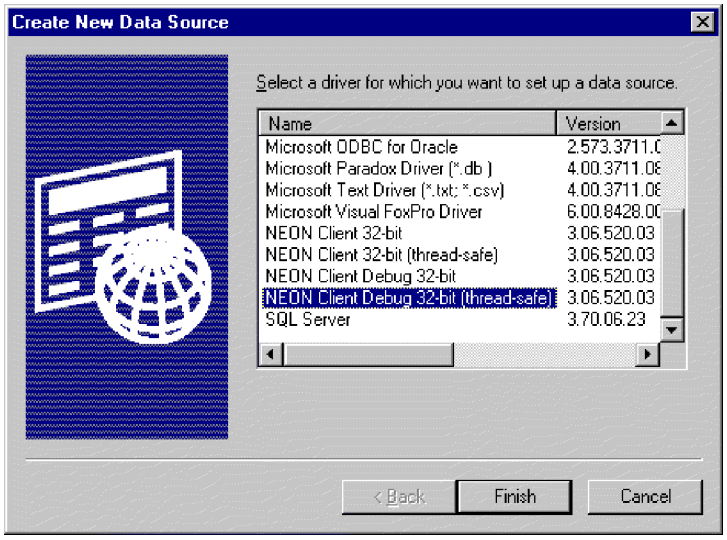


図 3. 「データ ソースの新規作成」ダイアログ・ボックス

5. Neon Client Debug 32-bit (thread-safe) または Neon Client 32-bit (thread-safe) ドライバーを選択して、**「完了」**をクリックします。

   注: スレッド・セーフのみをお勧めします。

   図 4 に示しているように、構成ダイアログ・ボックスが表示されます。

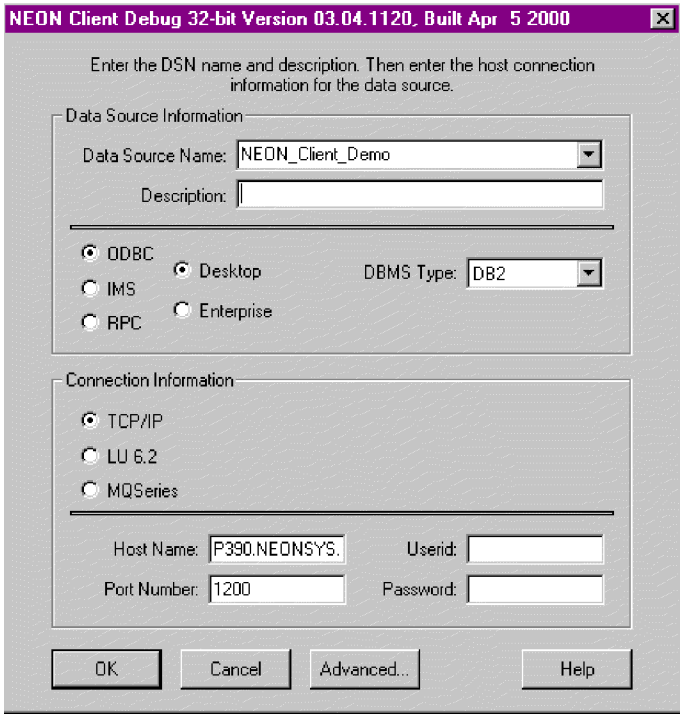


図 4. 「MAS デバッグ構成 (MAS debug configuration)」ダイアログ・ボックス

6. **「データ ソース名」** および **「説明」**に値を入力します。
7. データ・ソース・タイプに**「ODBC」**を選択します。 ダイアログ・ボックスのオプションは、データ・ソース・タイプによって変わります。

ODBC:

- **「ODBC」**を選択すると、図 5 に示したオプションが表示されます。

図 5. ODBC データ・ソース・タイプ

- **「DBMS タイプ (DBMS Type)」**を入力します。DBMS タイプ (DBMS Type) はデータにアクセスするのに使用する方法です。このケースでは、デフォルトは DB2 です。表 21 を参照してください。

表 21. データ・ソース DBMS タイプ

データ・ソース	「DBMS タイプ (DBMS Type)」フィールドへの入力
DB2	DB2
VSAM	VSAM
VSAMCICS	VSAMCICS
ADABAS	ADABAS
IMSDB	IMSDB
IMS/TM	DB2 (注を参照)
CICS/TS	DB2 (注を参照)
Natural	DB2 (注を参照)
IDMS Database	DB2 (注を参照)

注: IMS/TM、CICS/TS、Natural、および IDMS Database にアクセスするときには、「DBMS タイプ (DBMS Type)」に 「DB2」を選択できます。このケースでは、CALL ステートメントを使用してすべての作成、変更、検索、および削除操作を処理します。41 ページの『CICS/TS ステートメントおよび IMS/TM CALL ステートメントの構文』を参照してください。

8. **「接続情報 (Connection Information)」** (34 ページの図 4 参照) で **「TCP/IP」**、**「LU 6.2」**、または**「WebSphere MQ」**を選択します。 データ・ソース・タイプと同様、このダイアログ・ボックスも選択するオプションによって変わります。

   TCP/IP:

   - **「TCP/IP」**を選択すると、36 ページの図 6 に示したオプションが表示されます。

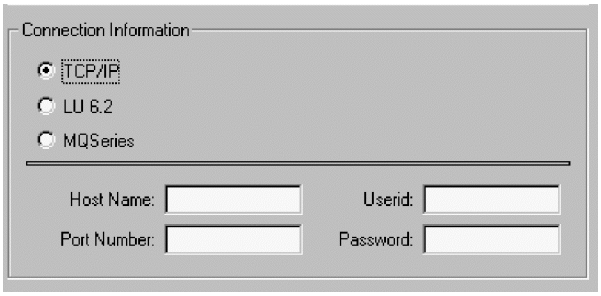


図 6. TCP/IP 接続タイプ

- 以下の情報を入力します。

  ホスト名 (必須): TCP/IP アドレス、または接続するホスト名。このフィールドには、ドット表記形式のホスト名ストリングか TCP/IP アドレスが含まれます。

  ポート番号 (必須): MAS Mainframe Agent が受信待機中のポート番号。この番号は Shadow Server を調査するとわかります。この番号は通常 1200 に設定されています。

  ユーザー ID: ご使用のメインフレーム・ユーザー ID。ユーザー ID を入力しないと、接続時に入力を要求されます。

  パスワード: ご使用のメインフレーム・パスワード。パスワード (画面上および `ODBC.INI` ファイル内の両方とも) は自動的に暗号化されます。パスワードを入力しないと、接続時に入力を要求されます。

LU 6.2:

- 「**LU 6.2**」を選択すると、図 7 に示したオプションが表示されます。

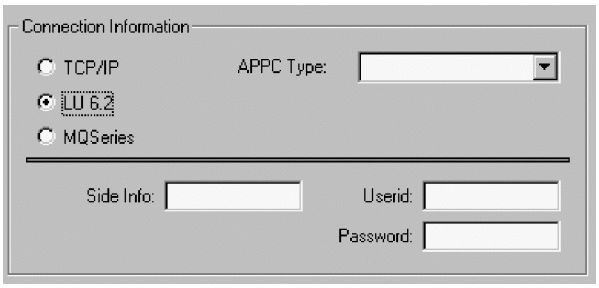


図 7. LU 6.2 接続タイプ

- 以下の情報を入力します。

  APPC タイプ: 表示されたドロップダウン・リストの APPC タイプ

  サイド情報 (必須): 表「*Side Info*」の項目。この表のサイド情報名の各値には、パートナー LU 名 、会話モード、およびトランザクション・プログラム名が含まれます。これらの値は、ホストとの LU 6.2 接続を確立するために使用します。

  ユーザー ID: ご使用のメインフレーム・ユーザー ID。ユーザー ID を入力しないと、接続時に入力を要求されます。

パスワード: ご使用のメインフレーム・パスワード。パスワード (画面上および ODBC.INI ファイル内の両方) は自動的に暗号化されます。パスワードを入力しないと、接続時に入力を要求されます。

WebSphere MQ:

- **「WebSphere MQ」**を選択すると、図 8 に示したオプションが表示されます。

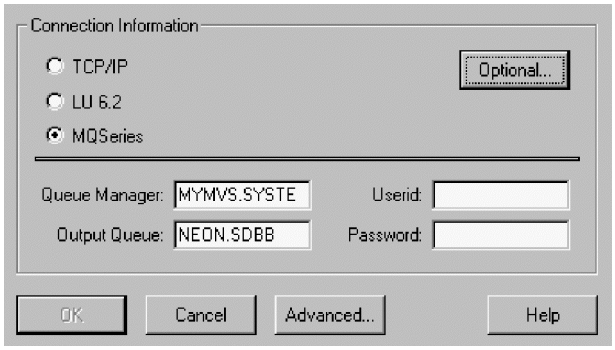


図 8. WebSphere MQ 接続タイプ

a. 以下の情報を入力します。

キュー・マネージャー (必須): キュー・マネージャーはクライアントとホストの両方に必要です。図 8 はクライアント・サイドを表しています。Windows 3.1 または Windows for Workgroups のようなクライアントがキュー・マネージャーをサポートできない場合、**「オプション (Optional)」**をクリックして **WebSphere MQ Client** および **MQSERVER** を定義する必要があります (8b 参照)。

出力キュー (必須): 要求が送信されるまで保管される場所。出力キューは必須です。サーバー側にキュー・マネージャーがある場合、サーバーに出力キューを定義する必要があります。

ユーザー ID: ご使用のメインフレーム・ユーザー ID。ユーザー ID を入力しないと、接続時に入力を要求されます。

パスワード: ご使用のメインフレーム・パスワード。パスワード (画面上および ODBC.INI ファイル内の両方) は自動的に暗号化されます。パスワードを入力しないと、接続時に入力を要求されます。

b. **WebSphere MQ Client** を定義する必要がある場合、**「オプション (Optional)」** ボタンをクリックします。**「WebSphere MQ の追加接続情報 (Optional WebSphere MQ Connection Information)」** ダイアログ・ボックス (38 ページの図 9) が表示されます。

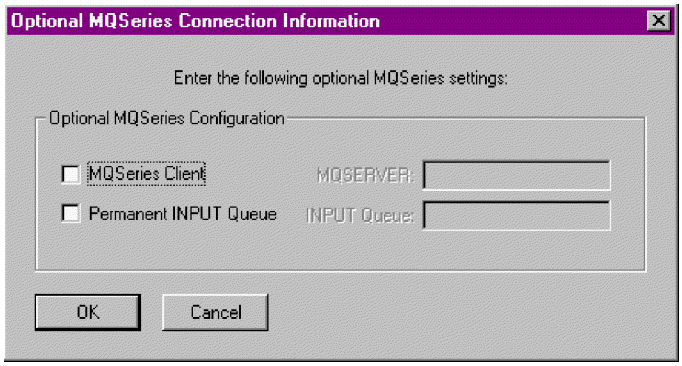


図 9. WebSphere MQ 接続情報

c. 以下の処理を行って適宜 WebSphere MQ 構成を定義します。

WebSphere MQ Client: このクライアントは、アプリケーションからの MQI 呼び出しの受信とサーバー・システム上のキュー・マネージャーとの通信が可能です。

MQSERVER: キュー・マネージャーが存在するサーバー。

d. **「OK」**をクリックして、 直前の画面に戻ります (34 ページの図 4 に示した構成ダイアログ・ボックス)。

9. **「拡張設定 (Advanced)」**をクリックします。図 10 に示した**「拡張情報 (Advanced Information)」** ダイアログ・ボックスが表示されます (選択したデータ・ソース・タイプに基づいているため、図に示したフィールドの一部が表示されない場合もあります)。

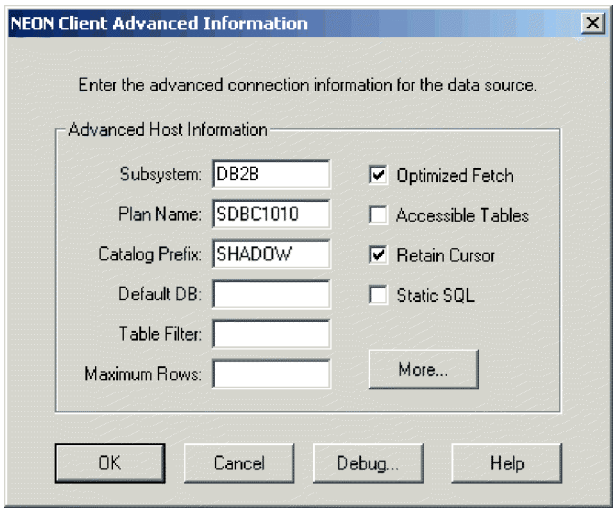


図 10. クライアント拡張情報

10. データ・ソース・タイプに ODBC を選択した場合、以下の情報を入力または選択します。

- **サブシステム:** 接続するホスト上にある DB2 のコピー。デフォルトは、データ・ソース名の最初の 4 文字です。 DB2 への接続をオープンしない場

合、NONE を指定します。これは、CICS/TS、IMS/TM、または ADABAS 用の MAS Connector、および DB2 への接続が不要なその他のアクセス・タイプを使用するお客様に適用できます。

- **計画名:** MAS はホスト上の計画名を使用して DB2 に接続します。この名前の最初の 2 文字は常に “SD” です。3 番目の文字は常に、Mainframe Agent サブシステム名 (デフォルトは “B”) の最後の文字です。 4 番目の文字は、計画がカーソル固定に設定されている場合は “C”、反復可能に設定されている場合は “R” となります。最後の 4 文字は常にバージョン番号 (例: 1010、1040 など) です。例えば、SDBC1010 です。このデフォルトはリリースによって異なります。

  注: 計画名の 4 番目の文字が R の場合、MAS ODBC ドライバーは、アプリケーションが反復可能読み取りの分離値を使用して計画が設定された箇所で、計画を使用しようとしているとみなします。反復可能読み取りを使用しない場合、デフォルトの計画 SDBR1010 のように、計画名の 4 番目の文字が R にならないようにしてください。4 番目の文字が R 以外の文字であれば、その計画はカーソル固定の分離値が設定されているとみなされます。

  ヒント: クライアントでサブシステムが DFLT に設定されていると、接続には Mainframe Agent に設定されている DEFAULTDB2SUBSYS パラメーターが使用されます。クライアントで計画名が DFLT か DEFAULT に設定されていると、接続には Mainframe Agent に設定されている DEFAULTDB2PLANNAME パラメーターが使用されます。

- **カタログ・プレフィックス (Catalog Prefix) :** 最適化カタログ機能を使用している場合**のみ**値を入力してください。カタログ・プレフィックス (Catalog Prefix) には、ODBC カタログ・クエリーをサポートするよう最適化された DB2 の表セットの所有者 (DB2 用語では許可 ID) が含まれている必要があります。このフィールドに値が指定されていない場合、標準のシステム・カタログ (`SYSIBM.SYS*`) が使用されます。

  注: WebSphere では、最適化カタログは、製品に含まれる ODBC アプリケーションの MAS Connector との円滑な動作に必要不可欠であると考えられています。

- **デフォルト DB:** ODBC アプリケーションを使用して DB2 に表を作成しなければならない場合、ご使用のユーザー ID に表作成の権限があるデータベースの名前を入力する必要があります。この値は CREATE TABLE ステートメントに追加されますが、表を作成するデータベースは指定されません。このステートメントに “IN DATABASE” 節が指定されていない場合、ドライバーによってこの節が追加されます。 DB2 のデフォルト・データベースに表を作成する権限を持たないユーザーではこの処理が**必須**ですが、それにはたいていのユーザーが含まれます。

- **表フィルター**: このフィールドは、後述するアクセス可能テーブル機能とともに、カタログ照会 (SQLTables、SQLTablePrivileges、および SQLColumns) から戻される情報を制限するのに使用します。 ODBC アプリケーションによって RDBMS インスタンスにあるすべての表のリストが要求されると、MAS が表フィルターの値に基づいてこのリストを作成します。第 1 レベル

の名前 (username) の下の表のみが返されます。このフィルターには 2 つの SQL ワイルドカード文字、 '%' と '_' が組み込まれています。ここで、'%' はゼロまたは 1 文字以上の置換文字、'_' は 1 文字の置換文字です。制限のない限り RDBMS は何千もの行を送信するため、このフィールドは便利です。

ヒント: 「表フィルター」フィールドの左方が空白の場合、このフィルターが `userid` の値のデフォルトとなります。つまり、使用している `userid` が所有する、または作成した表のみが MAS から返されます。フィルターが不要なことが明確な場合、このフィールドには "%" を入力してください。例: "ai38%" を指定すると、`ai38og.staff`、`ai38pds.staff`、および `ai38pds.xyz` が返されます。項目をコンマで区切ると、複数の表フィルターを指定できます。

- **最大行**: このフィールドには、1 つのクエリーから戻される行数を制限する整数が含まれます。値が入力されていないと、戻される行数は制限されません。
- **最適化フェッチ**: クエリーの処理速度を速めるためにドライバーのブロック・フェッチ機能を使用する場合、このボックスをチェックする必要があります。パフォーマンスは大きく向上します。

  注: ブロック・フェッチ機能の使用には制限事項が 1 つあります。ブロック・フェッチが使用可能な場合、カーソル・ベースの削除 (DELETE WHERE CURSOR OF) は使用できません。
- **アクセス可能テーブル**: 多くのアプリケーションには、接続されたデータ・ソースの表に関する情報が必要です。ただし DB2 の場合、この表リストは、サイズに制限を設定しないと管理不能なほど大きくなります。サイズに制限を設定するには、**「アクセス可能テーブル (Accessible Tables)」** ボックスを選択します。ユーザーが SELECT 権限を持つ表のみが戻されます。これには、PUBLIC および PUBLIC AT ALL LOCATIONS にある表が含まれますが、2 次権限 ID が SELECT 権限を持つ表は含まれません。
- **カーソルの保存 (Retain Cursor)**: このボックスは、DB2 COMMIT 操作がデータ・ソースのカーソルに及ぼす影響を制御します。COMMIT 操作を実行する前と同じ位置にカーソルを保存するには、**「カーソルの保存 (Retain Cursor)」**を選択する必要があります。これにより、再度ステートメントを準備しなくてもアプリケーションの実行およびフェッチが可能となります。デフォルト値は ON に設定されています。このボックスがチェックされている (これにより値は OFF に設定される) と、アプリケーションは COMMIT 操作の後にカーソルをクローズおよび削除します。この場合、次のステートメントを最初から準備して実行する必要があります。 この値に何を設定すればよいかわからない場合は ON (チェックする) に設定してください。

  注: カーソルの保存オプションを指定すると、アプリケーションによって自動コミットが使用可能に設定される限り、Mainframe Agent は SELECT ステートメントごとに COMMIT を発行できます。
- **静的 SQL**: MAS Connector を静的 SQL モードにするには、このボックスをチェックする必要があります。このモードでは、ドライバーに渡された各ク

エリーは、ホスト上の静的 SQL として変換および保管されます。このオプションを使用する前に、Dynamic-to-Static Analyzer を使用してアプリケーションを変換する必要があります。

11. **「詳細 (More)」**をクリックします。図 11 に示しているように、**「オプションの情報 (Optional Information)」**ダイアログ・ボックスが表示されます。

    このダイアログ・ボックスを使用して、キーワードの値を設定できます。

    注: 使用しているインターフェースによっては、このキーワードにデフォルト以外の値を設定する必要がある場合もあります。このキーワードとその値については 155 ページの『付録 C. MAS ドライバーのキーワード』 で説明しています。

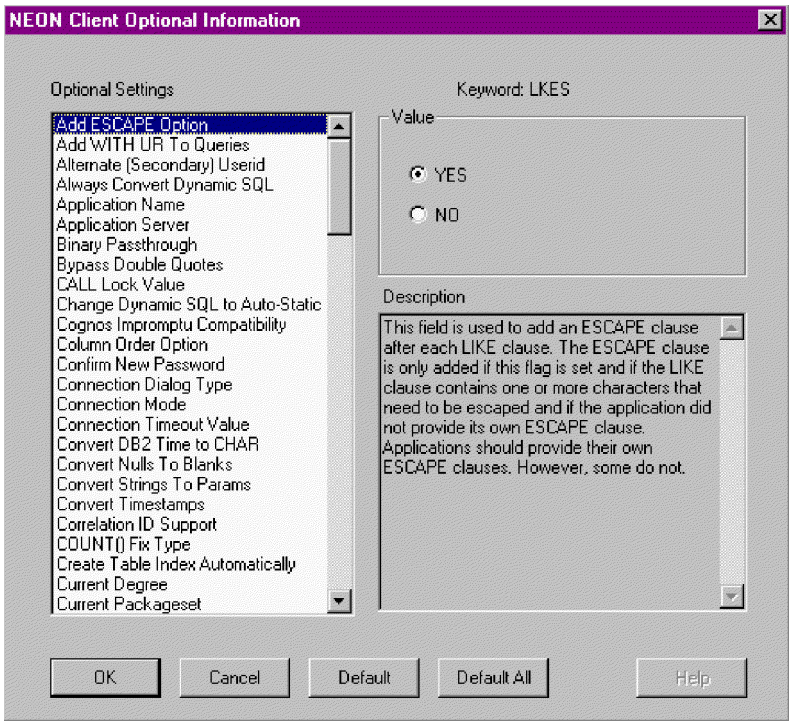


図 11. オプションの設定

12. 入力した値が ODBC.INI ファイルに書き込まれ、ODBC データ ソース アドミニストレータが終了するまで「OK」をクリックします。この値は次回データ・ソースに接続する際、デフォルトとして使用されます。

CICS/TS ステートメントおよび IMS/TM CALL ステートメントの構文

このセクションでは、CICS/TS ステートメントおよび IMS/TM CALL ステートメントの構文について説明します。

CICS/TS CALL 構文: CICS/TS ステートメントは、以下の構文を使用します。

```
CALL <CICS Stored procedure> (<parameters>)
```

例:

```
CALL CICSEX.NESPSEL ('logonid')
```

以下のような 4 つの列を持つ行が 1 行戻されます。

```
CA_IN_LID        CA_OUT_NAME        CA_OUT_PHONE        CA_OUT_EMAIL
         SCHFS                    Henry Sobieski
8005056366               HSOBIESKI@NEONSYS.COM
```

IMS/TM CALLSyntax: IMS/TM ステートメントは、以下の構文を使用します。

```
CALL <IMS Stored procedure> (<parameters>)
```

例:

```
call ims.imssp('','','','s','Sobieski', 'Henry','F','','','','','','','')
```

パラメーター 5 (D05751)、パラメーター 6 (D05757)、およびパラメーター 7 (D05752) にファーストネーム、ラストネーム、およびミドル・ネームのイニシャルを指定してレコードを検索します。

パラメーター 4 (REVFLD) は 's' 付きで取り込まれます。結果セットは、以下のようになります。

```
          ADDFLD                          CHGFLD                    REMFLD
REVFLD          D05751                  D05757              D05752
D05754          D05755                  D05756              D15333
D15334          D07786                              F00028
SOBIESKI        HENRY                   F          FAIRFAX           VA
           IBM    TEST          SDBB000          0SDBB00001246581
ITINERARY RETRIEVED
```

UNIX 用 ODBC データ・ソースの構成

UNIX用 ODBC ドライバーは UNIX の共有ライブラリーとして提供されますが、Windows 環境とは異なり、UNIX ドライバーには ODBC ドライバー・マネージャーは必要ありません。その代わり、.INI ファイルの編集のみが必要となります。ただし、ご使用のアプリケーションがサード・パーティーの ODBC マネージャーを必要とする場合、MAS がプロシージャーを提供します。各セクションの内容は、以下のとおりです。

- MAS UNIX ODBC ファイル構造
- .INI ファイルの編集による ODBC ドライバー・データ・ソースの構成方法
- サード・パーティーの ODBC ドライバー・マネージャーを使用可能にする方法
- サンプル・ファイル odbc.ini
- トレースを使用可能にする方法
- プログラミング上の注意

ODBC ドライバー・ファイルの構成

インストール・プロセスによって、shadow という名前のルート・ディレクトリーを持つディレクトリー・ツリーが作成されます。インストール済みファイルは untar されています。ディレクトリー構造は、以下のとおりです。

```
SHADOW - +--- SAMPLES (scodbcsm.c)
|  scimbmsm.c
|  scimccsm.c
|  schorpsm.c,
|  Readme files,
|  makefile)
|
+--- INCLUDE (header files for UNIX)
|
+--- LIB (shareable library)
|             libscodbc.so.1    (production version)
|             libscodbcts.so.1  (debug version)
|
+--- BIN (Dynamic-to-static)
|             dsa                 (dynamic-to-static)
|             scodbcdm executable (sample program - See
           appendix.)
```

odbc.ini ファイルの編集

SunOS/Solaris プラットフォームでデータ・ソースを構成するには、検索パスを指定し、テキスト・エディターを使用して odbc.ini ファイルを変更します。

1. 検索パスの指定

   MAS ランタイム・ライブラリーの検索パスが指定されていることを確認してください。例えば、SunOS および Solaris の csh を使用すると、LD_LIBRARY_PATH 環境変数を変更することによりライブラリーが存在するディレクトリー名を指定できます。

   ```
   setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/shadow/lib
   ```

2. .odbc.ini ファイルまたは odbc.ini ファイルを構成します。

   すべての ODBC データ接続情報が構成ファイルに保管されている必要があります。MAS は、.odbc.ini (UNIX ODBC ドライバー・マネージャー・ベンダーの多くが使用)、および odbc.ini という 2 つの命名規則をサポートします。この構成ファイルは、$HOME 環境変数で指定されているディレクトリーに保管されている必要があります。ドライバーは最初に .odbc.ini ファイル、次に odbc.ini ファイルを検索します。

   注: 構成ファイルのファイル名は小文字である必要があります。例えば、.odbc.ini というファイル名は有効ですが、.ODBC.INI は無効です。

ヒント: ODBC アプリケーションがデーモン・プロセスとして稼働している、またはマスター・プロセスによって作成される場合、ドライバーを呼び出すプロセスの内容に $HOME 環境変数が設定されません。この環境変数は、putenv(3C) を使用してプログラムで設定することや、親プロセスの内容に環境変数を設定することもできます。

サード・パーティーの ODBC ドライバー・マネージャーとリンクした UNIX ベース ODBC アプリケーションの使用

アプリケーションにサード・パーティーのドライバー・マネージャーが必要でランタイム・ライブラリーの依存関係が変更できない場合、アプリケーションで MAS ドライバーを使用するには、以下の処理を行う必要があります。

1. MAS ODBC ドライバーが正しく設定されていることを確認します。

2. 標準 MAS ドライバー・ライブラリーのドライバー・マネージャー・ライブラリー・ファイル名を使用して、2 つの新規シンボリック・リンクを作成します。このシンボリック・リンクの名前は、通常 libodbc.so と libodbcinst.so です。これには、次のものがあります。

   ```
   ln -s shadow/lib/libscodbcts.so shadow/lib/libodbc.so
   ln -s shadow/lib/libscodbcts.so shadow/lib/libodbcinst.so
   ```

3. LD_LIBRARY_PATH 変数を変更することによって、実際の ODBC ドライバーより先にこれら 2 つの新規シンボリック・リンクが検索できることを確認します。

odbc.ini サンプル・ファイル

odbi.ini ファイルの有効なキーワード・セクションを識別するには、DSN キーワードが必要です。

```
[ODBC Data Sources]
DB2.DEMO=Shadow Direct UNIX Debug Driver
DB2.TEST=Shadow Direct UNIX Debug Driver
VSAM.TEST=Shadow Direct UNIX Debug Driver

[DB2.DEMO]
Driver=<put the load path of the library here....>
Description=<your description here...>
APPL=ODBC
MR=0
HD=NO
CPFX=SHADOW
LINK=TCPIP
UID=<user id>
PWD=<password>
HOST=<hostname or IP address>
PORT=<server listening port>
CD=NO
PLAN=SDBC1010
SUBSYS=DSN2
# this is a comment line
# this is a comment line
NEONTRACE=INFO log=/tmp/neonlog.txt
```

UNIX odbc.ini ファイルの検索順序

以下は、odbc.ini ファイルの検索順序です。

1. /usr/local/lib/neon/odbc.ini
2. $ODBC_INI
3. $ODBC_INI (例: setenv ODBC_INI "/home/somebody/.my_odbc.ini")
4. $HOME/.odbc.ini
5. $HOME/.ODBC.INI
6. $HOME/odbc.ini
7. $HOME/ODBC.INI

ドライバーには、このファイルの READ 権限のみが必要です。

JDBC ドライバーの構成

MAS JDBC ドライバーは、2 つの異なるモードで作動できます。

- **タイプ 1:** このモードでは、MAS JDBC ドライバーは ODBC ドライバー・マネージャーと対話します。
- **タイプ 2:** このモードでは、MAS JDBC ドライバーは ODBC ドライバー・マネージャーをバイパスし、MAS ODBC ドライバーと直接対話します。

NEONTRACE 環境変数にキーワード TYPE1 または TYPE2 を入力すると、使用するモードが指定できます。 **モードを指定しないと、Windows NT プラットフォームでは TYPE1 が、SunOS プラットフォームでは TYPE2 が使用されます。**

MAS JDBC ドライバー接続ストリングのフォーマット

MAS JDBC ドライバー接続ストリングのフォーマットが完了していることを確認してください。driver.connect() にパスする接続ストリングのフォーマットは以下のとおりです。

```
jdbc:neon:<data-source-name>;
<attri1>=<value1>;...;<attrN>=<valueN>
```

ODBC 接続ストリングで有効なすべての属性は、JDBC 接続ストリングでも有効です。ただし、以下のような JDBC 固有の属性があります。

- **DRPM** は、ログオン情報に不足がある場合にドライバーからプロンプトを出力するかどうかを制御します。この属性には、以下のいずれかの値が入ります。
  - **NOPROMPT** (デフォルト) は SQL_DRIVER_NOPROMPT に対応。
  - **COMPLETE** は SQL_DRIVER_COMPLETE に対応。
  - **PROMPT** は SQL_DRIVER_PROMPT に対応。

この属性の例は、次のとおりです。

```
DRPM=COMPLETE
```

DBCS サポート

MAS JDBC ドライバーは、Windows NT および SUNOS で DBCS をサポートします。 Windows 95/98 では DBCS はサポートされません。

MAS ドライバー接続のテスト

VB Demo は、MAS Connector とともに提供される Windows のユーザー・インターフェースです。このインターフェースはデータ構造へのアクセスに使用します。以下のセクションで、VB Demo の使用方法について説明します。

- 概要
- 計画上の考慮事項
- VB Demo の管理
  - VB Demo の開始
  - SQL 要求の作成
  - VB Demo の終了

VB Demo の概要

VB Demo は、MAS のインストール時にコンピューターにインストールされます (9 ページの『第 2 章 コネクターのインストールと構成』 参照)。 デフォルトでは以下のように、C:¥ ドライブの「システム」フォルダーに保管されます。

```
IBM Shadow Client¥Shadow¥Bin32¥VBDemo32.exe
```

VB Demo では、Microsoft Access、Microsoft Excel、Crystal Reports、およびその他のさまざまなツールを使用して、DB2、IMS、CICS、および ADABAS データにアクセスできます。標準の SQL ステートメント呼び出しを使用して、IMS、ADABAS、および VSAM のデータにアクセスすることもできます。

また、VB Demo を使用して、ADABAS、DB2、IMS、CICS、および VSAM のシャドー・ダイレクト・インターフェースのインストールを検証することもできます。

計画上の考慮事項

始める前に、以下のアクションが完了していることを確認してください。

- Microsoft Visual Basic のインストール
- Microsoft Data Access Component (最新バージョン) のインストール
- 両コンポーネントの MAS へのインストール
- MAS の構成とデータ・ソースへの接続

VB Demo の管理

VB Demo の開始

VB Demo を稼働するには、以下のステップを実行します。

1. 「スタート」メニューを開いて、「プログラム」/「Neon システム (Neon Systems)」/「VB Demo 32-bi アプリケーション (VB Demo 32-bi application)」を選択します。

   VB Demo ウィンドウが表示されます。

2. **「接続 (Connections)」**メニューから、**「接続の新規追加 (Add New Connection)」を選択します。**

   注: 一度に複数の接続をオープンできます。

   47 ページの図 12 に示しているように、**「データ ソースの選択 (Select Data Source)」**ダイアログ・ボックスが表示されます。

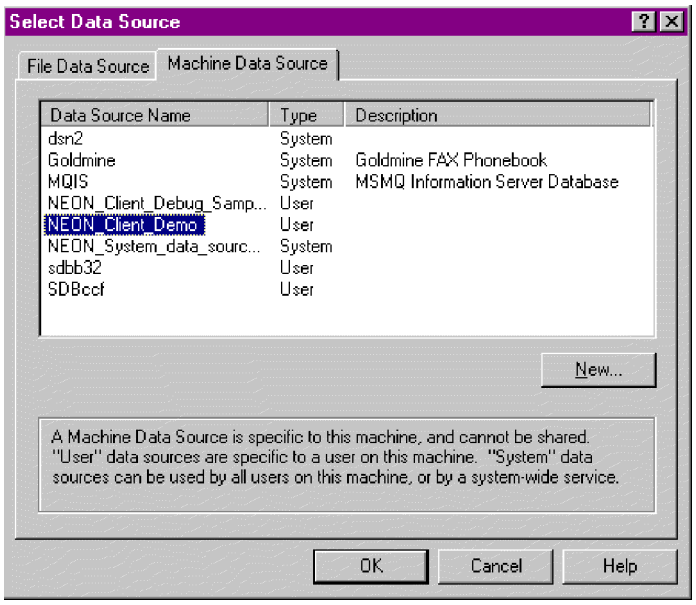


図 12. 「データ ソースの選択 (Select Data Source)」ダイアログ・ボックス

3. **「マシンのデータ ソース (Machine Data Source)」**タブを選択して、**「NEON_Client_Demo」**をクリックします。
4. **「OK」**をクリックします。**「ユーザー認証 (User Authentication)」**ダイアログ・ボックスが表示されます。
5. **「Userid」**フィールドにメインフレーム・ユーザー ID を入力し、**「Password」**フィールドにメインフレーム・パスワードを入力します。
6. **「OK」**をクリックします。接続が確立されたことを示すメッセージが発行されます。
7. **「OK」**をクリックして、**「VB Demo」**ウィンドウに戻ります。

SQL 要求の作成

SQL 要求を作成するには、以下のステップを実行します。

1. アクセスしているデータのタイプに従って、以下のステートメントのいずれかを入力します。

   DB2-OS/390 の場合:

   ```
   select * from q.staff
   ```

   注: 「テーブル (Tables)」ボタンを選択している場合、表フィルターに定義されている表のみが表示されます。表フィルターは、データ・ソース構成時に定義されました (30 ページの『ODBC ドライバーの構成』 および 44 ページの『JDBC ドライバーの構成』 参照)。表フィルターが定義されていないと、ユーザーが所有する、または使用しているユーザー ID で作成した表のみが表示されます。

   ADABAS の場合:

   ('select first_name birth sex from employees where last_name = "jones"')

CICS の場合:

```
CALL CICSEX.NESPSEL
```

IMS の場合:

IMS/TM ステートメントは以下の構文を使用します。

```
call ims.imssp('','','','s','Sobieski', 'Henry','F','','','','','','','')
```

パラメーター 5 (D05751)、パラメーター 6 (D05757)、およびパラメーター 7 (D05752) にファーストネーム、ラストネーム、およびミドル・ネームのイニシャルを指定して、レコードを検索します。パラメーター 4 (REVFLD) は 's' 付きで取り込まれます。

結果セットは以下のようになります。

```
          ADDFLD                    CHGFLD                    REMFLD
REVFLD          D05751                  D05757              D05752
D05754          D05755                  D05756              D15333
D15334          D07786                            F00028
SOBIESKI        HENRY                   F          FAIRFAX          VA
        IBM    TEST          SDBB000          0SDBB00001246581
ITINERARY RETRIEVED
```

2. 「照会 (Query)」をクリックします。 要求したデータのタイプに従って、以下の情報が表示されます。

VB Demo の終了

VB Demo を終了するには、以下のステップを実行します。

1. VB Demo プログラムを終了するには、**「切断 (Disconnect)」**をクリックします。接続がクローズされたことを示すメッセージが表示されます。
2. 「VB Demo」ウィンドウをクローズします。

トレース

MAS Trace Facility は、インストール時や使用時に発生する問題を解決するのに大変便利です。MAS ODBC ドライバーおよび MAS JDBC ドライバーの両方に対して、クライアント・サイドのトレースを使用可能にできます。以下のセクションでは、下記トピックについて説明します。

- 概要
- 計画上の考慮事項
  - データ・ソースの構成
  - パフォーマンスの問題
  - トレース・ログ・ファイルの制御
- MAS Trace Facility の使用可能化
  - MAS Trace Facility の制御

MAS Trace Facility の概要

MAS Trace Facility は、選択されたイベントをトレースします。この機能により、選択済みディレクトリーに NEONLOG.TXT という名前のファイルが作成され、そのファイルにトレースされたイベントが記録されます。すでに同じ名前のファイルが存在している場合、MAS Trace Facility は、そのファイルに新規記録されたイベントを追加します。

トレース出力データのフォーマットには常に、各イベントの日付、時間、および関連情報が含まれます。

トレース出力データの例

```
Fri Oct 01 22:19:30 1993   pcbColName       = 0x0a5f:759a
Fri Oct 01 22:19:30 1993   pfSqlType        = 0x0a5f:759e
Fri Oct 01 22:19:30 1993   pcbColDef        = 0x0a5f:75a0
Fri Oct 01 22:19:30 1993   pibScale         = 0x0a5f:7594
Fri Oct 01 22:19:30 1993   pfNullable       = 0x0a5f:759c
Fri Oct 01 22:19:30 1993 SQLDescribeCol exiting - return = SQL_SUCCESS(0)
Fri Oct 01 22:19:30 1993   szColName        = 'REMARKS'
Fri Oct 01 22:19:30 1993   *pcbColName      = 7
Fri Oct 01 22:19:30 1993   *pfSqlType       = SQL_CHAR(1)
Fri Oct 01 22:19:30 1993   *pcbColDef       = 64
Fri Oct 01 22:19:30 1993   *pibScale        = 9999
Fri Oct 01 22:19:30 1993   *pfNullable      = SQL_NULLABLE_UNKNOWN(2)
Fri Oct 01 22:19:30 1993 SQLFetch entered
Fri Oct 01 22:19:30 1993   lpstmt           = 0x095f:0000
Fri Oct 01 22:19:30 1993 internal error detected: file scodbcre.c line 1228 rc =
 0 from scclxlat
Fri Oct 01 22:19:30 1993 SQLFetch exiting - return = SQL_SUCCESS(0)
Fri Oct 01 22:19:30 1993 SQLGetData entered
Fri Oct 01 22:19:30 1993   lpstmt           = 0x095f:0000
```

計画上の考慮事項

MAS Trace Facility を使用する前に考慮すべき重要な要因がいくつかあります。それらの要因は以下のとおりです。

- データ・ソースの構成
- パフォーマンスの問題
- トレース・ログ・ファイルの制御

データ・ソースの構成

動的データ・ソースが構成済みであることを確認してください (30 ページの『ODBC ドライバーの構成』を参照)。

パフォーマンスの問題

MAS Trace Facility は、ODBC アプリケーションを低速で実行することができるデバッグ・ツールです。このため、ドライバーやアプリケーションが完璧に作動するようになったら、MAS Trace Facility をオフに設定することもできます。DOS 環境変数 SET NEONTRACE=NONE を設定すると、MAS Trace Facility をオフに設定できます。

トレース・ログ・ファイルの制御

NEONLOG.TXT ファイルの完全修飾パス名 (ディレクトリーを含む) が指定されていないと、MAS Mainframe Agent は NEONLOG.TXT ファイルを現行ディレクトリーに作

成します。このようにすると、いずれは複数のディレクトリーに複数の NEONLOG.TXT ファイルができることになります。イベント・トレースの重大度レベル・トリガーを低く設定している場合は特に、これらのファイルは非常に大きくなる可能性があります。 必要に応じてこれらのファイルを削除してください。

Windows 用 MAS ODBC ドライバー・トレースの使用可能化

1. **ODBC データ ソース アドミニストレータ**を開きます。
2. **「トレース」** タブを選択します。
3. **「トレースの開始」**をクリックします。
4. **「OK」**をクリックします。

Windows 用 MAS JDBC ドライバー・トレースの使用可能化

1. 50 ページの『Windows 用 MAS ODBC ドライバー・トレースの使用可能化』のステップ 1 からステップ 3 に従ってください。
2. DOS の NEONTRACE 環境変数に以下の項目を追加してください。

   ```
   JDBCLOG=xyz
   ```

   xyz は、ログ・ファイルの絶対パス名です。例えば、Windows NT では

   ```
   THREADID INFO BUFFER JDBCLOG=C:¥temp¥neonjdbc.txt LOG=C:¥temp¥neonlog.txt
   ```

   となります。

   注: JDBCLOG キーワードは LOG keyword の前にある必要があります。

   2 つの分離したファイルが作成されます。1 つは ODBC ドライバー用で、もう 1 つは JDBC ドライバー用です。
3. **「OK」**をクリックします。

Windows NT における JDBC トレースの使用

TYPE1 モード (デフォルト) で実行している場合に Windows NT 環境でトレースを実行するには、データ・ソースで指定したトレース・ファイル名が、LOG キーワードで指定したファイル名と同じであることを確認してください。NEONTRACE パラメーターで LOG キーワードを省略すると、トレースはデータ・ソースで指定したトレース・ファイルに記録されます。

MAS Trace Facility の制御

たいていのケースでは、MAS Trace Facility にはデフォルトで適切なトレースが設定されています。ただし、MAS Trace Facility の重大度レベル・トリガー、またはトレース・オプションを変更する必要があります。以下の方法のいずれかを使用します。

- **「デバッグ情報 (Debug Information)」**ダイアログ・ボックスを使用して、オプションを選択します。
- DOS の NEONTRACE 環境変数に適切な値を設定します。

デバッグ・ダイアログ・ボックスの使用

MAS Trace Facility を制御するのに最も一般的に使用される方法は、**「デバッグ情報 (Debug Information)」**ダイアログ・ボックスを使用する方法です。このボックスには、使用可能なすべてのオプションが表示され、ODBC.INI ファイルの正しいキーワードが変更されたことが確認できます。

1. **「スタート」**メニューを開いて、Windows 95 以上を使用している場合は**「設定」/「コントロール パネル」**を、Windows NT 3.0 以上を使用している場合は**「設定/コントロール パネル/管理ツール」**を選択します。
2. **「ODBC データ ソース」**アイコン (「一覧」または「詳細」で表示している場合はアイテム) をダブルクリックします。

   「ODBC データ ソース アドミニストレータ」ダイアログ・ボックスが表示されます。
3. **「ユーザー データ ソース」** リストから、適切なデータ・ソースを選択します。
4. **「構成」**をクリックします。 図 13 に示しているように、構成ダイアログ・ボックスが表示されます。

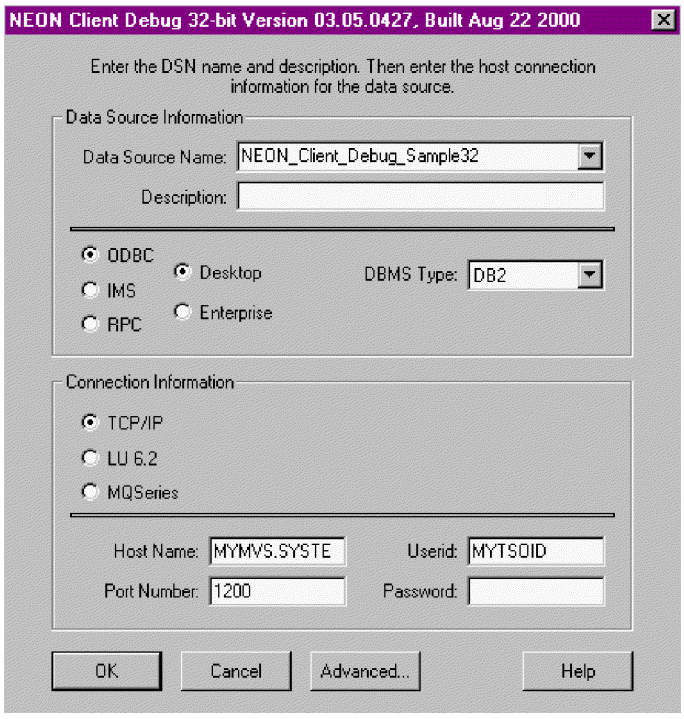


図 13. デバッグ構成

5. **「拡張設定」**をクリックします。**「拡張情報 (Advanced Information)** ダイアログ・ボックス (52 ページの図 14) が表示されます。

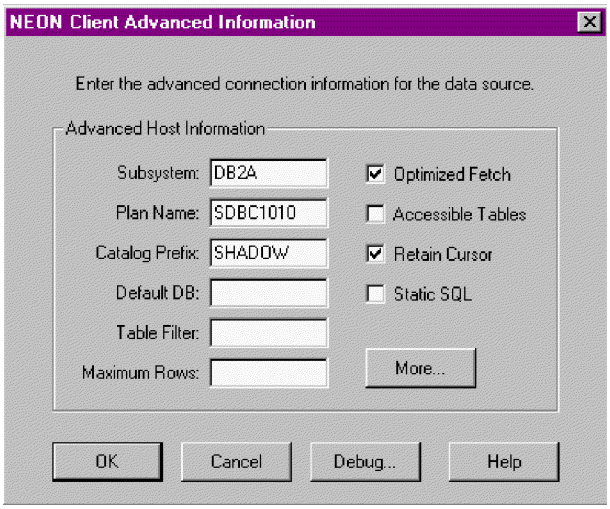


図 14. 拡張情報 (Advanced information)

6. **「デバッグ (Debug)」**をクリックします。**「デバッグ情報 (Debug Information)」**ダイアログ・ボックス (図 15) が表示されます。

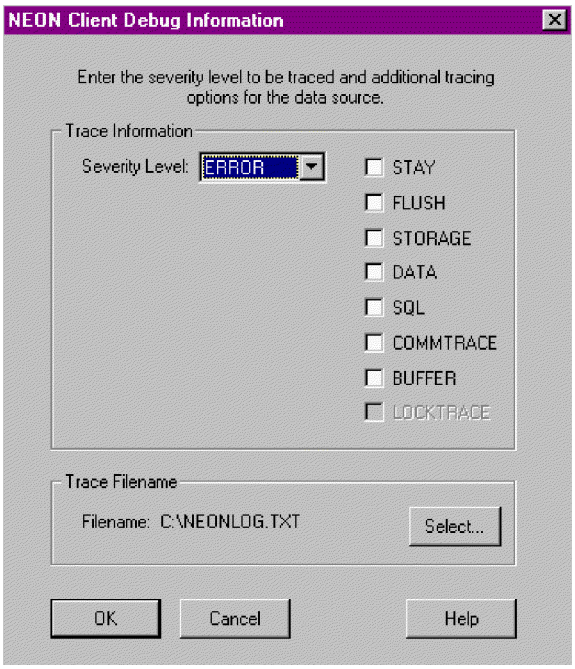


図 15. クライアントのデバッグ情報

注: 51 ページの図 13 に示した構成ダイアログ・ボックスでクライアント・タイプに IMS を選択した場合、図 14 に示した「拡張情報 (Advanced Information)」ダイアログ・ボックスに「デバッグ (Debug)」ボタンが表示されます。

7. トレースするイベントの適切な重大度レベル・トリガーを選択します。重大度レベルを選択すると、その重大度レベル以上のすべてのイベントがトレースおよび記録されます。重大度オプションの最高レベルから最低レベルまでを以下にリストします。

- **NONE:** このオプションを指定すると、メッセージのトレースおよび記録は行われません。トレース・ファイルの作成およびオープンも行われません。このオプションにより、効率的に MAS Trace Facility をオフに設定できます。
- **FATAL:** このオプションは最高レベルの重大度で、プログラムの終了メッセージのみがトレースおよび記録されます。
- **SEVERE:** (デフォルトの選択内容) このオプションを指定すると、重大レベル以上のイベントのみがトレースおよび記録されます。
- **ERROR:** このオプションを指定すると、すべてのエラー・メッセージがトレースおよび記録されます。これらのメッセージには通常、エラーの原因、エラーが検出されたレコード、関連する戻りコード、および詳細のエラー・テキスト・メッセージが含まれています。
- **WARNING:** このオプションを指定すると、MAS ODBC ドライバーによって発行されるすべての警告メッセージがトレースおよび記録されます。
- **INFO:** このオプションを指定すると、すべての通知イベントがトレースされます。これには、MAS ODBC ドライバーに渡される値や MAS ODBC ドライバーから戻される値が含まれます。このオプションを指定するとトレースおよび記録される情報量が多くなるため、通常の使用にはお勧めできません。ただし、アプリケーションのデバッグにはこのオプションが便利です。
- **DETAIL:** このオプションは最低レベルの重大度で、関数の呼び出しイベントを含むすべてのイベントがトレースされます。この値を指定すると大量の情報がトレースおよび記録されるため、通常の使用にはお勧めできません。

8. 適切なボックスをクリックして、追加オプションを選択します。追加オプションには、以下が含まれます。
   - **STAY:** このオプションを指定すると、トレース・ファイルがオープンされたままになります。トレース・メッセージの保管にディスク・ファイルを使用している場合、このオプションによってパフォーマンスを向上させることができます。
   - **FLUSH:** このオプションは STAY とともに使用します。このオプションを指定すると、マシンに障害が起こってもメッセージを逸失することはありません。
   - **STORAGE:** このオプションを指定すると、ストレージへのすべての GET および FREE 操作がトレースされます。トレース・レコードには、ストレージ域の要求または解放を行うファイルの名前と、ファイル内の行番号が示されます。さらに、このレコードには、取得または解放されるデータ域のサイズ、データ域のアドレス、およびその他の重要な情報も示されます。最終のストレージ報告書は、ストレージ・マネージャー機能が含まれる DLL のアンロード時に Windows 環境で生成されます。
   - **DATA:** このオプションを指定すると、DB2、ODBC C、および ODBC SQL データ・タイプのすべての変換操作がトレースされます。各ケースでは、入出力データは 16 進および文字フォーマットで表示されます。さらに、現在処理中の表の行および列を指定する情報が提供されます。
   - **SQL:** このオプションを指定すると、トレース重大度レベルが INFO より高い場合も、ホストに渡されたすべての SQL ステートメントがトレースされます。また、SELECT ステートメントの列情報 (列番号、列名、およびデー

タ・タイプ) も表示されます。データ・タイプが SQL_NUMERIC および SQL_DECIMAL の場合は、精度と位取りの値も表示されます。

9. 「**選択 (Select)**」をクリックします。「**トレース・ファイルの選択 (Select Trace File)**」ダイアログ・ボックス (図 16) が表示されます。

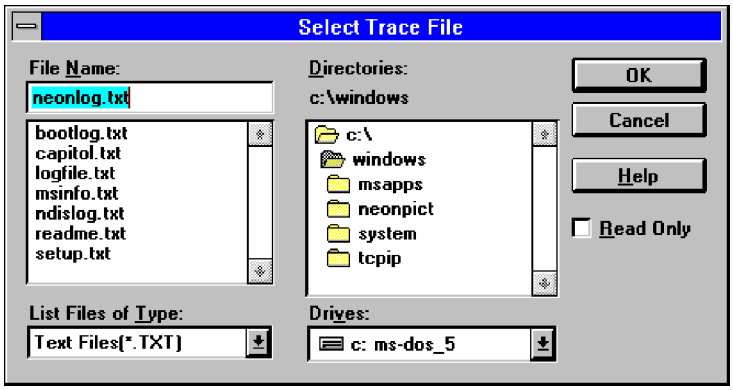


図 16. トレース・ファイルの選択 (Select Trace File)

10. トレース・ファイルの名前およびディレクトリーを選択します。
11. 「**OK**」をクリックして、「**デバッグ情報 (Debug Information)**」ダイアログ・ボックス (52 ページの図 15 に示したダイアログ・ボックス) に戻ります。
12. 「**OK**」をクリックします。

    これにより、ODBC.INI ファイルの適切なセクションに NEONTRACE キーワードが記述されます。 NEONTRACE キーワードには、**ODBC データ ソース アドミニストレータ**で選択したデータ・ソースで指定したオプションが反映されています。 ODBC.INI ファイルの内容によって、SET NEONTRACE=[string] コマンドを使用して以前に設定された DOS 環境変数の値が追加または指定変更されます。

DOS のトレース環境変数の設定

MAS Trace Facility を制御するその他の方法には、DOS のトレース環境変数を設定する方法があります。この環境変数を設定するには、以下のフォームを使用します。

```
SET NEONTRACE=[string]
```

[string] が、スペースで区切られた選択オプションで構成されている場合

以下のいずれかの方法でこの変数を設定します。

- DOS コマンド行から SET NEONTRACE=[string] コマンドを手動で入力
- AUTOEXEC.BAT および CONFIG.SYS ファイルに SET NEONTRACE=[string] コマンドを追加。コマンドを有効にするために、サーバーをリブートします。

注: コマンドを手動で入力する場合、Windows が稼働している間は必ず DOS でコマンド入力を行ってください。 DOS ウィンドウからは入力しないでください。DOS ウィンドウから環境変数のトレースを設定すると (2 次レベル・コマンド・プロセッサーを使用)、MAS Trace Facility には反映されません。

第 4 章 コネクターの構成

この章では、MAS コネクターの構成方法、およびコネクターと連携させるためのアプリケーションの構成方法を説明します。本章の内容は、次のとおりです。

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、アプリケーション/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。

- レガシー・アプリケーション・データベース内にイベント表とアーカイブ表を作成します。
- ユーザーのサイトで稼働しているコラボレーションに必要なビジネス・オブジェクトがサポートされるよう、アプリケーションの表上にデータベース・トリガーをインストールします。WebSphere は独自のデータベース・トリガーを開発することを前提としています。
- 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。

以後の各セクションでは、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` プロパティーが `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。また、この情報を使用してアプリケーションからエンティティー全体を取り出します。イベントが最初にログに記録された後でエンティティーが変更されると、コネクターは初期イベントとその後のすべての変更を取得します。つまり、コネクターがイベント表からエンティティーを取得する前にエンティティーが変更および更新されると、コネクターは 1 回の取り出しで両方のデータ変更を取得します。

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 通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
- イベントのサブスクライブなし

コネクターがイベントを取り出した後でイベント表からイベントが削除されないと、不必要なスペースを占有することになります。しかし、そのようなイベントが削除される場合には、未処理のイベントもすべて失われ、イベント処理の監査を実行できなくなります。そのため、この場合にはアーカイブ表も作成し、ArchiveProcessed プロパティーを true に設定したままにしておくことをお勧めします。これにより、イベント表からイベントが削除されると、必ず、コネクターによってアーカイブ表にそのイベントが挿入されます。

注: コネクターは、イベント表からイベントを削除する際、またはアーカイブ表にイベントを挿入する際に、アプリケーション・データベースへのアクセス中に発生した問題が原因で失敗した場合、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、構成プロパティーを使用して、以下の情報を指定する必要があります。

- イベント表の名前 (EventTableName)。ビジネス・オブジェクトからの要求を処理するためだけにコネクターを使用する場合は、このプロパティーの値を指定する必要はありません。
- 間隔 (頻度) (PollFrequency)。
- ポーリング間隔ごとのイベントの数 (PollQuantity)。
- アーカイブ表の名前 (ArchiveTableName)。
- アンサブスクライブされたイベントおよび未処理のイベントをコネクターがアーカイブするかどうか (ArchiveProcessed)。
- コネクターの固有 ID。この ID は、複数のコネクターが同じ表をポーリングするときに重要になります (ConnectorID)。

また、EventKeyDel プロパティーの値を指定して、イベントの処理順序を指定することも可能です。これらのプロパティーおよび他の構成プロパティーに関する詳細については、58 ページの表 23 を参照してください。

注: イベント表とアーカイブ表の作成は、オプションです。ただし、EventTableName の値を指定しているにもかかわらず、コネクターを使用してイベントをポーリングせず、イベント表も作成しない場合には、コネクターでタイムアウトが発生します。そうしたタイムアウトを回避するには、EventTableName の値をヌル (string) のままにしておいてください。

デフォルトでは、イベント・キュー表の名前は、xworlds_events になり、アーカイブ・キュー表の名前は、xworlds_ archive_events になります。

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。

コネクターを要求処理用にのみ使用する場合は、コネクターを始動するときや EventTableName の値をヌル (string) に設定するときに、-fno オプションを使用してください。

ご使用のドライバーが Java クラス DatabaseMetaData をサポートしておらず、コネクターでイベント表とアーカイブ表の有無の検査が行われないようにする必要がある場合には、CheckForEventTableInInit の値を false に設定して、このプロパティーを使用不可にします。デフォルトでは、このプロパティーは true に設定されています。

イベントおよびアーカイブ表スキーマ

表 22 に、イベント表とアーカイブ表の列を示します。以下の情報を、表の列内の順序とデータ型に特に注意して、これらの表の作成およびインストール時のガイドとして使用してください。

表 22. イベントおよびアーカイブ表スキーマ

名前	説明	型	制約
event_id	イベントの内部 ID	INTEGER	Primary key
connector_id	イベントの宛先コネクターの固有 ID。この値は、複数のコネクターが同じ表をポーリングする場合には重要です。	VARCHAR	
object_key	ビジネス・オブジェクトの基本キー。複数のキーを、コロンなどの構成可能な区切り文字で連結します。例: 1000065:10056:2333 詳細については、DateFormat プロパティーを参照してください。	VARCHAR	非ヌル
object_name	ビジネス・オブジェクトの名前	VARCHAR	非ヌル
object_verb	イベントに関連付けられている動詞	VARCHAR	非ヌル
event_priority	イベント優先順位 (0 が最高、n が最低) で、コネクターが優先順位に基づいてイベントを取得するために使用します。コネクターが、この値を使用して、優先順位を上下させることはありません。	INTEGER	非ヌル
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 (InterChange Server へのイベントの送信エラー) -1 (イベント処理エラー) 0 (ポーリング開始可能) 1 (InterChange Server に送信済み) 2 (ビジネス・オブジェクトのサブスクリプションなし)	INTEGER	非ヌル

表 22. イベントおよびアーカイブ表スキーマ (続き)

名前	説明	型	制約
	3 (処理中)。この状況は、イベント表にのみ使用され、アーカイブ表には使用されません。		
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	

コネクターの構成

コネクターには、標準構成プロパティーとコネクター固有の構成プロパティーの 2 種類の構成プロパティーがあります。コネクターを実行する前に、必ずこれらのプロパティーの値を設定してください。コネクターの構成プロパティーを設定するには、以下のツールのいずれかを使用します。

- Connector Configurator (統合ブローカーが ICS の場合) — このツールには System Manager からアクセスします。
- Connector Configurator (WebSphere MQ Integrator Broker が統合ブローカーの場合) — このツールには、WebSphere Business Integration Adapter のプログラム・フォルダーからアクセスします。詳細については、137 ページの『付録 B. Connector Configurator』を参照してください。

標準コネクター・プロパティー

標準構成プロパティーは、すべてのコネクターによって使用される情報を提供します。これらのプロパティーの詳細については、119 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

注: このコネクターは、ICS と WebSphere MQ Integrator Broker の両方を統合ブローカーとしてサポートしているので、どちらのブローカーに関する構成プロパティーとも関係があります。

コネクター固有の構成プロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。コネクター固有のプロパティーを使用すると、エージェントの再コード化と再構築をしなくても、コネクター内で静的な情報または論理を変更することもできます。

表 23 に、コネクターに対するコネクター固有の構成プロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

表 23. コネクター固有のプロパティー

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

表 23. コネクター固有のプロパティー (続き)

名前	指定可能な値	デフォルト値	必須
ArchiveTableName	*アーカイブ・キュー表の名前*	xworlds_archive_events	ArchiveProcessed が true の場合は 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	*コネクターの固有 ID*	null	いいえ
DatabaseURL	*データベース・サーバーの名前*		はい
DateFormat	*時刻パターン・ストリング*	MM/dd/yyyy HH:mm:ss	いいえ
DriverConnectionProperties	*追加のドライバー接続プロパティー*		いいえ
EventKeyDel	*イベント表のイベント・キー欄に対する区切り文字*	セミコロン (;) コロン (:) 等記号およびセミコロン (=;) 等記号およびコロン (=:)	いいえ
EventOrderBy	none または *列名* [, *列名*, ...]		いいえ
EventQueryType	Fixed または Dynamic	Fixed	いいえ
EventTableName	*イベント・キュー表の名前*	xworlds_events	はい (ポーリングが必要な場合)。ポーリングが不要な場合は (string) null
JDBCDriverClass	ドライバー・クラス名		いいえ
MaximumDatabaseConnections	*同時データベース接続数*	5	はい
PingQuery	SELECT 1 FROM<表名>		いいえ
PollQuantity	1 から 500 までの値	1	いいえ
PreserveUIDSeq	true または false	true	いいえ
RDBMS.initsession	*各データベース・セッションを初期化する SQL ステートメント*		いいえ
RDBMSVendor	その他		いいえ
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: System Manager では、これらの文字は、順に単一引用符、コンマ、および二重の単一引用符を表します。	いいえ
RetryCountAndInterval	*カウント、秒単位の間隔*	3,20	いいえ
SPBeforePollCall	ポーリング呼び出しごとに実行されるストアード・プロシージャーの名前		いいえ
SPBeforePollCall	ReplaceStrList プロパティーに使用される文字区切り文字および終了区切り文字	,:	いいえ
TimingStats	0、1、2	0	いいえ

表 23. コネクター固有のプロパティー (続き)

名前	指定可能な値	デフォルト値	必須
UniqueIDTableName	*ID* の生成に使用するテーブルの名前	xworlds_uid	いいえ
UseDefaults	true または false	false	はい
UseDefaultsForCreatingChildBOs	true または false	false	いいえ
UseDefaultsForRetrieve	true または false	false	いいえ

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

ApplicationPassword
WebSphere 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
WebSphere 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。

このプロパティーを true または false に設定すると、イベント表からイベントが削除された後、アーカイブ表にそのイベントが挿入されます。

このプロパティーを true または false に設定すると、コネクターはアーカイブ処理を実行しません。この場合、ArchiveTableName プロパティーの値は検査されません。ArchiveProcessed が false に設定されている場合、コネクターは次のように動作します。

- イベントが正常に処理された場合、イベント表からそのイベントを削除しますが、アーカイブは行いません。
- コネクターがそのイベントのビジネス・オブジェクトにサブスクライブしていない場合は、イベントをイベント表に残し、そのイベントの状況を Unsubscribed に変更します。
- ビジネス・オブジェクトの処理中に問題が発生した場合、イベントをイベント表に残し、イベントの状況を Error にします。

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

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

デフォルト値は true です。

ArchiveTableName
アーカイブ・キュー表の名前。

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

デフォルトの名前は xworlds_archive_events です。

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

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

デフォルト値は false です。

CheckforEventTableInInit
このコネクター・プロパティーを false に設定すると、コネクターは、コネクターの初期化時に、イベント表とアーカイブ表が存在するかどうかの確認を行わなくなります。使用する 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、このプロパティーは常に true に設定しておくことをお勧めします。

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

デフォルト値は true です。

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

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

データベース内の該当するデータ・レコードを、状況列で適切な値に設定することにより論理的に削除させるには、このプロパティーの値を false に設定します。コネクターは、ビジネス・オブジェクト・レベルのアプリケーション固有の情報に指定された StatusColumnValue (SCN) パラメーターから、状況列の名前およびその値を取得します。詳細については、 102 ページの『ビジネス・オブジェクト・レベルのアプリケーション固有情報』を参照してください。

デフォルト値は false です。

CloseDBConnection

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

デフォルト値は false です。

ConnectorID

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

デフォルト値は null です。

DatabaseURL

コネクターの接続先データベース・サーバーの名前です。MAS ODBC ドライバー・データ・ソースを構成したときに指定した名前です。

以下の例で説明します。

```
jdbc:neon:mydatasource
```

mydatasource はデータベース・サーバーの名前、つまり DatabaseURL の値です。

コネクターが処理を正常に実行するためには、ユーザーがこの値を指定しておく必要があります。データ・ソース名の構成に関する詳細については、 32 ページの『Windows 用 ODBC ドライバー・データ・ソースの構成』および 42 ページの『UNIX 用 ODBC データ・ソースの構成』を参照してください。

DateFormat

コネクターで受信および戻すことができる日付形式を指定します。このプロパティーは、 63 ページの表 24 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。

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

表 24. 時刻形式構文

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 (7 月の 2 度目の水曜日)
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 Std 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 25. US 標準使用時の実例

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

DriverConnectionProperties

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

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

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

例えば、JDBC ドライバーで、ライセンス情報とポート番号が必要になるとします。ライセンス情報として要求されるプロパティー名は MyLicense であり、値は ab23jk5 です。ポート番号として要求されるプロパティー名は PortNumber であ

り、値は 1200 です。DriverConnectionProperties は値 MyLicense=ab23jk5;PortNumber=1200 に設定しなければなりません。

EventKeyDel

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

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

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

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

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

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

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

EventOrderBy

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

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

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

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

このプロパティーのデフォルト値はありません。

EventQueryType
EventQueryType プロパティーは、イベント表のイベントの検索の際に、コネクターに照会を動的に生成させるか、または コネクターの組み込みの照会を使用させるかを指定するために使用します。動的に生成された照会に関しては、コネクターはそのイベント構造をイベント表の列にマップします。表列内のデータの順序は、非常に重要です。正しい順序については、57 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 (string) の場合、デフォルトの照会が実行されます。Dynamic (string) に設定されている場合は、『EventTableName』プロパティーに指定されている表から列名を取得して、新規の照会が作成されます。

イベント表列名は変更できます。ただし、列の順序とデータ型は、イベント表の作成のセクションで指定したものと同じでなければなりません。デフォルトの照会または動的に生成された照会には、 64 ページの『EventOrderBy』が追加されます。

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

デフォルト値は Fixed (string) です。

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

デフォルトの名前は xworlds_events です。

コネクターのポーリングをオフにする場合は、このプロパティーを null (string) 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。

詳細については、 57 ページの表 22 を参照してください。

デフォルト値はありません。

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。MAS コネクターの場合、次のように指定します。

```
com.neon.jdbc.Driver
```

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。

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

デフォルト値は 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 ステートメントまたはストアード・プロシージャーを指定します。

次に示すのは、ping 照会として使用される SQL ステートメントの一例です。

```
SELECT 1 FROM <表名>
```

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

```
call sampleSP( )
```

ストアード・プロシージャー呼び出し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。例えば、次のようになります。

```
Call checkproc(2)
```

デフォルト値はありません。詳細については、7 ページの『データベース接続不能の処理』を参照してください。

注: PingQuery は CICS アプリケーションおよび IMS アプリケーションではサポートされません。

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。許容値は 1 から 500 です。

デフォルト値は 1 です。

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

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

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

注: このプロパティーがコネクター構成に追加されていない場合のデフォルトの動作は、このプロパティーが追加され、`true` に設定されている場合の動作と同じです。また、61 ページの『AutoCommit』が `true` に設定されている場合は、コネクターは `PreserveUIDSeq` が `false` に設定されている場合と同様に振る舞います。

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

デフォルト値は true です。

RDBMS.initsession

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

デフォルト値はありません。

RDBMSVendor

特殊な処理の際に、コネクターに使用させる RDBMS を指定します。このプロパティーを MAS コネクター用に定義する必要はありません。

デフォルト値はありません。

ReplaceAllStr

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

ReplaceAllStr のデフォルト値は false です。

ReplaceStrList

1 つの置換対象文字、1 つの文字区切り文字、および 1 つの置換ストリングで構成された置換セットを、1 つ以上指定します。属性の AppSpecificInfo プロパティーの ESC=[true|false] パラメーターの値、またはコネクターの ReplaceAllStr プロパティーの値が指定されている場合にのみ、コネクターは属性値に対してこの置換を実行します。

この属性の構文は、次のとおりです。

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

ここで、以下のように説明されます。

single_char 置換対象文字。

substitution_str コネクターが置換対象文字の置換に使用する置換ストリング。

, 置換対象文字と置換ストリングを区切る、文字区切り文字。デフォルトでは、文字区切り文字はコンマ (,) です。この区切り文字を構成するには、StrDelimiter プロパティー内の最初の区切り文字を設定します。

: 置換セット (1 つの置換対象文字、1 つの文字区切り文字、および 1 つの置換ストリングで構成されたセット) を区切る終了区切り文

字。デフォルトでは、終了区切り文字はコロン (:) です。この区切り文字を変更するには、StrDelimiter プロパティーの 2 番目の区切り文字を設定します。

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

```
%,%%:^,¥^
```

注: System Manager は、単一引用符を入力できないように制限されています。このため、単一引用符は文字 Q で表す必要があります。また、二重の単一引用符は文字 DSQ で表す必要があります。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。

回数、間隔 (秒) というフォーマットで値を指定します。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。

SPBeforePollCall

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

注: ポーリング呼び出しが途中で失敗した場合 (データベースがダウンしている場合や、接続が失われた場合) には、connector-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、connector-id を定期的に null にリセットすることをお勧めします。

StrDelimiter

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

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

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

デフォルト値は、コンマとその直後に続くコロン (,:) です。

TimingStats

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

0 (タイミング統計なし)

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

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

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

デフォルト値は 0 です。

UniqueIDTableName

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

デフォルト値は xworlds_uid です。

UseDefaults

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

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

デフォルト値は false です。

UseDefaultsForCreatingChildBOs

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

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

UseDefaultsForRetrieve

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

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

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

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

コネクターの複数インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

2. この始動スクリプトを、『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。

4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 26 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 26. コネクターの始動スクリプト

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

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

- Windows システムで**「スタート」**メニューから。

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

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

    connector_manager_*connName* -start

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

注: Windows システム上の WebSphere Message Broker または WebSphere Application Server の場合は、-c オプションに続いてコネクター構成ファイルの名前を指定しなければなりません。ICS の場合は、-c はオプションです。

- Adapter Monitor から (WebSphere Business Integration Adapters 製品のみ)。Adapter Monitor は System Manager 始動時に起動されます。

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

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

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

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

コネクターの停止

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

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

    ```
    connector_manager_connName -stop
    ```

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

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

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

- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。

第 5 章 Mainframe Adapter Suite のビジネス・オブジェクトの開発

この章では、MAS コネクターのビジネス・オブジェクト処理方法について説明し、また、データの検索および変更時にコネクターが想定する前提事項についても説明します。本章の内容は、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』
- 81 ページの『ビジネス・オブジェクト動詞の処理』
- 98 ページの『ビジネス・オブジェクトの属性プロパティー』
- 101 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情報』

この情報は、既存のビジネス・オブジェクトを変更する場合のためのガイドとして、または新規ビジネス・オブジェクトのインプリメントに関する提案事項として役立てることができます。

コネクターでは、サポートされるビジネス・オブジェクトの構造、親ビジネス・オブジェクトと子ビジネス・オブジェクトの関係、アプリケーション固有の情報の形式、およびビジネス・オブジェクトのアプリケーション表記に関する前提事項が想定されます。したがって、コネクターが処理するビジネス・オブジェクトを作成または変更する場合は、コネクターが従うように指定されているルールに適合する必要があります。適合しないと、コネクターは新規または変更されたビジネス・オブジェクトを正しく処理できません。

ビジネス・オブジェクトおよび属性の命名規則

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

ビジネス・オブジェクトの構造

多くの場合、コネクターはすべての個別ビジネス・オブジェクトが 1 つのアプリケーション・データベース表またはビューによって表されると想定します。ただし、ビジネス・オブジェクトがストアード・プロシージャーに基づいている場合を除きます。IMS/TM システムまたは CICS/TS システムの場合、ビジネス・オブジェクトはデータ・ソースへの SHADOW 呼び出しに基づいています。**この章では表ベースまたはビュー・ベースのビジネス・オブジェクトについて説明しますが、一般的なアーキテクチャーは、IMS/TM または CICS/TS 用を含むストアード・プロシージャー・ベースのビジネス・オブジェクトにも当てはまります。** 唯一の違いは、IMS/TM オブジェクトや CICS/TS オブジェクトでは SQL ステートメントの代わりにデータ・ソースへの SHADOW 呼び出しを使用することです。

コネクターは、オブジェクト内部のそれぞれの**単純属性** (つまり、String または Integer または Date などの単一値を表す属性) がそのテーブルまたはビュー内の

列によって表されると想定します。したがって、同じ個別ビジネス・オブジェクトに含まれる属性を、別々のアプリケーション・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。

- アプリケーション・データベース表に、対応する個別ビジネス・オブジェクトに含まれる単純属性の数よりも多くの列が含まれる場合があります (つまり、データベース列の一部が、ビジネス・オブジェクト内に表されていません)。ユーザー設計のビジネス・オブジェクトには、ビジネス・オブジェクトの処理に必要な列のみを組み込んでください。
- 個別ビジネス・オブジェクトに、対応するアプリケーション・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、アプリケーション・データベース表内に表されていません)。アプリケーション・データベース表の列を表していないビジネス・オブジェクトの属性には、アプリケーション固有情報が含まれていないか、あるいはデフォルト値またはストアード・プロシージャーが指定されています。
- 個別ビジネス・オブジェクトは、複数のデータベース表にまたがるビューを表すことができます。コネクターでは、アプリケーション内で起動された Create、Retrieve、Update、および Delete の各イベントを処理するときに、そのようなビジネス・オブジェクトを使用することができます。ただし、ビジネス・オブジェクトからの要求を処理する場合には、Retrieve 要求に対してのみ、そのようなビジネス・オブジェクトを使用できます。
- 個別ビジネス・オブジェクトは、関連のないビジネス・オブジェクトのコンテナーとして使用されるラッパー・オブジェクトを表すことができます。ラッパー・オブジェクトはデータベース表やビューによって表されません。ラッパー・オブジェクトは他のオブジェクトの子として使用することはできません。

注: ビジネス・オブジェクトがストアード・プロシージャーを基にしている場合、各単純属性 (ストアード・プロシージャー用の特殊な SP 属性を除く) には、アプリケーション固有情報が含まれていることも、含まれていないこともあります。詳細については、90 ページの『ストアード・プロシージャーの使用』を参照してください。

WebSphere ビジネス・オブジェクトには、フラットなビジネス・オブジェクトと階層のあるビジネス・オブジェクトがあります。**フラット**・ビジネス・オブジェクトの属性は、すべて単純属性であり、単一の値を表します。

階層ビジネス・オブジェクトは、子ビジネス・オブジェクト、子ビジネス・オブジェクトの配列、またはその組み合わせを表す属性を持ちます。そのため、それぞれの子ビジネス・オブジェクトには、1 つの子ビジネス・オブジェクト、またはビジネス・オブジェクトの配列など、いろいろと含めることができます。**単一カーディナリティー関係**は、親ビジネス・オブジェクト内の属性が単一の子ビジネス・オブジェクトを表す場合に発生します。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場合に発生します。この場合、この属性は子ビジネス・オブジェクトと同じタイプの配列です。

注: **階層型**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。**トップレベル**・ビジネス・オブジェクトという用語は、それ自身は親ビジネス・オブジェクトを持たない階層構造のトップの個々のビジネス・オブジェクトを指します。

コネクターでは、ビジネス・オブジェクト間での以下の関係がサポートされます。

- 『単一カーディナリティー関係』
- 78 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』
- 79 ページの『複数カーディナリティー関係』
- 80 ページの『ラッパー・オブジェクト』

カーディナリティーのタイプを問わず、親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は、その関係が保管されるビジネス・オブジェクトのキー属性に含まれるアプリケーション固有情報に記述されています。このアプリケーション固有情報の詳細は、 104 ページの表 28 の 104 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・オブジェクトは、関係を表す属性を少なくとも 2 つ持っています。一方の属性は子のタイプと同一です。もう一方の属性は、単純属性で、子の基本キーを親の外部キーとして持っています。親は子が持つ基本キー属性と同数の外部キー属性を持ちます。

関係を確立する外部キーは親に保管されるため、各親は指定されたタイプの単一カーディナリティーの子は 1 つしか持つことができません。

図 17 に一般的な単一カーディナリティー関係を示します。この例では、`fk1` は単純属性で、子の基本キーを含み、`child[1]` は子ビジネス・オブジェクトを表す属性です。

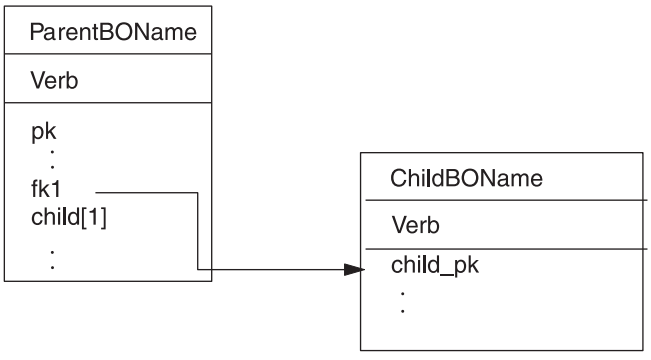


図 17. 典型的な単一カーディナリティー関係

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・オブジェクトが単一の Address ビジネス・オブジェクトを含む場合、新規カスタマーが作成されたときに、新規の行がカスタマー表とアドレス表に挿入されます。新規アドレスは新規カスタマーに固有です。同様に、カスタマー表からカスタマーを削除するときに、カスタマー・アドレスがアドレス表から削除されます。

ただし、複数の階層ビジネス・オブジェクトが同一のデータを含み、そのいずれもが所有していない場合もあります。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 参照表を表しているとします。参照表が更新されることは大変少なく、アドレス・データとは独立して保守されるため、アドレス・データの作成や変更は参照表中のデータに影響しません。コネクターは、既存の都道府県名を検出するか、失敗するかのいずれかです。参照表の値を追加したり、変更したりすることはありません。

複数のビジネス・オブジェクトが同一の単一カーディナリティーの子ビジネス・オブジェクトを含む場合、各親ビジネス・オブジェクトの外部キー属性の関係を NO_OWNERSHIP 指定する必要があります。ビジネス・オブジェクト要求のプロセスからコネクターに階層ビジネス・オブジェクトが、Create、Delete、および Update のいずれかの要求と共に送信されるとき、コネクターは所有権なしに含まれている単一カーディナリティーの子を無視します。コネクターはこれらのビジネス・オブジェクトについては検索のみを実施します。コネクターがこのような単一カーディナリティーのビジネス・オブジェクトの検索に失敗すると、エラーを返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、110 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、106 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

所有権なし包含には、静的参照表を簡単に使用できるようにするほかに、もう 1 つ、正規化データと非正規化データの同期をとる機能もあります。

正規化データから非正規化データへの同期化: 関係を NO_OWNERSHIP と指定すれば、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と同期をとったときに、データを作成または変更できます。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 つの表にデータを保管するとします。さらに正規化されていない宛先アプリケーションは、すべてのデータを単一の表に保管する、つまり表 A の各エンティティーは冗長的に表 B のデータ保管すると想定します。

この例では、ソース・アプリケーションから宛先アプリケーションに表 B のデータを同期させるため、表 B のデータが変更したときには必ず、表 A イベントを起動する必要があります。さらに、表 B のデータは冗長的に表 A に保管されるので、表 A の行ごとに表 B で変更になったデータを含むビジネス・オブジェクトを送信することが必要です。

非正規化データから正規化データへの同期化: 正規化されていないソース・アプリケーションから正規化された宛先アプリケーションにデータを同期させる場合、コネクターは正規化されたアプリケーション内に含まれている所有権のないデータについて、作成、削除、および更新のいずれも行いません。

正規化アプリケーションにデータを同期させる場合、コネクターは所有権なく含まれている単一カーディナリティーの子をすべて無視します。そのような子データを作成、削除、または修正する場合は、手動でデータを処理してください。

複数カーディナリティー関係

通常、子ビジネス・オブジェクト配列を含むビジネス・オブジェクトは、関係を表す属性を 1 つのみ持っています。属性タイプは子ビジネス・オブジェクトと同一タイプの配列になります。1 つ以上の子を持つ親もあるため、関係を確立する外部キーはそれぞれの子に保管されます。

そのため、子は少なくとも 1 つ、親の基本キーを外部キーとして含む単純属性を持つことになります。子は親が持つ基本キー属性と同数の外部キー属性を持ちます。

関係を確立する外部キーは子に保管されるため、各親は子を複数持つこともまったく持たないこともできます。

図 18 に複数カーディナリティー関係を示します。この例では、`parentId` は単純属性で、親の基本キーを含み、`child[n]` は子ビジネス・オブジェクト配列を表す属性です。

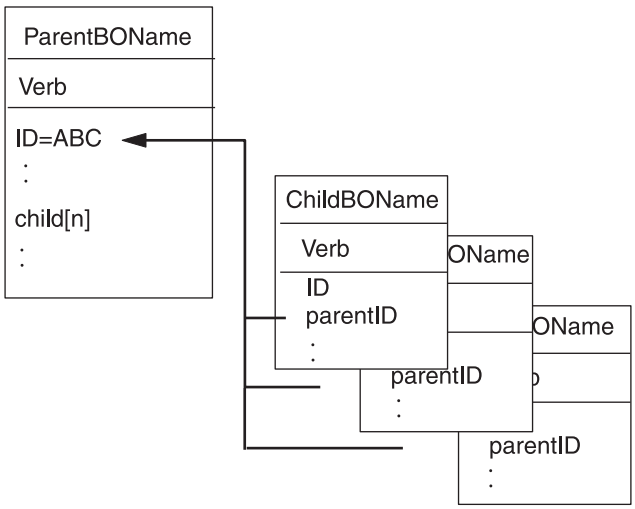


図 18. 複数カーディナリティー・ビジネス・オブジェクト関係

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、関係を親ではなく子の中に保管するように、単一の子エンティティーを保管します。つまり、子は親の基本キーと同一の値の外部キーを含みます。

図 19 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。

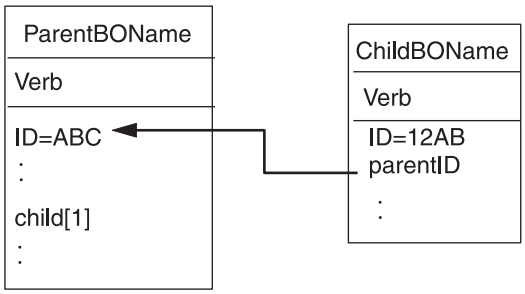


図 19. 関係を子に格納している単一のカーディナリティー関係

子データが親から独立して存在するのではなく、親を通してのみアクセスできる場合、アプリケーションはこのタイプの単一カーディナリティーの関係を使用します。このような子データは、複数の親に所有されることはなく、親とその基本キー値は、子とその外部キー値が作成されるより以前に存在することが必要です。

このようなアプリケーションに対応するため、コネクターは、単一カーディナリティーを持つ子を含む階層ビジネス・オブジェクトもサポートします。ただし、関係は親ではなく子に保管されます。

このような特殊な方法で親ビジネス・オブジェクトに単一カーディナリティーの子を含めるには、子が格納される属性のアプリケーション固有情報を指定するときに、CONTAINMENT パラメーターを含めないようにします。詳細については、110 ページの『単一カーディナリティーの子ビジネス・オブジェクトを表す属性』を参照してください。

ラッパー・オブジェクト

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

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

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

ビジネス・オブジェクト動詞の処理

このセクションでは、ビジネス・オブジェクトの動詞処理の以下の性質について解説します。

- 『動詞の判別』では、コネクターがそれぞれのソース・ビジネス・オブジェクトごとに使用する動詞を決定する方法を説明します。
- 『変更後イメージと差分』では、用語を定義し、コネクターが変更後イメージを扱う方法を説明します。
- 83 ページの『動詞の処理』では、ビジネス・オブジェクトを作成、検索、更新および削除を行う際に、コネクターが実行するステップを説明します。
- 90 ページの『SQL ステートメントの使用』では、コネクターがストアード・プロシージャーを使用する方法を説明します。
- 90 ページの『ストアード・プロシージャーの使用』では、コネクターがストアード・プロシージャーを使用するタイミングおよびその指定方法を説明します。
- 94 ページの『ストアード・プロシージャーまたは単純な SQL ステートメントを使用したビジネス・オブジェクトの処理』では、コネクターがストアード・プロシージャーおよび SQL ステートメントを処理する方法を説明します。
- 98 ページの『トランザクション・コミットとロールバック』では、コネクターがトランザクション・ブロックを使用する方法を簡単に説明します。

動詞の判別

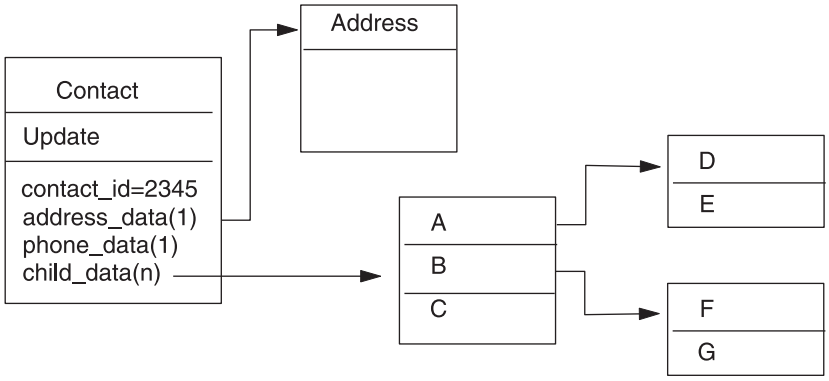
トップレベル・ビジネス・オブジェクト、およびその子にあたる個別ビジネス・オブジェクトには、それぞれ別々に動詞を格納することができます。したがって、ビジネス・オブジェクト要求のプロセスは、親と子のビジネス・オブジェクトに異なる動詞を持つビジネス・オブジェクトを、コネクターに渡すことができます。この場合、コネクターでは、トップレベルの親ビジネス・オブジェクトの動詞を参照して、ビジネス・オブジェクト全体をどのように処理するかを決定します。詳細については、 83 ページの『動詞の処理』を参照してください。

変更後イメージと差分

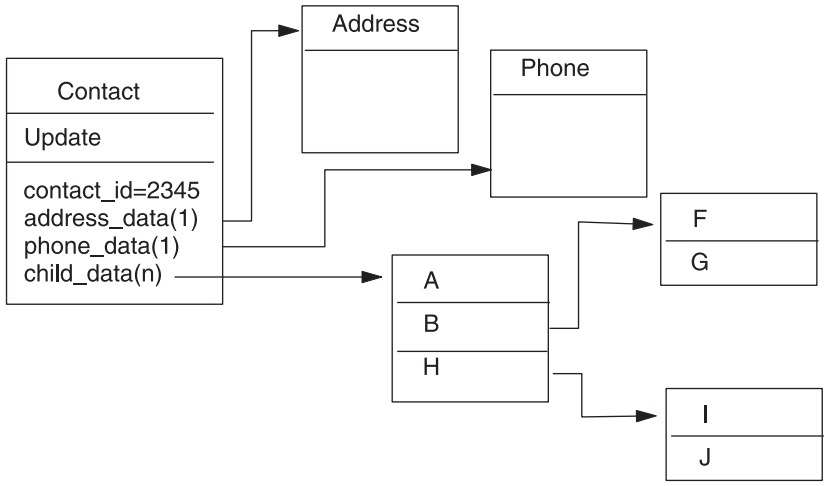
変更後イメージとは、ビジネス・オブジェクトに対するすべての変更が行われた後の、ビジネス・オブジェクトの状態です。**差分**とは、更新操作で使用される、キー値および変更対象のデータのみを含むビジネス・オブジェクトのことです。コネクターは変更後イメージのみをサポートしているため、更新用のビジネス・オブジェクトを受信した場合には、そのビジネス・オブジェクトが、データの更新後のあるべき状態を表していると見なします。

そのためコネクターは、Update 動詞が含まれているビジネス・オブジェクトを受信し、アプリケーション内のビジネス・オブジェクトがソース・ビジネス・オブジェクトと完全に一致するように、その現在の表記を変更します。これを行うために、コネクターは単純属性値を変更して、子ビジネス・オブジェクトを追加または除去します。

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



さらに、コネクターが次のビジネス・オブジェクトを受信すると想定します。



コネクターは、更新処理を行うため、次の変更をデータベースに適用します。

- Contract ビジネス・オブジェクト (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

コネクターは、受け取るそれぞれのビジネス・オブジェクトが変更後イメージを表すことと想定します。したがって、更新のためにコネクターに送信されたそれぞれのビジネス・オブジェクトに有効な既存の子ビジネス・オブジェクトが含まれていることを開発者が確認することが重要です。子ビジネス・オブジェクトの中に単純属性がまったく変更されていないものがある場合にも、その子ビジネス・オブジェクトがソース・ビジネス・オブジェクトに含まれていなければなりません。

ただし、一部のコネクターについては、更新操作時に欠落している子ビジネス・オブジェクトが削除されるのを防ぐ方法があります。子 (または子の配列) を表す属性

のアプリケーション固有情報を使用して、コネクターに対し、ソース・ビジネス・オブジェクトに含まれない子ビジネス・オブジェクトを保持するよう指示することができます。これを行うには、KEEP_RELATIONSHIP を true に設定します。詳細については、 106 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、ビジネス・オブジェクトを作成、検索、更新、または削除する際にコネクターが行うステップについて説明します。コネクターは、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、すべてのビジネス・オブジェクトを処理するまで、それぞれの子ビジネス・オブジェクトに同じステップを実行します。

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

ビジネス・オブジェクトの比較

以下に概要する処理のさまざまなポイントで、コネクターは 2 つのビジネス・オブジェクトを比較し、それらが同一であるかどうかを確認します。例えば、更新操作時には、ビジネス・オブジェクトの配列内に、特定のビジネス・オブジェクトが存在するかどうかを判定します。この検査を行うために、コネクターはビジネス・オブジェクトと配列内の各ビジネス・オブジェクトを比較します。2 つのビジネス・オブジェクトが同一であるには、次の 2 つの条件を満たす必要があります。

- 比較されるビジネス・オブジェクトのタイプが同じでなければなりません。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・オブジェクト内のすべてのキー属性には、同一の値が格納されている必要があります。両方のキー属性に CxIgnore が設定されていると、コネクターはそれらを同一と見なします。しかし、片方のビジネス・オブジェクトではキー属性に CxIgnore が設定されているが、もう一方では設定されていない場合、ビジネス・オブジェクトは同一と見なされません。

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。

コネクターは、階層ビジネス・オブジェクトを作成する場合、次のステップで行います。

1. 所有関係にある単一カーディナリティーの子ビジネス・オブジェクトのすべてを、アプリケーション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・オブジェクトと、その子に組み込まれるすべての子ビジネス・オブジェクトを作成します。

ビジネス・オブジェクト定義上ある属性がある単一カーディナリティーの関係の子ビジネス・オブジェクトを表すものとされている場合に、その属性が空になっていると、コネクターはその属性を無視します。ただし、ビジネス・オブジェクト定義上、その属性が子を表すことが必須であるにもかかわらず、子を表していない場合には、コネクターはエラーを戻して処理を停止します。

2. 所有関係にない単一カーディナリティーの子ビジネス・オブジェクトを、次のようにしてすべて処理します。
   a. ビジネス・オブジェクト要求によって渡されたキー値を使用して、データベース中で再帰的に子の検索を試みます。
   b. 検索に失敗した場合 (子がアプリケーションに現存しないことを意味します)、コネクターはエラーを戻して処理を停止します。検索に成功した場合、コネクターは子ビジネス・オブジェクトを再帰的に更新します。

   注: 子ビジネス・オブジェクトが既にアプリケーションに存在するときに、この方法が正確に動作するためには、開発者が作成操作において、子ビジネス・オブジェクトの基本キー属性が正しく相互参照されるように注意する必要があります。アプリケーション・データベースに子ビジネス・オブジェクトが存在していない場合は、基本キー属性を `CxBlank` に設定してください。
3. トップレベル・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. トップレベル・ビジネス・オブジェクトの外部キー値を、対応する単一カーディナリティーの関係にある子ビジネス・オブジェクトの基本キー値に設定します。子ビジネス・オブジェクトの値は、データベース・シーケンスまたはカウンター、あるいはデータベース自体によって、子ビジネス・オブジェクトの作成時に設定される場合があります。そのため、このステップでは、コネクターが親をデータベースに挿入する前に、親の外部キー値を正しいものにします。
   b. データベースによって自動的に設定される属性のそれぞれに対して、新しい固有 ID 値を生成します。データベース・シーケンスまたはカウンターの名前は、属性のアプリケーション固有情報に格納されています。属性にデータベース・シーケンスまたはカウンターが関連付けられている場合、コネクターによって生成された値により、要求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、 104 ページの『単純属性のアプリケーション固有情報』の `UID=AUTO` を参照してください。
   c. 属性のアプリケーション固有情報に含まれる `CA` (CopyAttribute) パラメーターの指定に従って、属性間で値をコピーします。`CA` パラメーターの使用については、 104 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. トップレベル・ビジネス・オブジェクトをデータベース内に挿入します。

   注: ラッパーであるトップレベル・ビジネス・オブジェクトは、データベース内に挿入されません。
4. 単一カーディナリティーの子ビジネス・オブジェクトのうち、親/子関係を子に保管するものすべてを、次のようにして処理します。

a. 子の外部キー値を、親に含まれる対応する基本キー属性値を参照するように設定します。親の基本キー値は、親の作成時に生成されている可能性があります。そのため、ここでは、コネクターが子をアプリケーションに挿入する前に、それぞれの子の外部キー値を正しいものにします。

b. 子をアプリケーションに挿入します。

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、コネクターが子をアプリケーションに挿入する前に、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

コネクターは、階層ビジネス・オブジェクトを検索する場合、次のステップで行います。

1. ビジネス・オブジェクト要求から受け取った子ビジネス・オブジェクトすべてを、トップレベル・ビジネス・オブジェクトから除去します。
2. トップレベル・ビジネス・オブジェクトを、アプリケーション内で検索します。
   - 検索の結果戻された行が 1 つの場合、コネクターは処理を継続します。
   - 検索の結果戻された行がない場合 (目的のトップレベル・ビジネス・オブジェクトがデータベース内に存在しないことを意味します)、コネクターは `BO_DOES_NOT_EXIST` を戻します。
   - 検索の結果戻された行が複数ある場合、コネクターは `FAIL` を戻します。

   注: ビジネス・オブジェクトには、データベース列にマップしないプレースホルダー属性などの属性が含まれている場合があります。コネクターが、検索時にトップレベル・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、ビジネス・オブジェクト要求から受信した値に設定されたまま保持されます)。子ビジネス・オブジェクトのそのような属性については、検索時にコネクターによってデフォルト値に設定されます。

   注: ラッパーであるトップレベル・ビジネス・オブジェクトには、ラッパー・オブジェクトの直下にあるオブジェクトのすべての属性値が含まれている必要があります。この値はキーおよびプレースホルダー属性などのオブジェクトの検索に必要となります。ラッパー・オブジェクトにはすべてのキーおよびプレースホルダー属性が取り込まれる必要があります。ラッパーの 1 レベル下のオブジェクトで外部キーとして使用されるラッパー・オブジェクトの単純属性は、ラッパー・オブジェクトのキーとしてマークされる必要があります。
3. 複数カーディナリティーの子ビジネス・オブジェクトのすべてを、再帰的に検索します。

   注: コネクターはビジネス・オブジェクトの配列に値を設定する際に、一意であることを強制しません。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。アプリケーションから戻された子ビジネス・オブジェクトに重複があると、コネクターは、それらの重複する子を戻します。

4. 子ビジネス・オブジェクトが所有権を持って含まれているか、持たずに含まれているかどうかに関係なく、単一カーディナリティーの子をそれぞれ、再帰的に検索します。

注: 単一カーディナリティーの子ビジネス・オブジェクトは、すべて、親ビジネス・オブジェクト内での出現順序に従って、親ビジネス・オブジェクトが処理される前に処理されます。子ビジネス・オブジェクトが所有されているかいないかは、処理シーケンスの決定には関係がありません。ただし、処理のタイプの決定には関係があります。

RetrieveByContent 操作

RetrieveByContent 動詞は、トップレベル・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、トップレベル・ビジネス・オブジェクトのみに含まれる属性を基に、コネクターによる検索が実行されるからです。

トップレベル・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。

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

注: RetrieveByContent 動詞はラッパーであるトップレベル・ビジネス・オブジェクトには適用されません。

Update 操作

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

コネクターは、階層ビジネス・オブジェクトを更新する場合、次のステップで行います。

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、対応するエンティティーをアプリケーションから検索します。検索されたビジネス・オブジェクトは、データベース内のデータの現在の状態を正確に表したものです。
   - 検索が失敗した場合 (目的のトップレベル・ビジネス・オブジェクトがアプリケーション内に存在しないことを意味します)、コネクターは `BO_DOES_NOT_EXIST` を戻します。この場合、更新は失敗します。

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

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

- 検索に成功した場合、コネクターは、検索されたビジネス・オブジェクトをソース・ビジネス・オブジェクトと比較して、どの子ビジネス・オブジェクトに関してアプリケーションに変更を加える必要があるかを判別します。ただし、ソース・ビジネス・オブジェクトの単純属性の値と、検索されたビジネス・オブジェクトの単純属性の値の比較は行いません。コネクターは、非キーの単純属性のすべてで、値を更新します。

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

2. トップレベル・ビジネス・オブジェクトの単一カーディナリティーの子をすべて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。両方に存在しない場合は、更新が失敗し、コネクターはエラーを戻します。

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

   - ソース・ビジネス・オブジェクトおよび検索したビジネス・オブジェクトの両方に子が存在する場合、コネクターは、データベース内の既存の子を更新するのではなく、既存の子を削除して新規の子を作成します。
   - その子がソース・ビジネス・オブジェクトには存在するにもかかわらず、検索されたビジネス・オブジェクトには存在しない場合は、データベース内にその子を再帰的に作成します。
   - その子が検索されたビジネス・オブジェクトには存在するにもかかわらず、ソース・ビジネス・オブジェクトには存在しない場合は、データベース内のその子を再帰的に削除します。削除タイプが物理的であるか、論理的であるかは `ChildUpdatePhyDelete` プロパティーの値に依存します。

   所有関係にない単一カーディナリティーの子に関しては、コネクターは、ソース・ビジネス・オブジェクトに存在するそのような子のすべてを、アプリケーションから検索しようとします。コネクターは、子を正常に検索した場合、子ビジネス・オブジェクトに値を取り込みますが、更新はしません。所有権を持たずに含まれている単一カーディナリティーの子はコネクターによって変更されることはありません。

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

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

5. 親/子関係を子に保管する子ビジネス・オブジェクト (複数カーディナリティーであるか、単一カーディナリティーであるかを問いません) のそれぞれにおいて、外部キー値のすべてを、対応する親ビジネス・オブジェクトの基本キー値に設定します。一般に、ICS が統合ブローカーである場合には、これらの値はデータ・マッピング時に相互参照されています。ただし、関係を子に保存する新しい子の外部キー値を、コネクターがそれらの子を更新する前に正しいものにするためには、このステップが重要です。
6. 検索されたビジネス・オブジェクトの複数カーディナリティーの子のそれぞれを、次のいずれかの方法で処理します。
   - ソース・ビジネス・オブジェクトおよび検索したビジネス・オブジェクトの両方に子が存在する場合、コネクターは、データベース内の既存の子を更新するのではなく、既存の子を削除して新規の子を作成します。
   - その子がソース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、アプリケーション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、ソース配列には存在しない場合は、アプリケーションからその子を再帰的に削除します。ただし、親に含まれている、その子を表す属性のアプリケーション固有情報で、KEEP_RELATIONSHIP が true に設定されている場合を除きます。この場合、コネクターは、アプリケーションからその子を削除しません。詳細については、 106 ページの『属性の外部キーの指定』を参照してください。削除タイプが物理的であるか、論理的であるかはChildUpdatePhyDelete プロパティーの値に依存します。

注: 開発者は確実に、ビジネス・オブジェクトが、ソース・ビジネス・オブジェクトで複数カーディナリティーを持って含まれているビジネス・オブジェクトが固有である (すなわち、同じビジネス・オブジェクトの 2 つ以上のコピーが配列に含まれない) ようにコーディングする必要があります。コネクターがソース配列内で重複するビジネス・オブジェクトを受け取ると、ビジネス・オブジェクトを 2 度処理し、予測できない結果が生じることがあります。

DeltaUpdate 操作

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

1. Update 動詞が処理される際には更新の前に検索が実行されますが、DeltaUpdate が処理される際には実行されません。
2. 着信ビジネス・オブジェクトとデータベース内のビジネス・オブジェクトの比較が行われません。
3. どの子も、各子オブジェクトに設定されている動詞セットに基づいて処理されます。子に動詞セットが設定されていない場合、コネクターはエラーを戻します。

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

1. 親オブジェクトの子のうち、単一カーディナリティーのものすべてを再帰的に処理します。ビジネス・オブジェクト定義で IsRequired が true に設定されている子は、インバウンド・オブジェクトに必ず存在していなければなりません。存在しない場合、差分更新は失敗し、コネクターはエラーを戻します。
2. 親に含まれる外部キー値のうち、単一カーディナリティーの子の属性を参照するものすべてを、それぞれ対応する子の値に設定します。この処理が必要なのは、これ以前のステップで単一カーディナリティーの子がデータベースに追加され、新しいシーケンス値が生成されている可能性があるためです。
3. 現在処理中のオブジェクトを、SQL UPDATE ステートメントまたはストアード・プロシージャーを使用して更新します。個々のビジネス・オブジェクトのすべての単純属性が更新されます。ただし、インバウンド・ビジネス・オブジェクトで IsIgnore に設定されている属性を除きます。コネクターでは、インバウンド・オブジェクトと現在のオブジェクトを属性レベルで比較して、UPDATE ステートメントに追加する必要がある属性を決定することはありません。つまり、属性はすべて更新されます。更新されるオブジェクトは一意である必要があるため、コネクターは、結果として 1 行のみが処理されることを確認します。複数の行が処理される場合、エラーが戻されます。
4. 現在のオブジェクトの子のうち、カーディナリティーが N のものすべてで、親の属性を参照する外部キー値のすべてを、それぞれ対応する親の値に設定します。通常、これらの値はデータ・マッピング時に相互参照されます。ただし、これはカーディナリティーが N のコンテナーに含まれる新しい子には該当しないことがあります。ここでの処理により、カーディナリティーが N の子のすべてで、これらの子が更新される前に外部キー値を確実に正しい値にすることができます。
5. 現在のオブジェクトの、カーディナリティーが N のコンテナーをすべて更新します。

子オブジェクトが処理されるときには、それぞれの子の動詞が取得されて適切な操作が実行されます。DeltaUpdate が処理される際に許可される子の動詞は、Create、Delete、および DeltaUpdate です。

- 子で Create 動詞が検出された場合、その子が所有関係にある子であれば、データベースにその子が作成されます。所有関係にない子に関しては、検索により、データベースに存在するかどうかが確認されます。
- 子で Delete 動詞が検出された場合、その子は削除されます。
- 子で DeltaUpdate 動詞が検出された場合、データベースでその子が更新されます。

Delete 操作

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

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

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

1. 所有権付きで含まれている単一カーディナリティーの子ビジネス・オブジェクトすべてを再帰的に削除します。
2. トップレベル・ビジネス・オブジェクトを削除します。
3. 複数カーディナリティーの子ビジネス・オブジェクトすべてを再帰的に削除します。

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

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

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

SQL ステートメントの使用

コネクターでは、単純な SQL ステートメントを使用して、選択、更新、検索、または削除の操作を行うことができます。SQL ステートメント用の列名は、属性の AppSpecificInfo プロパティーから取得されます。それぞれの照会のスパンは、ビューに POST されていない場合には、1 テーブルのみです。

ストアード・プロシージャーの使用

ストアード・プロシージャーとは、複数の SQL ステートメントのグループであり、1 つの論理単位を形成して特定のタスクを実行します。ストアード・プロシージャーは、コネクターがオブジェクトに対して実行する一連の操作または照会を、データベース・サーバー内にカプセル化したものです。属性の AppSpecificInfo プロパティーを使用して、ストアード・プロシージャーにパラメーターが受け渡されます。

コネクターは、次の目的でストアード・プロシージャーを呼び出します。

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

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

ストアード・プロシージャーの指定

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

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

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

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

コネクターでは、以下のビジネス・オブジェクト属性名が認識されます。

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

注: コネクターに実行させるストアード・プロシージャーについてのみ、属性を作成してください。例えば、ICS が統合ブローカーである場合、アプリケーション固有の情報またはマッピングを使用して、ビジネス・オブジェクトがコネクターに送信される前に、これらの属性の値を指定します。これらの値に変更が加えられた場合に、それ以後のビジネス・オブジェクトに対するストアード・

プロシージャーの呼び出しのためにコネクターにその変更を認識させるには、コネクターを再始動する必要があります。

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

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

ここで、以下のように説明されます。

StoredProcedureName	ストアード・プロシージャーの名前。
RS	結果セット: true の場合、ストアード・プロシージャーが結果セットを戻します。false の場合、結果セットは戻されません。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 ColumnName プロパティーは、結果セットの該当するカラムを指します。RS が出力パラメーター・リストの一部である場合は、その特定のパラメーターが結果セットを戻します。1 つの結果セット OUT パラメーターのみがサポートされます。複数の結果セットが OUT パラメーターとして戻された場合は、最初の結果セットのみが戻され、その他の結果セットはすべて無視されます。現在、この機能は Oracle 8i 以上、および Oracle JDBC ドライバーを使用するストアード・プロシージャーについてのみサポートされます。データベース内のストアード・プロシージャーの場合、対応するパラメーターは REFCURSOR タイプを戻します。
IP	入力パラメーター。コネクターがストアード・プロシージャーの処理時に入力値として使用する値が格納されているビジネス・オブジェクト属性のリスト。
OP	出力パラメーター。コネクターがストアード・プロシージャーの実行後に戻り値を格納するビジネス・オブジェクト属性のリスト。結果セットの記述については、RS を参照してください。
IO	入出力パラメーター。コネクターが入力値として使用し、コネクターがストアード・プロシージャーの実行後に戻り値を格納するビジネス・オブジェクト属性のリスト。

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

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

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

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

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

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

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

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

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

- 属性の AppSpecificInfo プロパティー

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

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

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

  [End]
  ```

- 属性の値

  ストアード・プロシージャーを指定するテキストの長さが 4000 バイトより大きい場合には、ストアード・プロシージャーを指定する必要があります。ICS が統合ブローカーであるときは、ビジネス・オブジェクトが要求を表す場合にのみ、マッピングを使用してストアード・プロシージャーを指定できます。つまり、コネクターがイベントをポーリングしている場合は、ストアード・プロシージャーの指定に属性の値を使用できません。

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

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

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

次の各セクションでは、コネクターがストアード・プロシージャーおよび SQL ステートメントをどのように処理するかを説明します。

- 『ビジネス・オブジェクトの Create 操作』
- 95 ページの『ビジネス・オブジェクトの Update 操作』
- 95 ページの『ビジネス・オブジェクトの Delete 操作』
- 96 ページの『ビジネス・オブジェクトの Retrieve 操作』
- 97 ページの『ビジネス・オブジェクトの RetrieveByContent 操作』
- 97 ページの『ビジネス・オブジェクトの Retrieve-for-Update 操作』

ビジネス・オブジェクトの Create 操作

Create ストアード・プロシージャーは、通常、コネクターがトップレベル・ビジネス・オブジェクトの単純属性を設定するために使用する値を戻します。コネクターは、SQL ステートメントおよびストアード・プロシージャー (BeforeCreate、Create、AfterCreate) を使用してビジネス・オブジェクトの Create 操作を処理するときに、次のステップを実行します。

1. ビジネス・オブジェクトが BeforeCreateSP 属性を含むかどうかをチェックします。含まれている場合、BeforeCreate ストアード・プロシージャーを呼び出します。
2. ストアード・プロシージャー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。
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. 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 属性を持っているか検査します。持っていれば、Create ストアード・プロシージャーを呼び出して、トップレベル・ビジネス・オブジェクトを作成します。持っていなければ、INSERT ステートメントをビルドし、実行して、トップレベル・ビジネス・オブジェクトを作成します。

6. Create ストアード・プロシージャー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それぞれの外部キー値を、その親の基本キー属性値にセットします。
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 属性を持っているか検査します。含まれている場合、AfterCreate ストアード・プロシージャーを呼び出します。
10. ストアード・プロシージャー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。

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

ビジネス・オブジェクトの Update 操作

Update ストアード・プロシージャーは、通常、コネクターがトップレベル・ビジネス・オブジェクトの単純属性を設定するために使用する値を戻します。コネクターは、SQL ステートメントおよびストアード・プロシージャー (BeforeUpdate、Update、AfterUpdate) を使用してビジネス・オブジェクトの Update 操作を処理するときに、次のステップを実行します。

1. ビジネス・オブジェクトが BeforeUpdateSP 属性を持っているか検査します。持っていれば、BeforeUpdate ストアード・プロシージャーを呼び出します。
2. BeforeUpdate ストアード・プロシージャー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。
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. 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 UpdateSP 属性を持っているか検査します。持っていれば、トップレベル・ビジネス・オブジェクトを更新するために Update ストアード・プロシージャーを呼び出します。持っていなければ、UPDATE ステートメントをビルドし、実行して、トップレベル・ビジネス・オブジェクトを更新します。
6. Update ストアード・プロシージャー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、対応する親の基本キー属性中の値を参照するようにセットします。
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 属性を持っているか検査します。持っていれば、AfterUpdate ストアード・プロシージャーを呼び出します。
10. ストアード・プロシージャー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。

ビジネス・オブジェクトの Delete 操作

Delete ストアード・プロシージャーは、コネクターに値を戻しません。コネクターは、単純な SQL ステートメントおよびストアード・プロシージャー (BeforeDelete、Delete、AfterDelete) を使用してビジネス・オブジェクトの Delete 操作を処理するときに、次のステップを実行します。

1. ビジネス・オブジェクトが BeforeDeleteSP 属性を持っているか検査します。持っていれば、BeforeDelete ストアード・プロシージャーを呼び出します。
2. 単一カーディナリティーの子ビジネス・オブジェクトを削除します。
3. 複数カーディナリティーの子ビジネス・オブジェクトを削除します。
4. ビジネス・オブジェクトが DeleteSP 属性を持っているか検査します。持っていれば、Delete ストアード・プロシージャーを呼び出して、トップレベル・ビジネス・オブジェクトを削除します。持っていなければ、DELETE ステートメントをビルドして実行します。
5. ビジネス・オブジェクトが AfterDeleteSP 属性を持っているか検査します。持っていれば、AfterDelete ストアード・プロシージャーを呼び出します。

ビジネス・オブジェクトの Retrieve 操作

単純な検索操作としては、トップレベル・ビジネス・オブジェクト、単一カーディナリティーの子、および複数のカーディナリティーの子に使用できるストアード・プロシージャーがあります。プロシージャーの順序は、次のとおりです。

- BeforeRetrieve
- Retrieve
- AfterRetrieve

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

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

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

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

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

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

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

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

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

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

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

ビジネス・オブジェクトの RetrieveByContent 操作

単純な RetrieveByContent 操作では、ストアード・プロシージャーはトップレベル・ビジネス・オブジェクト、およびその単一カーディナリティーの子にのみ使用できます。つまり、結果セットや複数の行を戻すためには使用できません。プロシージャーの順序は、次のとおりです。

- BeforeRetrieveByContent
- RetrieveByContent
- AfterRetrieveByContent

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

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

ビジネス・オブジェクトの Retrieve-for-Update 操作

以下のストアード・プロシージャーは、トップレベル・ビジネス・オブジェクトで呼び出され、単純な Retrieve と同様の方法ですべての子ビジネス・オブジェクトを検索します。

プロシージャーの順序は、次のとおりです。

- BeforeRetrieveUpdate

- RetrieveUpdate
- AfterRetrieveUpdate

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

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

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

トランザクション・コミットとロールバック

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

ビジネス・オブジェクトの属性プロパティー

ビジネス・オブジェクト・アーキテクチャーは、属性に適用されるさまざまなプロパティーを定義します。このセクションでは、コネクターがこれらのプロパティーのいくつかを解釈する方法と、ビジネス・オブジェクトを変更する際にそのプロパティーをセットする方法を解説します。

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 などのタイプか、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 タイプ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、それぞれ 1 または n の値を持っています。子ビジネス・オブジェクトを表す属性はすべて、ContainedObjectVersion プロパティー (子のバージョン番号を指定) と Relationship プロパティー (Containment 値を指定) も持っています。

Max length プロパティー

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

Key プロパティー

どのビジネス・オブジェクトでも、1 つ以上の単純属性がキーに指定されなければなりません。属性をキーとして定義するには、このプロパティーを Yes に設定します。ビジネス・オブジェクト属性が、 String タイプの場合、WebSphere ではデータベース中のデータ型を char 型ではなく、Varchar 型にすることをお勧めします。

注: 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を、キー属性として指定することはサポートしていません。

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

親/子関係を子に格納する子 (複数カーディナリティーであるか、単一カーディナリティーであるかを問いません) に含まれるある属性で、キー・プロパティーが true に設定されている場合、コネクターは、親の基本キーを SELECT ステートメントの WHERE 文節に使用します。Key プロパティーは使用しません。子の外部キー属性をセットするために値が使用されるビジネス・オブジェクト属性の名前を指定する方法は、103 ページの『属性レベルのアプリケーション固有情報』を参照してください。

Foreign key プロパティー

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

Required プロパティー

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

このプロパティーが単一カーディナリティーの子ビジネス・オブジェクトを表す属性に対して指定されている場合、その親ビジネス・オブジェクト内には、この属性の子ビジネス・オブジェクトが含まれている必要があります。

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

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

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

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

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

AppSpecificInfo

このプロパティーに関する詳細については、103 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、58 ページの表 23 を参照してください。

特殊属性値

ビジネス・オブジェクト内の単純属性は、特殊値 CxIgnore を持つことができます。コネクターは、ビジネス・オブジェクトを受け取るとき、CxIgnore の値を持つすべての属性を無視します。それらの属性は、不可視として取り扱われます。

コネクターがデータベースからデータを検索して、SELECT ステートメントが属性にヌル値を戻したときは、コネクターはデフォルトでは、その属性の値に CxIgnore をセットします。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指定されている場合、コネクターはヌルを表すのにその値を使用します。

コネクターは、すべてのビジネス・オブジェクトが少なくとも 1 つの基本キー属性を持つことを必要とするため、コラボレーションおよびマップの開発者は、コネクターに受け渡されたビジネス・オブジェクトが、CxIgnore に設定されていない基本キーを少なくとも 1 つ確実に持つようにする必要があります。この要件の例外は、コネクター がカウンターまたはシーケンスを使用して、あるいはデータベースによって基本キーを生成するビジネス・オブジェクトの場合のみです。

コネクターは、データベースへのデータの挿入時に、値が指定されていないビジネス・オブジェクト属性があると、その属性の UseNullValue プロパティーに指定されている値を使用します。UseNullValue に関する詳細については、104 ページの表 28 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情報

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

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

アプリケーション固有のビジネス・オブジェクトを拡張または変更する場合、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が、コネクターが予期する構文に一致することを確認する必要があります。

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

表 27 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 27. 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要

アプリケーション固有情報の有効範囲	機能
ビジネス・オブジェクト全体	以下のものを指定します。 • 対応するデータベース表の名前。 • コネクターが論理 (ソフト) 削除実行のために WHERE 文節内に使用する値が含まれる列を定義します。 • トップレベル・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 27. 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要 (続き)

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

以下のセクションでは、この機能をより詳細に解説します。

ビジネス・オブジェクト・レベルのアプリケーション固有情報

ビジネス・オブジェクト・レベルのアプリケーション固有情報により、次のことが可能になります。

- 対応するアプリケーション・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
- トップレベル・ビジネス・オブジェクトがラッパー・オブジェクトであることを指定

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

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

ここで、TableName は、データベース表を識別します。StatusColumnName は、論理的な削除を実行するために使用されるアプリケーション・データベース列の名前です。StatusValue は、ビジネス・オブジェクトが非アクティブであるか、または削除済みであるかを示す値です。

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

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

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

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

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

SCN プロパティーの値は、更新操作と削除操作の両方で使用できます。

- 更新を実行するとき、コネクターは `ChildUpdatePhyDelete` プロパティーの値を使用して、欠落している子データを物理的に削除するか、論理的に削除するかを判断します。子のデータを論理的に削除する場合には、SCN パラメーターの値を使用して、状況列の名前と状況値を示すテキストを取得します。詳細については、86 ページの『Update 操作』を参照してください。
- コネクターは、削除の実行時には、SCN パラメーターの値に基づいて、ビジネス・オブジェクト全体を物理的に削除するか、論理的に削除するかを決定します。SCN パラメーターに値が含まれている場合は、論理削除を実行します。SCN パラメーターに値が含まれていない場合は、物理削除を実行します。詳細については、89 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、トップレベル・ビジネス・オブジェクトはラッパー・オブジェクトです。ラッパー・オブジェクトはデータベース表やビューによって表されません。ラッパーは関連のないビジネス・オブジェクトのコンテナーとして使用されます。コネクターはトップレベル・オブジェクトを無視し、子のみを処理します。ラッパー・オブジェクトには N のカーディナリティーを持つエンティティーまたは N-1 のカーディナリティーを持つエンティティー、あるいはその両方を含めることができます。

属性レベルのアプリケーション固有情報

属性のアプリケーション固有情報は、属性が単純属性であるか、子ビジネス・オブジェクト (または子ビジネス・オブジェクトの配列) を表す属性であるかによって異なります。さらに、子を表す属性のアプリケーション固有情報は、親/子関係が子に保管されるか、親に保管されるかによって異なります。子または子ビジネス・オブジェクト配列を表す属性のアプリケーション固有情報については、106 ページの『属性の外部キーの指定』を参照してください。

単純属性のアプリケーション固有情報

単純属性では、アプリケーション固有情報の形式は、名前と値のペアを表す 11 個のパラメーターで構成されています。どのパラメーターにも、パラメーター名とその値が含まれます。各パラメーター・セットは、コロン (:) 区切り文字で区切られます。

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 (|) はオプション・セットのメンバー同士を区切ります。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]]]:
[CA=set_attr_name|..set_attr_name]:[OB=[ASC|DESC]]:[UNVL=value]:
[ESC=true|false]:[FIXEDCHAR=true|false]:
[BYTEARRAY=true|false]:[USE_LIKE=true|false:
[WILDCARD_POSITION=non-negative number|NONE|BEGIN|END|BOTH]]:
[CLOB=true]
```

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

```
CN=customer_id
```

表 28 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 28. アプリケーション固有情報の属性の名前値のパラメーター

パラメーター	説明
CN=*col_name*	この属性に対応するアプリケーション・データベース列の名前です。
FK=[*fk_object_name*.] *fk_attribute_name*	このプロパティーの値は、親/子関係が親ビジネス・オブジェクトまたは子に格納されているかどうかによって異なります。属性が外部キーでない場合は、このパラメーターをアプリケーション固有情報に含めないでください。詳細については、106 ページの『属性の外部キーの指定』を参照してください。
UID=AUTO UID=*uid_name* | schema_name.uid_name [=UseIfMissing] UID=CW.*uidcolumnname* [=UseIfMissing]	コネクターでは、ビジネス・オブジェクトの固有 ID の生成に、このパラメーターを使用します。属性で固有 ID の生成が必要とされていない場合には、このパラメーターをアプリケーション固有情報に含めないでください。ビジネス・オブジェクトの処理中に固有の ID を保存する方法の詳細は、 66 ページの『PreserveUIDSeq』プロパティー記述を参照してください。詳細については、109 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。
CA=*set_attr_name*|..set_attr_name	*set_attr_name* は、現在の個別のビジネス・オブジェクト内にある別の属性の名前に設定すると、コネクターは、指定された属性の値を使ってこの属性の値を設定してから、Create 操作時にアプリケーション・データベースにビジネス・オブジェクトを追加します。 *set_attr_name* の値は親ビジネス・オブジェクトまたは子ビジネス・オブジェクトの属性を参照できませんが、set_attr_name の前にピリオドが 2 つある場合は親ビジネス・オブジェクトの属性を参照できます。 このパラメーターがアプリケーション固有情報に含まれていない場合、コネクターは、別の属性の属性値 (CA) をコピーせずに、現在の属性の値を使用します。

表 28. アプリケーション固有情報の属性の名前値のパラメーター (続き)

パラメーター	説明
OB=[ASC|DESC]	このパラメーターに値が指定されている場合、このパラメーターが指定されている属性が子ビジネス・オブジェクト内に存在するものであれば、コネクターでは、検索照会の ORDER BY 文節に、その属性の値を使用します。 コネクターは、子ビジネス・オブジェクトを昇順または降順で検索することができます。 昇順での検索を指定するには ASC を使用します。 降順での検索を指定するには DESC を使用します。 このパラメーターがアプリケーション固有情報に含まれていない場合、コネクターは、検索順序を指定するときに、このパラメーターが指定されている属性を使用しません。
UNVL=*value*	コネクターが、値が null の属性を持つビジネス・オブジェクトを検索するときに、null を表すため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。
ESC=[true|false]	コネクターが、ReplaceAllStr プロパティーで特定された各文字のすべてのインスタンスを、ReplaceStrList プロパティーで指定された置換ストリングに置き換えるかどうかを決定します。このパラメーターが値を含んでいなければ、コネクターは ReplaceStrList プロパティーの値を使用して決定します。
FIXEDCHAR= true|false	表内の列が (VARCHAR 型ではなく) CHAR 型である場合に、このパラメーターが指定されている属性を固定長とするかどうかを指定します。例えば、ある特定の属性が CHAR 型の列にリンクされている場合は、その属性のアプリケーション固有の情報では FIXEDCHAR=true が指定されるため、コネクターはその属性の値をFIXEDCHARと見なします。このパラメーターが指定されている属性の MaxLength プロパティーの指定値は、アプリケーション・データベース内に指定されている CHAR の長さと一致するようにしてください。デフォルトでは、FIXEDCHAR=false です。
BYTEARRAY=true|false	BYTEARRAY=true の場合、コネクターはデータベースに対するバイナリー・データの読み取りおよび書き込みを実行し、そのデータをストリングとして ICS/WebSphere MQ Integration Broker に送信します。BYTEARRAY=false がデフォルトです。
USE_LIKE=true|false	コネクターがストリングを比較する時に = 演算子または LIKE 演算子のどちらを使用するかを指定します。USE_LIKE が true に設定されている場合、ワイルドカード照会を実行するには WILDCARD_POSITION を設定します。USE_LIKE が false に設定されている場合は、= 演算子が使用されます。
WILDCARD_POSITION=non-negative number|NONE|BEGIN|END|BOTH	USE_LIKE が true の場合、ワイルドカードの位置を指定するために WILDCARD_POSITION が使用されます。この値は負以外の任意の数値、NONE、BEGIN、END、または BOTH に設定できます。例えば、BEGIN を使用すると、ワイルドカード文字がストリングの先頭に置かれます (%string)。END を使用すると、ワイルドカード文字がストリングの末尾に置かれます (string%)。BOTH を使用すると、ワイルドカード文字がストリングの先頭と末尾の両方に置かれます (%string%)。

表 28. アプリケーション固有情報の属性の名前値のパラメーター (続き)

パラメーター	説明
CLOB=true	String 属性タイプにのみ適用可能。この属性に対応するデータベース列が CLOB データ型であることを指定します。 **注:** CLOB データ型については、以下のように定義されています。 • CLOB に対応する属性では Type が String に設定されており、長さを示す値は CLOB の長さを規定するために使用されています。 • CLOB に対応する属性では、ASI=CN=xyz; CLOB=true と指定されています。 • その他のタイプの属性の ASI で CLOB を使用すると、エラーが発生します。 • CLOB=false と指定すると、エラーが発生します。 通常の String 型の属性は CLOB 対応の属性とほぼ同じですが、ASI に CLOB が使用されていません。CLOB データ型を使用する場合、4 KB 以上のサイズのデータを挿入または更新することができます。ただし、このデータ型を使用できるのは Oracle に限られており、また、Oracle でこのデータ型を使用するためには CLOB をサポートするシン・ドライバーが必要です。それ以外のドライバーを使用すると、エラーが発生する可能性があります。

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

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

- 親に保管される場合: 子ビジネス・オブジェクトのタイプと、外部キーとして使用される子ビジネス・オブジェクト内の属性の名前の両方が含まれるように値を設定します。
- 子に保管される場合: 外部キーとして使用される親ビジネス・オブジェクト内の属性の名前のみを含むように値を設定します。

fk_object_name の値が子ビジネス・オブジェクトのタイプと一致せず、*fk_attribute_name* の値が親または子 (該当する場合) の属性の名前と一致しない場合、コネクターはこの属性を外部キーとして処理できません。ビジネス・オブジェクト名、および属性名は大文字小文字を区別します。

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

外部キー属性の命名: 複数の親ビジネス・オブジェクトに、同一の子ビジネス・オブジェクトを含めることができます。このとき、子ビジネス・オブジェクトは、単

一カーディナリティーの関係にあっても、複数カーディナリティーの関係にあってもかまいません、また、親/子関係は、親に保管しても、子に保管してもかまいません。ただし、親/子関係が保管される親ビジネス・オブジェクトは、いずれも同じ名前の属性を使用して、子の基本キーを格納しなければなりません。さらに、親/子関係が保管される子ビジネス・オブジェクトは、いずれも同じ名前の属性を使用して、親の基本キーを格納しなければなりません。図 20 にこれらの関係を示します。

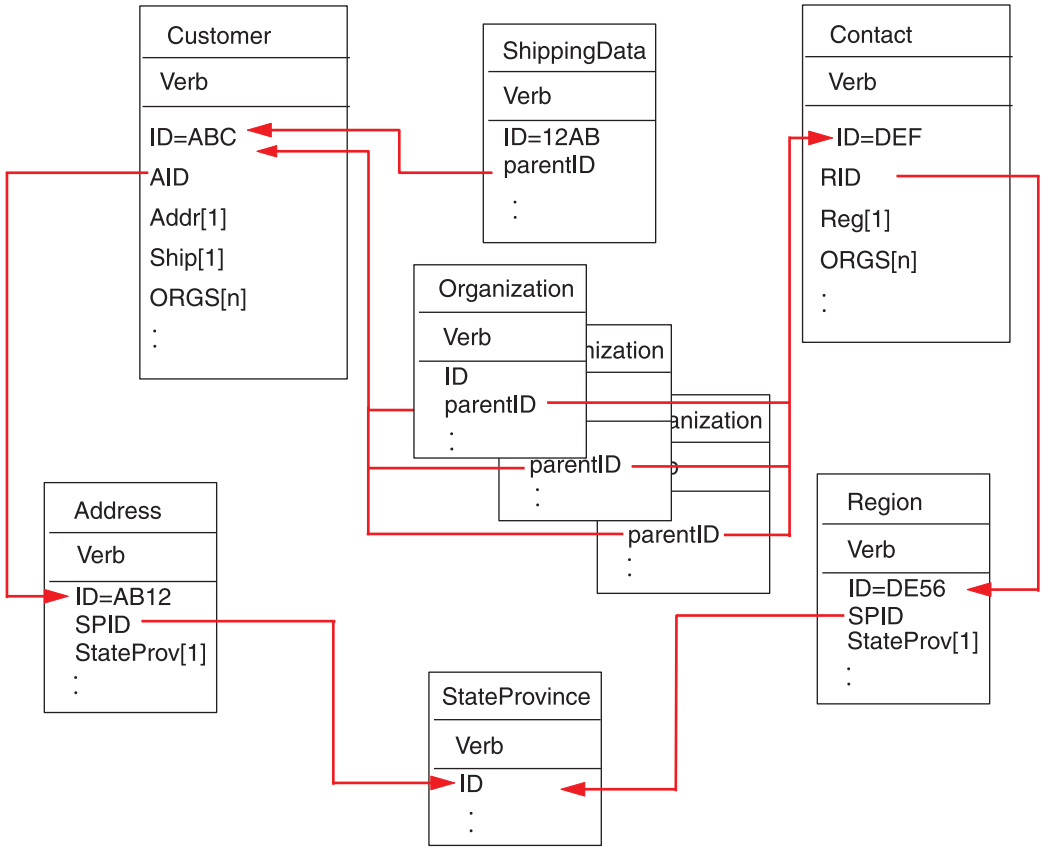


図 20. ビジネス・オブジェクト間の関係の例

図 20 に以下の関係を示します。

- Customer (ID=ABC) および Contact (ID=DEF) の ORGS[n] 属性は、Organization ビジネス・オブジェクトの配列を表しています。Organization ビジネス・オブジェクトの配列に含まれる各ビジネス・オブジェクトの外部キー値は、Customer ビジネス・オブジェクトおよび Contact ビジネス・オブジェクトに含まれる ID 属性の基本キー値に対応しています。この場合、配列内の各ビジネス・オブジェクトは、複数の親に含まれています。

  ORGS 属性のアプリケーション固有の情報は以下のようになっています。

  ```
  KEEP_RELATIONSHIP=true
  ```

KEEP_RELATIONSHIP パラメーターに関する詳細については、110 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アプリ

ケーション・データベース内の列の名前が含まれています。また、この属性の外部キーとなる、親の基本キー属性の名前も含まれています。これらは、次のような形式で指定されています。

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

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

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

  `Addr` 属性のアプリケーション固有の情報は、次のとおりです。

  ```
  CONTAINMENT=OWNERSHIP
  ```

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

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

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

  ```
  CN=pk
  ```

- Address ビジネス・オブジェクトおよび Region ビジネス・オブジェクトの `StateProv[1]` 属性は、所有関係にない StateProvince ビジネス・オブジェクトを表しています。Address ビジネス・オブジェクトおよび Region ビジネス・オブジェクトの `SPID` 属性には、子ビジネス・オブジェクトのタイプ (StateProvince) と、この子ビジネス・オブジェクトの基本キー属性 (親の外部キー) の名前が含まれています。このようにして、複数の親に、同じ単一カーディナリティーの子 (StateProvince) が含まれています。

  `SPID` 属性のアプリケーション固有の情報は、次のとおりです。

  ```
  CONTAINMENT=NOOWNERSHIP
  ```

  `CONTAINMENT` パラメーターに関する詳細については、110 ページの『子を表す属性のアプリケーション固有情報』を参照してください。

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

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

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

```
CN=SP_ID
```

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

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

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

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

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

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=CW.*uidcolumnname* の場合、コネクターは WebSphere のカウンター表を使用して、属性に固有の ID を生成します。この表の作成時には、id 列のみが含まれています。この表の名前は変更することができます。この表をカスタマイズすることにより、UID (固有 ID) の生成を必要とする属性ごとに列を 1 つずつ追加することができます。固有 ID の生成時にコネクターに使用させる列の名前を指定するには、*uidcolumnname* パラメーターを使用します。コネクターでは、UID の生成を必要とする列に関しては数値データ型のみがサポートされていることに注意してください。

  WebSphere のカウンター表を使用して 1 つの属性のみの UID を生成する場合は、integer タイプ (データベースに適したデータ型を選択してください) の 1 列の表を作成します。コネクター・プロパティー *UniqueIdTableName* を適切な表名で構成します。

  WebSphere のカウンター表を使用して複数の属性の UID を生成する場合は、integer タイプのみの列を属性の数と同じだけ持つ表を作成します。

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

処理中に固有 ID シーケンスを保持する方法については、 66 ページの『PreserveUIDSeq』プロパティーを参照してください。

子を表す属性のアプリケーション固有情報

単一カーディナリティーの子ビジネス・オブジェクトを表す属性では、その子が親に所有されるか、または複数の親の間で共用されるかを指定することができます。

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

単一カーディナリティーの子ビジネス・オブジェクトを表す属性: 単一カーディナリティーの子を表す属性の、アプリケーション固有情報の形式は、次のとおりです。

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

親ビジネス・オブジェクトが子ビジネス・オブジェクトを所有する単一カーディナリティーの関係を表すには、CONTAINMENT を OWNERSHIP に設定します。親ビジネス・オブジェクトが子ビジネス・オブジェクトを共用する単一カーディナリティーの関係を表すには、CONTAINMENT を NO_OWNERSHIP に設定します。関係が親ではなく子に保管される単一カーディナリティーの関係を表す場合は、CONTAINMENT パラメーターを含めないでください。

詳細については、 78 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 80 ページの『ラッパー・オブジェクト』を参照してください。

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

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

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

このアプリケーション固有情報の形式は、次のとおりです。

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

このアプリケーション固有情報の検査の際には、大文字小文字は区別されません。

バイナリー・データ使用の作業: BYTEARRAY=true の場合、コネクターはデータベースに対するバイナリー・データの読み取りおよび書き込みを実行します。WebSphere Business Integration システム・フレームワークの現行バージョンではバイナリー・データがサポートされないため、バイナリー・データは String に変換されてから統合ブローカーに送信されます。このストリングの形式は、1 バイトに

つき 2 文字を使用した 16 進数になります。例えば、データベース内のバイナリー・データが 3 バイトで (10 進数の) 値が (1, 65, 255) の場合、ストリングは ″0141ff″ となります。

動詞のアプリケーション固有情報形式

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

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

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

ここで、以下のように説明されます。

condition_variable	アプリケーション・データベース列の名前。
conditonal_operator	アプリケーション・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、および IN (*value1*、*value2*)。
@	getAttrValue(*attribute_name*) によって取得した値で置換される変数。置換は定位置形式です。このため、コネクターは、: 区切り文字の後に指定された最初の *attribute_name* 変数の値で最初の @ を置換します。
..	attribute_name 変数に指定されている属性は、直接の親にあたるビジネス・オブジェクトに属すると見なされます。この値が欠落している場合は、現在のビジネス・オブジェクトに属すると見なされます。
attribute_name	コネクターが @ を置換する値を持つ属性の名前。

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

```
Color='RED'
```

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

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

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

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

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

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

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

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

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

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

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

例: Order ビジネス・オブジェクトに複数カーディナリティーの品目用ビジネス・オブジェクトが含まれています。このうち、特定の品目のみを検索する必要があります。この検索は、Order ビジネス・オブジェクト内のプレースホルダー属性を使用して処理することができます。子オブジェクトはすべて除去されるので、このプレースホルダーは親オブジェクトに含まれていなければなりません。ICS が統合ブローカーの場合、このプレースホルダー属性は、コンマ (,) で区切られた、特定の勘定項目のリストを持つマップによって実行時にデータを取り込むことができます。

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(0):..placeholder
```

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

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

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

RetrieveByContent 動詞で、WHERE 文節の長さが 0 の場合、コネクターは RETRIEVE ステートメントの WHERE 文節内のアプリケーション固有情報を使用します。この機能を使用すると、ユーザーは属性値が取り込まれていないビジネス・オブジェクトを送信し、RetrieveByContent に動詞に関するアプリケーション固

有情報を指定できます。また、コネクターは動詞に関するアプリケーション固有情報のみに指定された情報に基づいて WHERE 文節を作成できます。

第 6 章 トラブルシューティングおよびエラー処理

この章では、MAS コネクターの始動または稼働時に発生する可能性のある問題について説明します。以下のセクションがあります。

- 『始動時の問題』
- 『イベント処理』
- 『マッピング (ICS 統合ブローカーのみ)』
- 『エラー処理とロギング』

始動時の問題

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

イベント処理

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

- 関連するビジネス・オブジェクト要求が稼働している。
- イベント表内のビジネス・オブジェクトの名前が、関連するポートに指定したビジネス・オブジェクトの名前と一致している。

マッピング (ICS 統合ブローカーのみ)

ビジネス・オブジェクトがマップされていないかまたはマッピングが呼び出されていない場合は、適切なディレクトリーにマップがインストールされていることを確認します。

エラー処理とロギング

コネクターは、ビジネス・オブジェクトと動詞の現在の処理がエラーとなる条件を検出すると、常にエラー・メッセージをログに記録します。また、そのようなエラーが発生した場合、コネクターは、処理に失敗したビジネス・オブジェクトが、受信時点でどのような状態であったかを示すテキスト表現も出力します。テキストは、コネクターの構成に応じて、コネクター・ログ・ファイルまたは標準の出力ストリームに書き込まれます。エラーの発生元を判別するための補助資料としてこのテキストを使用できます。

エラー・タイプ

116 ページの表 29 では、コネクターが各トレース・レベルで出力するトレース・メッセージのタイプについて説明します。これらのメッセージは、WebSphere アーキテクチャーによって出力されるトレース・メッセージに追加されます。

表 29. コネクター・トレース・メッセージ

トレース・レベル	トレース・メッセージ
レベル 0	コネクターのバージョンを識別するメッセージ。このレベルでは他のトレースは行われません。これはデフォルト値です。
レベル 1	• 状況メッセージ。 • 処理される各ビジネス・オブジェクトの識別 (キー) 情報を指定するメッセージ。 • pollForEvents メソッドが実行されるたびに配信されるメッセージ。
レベル 2	• コネクターがビジネス・オブジェクトを処理時に検出または検索する配列や子ビジネス・オブジェクトなどの情報が格納されるビジネス・オブジェクト・ハンドラー・メッセージ。 • `gotApplEvent()` または `executeCollaboration()` のいずれかから、ビジネス・オブジェクトが InterChange Server に送られるたびにログに記録されるメッセージ。 • ビジネス・オブジェクトを要求どおりに受け取ったことを示すメッセージ。
レベル 3	• コネクターがビジネス・オブジェクト内で外部キーをいつ検出または設定したかなどの情報を格納している、外部キー処理メッセージ。 • ビジネス・オブジェクト処理に関する情報を提供するメッセージ。例えば、これらのメッセージは、コネクターがビジネス・オブジェクト間の一致を検出したとき、または子ビジネス・オブジェクトの配列でビジネス・オブジェクトを検出したときにデリバリーされます。
レベル 4	• 例えば、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した機能から戻された値を示すメッセージなどの、アプリケーション固有のテキスト・メッセージ。 • コネクターが、コネクターのプロセス・フローのトレースに役立つ関数を開始または終了したかどうかを識別するメッセージ。 • 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。

表 29. 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 5	• コネクターの初期設定を示すメッセージ。例えば、統合ブローカーから検索した各構成プロパティーの値を示すメッセージ。 • アプリケーションで実行されるステートメントが格納されるメッセージ。このトレース・レベルでは、コネクターのログ・ファイルには、宛先アプリケーションで実行されるすべてのステートメントと置換される任意の変数の値が格納されます。 • 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・オブジェクトのダンプからなるメッセージ。 • 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。

付録 A. コネクターの標準構成プロパティー

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

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

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

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

Adapter コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有のプロパティー

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

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

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

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

標準プロパティーの要約

表 30 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

表 30. 標準構成プロパティーの要約

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

表 30. 標準構成プロパティーの要約 (続き)

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

表 30. 標準構成プロパティーの要約 (続き)

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

表 30. 標準構成プロパティーの要約 (続き)

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

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

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

ConcurrentEventTriggeredFlows

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

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

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

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

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

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

ContainerManagedEvents

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

デフォルト値は No value です。

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

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

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

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

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

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

JMS

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

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

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

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

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

- CWSharedEnv.sh スクリプト内で LDR_CNTRL 環境変数を設定する。

  このスクリプトは、製品ディレクトリー配下の ¥bin ディレクトリーにあります。テキスト・エディターを使用して、CWSharedEnv.sh スクリプトの最初の行として次の行を追加します。

  ```
  export LDR_CNTRL=MAXDATA=0x30000000
  ```

  この行は、ヒープ・メモリーの使用量を最大 768 MB (3 セグメント * 256 MB) に制限します。プロセス・メモリーがこの制限値を超えると、ページ・スワッピングが発生し、システムのパフォーマンスに悪影響を与える場合があります。
- IPCCBaseAddress プロパティーの値を 11 または 12 に設定する。このプロパティーの詳細については、「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

DuplicateEventElimination

このプロパティーを true に設定すると、JMS 対応コネクターによるデリバリー・キューへの重複イベントのデリバリーが防止されます。この機能を使用するには、コネクターに対し、アプリケーション固有のコード内でビジネス・オブジェクトの **ObjectEventId** 属性として一意のイベント ID が設定されている必要があります。これはコネクター開発時に設定されます。

このプロパティーは、false に設定することもできます。

注: DuplicateEventElimination を true に設定する際は、MonitorQueue プロパティーを構成して保証付きイベント・デリバリーを使用可能にする必要があります。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、そのメッセージを状況表示および問題説明とともにこのプロパティーに指定されているキューに移動します。

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128M です。

JvmMaxNativeStackSize

エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128K です。

JvmMinHeapSize

エージェントの最小ヒープ・サイズ (メガバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 1M です。

jms.FactoryClassName

JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクター・プロパティーを必ず設定してください。

デフォルト値は CxCommon.Messaging.jms.IBMMQSeriesFactory です。

jms.MessageBrokerName

JMS プロバイダーのために使用するブローカー名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクター・プロパティーを必ず設定してください。

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

コネクターに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出し要求はブロックされ、既存のいずれかの要求が完了した後で処理されます。

デフォルト値は 10 です。

jms.Password

JMS プロバイダーのためのパスワードを指定します。このプロパティーの値はオプションです。

デフォルトはありません。

jms.UserName

JMS プロバイダーのためのユーザー名を指定します。このプロパティーの値はオプションです。

デフォルトはありません。

ListenerConcurrency

このプロパティーは、統合ブローカーとして ICS を使用する場合の MQ Listener でのマルチスレッド化をサポートしています。このプロパティーにより、データベースへの複数イベントの書き込み操作をバッチ処理できるので、システム・パフォーマンスが向上します。デフォルト値は 1 です。

このプロパティーは、MQ トランスポートを使用するコネクターにのみ適用されます。DeliveryTransport プロパティーには MQ を設定してください。

Locale

言語コード、国または地域、および、希望する場合には、関連した文字コード・セットを指定します。このプロパティーの値は、データの照合やソート順、日付と時刻の形式、通貨記号などの国/地域別情報を決定します。

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

ll	2 文字の言語コード (普通は小文字)
TT	2 文字の国または地域コード (普通は大文字)
codeset	関連文字コード・セットの名前。名前のこの部分は、通常、オプションです。

デフォルトでは、ドロップ・リストには、サポートされるロケールの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、Connector Configurator に関する付録を参照してください。

デフォルト値は en_US です。コネクターがグローバル化に対応していない場合、このプロパティーの有効な値は en_US のみです。特定のコネクターがグローバル化に対応しているかどうかを判別するには、以下の Web サイトにあるコネクターのバージョン・リストを参照してください。

http://www.ibm.com/software/websphere/wbiadapters/infocenter、または
http://www.ibm.com/websphere/integration/wicserver/infocenter

LogAtInterchangeEnd

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

統合ブローカーのログ宛先にエラーを記録するかどうかを指定します。ブローカーのログ宛先にログを記録すると、電子メール通知もオンになります。これにより、エラーまたは致命的エラーが発生すると、InterchangeSystem.cfg ファイルに指定された MESSAGE_RECIPIENT に対する電子メール・メッセージが生成されます。

例えば、LogAtInterChangeEnd を true に設定した場合にコネクターからアプリケーションへの接続が失われると、指定されたメッセージ宛先に、電子メール・メッセージが送信されます。デフォルト値は false です。

MaxEventCapacity

コントローラー・バッファー内のイベントの最大数。このプロパティーはフロー制御が使用し、RepositoryDirectory プロパティーの値が <REMOTE> の場合にのみ適用されます。

値は 1 から 2147483647 の間の正整数です。デフォルト値は 2147483647 です。

MessageFileName

コネクター・メッセージ・ファイルの名前です。メッセージ・ファイルの標準位置は ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクター・メッセージ・ファイルが存在しない場合は、コネクターは InterchangeSystem.txt をメッセージ・ファイルとして使用します。このファイルは、製品ディレクトリーに格納されています。

注: 特定のコネクターについて、コネクター独自のメッセージ・ファイルがあるかどうかを判別するには、該当するアダプターのユーザーズ・ガイドを参照してください。

MonitorQueue

コネクターが重複イベントをモニターするために使用する論理キューです。このプロパティーは、DeliveryTransport プロパティー値が JMS であり、かつ DuplicateEventElimination が TRUE に設定されている場合にのみ使用されます。

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

RepositoryDirectory が <REMOTE> の場合のみ有効です。

コネクターが自動再始動およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用して、異常シャットダウン後にコネクターを再始動したり、System Monitor からリモート・コネクターを始動したりします。

自動再始動機能およびリモート再始動機能を使用可能にするには、このプロパティーを true に設定する必要があります。MQ により起動される OAD 機能の構成方法については、「*システム・インストール・ガイド (Windows 版)*」または「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

RepositoryDirectory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後で MQ により起動される OAD がコネクターの再始動を自動的に試行する回数の最大数を指定します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

RepositoryDirectory が <REMOTE> の場合のみ有効です。

MQ により起動される OAD の再試行時間間隔の分数を指定します。コネクター・エージェントがこの再試行時間間隔内に再始動しない場合は、コネクター・コントローラーはコネクター・エージェントを再び再始動するように OAD に要求します。OAD はこの再試行プロセスを OADMaxNumRetry プロパティーで指定された回数だけ繰り返します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 10 です。

PollEndTime

イベント・キューのポーリングを停止する時刻です。形式は HH:MM です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

PollFrequency

ポーリング・アクション間の時間の長さです。PollFrequency は以下の値のいずれかに設定します。

- ポーリング・アクション間のミリ秒数。
- ワード key。コネクターは、コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力されたときにのみポーリングを実行します。このワードは小文字で入力します。
- ワード no。コネクターはポーリングを実行しません。このワードは小文字で入力します。

デフォルト値は 10000 です。

重要: 一部のコネクターでは、このプロパティーの使用が制限されています。このプロパティーが使用されるかどうかを特定のコネクターについて判別するには、該当するアダプター・ガイドのインストールと構成についての章を参照してください。

PollQuantity

コネクターがアプリケーションからポーリングする項目の数を指定します。アダプターにコネクター固有のポーリング数設定プロパティーがある場合、標準プロパティーの値は、このコネクター固有のプロパティーの設定値によりオーバーライドされます。

PollStartTime

イベント・キューのポーリングを開始する時刻です。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

RequestQueue

統合ブローカーが、ビジネス・オブジェクトをコネクターに送信するときに使用されるキューです。

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

コネクターが XML スキーマ文書を読み取るリポジトリーの場所です。この XML スキーマ文書には、ビジネス・オブジェクト定義のメタデータが含まれています。

統合ブローカーが ICS の場合はこの値を <REMOTE> に設定する必要があります。これは、コネクターが InterChange Server リポジトリーからこの情報を取得するためです。

統合ブローカーが WebSphere Message Broker または WAS の場合は、この値を <*local directory*> に設定する必要があります。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用可能で、RepositoryDirectory が <REMOTE> の場合のみ必須です。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・フレームワークから統合ブローカーへデリバリーします。統合ブローカーが ICS の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartRetryCount

コネクターによるコネクター自体の再始動の試行回数を指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する回数が指定されます。

デフォルト値は 3 です。

RestartRetryInterval

コネクターによるコネクター自体の再始動の試行間隔を分単位で指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する間隔が指定されます。指定可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

WebSphere Message Brokers および WAS でのみ使用されます。

このプロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WMQI に送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 mrm が書き込まれます。この構成可能なドメイン名により、ユーザーは WMQI ブローカーによるメッセージ・データの処理方法を追跡できます。

サンプル・ヘッダーを以下に示します。

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

デフォルト値は mrm ですが、このプロパティーには xml も設定できます。このプロパティーは、DeliveryTransport が JMS に設定されており、かつ WireFormat が CwXML に設定されている場合にのみ表示されます。

SourceQueue

DeliveryTransport が JMS で、ContainerManagedEvents が指定されている場合のみ適用されます。

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、126 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期応答を要求する要求メッセージを、コネクター・フレームワークからブローカーに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクター・フレームワークは、SynchronousRequestQueue にメッセージを送信し、SynchronousResponseQueue でブローカーから戻される応答を待機します。コネクターに送信される応答メッセージには、元のメッセージの ID を指定する 相関 ID が含まれています。

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期要求に対する応答として送信される応答メッセージを、ブローカーからコネクター・フレームワークに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

- RepositoryDirectory がローカル・ディレクトリーの場合は、設定は CwXML になります。
- RepositoryDirectory の値が <REMOTE> の場合には、設定値は CwBO です。

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

XMLNameSpaceFormat

WebSphere Message Brokers および WAS 統合ブローカーでのみ使用されます。

ビジネス・オブジェクト定義の XML 形式でネーム・スペースを short と long のどちらにするかをユーザーが指定できるようにするための、強力なプロパティーです。

デフォルト値は short です。

付録 B. Connector Configurator

この付録では、Connector Configurator を使用してアダプターの構成プロパティー値を設定する方法について説明します。

Connector Configurator を使用して次の作業を行います。

- コネクターを構成するためのコネクター固有のプロパティー・テンプレートを作成する
- 構成ファイルを作成する
- 構成ファイル内のプロパティーを設定する

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

Connector Configurator では、次の統合ブローカーで使用するアダプターのコネクター・コンポーネントを構成できます。

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

Connector Configurator を使用して次の作業を行います。

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成する。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定する。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、ICS の場合はコラボレーションとともに使用するマップを

指定し、必要に応じてメッセージング、ロギング、トレース、およびデータ・ハンドラー・パラメーターを指定する必要があります。

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なります。例えば、使用している統合ブローカーが WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (138 ページの『スタンドアロン・モードでの Configurator の実行』を参照)。

コネクター構成プロパティーには、標準の構成プロパティー (すべてのコネクターがもつプロパティー) と、コネクター固有のプロパティー (特定のアプリケーションまたはテクノロジーのためにコネクターで必要なプロパティー) とが含まれます。

標準プロパティーはすべてのコネクターにより使用されるので、標準プロパティーを新規に定義する必要はありません。ファイルを作成すると、Connector Configurator により標準プロパティーがこの構成ファイルに挿入されます。ただし、Connector Configurator で各標準プロパティーの値を設定する必要があります。

標準プロパティーの範囲は、ブローカーと構成によって異なる可能性があります。特定のプロパティーに特定の値が設定されている場合にのみ使用できるプロパティーがあります。Connector Configurator の「標準のプロパティー」ウィンドウには、特定の構成で設定可能なプロパティーが表示されます。

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既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、139 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

注: Connector Configurator は、Windows 環境内でのみ実行されます。UNIX 環境でコネクターを実行する場合には、Windows で Connector Configurator を使用して構成ファイルを変更し、このファイルを UNIX 環境へコピーします。

Connector Configurator の始動

以下の 2 種類のモードでConnector Configurator を開始および実行できます。

- スタンドアロン・モードで個別に実行
- System Manager から

スタンドアロン・モードでの Configurator の実行

どのブローカーを実行している場合にも、Connector Configurator を個別に実行し、コネクター構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Connector Configurator」**をクリックします。
- **「ファイル」>「新規」>「構成ファイル」**を選択します。

- **「システム接続: Integration Broker」**の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (144 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

System Manager から Connector Configurator を実行できます。

Connector Configurator を実行するには、以下のステップを実行します。

1. System Manager を開きます。
2. 「System Manager」ウィンドウで、**「統合コンポーネント・ライブラリー」**アイコンを展開し、**「コネクター」**を強調表示します。
3. System Manager メニュー・バーから、**「ツール」>「Connector Configurator」**をクリックします。「Connector Configurator」ウィンドウが開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。
4. **「システム接続: Integration Broker」**の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーでいずれかの構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
2. 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

コネクター固有プロパティーのテンプレートを新規に作成するか、または既存のファイルをテンプレートとして使用します。

- テンプレートの新規作成については、139 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティーを作成し、プロパティーの一般特性および値を定義し、プロパティー間の依存関係を指定する方法について説明します。次にそのテンプレートを保管し、新規コネクター構成ファイルを作成するためのベースとして使用します。

テンプレートは以下のように作成します。

1. **「ファイル」>「新規」>「コネクター固有プロパティー・テンプレート」**をクリックします。
2. 以下のフィールドを含む**「コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。
   - **「テンプレート」、「名前」**

     このテンプレートが使用されるコネクター (またはコネクターのタイプ) を表す固有の名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - **「旧テンプレート」、「変更する既存のテンプレートを選択してください」**

     「テンプレート名」表示に、現在使用可能なすべてのテンプレートの名前が表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、**「テンプレート名」**表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが**「テンプレートのプレビュー」**表示に表示されます。テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。
3. **「テンプレート名」**表示からテンプレートを選択し、その名前を**「名前の検索」**フィールドに入力し (または**「テンプレート名」**で自分の選択項目を強調表示し)、**「次へ」**をクリックします。

ご使用のコネクターで使用するコネクター固有のプロパティーが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。

一般特性の指定

「次へ」をクリックしてテンプレートを選択すると、**「プロパティー: コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスには、定義済みプロパティーの「一般」特性のタブと「値」の制限のタブがあります。「一般」表示には以下のフィールドがあります。

- **一般:**
  プロパティー・タイプ
  更新されたメソッド
  説明
- **フラグ**
  標準フラグ
- **カスタム・フラグ**
  フラグ

プロパティーの一般特性の選択を終えたら、**「値」**タブをクリックします。

値の指定

「値」タブを使用すると、プロパティーの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。編集可能な値も許可されます。これを行うには、以下のステップを実行します。

1. **「値」**タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドで、変更を行います。次のステップで説明するように、プロパティーの**「プロパティー値」**ダイアログ・ボックスを開かない限り、そのプロパティーの変更内容は受け入れられませんので、注意してください。
4. 値テーブルの左上の隅にあるボックスを右マウス・ボタンでクリックしてから、**「追加」**をクリックします。**「プロパティー値」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスではプロパティーのタイプに応じて、値だけを入力できる場合と、値と範囲の両方を入力できる場合があります。適切な値または範囲を入力し、**「OK」**をクリックします。
5. **「値」**パネルが最新表示され、**「最大長」**および**「最大複数値」**で行った変更が表示されます。以下のような 3 つの列があるテーブルが表示されます。

   「値」の列には、**「プロパティー値」**ダイアログ・ボックスで入力した値と、以前に作成した値が表示されます。

   「デフォルト値」の列では、値のいずれかをデフォルトとして指定することができます。

   「値の範囲」の列には、**「プロパティー値」**ダイアログ・ボックスで入力した範囲が表示されます。

   値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。テーブルにある既存の値の変更を行うには、その行の行番号をクリックして行全体を選択します。次に**「値」**フィールドを右マウス・ボタンでクリックし、**「値の編集 (Edit Value)」**をクリックします。

依存関係の設定

「一般」タブと**「値」**タブで変更を行ったら、**「次へ」**をクリックします。**「依存関係: コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。

依存プロパティーは、別のプロパティーの値が特定の条件に合致する場合にのみ、テンプレートに組み込まれて、構成ファイルで使用されるプロパティーです。例えば、テンプレートに `PollQuantity` が表示されるのは、トランスポート機構が JMS であり、`DuplicateEventElimination` が `True` に設定されている場合のみです。プロパティーを依存プロパティーとして指定し、依存する条件を設定するには、以下のステップを実行します。

1. **「使用可能なプロパティー」**表示で、依存プロパティーとして指定するプロパティーを選択します。
2. **「プロパティーを選択」**フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティーを選択します。
3. **「条件演算子」**フィールドで以下のいずれかを選択します。

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

>= (より大か等しい)

<= (より小か等しい)

4. **「条件値」**フィールドで、依存プロパティーをテンプレートに組み込むために必要な値を入力します。
5. **「使用可能なプロパティー」**表示で依存プロパティーを強調表示させて矢印をクリックし、**「依存プロパティー」**表示に移動させます。
6. **「完了」**をクリックします。Connector Configurator により、XML 文書として入力した情報が、Connector Configurator がインストールされている ¥bin ディレクトリーの ¥data¥app の下に保管されます。

新しい構成ファイルを作成

構成ファイルを新規に作成するには、最初に統合ブローカーを選択します。選択したブローカーによって、構成ファイルに記述されるプロパティーが決まります。

ブローカーを選択するには、以下のステップを実行します。

- Connector Configurator のホーム・メニューで、**「ファイル」>「新規」>「コネクター構成」**をクリックします。**「新規コネクター」**ダイアログ・ボックスが表示されます。
- **「Integration Broker」**フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って**「新規コネクター」**ウィンドウの残りのフィールドに入力します。

また、以下の作業も実行できます。

- 「System Manager」ウィンドウで**「コネクター」**フォルダーを右クリックし、**「新規コネクターの作成」**を選択します。Connector Configurator が開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。

コネクター固有のテンプレートからの構成ファイルの作成

コネクター固有のテンプレートを作成すると、テンプレートを使用して構成ファイルを作成できます。

1. **「ファイル」>「新規」>「コネクター構成」**をクリックします。
2. 以下のフィールドを含む**「新規コネクター」**ダイアログ・ボックスが表示されます。
   - **名前**

     コネクターの名前を入力します。名前では大文字と小文字が区別されます。入力する名前は、システムにインストールされているコネクターのファイル名に対応した一意の名前でなければなりません。

     重要: Connector Configurator では、入力された名前のスペルはチェックされません。名前が正しいことを確認してください。
   - **システム接続**

     ICS 接続、WebSphere Message Brokers 接続、WAS のいずれかをクリックします。

- **「コネクター固有プロパティー・テンプレート」を選択します**

  ご使用のコネクター用に設計したテンプレートの名前を入力します。**「テンプレート名」**表示に、使用可能なテンプレートが表示されます。「テンプレート名」表示で名前を選択すると、**「プロパティー・テンプレートのプレビュー」**表示に、そのテンプレートで定義されているコネクター固有のプロパティーが表示されます。

  使用するテンプレートを選択し、**「OK」**をクリックします。

3. 構成しているコネクターの構成画面が表示されます。タイトル・バーに統合ブローカーとコネクターの名前が表示されます。ここですべてのフィールドに値を入力して定義を完了するか、ファイルを保管して後でフィールドに値を入力するかを選択できます。
4. ファイルを保管するには、**「ファイル」>「保管」>「ファイルに」**をクリックするか、**「ファイル」>「保管」>「プロジェクトに」**をクリックします。プロジェクトに保管するには、System Manager が実行中でなければなりません。
   ファイルとして保管する場合は、**「ファイル・コネクターを保管」**ダイアログ・ボックスが表示されます。`*.cfg` をファイル・タイプとして選択し、「ファイル名」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、**「保管」**をクリックします。Connector Configurator のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

   重要: ここで設定するディレクトリー・パスおよび名前は、コネクターの始動ファイルで指定するコネクター構成ファイルのパスおよび名前に一致している必要があります。
5. この章で後述する手順に従って、「Connector Configurator」ウィンドウの各タブにあるフィールドに値を入力し、コネクター定義を完了します。

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

- コネクター定義ファイル。
  コネクター定義ファイルは、特定のコネクターのプロパティーと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクターの配布パッケージの `¥repository` ディレクトリー内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は `.txt` です。例えば、XML コネクターの場合は CN_XML.txt です)。
- ICS リポジトリー・ファイル。
  コネクターの以前の ICS インプリメンテーションで使用した定義は、そのコネクターの構成で使用されたリポジトリー・ファイルで使用可能になります。そのようなファイルの拡張子は、通常 `.in` または `.out` です。
- コネクターの以前の構成ファイル。
  これらのファイルの拡張子は、通常 *.cfg です。

これらのいずれのファイル・ソースにも、コネクターのコネクター固有プロパティーのほとんど、あるいはすべてが含まれますが、この章内の後で説明するように、コネクター構成ファイルは、ファイルを開いて、プロパティーを設定しない限り完成しません。

既存ファイルを使用してコネクターを構成するには、Connector Configurator でそのファイルを開き、構成を修正し、そのファイルを再度保管する必要があります。

以下のステップを実行して、ディレクトリーから *.txt、*.cfg、または *.in ファイルを開きます。

1. Connector Configurator 内で、**「ファイル」>「開く」>「ファイルから」**をクリックします。
2. **「ファイル・コネクターを開く」**ダイアログ・ボックス内で、以下のいずれかのファイル・タイプを選択して、使用可能なファイルを調べます。
   - 構成 (*.cfg)
   - ICS リポジトリー (*.in、*.out)

     ICS 環境でのコネクターの構成にリポジトリー・ファイルが使用された場合には、このオプションを選択します。リポジトリー・ファイルに複数のコネクター定義が含まれている場合は、ファイルを開くとすべての定義が表示されます。
   - すべてのファイル (*.*)

     コネクターのアダプター・パッケージに *.txt ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。
3. ディレクトリー表示内で、適切なコネクター定義ファイルへ移動し、ファイルを選択し、**「開く」**をクリックします。

System Manager プロジェクトからコネクター構成を開くには、以下のステップを実行します。

1. System Manager を始動します。System Manager が開始されている場合にのみ、構成を System Manager から開いたり、System Manager に保管したりできます。
2. Connector Configurator を始動します。
3. **「ファイル」>「開く」>「プロジェクトから」**をクリックします。

構成ファイルの完成

構成ファイルを開くか、プロジェクトからコネクターを開くと、「Connector Configurator」ウィンドウに構成画面が表示されます。この画面には、現在の属性と値が表示されます。

構成画面のタイトルには、ファイル内で指定された統合ブローカーとコネクターの名前が表示されます。正しいブローカーが設定されていることを確認してください。正しいブローカーが設定されていない場合、コネクターを構成する前にブローカー値を変更してください。これを行うには、以下のステップを実行します。

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 147 ページの

『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。

3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

   ファイルに保管する場合は、`*.cfg` を拡張子として選択し、ファイルの正しい格納場所を選択して、**「保管」**をクリックします。

   複数のコネクター構成を開いている場合、構成をすべてファイルに保管するには**「すべてファイルに保管」**を選択し、コネクター構成をすべて System Manager プロジェクトに保管するには**「すべてプロジェクトに保管」**をクリックします。

   Connector Configurator では、ファイルを保管する前に、必須の標準プロパティーすべてに値が設定されているかどうかが確認されます。必須の標準プロパティーに値が設定されていない場合、Connector Configurator は、検証が失敗したというメッセージを表示します。構成ファイルを保管するには、そのプロパティーの値を指定する必要があります。

構成ファイル・プロパティーの設定

新規のコネクター構成ファイルを作成して名前を付けるとき、または既存のコネクター構成ファイルを開くときには、Connector Configurator によって構成画面が表示されます。構成画面には、必要な構成値のカテゴリーに対応する複数のタブがあります。

Connector Configurator では、すべてのブローカーで実行されているコネクターで、以下のカテゴリーのプロパティーに値が設定されている必要があります。

- 標準のプロパティー
- コネクター固有のプロパティー
- サポートされるビジネス・オブジェクト
- トレース/ログ・ファイルの値
- データ・ハンドラー (保証付きイベント・デリバリーで JMS メッセージングを使用するコネクターの場合に該当する)

注: JMS メッセージングを使用するコネクターの場合は、データをビジネス・オブジェクトに変換するデータ・ハンドラーの構成に関して追加のカテゴリーが表示される場合があります。

ICS で実行されているコネクターの場合、以下のプロパティーの値も設定されている必要があります。

- 関連付けられたマップ
- リソース
- メッセージング (該当する場合)

重要: Connector Configurator では、英語文字セットまたは英語以外の文字セットのいずれのプロパティー値も設定可能です。ただし、標準のプロパティーおよびコネクター固有プロパティー、およびサポートされるビジネス・オブジェクトの名前では、英語文字セットのみを使用する必要があります。

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

- コネクターの標準プロパティーは、コネクターのアプリケーション固有のコンポーネントとブローカー・コンポーネントの両方によって共用されます。すべてのコネクターが同じ標準プロパティーのセットを使用します。これらのプロパティーの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。
- アプリケーション固有のプロパティーは、コネクターのアプリケーション固有コンポーネント (アプリケーションと直接対話するコンポーネント) のみに適用されます。各コネクターには、そのコネクターのアプリケーションだけで使用されるアプリケーション固有のプロパティーがあります。これらのプロパティーには、デフォルト値が用意されているものもあれば、そうでないものもあります。また、一部のデフォルト値は変更することができます。各アダプター・ガイドのインストールおよび構成の章に、アプリケーション固有のプロパティーおよび推奨値が記述されています。

「標準プロパティー」と**「コネクター固有プロパティー」**のフィールドは、どのフィールドが構成可能であるかを示すために色分けされています。

- 背景がグレーのフィールドは、標準のプロパティーを表します。値を変更することはできますが、名前の変更およびプロパティーの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティーを表します。これらのプロパティーは、アプリケーションまたはコネクターの特定のニーズによって異なります。値の変更も、これらのプロパティーの除去も可能です。
- 「値」フィールドは構成できます。
- **「更新メソッド」**フィールドは通知用であり、構成できません。このフィールドは、値が変更されたプロパティーをアクティブにするために必要なアクションを示します。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下のステップを実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。
3. 標準のプロパティーの値をすべて入力後、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator を終了するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「いいえ」**をクリックします。
   - Connector Configurator 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。**「標準のプロパティー」**(またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
   - 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

アプリケーション固有の構成プロパティーの場合、プロパティー名の追加または変更、値の構成、プロパティーの削除、およびプロパティーの暗号化が可能です。プロパティーのデフォルトの長さは 255 文字です。

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 146 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

各プロパティーごとに表示される「更新メソッド」は、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。

重要: 事前設定のアプリケーション固有のコネクター・プロパティー名を変更すると、コネクターに障害が発生する可能性があります。コネクターをアプリケーションに接続したり正常に実行したりするために、特定のプロパティー名が必要である場合があります。

コネクター・プロパティーの暗号化

「プロパティーを編集」ウィンドウの**「暗号化」**チェック・ボックスにチェックマークを付けると、アプリケーション固有のプロパティーを暗号化することができます。値の暗号化を解除するには、**「暗号化」**チェック・ボックスをクリックしてチェックマークを外し、**「検証」**ダイアログ・ボックスに正しい値を入力し、**「OK」**をクリックします。入力された値が正しい場合は、暗号化解除された値が表示されます。

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

プロパティーに複数の値がある場合には、プロパティーの最初の値に**「暗号化」**チェック・ボックスが表示されます。**「暗号化」**を選択すると、そのプロパティーのすべての値が暗号化されます。プロパティーの複数の値を暗号化解除するには、そのプロパティーの最初の値の**「暗号化」**チェック・ボックスをクリックしてチェックマークを外してから、**「検証」**ダイアログ・ボックスで新規の値を入力します。入力値が一致すれば、すべての複数値が暗号化解除されます。

更新メソッド

付録『コネクターの*標準構成プロパティー*』の 120 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

コネクターで使用するビジネス・オブジェクトを指定するには、Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブを使用します。

汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

注: コネクターによっては、アプリケーションでイベント通知や (メタオブジェクトを使用した) 追加の構成を実行するために、特定のビジネス・オブジェクトをサポートされているものとして指定することが必要な場合もあります。詳細は、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

ご使用のブローカーが ICS の場合

ビジネス・オブジェクト定義がコネクターでサポートされることを指定する場合や、既存のビジネス・オブジェクト定義のサポート設定を変更する場合は、**「サポートされているビジネス・オブジェクト」**タブをクリックし、以下のフィールドを使用してください。

ビジネス・オブジェクト名: ビジネス・オブジェクト定義がコネクターによってサポートされることを指定するには、System Manager を実行し、以下のステップを実行します。

1. **「ビジネス・オブジェクト名」**リストで空のフィールドをクリックします。System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの**「エージェント・サポート」**(以下で説明) を設定します。
4. 「Connector Configurator」ウィンドウの「ファイル」メニューで、**「プロジェクトに保管」**をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager のプロジェクトに保管されます。

サポートされるリストからビジネス・オブジェクトを削除する場合は、以下のステップを実行します。

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、**「プロジェクトの保管」**をクリックします。

サポートされるリストからビジネス・オブジェクトを削除すると、コネクター定義が変更され、削除されたビジネス・オブジェクトはコネクターのこのインプリメンテーションで使用不可になります。コネクターのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

エージェント・サポート: ビジネス・オブジェクトがエージェント・サポートを備えている場合、システムは、コネクター・エージェントを介してアプリケーションにデータを配布する際にそのビジネス・オブジェクトの使用を試みます。

一般に、コネクターのアプリケーション固有ビジネス・オブジェクトは、そのコネクターのエージェントによってサポートされますが、汎用ビジネス・オブジェクトはサポートされません。

ビジネス・オブジェクトがコネクター・エージェントによってサポートされるよう指定するには、**「エージェント・サポート」**ボックスにチェックマークを付けます。「Connector Configurator」ウィンドウでは「エージェント・サポート」の選択の妥当性は検査されません。

最大トランザクション・レベル: コネクターの最大トランザクション・レベルは、そのコネクターがサポートする最大のトランザクション・レベルです。

ほとんどのコネクターの場合、選択可能な項目は「**最大限の努力**」のみです。

トランザクション・レベルの変更を有効にするには、サーバーを再始動する必要があります。

ご使用のブローカーが WebSphere Message Broker の場合

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス名を入力する必要があります。

System Manager を実行している場合、**「サポートされているビジネス・オブジェクト」**タブの**「ビジネス・オブジェクト名」**列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。リストから必要なビジネス・オブジェクトを選択します。

「メッセージ・セット ID」は、WebSphere Business Integration Message Broker 5.0 のオプションのフィールドです。この ID が提供される場合、一意である必要はありません。ただし、WebSphere MQ Integrator および Integrator Broker 2.1 の場合は、一意の **ID** を提供する必要があります。

ご使用のブローカーが WAS の場合

使用するブローカー・タイプとして WebSphere Application Server を選択した場合、Connector Configurator にメッセージ・セット ID は必要ありません。**「サポートされているビジネス・オブジェクト」**タブには、サポートされるビジネス・オブジェクトの**「ビジネス・オブジェクト名」**列のみが表示されます。

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス・オブジェクト名を入力する必要があります。

System Manager を実行している場合、「サポートされているビジネス・オブジェクト」タブの「ビジネス・オブジェクト名」列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。このリストから必要なビジネス・オブジェクトを選択します。

関連付けられているマップ (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、関連付けられたマップを明示的にバインドすることが必要になります。

  明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバインドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見付けて、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。

以下のをステップを実行して、マップを明示的にバインドします。

1. **「明示的 (Explicit)」**列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
2. ビジネス・オブジェクトに関連付けるマップを選択します。
3. 「Connector Configurator」ウィンドウの**「ファイル」**メニューで、**「プロジェクトに保管」**をクリックします。
4. プロジェクトを ICS に配置します。
5. 変更を有効にするため、サーバーをリブートします。

リソース (ICS)

「リソース」タブでは、コネクター・エージェントが、コネクター・エージェント並列処理を使用して同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定できます。
すべてのコネクターがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

メッセージング・プロパティーは、`DeliveryTransport` 標準プロパティーの値として MQ を設定し、ブローカー・タイプとして ICS を設定した場合にのみ、使用可能です。これらのプロパティーは、コネクターによるキューの使用方法に影響します。

トレース/ログ・ファイル値の設定

コネクター構成ファイルまたはコネクター定義ファイルを開くと、Connector Configurator は、そのファイルのログおよびトレースの値をデフォルト値として使用します。Connector Configurator 内でこれらの値を変更できます。

ログとトレースの値を変更するには、以下のステップを実行します。

1. **「トレース/ログ・ファイル」**タブをクリックします。
2. ログとトレースのどちらでも、以下のいずれかまたは両方へのメッセージの書き込みを選択できます。
   - コンソールに (STDOUT):
     ログ・メッセージまたはトレース・メッセージを STDOUT ディスプレイに書き込みます。

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの**「トレース/ログ・ファイル」**タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、**「保管」**をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

注: ログ・ファイルとトレース・ファイルはどちらも単純なテキスト・ファイルです。任意のファイル拡張子を使用してこれらのファイル名を設定できます。ただし、トレース・ファイルの場合、拡張子として .trc ではなく .trace を使用することをお勧めします。これは、システム内に存在する可能性がある他のファイルとの混同を避けるためです。ログ・ファイルの場合、通常使用されるファイル拡張子は .log および .txt です。

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。すべてのアダプターでデータ・ハンドラーを使用できるわけではありません。

これらのプロパティーに使用する値については、付録 A の『コネクターの標準構成プロパティー』の ContainerManagedEvents の下の説明を参照してください。その他の詳細は、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

構成ファイルの保管

コネクターの構成が完了したら、コネクター構成ファイルを保管します。Connector Configurator では、構成中に選択したブローカー・モードでファイルを保管します。Connector Configurator のタイトル・バーには現在のブローカー・モード (ICS、WMQI、または WAS) が常に表示されます。

ファイルは XML 文書として保管されます。XML 文書は次の 3 通りの方法で保管できます。

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- 指定したディレクトリーに保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。

System Manager でのプロジェクトの使用法、および配置の詳細については、以下のインプリメンテーション・ガイドを参照してください。

- ICS: 「*WebSphere InterChange Server インプリメンテーション・ガイド*」
- WebSphere Message Brokers: 「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」
- WAS: 「*アダプター実装ガイド (WebSphere Application Server)*」

構成ファイルの変更

既存の構成ファイルの統合ブローカー設定を変更できます。これにより、他のブローカーで使用する構成ファイルを新規に作成するときに、このファイルをテンプレートとして使用できます。

注: 統合ブローカーを切り替える場合には、ブローカー・モード・プロパティーと同様に他の構成プロパティーも変更する必要があります。

既存の構成ファイルでのブローカーの選択を変更するには、以下のステップを実行します (オプション)。

- Connector Configurator で既存の構成ファイルを開きます。
- **「標準のプロパティー」**タブを選択します。
- 「標準のプロパティー」タブの**「BrokerType」**フィールドで、ご使用のブローカーに合った値を選択します。
  現行値を変更すると、プロパティー画面の利用可能なタブおよびフィールド選択がただちに変更され、選択した新規ブローカーに適したタブとフィールドのみが表示されます。

構成の完了

コネクターの構成ファイルを作成し、そのファイルを変更した後で、コネクターの始動時にコネクターが構成ファイルの位置を特定できるかどうかを確認してください。

これを行うには、コネクターが使用する始動ファイルを開き、コネクター構成ファイルに使用されている格納場所とファイル名が、ファイルに対して指定した名前およびファイルを格納したディレクトリーまたはパスと正確に一致しているかどうかを検証します。

グローバル化環境における Connector Configurator の使用

Connector Configurator はグローバル化され、構成ファイルと統合ブローカー間の文字変換を処理できます。 Connector Configurator では、ネイティブなエンコード方式を使用しています。構成ファイルに書き込む場合は UTF-8 エンコード方式を使用します。

Connector Configurator は、以下の場所で英語以外の文字をサポートします。

- すべての値のフィールド
- ログ・ファイルおよびトレース・ファイル・パス (**「トレース/ログ・ファイル」**タブで指定)

CharacterEncoding および Locale 標準構成プロパティーのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーの ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。

例えば、Locale プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太文字で示した行を追加してください。

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
                <ValidType>String</ValidType>
            <ValidValues>
                                <Value>ja_JP</Value>
                                <Value>ko_KR</Value>
                                <Value>zh_CN</Value>
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
                                <Value>it_IT</Value>
```

```
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>

                        <DefaultValue>en_US</DefaultValue>
                </ValidValues>
        </Property>
```

付録 C. MAS ドライバーのキーワード

MAS ODBC ドライバーは、特定のドライバー・キーワードを使用して制御されます。これらのキーワードには、自動的にデフォルト値が指定されます。デフォルト値は、使用しているコネクターによって変わります。この章の内容は次のとおりです。

- キーワード設定の変更方法
- 各キーワードの詳細

ドライバー・キーワードの設定

ドライバー・キーワードは、次のメソッドまたはメソッドの組み合わせを使用して設定できます。

- ODBC 接続ストリング (アプリケーション・プログラムによって渡される)
- ODBC.INI ファイル
- ODBC Datasource Administrator
- SQLConnect 関数
- SQLSetConnectOption 関数
- SQLSetStmtOption 関数

ODBC 接続ストリングの使用

ODBC 接続ストリングを使用して設定を指定するには、次の例に示されているように、プログラムでキーワードをコード化します。

```
rc = SQLDriver Connect (hdbc, NULL, (u_char far *)
   "NOAS=no;"
   "UID=ai00xyz;PWD=abcdef;"
   "PORT=1200;HOST=129.4.0.160;"
   "LGID=dan;"
   "DSN=Tuld;"
```

ODBC.INI ファイルの使用

次の例は、 ODBC.INI ファイルを使用してキーワード値を設定する方法を示しています。

```
Driver=C:\WINDOWS\SYSTEM\scodbc.dll
Description=Sample Shadow Direct Driver
APPL=ODBC
LINK=TCPIP
LGID=ita
UID=ai00xyz
PWD=abcdef
HOST=129.4.0.160
```

ODBC Datasource Administrator の使用

MAS ODBC Datasource Administrator を使用してキーワードを設定することもできます。このメソッドは、 MAS ODBC ドライバーをインストールして構成した後に使用できます。このメソッドを使用するには、以下の手順を実行します。

1. 「コントロール パネル」を開き、ODBC アイコンをダブルクリックします。
   次の画面が表示されます。

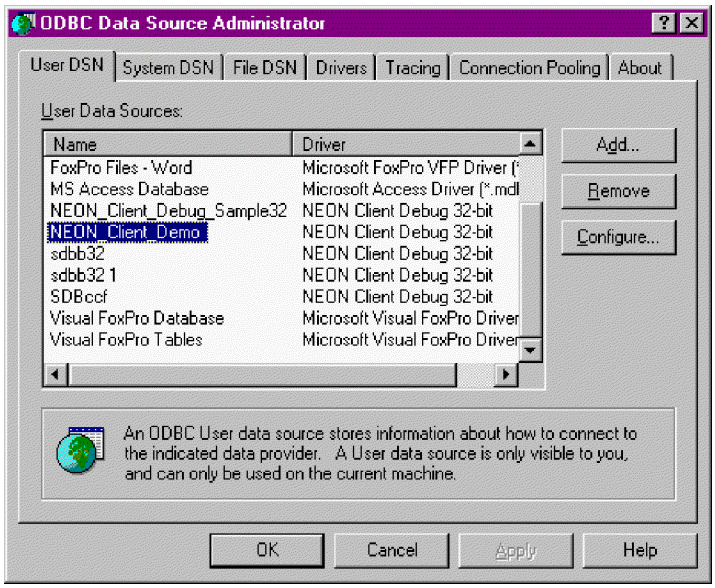


図 21. 「ODBC Data Source Administrator」画面

2. **「構成 (Configure)」**をクリックします。
   次の画面が表示されます。

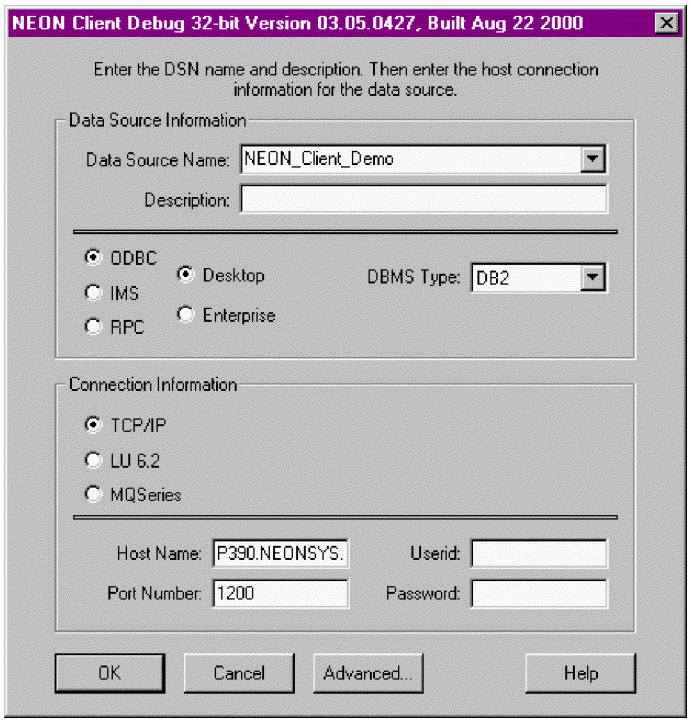


図 22. 「データ・ソース名 (Data source name)」画面

3. **「拡張」**をクリックします。

次の画面が表示されます。

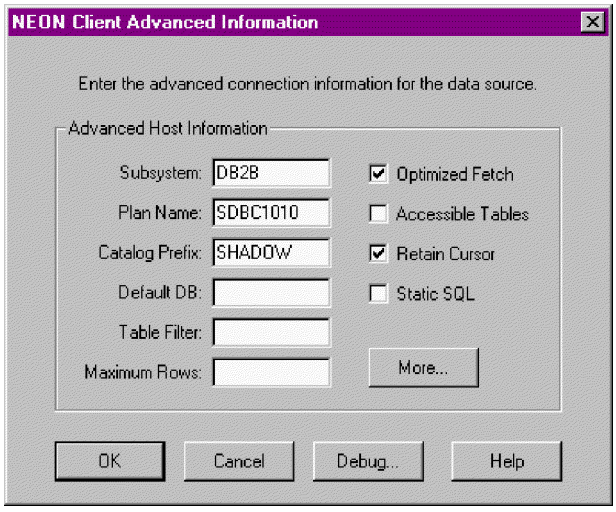


図 23. 「拡張情報 (Advanced Information)」画面

4. **「詳細 (More)」**をクリックします。
   次の画面が表示されます。

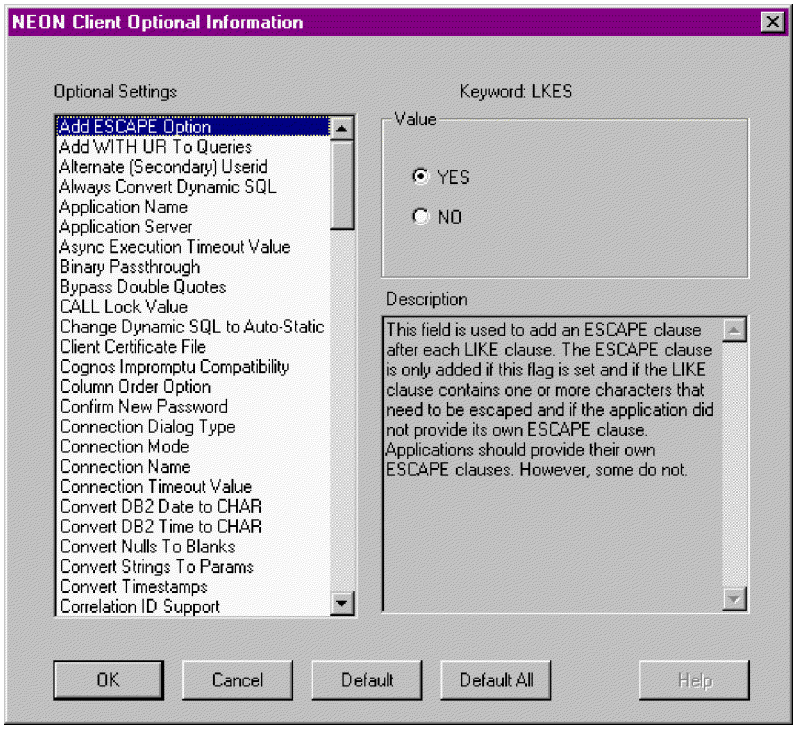


図 24. 「オプションの情報 (Optional Information)」画面

5. キーワードを選択します。この例では、LKES です。設定を「はい」から「いいえ」に変更します。

ドライバー・キーワードの説明

表 31 に、設定可能なドライバー・キーワードをリストし、それぞれの簡単な説明とデフォルト設定を示します。前述の全メソッドですべてのキーワードを設定できなければならないわけではありません (178 ページの表 32 を参照してください)。

表 31. ドライバー・キーワード

説明	キーワード	(フォーマット) 説明	デフォルト
Add ESCAPE オプション	LKES	このキーワードは、各 LIKE 文節の後に ESCAPE 文節を追加するために使用されます。 ESCAPE 文節が追加されるのは次の場合だけです。 • このキーワードは設定されています。 • LIKE 文節にエスケープする必要のある文字が 1 つ以上ある場合。 • アプリケーションが独自の ESCAPE 文節を提供しなかった場合。	YES
WITH UR を照会に追加 (Add WITH UR to Queries)	WIUR	このキーワードは、WITH UR 文節を照会に追加するかどうかを制御します。 WITH UR 文節は、いくつかの照会に必要な CPU 時間の量を減らします。コミットされていないデータを読み取り可能にすることによって、これらの照会の結果が変更される場合もあります。このキーワードを YES に設定すると、WITH UR 文節は照会に追加されます。このキーワードを NO に設定すると、WITH UR 文節は照会に追加されません。	NO
代替 (2 次) ユーザー ID (Alternate (Secondary) Userid)	ALUS	このキーワードは、クライアント・アプリケーションのホストの 2 次ユーザー ID を設定するために使用されます。このキーワードは、8 文字以下である必要があります。このキーワードをブランク以外の非ヌル値に設定すると、DSNALI OPEN 処理の完了後に SET CURRENT SQLID ステートメントが発行されます。	
常に動的 SQL を変換 (Always Convert Dynamic SQL)	ALCD	このキーワードは、動的 SQL を静的 SQL に変換できない場合に何を実行するかを制御します。このキーワードを YES に設定すると、動的 SQL を静的 SQL に変換できない場合に、アプリケーションにエラーが報告されます。このキーワードを NO に設定すると、動的 SQL は処理のためホストに送信されます。 **注:** 1 次 Dynamic-To-Static SQL キーワードが YES に設定されていなければ、このキーワードはテストもされません。動的 SQL と静的 SQL をともに使用する場合は、このキーワードを NO に設定する必要があります。	YES
アプリケーション名	APNA	このキーワードは、アプリケーション名です。アプリケーション名は、ログオン情報の一部としてホストに送信されます。アプリケーション名は、通常、計画内で SQL ステートメントをグループ化するために使用されます。アプリケーション名が設定されていない場合は、計画に関連するすべての SQL が、1 つの大きいグループの一部と見なされます。ある計画で使用する SQL を複数のサブグループに分割する必要がある場合 (静的 SQL への変換のため) は、アプリケーション名を設定する必要があります。	

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
アプリケーション・サーバー	SECN	このキーワードは、ホストで初期 Application Server 値を選択するために使用されます。これは、16 文字以下である必要があります。このキーワードをブランク以外の非ヌル値に設定すると、DSNALI OPEN 処理の完了後に、現在のアプリケーションが、指定されたアプリケーション・サーバーに接続されます。	
非同期実行タイムアウト値 (Async Execution Timeout Value)	ASTM	このキーワードは、非同期に実行されている ODBC 関数のデフォルトの待ち時間を制御します。このキーワードは秒単位で測定されます。これは、整数で指定する必要があります。	2
バイナリー・パススルー (Binary Passthrough)	BIPA	このキーワードは、ホスト・バイナリー・データを 16 進数に変換せずにクライアント・アプリケーションに戻すかどうかを制御します。このキーワードを YES に設定すると、ホスト・バイナリー・データは変更されずにクライアント・アプリケーションに渡されます。このキーワードを NO に設定すると、ホスト・バイナリー・データは 16 進数に変換されます (ODBC 仕様に従って)。	NO
二重引用符をバイパス (Bypass Double Quote)	BYDB	二重引用符をそのままにしておく場合は、このキーワードを YES に設定します。このキーワードは、特定のアプリケーション・バグを修正するために提供されています。	NO
CALL ロック値 (CALL Lock Value)	CALK	このキーワードは、ホスト上の CALL ステートメントに関連したロックのタイプを制御します。このキーワードを NONE に設定すると、ホスト・コードは、 CALL ステートメントがホスト・データベース・ロックを取得しないと想定します。使用可能なその他の値は、SHARE、UPDATE、および EXCLUSIVE です。	EXCLUSIVE
動的 SQL から静的 SQL への変更 (Change Dynamic SQL to Static)	AUST	このキーワードは、 Shadow Direct が動的 SQL の Auto-Static SQL への変換を試行するかどうかを制御します。動的 SQL を Auto-Static SQL に変換できるのは、このキーワードを YES に設定し、ホストが Auto-Static SQL をサポートしている場合だけです。 **注:** このキーワードを YES に設定すると、ルックアサイド・バッファーですべての動的 SQL ステートメントがルックアップされ、可能な場合は静的 SQL に変換されます。このキーワードを NO に設定すると、標準の動的 SQL 処理が実行されます。	YES
クライアント証明書ファイル (Client Certificate File)	CLCT	説明はありません。	
Cognos Impromptu の互換性 (Cognos Impromptu Compatibility)	CGFX	Cognos Impromptu で特定の問題を解決するには、このキーワードを YES に設定します。このキーワードは、その他の目的では設定しないでください。	YES
列の順序オプション (Column Order Option)	CLOR	このキーワードは、ODBC カタログ関数によって列名がどのように戻されるかを示します。関数によっては、順序が明示的に指定されます。また、順序を指定しない関数もあります。このキーワードは、列名の順序を指定しない ODBC カタログ関数 (SQLColumns) によって使用されます。	NUMBER

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
新規パスワードの確認	CMNP	このキーワードを設定すると、ODBC PWD キーワードで新規パスワードが設定されている場合に、ユーザーは新規パスワード・ストリングを確認するよう求められます。	NO
接続ダイアログ・タイプ (Connection Dialog Type)	CNDG	このキーワードは、どのタイプの接続ダイアログを表示するかを示します。このキーワードは、ODBC.INI ファイルまたは接続ストリングを使用して設定できます。動的ダイアログが指定されている場合は、必要な情報量に応じて、単純なダイアログまたは詳細なダイアログが表示されます。	DYNAMIC
接続モード	CNMD	このキーワードは、ODBC アプリケーションによって使用される接続モードを制御します。接続モードは、各物理接続 (セッションまたは会話) がどれくらい持続するかと、SQL 操作がブロック化されているかどうかを判別します。デフォルトでは、永続接続が使用され、各 SQL 操作が独立してサーバーに送信されます。 BLOCK モードでは、セッションは永続です。ただし、SQL 操作はブロック化され、まとめて送信されます。 TRANSACTION モードでは、各 SQL 操作が独立して送信されますが、各 Logical Unit Of Work (LUOW) の終わりにセッションが終了します。 TRANSBLOCK モードでは、各 LUOW の終わりにセッションが終了し、SQL 操作はブロック化され、まとめて送信されます。 MESSAGE モードでは、SQL 操作はメッセージを使用してブロック化され、まとめて送信されます。 MESSAGE モードでは、セッションは維持されません。	PERMANENT
接続名	CNNA	このキーワードは、接続名を指定するために使用されます。接続名の使用は、アプリケーション固有です。名前の長さは最大 8 バイトですが、アプリケーションはそのバイトをすべて使用する場合もあれば、使用しない場合もあります。接続名は、右側をブランクで埋め込まれます。	
接続タイムアウト値 (Connection Timeout Value)	CNTM	このキーワードは、ODBC クライアントがホスト・サーバーへの接続の完了を待つ時間を秒単位で制御します。このキーワードをゼロに設定した場合は、システム・デフォルト値が使用されます。それ以外の場合、指定された値が使用されます。この値に負の数を指定することはできませんが、ゼロは指定できます。この値は、UDP メッセージングには適用されません。 **注:** このキーワードを使用すると、多くの環境で予測不能な結果が生じることがあります。このオプションは TCP/IP リンク・タイプと MQ リンク・タイプでのみ使用できますが、永続モードとメッセージング・モードの両方に適用されます。	0
DB2 日付を CHAR に変換 (Convert DB2 Date to CHAR)	DTCH	このキーワードを設定すると、DB2 日付キーワードが SQL_CHAR に変換されます。	NO

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
DB2 時刻を CHAR に変換 (Convert DB2 Time to CHAR)	TSCH	このキーワードを設定すると、DB2 タイム・スタンプと時刻キーワードが SQL_CHAR に変換されます。キーワード AF (MS Access Compatibility) を YES に設定した場合、このキーワードを設定する必要はありません。キーワード AF には、キーワード TSCH の機能は含まれていません。	NO
ヌルをブランクに変換 (Convert Nulls to Blanks)	CVNL	このキーワードは、文字ストリング・データのヌル (ゼロ・バイト) をサーバーからクライアントへ戻すかどうか、またはブランクに変換するかどうかを制御します。このキーワードは、固定長文字データと可変長文字データの両方に影響します。このキーワードを YES に設定すると、ヌルはブランクに変換されます。このキーワードを NO に設定すると、ヌルは変換されません。	NO
ストリングをパラメーターに変換 (Convert Strings to Params)	LGPA	このキーワードは、長ストリングを LONG VARCHAR パラメーターに変換するかどうかを制御します。ホスト・データベースで処理できないほど長いリテラルを生成するアプリケーションがあるため、この変換が必要となります。このキーワードを NO に設定すると、各 SQL ストリングがスキャンされて長ストリングが検索されることはなくなります。このキーワードを YES に設定すると、すべての長ストリングがパラメーター・マーカーに変換されます。	NO
タイム・スタンプの変換 (Convert Timestamps)	CVTS	このキーワードは、タイム・スタンプ値をその他のタイプに変換するかどうかを制御します。このキーワードを YES に設定すると、日付として表示されるタイム・スタンプは日付に変換され、時刻として表示されるタイム・スタンプが時刻に変換されます。これらの変換は、MS Access and Crystal Reports などいくつかの製品のバグを回避するために必要です。このキーワードを NO に設定すると、タイム・スタンプは変更されません。	NO
相関 ID サポート (Correlation ID Support)	COID	このキーワードは、SQLGetInfo が相関 ID に関する情報を戻すかどうかを制御します。このキーワードを YES に設定すると、SQLGetInfo は相関 ID に関する情報を戻します。このキーワードを NO に設定すると、相関 ID に関する情報は戻されません。一部の ODBC ツールを機能させるために、このキーワードを NO に設定しなければならない場合もあります。	YES
Count() 修正タイプ (Count() Fix Type)	COFX	このキーワードは、COUNT (列名) SQL 関数の修正方法を制御します。 Microsoft Access は、COUNT (列名) 関数を 2 つの方法で使用します。これらはどちらも間違っています。どちらの場合も、COUNT (列名) は実際には COUNT (DISTINCT 列名) を意味します。別の場合には、COUNT(*) を意味します。このキーワードを DISTINCT に設定すると、DISTINCT キーワードが挿入されます。このキーワードを ASTERISK に設定すると、列名は「*」に置換されます。このキーワードを NONE に設定すると、変更は実行されません。 **注:** Access 互換モードがアクティブでなければ、COUNT (列名) 関数は変更されません。	DISTINCT

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
テーブル・インデックスの自動作成 (Create Table Index Automatically)	CRIN	このキーワードは、基本キーまたは固有の制約でテーブルを作成したときにインデックスを自動的に作成するかどうかを制御します。このインデックスは固有です。このオプションを使用不可にするには、値を NO に設定します。	YES
Current Degree	SEDG	このキーワードは、ホストで初期 Current Degree 値を設定するために使用されます。このキーワードで使用できる値は ANY または 1 だけです。その他の値は現時点ではサポートされていません。このキーワードをブランク以外の非ヌル値に設定すると、DSNALI OPEN 処理の完了後に SET CURRENT DEGREE ステートメントが発行されます。	
Current Packageset	SEPK	このキーワードは、ホストで初期 Current Packageset 値を設定するために使用されます。この値は、18 文字以下である必要があります。このキーワードをブランク以外の非ヌル値に設定すると、DSNALI OPEN 処理の完了後に SET CURRENT PACKAGESET ステートメントが発行されます。 SET ステートメントは、CURRENT PACKAGESET 特殊レジスターに指定値を割り当てます。	
Current Rules	SERL	このキーワードは、ホストで初期 Current Rules 値を設定するために使用されます。このキーワードで使用できる値は DB2 または STD だけです。その他の値は現時点ではサポートされていません。このキーワードをブランク以外の非ヌル値に設定すると、DSNALI OPEN 処理の完了後に SET CURRENT RULES ステートメントが発行されます。	
カーソルのコミット/ロールバックの振る舞い (Cursor Commit/Rollback Behavior)	CRBH	このキーワードは、カーソルのコミット/ロールバックの振る舞いに関する SQLGetInfo 要求に対して、どの値を戻すかを制御します。 DELETE は、カーソルをクローズし、作成されたステートメントを削除します。カーソルを再び使用するには、アプリケーションでステートメントを再作成して再実行する必要があります。 CLOSE はカーソルを閉じるだけです。作成されたステートメントに対して、アプリケーションは SQLPrepare を再び呼び出すことなく、そのステートメントで SQLExecute を呼び出します。	DELETE
カスタマー秘密鍵 (Customer Secret Key)	SKEY	説明はありません。	
日付形式	DTFM	このキーワードは、ODBC 日付を文字ストリングに変換する方法を指定するために使用されます。このキーワードを ODBC に設定した場合は、標準の ODBC/ISO 形式 yyyy-mm-dd が使用されます。このキーワードを UK に設定した場合は、dd-mm-yyyy 形式が使用されます。このキーワードを EUR に設定した場合は、dd.mm.yyyy 形式が使用されます。	ODBC

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
DB2 バージョン・ストリング (DB2 Version String)	D2VR	このキーワードは、ホストによって戻された DB2 バージョン・ストリング・データを上書きできます。このストリングは、x.y.z という形式にする必要があります。x、y、z はすべて数字です。 x は DB2 バージョン・バイトです。 y は DB2 修正レベルです。 z は DB2 リリース・レベルです。バージョン・バイトと修正レベルは両方とも設定する必要があります。リリース・レベルが設定されていない場合は、0 が想定されます。例えば、2.3 は、DB2 バージョン 2、修正レベル 3 を表し、4.1.1 は、DB2 バージョン 4、修正レベル 1、リリース・レベル 1 を表します。	
DBCS モード	DBMD	このキーワードは、DBCS データの処理方法を指定するために使用されます。 BINARY を指定すると、すべての DBCS データがバイナリー・データとして扱われます。 **注:** DBCS データはホストの形式からクライアントの形式に変換されますが、BINARY モードでは、ホストでバイナリー列に保管されていなくても、データはバイナリーとして記述されます。 CHARACTER を指定すると、GRAPHIC 列に保管されている場合でも、すべての DBCS データが文字データとして扱われます。 DEFAULT および GRAPHIC は、現時点ではサポートされていません。	DEFAULT
DBCS ブランクの除去 (DBCS Remove Blanks)	DBQO	このキーワードは、2 バイト文字 (中国語、日本語、韓国語など) で引用符付きストリング内からブランクを除去するかどうかを制御します。このキーワードを YES に設定すると、ホストに送信される SQL の引用符付きストリング内にあるブランクが除去されます。このキーワードを NO に設定すると、ホストに送信される SQL の引用符付きストリング内にあるブランクは変更されません。	YES
デフォルト列名 (Default Column Names)	DFCL	このキーワードは、DBMS が列の名前を戻さない場合にドライバーがデフォルト列名 (形式は COLnnn。nnn は列番号) を割り当てるかどうかを制御します。	NO
小数点はコンマ (Decimal Point is Comma)	DCCM	このキーワードは、小数点をコンマにするかどうかを制御します。このキーワードを YES に設定すると、コンマが小数点と見なされます。このキーワードを NO に設定すると、ピリオドが小数点と見なされます。このキーワードを YES に設定できるのは、ホストがコンマを小数点として使用している場合だけです。このキーワードを指定すると、データが Auto-Static SQL 変換機能に渡される前に、コンマがピリオドに変換されます。	NO
CALL の準備の据え置き (Defer Prepare for CALLS)	DFPR	このキーワードは、CALL ステートメントの準備を据え置くかどうかを制御します。このキーワードを YES に設定すると、CALL ステートメントの準備は常に据え置かれます。このキーワードを NO に設定すると、1 つ以上のパラメーター・マーカーを持つ CALL ステートメントの準備だけが据え置かれます。つまり、パラメーター・マーカーを持つ CALL ステートメントの準備は常に据え置かれます。このキーワードを YES に設定した場合は、パラメーター・マーカーを持たない CALL ステートメントの準備も据え置かれます。	NO

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
非同期オプションを使用不可にする (Disable Async Option)	NOAS	このキーワードは、非同期実行を許可するかどうかを示すために使用されます。このオプションを YES に設定すると、すべての非同期処理が妨げられます。	YES
CTL3D を使用不可にする (Disable CTL3D)	NO3D	このキーワードは、CTL3D 処理が発生するかどうかを示すために使用されます。このオプションを YES に設定すると、接続ダイアログで CTL3D DLL の使用が妨げられます。 CTL3D の古いコピーがインストールされている場合、アプリケーションによっては、このオプションを YES に設定する必要があります。	NO
最適化の準備/オープンを使用不可にする (Disable Prepare/Open Optimization)	DIPO	このキーワードは、最適化の準備/オープンを使用不可にし、その結果としてネットワークの往復を追加します (SQLExecute 用)。 VB の上部にある RDO は、各 select ステートメントごとに 2 つの SQLPrepares と 1 つの SQLExecute を発行します。これにより、照会ごとに 2 つの結果セットが戻されます。	NO
SQLCancel トレースを使用不可にする (Disable SQLCancel Tracing)	DIPO	このキーワードは、最適化の準備/オープンを使用不可にし、その結果としてネットワークの往復を追加します (SQLExecute 用)。 VB の上部にある RDO は、各 select ステートメントごとに 2 つの SQLPrepares と 1 つの SQLExecute を発行します。これにより、照会ごとに 2 つの結果セットが戻されます。	NO
すべてのプロンプトを使用不可にする (Disable All Prompts)	NOPM	このキーワードを YES に設定すると、対話式プロンプトまたは情報メッセージ・ボックスがすべて使用不可になります。この機能は、NT サービス、UNIX デーモン・プロセス、または割り込みができないサーバー・タイプのアプリケーションからドライバーが呼び出されている場合に必要です。	NO
待機カーソルの表示 (Display Wait Cursor)	WACU	このキーワードは、カーソルを (砂時計で) 更新するかどうかを示します。このオプションを NO に設定すると、パフォーマンスが大幅に向上する場合があります。その場合、カーソルは更新されず、砂時計も表示されません。	NO
ODBC.INI からパスワードを取得しない (Don't Get Passwords from ODBC.INI)	NOPW	このキーワードは、ODBC.INI ファイルからパスワードを取得するかどうかを制御します。このキーワードを YES に設定した場合は、接続ストリングまたは接続ダイアログでパスワードを提供する必要があります。 **注:** このキーワードを YES に設定すると、ODBC.INI 内のパスワード値はすべて無視されます。このキーワードを NO に設定すると、ODBC.INI ファイルからパスワードが取得される場合があります。	NO
ODBC.INI からユーザー ID を取得しない (Don't Get Userids from ODBC.INI)	NOUS	このキーワードは、ODBC.INI ファイルからユーザー ID を取得するかどうかを制御します。このキーワードを YES に設定した場合は、接続ストリングまたは接続ダイアログでユーザー ID を提供する必要があります。 **注:** このキーワードを YES に設定すると、ODBC.INI 内のユーザー ID 値はすべて無視されます。このキーワードを NO に設定すると、ODBC.INI ファイルからユーザー ID が取得される場合があります。	NO

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
DTS 計画ファイル (DTS Plan File)	DSFL	このキーワードは、動的 SQL から静的 SQL への変換に使用されるローカル計画ファイルの名前を指定するために使用されます。計画ファイル名は .pln で終了する必要があります。これは、必要に応じて絶対パス名にすることもできます。このキーワードを使用するのは、ローカル計画ファイルにデフォルト名がない場合や、ローカル計画ファイルがアプリケーションの作業ディレクトリーにない場合だけにしてください。ローカル計画ファイルのデフォルト名は、常に planname.pln です。ここで、planname は、アプリケーションによって使用される DB2 計画の名前です。	
重複ソケット記述子 (Duplicate Socket Descriptor)	DPSO	このキーワードは、接続ソケット用の新規または未使用のファイル記述子を戻すために使用される、負の数以外の整数です。新規ソケット番号は、この数と同じかそれよりも大きい記述子番号になります。古い記述子は置換後に閉じられます。このキーワードは UNIX プラットフォームでのみ使用可能であり、C ランタイム・アプリケーションがスタジオ・ライブラリーで 255 の最大オープン・ファイル・ハンドル制限を回避できるように設計されています。このキーワードでは、例えば、256 などの値を設定するとよいでしょう。	0
マルチタスキングを使用可能にする (Enable Multitasking)	MT	このキーワードは、Shadow Direct のマルチタスキング機能を制御します。この機能を使用すると、ODBC 呼び出しの保留中に画面がロックされずに済みます。ただし、この機能はよく注意して使用してください。この機能をアクティブにすると、正しく動作しないアプリケーションが多数あります。	NO
拡張カーソル・プール (Extended Cursor Pool)	EXCU	このキーワードは、拡張カーソル・プールを使用するかどうかを制御します。このキーワードを YES に設定すると、拡張カーソル・プールが使用されます。拡張カーソル・プールは、標準カーソル・プールよりはるかに大規模です。 **注:** 拡張カーソル・プールは、Auto-Static SQL では使用できません。このキーワードを NO に設定すると、標準カーソル・プールが使用されます。	NO
高速ログオン (Fast Logon)	FALG	このキーワードは、高速ログオンを使用するかどうかを制御します。このキーワードを YES に設定すると、最小のネットワーク I/O でログオン処理が実行されます。これを使用できるのは、ホスト・サーバーが高速ログオンをサポートしている場合だけです。高速ログオンが要求された場合にホスト・サーバーが高速ログオンをサポートしていないと、障害が発生します。このオプションを使用する前に、高速ログオンを処理するサーバーの機能を確認する必要があります。このキーワードを NO に設定すると、標準ログオン処理が使用されます。	NO

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
INSERT ステートメントの修正 (Fix INSERT Statements)	FXIS	このキーワードは、各 INSERT ステートメントの VALUES 文節をスキャンして日付、時刻、タイム・スタンプを検索するかどうかと、引用符を追加するかどうかを制御します。このキーワードを YES に設定すると、必要に応じて日付、時刻、タイム・スタンプに引用符が追加されます。このキーワードを NO に設定すると、INSERT ステートメントは変更されません。	NO
DB2 外部結合の修正 (Fix DB2 Outer Joins)	OJFX	このキーワードを設定すると、DB2 外部結合の ON 文節を囲む括弧が除去されます。この構文は有効な ANSI SQL ですが、DB2 は現行リリース (バージョン 5) ではこれを処理できません。この種の SQL は、Brio などの ODBC ツールによって生成されます。	NO
Oracle 8.0.5 バグの修正 (Fix Oracle 8.0.5 Bug)	OJFX	このフィールドは、oracle8.0.5 バグを修正するために使用されます。	NO
ストリングの長さの修正 (Fix String Length)	STFX	このキーワードは、SQL_CHAR キーワードにブランクだけが含まれている場合にストリングの長さを誤ってゼロに設定する、Microsoft Access など一部の製品で発生する特定のバグを修正します。キーワード AF (MS Access Compatibility) を YES に設定した場合、このキーワードを設定する必要はありません。キーワード AF には、キーワード STFX の機能が含まれています。	NO
浮動小数点から文字への変換ディジット (Floating to Character Digits)	MXDG	このキーワードは、浮動小数点から文字への変換で生成されるディジットの最大数を制御します。このキーワードを指定するのは、デフォルト値を許容できない場合だけにしてください。	6
外字拡張サポート (GAIJI Extension Support)	GAJI	このキーワードは、コード・ページで外字拡張サポートを使用可能にします。	YES
外字拡張テーブル名 (GAIJI Extension Table Name)	GATB	このキーワードは、外字拡張テーブル名を設定します。デフォルト名はありません。このキーワードをブランクにすると、サーバー・サイドのデフォルト・テーブル名が使用されます。このキーワードの最大キーワード・サイズは 4 文字です。	
ドイツ語 NLS サポート (German NLS support)	NLGR	ドイツ語の言語環境で SQL の処理に関する特定の問題を解決するには、このキーワードを YES に設定します。このキーワードは、その他の目的では設定しないでください。	NO
ホスト・デバッグ値 (Host Debugging Values)	HODB	このキーワードは、ホスト・プログラム (通常はストアード・プロシージャー) のデバッグを制御します。このキーワードは、ホスト・プログラムが記述された言語のタイプまたは NONE に設定します。ホスト・プログラム言語は、プログラムが正しいデバッグ・オプションで起動できるように正しく指定する必要があります。サポートされる値は、COBOL、PLI、C、および C++ です。	NONE
ホスト・ユーザー PARM (Host User Parm)	USERPARM	このキーワードは、ログオン情報の一部としてホストに送信されます。ホストはこのキーワードを使用して、ホスト・セキュリティー・システムまたはホスト・データベースへのログオンを完了します。	

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
ID 引用符オプション (Identifier Quote Option)	IDQO	このキーワードは、SQL_IDENTIFER_QUOTE_CHAR を指定した SQLGetInfo を使用するアプリケーションにどの ID 引用符文字を戻すかを示します。このキーワードは、特定のアプリケーション・バグを修正するために必要です。	BLANK
高バインド列エラーを無視 (Ignore High Bound Column Errors)	IGHI	このキーワードは、SQLFetch が高バインド列エラーを無視するかどうかを制御します。このキーワードを YES に設定すると、SQLFetch は列バインド・エラーを無視します。このキーワードを NO に設定すると、列バインド・エラーは通常通り処理されます。一部の ODBC ツール (Microsoft Excel など) を機能させるために、このキーワードを YES に設定しなければならない場合もあります。	NO
下線文字を無視 (Ignore Underscore Characters)	IGUN	このキーワードは、下線文字をワイルドカードと通常の文字のいずれと見なすかを制御します。このキーワードを YES に設定すると、下線 (「_」) は標準バイトとして処理されます。このキーワードを NO に設定すると、下線は、他のすべての単一バイトに一致するワイルドカード文字として処理されます。このキーワードは、テーブルまたはプロシージャーの名前に実際の下線があり、その下線がワイルドカードと誤解される問題を解決するために使用されます。	NO
リテラル引用符の保持 (Keep Literal Quotes)	KEQU	このキーワードは、ストリング・リテラルを囲む組み込み引用符を保持するために使用されます。このキーワードを YES に設定すると、ストリング・リテラルを囲む組み込み引用符が保持されます。 **注:** これは、MDI ストアード・プロシージャーなど、ストアード・プロシージャーに渡されるストリング・リテラルだけに適用されます。このキーワードを NO に設定すると、ストリング・リテラルから引用符が除去されます。	NO
言語 ID (Language ID)	LGID	このキーワードは使用する言語を指定するために使用されます。指定できる値は、アラビア語 (ARB)、中国語 (簡体字) (CHS)、中国語 (繁体字) (CHT)、デンマーク語 (DAN)、デフォルト (DFT)、ドイツ語 (DEU)、英語 (互換性) (ENC)、英語 (英国) (ENG)、英語 (米国) (ENU)、スペイン語 (現代) (ESN)、カスティリャ・スペイン語 (ESP)、フィンランド語 (FIN)、フランス語 (FRA)、カナダ・フランス語 (FRC)、カナダ・アイスランド語 (ISL)、イタリア語 (ITA)、日本語 (JPL)、日本語英数小文字 (JPX)、韓国語 (KOR)、Micro Decisionware (MDI)、オランダ語 (NLD)、ノルウェー語 (NOR)、PeopleSoft English (PPS)、ポルトガル語 (PTG)、スウェーデン語 (SVE)、およびトルコ語 (TUR) です。	ENU

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
長いデータの修正 (Long Data Fix)	LGFX	このキーワードは、LONG VARCHAR キーワードの長さと精度を表すために、実際の長さと精度ではなく大きい数値を戻すかどうかを制御します。この修正は、ODBCDirect GetChunk 処理における特定の問題を解決するために必要です。このキーワードを YES に設定すると、LONG VARCHAR キーワードの長さと精度を表す大きい数値が戻されます。このキーワードを NO に設定すると、LONG VARCHAR キーワードの長さと精度を表す実際の数値が戻されます。	NO
Lotus Approach の互換性 (Lotus Approach Compatibility)	LAFX	Lotus Approach で特定の問題を解決するには、このキーワードを YES に設定します。このキーワードは、その他の目的では設定しないでください。	NO
LZ 圧縮 (LZ Compression)	LZCM	このキーワードを YES に設定し、サーバーがこの機能をサポートしている場合は、クライアントとサーバーの間のすべてのデータ・フローの圧縮に Lempel-Ziv 圧縮のバリアントが使用されます。	YES
最大バッファー・サイズ (Maximum Buffer Size)	MXBU	このキーワードは、すべてのデータ交換に対してドライバー側の最大通信バッファー・サイズを設定します。このキーワードのデフォルトは 0 です。これは、現在、最大バッファー・サイズを 100,000 バイト (16 ビットのウィンドウでは 32K バイト) に設定しています。 **注:** この値は、サーバー・サイドの限界の折衝のために使用されます。実際のランタイム・バッファー・サイズは、これより小さくなる場合がありますが、これより大きくなることはありません。	0
メッセージ・タイプ	MGTY	このキーワードは、メッセージ指向の接続に使用されるメッセージ・タイプを制御します。これが使用されるのは、メッセージ接続モードが使用されている場合だけです。デフォルト値は NETWORK です。これは、各メッセージに対して TCP/IP または LU 6.2 セッションあるいは会話を作成します。リンク・タイプを TCP/IP に設定すると、ユーザーは UDP メッセージ・タイプをメッセージング・プロトコルとして選択することもできます。メッセージングに UDP データグラムを使用すると、通常はネットワークのオーバーヘッドが低くなりますが、WAN を介して接続する場合、UDP 通信は信頼性が低く、お勧めできません。HTTP および HTTPS メソッドは、 HTTP プロトコル・ストリーム内の通信セッションをトンネル化し、ファイアウォールと HTTP プロキシーを介してセッションを確立できるようにします。 HTTPS は、SSL をインプリメントした Netscape セキュア HTTP プロトコルです。これは、2 つのエンドポイント間で安全なチャネルが必要な場合に使用できます。	NETWORK

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
MS アクセス互換性 (MS Access Compatibility)	AF	このキーワードを YES に設定すると、すべてのホストの 10 進データ値が、小数点の右側に後続ゼロを付けずに、ODBC クライアント・アプリケーションに戻されます。このキーワードは、Microsoft Access が DB2 10 進データを処理できるよう、YES に設定する必要があります。このキーワードは、Visual Basic などの Microsoft Jet Database Engine を使用する他のあらゆる製品に必要です。	NO
MTS セキュリティー SID タイプ (MTS Security SID Type)	MSMT	このキーワードは、MS Transaction Server ベースのネットワーク認証/シングル・サインオン・モデルを使用する際のセキュリティー SID オプションです。	DIRECT CALLER SID
MultiNet TCP/IP 互換性 (MultiNet TCP/IP Compatibility)	MUNT	このキーワードは、ホスト・アクセスに対して Multi-Net TCP/IP 製品を使用するアプリケーションをサポートしています。	NO
ネットワーク認証 (Network Authentication)	SECU	DOMAIN ベースのアプリケーションを選択すると、ドライバーは、現行プロセスに関連したユーザー ID がドメイン・ベースのシステムによって認証されているかどうかを検証します。 Win32 プラットフォームでは、ユーザーがまず NT ドメインにログオンする必要があります。 UNIX プラットフォームでは、ローカル・マシンが NIS ドメインのメンバーである必要があります。また、ユーザーの認証に使用されるパスワード・データベースが NIS によってマップされている必要があります。 MTS ベースの認証モデルを使用すると、MTS サーバー COM コンポーネントが、オブジェクトを起動するクライアント・アプリケーションのアカウント名を渡すことができます。 MyNet 設定は、CKS MyNet シングル・サインオン製品サポートで使用されます。 SNA 設定は、MS SNA サーバーのシングル・サインオン機能を処理します。 ODBC クライアントは、MS/SNA CPI-C である必要があります。	NONE
ヌルなし (No Nulls)	NONL	このキーワードは、長さゼロのストリングを戻すかどうかを制御します。このキーワードを YES に設定すると、長さゼロのストリングが 1 つのブランクに置換されます。このキーワードを NO に設定すると、長さゼロのストリングは変更されません。	NO
非ブロッキング・ネットワーク I/O (Non-blocking Network I/O)	NBIO	このキーワードは、TCP/IP ネットワーク I/O 操作をブロックするかどうかを制御します。このキーワードを YES に設定すると、非ブロッキング・ネットワーク I/O 操作が使用され、I/O 操作の完了を待つために Windows メッセージ・キューが使用されます。このキーワードを NO に設定すると、ブロッキング・ネットワーク I/O 操作が使用されます。	NO
ODBC クライアント・コード・ページ (ODBC Client Code Page)	ODPG	このキーワードは、ODBC クライアント・コード・ページを指定するために使用されます。デフォルトは、UNIX 用の UNIX コード・ページのセットおよび Windows 95/NT 用の Windows Latin 1 (ANSI) です。 Windows Latin 1 は、ISO 8859 としても知られています。 LATIN1 を指定すると、あらゆる環境で Windows Latin 1 コード・ページの使用が強制されます。 UNIX を指定すると、あらゆる環境で UNIX コード・ページの使用が強制されます。	DEFAULT

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
片方向メッセージ (One-Way Messages)	ONWY	このキーワードは、メッセージへの応答を ODBC クライアントに送信するかどうかを制御します。このキーワードを YES に設定すると、片方向メッセージ処理が使用されます。これは、ODBC クライアント・アプリケーションに応答が戻されないことを意味します。このキーワードを NO に設定すると、メッセージに対して正規応答が戻されます。 **注:** このキーワードは、メッセージでのみ使用できます。	NO
操作タイムアウト値 (Operation Timeout Value)	OPTM	このキーワードは、接続の確立後に、すべてのクライアント・ネットワーク操作 (照会の作成、実行、結果セットの検索など) のタイムアウト値 (秒単位) を制御します。この値に負の数を指定することはできませんが、ゼロは指定できます。この値は、UDP メッセージングには適用されません。 **注:** このキーワードを使用すると、多くの環境で予測不能な結果が生じることがあります。このオプションは TCP/IP リンク・タイプと MQ リンク・タイプでのみ使用できますが、永続モードとメッセージング・モードの両方に適用されます。	0
最適な行カウント (Optimal Row Count)	OPRW	このキーワードは、行の要求が作成されるたびにホストから戻される行の数を制限します。行の要求は 1 つの照会に対していくつでも作成できるので、この値は、照会によって戻される行の合計数には影響を及ぼしません。この値を使用すると、ホストから行を戻すために使用されるバッファーのサイズを制御できます。	0
パラメーター処理 (Parameter Handling)	PAHN	このキーワードは、パラメーターが ODBC アプリケーションによって渡されたときにパラメーターがどう処理されるかを制御します。一部のアプリケーションから渡された値を修正するために特別な処理が必要になる場合もあります。例えば、Crystal Reports は、パラメーターを渡す際に、現在では使用されていない値を使用するため、それらを修正する必要があります。 Crystal Reports では、このキーワードを INPUT に設定する必要があります。一部のアプリケーションのパラメーター長のエラーを修正するには、このキーワードを LENGTH に設定する必要があります。一部の ADO 関連のエラーを修正するには、この値を ADO に設定する必要があります。このキーワードを NONE に設定すると、アプリケーションによって渡されるパラメーター値は変更されません。	NONE
PowerBuilder 互換性 (PowerBuilder Compatibility)	PBFU	PowerBuilder で特定の問題を解決するには、このキーワードを YES に設定します。このキーワードは、その他の目的では設定しないでください。	NO

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
プロシージャー名フィルター (Procedure Name Filter)	PRNF	このキーワードは、製品によって戻されたプロシージャー名をフィルターに掛けるために使用されます。このキーワードが使用されるのは、これが非ブランク値に設定され、ホスト・アプリケーションがプロシージャー名フィルター・ストリングを提供していない場合だけです。アプリケーションまたはこのキーワードからのプロシージャー名フィルター・ストリングは、プロシージャー・カタログ照会 (SQLProcedures および SQLProcedureColumns) によって戻される情報を制限します。提供されたプロシージャー名のパターンに一致する行だけが戻されます。このキーワードを単一の % 記号に設定すると、すべてのプロシージャーが戻されます。このキーワードのデフォルト値はありません。	
プロシージャー所有者フィルター (Procedure Owner Filter)	PROF	このキーワードは、製品によって戻されたプロシージャー所有者をフィルターに掛けるために使用されます。このキーワードが使用されるのは、これが非ブランク値に設定され、ホスト・アプリケーションがプロシージャー所有者フィルター・ストリングを提供していない場合だけです。アプリケーションまたはこのキーワードからのプロシージャー所有者フィルター・ストリングは、プロシージャー・カタログ照会 (SQLProcedures および SQLProcedureColumns) によって戻される情報を制限します。提供されたプロシージャー所有者のパターンに一致する所有者 ID のある行だけが戻されます。このキーワードを単一の % 記号に設定すると、すべてのプロシージャーが戻されます。デフォルトでは、このキーワードは現在のユーザー ID または代替ユーザー ID に設定されます。	
プロシージャー所有者処理 (Procedure Owner Handling)	PROW	このキーワードは、プロシージャー所有者値が ODBC アプリケーションにどのように戻されるかを制御します。このキーワードを NULL に設定すると、実際の所有者値がプロシージャー名の接頭部として使用され、所有者キーワードがヌル値として戻されます。このキーワードを EMPTY に設定すると、実際の所有者値がプロシージャー名の接頭部として使用され、所有者キーワードが空ストリングとして戻されます。このキーワードを NONE に設定すると、プロシージャー所有者値は変更されません。	NONE
プロセスのエスケープ (Process Escapes)	PRES	このキーワードは、エスケープ文節処理をオフにするために使用されます。 PRES=NO は、エスケープ文節処理をオフにします。これにより、CPU オーバーヘッドが減少し、パフォーマンスが向上する場合があります。このキーワードはオプションです。指定しない場合は、デフォルトで PRES=YES になります。	YES

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
プロキシー・サーバー情報 (Proxy Server Information)	PXSR	HTTP トンネリングがサーバーのメッセージングに使用され、HTTP ストリームをまずプロキシー・サーバーに接続する必要がある場合は、プロキシー・サーバーのホスト名 (または IP アドレス) とポート番号を特定するためにこのキーワードを設定する必要があります。このキーワードで使用できる形式は MY_PROXY:80 または 10.22.33.44:1200 です。値の最初の部分はプロキシー・サーバーのホスト名または IP アドレスで、 2 番目の部分はプロキシー・サーバーが listen するポート番号です。 **注:** プロキシー・サーバーのセットアップによっては、ユーザー認証が必要です。	
プロキシー・サーバー・ユーザー ID (Proxy Server Userid)	PXUS	HTTP トンネリング・メッセージング・モードが有効で、HTTP プロキシー・サーバーにユーザー認証が必要な場合は、このキーワードを使用すると、ユーザーがプロキシー・ユーザー ID (または名前) ストリングを指定できます。現時点でサポートされているメソッドは、Basic HTTP Access 認証だけです。	
プロキシー・サーバー・パスワード (Proxy Server Password)	PXPW	HTTP トンネリング・メッセージング・モードが有効で、HTTP プロキシー・サーバーにユーザー認証が必要な場合は、このキーワードを使用すると、ユーザーがプロキシー・ユーザー・パスワード・ストリングを指定できます。現時点でサポートされているメソッドは、Basic HTTP Access 認証だけです。	
修飾子名分離文字 (Qualifier Name Separator)	QUNA	SQL_QUALIFIER_NAME_SEPARATOR を取得するために呼び出されたときに SQLGetInfo がピリオドを戻す場合は、このキーワードを YES に設定します。このキーワードは、特定のアプリケーション・バグを修正するために提供されています。	NO
RDO 互換性 (RDO Compatibility)	RDFX	Remote Data Objects で特定の問題を解決するには、このキーワードを YES に設定します。このキーワードは、その他の目的では設定しないでください。	NO
読み取り専用	RDON	データ・ソースを読み取り専用にするには、このキーワードを YES に設定します。データ・ソースを読み取り専用に設定しても、更新操作は実際には回避されません。しかし、このキーワードを読み取り専用に設定すると、インデックス情報がアプリケーションに戻されるのを回避されます。それにより、通常は更新が試行されなくなります。 **注:** このキーワードを YES に設定すると、標準 DB2 カタログを使用する一部のアプリケーションのパフォーマンスが大幅に向上します。	NO
結果セットのキャッシュ・サイズ (Result-set Cache Size)	RSBK	このキーワードは、結果セットのキャッシュに組み込まれる SQL ステートメントの数を示します。 **注:** 高い値を指定するほど、ディスク・スペースの量が大きくなります。	256

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
結果セットのキャッシング (Result-set Caching)	RSCH	このキーワードを YES に設定すると、ODBC ドライバーが SQL ステートメントの結果セットをキャッシュに入れます。最後のキャッシュの更新から基礎のテーブルが変更されていない場合は、同じ SQL ステートメントが検出されたときに、ローカル・キャッシュから結果が検索されます。	NO
結果セットのキャッシング間隔 (Result-set Caching Interval)	RSIN	このキーワードは、古くなった結果が許容される間隔を秒数で指定します。結果セットのキャッシングを有効にすると、前にキャッシュに入れられた SQL ステートメントが検出されたときに、結果セットが保管されてから経過した秒数がこの間隔内であれば、ホスト処理なしで結果セットが戻されます。値ゼロを指定すると、古い結果が戻されることはなくなります。すべてのステートメントがホストに送信され、そこで基礎のテーブルの変更が検査されます。値に -1 を指定すると、使用可能な場合は常にローカル・キャッシュから検索されます。 **注:** ゼロより大きい許容値を指定するとネットワークの往復を省略できますが、値ゼロを指定した場合でも、基礎のテーブルが変更されていない場合は結果セットがローカル・キャッシュから検索されるので、便利です。	0
行が影響を受けない場合の戻りコード (Return code if no rows affected)	NODA	このキーワードは、INSERT/UPDATE/DELETE がどの行にも影響を与えない場合に戻りコードを何に設定するかを制御します。このキーワードを NODATA に設定すると、どの行にも影響を与えない SQL 操作からの戻りコードは SQL_NO_DATA_FOUND になります。このキーワードを INFO に設定すると、戻りコードは (ODBC 仕様に従って) SQL_SUCESS_WITH_INFO に設定されます。このキーワードは、SUCCESS または ERROR に設定することもできます。その場合、戻りコードは SQL_SUCCESS または SQL_ERROR になります。	INFO
グローバル・テーブルを戻す (Return Global Tables)	RTGL	このキーワードは、グローバル一時テーブルを標準テーブルとして戻すかどうかを制御します。このキーワードを YES に設定すると、グローバル一時テーブルは標準テーブルとして処理され、示されます。このキーワードを NO に設定すると、グローバル一時テーブルはシステム・テーブルとして処理されます。このキーワードは、標準 ODBC ツールをグローバル一時テーブルで使用できるようにするためのものです。これらのツールの多くは、グローバル一時テーブルとして示されたグローバル一時テーブルをバイパスします。しかし、標準テーブルとして示されていれば、これらのツールはグローバル一時テーブルを正しく処理します。	YES

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
ログオン・メッセージを戻す (Return Logon Messages)	LGMG	このキーワードは、エラー以外のログオン・メッセージを ODBC クライアント・アプリケーションに戻すかどうかを制御します。このキーワードを YES に設定し、メッセージがアプリケーションに戻される場合、戻りコードは SQL_SUCCESS_WITH_INFO に設定され、エラー・メッセージ・テキストは Solderer 関数 (またはそれより高いレベルの API) によって使用可能になります。このキーワードを NO に設定すると、メッセージはダイアログ・ボックスを使用して表示されます。	NO
システム・テーブルを戻す (Return System Tables)	RTSY	このキーワードは、システム・テーブル (SYSIBM) を標準テーブルとして戻すかどうかを制御します。このキーワードを YES に設定すると、システム・テーブルは標準テーブルとして処理され、示されます。このキーワードを NO に設定すると、システム・テーブルはシステム・テーブルとして処理されます。このキーワードは、標準 ODBC ツールをシステム・テーブルで使用できるようにするためのものです。これらのツールの多くは、システム・テーブルとして示されたシステム・テーブルをバイパスします。しかし、標準テーブルとして示されていれば、これらのツールはシステム・テーブルを正しく処理します。	NO
Equals の除去 (Remove Equals)	RMEQ	このキーワードは、MDI RSP (リモート・ストアード・プロシージャー) の起動の一環として、equals byte を MDI キーワード名から除去するかどうかを制御します。このキーワードを YES に設定すると、各キーワード名から equals byte が除去されます。このキーワードを NO に設定すると、キーワード名から equals は除去されません。 **注:** このキーワードが適用されるのは、TSQL 0/1/2 構文を使用して起動された MDI RSP だけです。拡張 ODBC CALL 構文を使用して起動された MDI RSP には適用されません。	NO
SBCS モード	SBMD	このキーワードは、SBCS データの処理方法を指定するために使用されます。 SBCS ストリングでは、単一バイト文字と 2 バイト文字が混合していると想定されます。 DBCS 文字の各セットは、SO バイトで始まり、SI バイトで終了します。 BLANK を指定すると、すべての SO/SI バイトがブランクに変換されます。 DELETE を指定すると、すべての SO/SI バイトが各ストリングから削除されます。 **注:** DEFAULT は、現時点ではサポートされていません。	DEFAULT
セキュア・チャネル・プロトコル (Secured Channel Protocol)	SSLX	このキーワードを使用すると、ユーザーは、セキュア・チャネル・プロトコルを選択してクライアント/サーバー・セッションを暗号化し、認証することができます。デフォルトは、最適なパフォーマンスを提供するが強い暗号化と認証はサポートしない標準 NEON セキュア・データ・ストリームです。 SSL2 (SSL バージョン 2) および SSL3 (SSL バージョン 3) は、SSL 規格に基づいています。	STANDARD

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
SQLGetData の順不同呼び出し (SQLGetData out of order)	GTDA	このキーワードは、SQLGetData 呼び出しを順不同で実行できるかどうかを制御します。通常、SQLGetData 呼び出しは、最後にバインドされた列の後から開始して、昇順でのみ行われます。しかし、このキーワードを YES に設定すると、SQLGetData 呼び出しを任意の順序で実行できます。このキーワードを NO に設定すると、SQLGetData 呼び出しに対して標準の制限が適用されます。	NO
SQL ParamOptions サポート (SQL ParamOptions support)	PAOP	このキーワードは、制限された形式で SQLParamOptions をサポートするかどうかを制御します。このキーワードを YES に設定すると、SQLParamOptions がある程度サポートされます。このキーワードを NO に設定すると、SQLParamOptions はサポートされません。このキーワードを SERVER に設定すると、このサポートを有効にするかどうかがサーバー・コードによって決定されます。	YES
SQL サーバー・サポート (SQL Server Support)	SQVA	このキーワードは、SQLTypeInfo 呼び出しが、変数データの長さが 255 バイトであると示すかどうかを制御します。DB2 変数データの長さは実際には 254 バイトしかありません。ただし、これをサポートしないアプリケーションもあります。このキーワードを YES に設定すると、SQLTypeInfo は変数データの長さとして 255 を戻します。このキーワードを NO に設定すると、SQLTypeInfo は 254 を戻します。注: このキーワードを YES に設定すると、ベンダーのインプリメンテーション・エラーを回避するために、SQLCancel の振る舞いが変更されます。	NO
10 進数の後続ゼロの抑制 (Suppress Decimal Trailing Zeros)	STZO	このキーワードは、10 進数のキーワードから後続ゼロを除去するかどうかを制御します。デフォルトは NO です。キーワード AF (MS Access Compatibility) を YES に設定した場合、このキーワードを設定する必要はありません。キーワード AF には、キーワード STZO の機能が含まれています。	NO
Sybase の互換性 (Sybase Compatibility)	SYFU	このキーワードを YES に設定すると、Sybase フィックスのセットが使用されます。これらのフィックスによって、Sybase ODBC サポートでバグがバイパスされます。このキーワードを NO に設定すると、Sybase フィックスは使用されません。	NO
システム・エンジニアリング値 (System Engineering Value)	SEVL	システム・エンジニアリング値は、診断データとデバッグ・データを取得するため、さまざまな値に設定されます。このキーワードを使用するのは、IBM ソフトウェア・サポートから特に要求された場合だけにしてください。	0

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
テーブル名フィルター (Table Name Filter)	TBFL	このキーワードは、製品によって戻されたテーブル名をフィルターに掛けるために使用されます。このキーワードが使用されるのは、これが非ブランク値に設定され、ホスト・アプリケーションがテーブル名フィルター・ストリングを提供していない場合だけです。アプリケーションまたはこのキーワードからのテーブル名フィルター・ストリングは、カタログ照会 (SQLTables、SQLColumns、SQLTablePrivileges) によって戻される情報を制限します。提供されたテーブル名のパターンに一致する行だけが戻されます。このキーワードを単一の % 記号に設定すると、すべてのテーブルが戻されます。このキーワードのデフォルト値はありません。 **例:** TBFL=STAFF は、STAFF というラベルのテーブルからの情報だけを表示します。TBFL=STA% は、STA で始まるすべてのテーブルからの情報を表示します。TBFL=STAFF BOOK% は、STAFF というラベルのテーブルからの情報および BOOK で始まるすべてのテーブルからの情報を表示します。	
テーブル所有者/同義語オプション (Table Owner/Synonym Option)	SYOP	このキーワードは、実際には同義語であるテーブルのテーブル所有者をどのように処理するかを示すために使用されます。これらの選択項目のいくつかは、デスクトップ生産性向上ツールで NEON ODBC ドライバーを使用するために必要です。	ZERO
テーブル修飾子オプション (Table Qualifier Option)	TQOP	このキーワードは、テーブル修飾子がどのように ODBC アプリケーション・プログラムに戻されるかを指定するために使用されます。テーブル修飾子を各テーブルの名前全体の一部として使用するデータベースもあります。つまり、テーブル修飾子が異なっていれば、2 つのテーブルがその他のすべての点でまったく同じ名前を持つこともできます。 TQOP オプションは、テーブル修飾子情報が ODBC アプリケーションにどのように戻されるかを制御します。使用できる値は、NORMAL、NULL、および ZERO の 3 つです。	NORMAL
トランザクション名 (Transaction Name)	TRNA	このキーワードは、トランザクション名を指定するために使用されます。トランザクション名の使用は、アプリケーション固有です。長さは、最大 8 バイトです。トランザクション名は、右側をブランクで埋め込まれます。	
リテラル・ストリングの切り捨て (Truncate Literal String)	TRLT	このキーワードは、リテラル・ストリングの切り捨てをオフにするために使用されます。 TRLT=NO を指定すると、ストリングの長さが 20 を超えていても、リテラル・ストリングは切り捨てられません。	YES

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
通信 DLL のアンロード (Unload Communication DLLs)	FEDL	このキーワードは、アプリケーションの切断後に通信 DLL (WINSOCK.DLL や CPIC.DLL など) をアンロードするかどうかを判別するために使用されます。デフォルトでは、アプリケーションの切断後に Shadow Direct によって必ず通信 DLL がアンロードされます。このキーワードを NO に設定すると、通信サービスがまだアクティブであるときに通信 DLL をアンロードした場合に、エラーが発生します。このキーワードを NO に設定すると、ODBC ドライバーによって DLL がアンロードされることはありません。	YES
すべての文字データを大文字にする (Upcase all character data)	UPCH	このキーワードは、ホストに送信されるすべての文字データを大文字に変換するかどうかを制御します。このキーワードを YES に設定すると、すべての文字データが大文字に変換されます。このキーワードを NO に設定すると、文字データは大文字に変換されません。	NO
二重引用符付きストリングを大文字にする (Upcase double quote string)	UPDB	このキーワードは、二重引用符で囲まれたストリングを大文字に変換するかどうかを制御します。このキーワードを YES に設定すると、二重引用符で囲まれたストリングが大文字に変換されます。このキーワードを NO に設定すると、二重引用符で囲まれたストリングは変更されません。DB2 は二重引用符で囲まれたテーブル名をリテラルとして扱うので、ストリングを大文字に変換しなければならない場合があります。	NO
UDP 最大再試行 (UDP Maximum Retry Attempts)	MXRT	このキーワードは、未確認の要求の UDP 再試行の最大数を制御します。この値は、負の数以外の数値にする必要があります。このキーワードを ZERO に設定すると、UDP 要求の再試行は使用不可になります。	10
UDP データグラム再試行タイムアウト (UDP Datagram Retry Timeout)	RTTM	このキーワードは、UDP 要求が再送されるまでの初期タイムアウト期間を制御します。このタイムアウト値は後続の再試行では 2 倍になりますが、この値はバウンドなしには増加しません。指定できる最大タイムアウトは 30 秒です。最大再試行を ZERO に設定すると、このキーワードの効果はなくなります。	30
Visual Age の互換性 (Visual Age Compatibility)	VAFX	Visual Age で特定の問題を解決するには、このキーワードを YES に設定します。このキーワードは、その他の目的では設定しないでください。	NO
Wollongong Pathworks 3.1 サポート (Wollongong Pathworks 3.1 Support)	WGPH	このキーワードは、 Wollongong Pathworks 3.1 16 ビット TCP/IP 環境で稼働している場合に必要です。	NO

表 31. ドライバー・キーワード (続き)

説明	キーワード	(フォーマット) 説明	デフォルト
X/OPEN XA サポート (X/OPEN XA Support)	XAEN	このキーワードは、ターゲット・サーバーが X/OPEN XA トランザクションに参加する場合に設定する必要があります。値 TWO-PHASE は、トランザクションに複数のリソース・マネージャーが含まれ、ターゲット・サーバーが 2 フェーズ・コミット・プロトコルをサポートしている場合に使用します。値 TWO-PHASE を指定すると、コーディネーターが適切と判断した場合には、1 フェーズ・コミット・シナリオも使用できます。デフォルトは NONE です。これは、トランザクションのどの部分も XA プロトコルを使用しないことを示します。	NONE
X/OPEN XA Transaction Manager	XAOP	このキーワードは、 XA Transaction Manager のタイプを設定するために使用されます。モニターによって調整される分散トランザクションに参加する場合は、このオプションを正しく設定する必要があります。 MSDTC は、主に MTS アプリケーションによって使用される Microsoft Distributed Transaction Manager です。 BEA Tuxedo によって調整されるトランザクションに対しては、2 つの値を使用できます。 TUXEDO-TMS を指定すると、ODBC 接続が Tuxedo TMS サーバーによって所有されることが示されます。 TUXEDO-SQL を指定すると、ODBC 接続が Tuxedo SQL アプリケーション・サーバーによって所有されることが示されます。デフォルトは NONE です。これは、これが分散トランザクションではないことを意味します。	NONE
アクティブ・ステートメントがゼロ (Zero Active Statements)	ZEAS	このキーワードは、アクティブ・ステートメントの数としてゼロを戻すかどうかを制御します。このキーワードを YES に設定すると、ODBC ドライバーによってサポートされるアクティブ・ステートメントの数としてゼロが戻されます。このキーワードを NO に設定すると、サポートされるアクティブ・ステートメントの実際の数が戻されます。	NO
列名がゼロ (Zero Column Names)	ZECL	列名を 2 進ゼロに設定する場合は、このキーワードを YES に設定します。これは、実動アプリケーションのパフォーマンスの最適化です。	NO

ドライバー・キーワードの設定値

すべてのキーワードを、前のセクションで説明したすべてのメソッドを使用して設定できるわけではないので、次の表に、それぞれの設定方法と、キーワードがオプション (チェックマークが付けられている) かどうか (つまり、ODBC.INI ファイルまたは接続ストリングのいずれかに組み込む必要があるかどうか) を示します。

表 32. ドライバー・キーワードおよび設定値

キーワード	ODBC.INI	接続ストリング
ALCD	X	X
ALUS	X	X
APPC	X	X
APPL	X	X
AT	X	X

表 32. ドライバー・キーワードおよび設定値 (続き)

キーワード	ODBC.INI	接続ストリング
AF	X	X
BMPA[1]	X	X
BYDB	X	X
CD	X	X
CGFX	X	X
CLOR	X	X
CNDG	X	X
CPFX	X	X
CTLV		X
DBD	X	X
説明	X	X
DP	X	X
DRIVER		X
DSFL	X	X
DSN		X
FEDL	X	X
HD	X	X
HOST	X	X
IDQO	X	X
LAFX	X	X
LGID	X	X
LGPA	X	X
LINK	X	X
LKES	X	X
LKTH	X	X
LLUA[2]	X	X
LNDN		X
LNID		X
MODE[2]	X	X
MR	X	X
MT	X	X
MXBU	X	X
NB10		
NEONTRACE	X	X
NOAS	X	X
NO3D	X	X
NOUS	X	X
NOPW	X	X
NWPW		X
PBFU	X	X
PGNA[3]	X	X
PGPA	X	X
PLAN	X	X
PLUA[4]	X	X
PORT[5]	X	X
PRES	X	X
PSB[6]	X	X
PWD	X	X
QUNA	X	X
RDON	X	X
RO	X	X
SECU	X	X
SIDEINFO	X	X

表 32. ドライバー・キーワードおよび設定値 (続き)

キーワード	ODBC.INI	接続ストリング
SRN	X	X
SUBSYS	X	X
SYOP	X	X
TQOP	X	X
UID	X	X
USERPARM	X	X
VAFX	X	X
WACU	X	X
ZECL	X	X

1. BMP 名キーワードは、単一スレッド IMS 接続に必要です。
2. このフィールドは、OS/2 および LU 6.2 ホスト接続に必要です。
3. PGNA キーワードは、ホストへの RPC 接続に必要です。
4. このフィールドは、OS/2 および LU 6.2 ホスト接続に必要です。
5. ポート番号は、TCP/IP ホスト接続に必要です。
6. PSB 名は、マルチスレッド IMS ホスト接続に必要です。

付録 D. ビジネス・オブジェクトのサンプル

この付録では、MAS 用コネクターに組み込まれているサンプル・ビジネス・オブジェクトについて詳しく説明します。以下のコネクター用にビジネス・オブジェクトのサンプルが用意されています。

- DB2 コネクター
- VSAM コネクター
- NATURAL コネクター
- CICS コネクター
- IDMS(DB) コネクター
- ADABAS コネクター
- IMS_TM コネクター

DB2 コネクター

テストに使用するビジネス・オブジェクト

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = DB2XWORLDS_EVENTS
Version = 3.0.0
AppSpecificInfo = TN=SCJAB.XWORLDS_EVENTS

    [Attribute]
    Name = EVENT_ID
    Type = Integer
    MaxLength = 255
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=EVENT_ID
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = OBJECT_KEY
    Type = Integer
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=OBJECT_KEY
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = OBJECT_NAME
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=OBJECT_NAME
```

```
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = EVENT_PRIORITY
    Type = Integer
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=EVENT_PRIORITY
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = EVENT_STATUS
    Type = Integer
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=EVENT_STATUS
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = OBJECT_VERB
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=OBJECT_VERB
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
[End]
[End]
```

VSAM コネクター

テストに使用するビジネス・オブジェクト

```
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = VSAM_DEMOVSAM
Version = 3.0.0
AppSpecificInfo = TN=DEMOVSAM

    [Attribute]
    Name = FILEREC
    Type = String
    MaxLength = 80
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=FILEREC
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = STAT
    Type = String
    MaxLength = 80
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=STAT
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = NUMB
    Type = String
    MaxLength = 80
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=NUMB
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = NAME
    Type = String
    MaxLength = 80
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=NAME
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ADDRX
    Type = String
    MaxLength = 80
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=ADDRX
    IsRequiredServerBound = false
    [End]

    [Attribute]
```

```
    Name = PHONE
    Type = String
    MaxLength = 80
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=PHONE
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = AMOUNT
    Type = String
    MaxLength = 80
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=AMOUNT
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]
```

NATURAL コネクター

テストに使用するビジネス・オブジェクト

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = NaturalWrap
Version = 3.0.0
AppSpecificInfo = WRAPPER=true

    [Attribute]
    Name = TestNatural
    Type = TestNatural
    ContainedObjectVersion = 3.0.0
    Relationship = Containment
    Cardinality = N
```

```
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Func
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=Func
DefaultValue = SEND
IsRequiredServerBound = false
[End]

[Attribute]
Name = SerInOut
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=SerInOut
DefaultValue= ACISP03,SP03MIN1,SP03MOT1
IsRequiredServerBound = false
[End]

[Attribute]
Name = Param1
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=Param1
DefaultValue = FIND
IsRequiredServerBound = false
[End]

[Attribute]
Name = Param2
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=Param2
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength =
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]
```

```
    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]

[BusinessObjectDefinition]
Name = TestNatural
Version = 3.0.0
AppSpecificInfo = TN=Dummy

    [Attribute]
    Name = FuncC
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = true
    IsRequired = false
    AppSpecificInfo = CN=FuncC:FK=Func
    DefaultValue = SEND
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = SerInpOutC
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = true
    IsRequired = false
    AppSpecificInfo = CN=SerInOutC:FK=SerInOut
    DefaultValue = ACISP05B,SP03MIN1,SP03MOT1
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = Param1C
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = true
    IsRequired = false
    AppSpecificInfo = CN=Param1C:FK=Param1
    DefaultValue = FIND
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = Param2C
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = true
    IsRequired = false
    AppSpecificInfo = CN=Param2C:FK=Param2
    IsRequiredServerBound = false
    [End]

    [Attribute]
```

```
Name = RetrieveSP
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = SPN=shadow_aci;RS=true;
 IP=FuncC:SerInpOutC:Param1C:Param2C
IsRequiredServerBound = false
[End]

[Attribute]
Name = REcInfo
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=RECORD_INFO
IsRequiredServerBound = false
[End]

[Attribute]
Name = PersonnelId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=PERSONNEL_ID
IsRequiredServerBound = false
[End]

[Attribute]
Name = LastName
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=LAST_NAME
IsRequiredServerBound = false
[End]

[Attribute]
Name = FirstName
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=FIRST_NAME
IsRequiredServerBound = false
[End]

[Attribute]
Name = MiddleI
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=MIDDLE_I
IsRequiredServerBound = false
[End]

[Attribute]
```

```
Name = Addr01
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ADDRESS_LINE001
IsRequiredServerBound = false
[End]

[Attribute]
Name = Addr02
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ADDRESS_LINE002
IsRequiredServerBound = false
[End]

[Attribute]
Name = Addr03
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ADDRESS_LINE003
IsRequiredServerBound = false
[End]

[Attribute]
Name = City
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CITY
IsRequiredServerBound = false
[End]

[Attribute]
Name = ZIP
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ZIP
IsRequiredServerBound = false
[End]

[Attribute]
Name = Dept
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=DEPT
IsRequiredServerBound = false
[End]

[Attribute]
Name = JobTitle
```

```
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=JOB_TITLE
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]
```

CICS コネクター

テストに使用するビジネス・オブジェクト

```
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = CICSUPD_WRAP
Version = 3.0.0
AppSpecificInfo = WRAPPER=true

    [Attribute]
    Name = child
    Type = CICSUPD_CHILD
    ContainedObjectVersion = 1.0.0
    Relationship = Containment
    Cardinality = N
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = CA_IN_UPD
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = false
```

```
IsRequired = false
AppSpecificInfo = CN=CA_IN_UPD
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_LID
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CA_IN_LID
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_NAME
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CA_IN_NAME
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_PHONE
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CA_IN_PHONE
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_EMAIL
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CA_IN_EMAIL
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength =
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Delete
[End]
```

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

    [Verb]
    Name = Update
    [End]
[End]
[BusinessObjectDefinition]
Name = CICSUPD_CHILD
Version = 3.0.0
AppSpecificInfo = TN=Dummy

    [Attribute]
    Name = DeleteSP
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = SPN=CICSEX.NESPUPD;RS=true;
     IP=CA_IN_UPD:CA_IN_LID:CA_IN_NAME:CA_IN_PHONE:CA_IN_EMAIL
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = UpdateSP
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = SPN=CICSEX.NESPUPD;RS=true;
     IP=CA_IN_UPD:CA_IN_LID:CA_IN_NAME:CA_IN_PHONE:CA_IN_EMAIL
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = CreateSP
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = SPN=CICSEX.NESPUPD;RS=true;
     IP=CA_IN_UPD:CA_IN_LID:CA_IN_NAME:CA_IN_PHONE:CA_IN_EMAIL
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = BeforeRetrieveSP
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = SPN=CICSEX.WBIRETRV;RS=false
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = CA_IN_UPD
    Type = String
    MaxLength = 2
    IsKey = true
    IsForeignKey = true
    IsRequired = false
```

```
AppSpecificInfo = CN=CA_IN_UPD:FK=CA_IN_UPD
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_LID
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = true
IsRequired = false
AppSpecificInfo = CN=CA_IN_LID:CA_IN_LID
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_NAME
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = true
IsRequired = false
AppSpecificInfo = CN=CA_IN_NAME:CA_IN_NAME
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_PHONE
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = true
IsRequired = false
AppSpecificInfo = CN=CA_IN_PHONE:CA_IN_PHONE
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_IN_EMAIL
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = true
IsRequired = false
AppSpecificInfo = CN=CA_IN_EMAIL:CA_IN_EMAIL
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_MESSAGE
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = CN=CA_MESSAGE
DefaultValue = test
IsRequiredServerBound = false
[End]

[Attribute]
Name = CA_STATUS
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
    AppSpecificInfo = CN=CA_STATUS
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]

 [BusinessObjectDefinition]
Name = CICSEX_RET_WRAP
Version = 3.0.0
AppSpecificInfo = WRAPPER=true

    [Attribute]
    Name = Child
    Type = CICSEX_RET
    ContainedObjectVersion = 1.0.0
    Relationship = Containment
    Cardinality = N
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = CA_IN_LID
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=CA_IN_LID
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
```

```
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]
[BusinessObjectDefinition]
Name = CICSEX_RET
Version = 3.0.0
AppSpecificInfo = TN=Dummy

    [Attribute]
    Name = RetrieveSP
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = SPN=CICSEX.NESPSEL;RS=true;IP=CA_IN_LID
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = CA_IN_LID
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = true
    IsRequired = false
    AppSpecificInfo = CN=CA_IN_LID:FK=CA_IN_LID
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = CA_OUT_NAME
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=CA_OUT_NAME
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = CA_OUT_PHONE
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=CA_OUT_PHONE
    IsRequiredServerBound = false
    [End]
```

```
    [Attribute]
    Name = CA_OUT_EMAIL
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=CA_OUT_EMAIL
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]
```

IDMS(DB) コネクター

テストに使用するビジネス・オブジェクト

```
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = IDMSBO
Version = 3.0.0
AppSpecificInfo = WRAPPER=true

    [Attribute]
    Name = Dummy
    Type = String
    MaxLength = 255
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=Dummy
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = IDMSChildBO
    Type = IDMSChildBO
    ContainedObjectVersion = 1.0.0
    Relationship = Containment
```

```
    Cardinality = N
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
    [End]
 [BusinessObjectDefinition]
Name = IDMSChildBO
Version = 3.0.0
AppSpecificInfo = TN=Dummy

    [Attribute]
    Name = RetrieveSP
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = SPN=SDCOIM;RS=true
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = INVCODE
    Type = Integer
    MaxLength = 255
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=INVCODE
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = PARTNUMBER
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
```

```
    IsRequired = false
    AppSpecificInfo = CN=PARTNUMBER
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = PARTDESC
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=PARTDESC
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]
```

ADABAS コネクター

テストに使用するビジネス・オブジェクト

```
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = ADABAS_VEHICLES
Version = 3.0.0
AppSpecificInfo = TN=Vehicles

    [Attribute]
    Name = REG_NUM
    Type = String
    Cardinality = 1
    MaxLength = 255
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=REG_NUM
    IsRequiredServerBound = false
    [End]
```

```
[Attribute]
Name = CHASSIS_NUM
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CHASSIS_NUM
IsRequiredServerBound = false
[End]

[Attribute]
Name = PERSONNEL_ID
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=PERSONNEL_ID
IsRequiredServerBound = false
[End]

[Attribute]
Name = MAKE
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=MAKE
IsRequiredServerBound = false
[End]

[Attribute]
Name = MODEL
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=MODEL
IsRequiredServerBound = false
[End]

[Attribute]
Name = COLOR
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=COLOR
IsRequiredServerBound = false
[End]

[Attribute]
Name = YEAR
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
```

```
IsRequired = false
AppSpecificInfo = CN=YEAR
IsRequiredServerBound = false
[End]

[Attribute]
Name = CLASS
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CLASS
IsRequiredServerBound = false
[End]

[Attribute]
Name = LEASE_PUR
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=LEASE_PUR
IsRequiredServerBound = false
[End]

[Attribute]
Name = DATE_ACQ
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=DATE_ACQ
IsRequiredServerBound = false
[End]

[Attribute]
Name = CURR_CODE
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CURR_CODE
IsRequiredServerBound = false
[End]

[Attribute]
Name = MAINT_COST_1
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=MAINT_COST_1
IsRequiredServerBound = false
[End]

[Attribute]
Name = MAINT_COST_2
```

```
    Type = Integer
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=MAINT_COST_2
    DefaultValue = 0
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = MAINT_COST_3
    Type = Integer
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=MAINT_COST_3
    DefaultValue = 0
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]
```

IMS_TM コネクター

テストに使用するビジネス・オブジェクト

```
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = IMSTMBO
Version = 3.0.0
AppSpecificInfo = TN=Dummy

    [Attribute]
    Name = ADDFLD
    Type = String
```

```
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ADDFLD
IsRequiredServerBound = false
[End]

[Attribute]
Name = CHGFLD
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=CHGFLD
IsRequiredServerBound = false
[End]

[Attribute]
Name = REMFLD
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=REMFLD
IsRequiredServerBound = false
[End]

[Attribute]
Name = REVFLD
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=REVFLD
IsRequiredServerBound = false
[End]

[Attribute]
Name = D05751
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D05751
IsRequiredServerBound = false
[End]

[Attribute]
Name = D05757
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D05757
IsRequiredServerBound = false
[End]

[Attribute]
Name = D05752
Type = String
MaxLength = 255
```

```
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D05752
IsRequiredServerBound = false
[End]

[Attribute]
Name = D05754
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D05754
IsRequiredServerBound = false
[End]

[Attribute]
Name = D05755
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D05755
IsRequiredServerBound = false
[End]

[Attribute]
Name = D05756
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D05756
IsRequiredServerBound = false
[End]

[Attribute]
Name = D15333
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D15333
IsRequiredServerBound = false
[End]

[Attribute]
Name = D15334
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=D15334
IsRequiredServerBound = false
[End]

[Attribute]
Name = D07786
Type = String
MaxLength = 255
IsKey = false
```

```
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=D07786
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = F00028
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=F00028
    IsRequiredServerBound = false
    [End]

[Attribute]
    Name = RetrieveSP
    Type = String
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = SPN=ims.imssp;IO=ADDFLD:CHGFLD:REMFLD:REVFLD:
D05751:D05757:D05752:D05754:D05755:D05756:D15333:D15334:D07786:F00028
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = ObjectEventId
    Type = String
    MaxLength =
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    [End]

    [Verb]
    Name = Delete
    [End]

    [Verb]
    Name = Retrieve
    [End]

    [Verb]
    Name = Update
    [End]
[End]
```

付録 E. ヌル値およびブランク値のサポート

この付録では、ビジネス・オブジェクトのキー値がブランクまたはヌルの場合のさまざまな合格/不合格シナリオについて詳しく説明します。この付録では、ブランクまたはヌルのビジネス・オブジェクト値を持つために必要な機能的変更についても説明します。

成功と失敗のシナリオ

ビジネス・オブジェクトのキー値がデータベース内にブランク値またはヌル値を持っている場合は、「=」演算子タイプではなく「is null」タイプの where 文節を構築してください。

IBM では、ビジネス・オブジェクトに対して、ブランク値を持たないキー属性を 1 つ以上定義することをお勧めします。

以下は、1 つのキーを持ち、このキーがヌル値を持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは不合格です。

表 33. Customer

属性	タイプ
cid	Integer (キー)
name	String
comments	String

以下は、2 つのキーを持ち、そのうちの 1 つがヌル値を持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは合格です。

表 34. Customer

属性	タイプ
cid	Integer (キー)
name	String
comments	String

2 つ目の例では、cid=1000 かつ name がヌルに設定されているという条件で customer から cid、name、および comments を選択することにより、retrieve 照会を構築します。

以下は、コンテナー・オブジェクト内に 1 つの外部キー参照を持つ子オブジェクトを 1 つ持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは不合格です。

表 35. Customer

属性	タイプ
cid	Integer (キー)

表 35. Customer (続き)

属性	タイプ
name	String (キー)
comments	String
Address	Address
Aid	Integer (キー) ASI:FK=cid
Acity	String
Azip	String

cid にヌル値が含まれている場合は、address から Aid、Acity、および Azip を選択することにより、retrieve 照会を構築します。Aid の値はヌルに設定してください。

以下は、コンテナー・オブジェクト内に 2 つの外部キー参照を持つ子オブジェクトを 1 つ持っている親オブジェクトのシナリオです。このような条件に当てはまるシナリオは合格です。

表 36. Customer

属性	タイプ
cid	Integer (キー)
name	String
comments	String
Address	Address
Aid	Integer (キー) ASI:FK=cid
Acity	String (キー) ASI:FK=name
Azip	String

name にヌル値が含まれている場合は、Aid=Cid と Acity がヌル値を持っているという条件で address から Aid、Acity、および Azip を選択することにより、Retrieve 照会を構築します。

機能性

コネクターは、キーにブランク値を見つけると、その値を属性の UseNull 値と比較します。この結果の値が true の場合、コネクターはヌル値を照会に追加します。これにより、以下の動詞の操作が影響を受けます。

- Retrieve
- RetrieveBy Content
- Update
- Delete

特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

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

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory

577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

汎用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

以下は、IBM Corporation の商標です。

IBM
IBM ロゴ
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

MMX、Pentium および ProShare は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

Mainframe Adapter Suite には、Eclipse Project (http://www.eclipse.org) により開発されたソフトウェアが組み込まれています。

IBM WebSphere Business Integration Adapter Framework V2.4