IBM WebSphere Business Integration Server Express and Express Plus

Adapter for mySAP.com (SAP R/3 Version 4.x) ユーザーズ・ガイド

アダプター・バージョン 6.0.x

IBM WebSphere Business Integration Server Express and Express Plus

Adapter for mySAP.com (SAP R/3 Version 4.x) ユーザーズ・ガイド

アダプター・バージョン 6.0.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、“特記事項”に記載されている情報をお読みください。

本書は、Adapter for mySAP.com for BASIS 4.0-4.6, SAP Web AS 6.20 (5724-H01)、バージョン 6.0.x に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Server Express and Express Plus
Adapter for mySAP.com User Guide (for BASIS 4.0-4.6, SAP Web AS 6.20)
Adapter Version 6.0.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 2005.8

この文書では、平成明朝体™W3、平成明朝体™W7、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™W7、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

製品 IBMR WebSphere Business Integration Server Express および IBMR WebSphere Business Integration Server Express Plus は、InterChange Server Express、関連する Toolset Express、CollaborationFoundation、およびソフトウェア統合アダプターのセットで構成されています。Toolset Express に含まれるツールは、ビジネス・オブジェクトの作成、変更、および管理に役立ちます。プリパッケージされている各種アダプターは、お客様の複数アプリケーションにまたがるビジネス・プロセスに応じて、いずれかを選べるようになっています。標準的な処理のテンプレートである CollaborationFoundation は、カスタマイズされたプロセスを簡単に作成できるようにするためのものです。

IBM[R] WebSphere[R]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、レガシーおよびメインフレーム・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、ビジネスの統合を実現します。

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

本書に記載する内容

本書では、Adapter for mySAP.com のインストール、コネクター・プロパティーの構成、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。

本書に記載しない内容

本書では、サーバーのロード・バランシング、アダプターの処理スレッド数、最大スループットと最小スループット、許容度のしきい値など、配置のメトリックやキャパシティー・プランニングの問題については触れていません。

このような問題は、お客様の配置ごとに固有のものであり、アダプターを配置する正確な環境の中で、あるいはそれに近い環境で測定する必要があります。使用する配置サイトの構成についての質問や、特定の構成を前提とした、この種のメトリックの計画と評価の詳細については、IBM サービス技術員にお問い合わせください。

対象読者

本書は、IBM のコンサルタントおよびお客様を対象としています。本書の読者は、SAP および WebSphere Business Integration システム・アダプターの開発について十分な知識と経験を持っている必要があります。

関連文書

本書の対象製品の一連の関連文書には、WebSphere Business Integration Server Express のどのインストールにも共通する機能とコンポーネントの解説のほか、特定のコンポーネントに関する参考資料が含まれています。

関連文書は、http://www.ibm.com/websphere/wbiserverexpress/infocenter でダウンロード、インストール、および表示することができます。

注: 本書の発行後に公開されたテクニカル・サポートの技術情報や速報に、本書の対象製品に関する重要な情報が記載されている場合があります。これらの技術情報や速報は、WebSphere Business Integration のサポート Web サイト (http://www.ibm.com/software/integration/websphere/support/) で参照できます。適切なコンポーネント領域を選択し、「Technotes (技術情報)」セクションと「Flashes (速報)」セクションを参照してください。IBM Redbooks (http://www.redbooks.ibm.com/) に、追加情報が提供されている場合もあります。

表記上の規則

本書は下記の規則に従って編集されています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
太字	初出語を示します。
italic、*italic*	変数名または相互参照を示します。
青のアウトライン	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	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 つのオプションだけ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、オプショナル・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[,...]` 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・パスの規則として円記号 (¥) を使用します。すべての製品のパス名は、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。
%text%	パーセント記号 (%) 内のテキストは、Windows™ `text` システム変数またはユーザー変数の値を示します。

ProductDir	製品のインストール先ディレクトリーを表します。各プラットフォームのデフォルトは、以下のとおりです。 Windows: IBM¥WebSphereServer i5/OS: /QIBM/ProdData/WBIServer44/AdapterCapacityPack Linux: /home/${username}/IBM/WebSphereServer

命名規則

本書では、以下の命名規則を使用します。

- Adapter for mySAP.com のコネクター・コンポーネントは、「コネクター」と略記します。
- 「コネクター」は、Vision コネクター・フレームワークとコネクター・モジュールの組み合わせを意味します。

本リリースの新機能

バージョン 6.0.x の新機能

編集上の変更:

アダプター環境に関する変更:

SAPODA に関する変更:

BAPI Module に関する変更:

ABAP Extension Module に関する変更:

ALE Module に関する変更:

使用中止となった機能に関する変更:

OS/400 V5R2 および i5/OS V5R3 でもサポート

Windows 2003 のアダプター V 6.x で BIDI をサポート

概要

この章では、Connector for mySAP.com について概説します。この章の内容は以下のとおりです。

- 『アダプター環境』
- xix ページの『コネクター・アーキテクチャー』
- xxi ページの『Vision コネクター・フレームワークの動作方法』

Connector for mySAP.com は、Adapter for mySAP.com のランタイム・コンポーネントです。mySAP.com Adapter には、コネクター、メッセージ・ファイル、構成ツール、および Object Discovery Agent (ODA) が含まれています。コネクターは、統合ブローカーが SAP アプリケーションとの間でビジネス・オブジェクトを交換できるようにします。

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・フレームワークの 2 種類で構成されています。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合わせたコードが格納されています。コネクター・フレームワークのコードはすべてのコネクターに共通なので、コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの仲介役の機能を果たします。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・オブジェクトの送信および受信
- 始動メッセージや管理メッセージの交換の管理

この資料には、アプリケーション固有のコンポーネントとコネクター・フレームワークに関する情報が記載されています。この資料では、この 2 つのコンポーネントのことを、どちらもコネクターと呼んでいます。

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。このセクションには、以下のトピックがあります。

- 『ブローカーの互換性』
- xvi ページの『アダプターがサポートするソフトウェアと標準』
- xvi ページの『アダプターのプラットフォーム』
- xvi ページの『アダプターの依存関係』
- xvii ページの『ロケール依存データ』

ブローカーの互換性

このアダプターは、統合ブローカー InterChange Server Express と互換性があります。

アダプターのプラットフォーム

このアダプターには、ブローカーのほかに、以下のいずれかのオペレーティング・システムが必要です。

- すべてのオペレーティング・システム環境で、カスタム・アダプターのコンパイル用に Java コンパイラー (IBM JDK 1.4.2 for Windows 2003) が必要です。
- **Linux:**
  Red Hat Linux AS 3.0 Update 1、Intel (IA32)
  SuSE Linux 8.1、SP3、Intel (IA32)
  SuSE Linux ES 9.0、Intel (IA32)

  注: WebSphere Business Integration Adapter FrameworkV2.6 の TMTP (Tivoli Monitoring for Transaction Performance) コンポーネントは、Linux Red Hat ではサポートされません。
- **Windows:**
  Windows XP Service Pack 1A (WebSphere Business Integration Adapter Framework (管理ツールのみ) 用)
  Windows 2003 (Standard Edition または Enterprise Edition)
- **OS/400 V5R2 と i5/OS V5R3:**
  特に明記しない限り、i5/OS は OS/400 および i5/OS を指します。

アダプターがサポートするソフトウェアと標準

- SAP アプリケーション・サーバーの バージョン 4.0、4.5、4.6、および 6.20 (6.20 Unicode を含む) で稼働する SAP アプリケーションのサポート
- アダプターが Microsoft Windows 2003 上で稼働し、SAP 6.20 Unicode システムに接続する場合に限り、双方向 (bi-di) データをサポート

アダプターの依存関係

コネクターをインストールする前の作業は以下のとおりです。

- SAP に対して、使用するバージョン用の最新の SAP サポート・パッケージをインストールしてください。

  SAP では、サポート・パッケージとして Basis、R/3 アプリケーション、ABAP、および HR を提供しています。SAP アプリケーションの ABAP コードに対するバグ・フィックスが提供されています。更新済みの SAP カーネルを使用してください。このカーネルは C++ で記述された実行可能ファイルで、トランスポートを実行し、オペレーティング・システムとのインターフェースとなり、データベースと通信し、システムを実行します。
- SAP アプリケーションで CPIC ユーザー・アカウントをセットアップしてください。このアカウントに、コネクターがサポートしているビジネス・オブジェクトが必要とするデータを操作するために必要な特権を与えます。

  例えば、コネクターが何らかの SAP ビジネス・トランザクションを実行する必要がある場合、SAP アプリケーションにおけるコネクターのアカウントには、これらのトランザクションを実行する権限が必要です。このアカウント情報を使用することにより、コネクター固有構成プロパティーである `ApplicationUserName` および `ApplicationPassword` を設定することが必要です。これらのプロパティー

の設定方法の詳細については、 13 ページの『第 2 章 コネクターの構成』 と 341 ページの『付録 C. コネクター固有の構成プロパティー』 を参照してください。

- SAP で、コネクターのインストール特権および管理特権を持つユーザー・アカウントをセットアップします。このアカウントには、以下の特性を付与する必要があります。
  - 有効な SAP ユーザー名およびパスワード
  - ABAP 開発者アクセス
  - テーブル構成アクセス
  - コネクターを管理およびモニターするためのトランザクション SM21 および SM50 に対する管理アクセス

コネクターをインストールした後の作業は以下のとおりです。

- SAP JavaAPI をインストールします。

  SAP の Java API は、Java コネクター (SAP JCo) と呼ばれています。現在、SAP アダプターは SAPJCo V.2.1.3 および V.2.1.4 をサポートします。本書で示す SAPJCo のバージョンは SAP サービス・マーケットプレイスからダウンロードすることができないため、IBM 担当員にお問い合わせください。

  SAP JCo は接続をプールして、要求の実行にどの接続を使用すべきかをアダプターに通知します。アダプターのすべての接続プロパティーは、コネクター構成ファイルに設定されます。JCo.PoolManager は、SAP アプリケーションの接続プールに関するすべての構成を管理します。

  このコネクターの依存関係のインストールの詳細については、 8 ページの『SAP の Java コネクター (SAPJCo) のインストール』を参照してください。

  コネクター・プロパティーの詳細については、 13 ページの『第 2 章 コネクターの構成』、 315 ページの『付録 B. 標準構成プロパティー』、および 341 ページの『付録 C. コネクター固有の構成プロパティー』 を参照してください。

ロケール依存データ

コネクターは、Unicode 以外の SAP システムに対して、マルチバイト文字セットをサポートできるように国際化されています。このセクションの情報は、Unicode ベースの SAP システムには当てはまらないことに注意してください。

このアダプターは、アラビア語、ヘブライ語、ウルドゥー語、ペルシア語、イディッシュ語などの言語に対する、双方向スクリプト・データの処理をサポートします。双方向の能力を使用するには、双方向の標準プロパティーを構成する必要があります。詳細については、双方向の標準プロパティーを構成する必要があります。詳細については、 315 ページの『付録 B. 標準構成プロパティー』にある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。

コネクターは、1 つの文字コード・セットを使用する場所から別のコード・セットを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。

注: WebSphere BI Station ログは英語でのみ、使用可能です。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系と複数バイト系を含む) の文字に対応できるエンコード方式が組み込まれています。IBM WebSphere Business Integration システムのほとんどのコンポーネントは Java で記述されています。したがって、ほとんどの IBM WebSphere Business Integration コンポーネントの間でデータが転送されても、文字変換の必要はありません。

このコネクターは Java で作成されているため、固有のエンコード方式で作成されているアプリケーション・データ (IDoc ファイル内のデータを含む) は変換の必要がありません。SAP JCo ライブラリーは、このようなアプリケーション・データを、コネクターが処理する前に、Unicode に変換します。図 1 に、データ変換に関係するコンポーネントを示します。

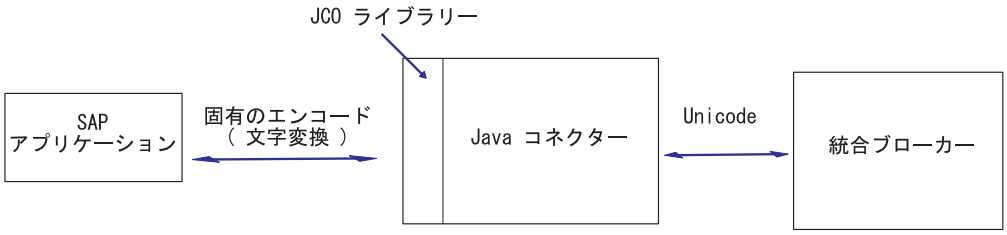


図 1. エージェント・プロパティーの構成

エラー・メッセージと通知メッセージを適切な言語で適切な国と地域に合わせて記録するには、該当する環境の Locale 標準構成プロパティーを設定します。これらのプロパティーの詳細については、315 ページの『付録 B. 標準構成プロパティー』を参照してください。

重要: SAP アプリケーションが日本語用に使用する文字コード・セットは SAP-8000 です。このコード・セットは MS932 文字をサポートしていません。また、SJIS により標準でない Unicode 文字にマップされる文字もあります。したがって、SAP JCo ライブラリーは一部の文字を処理できません。これらの処理できない文字が SAP アプリケーションまたは IBM WebSphere ビジネス・オブジェクトに含まれていると、コネクターはそれらの文字を # または ? 文字に置換します。データにこれらの文字が含まれていると、データが正しく処理されないばかりでなく、コネクターはエラーを通知しません。

用語

このガイドでは、以下の用語が使用されています。

- **ASI (アプリケーション固有の情報)** 特定のアプリケーションまたはテクノロジーに合わせて作成されたメタデータ。ASI は、ビジネス・オブジェクトの属性レベルおよびビジネス・オブジェクト・レベルに存在します。*動詞 ASI* も参照してください。
- **BO (ビジネス・オブジェクト)** ビジネス・エンティティー (Employee など) およびデータへのアクション (create または update 操作など) を表す属性のセット。WebSphere Business Integration システムのコンポーネントは、ビジネス・オブジェクトを使用して情報を交換し、アクションを起動します。

- **BO (ビジネス・オブジェクト) ハンドラー**・アプリケーションと対話し、要求ビジネス・オブジェクトをアプリケーションのオペレーションに変換するメソッドを格納するコネクター・コンポーネント。
- **外部キー** 依存する属性のマッピングを定義するビジネス・オブジェクト属性。
- **ODA (Object Discovery Agent)** アプリケーション内部の指定されたエンティティーを調べ、ビジネス・オブジェクト属性に対応するこれらのエンティティーの要素を「検出」することによって、自動的にビジネス・オブジェクト定義を生成するツール。アダプターをインストールすると、ODA も自動的にインストールされます。Business Object Designer Express は、ODA にアクセスして、ODA で対話式に作業を行うためのグラフィカル・ユーザー・インターフェースとなります。
- **動詞 ASI (アプリケーション固有の情報)** 指定された動詞について、その動詞がアクティブであるときにコネクターがビジネス・オブジェクトを処理する方法を指定する。動詞 ASI には、現在の要求ビジネス・オブジェクトを処理するために呼び出すメソッドの名前が含まれています。

コネクター・アーキテクチャー

Connector for SAP は Java で記述され、Vision コネクター・フレームワークとコネクター・モジュール (コネクターのアプリケーション固有コンポーネント、コネクター・フレームワーク、およびビジネス・オブジェクト・ハンドラー) の 2 つの部分から構成されています。 Vision コネクター・フレームワークは、すべての WebSphere Business Integration システム・アダプターにより使用されるコネクター・フレームワークに対して、メタデータ主導型の抽象化レイヤーを提供します。

Vision コネクター・フレームワークは、アダプター・フレームワークのメソッドを拡張します。コネクター・モジュールは、Vision コネクター・フレームワークのメソッドを拡張し、SAP アプリケーションと通信します。

xx ページの図 2 に、コネクターのアーキテクチャーを示すとともに、アダプター・フレームワークと Vision コネクター・フレームワークの関係を示します。visionConnectorAgent クラスには、任意の数のコネクター・モジュールを実装できます。

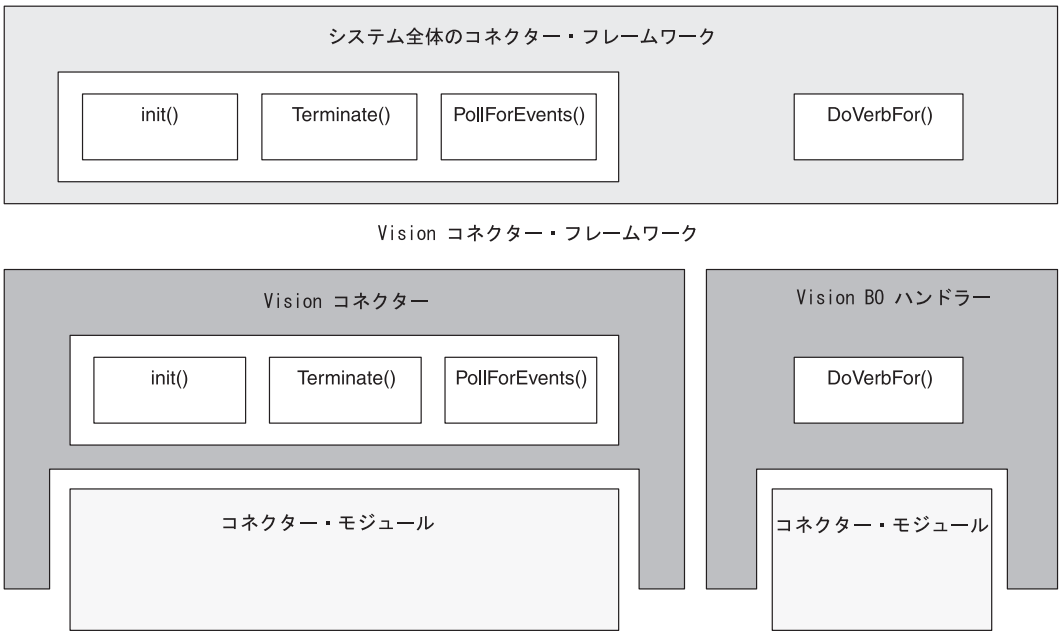


図 2. Connector for SAP のアーキテクチャー

Vision コネクター・フレームワーク

Vision コネクター・フレームワークは、初期化、ポーリング、および終了要求を、コネクター・モジュールに対して動的にルーティングします。また、ビジネス・オブジェクトをビジネス・オブジェクト・ハンドラーに対して動的にルーティングします。ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクトをサポートするために特別に設計されたコネクター・モジュールです。コネクターでは、要求やビジネス・オブジェクトを動的にルーティングするために、ビジネス・オブジェクトの動詞のアプリケーション固有情報と、特定のアプリケーション固有コネクター構成プロパティーの値を使用します。

Vision コネクター・フレームワークは、visionConnectorAgent および visionBOHandler の 2 つのクラスから構成されます。

xxi ページの図 3 に、Vision コネクター・フレームワークおよびコネクター・モジュールとの関連を示します。

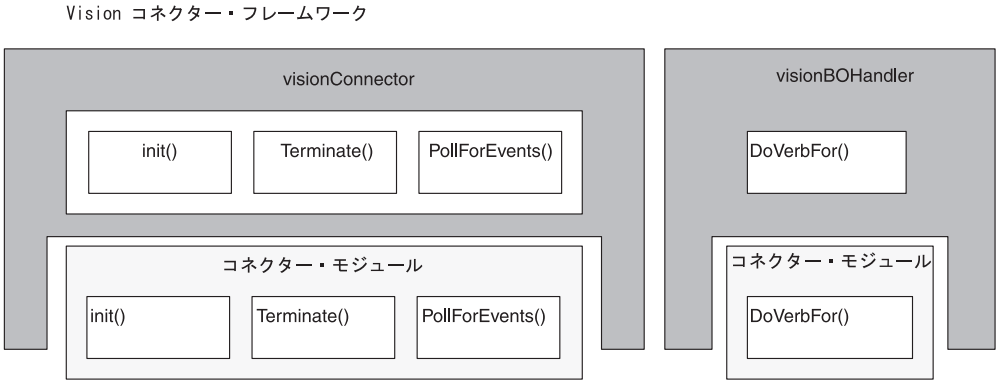


図 3. Vision コネクター・フレームワークとコネクター・モジュール

Vision コネクター・フレームワークは、コネクターに次の機能を提供します。

- 実装された `init()`、`pollForEvents()`、および `terminate()` メソッドのいずれかを呼び出す機能。
- ビジネス・オブジェクトの動詞のアプリケーション固有情報に基づいて、ビジネス・オブジェクトを特定のビジネス・オブジェクト・ハンドラーにルーティングする機能。

コネクター・モジュール

コネクター・モジュールは、Vision コネクター・フレームワークのメソッドを拡張する Java クラスです。これらは、SAP アプリケーションへのログイン、イベントおよびビジネス・オブジェクトの処理、および SAP アプリケーションへの接続の終了などの特定の機能を提供することによって、Vision コネクター・フレームワークをサポートします。コネクター・モジュールは、Vision コネクター・フレームワークと SAP アプリケーションとの間の要求を実行します。デフォルトでは、Vision コネクター・フレームワークはコネクター・モジュールのルート・ディレクトリーとして `connectors¥SAP` ディレクトリーを使用します。

コネクター・モジュールでは、一部のフレームワーク・メソッドが使用されない場合があります。例えば、あるモジュールが `init()` メソッドと `terminate()` メソッドを使用し、別のモジュールが `pollForEvents()` メソッドのみを使用するという状況も考えられます。ただし、visionConnectorAgent クラスと visionBOHandler クラスのすべてのメソッドは、どのコネクター・モジュールに対しても実装されている必要があります。コネクターが使用しないメソッドは、ダミー・メソッド、つまり存在するという以外には何もしないメソッドとして実装されている必要があります。

Vision コネクター・フレームワークの動作方法

コネクターは、コネクター・モジュールを使用して SAP アプリケーションと対話します。コネクター・モジュールは SAP のネイティブ・インターフェースに対して呼び出しを行い、SAP アプリケーションとの間でデータ (ビジネス・オブジェクトまたはイベント・データ) の受け渡しを行います。コネクターは柔軟に設計されているため、SAP アプリケーション用コネクターの初期化や、ビジネス・オブジェクト・データの受け渡しなどの異なるタスクに対して、異なるモジュールを使用できます。

コネクターと SAP アプリケーションの間の通信

コネクターでは、SAP の RFC (Remote Function Call) ライブラリーを使用して、SAP アプリケーションと通信します。SAP の RFC API を使用することで、外部プログラムから SAP アプリケーション内の ABAP 機能モジュールを呼び出すことができます。

ビジネス・オブジェクトの処理

コネクターは、メタデータ主導型です。**メタデータ** (WebSphere Business Integration システム内の場合) は、ビジネス・オブジェクトに格納されるアプリケーション固有データであり、コネクター・モジュールとアプリケーションとの対話を支援します。メタデータ主導型コネクター・モジュールは、サポートする各ビジネス・オブジェクトを処理する際に、コネクター・モジュール内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義にエンコードされたメタデータに基づいて処理を行います。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、その属性プロパティーの設定、およびアプリケーション固有情報の内容が含まれます。コネクター・モジュールはメタデータ主導型なので、新規ビジネス・オブジェクトでも変更されたビジネス・オブジェクトでも、コネクター・モジュール・コードの変更なしに処理できます。

Vision コネクター・フレームワークでは、トップレベルのビジネス・オブジェクトの動詞のアプリケーション固有情報の値を使用して、ビジネス・オブジェクトを処理するために適切なコネクター・モジュールを呼び出します。動詞のアプリケーション固有情報は、コネクター・モジュールのクラス名を提供します。

ほとんどのトップレベル・ビジネス・オブジェクトの動詞のアプリケーション固有情報では、コネクター・モジュールのクラス名を示す必要があります。この動詞のアプリケーション固有情報の構文は、次のとおりです。

```
AppSpecificInfo = PartialPackageName.ClassName,
```

例えば、次のようにします。

```
AppSpecificInfo = sap.sapextensionmodule.VSapBOHandler,
```

この例では、`sap.sapextensionmodule` は部分パッケージ名であり、`VSapBOHandler` はクラス名です。完全パッケージ名には `com.crossworlds.connectors` 接頭部が含まれます。これは、WebSphere Business Integration システムによってパッケージ名に自動的に追加されます。つまり、この例の完全なテキストは次のようになります。

```
com.crossworlds.connectors.sap.sapextensionmodule.VSapBOHandler
```

注: ほとんどのトップレベル・ビジネス・オブジェクトの動詞のアプリケーション固有情報では、コネクターのクラス名の後に、コンマ (**,**) 区切り文字を入れる必要があります。ただし、RFC Server Module で使用される Server 動詞は、セミコロン (**;**) で区切られます。サーバー動詞については、161 ページの『RFC Server Module の動作方法』および 170 ページの『サポートされる動詞』を参照してください。

ビジネス・オブジェクトが次のように使用される場合は、動詞のアプリケーション固有情報にパッケージ名およびクラス名を指定する必要はありません。

- アプリケーション・イベントを処理するために、ALE Module によって使用される場合。ただし、サービス呼び出し要求を処理するために ALE Module を使用している場合は、パッケージ名およびクラス名を指定する必要があります。
- デフォルトのビジネス・オブジェクト・ハンドラー (`sap.sapextensionmodule.VSapBOHandler`) を使用する ABAP Extension Module によって使用される場合。

重要: RFC Server Module 向けのビジネス・オブジェクトを処理するカスタマー生成コネクター・モジュールでは、完全パッケージ名を指定する必要があります。この完全パッケージ名は、`bapi` で始まる必要があります。例えば、`bapi.client.Bapi_customer_getdetail2` となります。この例での完全パッケージ名は、`bapi.client`、クラス名は `Bapi_customer_getdetail2` です。

ほとんどのビジネス・オブジェクト処理は、それぞれのコネクター・モジュールに固有です。 ABAP Extension Module 用のビジネス・オブジェクト処理の詳細については、210 ページの『ビジネス・オブジェクトの処理』、 239 ページの『ビジネス・オブジェクト・データおよび ABAP Handler』、および 285 ページの『アーカイブ対象オブジェクトの構成』を参照してください。

ALE Module 用のアプリケーション固有動詞情報について詳しくは、xxiv ページの『イベント処理』および 154 ページの『ラッパー・ビジネス・オブジェクトがある複数の IDocs の処理』を参照してください。

同時実行可能な複数の相互作用の処理

アダプター・フレームワークは、アプリケーション・イベントを処理するスレッドと、ビジネス・オブジェクト要求を処理するスレッドを別個に作成できます。統合ブローカーから受け取った複数の要求を処理するとき、複数のスレッドを作成して、複数のビジネス・オブジェクト要求を処理することができます。例えば、InterChange Server Express では、コネクターが、複数のコラボレーションやマルチスレッド化されたコラボレーションから複数のビジネス・オブジェクト要求を受け取ることができます。

図 4 に、マルチスレッド化アーキテクチャーを示します。

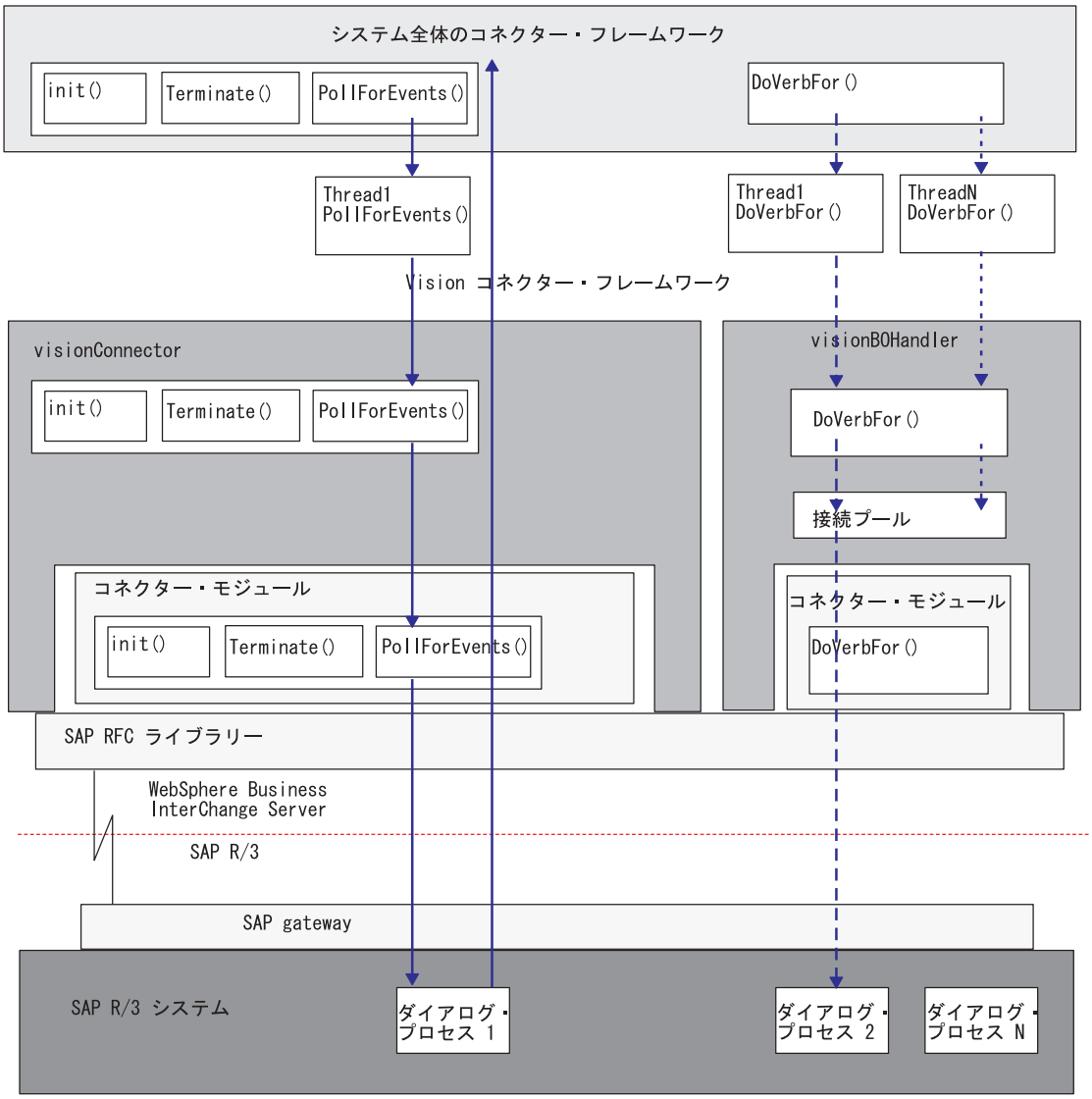


図 4. Connector for SAP のマルチスレッド化アーキテクチャー

イベント処理

コネクターでは、ポーリング呼び出しを処理する際に、以下に示すステップが実行されます。

1. アダプター・フレームワークは、ポーリング呼び出しを処理するため専用のスレッドを 1 つ作成します。このスレッドは、Vision コネクター・フレームワークの `pollForEvents()` メソッドを、`PollFrequency` 構成プロパティーに指定された頻度で呼び出します。
2. スレッドは SAP をポーリングします。SAP では、ダイアログ・プロセスを使用してイベントを位置指定して戻します。

   注: コネクターの `MaxNumberOfConnections` 構成プロパティーが 1 より大きい数値として評価された場合、Vision コネクター・フレームワークは、1 つの

接続を SAP のポーリング専用に使用します。MaxNumberOfConnections の値が 1 であれば、イベント処理とサービス呼び出し要求処理は、SAP との 1 つの接続を共有して実行されます。

ポーリング・スレッドは、コネクターがシャットダウンされるときにのみ終了します。

注: RFC Server コネクター・エージェントは、イベントをポーリングするのではなくイベントを SAP からプッシュするため、コネクター・フレームワークによって作成されたスレッドを使用する代わりにそれ自身でスレッドを作成します。ALE コネクター・エージェントは、RFC Server コネクター・エージェントを使用してイベントにアクセスするため、イベントを処理する際には、コネクター・フレームワークによって作成されたスレッドを使用する代わりにそれ自身でスレッドを作成します。

要求処理

アダプター・フレームワークは、ポーリング用スレッドとは別に、要求処理用の複数のスレッド (要求ビジネス・オブジェクト 1 つに 1 つのスレッド) を作成することができます。それぞれの要求スレッドは、次に示す場合に応じて、適切なビジネス・オブジェクト・ハンドラーのインスタンスを生成します。

例えば、InterChange Server Express からのビジネス・オブジェクト要求を処理するとき、ビジネス・オブジェクト・ハンドラーの数とタイプは、要求の送り元コラボレーションの数とタイプにより決まります。

- 複数のコラボレーションからビジネス・オブジェクトが送信された場合、各要求スレッドは、適切なタイプのビジネス・オブジェクト・ハンドラーのインスタンスを生成します。
- マルチスレッド化された単一のコラボレーションから同じタイプの複数のビジネス・オブジェクトが送信された場合、各要求スレッドは、そのタイプのビジネス・オブジェクト・ハンドラーのインスタンスを同じ数だけ生成します。

コネクターの MaxNumberOfConnections 構成プロパティーが 1 より大きい数値として評価された場合、Vision コネクター・フレームワークは、1 つの接続を SAP のポーリング専用に使用し、それ以外の接続は、要求処理専用のプールに割り振ります。

図 4 に示すように、コネクターは、ビジネス・オブジェクト要求を処理するとき、次のステップを実行します。

1. アダプター・フレームワークは、ビジネス・オブジェクト要求ごとにスレッドを作成します。各スレッドは、Vision ビジネス・オブジェクト・ハンドラーの doVerbFor() メソッドを呼び出します。
2. コネクターの MaxNumberOfConnections 構成プロパティーが 1 より大きい数値として評価された場合、Vision ビジネス・オブジェクト・ハンドラーは Vision コネクター・フレームワークの接続プールを検査して、接続ハンドルが使用可能かどうかを判断します。
   - ハンドルが使用可能な場合、スレッドは SAP に要求を送信します。SAP では、ダイアログ・プロセスを使用してその要求を処理します。

- ハンドルが使用可能でない場合、スレッドはハンドルが使用可能になるまで待機します。それぞれのビジネス・オブジェクト・ハンドラー・スレッドが使用可能な接続ハンドルを要求あるいは待機する順序は、スレッドの順序付けによって決まります。

コネクターの MaxNumberOfConnections 構成プロパティーの値が 1 であれば、Vision ビジネス・オブジェクト・ハンドラーは、イベント処理と 1 つの接続を共有します。

以下に注意してください。

3. SAP は、処理が完了し、戻りコードを送信した後、ダイアログ・プロセスを解放します。
4. コネクターは、SAP からの戻りコードを受信した後、接続ハンドルを解放します。

使用可能な接続数の設定

使用可能な接続ハンドルの最大数を指定するには、MaxNumberOfConnections 構成プロパティーを使用します。接続数は、ダイアログ・プロセスの数を超えることはできません。

SAP では、相互作用の処理中にはダイアログ・プロセスをロックし、相互作用が完了したときにのみダイアログ・プロセスを解放します。そのため、並行する複数の要求は、処理が終了するまで同じ数のダイアログ・プロセスをロックします。

重要: MaxNumberOfConnections の値を設定する前に、SAP BASIS 管理者に問い合わせて、アプリケーション・サーバーのパフォーマンスに悪影響を与えずにスループットを最大化できる適切な値を決定してください。

複数の接続のサポート

デフォルトでは、コネクターは単一スレッド化だけをサポートします。コネクターで複数のスレッドがサポートされるようにするには、コネクター始動スクリプトから次のフラグを除去してください。

- i5/OS または Linux の場合: -tMAIN_SINGLE_THREAD
- Windows の場合: -tSINGLE_THREAD

第 1 部 コネクターの概要およびセットアップ

第 1 章 コネクターのインストール

この章では、Adapter for mySAP.com のコネクター・コンポーネントのインストール方法について説明します。この章の内容は以下のとおりです。

重要: コネクターのバージョンをアップグレードする場合は、コネクターの JAR ファイル (.jar) を置換する必要があります。また、以前にインストールしたすべてのビジネス・オブジェクト・トランスポートと、コネクター・トランスポート・ファイルもアップグレードする必要があります。コネクターに加えた変更によっては、SAPConnector.txt ファイルをリポジトリーに新たにコピーすることが必要な場合があります。詳細については、『リリース情報』を参照してください。

インストール・タスク

Connector for mySAP.com をインストールするには、ご使用の環境に存在する必要なコネクターの前提条件を確認し、統合ブローカーをインストールした後で、コネクターのインストールを行う必要があります。

アダプターの前提条件の確認

アダプターをインストールする前に、コネクターをインストールし、実行するための環境の前提条件すべてがシステムに揃っていることを確認します。詳細については、xv ページの『アダプター環境』 を参照してください。

統合ブローカーのインストール

統合ブローカー InterChange Server Express のインストールには、WebSphere Business Integration システムのインストールとブローカーの開始という作業が含まれていますが、これについては、ブローカーの資料で説明しています。

コネクターのインストール

WebSphere Business Integration Server Express アダプター製品のインストールの詳細については、「*WebSphere Business Integration Server Express インストール・ガイド」(Windows 版、Linux 版、OS/400 および i5/OS 版)* を参照してください。このガイドは、WebSphere Business Integration Server Express Adapters Infocenter (http://www.ibm.com/websphere/wbiserverexpress/infocenter) にあります。

Connector for mySAP.com と関連ファイルのインストール

コネクターは、コネクターのアプリケーション固有コンポーネントと SAP の RFC ライブラリーの 2 つの部分から成り、これらをインストールする必要があります。

必要なコネクター・ファイルをインストールした後、Java コネクター (SAP JCo) ファイルをダウンロードおよびインストールすることが必要です。SAP JCo ファイルのダウンロードとインストールの詳細については、8 ページの『SAP の Java コネクター (SAPJCo) のインストール』 を参照してください。

注: インストーラーは、コネクターのアプリケーション固有コンポーネントに対応するメニュー・オプションをアダプター・メニューに追加します。コネクターをすばやく始動するには、このコンポーネントへのショートカットをデスクトップに作成してください。

リモート・マシンへのコネクターのインストール

コネクターは、リモート・マシン上にインストールして実行することができます。1 つのマシンに統合ブローカーを、別のマシンにコネクターをインストールします。両方のマシンを同じサブネット上に置くことをお勧めしますが、必須ではありません。

複数のコネクター・インスタンスの作成

コネクターの複数インスタンスの作成は、多くの点でカスタム・コネクターの作成と似ています。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。それには、以下の作業を行う必要があります。

- コネクター・インスタンスの新規ディレクトリーを作成する
- 必要なビジネス・オブジェクト定義が存在することを確認する
- 新規コネクター定義ファイルを作成する
- 新規始動スクリプトを作成する

新規ディレクトリーの作成

- **Windows プラットフォームの場合**:

  ProductDir\connectors\connectorInstance

  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は、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

  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。

  InterChange Server Express サーバー名を start_SAP.bat のパラメーターとして指定できます。例えば、start_SAP.bat connName serverName などです。

- **i5/OS プラットフォームの場合:**

  ```
  /QIBM/UserData/WBIServer44/WebShereICSName/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、WebSphereICSName はコネクターの実行に使用する Interchange Server Express インスタンスの名前です。

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は、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。

/QIBM/UserData/WBIServer44/WebSphereICSName/repository
/connectorInstance。ここで WebSphereICSName はコネクターの実行に使用する Interchange Server Express
インスタンスの名前です。

- **Linux プラットフォームの場合**:
  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は、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、ディレクトリー ProductDir/repository/connectorInstance を作成し、ファイルをここに格納します。InterChange Server Express のサーバー名を connector_manager のパラメーターとして指定することができます。例: connector_manager -start connName WebSphereICSName [-cConfigFile]

ビジネス・オブジェクト定義の作成

プロジェクト内にコネクター・インスタンスごとのビジネス・オブジェクト定義が存在しない場合は、ビジネス・オブジェクト定義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 Express 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

次の手順を使用して、Connector Configurator Express 内のコネクター・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。

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、 5 ページの『ビジネス・オブジェクト定義の作成』で作成したコネクター・ディレクトリーに格納します。
3. (Windows の場合のみ) 始動スクリプトのショートカットを作成します。
4. (Windows の場合のみ) 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。
5. (i5/OS の場合のみ) 次の情報を使用して、コネクターのジョブ記述を作成します。
   CRTDUPOBJ(QWBISAP) FROMLIB(QWBISVR44)OBJTYPE(*JOBD)TOLIB (QWBISVR44) NEWOBJ(newemailname)。ここで、newemailname は新規コネクターのジョブ記述で使用する 10 文字の名前です。
6. (i5/OS の場合のみ) 新規コネクターを WebSphere Business Integration Server Express Console に追加します。WebSphere Business Integration Server Express Console の詳細については、コンソールに付属のオンライン・ヘルプを参照してください。

インストールの検証

インストーラーは、コネクターに関連付けられた標準ファイルをご使用のシステムにコピーします。コネクターは、*ProductDir*¥connectors¥SAP ディレクトリーにインストールされます。

ProductDir は、コネクターのインストール先ディレクトリーを表します。ディレクトリー・パスの規則として円記号 (¥) を使用します。Linux のインストールの場合には、円記号 (¥) をスラッシュ (/) に置き換えてください。ファイルのパス名はすべて、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

インストール後のファイル構造の確認: Linux 環境

表 1 に、Linux 環境でコネクターによってインストールされるファイルを示します。インストールした後、以下に示すすべてのファイルがマシン上の正しいディレクトリーにコピーされていることを確認してください。

表 1. アダプター: Linux のファイル構造

ディレクトリー/ファイル名	説明
connectors/SAP/bapi/client	BAPI Module ビジネス・オブジェクト・ハンドラー・ファイルが格納されているディレクトリー。
connectors/SAP/bapi/server	RFC Server Module ビジネス・オブジェクト・ハンドラー・ファイルが格納されているディレクトリー。
connectors/SAP/dependencies	すべてのバージョン固有トランスポート・ファイルが格納されているディレクトリー。
connectors/connector for mySAP.commessages	connector for mySAP.commessages ファイルが格納されているディレクトリー。
connectors/SAP/samples	サンプルの ABAP オブジェクトが格納されているディレクトリー。
connectors/SAP/CWSAP.jar	コネクターのクラス・ファイル。

表 1. アダプター: Linux のファイル構造 (続き)

ディレクトリー/ファイル名	説明
connectors/SAP/start_SAP.sh	コネクターのシステム始動スクリプト。 このスクリプトは、汎用のコネクター・マネージャー・スクリプトから呼び出されます。製品インストーラーは、このコネクター・マネージャー・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。 InterChange Server Express でコネクターが動作している場合は、このカスタマイズされたラッパーを使用してコネクターを始動および停止してください。
repository/SAP	sap_idoccontrol.xsd ファイルが格納されているディレクトリー
/lib	WBIA.jar ファイルが格納されています。
/bin	CWConnEnv.shファイルが格納されています。
/bin/Data/app	SAPConnectorTemplate ファイルが格納されています。

コネクターを使用するためには、インストーラーの「Connector Configuration」画面からコネクターを構成することが必要です。この画面から、次の操作を実行します。

- 「コネクター名を選択 (Select Connector Name)」リストから「SAP」を選択します。
- 「構成」をクリックしてコネクターを構成します。

インストール後のファイル構造の確認: i5/OS 環境

表 2 に、i5/OS 環境でコネクターによってインストールされるファイルを示します。インストールした後、以下に示すすべてのファイルがマシン上の正しいディレクトリーにコピーされていることを確認してください。

表 2. アダプター: i5/OS のファイル構造

ディレクトリー/ファイル名	説明
connectors/SAP/bapi/client	BAPI Module ビジネス・オブジェクト・ハンドラー・ファイルが格納されているディレクトリー。
connectors/SAP/bapi/server	RFC Server Module ビジネス・オブジェクト・ハンドラー・ファイルが格納されているディレクトリー。
connectors/SAP/dependencies	すべてのバージョン固有トランスポート・ファイルが格納されているディレクトリー。
connectors/messages	SAPConnector.txt ファイルが格納されているディレクトリー。
connectors/SAP/CWSAP.jar	コネクターのクラス・ファイル。
connectors/SAP/start_SAP.sh	コネクターを始動するために使用されるバッチ・ファイル。
repository/SAP	CN_SAP.txt ファイルが格納されているディレクトリー。
/lib	WBIA.jar ファイルが格納されています。
/bin	CWConnEvn.sh ファイルが格納されています。

インストール後のファイル構造の確認: Windows 環境

8 ページの表 3 に、Windows 環境でコネクターによってインストールされるファイルを示します。インストールした後、以下に示すすべてのファイルがマシン上の正しいディレクトリーにコピーされていることを確認してください。

表 3. アダプター: Windows のファイル構造

ディレクトリー/ファイル名	説明
connectors¥SAP¥bapi¥client	BAPI Module ビジネス・オブジェクト・ハンドラー・ファイルが格納されているディレクトリー。
connectors¥SAP¥bapi¥server	RFC Server Module ビジネス・オブジェクト・ハンドラー・ファイルが格納されているディレクトリー。
connectors¥SAP¥dependencies	すべてのバージョン固有トランスポート・ファイルが格納されているディレクトリー。
connectors¥connector for mySAP.commessages	connector for mySAP.commessages ファイルが格納されているディレクトリー
connectors¥SAP¥samples	サンプルの ABAP オブジェクトが格納されているディレクトリー。
connectors¥SAP¥CWSAP.jar	コネクターのクラス・ファイル。
connectors¥SAP¥start_SAP.bat	コネクターを始動するために使用されるバッチ・ファイル。
repository¥SAP	CN_SAP.txt ファイルが格納されているディレクトリー。
¥lib	WBIA.jar ファイルが格納されています。
¥bin	CWConnEvn.bat ファイルが格納されています。

SAP の Java コネクター (SAPJCo) のインストール

コネクターをインストールし、すべてのファイルが適切なディレクトリーにインストールされていることを確認したら、SAP JavaAPI をダウンロードしてインストールする必要があります。これは SAPODA の前提条件です。これについては 35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』で説明します。

SAP の Java API は、Java コネクター (SAP JCo) と呼ばれています。現在、Connector for SAP は SAP JCo V.2.1.3 および V.2.1.4 をサポートしています。

1. コネクターを稼働するオペレーティング・システムの SAP JCo をダウンロードします。SAP JCo は、SAP の Web サイト、http://service.sap.com/connectors からダウンロードできます。SAP JCo にアクセスするには、SAPNet アカウントが必要です (アカウントがない場合は、ローカルの SAP BASIS 管理者にご連絡ください)。

   コネクターがサポートするバージョンの SAP JCo を SAP サービス・マーケットプレイスからダウンロードできない場合は、最新のアダプター・パッチについての注記を調べて、サポートされる JCo の最新バージョンを確認するか、IBM 担当員にお問い合わせください。
2. unzip した以下の SAP JCo ファイルをご使用の環境にコピーします。

   Linux:

   ZIP ファイルから、実行可能な JAR ファイル (sapjco.jar) とランタイム・ライブラリー (librfccm および libsapjcorfc) を抽出します。

   SAPODA をインストールするマシンに、既にアダプターが指示どおりにインストールされている場合には、これらのファイルを \connectors\SAP ディレクトリーから \ODA\SAP ディレクトリーにコピーしてください。SAPODA をコネク

ターとは別のマシンにインストールする場合には、SAP JCo ファイルを解凍した後、これら 3 つのファイルを \ODA\SAP ディレクトリーにコピーしてください。

i5/OS:

ZIP ファイルから、実行可能な JAR ファイル (拡張子 .jar) およびランタイム・ライブラリーを抽出します。次に、以下の構成を適用します。

a. 環境変数 LIBPATH を追加し、値として sapjco インストール・パスを指定します。
b. sapjco.jar のパスを環境変数 CLASSPATH の値に追加します。
c. 環境変数 QIBM_JAVA_PASE_STARTUP (値は「/usr/lib/start64」) を追加します。

Windows:

ZIP ファイルから、実行可能な JAR ファイル (拡張子 .jar) およびランタイム・ライブラリー (拡張子 .dll) を抽出します。SAPODA をインストールするマシンに、既にアダプターが指示どおりにインストールされている場合には、これらのファイルを ¥connectors¥SAP ディレクトリーから ¥ODA¥SAP ディレクトリーにコピーしてください。

SAPODA をコネクターとは別のマシンにインストールする場合には、SAP JCo ファイルを解凍した後、これら 3 つのファイル (librfc32.dll、sapjco.jar、sapjcorfc.dll) を ¥ODA¥SAP ディレクトリーにコピーしてください。Windows の場合、librfc32.dll には 1 つ以上の C ランタイム dll が必要です。C ランタイム dll は、使用する SAP リリースのバージョンによって異なります。

注: SAP リリース 45B までの場合、必要な C ランタイム dll は、msvcrt.dll バージョン 5.00.7022 以降です。SAP リリース 46A 以降の場合、必要な C ランタイム dll は、msvcrt.dll バージョン 6.00.8267.0 以降および msvcp60.dll バージョン 6.00.8168.0 以降です。これらの dll は、C:¥WINNT¥system32 ディレクトリーにコピーする必要があります。これらの dll は、既に存在している場合があります。まだ存在していない場合でも、これらの dll は「Presentation CD」に収録されています。この CD の *<cddrive>*:¥GUI¥Windows¥Win32¥system フォルダーには、Windows SAPGUI セットアップが収録されています。詳細については、「SAP OSS note number 0182805」を参照してください。

ALE Module の TID の管理のためのコネクターのアップグレード

ALE Module は、SAP アプリケーションから受け取ったすべてのトランザクションの IDoc オブジェクトとトランザクション ID (TID) を永続的に保管します。バージョン 4.8.x 以前のコネクターのリリースでは、コネクターは IBM WebSphere コラボレーション、ビジネス・オブジェクトおよびマップを使用してデータをリポジトリーに格納していました。

重要: ALE Module が SAP アプリケーションとの間でやりとりされる IDoc を処理できるようにするには、コネクターをアップグレードする必要があります。ただし、コネクターをアップグレードする前に、現在の IDoc の処理サイクルを最後まで実行できるようにすることが不可欠です。

コネクターをアップグレードして、ALE Module が IDoc を処理できるようにする前に、イベント・ディレクトリーおよび WIP ディレクトリー内の現在のファイルの処理を完了しておく必要があります。また、アーカイブ・ディレクトリーを調べて、失敗したイベントやアンサブスクライブされたイベントがないか確認してください。

イベント・ディレクトリーおよび WIP ディレクトリー内の現在のファイルの処理を完了するには、次の作業を行います。

- コネクターからの IDoc の送信とコネクターへの IDoc の送信の両方を一時停止します。
- アップグレード時に、以下のディレクトリー内に IDoc (ファイル) がないことを確認します。

Linux

```
$CROSSWORLDS/connectors/SAP/ale/events
$CROSSWORLDS/connectors/SAP/ale/wip
```

Windows

```
%CROSSWORLDS%¥connectors¥SAP¥ale¥events
%CROSSWORLDS%¥connectors¥SAP¥ale¥wip
```

失敗したイベントやアンサブスクライブされたイベントがあった場合、この処理を完了するには、次の作業を行います。

- コネクターからの IDoc の送信とコネクターへの IDoc の送信の両方を一時停止します。
- アップグレード時に、以下のディレクトリー内の IDoc (ファイル) の状況を確認します。

Linux

```
$CROSSWORLDS/connectors/SAP/ale/archive
```

Windows

```
%CROSSWORLDS%¥connectors¥SAP¥ale¥archive
```

- 失敗したイベントまたはアンサブスクライブされたイベントに対するエラーを訂正します。

- 訂正したファイルを処理するよう、イベント・ディレクトリーに移動します。

注: トランザクション SM58 の使用時に SAP システムで正常に処理されない IDoc が見つかった場合、コネクターがアップグレードされるまで待って、その IDoc を再サブミットしてください。コネクターのアップグレードが完了したら、エラーを訂正して IDoc を再サブミットし、新しい TID 管理で処理されるようにしてください。

ディレクトリーに処理すべきものがなくなったら、アップグレードを適用して、以下のセクションにある構成手順に従ってください。

- 341 ページの『コネクター固有の構成プロパティー』
- 117 ページの『第 9 章 ALE Module の概要』
- 125 ページの『第 10 章 ALE Module の使用』

第 2 章 コネクターの構成

この章では、Connector Configurator Express を使用して、アダプターをインストールし、構成する方法について説明します。

Connector Configurator Express の概要

Connector Configurator Express では、InterChange Server Express 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。

Connector Configurator Express 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**を作成します。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに 1 つ構成ファイルを作成する必要があります。
- 構成ファイルのプロパティーを設定します。
  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、InterChange Server Express 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。

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で使用されるので、最初から定義する必要はありません。構成ファイルを作成すると、Connector Configurator Express によって標準プロパティーがそのファイルに挿入されます。ただし、Connector Configurator Express で各標準プロパティーの値を設定する必要があります。

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、15 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。

Connector Configurator Express の始動

Connector Configurator Express は、以下の 2 種類のモードで始動し、実行することができます。

- スタンドアロン・モードで個別に実行
- System Manager から実行

スタンドアロン・モードでの Connector Configurator Express の実行

System Manager を実行せずに Connector Configurator Express を実行して、コネクター構成ファイルを編集することができます。

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 Business Integration Server Express**」>「**Toolset Express**」>「**開発**」>「**Connector Configurator Express**」をクリックします。
- 「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。
- 「**システム接続 Integration Broker**」の隣のプルダウン・メニューをクリックして、InterChange Server Express を選択します。

Connector Configurator Express 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方法が便利です (20 ページの『構成ファイルの完成』を参照)。

System Manager からの Connector Configurator Express の実行

System Manager から Connector Configurator Express を実行できます。

Connector Configurator Express 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 Express**」をクリックします。「Connector Configurator Express」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て、InterChange Server Express を選択します。

既存の構成ファイルを編集するには、以下のステップを実行します。

- 「System Manager」ウィンドウの「コネクター」フォルダーで構成ファイルを選択し、右クリックします。Connector Configurator Express が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
- Connector Configurator Express で「**ファイル**」>「**開く**」を選択します。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクター構成ファイルを選択します。
- 「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、または既存のコネクター定義をテンプレートとして使用します。

- テンプレートの新規作成については、『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。既存のテンプレートは ¥WebSphereAdapters¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティーを作成し、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。

Connector Configurator Express では、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 (Connector-Specific Property Template)**」とクリックします。
2. 「**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。
   - 「**新規テンプレート名を入力してください**」の下の「**名前**」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - テンプレートに含まれているコネクタ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。
3. テンプレートを作成するときには、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。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。
   - 既存のテンプレートを変更する場合には、「**変更する既存のテンプレートを選択してください: 検索テンプレート**」の下の「**テンプレート名**」テーブルのリストから、テンプレート名を選択します。
   - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

「**次へ**」をクリックしてテンプレートを選択すると、「**プロパティー: 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
  プロパティー・サブタイプ
  更新されたメソッド
  説明

- **フラグ**
  標準のフラグ
- **カスタム・フラグ**
  フラグ

「**プロパティー・サブタイプ**」 は、「**プロパティー・タイプ**」が String のときに選択することができます。これは、構成ファイルの保管時に構文検査を行うオプションの値です。デフォルトはブランク・スペースで、プロパティーがサブタイプ付きでなかったことを示します。

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。また、編集可能な値も設定できます。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**」のフィールドに値を入力します。

新規プロパティー値を作成するには、以下のステップを実行します。

1. 「**プロパティーを編集**」リストからプロパティーを選択し、右クリックします。
2. ダイアログ・ボックスから、「**追加**」を選択します。
3. 新規プロパティー値の名前を入力し、「OK」をクリックします。右側の「**値**」パネルに値が表示されます。

「**値**」パネルには、3 つの列からなるテーブルが表示されます。

「**値**」の列には、「**プロパティー値**」ダイアログ・ボックスで入力した値と、作成した以前の値が表示されます。

「**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**」表示で強調表示された依存プロパティーで、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 (Finish)**」をクリックします。入力した情報が、Connector Configurator Express によって、Connector Configurator Express 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 XML 文書として保管されます。

パス名の設定

パス名を設定するには、以下のようないくつかの一般規則があります。

- Windows および Linux でのファイル名の最大長は 255 文字です。
- Windows では、絶対パス名は [Drive:][Directory]¥filename の形式 (例: C:¥WebSphereAdapters¥bin¥Data¥Std¥StdConnProps.xml) に従っている必要があります。
  Linux では、先頭文字は必ず / にします。
- キュー名の先頭にはスペースを指定できません。また、スペースを埋め込むことはできません。

新しい構成ファイルを作成

構成ファイルを新規に作成するには、構成ファイルの名前を指定し、統合ブローカーを選択する必要があります。

- System Manager ウィンドウで、「**コネクター**」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 Express 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。

- スタンドアロン・モード: Connector Configurator Express から、「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。「新規コネクター」ウィンドウで、新規コネクターの名前を入力します。

また、統合ブローカーも選択する必要があります。ブローカーを選択するには、以下のステップを実行します。

- 「**Integration Broker**」フィールドで、InterChange Server Express 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。

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、その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と一貫性をもつ一意の名前である必要があります。

     重要: Connector Configurator Express 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**

     InterChange Server Express 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**

     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 が実行中である必要があります。
   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 Express 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 Express」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、SAP コネクターの場合は CN_SAP.txt です)。
- InterChange Server Express リポジトリー・ファイル。
  以前にコネクターの InterChange Server Express インプリメンテーションの際に使用された定義が、そのコネクターの構成に使用されたリポジトリー・ファイルに残されていることがあります。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 Express でそのファイルを開き、構成を修正してから、再度保管する必要があります。

ディレクトリーから *.txt, *.cfg、または *.in ファイルを開くには、以下のステップを実行します。

1. Connector Configurator Express 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)
   - InterChange Server Express リポジトリー (*.in、*.out) (InterChange Server Express Repository (*.in, *.out))

     InterChange Server Express 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 Express を始動します。
3. 「**ファイル**」>「**開く**」>「**プロジェクトから**」とクリックします。

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 Express」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。

ツールバーには、「**ターゲット・システム**」というドロップ・リストがあります。これを使用して、プロパティーの拡張検証のターゲット・オペレーティング・システムを選択することができます。使用可能なオプションには、「Windows」、「Linux」、「その他」 (Windows でも Linux でもない場合)、「なし (拡張検証なし)」(拡張検証をオフに切り替え) があります。始動時のデフォルトは「Windows」です。

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。

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。値 InterChange Server Express (ICS) を選択します。
2. InterChange Server Express に関連付けられているコネクター・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。テーブルには、「**プロパティー名**」、「**値**」、「**タイプ**」および「**サブタイプ**」 (「タイプ」が String の場合) が表示されます。
3. ここでファイルを保管するか、または 23 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。
4. 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 Express 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 Express 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、または既存のコネクター構成ファイルを開くと、Connector Configurator Express に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 Express では、InterChange Server Express ブローカーで稼働するコネクターに対して、以下のカテゴリーのプロパティー値を入力する必要があります。

- 標準プロパティー
- 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)
- 関連付けられたマップ

重要: Connector Configurator Express 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。

「**標準のプロパティー**」と「**コネクター固有プロパティー (Connector-Specific Properties)**」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。
- プロパティーごとに「**更新メソッド**」フィールドが表示されます。変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。この設定を構成することはできません。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示される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すると、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 Express 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 Express 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きに、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。

特定の標準プロパティーに関する詳細情報を取得するには、「標準のプロパティー」タブ付きシートの中のそのプロパティーの「説明」列の項目上にマウスを移動します。全般ヘルプをインストールしている場合は、「ヘルプ」ウィンドウが開き、標準プロパティーの詳細が表示されます。

全般ヘルプ・ファイルの位置については、標準プロパティーの付録の `AdapterHelpName` プロパティーを参照してください。

コネクター固有構成プロパティーの設定

コネクター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。「**追加**」をクリックしてプロパティーを追加します。子プロパティーを追加するには、親行番号を右クリックして、「**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. 22 ページの『標準コネクター・プロパティーの設定』で説明したように、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

「コネクター固有プロパティー」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・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 A 『*コネクターの標準構成プロパティー*』の 316 ページの『構成プロパティー値の概要』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 Express 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。

ご使用のブローカーが InterChange Server Express の場合

ビジネス・オブジェクト定義がコネクターでサポートされることを指定する場合や、既存のビジネス・オブジェクト定義のサポート設定を変更する場合は、「**サポートされているビジネス・オブジェクト**」タブをクリックし、以下のフィールドを使用してください。

ビジネス・オブジェクト名: ビジネス・オブジェクト定義がコネクターによってサポートされることを指定するには、System Manager を実行し、以下の手順を実行します。

1. 「**ビジネス・オブジェクト名**」リストの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・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 Express」ウィンドウの「ファイル」メニューで、「**プロジェクトに保管**」をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

サポートされるリストからビジネス・オブジェクトを削除する場合は、以下の手順を実行します。

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator Express」ウィンドウの「**編集**」メニューから、「**行を削除**」をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. 「**ファイル**」メニューから、「**プロジェクトに保管**」をクリックします。

サポートされるリストからビジネス・オブジェクトを削除すると、コネクター定義が変更され、削除されたビジネス・オブジェクトは、コネクターのこのインプリメンテーションで使用不可になります。コネクターのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

エージェント・サポート: ビジネス・オブジェクトにエージェント・サポートがある場合、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 Express」ウィンドウでは、「エージェント・サポート」を選択しても問題ないかどうかの検証は行われません。

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。

関連付けられたマップ (InterChange Server Express)

各コネクターは、ビジネス・オブジェクト定義とそれらに関連付けられたマップのうち現在 InterChange Server Express でアクティブであるものを示す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 Express」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。InterChange Server Express は、ブート時、各コネクターのサポートされるビジネス・オブジェクトのそれぞれ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。

以下の手順を実行して、マップを明示的にバインドします。

1. 「**明示 (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 Express」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 InterChange Server Express 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。

セキュリティー・レベル (InterChange Server Express)

Connector Configurator Express の「**セキュリティー**」タブを使用して、メッセージにさまざまなプライバシー・レベルを設定することができます。この機能は、DeliveryTransport プロパティーが JMS に設定されている場合にのみ使用することができます。

デフォルトでは、「プライバシー」はオフになります。これを使用可能にするには、「**プライバシー**」ボックスにチェック・マークを付けてください。

「**鍵ストア・ターゲット・システムの絶対パス名**」は、以下のようになります。

- Windows の場合:
  `<ProductDir>¥connectors¥security¥<connectorname>.jks`
- Linux の場合:
  `opt/IBM/WebSphereAdapters/connectors/security/<connectorname>.jks`
- i5/OS の場合:
  `/ProductDir/connectors/security/<connectorname>.jks`

このパスおよびファイルは、Connector Configurator Express と同じシステム上になければなりません。

ターゲット・システムが現在稼働中の場合にのみ、右側にある「参照」ボタンを使用できます。このボタンは、「**プライバシー**」が使用可能に設定され、メニュー・バーの「**ターゲット・システム**」が「Windows」に設定されている場合を除いて、グレーで表示されます。

3 つのメッセージ・カテゴリー (全メッセージ、全管理メッセージ、全ビジネス・オブジェクト・メッセージ) では、「**メッセージのプライバシー・レベル**」を以下のように設定することができます。

- “” がデフォルトです。メッセージ・カテゴリーにプライバシー・レベルが設定されなかった場合に使用されます。
- none
  デフォルトと同じではありません。メッセージ・カテゴリーに故意に「なし」のプライバシー・レベルを設定する場合にこれを使用します。
- integrity
- privacy
- integrity_plus_privacy

「**鍵の保守**」機能では、サーバーおよびアダプター用の公開鍵を生成、インポート、およびエクスポートすることができます。

- 「**鍵の生成**」を選択すると、「鍵の生成」ダイアログ・ボックスが、鍵を生成する鍵ツールのデフォルト値を示した状態で表示されます。
- 鍵ストアの値は、デフォルトでは「セキュリティー」タブの「**鍵ストア・ターゲット・システムの絶対パス名**」に入力した値になります。
- 「OK」を選択すると、記入項目が検証され、鍵の証明書が生成されて、Connector Configurator Express のログ・ウィンドウに出力が送信されます。

証明書をアダプターの鍵ストアにインポートするためには、サーバーの鍵ストアからエクスポートする必要があります。「**アダプター公開鍵のエクスポート**」を選択すると、「アダプター公開鍵のエクスポート」ダイアログ・ボックスが表示されます。

- エクスポートの証明書は、デフォルトでは、ファイル拡張子が <filename>.cer であることを除いて、鍵ストアと同じ値になります。
- 「**アダプター公開鍵のインポート (Import Adapter Public Key)**」を選択すると、「アダプター公開鍵のインポート (Import Adapter Public Key)」ダイアログ・ボックスが表示されます。
- インポートの証明書は、デフォルトでは *<ProductDir>*¥bin¥ics.cer (ファイルがシステム上に存在する場合) になります。
- インポートの「証明書関連」は、サーバー名にする必要があります。サーバーが登録済みである場合は、ドロップ・リストから選択することができます。

「**アダプター・アクセス制御**」機能は、DeliveryTransport の値が IDL の場合にのみ使用可能になります。デフォルトでは、アダプターはゲスト ID でログインします。「**ゲスト ID の使用**」ボックスのチェック・マークを外すと、「**アダプター ID**」フィールドと「**アダプター・パスワード**」フィールドが使用可能になります。

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 Express は、そのファイルに含まれるロギングとトレースに関する値をデフォルト値として使用します。これらの値は、Connector Configurator Express 内で変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

1. 「**トレース/ログ・ファイル**」タブをクリックします。

2. ログとトレースのどちらでも、以下のいずれかまたは両方へのメッセージの書き込みを選択できます。
   - コンソールに (STDOUT):
     ログ・メッセージまたはトレース・メッセージを STDOUT ディスプレイに書き込みます。

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの「**トレース/ログ・ファイル**」タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、「**保管**」をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

     注: ログ・ファイルとトレース・ファイルはどちらも単純なテキスト・ファイルです。任意のファイル拡張子を使用してこれらのファイル名を設定できます。ただし、トレース・ファイルの場合、拡張子として `.trc` ではなく `.trace` を使用することをお勧めします。これは、システム内に存在する可能性がある他のファイルとの混同を避けるためです。ログ・ファイルの場合、通常使用されるファイル拡張子は `.log` および `.txt` です。

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。すべてのアダプターでデータ・ハンドラーを使用できるわけではありません。

これらのプロパティーに使用する値については、付録 A の『コネクターの標準構成プロパティー』の `ContainerManagedEvents` の下の説明を参照してください。

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 Express では、InterChange Server Express でファイルを保管します。Connector Configurator Express のタイトル・バーには常に、使用中のブローカーのモードである InterChange Server Express が表示されます。

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、`*.con` 拡張子付きファイルとして統合コンポーネント・ライブラリーに保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに `*.cfg` 拡張子付きファイルとして保管します。デフォルトでは、このファイルは `¥WebSphereAdapters¥bin¥Data¥App` に保管されます。

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 Express の使用

Connector Configurator Express はグローバル化されており、構成ファイルと統合ブローカーの間での文字変換を処理できます。Connector Configurator Express 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 Express 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、以下に太字で示される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>
```

第 3 章 コネクターの実行

この章では、Adapter for mySAP.com のコネクター・コンポーネントの始動方法および実行方法について説明します。この章の内容は以下のとおりです。

- 『コネクターの始動』
- 33 ページの『ロード・バランシングの活用』

重要: コネクターのバージョンをアップグレードする場合は、コネクターの JAR ファイル (.jar) を置換する必要があります。また、以前にインストールしたすべてのビジネス・オブジェクト・トランスポートと、コネクター・トランスポート・ファイルもアップグレードする必要があります。コネクターに加えた変更によっては、SAPConnector.txt ファイルをリポジトリーに新たにコピーすることが必要な場合があります。詳細については、『リリース情報』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開始する必要があります。Windows システムでは、始動スクリプトはコネクターのランタイム・ディレクトリー (*ProductDir*¥connectors¥*connName*(*connName* はコネクターを表す) に存在しています。

Linux システムでは、始動スクリプトは *ProductDir*/bin ディレクトリーに存在しています。

i5/OS システムでは、始動スクリプトは、コネクターが稼働する際に使用する /QIBM/UserData/WBIServer44/<instance>/connectors/<ConnInstance/ に存在しています。

表 4 が示すとおり、始動スクリプトの名前はオペレーティング・システム・プラットフォームにより異なります。

表 4. 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
Linux	connector_manager
i5/OS	start_connName.sh
Windows	start_*connName*.bat

始動スクリプトを実行すると、デフォルトで *Productdir* 内で構成ファイルが検索されるはずです (以下のコマンドを参照)。これは構成ファイルを格納する場所です。

注: アダプターが JMS トランスポートを使用している場合は、ローカル構成ファイルが必要です。

- **Windows システムでのコネクターの開始:**
  - 「**スタート**」メニューから、「**プログラム**」>「**IBM WebSphere Business Integration Server Express**」>「**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 Server Express」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。または、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。

- Windows コマンド行から、次を入力します。
  start_connName connName brokerName {-cconfigFile}
- 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、自動サービスでは Windows システムがブートするとき、手動サービスでは「Windows サービス」ウィンドウからサービスを開始するときに、コネクターが開始されます。

- **Linux システムでのコネクターの開始:**
  - コマンド行から次のように入力します。

    ```
    connector_manager -start connName brokerName [-cconfigFile ]
    ```

    ここで、*connName* はコネクターの名前で、*brokerName* はご使用の統合ブローカーを示します。
  - InterChange Server Express の場合は、*brokerName* に InterChange Server Express のインスタンスの名前を指定します。
- **i5/OS システムでのコネクターの開始:**
  - WebSphere Business Integrations Server Express Console がインストールされている Windows システムから、**「IBM WebSphere Business Integration Server Express」>「Toolset Express」>「管理」>「コンソール」**を選択します。次に、OS/400 または i5/OS システムの名前または IP アドレス、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。コネクターのリストからコネクターを選択して、「開始」をクリックします。
  - コンソールを使用してアダプターを自動的に開始するには、submit_adapter.sh スクリプトを使用します。これは、サーバーの自動開始ジョブ・エントリー内でサブシステムを使用してアダプターを開始する唯一の方法です。
  - バッチ・モードでは、i5/OS のコマンド行で、CL コマンド QSH を実行する必要があります。QSHELL 環境から、
    /QIBM/ProdData/WBIServer44/bin/submit_adapter.sh *connName WebSphereICSName pathToConnNameStartScript jobDescriptionName* を実行します。ここで、*connName* はコネクター名、*WebSphereICSName* は Interchange Server Express サーバー名 (デフォルトは QWIBDFT44)、*pathToConnNameStartScript* はコネクターの始動スクリプトの絶対パス、*jobDescriptionName* は QWIBSVR44 ライブラリーで使用するジョブ記述の名前です。
  - 対話モードでは、CL コマンド QSH を実行する必要があります。QSHELL 環境から、 */QIBM/UserData/WBIServer44/WebSphereICSName/connectors/connName/start_connName.sh connNameWebsphereICSName* [-cConfigFile] を実行します。ここで、*connName* はコネクターの名前で、*WebSphereICSName* は InterChange Server Express のインスタンスの名前です。

コマンド行の始動オプションを含む、コネクターの開始方法の詳細については、「*システム管理ガイド*」を参照してください。

ロード・バランシングの活用

ログオン時のロード・バランシングによる次の効果の結果、定義済みワークグループの効率が上がります。

- パフォーマンスの改善
- システム・リソースの消費の削減
- ワークグループのサービスおよび負荷感度についての要件に基づいて、使用可能なアプリケーション・サーバーにユーザーを配分すること

負荷平準化機能を持つコネクターを開始すると、Hostname プロパティーで指定されたメッセージ・サーバーとの通信が開始されます。そのあと、メッセージ・サーバーは、負荷が最小のアプリケーション・サーバーを探します。このアプリケーション・サーバーがいったん決定されると、メッセージ・サーバーは、今後、コネクターとのすべての RFC 通信をこの 1 つのアプリケーション・サーバーを通して経路指定します。コネクターは、メッセージ・サーバーとのダイアログ・ユーザーの 1 つと見なされます。

負荷平準化機能は、コネクターが処理するボリュームが少なく、ユーザーの数が多い、という SAP 環境の場合に最も効果があります。ボリュームが多い場合は、ご使用の中で大きい方のアプリケーション・サーバーのいずれかに直接接続することを考慮してください。

ロード・バランシングのためにコネクターを構成する方法については、次のコネクター・プロパティーの説明を参照してください。

- 346 ページの『ApplicationPassword』
- 346 ページの『ApplicationUserName』
- 346 ページの『Client』
- 346 ページの『Group』
- 347 ページの『Hostname』
- 347 ページの『InDoubtEvents』
- 349 ページの『SAPSystemID』

第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成

この章では、オブジェクト・ディスカバリー・エージェント (ODA) の一種である SAPODA について説明します。SAPODA は、Adapter for mySAP.com のビジネス・オブジェクト定義を生成します。コネクターは、IDoc タイプ、BAPI、および SAP システムに定義された RFC 対応の機能モジュール、およびビジネス・プロセスを表す SAP テーブルに基づいたオブジェクトを処理の対象としているため、SAPODA はこれらのオブジェクトを使用して、その SAP データ・ソースに固有のビジネス・オブジェクト要件を発見します。

注: IDoc タイプ、BAPI、および SAP システム内の RFC 対応の機能モジュール、および SAP テーブルについての十分な知識と経験があれば、SAPODA の動作方法を理解するために役立ちます。

重要: 属性名や動詞名などのビジネス・オブジェクト名には、U. S. English ロケール (en_US) に関連づけられたコード・セット内に定義された文字のみを使用する必要があります。詳細については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

この章の内容は以下のとおりです。

インストールと使用法

以下のセクションでは、SAPODA のインストールと使用法について説明します。

SAPODA のインストール

SAPODA をインストールするには、アダプター用のインストーラーを使用します。「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」の説明に従ってください。インストールが完了すると、次のファイルがシステムの製品ディレクトリーにインストールされます。

- ODA¥SAP¥SAPODA.jar
- ODA¥messages¥SAPODAAgent.txt
- ODA¥messages¥SAPODAAgent_*ll_TT*.txt (言語 (*_ll*) 国または地域 (*_TT*) に固有のメッセージ・ファイル)
- ODA¥SAP¥start_SAPODA.bat (Windows のみ)
- ODA/SAP/start_SAPODA.sh (Linux のみ)
- ODA/SAP/start_SAPODA.sh (iSeries)

注: 本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。Linux のインストールの場合には、円記号をスラッシュ (/) に置き換えてください。すべての製品のパス名は、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

SAPODA を使用する前に

このセクションの内容は以下のとおりです。

- 『SAPODA を実行する前に』
- 『SAPODA を使用して ALE または ABAP Extension Module の定義を作成する前に』
- 『SAPODA の使用方法』

SAPODA を実行する前に

SAPODA を実行する前に、以下の作業を行う必要があります。

- SAP システムに対する有効なログオン ID を取得する。
- SAP Java API (SAP では Java コネクター (SAP JCo) と呼ばれています) をダウンロードする。このステップは、コネクターのインストール・プロセスの一環として実行する必要があります。このステップについては、8 ページの『SAP の Java コネクター (SAPJCo) のインストール』で説明しています。

SAPODA を使用して ALE または ABAP Extension Module の定義を作成する前に

SAPODA を使用して、ABAP Extension Module および ALE Module のビジネス・オブジェクト定義を以下のような IDoc (Intermediate Document) を基にして生成できます。

- ファイルに抽出された IDoc
- SAP システム内で定義された IDoc

重要: SAPODA を使用して SAP IDoc 定義ファイルからビジネス・オブジェクト定義を生成する前に、サポートする必要のある各 IDoc タイプに対して IDoc 定義ファイルを作成する必要があります。このステップは、抽出した IDoc 定義ファイルをビジネス・オブジェクト定義のテンプレートとして使用する場合にのみ必要です。詳細については、141 ページの『IDoc 定義ファイルの作成』を参照してください。

SAPODA の使用方法

SAPODA のインストール後、ビジネス・オブジェクトを生成するには、以下の作業を行う必要があります。

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

このステップについては、次のセクションで詳しく説明します。

SAPODA の起動

次のファイルを実行すると SAPODA を起動することができます。

Linux

start_SAPODA.sh

Windows

start_SAPODA.bat

SAPODA の構成と実行には、Business Object Designer Express を使用します。Business Object Designer Express はエージェントのホストとポートを使用して ODA を見付けます。エージェント名は各スクリプトまたはバッチ・ファイルの AGENTNAME 変数で指定されます。デフォルト ODA の名前は、SAPODA です。 ODA とビジネス・オブジェクト定義および ODA の構成、始動および使用方法については、「*IBM WebSphere ビジネス・オブジェクト開発ガイド*」を参照してください。

i5/OS での ODA の起動

i5/OS で ODA を起動するには、以下の手法のいずれかを使用します。

1. WBI SE Console for i5/OS がインストールされている Windows システムから、「プログラム」>「IBM WebSphere Business Integration Console for i5/OS」>「Console for i5/OS」を選択します。次に、i5/OS システムの名前または IP アドレス、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。ODA のリストから対象の ODA を選択し、「ODA を始動」ボタンを選択します。
2. i5/OS コマンド入力から QSH CL コマンドを実行し、QSHELL 環境から次のスクリプトを実行します。
   ```
   /QIBM/ProdData/WBIServer44/bin/submit_ODA.sh
   /QIBM/ProdData/WBIServer44/AdapterCapacityPack/start_SAPODA.sh
   QWBISAPODA
   ```
3. i5/OS コマンド入力から QSH CL コマンドを実行し、QSHELL 環境から次のスクリプトを実行します。
   ```
   /QIBM/ProdData/WBIServer44/AdapterCapacityPack/ODA/SAP/Start_SAPODA.sh
   ```

i5/OS での ODA の停止

ODA の停止方法は、ODA を始動したときの方法によって決まります。『i5/OS での ODA の起動』 のステップ 1 またはステップ 2 で説明される方法のいずれかを使用して始動した場合は次のようになります。

1. CL コマンド WRKACTJOB SBS(QWBISVR44) を実行します。画面にサブシステムで実行されているすべてのジョブが表示されます。
2. リストをスクロールして、ODA のジョブ記述に一致するジョブ名を持つジョブを探し出します。SAPODA の場合は、QWBISAPODA です。
3. オプション 4 を選択し、F4 を押して ENDJOB コマンドのプロンプトを取得し、OPTION パラメーターを *IMMED に指定します。

4. Enter を押します。

start_ODAName.sh スクリプトを使用して ODA を始動した場合は、
start_ODAName.sh スクリプトを実行した場所で F3 キーを押します。

エラーおよびトレース・メッセージ・ファイルでの作業

エラーおよびトレース・メッセージ・ファイル (デフォルトは SAPODAAgent.txt) は製品ディレクトリー下の ¥ODA¥messages¥ に配置されます。これらのファイルは言語および国または地域に固有であり、以下の命名規則を使用しています。

*AgentName*Agent_ll_TT.txt

ここで、_ll は言語であり、_TT は国または地域です。

例えば、中国本国の場合のファイル名は
SAPODAAgent_zh_CN.txt となります。

台湾の場合の対応するファイル名は
SAPODAAgent_zh_TW.txt となります。

Business Object Designer Express は、メッセージ・ファイルの選択時にこの情報を使用します。デフォルトの検索順序では、最初に Business Object Designer Express が実行されているロケールに一致するロケール固有のファイルが検索されます。該当するものが見つからない場合は、Business Object Designer Express はデフォルトで English-US (en_US) バージョンとなり、最終的に Business Object Designer Express はロケールや言語の情報がないファイル名を検索します。

必須ではありませんが、ODA スクリプト・ファイルまたはバッチ・ファイルの複数のインスタンスを作成し、各インスタンスに対応する ODA に固有の名前を指定した場合には、各 ODA インスタンスに対応するメッセージ・ファイルを持つことができます。異なる名前の付いた ODA インスタンスが複数存在しても、メッセージ・ファイルは共通にすることも可能です。

有効なメッセージ・ファイルを指定には 2 つの方法があります。

- ODA の名前を変更し、それに対応するメッセージ・ファイルを作成しない場合には、ODA 構成の一部として、Business Object Designer Express でメッセージ・ファイルの名前を変更する必要があります。Business Object Designer Express はメッセージ・ファイルの名前を指定しますが、実際にファイルを作成するわけではありません。ODA 構成の一部として表示されたファイルが存在しない場合には、既存のファイルを指すように値を変更してください。
- 特定の ODA に対応する既存のメッセージ・ファイルをコピーし、必要に応じて変更することもできます。Business Object Designer Express は、各ファイルが命名規則に従って命名されることを前提としています。例えば、AGENTNAME 変数が SAPODA1 を指定する場合、Business Object Designer Express は、対応するメッセージ・ファイルの名前が SAPODA1Agent.txt であると想定します。したがって、Business Object Designer Express が確認のため ODA 構成の一部としてファイル名を提供するとき、このファイル名は ODA 名に基づいています。デフォルトのメッセージ・ファイルが正しく命名されていることを確認し、必要ならば訂正してください。

ODA のルート・ディレクトリーに存在する配置記述子の odk_dd.xml ファイルを使用する場合は、Business Object Designer Express のエージェント・プロパティーの構成のウィンドウに MessageFile プロパティーは表示されません。

注: 英語以外のロケールが必要な場合も、同じ命名規則を適用します (例: SAPODA1Agent_zh_TW.txt)。

重要: ODA の構成時にメッセージ・ファイルの名前を正しく指定できなかった場合には、ODA はメッセージなしで稼働します。メッセージ・ファイル名の指定方法の詳細については、41 ページの『初期化プロパティーの構成』を参照してください。

構成プロセスの間に、以下の項目を指定します。

- SAPODA がエラーおよびトレース情報を書き込むファイルの名前
- メッセージ・ファイルの名前
- 0 から 5 の範囲のトレース・レベル

表 5 に、トレース・レベルの値を説明します。

表 5. トレース・レベル

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

これらの値の構成方法については、41 ページの『初期化プロパティーの構成』を参照してください。

Business Object Designer Express での SAPODA の使用

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

ODA の起動後、Business Object Designer Express を起動させ、ODA を構成し、実行します。Business Object Designer Express で ODA を使用してビジネス・オブジェクト定義を生成する手順は、6 つのステップから構成されます。Business Object Designer Express は、これらのステップを順にガイドしていくウィザードを提供します。

ODA の起動後、このウィザードを起動するには、次の手順を実行します。

1. Business Object Designer Express を開きます。
2. 「ファイル」メニューから、「ODA を使用して新規作成...」サブメニューを選択します。

   Business Object Designer Express に、ウィザードの最初のウィンドウ (「エージェントの選択」という名前) が表示されます。 図 5 に、このウィンドウを示します。

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

1. 40 ページの『ODA の選択』
2. 41 ページの『初期化プロパティーの構成』
3. 43 ページの『ノードの展開およびオブジェクトの選択』
4. 46 ページの『オブジェクト選択の確認』
5. 47 ページの『定義の生成』(任意で 48 ページの『追加情報の指定』)
6. 67 ページの『定義の保管』

ODA の選択

図 5 に、Business Object Designer Express の 6 ステップのウィザードの最初のダイアログ・ボックスを示します。このウィンドウで、実行する ODA を選択します。

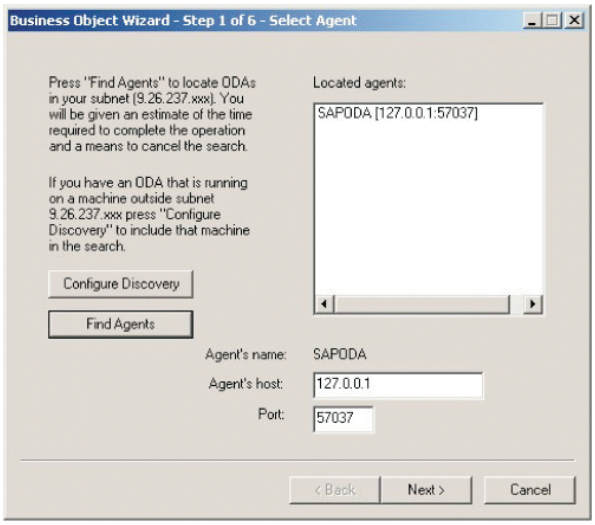


図 5. ODA の選択

ODA を選択するには、以下の手順を行います。

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

   注: Business Object Designer Express で目的の ODA が見つからない場合は、対応するフィールドにホストとポートを入力してください。
2. 表示リストから、目的の ODA を選択します。

初期化プロパティーの構成

Business Object Designer Express は、SAPODA と初めて通信する際に、図 6 に示すように、一連の初期化プロパティーの入力を要求します。これらのプロパティーは、SAPODA を使用するたびに入力せずに済むように、名前を付けたプロファイルに保存できます。ODA プロファイルの指定については、「*IBM WebSphere ビジネス・オブジェクト開発ガイド*」を参照してください。

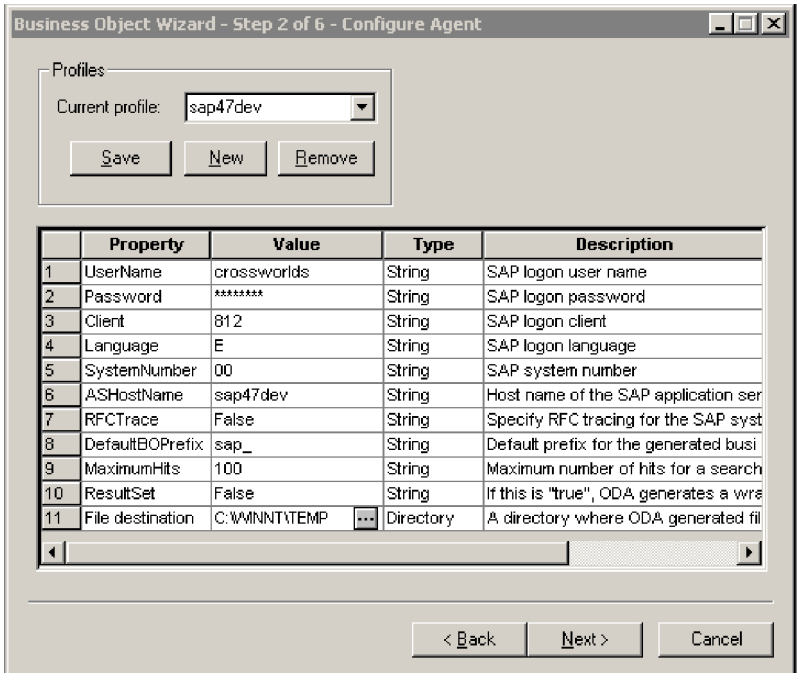


図 6. エージェント・プロパティーの構成

SAPODA プロパティーの構成を 表 6 に示します。

表 6. SAPODA のプロパティー

プロパティー名	プロパティー・タイプ	説明
UserName	String	SAP ログオン・ユーザー名 (抽出した IDoc 定義ファイルのみから定義を生成する場合は不要)
Password	String	SAP ログオン・パスワード (抽出した IDoc 定義ファイルのみから定義を生成する場合は不要)
Client	String	SAP ログオン・クライアント番号 (抽出した IDoc 定義ファイルのみから定義を生成する場合は不要)
Language	String	SAP ログオン言語 (抽出した IDoc 定義ファイルのみから定義を生成する場合は不要) デフォルトは、E で、英語に設定されます。
SystemNumber	String	SAP システム番号 (抽出した IDoc 定義ファイルのみから定義を生成する場合は不要)
ASHostName	String	SAP アプリケーション・サーバーのホスト名 (抽出した IDoc 定義ファイルのみから定義を生成する場合は不要)
RFCTrace	True/False ブール値	SAP システムの RFC トレースデフォルト値は False です。

表 6. SAPODA のプロパティー (続き)

プロパティー名	プロパティー・タイプ	説明
DefaultBOPrefix	String	ビジネス・オブジェクトの名前を固有にするために、名前の前に付加するテキスト sap_ がデフォルト値です。 Business Object Designer Express からビジネス・オブジェクト固有のプロパティーの指定を求めるプロンプトが出されたときに、必要に応じて後からこの値を変更できます。
MaximumHits	String	ノード検索時に戻されるオブジェクトの最大数。詳細については、43 ページの『ノードの展開およびオブジェクトの選択』を参照してください。 ドロップダウン・リストから選択できる指定可能な値は、100 (デフォルト) および 50 です。
TraceFileName	String	トレース・ファイルの名前。ファイルが存在しない場合、SAPODA はファイルを ¥ODA¥SAP ディレクトリーに作成します。ファイルが既に存在している場合、SAPODA はトレース情報をファイルの後に追加します。 SAPODA は次の命名規則に従ってファイルに名前を付けます。例えば、エージェントの名前が SAPODA の場合、SAPODAtrace.txt という名前のトレース・ファイルを生成します。 このプロパティーを使用して、このファイルとは異なる名前を指定します。 **注**: ODA のルート・ディレクトリーに存在する配置記述子の odk_dd.xml ファイルを使用する場合は、「エージェントの構成」画面には、このプロパティーは表示されません。
TraceLevel	整数	SAPODA に対して有効なトレースのレベル トレースの詳細については、38 ページの『エラーおよびトレース・メッセージ・ファイルでの作業』を参照してください。 **注**: ODA のルート・ディレクトリーに存在する配置記述子の odk_dd.xml ファイルを使用する場合は、「エージェントの構成」画面には、このプロパティーは表示されません。

表 6. SAPODA のプロパティー (続き)

プロパティー名	プロパティー・タイプ	説明
MessageFile	String	エラー/メッセージ・ファイルの名前。 SAPODA は次の命名規則に従ってファイルに名前を付けます。例えば、エージェントの名前が SAPODA の場合、メッセージ・ファイルに SAPODAAgent.txt という名前を付けます。詳細については、 38 ページの『エラーおよびトレース・メッセージ・ファイルでの作業』を参照してください。 **重要**: エラーおよびメッセージ・ファイルは ¥ODA¥messages ディレクトリーに配置する必要があります。 このプロパティーを使用して、既存のファイルの確認や指定をします。
File destination	Directory	**注**: ODA のルート・ディレクトリーに存在する配置記述子の odk_dd.xml ファイルを使用する場合は、「エージェントの構成」画面には、このプロパティーは表示されません。 ODA によって生成されたファイル (ビジネス・オブジェクトおよびクラス・ファイル) が格納されるディレクトリー。ビジネス・オブジェクトが生成され、このディレクトリーに格納された後、そのオブジェクトを明示的に別のディレクトリーに保管できることに注意してください。 デフォルトは、Windows のデフォルトのシステム temp ディレクトリーです。デフォルト設定を ¥connectors¥SAP¥utilities¥generatedfiles ディレクトリーに変更することを推奨します。 **注**: Business Object Designer Express と同じマシン上で SAPODA を実行する場合、File destination としてディレクトリー ODA¥SAP を使用しないでください。Business Object Designer Express は、このディレクトリーを、リモート ODA の一時保管場所として使用します。

重要: Business Object Designer Express で表示されているデフォルト値が存在しないファイルを指している場合には、メッセージ・ファイルの名前を訂正します。このダイアログ・ボックスから移動したときに、名前が不正確であった場合に、Business Object Designer Express は、ODA の起動元となったウィンドウにエラー・メッセージを表示します。このメッセージは、Business Object Designer Express ではポップアップしません。有効なメッセージ・ファイルの指定に失敗すると、ODA はメッセージなしに稼働します。詳細については、38 ページの『エラーおよびトレース・メッセージ・ファイルでの作業』を参照してください。

ノードの展開およびオブジェクトの選択

SAPODA のすべてのプロパティーの構成が完了すると、Business Object Designer Express には以下を最初のノードとするツリーが表示されます。

- IDoc タイプ — 次のことが可能です。

- 抽出した IDoc 定義ファイルのブラウズ
- SAP システムでの IDocs の選択 (基本 IDoc タイプおよび拡張タイプ)

  注: 拡張タイプはユーザー定義の IDoc タイプです。

- BOR — BAPI を表すオブジェクトを SAP アプリケーションから選択します。
- RFC — RFC 対応の機能を表すオブジェクトを SAP アプリケーションから選択します。
- 動的トランザクションおよび検索 — 動的トランザクションおよび動的検索メタデータ表からオブジェクトを表す定義を選択します。
  - HDR — Hierarchical Dynamic Retrieve モジュールによって処理される SAP トランザクション用のエンティティーを表すために必要なテーブルを選択します。

名前の前に正符号 (+) が付いたノードは、展開可能です。これらのノードをクリックすると、さらにノードまたはリーフが表示されます。SAPODA では、ビジネス・オブジェクト定義は、リーフのみから生成されます。

図 7 に、このダイアログ・ボックスの最初の状態と、いくつかのノードを展開した状態を示します。

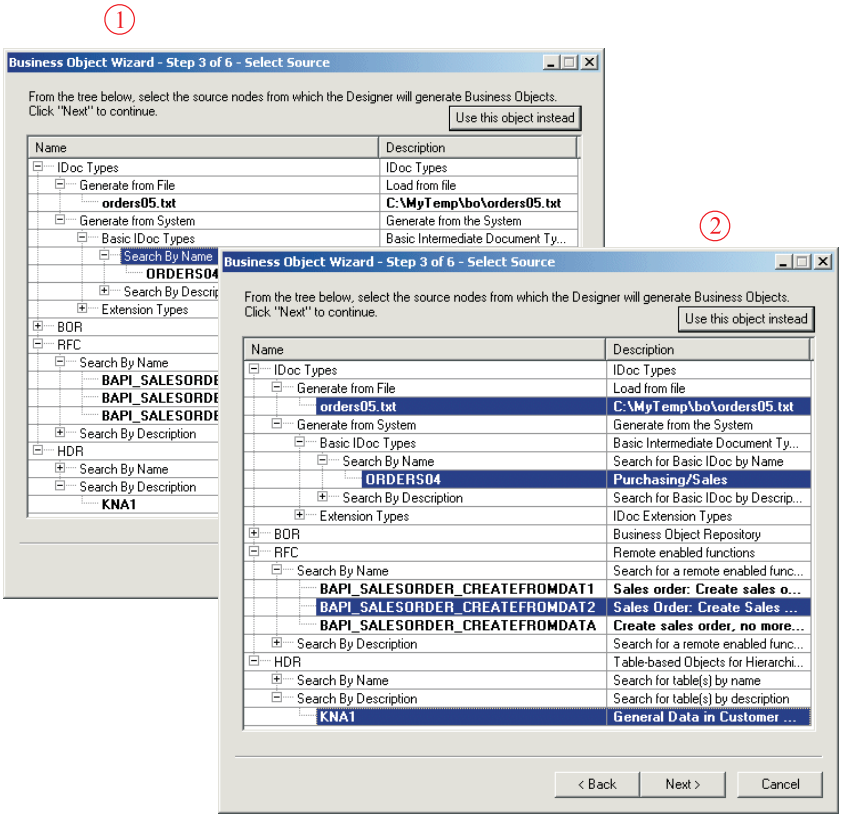


図 7. ノードを展開したツリー

リーフの名前が太字で表示された場合は、そのリーフをビジネス・オブジェクトを生成するための基礎として選択できます。複数のリーフを選択するには、Windows の標準の手順を使用します。つまり、Ctrl キーを押したまま、マウスを使用して複数のリーフを選択します。

SAPODA では、フラットなファイルをノードと関連付けることができる、ポリモアフィック・ノード・タイプが使用されます。最初に、リーフのないノードが表示されます。ファイル・システムをブラウズして、そのノードに追加するファイルを選択できます。ノードは、1 つ以上のファイルに関連付けられると性質がリーフから枝に変化するため、**ポリモアフィック**と呼ばれます。

RFC ノードを展開すると、RFC ノード内の検索結果がキャッシュされたことを示す、以下のメッセージ (図 8) が表示されます。このキャッシュ・サービスにより、検索基準に応じて、リーフ・ノードの数を減らすことができ、したがって、SAPODA が ResultSet や BAPI トランザクション・ビジネス・オブジェクトをより効率的に生成できます。検索結果はソートされ、表示されます。SAPODA の始動時にはいつも、キャッシュ・サービスはバックグラウンドで実行され、キャッシュされた検索はセッションの終了時にパージされます。キャッシュできる検索結果の数は、41 ページの図 6 に示したエージェント・プロパティーの構成のウィンドウで設定した `MaximumHits` プロパティーの値によって決まります。

図 8. キャッシュの通知

46 ページの図 9 に、Business Object Designer Express が戻すリーフの数を制限するための、次の 2 つの方法を示します。

- ファイルをブラウズするためのウィンドウを開くことのできる、コンテキスト依存メニュー。このウィンドウで、関連付けるファイルを選択できます。
- オブジェクトの名前または説明について検索する文字を指定できるウィザード。

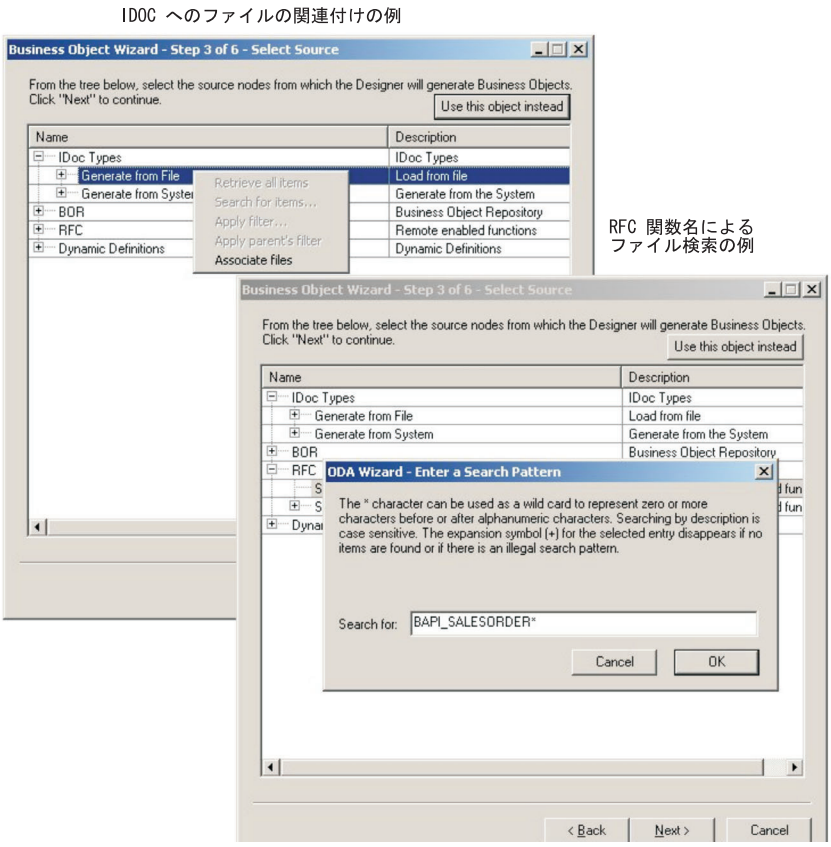


図 9. ファイルの関連付けと検索基準の入力

オブジェクトを生成するために必要なすべてのリーフを選択したら、「次へ」ボタンをクリックします。戻されるオブジェクトをフィルター操作する方法については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

オブジェクト選択の確認

生成したビジネス・オブジェクト定義に関連付けられたすべてのオブジェクトを識別すると、Business Object Designer Express には、選択したリーフとそれらのノード・パスのみを表示したダイアログ・ボックスが表示されます。 47 ページの図 10 にこのダイアログ・ボックスを示します。

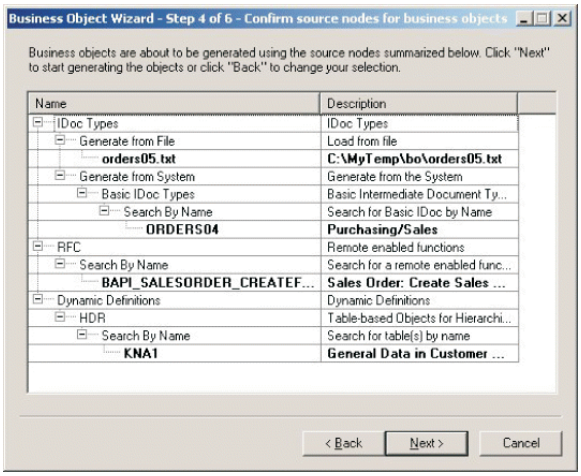


図 10. ノードとリーフの選択の確認

このウィンドウには、以下のオプションが表示されます。

- 選択を確認するには、「次へ」をクリックします。
- 選択に誤りがあった場合には、「戻る」をクリックして、直前のウィンドウに戻り、必要な変更を加えます。選択が正しい場合には、「次へ」をクリックします。

定義の生成

選択したオブジェクトを確認すると、Business Object Designer Express が定義を生成中であることを知らせるダイアログ・ボックスが表示されます。

図 11 にこのダイアログ・ボックスを示します。

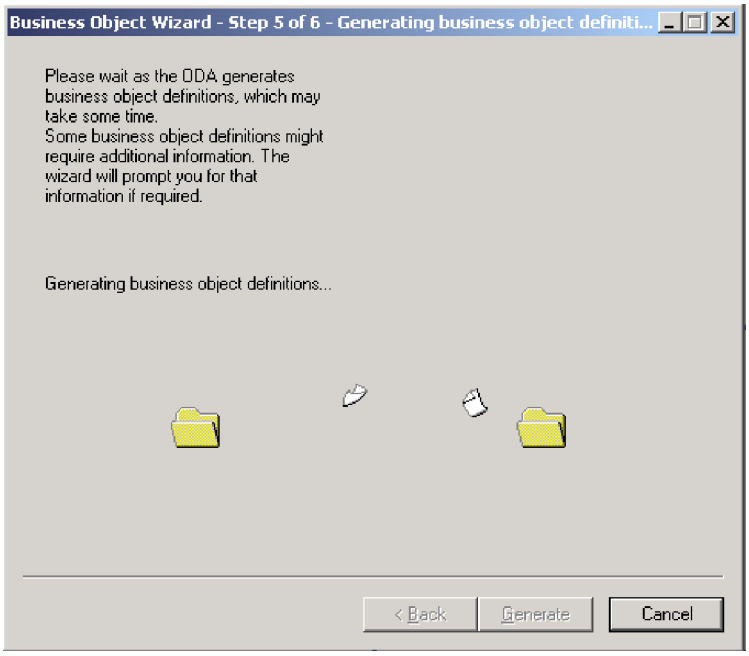


図 11. 定義の生成

追加情報の指定

SAPODA から、追加情報の入力が要求されます。トップレベル・ノードのタイプ (IDoc タイプ、BOR、RFC、または動的定義) によって、以下のことが決定されます。

- Business Object Designer Express で「BO プロパティー」ウィンドウに表示されるプロパティーのセット。
- Business Object Designer Express で、追加のオブジェクト生成情報の入力を要求する第 2 のウィンドウを表示するかどうか。

IDoc タイプ: 追加情報の指定

SAPODA で表示される「BO プロパティー」ウィンドウで、IDoc タイプに基づいてビジネス・オブジェクトに必要な情報を指定することができます。このウィンドウに表示されるプロパティーは、IDoc のソース (抽出されたファイルであるか、または SAP システム内の定義であるか)、および定義が ABAP Extension Module に定義されているかどうかによって異なります。このセクションでは、以下のトピックについて説明します。

- 『「BO プロパティー」ウィンドウ — 共通プロパティー』
- 49 ページの『「BO プロパティー」ウィンドウ — SAP システムに定義された IDoc のプロパティー』
- 49 ページの『「BO プロパティー」ウィンドウ — ABAP Handler に対する機能モジュールの指定』

「BO プロパティー」ウィンドウ — 共通プロパティー: SAPODA がビジネス・オブジェクト定義を IDoc ファイルから生成するか、SAP システムに定義された IDoc から生成するかにかかわらず、IDoc タイプの「BO プロパティー」ウィンドウを使用すると以下の情報を指定または変更できます。

- プレフィックス情報

  このプレフィックスは、ビジネス・オブジェクトの名前を固有にするために、名前の前に付加するテキストです。「エージェントの構成」ウィンドウ (41 ページの図 6) で DefaultBOPrefix プロパティーに対して入力した値が適切であれば、ここでこの値を変更する必要はありません。
- モジュール・タイプ

  モジュール・タイプの選択項目は ALE または Extension です。ALE と ABAP Extension Module ではビジネス・オブジェクト定義の要件が異なるため、どのモジュールがビジネス・オブジェクトを使用するかを指定することが重要です。

  注: 複数のセグメントが IDoc のトップレベルにある場合に、SAPODA が ABAP Extension Module 用のビジネス・オブジェクト定義を生成するときは、最初の IDoc セグメントを使用してトップレベルのビジネス・オブジェクトを表します。SAPODA は他のトップレベル・セグメントを子ビジネス・オブジェクトとして表します。

- UseFieldName

  SAP フィールド名または SAP フィールド記述から属性名を生成します。デフォルトでは SAP フィールド記述です。

「BO プロパティー」ウィンドウ — SAP システムに定義された IDoc のプロパティー: prefix プロパティーと module プロパティー以外にも、SAP システムに定義された IDoc を表す「BO プロパティー」ウィンドウには Release プロパティーが表示されます。このプロパティーを使用して、IDoc タイプの以前のバージョンを識別できます。

重要: 以前のバージョンの IDoc タイプに含まれるセグメントが現行バージョンよりも少ない場合は、SAPODA によって欠落しているセグメントが作成される場合があります。あるいは、ビジネス・オブジェクト定義の生成が失敗したことを示すエラーが表示されます。このような不整合が発生するのは、SAP のバージョンごとに必要な API 呼び出しが異なるためです。

図 12 に、2 つのバージョンの「BO プロパティー」ウィンドウを示します。一方は抽出した IDoc タイプ定義ファイル、他方は SAP システムに定義された IDoc の「BO プロパティー」ウィンドウです。

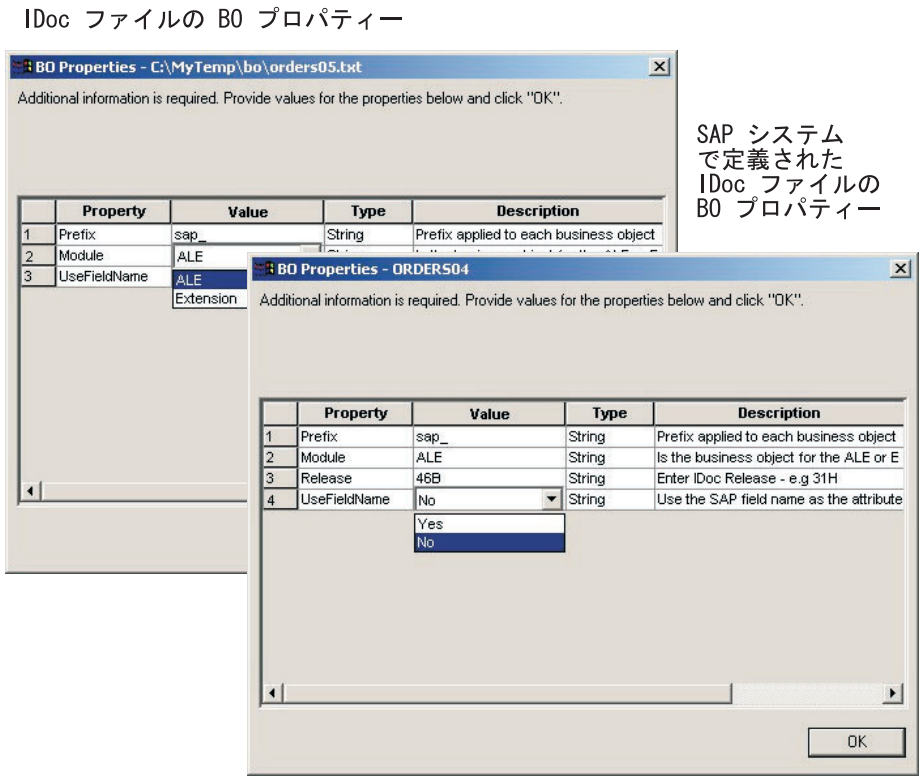


図 12. IDoc タイプ・ビジネス・オブジェクトの追加情報の指定

「BO プロパティー」ウィンドウ — ABAP Handler に対する機能モジュールの指定: モジュール・タイプとして Extension を選択した場合は、デフォルトの動詞のいずれかに対して機能モジュール名を入力するかどうかを尋ねるプロンプトが SAPODA によって表示されます。

デフォルトでは、ABAP Extension Module の定義を生成すると、SAPODA によって以下のテキストがトップレベル・ビジネス・オブジェクトのビジネス・オブジェクト・レベルの動詞のアプリケーション固有情報に指定されます。

```
:/CWLD/IDOC_HANDLER
```

ABAP Handler に渡す機能モジュール名が既にわかっている場合は、このプロンプトで「はい」を選択してください。SAPODA により、図 13 に示す画面が表示されます。

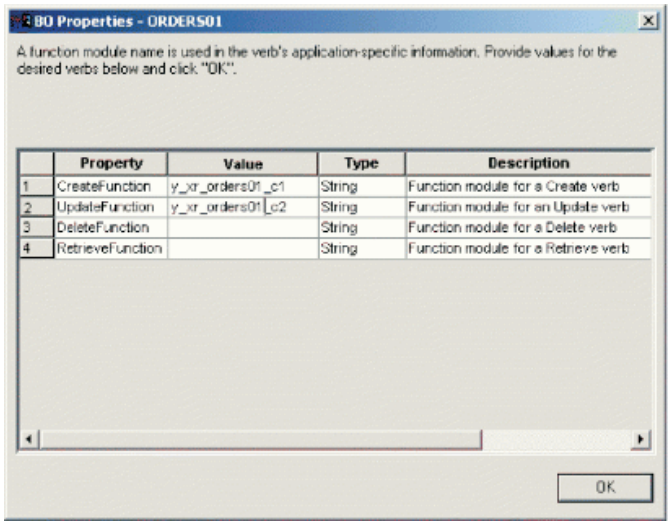


図 13. ABAP Handler に対する機能モジュールの指定

図 13 は、2 つの機能モジュールが指定された「BO プロパティー」ウィンドウを示しています。

注: 多数の IDoc タイプがファイル内に定義されている場合には、ファイル内の各 IDoc タイプに対して「機能モジュール BO プロパティー (Function module BO Properties)」ウィンドウが表示されます。「汎用 IDoc タイプ BO プロパティー (General IDoc type BO properties)」ウィンドウは 1 回のみ表示されます。

ビジネス・オブジェクト定義を保管した後は、トップレベル・ビジネス・オブジェクトのビジネス・オブジェクト・レベルの必要なアプリケーション固有情報が Business Object Designer Express の「一般」タブに表示されます。 51 ページの図 14 に、2 つの機能モジュールが指定されたこのウィンドウを示します。

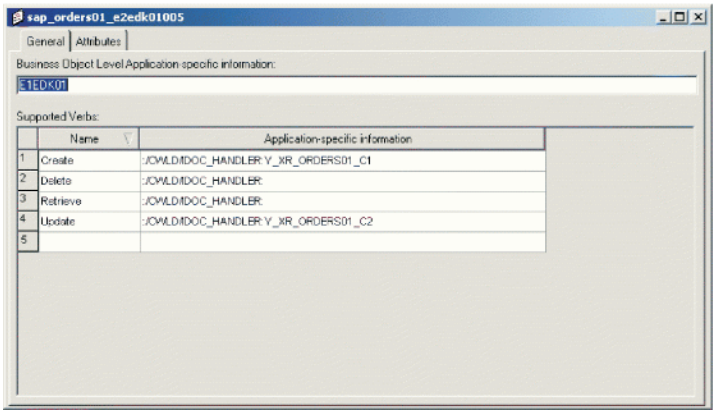


図 14. Business Object Designer Express での ABAP Handler の指定

ABAP Handler の詳細については、238 ページの『ABAP Handler へのビジネス・オブジェクト・データ発送』を参照してください。ABAP Extension Module で必要なアプリケーション固有の情報の詳細については、253 ページの『SAPODA を使用したビジネス・オブジェクト定義の開発』を参照してください。

BOR または RFC: 追加情報の指定

SAPODA は、以下のタイプのオブジェクトを作成します。

- 単一の BAPI ビジネス・オブジェクト
- BAPI トランザクションのトップレベルのビジネス・オブジェクト
- ResultSet ビジネス・オブジェクト

単一の BAPI ビジネス・オブジェクト: 41 ページの図 6 に示したように、「エージェントの構成」ウィンドウで `ResultSet` プロパティーを `False` に設定した場合、SAPODA を使用して、単一の BAPI トランザクション・ビジネス・オブジェクトや複数の BAPI 呼び出しを含む BAPI トランザクション・ビジネス・オブジェクトを作成することができます。このセクションでは、単一の BAPI 呼び出し用のビジネス・オブジェクトについて詳しく説明します。

BAPI トランザクション用のビジネス・オブジェクトの作成方法の詳細については、53 ページの『BAPI トランザクション・ビジネス・オブジェクト』を参照してください。 ResultSet 用のビジネス・オブジェクトの作成方法の詳細については、58 ページの『ResultSet ビジネス・オブジェクト』を参照してください。

BOR タイプまたは RFC タイプの単一 BAPI オブジェクト用の、2 つの「BO プロパティー」ウィンドウがあります。最初のウィンドウに表示されるプロパティーでは、以下の項目を指定または変更できます。

- `Prefix` — 「エージェントの構成」ウィンドウ (41 ページの図 6) で `DefaultBOPrefix` プロパティーに入力した値が適切であれば、ここでこの値を変更する必要はありません。
- `Verb` — 動詞を指定します。
- `Server Support`— コネクターの RFC Server Module について定義を生成する場合は、`yes` を指定します。コネクターの BAPI Module について定義を生成する場合は、`no` に指定します。

- UseFieldName — 属性名は、SAP フィールド名または SAP フィールド記述から派生させることができます。

「OK」をクリックして、最初の「BO プロパティー」ウィンドウから次に進むと、SAPODA によって生成される定義のサイズを減らすことのできるダイアログ・ボックスが表示されます。定義から、オプショナル・パラメーターを表すいずれかの属性を削除するかどうかを選択するように要求されます。このプロンプトは、削除できるオプショナル・パラメーターが存在する場合にのみ表示されます。定義のサイズを減らすと、後にコネクターがビジネス・オブジェクトのインスタンスを処理する際に、パフォーマンスが向上する可能性があります。

図 15 に、BOR または RFC タイプのオブジェクトに対して表示されるプロパティーと、「OK」をクリックした後に表示されるプロンプトを示します。このプロンプトは、単一の BAPI 呼び出しオブジェクトを作成するために選択したものと同数の個別の BAPI 呼び出しに対して表示されます。

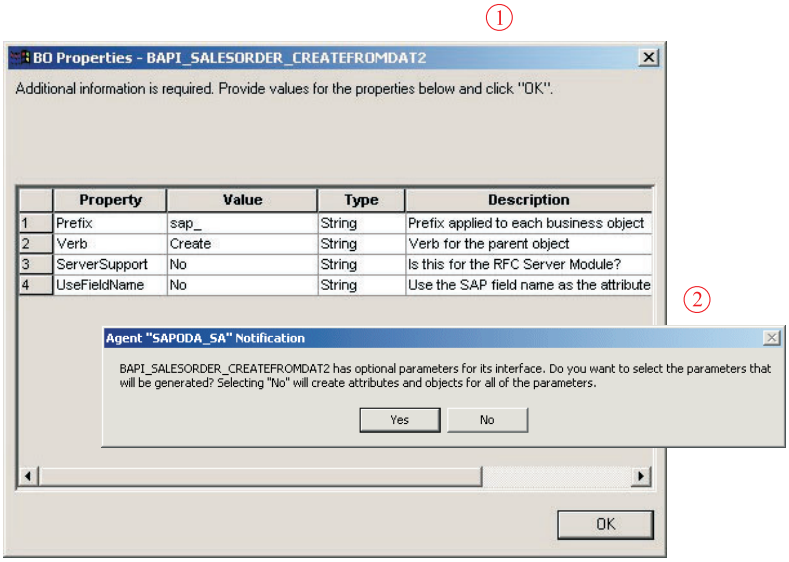


図 15. BOR または RFC ビジネス・オブジェクトの追加情報の指定

上に示したプロンプトで「**はい**」をクリックすると、2 番目の「BO プロパティー」ウィンドウが表示されます。BAPI/RFC インターフェースの各オプショナル・パラメーターを削除するように指定するには、その「値」を `Yes` (対応する属性を生成される定義に組み込む) から `No` (属性を組み込まない) に変更します。

上に示したプロンプトで「**いいえ**」をクリックすると、最終ウィザードが表示されます。詳細については、 67 ページの『定義の保管』を参照してください。

53 ページの図 16 に、2 番目の「BO プロパティー」ウィンドウを示します。

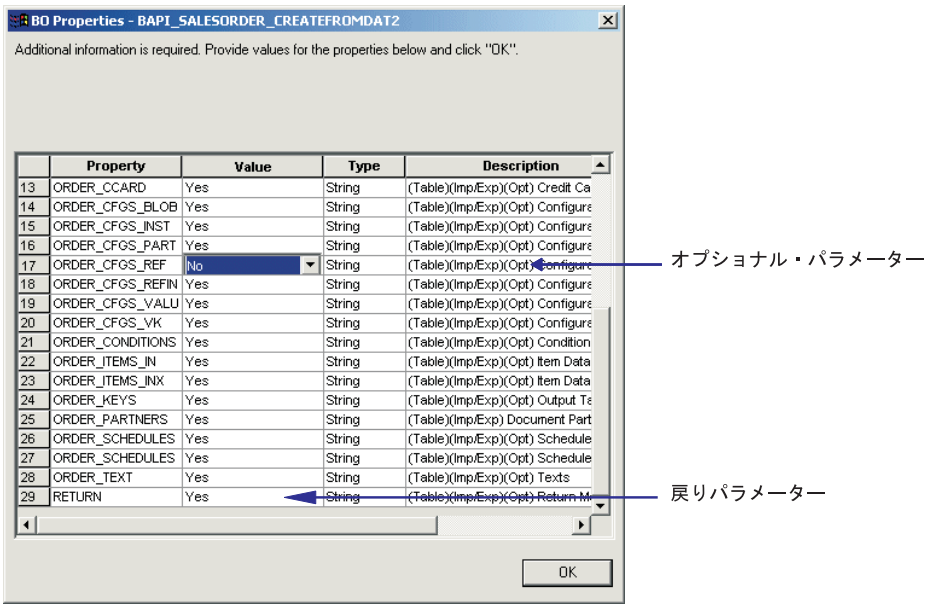


図 16. 定義から削除する属性の指定

重要: 「Bapi」で始まる RFC 対応機能のビジネス・オブジェクト定義には、戻り構造体またはテーブルに対応するビジネス・オブジェクトを表す属性が含まれている必要があります。この属性が定義に含まれていない場合は、対応する生成コードがコンパイルされるときにエラーが発生します。このコンパイル・エラーが発生したら、BAPI を検査して、SAP が異なる戻り構造体を使用していなかったかどうかを判別してください。そうであった場合には、生成された Java コードを、正しいパラメーターを指すように変更します。

SAPODA での指定のほかに、RFC Server Module の定義を作成する場合にも、ビジネス・オブジェクト定義を保管した後にアプリケーション固有の情報を変更することもできます。詳細については、167 ページの『第 14 章 RFC Server Module のビジネス・オブジェクトの開発』を参照してください。

BAPI トランザクション・ビジネス・オブジェクト: 41 ページの図 6 に示したように、「エージェントの構成」ウィンドウで ResultSet プロパティーを False に設定した場合、SAPODA を使用して、BAPI トランザクション・ビジネス・オブジェクトを作成することができます。BAPI トランザクション・ビジネス・オブジェクトには、複数の BAPI ビジネス・オブジェクトが含まれます。

「エージェントの構成」ウィンドウで ResultSet プロパティーを False に設定した場合、このウィンドウで「**次へ**」をクリックして、キャッシュの通知のウィンドウ (59 ページの図 27) に進み、「**OK**」をクリックします。

図 17. キャッシュの通知

図 18 に、次に表示されるウィンドウを示します。このウィンドウでは、SAPODA が BAPI 呼び出しを検索し、これを表示するために使用する基準を指定することができます。このセクションで使用する例では、基準は「BAPI_SALESORDER」というテキストで始まるすべての BAPI 呼び出しです。

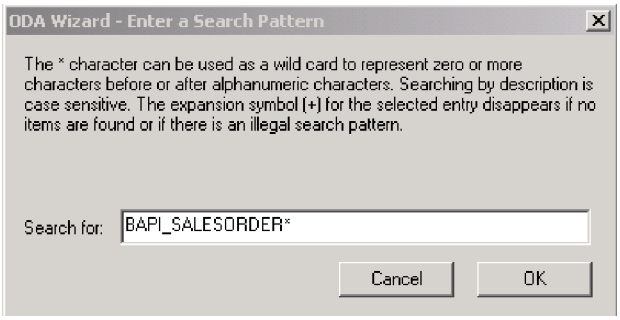


図 18. 「検索パターンの入力」

「検索パターンの入力」ウィンドウで検索基準を指定した後、「**OK**」をクリックして基準を設定します。 60 ページの図 29 に、RFC ノードを展開した状態の検索結果ツリーを示します。このウィンドウで、BAPI トランザクション・ビジネス・オブジェクトの属性を作成するために SAPODA が使用する BAPI 呼び出しを選択します。この例では、BAPI_SALESORDER_CHANGE と BAPI_SALESORDER_CONFIRMDELVRY の BAPI 呼び出しが選択されています。

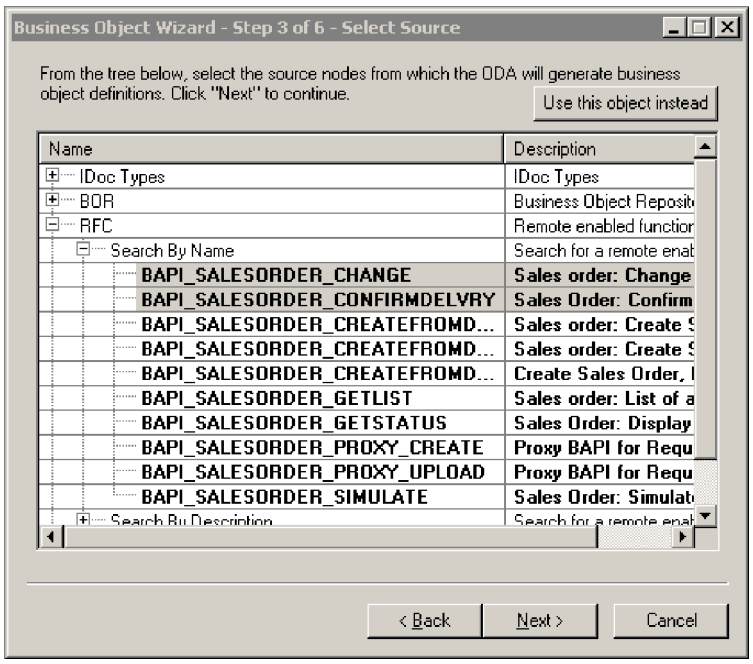


図 19. RFC ノードでの検索結果ツリーの展開

「**次へ**」をクリックして、図 20 に示す確認ウィンドウに進みます。図 19 で選択された 2 つの BAPI 呼び出しがリストされます。

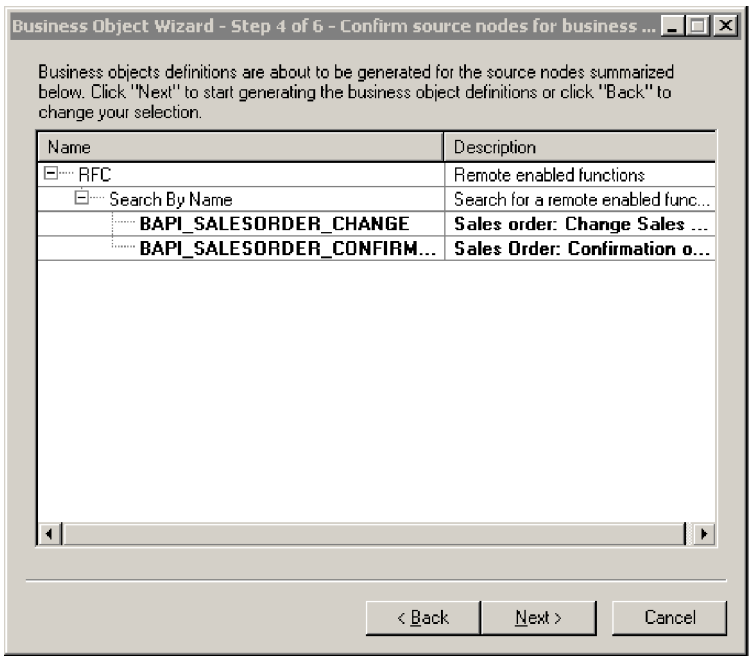


図 20. トランザクション内の BAPI 呼び出しのソース・ノードの確認

この画面で「**次へ**」をクリックします。複数の BAPI 呼び出しが選択されていることを通知するメッセージ・ウィンドウが表示されます。「**はい**」をクリックして、これらの複数の BAPI 呼び出しから 1 つの BAPI トランザクション・ビジネス・オブジェクトを作成する意図を示します。

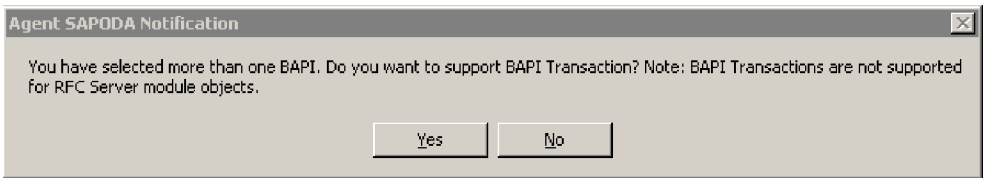


図 21. 複数の BAPI 呼び出しの選択を示すメッセージ

次の画面では、BAPI トランザクション・オブジェクトにプレフィックスと名前を設定することができます。この例で入力したプレフィックスは sap_ で、トランザクション・オブジェクトの名前は salesorder_txn です。UseFieldName プロパティーは、属性名を SAP のフィールド名を使用して生成するか、フィールド記述を使用して生成するかを決定します。

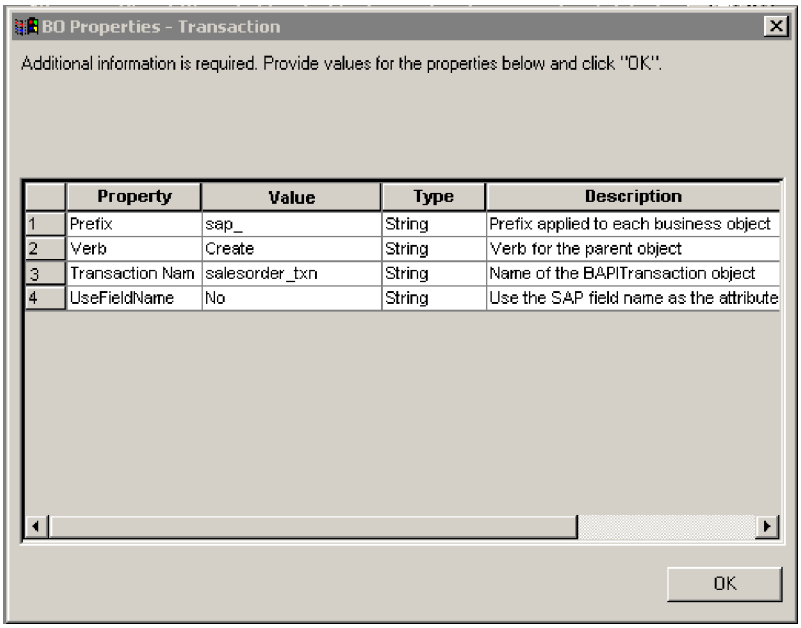


図 22. トランザクションへのプレフィックスとビジネス・オブジェクト名の割り当て

次に、トランザクション・オブジェクトの処理時に、選択した BAPI 呼び出しを実行する順序を指定します。この例では、最初に BAPI_SALESORDER_CHANGE 呼び出しが実行され、その後に BAPI_SALESORDER_CONFIRMDELVRY 呼び出しが実行されます。必要な BAPI の後で COMMIT を適用して、トランザクション内でコミットすることができます。SAPODA では、COMMIT がトランザクションの最終ステップと見なされます。

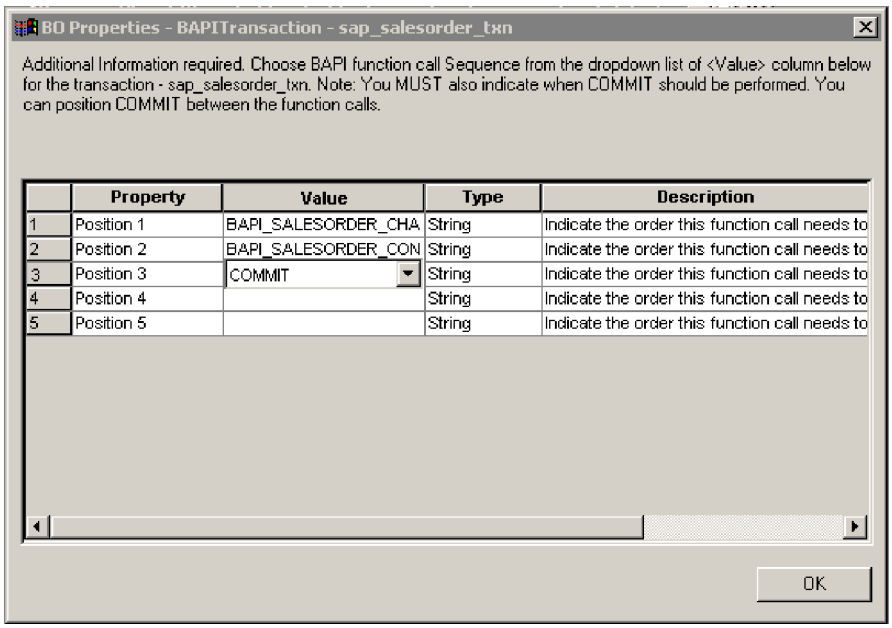


図 23. BAPI トランザクション・オブジェクト内での BAPI の呼び出し順序の割り当て

オプショナル・パラメーターが存在する順序内の BAPI 呼び出しごとに、以下のメッセージが表示されます。図 24 に、順序内の最初の BAPI 呼び出し (BAPI_SALESORDER_CHANGE) に対するこのメッセージを示します。

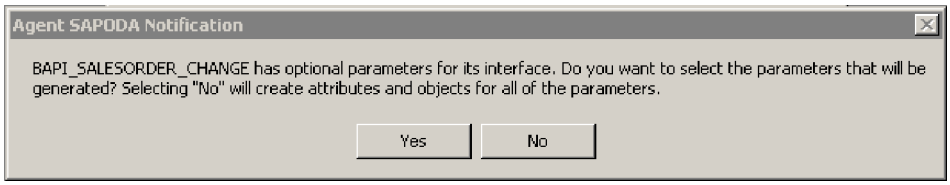


図 24. オプションの BAPI 呼び出しパラメーターのメッセージ

「**はい**」をクリックして、BAPI トランザクション・オブジェクトに含める個別の BAPI ビジネス・オブジェクトの属性として追加するオプショナル・パラメーターを選択します。「**いいえ**」をクリックすると、すべてのオプショナル・パラメーターが、BAPI トランザクション・オブジェクト内の個別の BAPI オブジェクトの属性として適用されます。

BAPI トランザクション・オブジェクト内の BAPI 呼び出しごとに BAPI オブジェクトの属性を作成し終わると、ビジネス・オブジェクト・ウィザードに BAPI トランザクション・オブジェクトのオブジェクト・ツリーが表示されます。 58 ページの 図 25 に、この例で作成した sap_salesorder_txn ビジネス・オブジェクト用の「**属性**」 タブを示します。

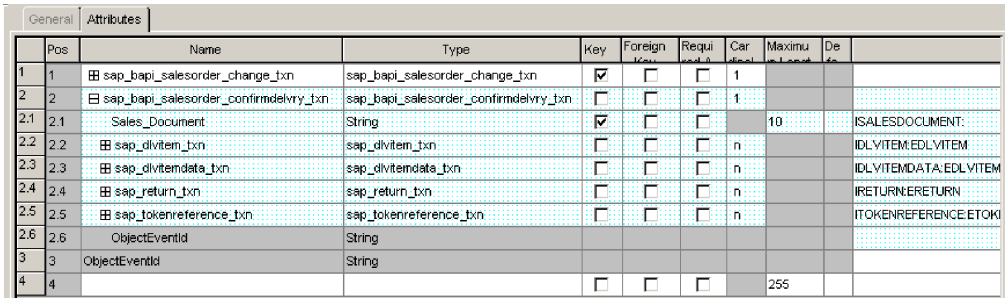


図 25. sap_salesorder_txn ビジネス・オブジェクトの「属性」タブ

sap_salesorder_change_txn と sap_salesorder_confirmdelvry_txn の 2 つの BAPI 呼び出し属性に注意してください。これらの各属性には、BAPI トランザクション・オブジェクト・ラッパー内に単一の BAPI 呼び出しオブジェクトが含まれています。sap_salesorder_change_txn 属性には、BAPI_SALESORDER_CHANGE 呼び出しに対応するビジネス・オブジェクトが含まれており、この呼び出しは、(57 ページの図 23 で指定したように) トランザクション・フローの中で最初に実行されます。sap_salesorder_confirmdelvry_txn 属性には、BAPI_SALESORDER_CONFIRMDELVRY BAPI 呼び出しに対応するビジネス・オブジェクトが含まれています。属性には、SAPODA が付加する _txn というサフィックスがあることに注意してください。このサフィックスにより、以前のバージョンのコネクターで作成されたビジネス・オブジェクトが同じ名前を持つ新規のビジネス・オブジェクトで上書きされることはありません。

ResultSet ビジネス・オブジェクト: 「エージェントの構成」ウィンドウで ResultSet プロパティーを True に設定すると、59 ページの図 26 に示すように、SAPODA がトップレベルの ResultSet ビジネス・オブジェクトを作成します。ResultSet ビジネス・オブジェクトでは、DB2 に対する Information Integrator のサポートが可能になります。

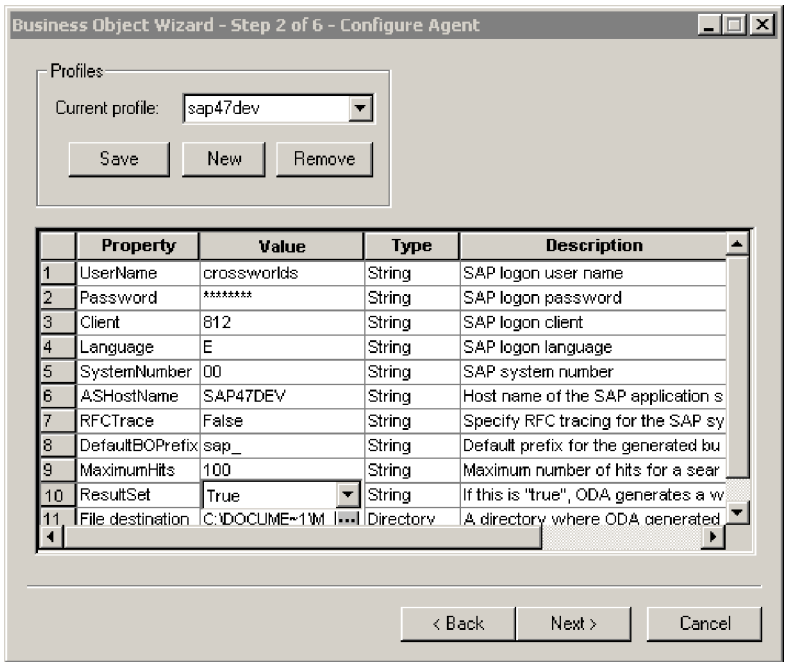


図 26. ResultSet プロパティーを True に設定した「エージェントの構成」ウィンドウ

「エージェントの構成」ウィンドウで「**次へ**」をクリックした後、キャッシュの通知のウィンドウ (図 27) で「**OK**」をクリックします。

図 27. キャッシュの通知

ウィザードの次のウィンドウでは、SAPODA が BAPI 呼び出しを検索し、これを表示するための基準を指定することができます。この例では、ワイルドカードであるアスタリスクは、基準が「BAPI_CUSTOMER_GET」というテキストで始まるすべての BAPI 呼び出しであることを示しています。

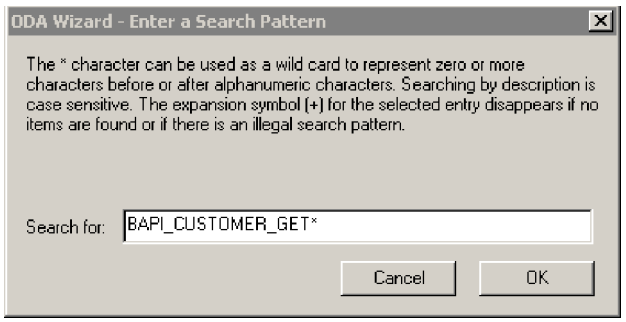


図 28. ResultSet の検索基準

「**OK**」をクリックして、基準を設定します。図 29 に、RFC ノードを展開した状態の検索結果ツリーを示します。このウィンドウで、SAPODA が ResultSet ビジネス・オブジェクトの属性を作成するために使用する BAPI 呼び出しを選択します。

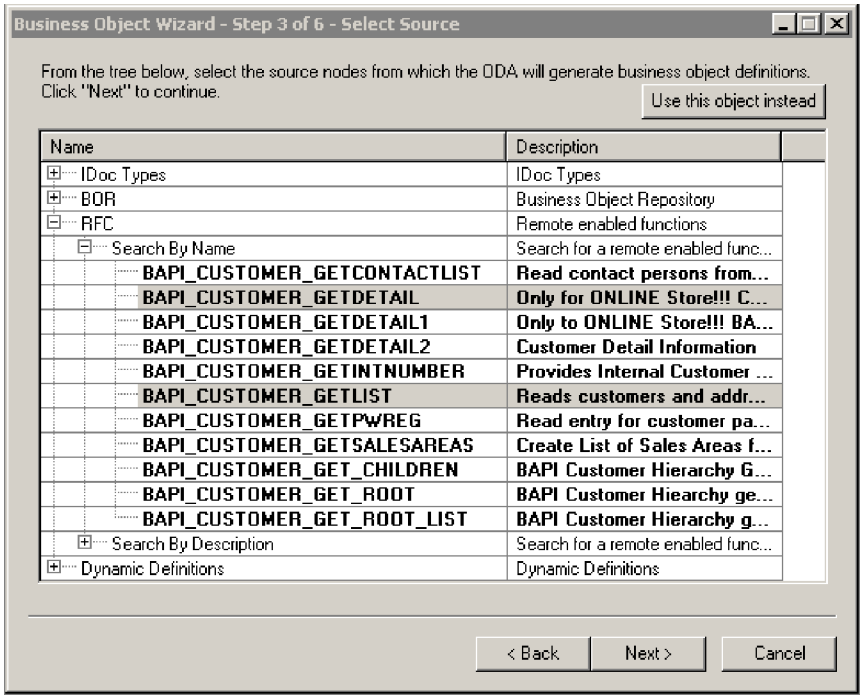


図 29. RFC ノードでの検索結果ツリーの展開

ResultSet オブジェクトには、Query (照会オブジェクト・タイプ) と Result (結果オブジェクト・タイプ) の 2 つの属性があります。Query 属性は通常 GETLIST BAPI 呼び出しから生成され、Result 属性は GETDETAIL BAPI 呼び出しから生成されます。

したがって、図 29 に示したように、これらに対応する BAPI 呼び出しを選択します。この例では、展開した RFC ノードから BAPI_CUSTOMER_GETDETAIL と BAPI_CUSTOMER_GETLIST を選択します。 59 ページの図 26 で示したように、「エージェントの構成」ウィンドウで ResultSet プロパティーが True に設定されているため、2 つの BAPI 呼び出しの選択のみが許可されます。

「**次へ**」をクリックして、 61 ページの図 30 に示す確認ウィンドウに進みます。このウィンドウでは、ビジネス・オブジェクトの属性のソースの BAPI 呼び出しを確認できます。

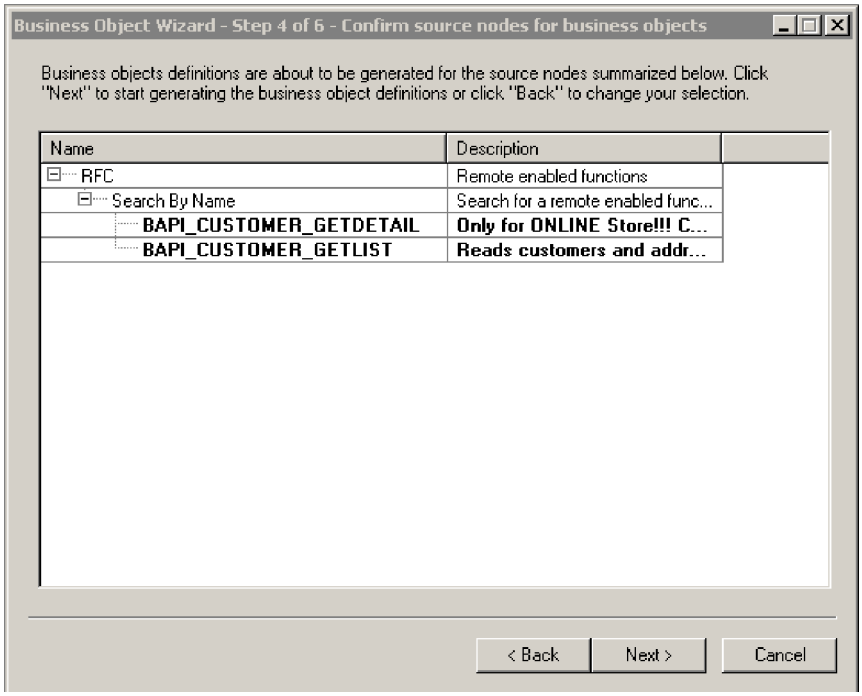


図 30. 属性のソース・ノードの確認

図 31 に示すように、ビジネス・オブジェクト・ウィザードにより、SAPODA がそのビジネス・オブジェクトに適用するビジネス・オブジェクト名のプレフィックス (この例では、sap_) と BAPI ResultSet オブジェクトの名前 (この例では、customer_rs) を指定するよう要求されます。UseFieldName プロパティーは、属性名を SAP のフィールド名を使用して生成するか、フィールド記述を使用して生成するかを決定します。

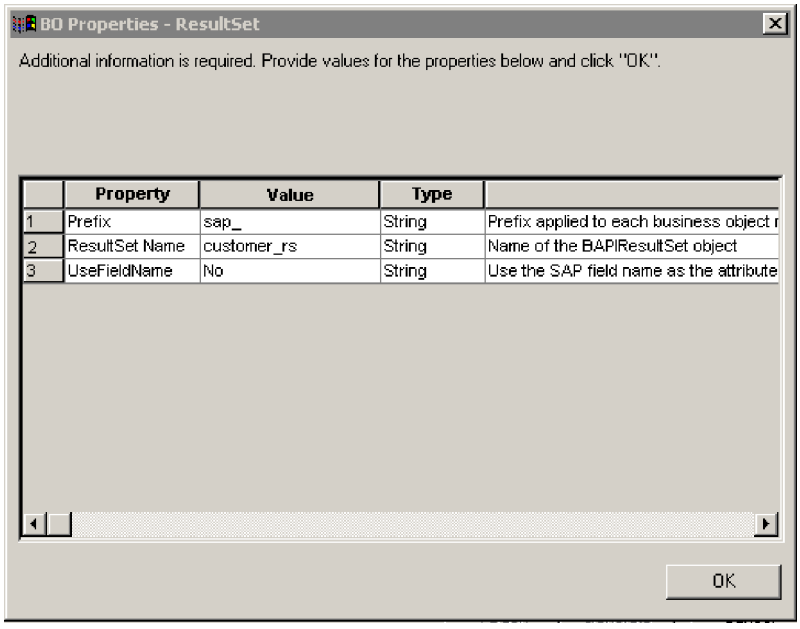


図 31. ビジネス・オブジェクト名情報の指定

また、ビジネス・オブジェクト・ウィザードにより、 60 ページの図 29 で選択した BAPI 呼び出しのうちのどちらを Query 属性に使用するかを指示するように要求されます。図 32 の例のように、ドロップダウン・リストから、GETLIST BAPI 呼び出しを選択します。SAPODA は、選択されたその他の呼び出しを、自動的に ResultSet オブジェクトの Result 属性として扱います。

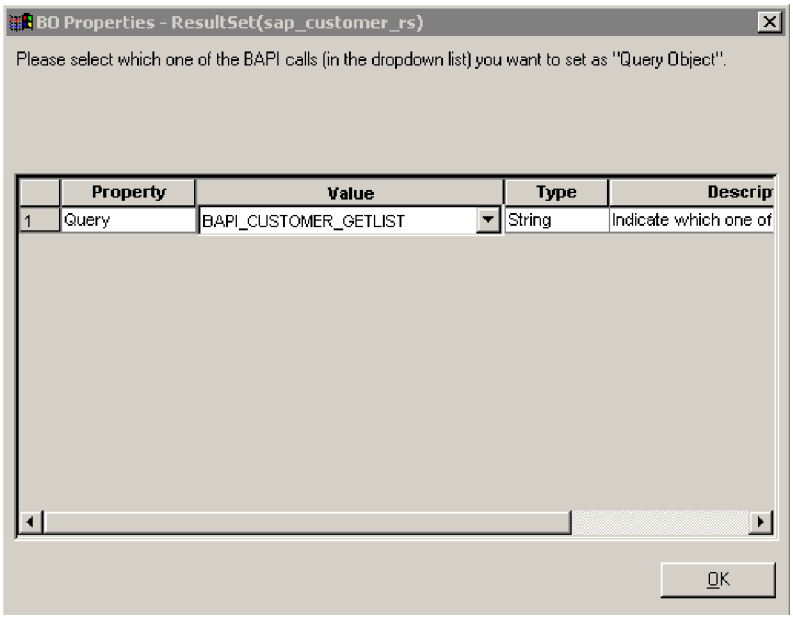


図 32. ビジネス・オブジェクトの Query 属性の指定

「**OK**」をクリックして、次のウィンドウに進みます。このウィンドウでは、図 32 で選択した Query BAPI の Query パラメーター (基本キー) を指定します。この例では、BAPI 呼び出しは GETLIST です。

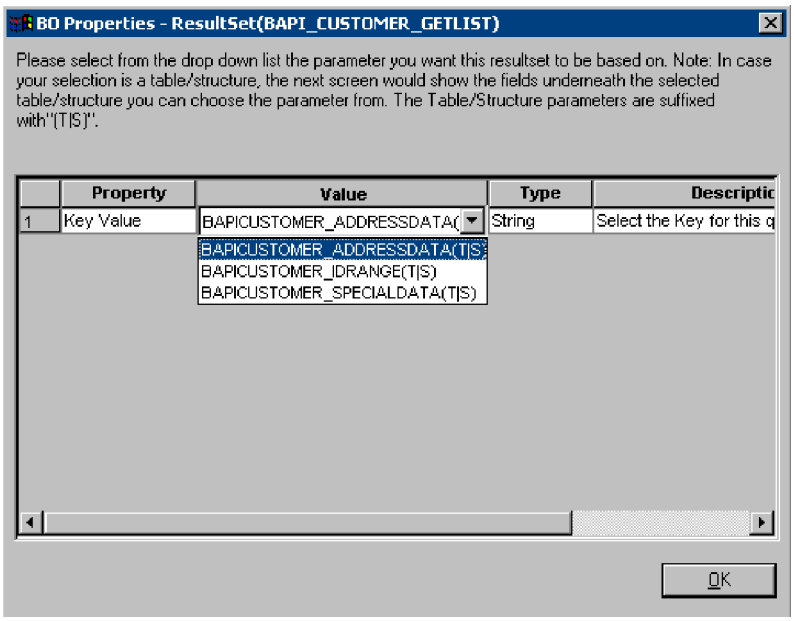


図 33. 基本キーの選択

前のウィンドウで表/構造体 (T|S で示される) の値を選択した場合、次に表示されるウィンドウで、表/構造体を基本キーとする特定のフィールドを選択することができます。この例では、CUSTOMER というフィールドが選択されています。

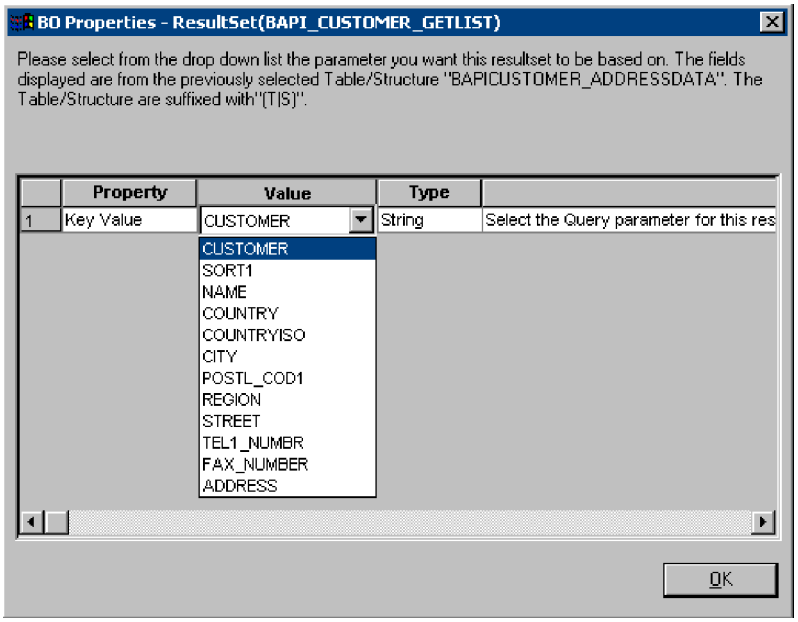


図 34. 表/構造体のフィールドの選択

64 ページの図 35 に示すように、Query パラメーターの絶対パスを示すメッセージ・ウィンドウが表示されます。このパスには、前の 2 つのウィンドウで選択した BAPI 呼び出しパラメーターが含まれています。この例では、BAPICUSTOMER_ADDRESSDATA と CUSTOMER が入っています。

図 35. Query 属性名の通知

ResultSet オブジェクトの外部キーも指定する必要があります。図 36 に例を示します。外部キーは、ResultSet オブジェクトの Query 属性と Result 属性間の関係を設定します。

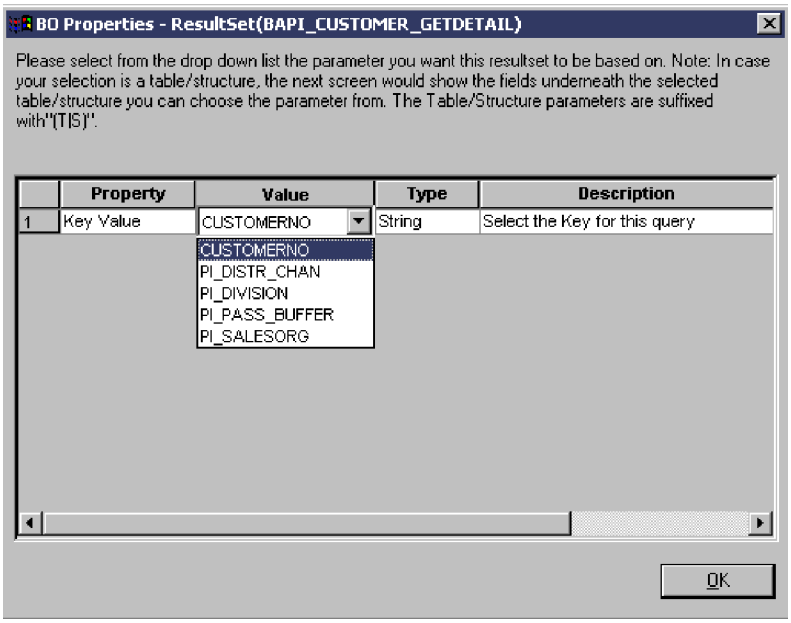


図 36. 外部キーの選択

ビジネス・オブジェクト・ウィザードにより、外部キーの絶対パスを確認するメッセージが表示されます。この例では、図 37 に示す `BAPI_CUSTOMER_GETDETAIL.CUSTOMERNO` です。

図 37. 外部キーのパスの確認

次に示すウィンドウは、`GETLIST` BAPI にはビジネス・オブジェクトの対応する属性を作成するためのオプショナル・パラメーター群があり、その中から選択できることを示しています。この例のように「**いいえ**」を選択した場合、ウィザードがすべてのパラメーターに対応するビジネス・オブジェクト属性を生成することを意味します。

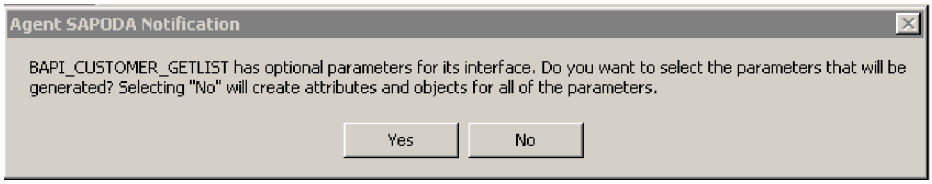


図 38. GETLIST BAPI のオプショナル・パラメーターの通知

図 39 に示す次の画面では、ResultSet オブジェクトのプロパティー値を設定することができます。

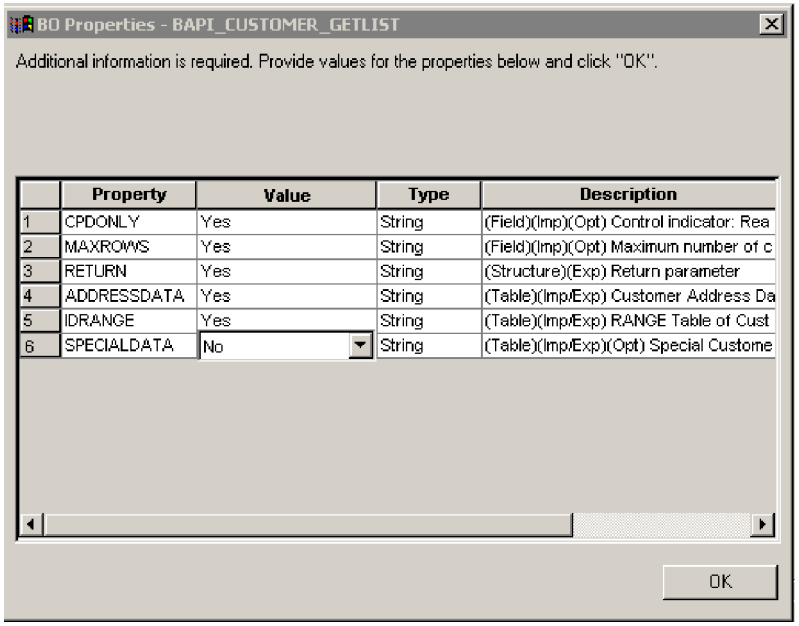


図 39. ResultSet オブジェクトのプロパティー値の設定

次に示すウィンドウは、GETDETAIL BAPI にはビジネス・オブジェクトの対応する属性を作成するためのオプショナル・パラメーター群があり、その中から選択できることを示しています。この例のように「**いいえ**」を選択した場合、ウィザードがすべてのパラメーターに対応するビジネス・オブジェクト属性を生成することを意味します。

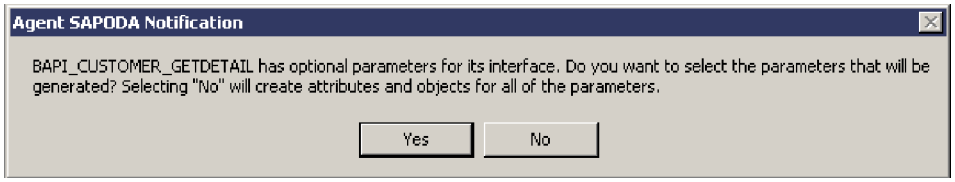


図 40. GETDETAIL BAPI のオプショナル・パラメーターの通知

66 ページの図 41 に、Business Object Designer Express の「**属性**」タブを示します。このタブでは、ResultSetビジネス・オブジェクトの BAPI_Query と BAPI_Result の 2 つの属性をリストしています。ビジネス・オブジェクト・ツリーを展開して、

各属性の階層を表示することができます。属性には、SAPODA が付加する _rs というサフィックスがあることに注意してください。このサフィックスにより、以前のバージョンのコネクターで作成されたビジネス・オブジェクトが同じ名前を持つ新規のビジネス・オブジェクトで上書きされることはありません。

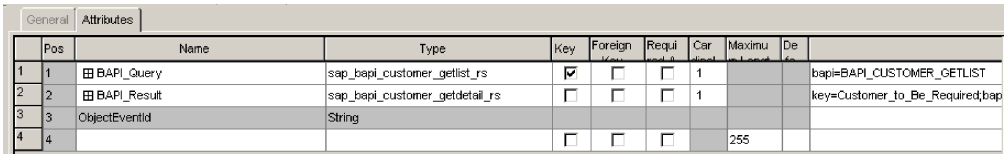


図 41. ResultSet オブジェクトの「属性」タブ

HDR タイプ: 追加情報の指定

HDR 表ベースのオブジェクトには、2 つの「BO プロパティー」ウィンドウがあります。最初のウィンドウに表示されるプロパティーを使用して、ビジネス・オブジェクトのプレフィックスを指定または変更できます。「エージェントの構成」ウィンドウ (41 ページの図 6) で DefaultBOPrefix プロパティーに対して入力した値が適切であれば、ここでこの値を変更する必要はありません。

図 42 に、このウィンドウを示します。

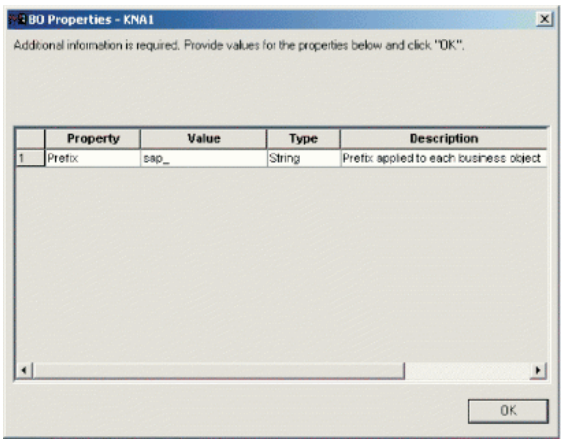


図 42. HDR ビジネス・オブジェクトの追加情報の指定

さらに、テーブルからの情報のうち 512 バイトのみを戻すことができます。テーブルが 512 バイトを超えるデータを戻す場合は、67 ページの図 43 に示すダイアログが表示されます。「いいえ」で応答すると、テーブルの先頭から最大 512 バイトに達するまで属性 (列の記述) が戻されます。

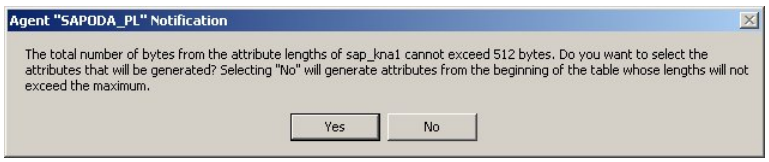


図 43. 512 バイトの警告

「はい」で応答すると、図 44 に示す 2 番目の「BO プロパティー」ウィンドウが表示されます。各属性のバイト長はウィンドウの記述で示されます。ビジネス・オブジェクトの属性を含めるか除外するかを指定するには、値を「はい」または「いいえ」に切り替えます。

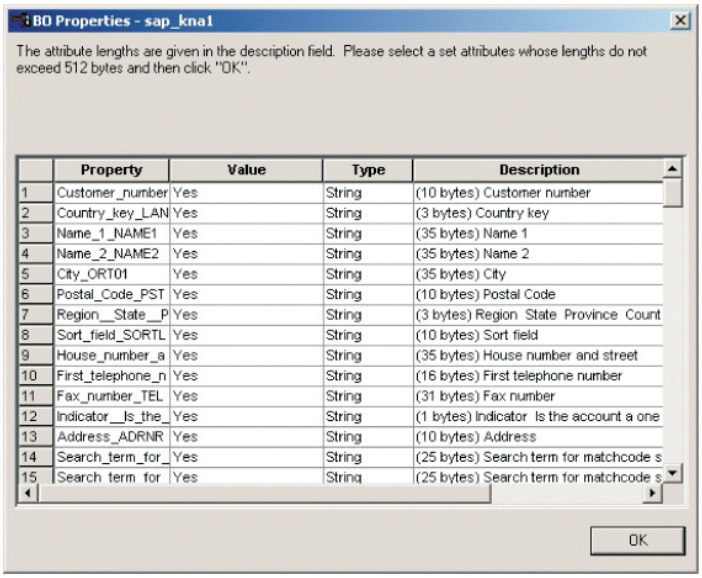


図 44. HDR ビジネス・オブジェクトに対する BO プロパティーのサイズおよびタイプ

定義の保管

「BO プロパティー」ダイアログ・ボックスで必要なすべての情報を指定し、「OK」をクリックすると、Business Object Designer Express にウィザードの最終ダイアログ・ボックスが表示されます。このダイアログ・ボックスでは、定義をサーバーまたはファイルに保管したり、定義を Business Object Designer Express で開いて編集することができます。詳細について、または追加の修正を行うには、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

68 ページの図 45 にこのダイアログ・ボックスを示します。

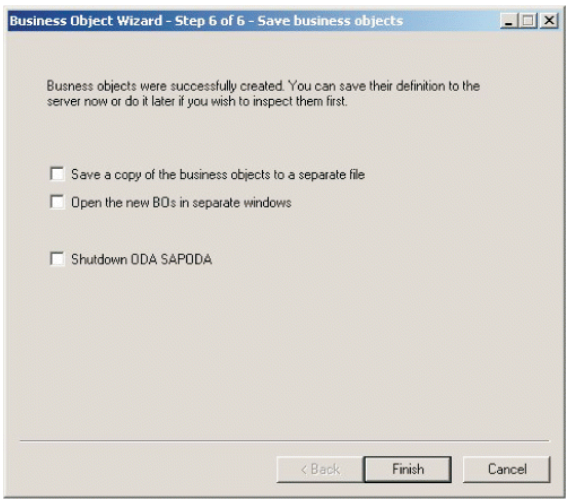


図 45. ビジネス・オブジェクト定義の保管

SAPODA を使用した後に

SAPODA が IDoc タイプ、BAPI、RFC 対応機能モジュール、ビジネス処理を表現する SAP テーブルから生成したビジネス・オブジェクト定義には、コネクターがビジネス・オブジェクトを処理するために必要なすべての情報が含まれているわけではありません。そのため、SAPODA による定義の生成が完了したら、必要なすべての情報を定義に追加する必要があります。Business Object Designer Express を使用して、ビジネス・オブジェクト定義を検査および修正し、変更した定義をリポジトリーに再ロードまたはコピーしてください。

ALE Module で Business Object Designer Express を使用してファイル・システムにビジネス・オブジェクトを保管するときは、最初に `sap_idoccontrol` オブジェクトをロードする必要があります。このオブジェクトは提供されており、SAPODA により生成されるものではありませんが、親ビジネス・オブジェクトをファイル・システムに保管する際に必要になります。

ビジネス・オブジェクト定義の変更については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。特定のコネクター・モジュールが必要とするビジネス・オブジェクト定義や、それをコネクターが処理できるようにするために行う必要のある変更については、以下に示す適切なモジュールの資料を参照してください。

- 247 ページの『第 21 章 ABAP Extension Module のビジネス・オブジェクトの開発』
- 141 ページの『第 11 章 ALE Module のビジネス・オブジェクトの開発』
- 97 ページの『第 8 章 BAPI Module のビジネス・オブジェクトの開発』
- 167 ページの『第 14 章 RFC Server Module のビジネス・オブジェクトの開発』
- 189 ページの『第 17 章 Hierarchical Dynamic Retrieve Module のビジネス・オブジェクトの開発』

第 5 章 コネクターのトラブルシューティング

この章では、Adapter for mySAP.com のコネクター・コンポーネントを始動または実行する際に発生する可能性のある問題について説明します。

この章の内容は以下のとおりです。

一般的なトラブルシューティング

このセクションでは、アダプターのモジュールの始動時または実行時に発生する可能性のある問題について説明します。以下のトラブルシューティングの項目について説明します。

始動時の問題

以下のサブセクションでは、共通の始動時の問題に関する推奨事項を示します。

コネクターが始動しない

コネクターの始動時に障害が発生した場合には、以下のように対処します。

- 統合ブローカーが稼働しているかどうかを確認します。
- アプリケーションが実行中かどうかを確認します。
- 標準構成プロパティーおよびコネクター固有の構成プロパティーが正しく設定されているかどうかを検査します。詳細については、 13 ページの『第 2 章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 69 ページの『第 5 章 コネクターのトラブルシューティング』 を参照してください。

コネクターが SAP アプリケーションにログオンできない

コネクターが SAP アプリケーションにログオンできない場合には、次のように対処します。

- SAP アプリケーションが使用可能かどうかを検査します。
- 標準構成プロパティーおよびコネクター固有のコネクター構成プロパティーが正しく設定されているかどうかを検査します。特に、Sysnr、 Client、 Hostname、および Modulesの各プロパティーを検査します。詳細については、 13 ページの『第 2 章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 315 ページの『付録 B. 標準構成プロパティー』 を参照してください。
- コネクターに対してセットアップしたユーザー名とパスワードに、適切なレベルの特権が与えられているかどうかを検査します。

コネクターのログオン直後にセッションがクローズする

コネクターが SAP アプリケーションに正常にログオンし、その直後にセッションがクローズする場合は、データベースに問題がある可能性があります。PSAPUSER1D および PSAPUSER1I 表スペースに対して十分なスペースが割り振られているかどうかを確認します。デフォルトでは、SAP システムは、これら 2 つの表スペースに対して最小のスペースを割り振ります。コネクターは、デフォルトのスペースよりも大きなスペースを必要とします。詳細については、 231 ページの『ログ表スペース・サイズの拡張』を参照してください。

注: この問題は、RFC Server Module を除くすべてのコネクター・モジュールに関係します。

コネクターがダウンする

コネクターが「connection to the SAP application is lost」というメッセージを表示してダウンしたり、RFC システム例外が発生した場合には、ネットワークに問題がある可能性があります。コネクター・ユーザー向けの短いダンプ、またはエラーが発生した時刻を検査してください。IBM WebSphere BI Station ツールを使用するか、またはトランザクション ST22 に移動します。さらに情報が必要な場合は、トランザクション SM21 に移動して、システム・ログを検査します。

デフォルト値が設定されていない

ビジネス・オブジェクトではデフォルト値が設定されていても、コネクターがその値を取り出していません。これは構成の問題です。デフォルト値を使用するには、UseDefaults コネクター・プロパティーを true に設定し、デフォルト値を必要とする各属性を、ビジネス・オブジェクト定義で必須としてマークする必要があります。

コラボレーションがビジネス・オブジェクトにサブスクライブされない (InterChange Server Express のみ)

InterChange Server Express で、コラボレーションが特定のビジネス・オブジェクトにサブスクライブされていない場合には、以下のように対処します。

- コラボレーションが、その特定のビジネス・オブジェクトにサブスクライブするように構成されているかどうかを検査します。
- コラボレーションが実行中かどうかを検査します。
- マップ参照に対して、正しいビジネス・オブジェクトがソース・ビジネス・オブジェクトとして指定されているかどうかを検査します。

WBI のパフォーマンス・チューニングおよびメモリー管理

Java 仮想マシン (JVM) では複数のチューニング手段が外部化されており、WebSphere Business Integration アプリケーションのパフォーマンスを向上させるために使用できます。これらの手段は、ガーベッジ・コレクション、ヒープ・サイズ、スレッド化、およびロックに関連する事項を制御します。InterChange Server Express サーバーとそのコンポーネント (マップ、コラボレーション)、およびほとんどのアダプターは Java で作成されているため、InterChange Server Express アプリケーションによって実現されるパフォーマンスには JVM のパフォーマンスが大きく影響します。

WebSphere Business Integration のパフォーマンスの考慮点の詳細については、IBM ソフトウェア・サポートの Web サイトにナビゲートして「WBI パフォーマンス・チューニング」について検索するか、以下に示す文書を参照してください。この文書は定期的に更新されます。

http://www-1.ibm.com/support/
docview.wss?rs=203&context=SW000&q1=wbi+performance
+tuning&uid=swg21173114&loc=en_US&cs=utf-8&lang=en

JVM オプションの概要については、次の URL を参照してください。

http://java.sun.com/docs/hotspot/VMOptions.html

次の URL に、HotSpot エンジンに関する有用な FAQ が記載されています。

http://java.sun.com/docs/hotspot/PerformanceFAQ.html

ABAP Extension Module のトラブルシューティング

このセクションでは、ABAP Extension Module の始動時または実行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 3 つの項目に関するトラブルシューティングについて説明します。

- 『トランスポート・ファイル』
- 72 ページの『始動時の問題』
- 73 ページの『イベント処理』
- 73 ページの『Microsoft Windows でのイベント分配問題 (コネクター・バージョン 4.2.7 のみ)』

トランスポート・ファイル

ABAP Extension Module 用のアダプターのトランスポート・ファイルをインストールする際にエラーが発生した場合には、以下のように対処します。

- 正しいトランスポート・ファイルをインストールしたかどうかを検査します。R/3 バージョン 3.x システムではバージョン 3.x のトランスポート・ファイルを、BASIS 4.x から 4.7 および SAP Web AS 6.20 を実行する SAP アプリケーションではバージョン 4.x のトランスポート・ファイルをそれぞれインストールする必要があります。トランスポート・ファイルはそれぞれのディレクトリーにイン

ストールされます。トランスポート・ファイルとそのインストール・ディレクトリーの詳細については、 225 ページの『コネクター・トランスポート・ファイル』 を参照してください。

- トランスポートを正しい順序でインストールしたかどうかを検査します。一部のトランスポート・ファイルは、既存のテーブルなどに対して依存関係を持ちます。

  例えば、あるトランスポート・ファイルは、あるテーブルに対するデータ・エレメントを作成し、別のトランスポートは、そのデータ・エレメントに対するテーブルを作成するとします。テーブルが先に作成されていないと、システムはエラーを戻します。

- 必要なトランスポート・ファイルが正しくインストールされているかどうかを検査します。各トランスポート・ファイルは、特定の機能をコネクターに追加します。

  詳細については、 225 ページの『コネクター・トランスポート・ファイル』を参照してください。

始動時の問題

コネクターは SAP アプリケーションに正常にログインしたが、SAP アプリケーション内のコネクターのログに何も記録されない場合には、以下のように対処します。

- ロギングがオンになっているかどうかを検査します。ロギングがオフになっている場合は、IBM WebSphere BI Station を使用してオンにします。デフォルトでは、ロギングは 1 に設定されています。詳細については、 283 ページの『第 23 章 ABAP Extension Module の管理』を参照してください。
- コネクターがログオンしたマシンが、コネクター・ログ・ファイルを表示しているマシンと同じかどうかを確認します。
- `Namespace` 構成プロパティーが `true` に設定されているかどうかを確認します。以前の `YXR` 環境からコネクターのネーム・スペースにアップグレードした場合は、コネクターが依然として `YXR` 環境にログインしている可能性があります。このような場合は、`Namespace` 構成プロパティーを `true` に設定してください。詳細については、 341 ページの『コネクター固有の構成プロパティー』内の 348 ページの『Namespace』プロパティーを参照してください。
- コネクター・ログの数値範囲が同期しているかどうかを検査します。NumberRange トランスポート番号をアップグレードすると、数値範囲間隔が同期しなくなる可能性があります。数値範囲オブジェクト番号が、コネクター・ログの最初の番号よりも小さいことを確認します。

  数値範囲を検査するには、トランザクション SNRO に移動し、「Number Range Object」フィールドに `/CWLD/LOG` と入力します。「Number Ranges」ボタンをクリックし、「Display Intervals」ボタンをクリックして、数値範囲オブジェクト番号をメモします。コネクター・ログを開き、最初のエントリーの番号をメモします。この番号が数値範囲オブジェクト番号よりも大きい場合は、コネクター・ログのログ・エントリー番号を 1 つ上の番号に変更する必要があります。詳細については、 231 ページの『トランスポート・オブジェクトの数値範囲の検証』を参照してください。

Microsoft Windows でのイベント分配問題 (コネクター・バージョン 4.2.7 のみ)

以下の条件下では、Windows 上で IBM CrossWorlds Connector for SAP バージョン 4.2.7 にアップグレードした後、イベントがイベント表に残り、コネクターによって選出されず、処理されなくなります。

- イベント分配を複数のコネクターにまたがって構成した場合。
- コネクターが、非ネーム・スペース (yxr) をロードした SAP システムに対して実行されている場合。

この問題は、SAP がそれ自身の java API (SAP JCo) で行った変更によって発生します。

この問題を修正するには、アダプターが提供するイベント要求機能モジュールおよびイベント戻り機能モジュールのみを変更するパッチ・トランスポートをロードします。このパッチ・トランスポートを、ネーム・スペース (/CWLD/) インフラストラクチャーを持たない 4.0 および 4.5 SAP システムにロードします。

注: ネーム・スペース ABAP インフラストラクチャーにはこの問題はありません。

イベント処理

以下のサブセクションでは、イベント処理の問題に関する推奨事項を示します。

サブスクライブしているビジネス・オブジェクトによって ABAP Extension Module が呼び出されない

サブスクライブしているビジネス・オブジェクトが ABAP Extension Module によって処理されない場合には、以下のように対処します。

- Vision コネクター・フレームワークが ABAP Extension Module を呼び出すように設定されているかどうかを検査します。Modules プロパティーには Extension を設定してください。
- コネクターがビジネス・オブジェクトにサブスクライブしているかどうかを検査します。

コネクターがイベントを選出しない

コネクターが SAP アプリケーションからイベントを選出しない場合には、以下のように対処します。

- SAP アプリケーションのコネクターのイベント表を検査し、そのコネクターに対するイベントが待機しているかどうかを調べます。
- 複数コネクター環境で、イベントが待機していない場合は、Event Distribution テーブル (/CWLD/EVT_DIS) に、そのコネクターとビジネス・オブジェクトの組み合わせに関するエントリーが存在することを確認します。この組み合わせが固有であるかどうかを検査します。

  同じビジネス・オブジェクトに複数のコネクターがサブスクライブされている場合、一方のコネクターが間違ったイベントを処理している可能性があります。複数コネクター間のイベント分配についての詳細については、219 ページの 『イベント分配』を参照してください。

- SAP アプリケーションにイベントに対するロック・オブジェクトが存在すると、SAP アプリケーションはそのイベントの保管プロセスの処理を完了できない場合があります。

  SAP アプリケーションのイベント表を検査し、そのイベントの状況が L (Locked) であるかどうかを調べてください。状況が L の場合は、ほとんどの場合、コネクターではなく SAP アプリケーションで問題が発生しています。
- コネクターがイベントの処理中にダウンしてしまった可能性があります。SAP アプリケーションのコネクターのイベント表で、そのイベントの状況を検査してください。状況が R (Retrieved) の場合、そのアーカイブ表に移動されていません。イベントの状況が R の場合には、イベントが宛先に到達していないことを確認してください。

  イベントが宛先に到達していない場合は、状況を R から Q (Queued) に変更します。状況が Q のイベントは、次のポーリング間隔でコネクターによって選出されます。状況を R から Q に変更するには、イベント表に移動し、イベントを選択してから、「編集」ボタンをクリックします。表示されるウィンドウで、「Event Status」フィールドを R から Q に変更します。

ビジネス・オブジェクトの処理に障害が起こる

ビジネス・オブジェクトが正常に処理されなかった場合は、SAP アプリケーションのコネクター・ログを検査してください。失敗したイベントに関するエントリーは赤で表示されます。イベントは、再処理ツールを使用して再処理します。再処理ツールを使用すると、トランザクションを段階ごとにたどりながら、コードにブレークポイントを設定することができます。オブジェクトの再処理の詳細については、284 ページの『アーカイブされたオブジェクトの再処理』を参照してください。

重要: 再処理ツールを使用すると、WebSphere Business Integration システムと SAP アプリケーションが同期しなくなるため、再処理ツールは実稼働環境では使用しないでください。

イベント表でのデッドロック

現在のイベント表、および将来のイベント表において、一度に多数のイベントが追加されたときに、デッドロック状態が発生する可能性があります。この状況は、データベースのチューニングのために、イベント表の索引が使用されない場合に生じます。チューニングは通常は、イベント表内のイベントが少ないか存在しないオフピーク時に行われます。データベース表上のエントリーが少ないかまたは存在しない場合には、テーブルの読み出しに索引を使用しない方が効率的です。デッドロック状態を回避するには、データベース・チューニング・ユーティリティーの実行時に現在のイベント表と将来のイベント表を除外します。

ラージ・オブジェクト

ラージ・オブジェクトを正常に処理するには、追加の変更が必要となります。ABAP Extension Module オブジェクトは、データを SAP アプリケーションに渡す前にフラット構造に変換されるか、またはデータを SAP アプリケーションから受信するときにフラット構造から変換されます。詳細については、234 ページの『ビジネス・オブジェクトのフラット構造への変換』を参照してください。このフラット構造は、構造内の行となるオブジェクト・インスタンスの各属性と共に、メモリー内に保持されます。各属性に対して 373 バイトのデータがコネクターと SAP アプリ

ケーション間で受け渡されます。属性の数に 373 を掛けた値がフラット構造のサイズの近似値です。同様に、オブジェクトのインスタンスもメモリー上にあります。そのため、メモリー不足エラーを回避するために、オブジェクトに多数の子オブジェクト (セグメント) がある場合は、コネクターの Java プロセスの始動スクリプト内で Java ヒープ・サイズを変更する必要があるかもしれません。

Windows

start_SAP.bat スクリプト内で、Java ヒープ・サイズ・オプション・パラメーターのデフォルト値の -mx128m を、フラット構造とオブジェクトのインスタンスを処理するために充分大きな値に変更します。Java プロセスが実行されるマシンで使用可能なメモリーより大きな値を指定した場合にも、メモリー不足エラーが発生します。128m は 128 MB の最大 Java ヒープ・サイズを表します。

Linux

CWConnEnv.sh スクリプト内で、JVM_FLAGS 環境変数の Java ヒープ・サイズ・オプション・パラメーターのデフォルト値 -Xmx128m を、フラット構造とオブジェクトのインスタンスを処理するために充分大きな値に変更します。Java プロセスが実行されるマシンで使用可能なメモリーより大きな値を指定した場合にも、メモリー不足エラーが発生します。128m は 128 MB の最大 Java ヒープ・サイズを表します。

SAP アプリケーションは、ラージ・オブジェクトを正常に処理するために、ABAP タイムアウト・パラメーターの変更も必要とします。

BAPI Module のトラブルシューティング

このセクションでは、BAPI Module の実行時に発生する可能性のある問題について説明します。

要求処理の取り扱い

以下のサブセクションでは、一般的な要求処理の取り扱いの問題に関する推奨事項を示します。

サブスクライブしているビジネス・オブジェクトによって BAPI Module が呼び出されない

サブスクライブしているビジネス・オブジェクトが BAPI Module によって処理されない場合には、以下のように対処します。

- Vision コネクター・フレームワークが BAPI Module を呼び出すように設定されているかどうかを検査します。Modules プロパティーを Bapi に設定する必要があります。
- コネクターがビジネス・オブジェクトにサブスクライブしているかどうかを検査します。

- カスタム・ビジネス・オブジェクト・ハンドラーのファイルが ¥bapi¥client ディレクトリーに格納されているかどうかを確認します。クラス・ファイルがこのディレクトリーに格納されていないと、ビジネス・オブジェクトを処理する際に、カスタム・ ビジネス・オブジェクト・ハンドラーが呼び出されません。詳細については、106 ページの『生成したビジネス・オブジェクト定義の使用』を参照してください。
- カスタム・ビジネス・オブジェクト・ハンドラーを使用する場合は、ビジネス・オブジェクトの動詞のアプリケーション固有情報のカスタム・ビジネス・オブジェクト・ハンドラー名が正しいかどうかを検査します。詳細については、172 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』を参照してください。
- カスタム・ビジネス・オブジェクト・ハンドラーを生成した際に、BAPI に合わせた適切な動詞を指定したかどうかを確認してください。詳細については、106 ページの『生成したビジネス・オブジェクト定義の使用』を参照してください。

ビジネス・オブジェクトの処理に障害が起こる

ビジネス・オブジェクトが正常に処理されなかった場合には、以下のように対処します。

- 使用している BAPI がリターン・ビジネス・オブジェクトを持っているかどうかを検査してください。BAPI Module はリターン・ビジネス・オブジェクト内を検索して、キー e (error) または a (abort) を持つメッセージを見つけます。モジュールは、いずれかのキーを見つけると、イベントが失敗したことを通知します。BAPI がリターン・ビジネス・オブジェクトを持っていない場合は、独自のエラー処理を必ず実装してください。
- トランザクション SE37 を使用して、失敗したイベントに関連付けられた BAPI をテストします。これにより、失敗を再現することができます。

  失敗が再現できない場合は、内部形式から外部形式に変換する際に問題が発生している可能性があります。値を正しい形式で指定しているかどうかを検査します。例えば日付の場合、SAP の内部形式は YYYYMMDD であるのに対し、MMDDYYYY という形式を指定している場合があります。その場合には、指定した形式を解釈できないため、BAPI は失敗します。
- 各属性のアプリケーション固有の情報が正しいかどうかを検査します。これらの値が正しくない場合、BAPI Module はオブジェクトを SAP アプリケーションに戻す前に、そのオブジェクトに正しくデータを取り込むことができません。
- I パラメーターおよび E パラメーターが正しく指定されているかどうかを検査します。I はインポート・パラメーターを示し、E はエクスポート・パラメーターを示します。詳細については、78 ページの『ビジネス・オブジェクトの処理に障害が起こる』を参照してください。

コネクターがポーリング中に見えるが、イベントは選出されていない

BAPI Module には、pollForEvents() メソッドのダミーの実装が含まれています。コネクターは、ポーリング・メッセージを戻しているため、ポーリングを行っているように見えます。BAPI Module はポーリングをサポートしないため、このメッセージは無視してください。

BAPI Module に対応したポーリングを実装するには、ABAP Extension Module のポーリング機能を使用する必要があります。詳細については、207 ページの『第 18 章 ABAP Extension Module の概要』を参照してください。

RFC Server Module のトラブルシューティング

このセクションでは、RFC Server Module の始動時または実行時に発生する可能性のある問題について説明します。以下の項目について説明します。

始動時の問題

コネクターが SAP アプリケーションに登録できない場合には、以下のように対処します。

- SAP アプリケーションが使用可能かどうかを検査します。
- 標準構成プロパティーおよびコネクター固有のコネクター構成プロパティーが正しく設定されているかどうかを検査します。特に、gwService、 Hostname、RfcProgramId、および Modulesの各プロパティーを検査します。詳細については、13 ページの『第 2 章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 69 ページの『第 5 章 コネクターのトラブルシューティング』 を参照してください。

コネクターがダウンする

コネクターが停止した場合は、以下の項目について検査してください。

- スレッドが RFC Server Module によって作成されているかどうかを検査します。NumberOfListeners プロパティーが正しく設定されているかどうかを検査します。詳細については、348 ページの『NumberOfListeners』を参照してください。
- RFC Server Module が自分自身を SAP Gateway に登録するように、RFC プログラム ID がセットアップされているかどうかを検査します。詳細については、348 ページの『RfcProgramId』および 165 ページの『RFC Server Module の SAP gateway への登録』を参照してください。

イベント処理

以下のサブセクションでは、共通のイベント処理の問題に関する推奨事項を示します。

サブスクライブしているビジネス・オブジェクトによって RFC Server Module が呼び出されない

サブスクライブしているビジネス・オブジェクトが RFC Server Module によって処理されない場合には、以下のように対処します。

- Vision コネクター・フレームワークが RFC server Module を呼び出すように設定されているかどうかを検査します。347 ページの『Modules』プロパティーを RfcServer に設定する必要があります。

- コネクターがビジネス・オブジェクトにサブスクライブしているかどうかを検査します。
- SAPODA により生成された BAPI 固有のビジネス・オブジェクト・ハンドラーのクラス・ファイルが ¥bapi¥server ディレクトリーにあるかどうかを確認します。クラス・ファイルがこのディレクトリーに格納されていないと、ビジネス・オブジェクトを処理する際に BAPI ビジネス・オブジェクト・ハンドラーが呼び出されません。詳細については、176 ページの『生成したビジネス・オブジェクトおよびビジネス・オブジェクト・ハンドラーの使用』を参照してください。
- ビジネス・オブジェクト動詞アプリケーション固有の情報の BAPI ビジネス・オブジェクト・ハンドラー名が正しいかどうかを検査します。詳細については、74 ページの『ビジネス・オブジェクトの処理に障害が起こる』を参照してください。
- BAPI 固有ビジネス・オブジェクト・ハンドラーに対して指定した動詞が、必要な処理のタイプに対して適切かどうかを検査します。特に、ビジネス・オブジェクト・ハンドラーを生成した際に、BAPI に合わせた適切な動詞を指定したかどうかを確認してください。詳細については、176 ページの『生成したビジネス・オブジェクトおよびビジネス・オブジェクト・ハンドラーの使用』を参照してください。

ビジネス・オブジェクトの処理に障害が起こる

ビジネス・オブジェクトが正常に処理されなかった場合には、以下のように対処します。

- 使用している BAPI がリターン・ビジネス・オブジェクトを持っているかどうかを検査してください。RFC Server Module はリターン・ビジネス・オブジェクト内を検索して、キー e (error) または a (abort) を持つメッセージを見つけます。モジュールは、いずれかのキーを見つけると、イベントが失敗したことを通知します。BAPI がリターン・ビジネス・オブジェクトを持っていない場合は、独自のエラー処理を必ず実装してください。
- トランザクション SE37 を使用して、失敗したイベントに関連付けられた BAPI をテストします。これにより、失敗を再現することができます。

  失敗が再現できない場合は、内部形式から外部形式に変換する際に問題が発生している可能性があります。値を正しい形式で指定しているかどうかを検査します。例えば日付の場合、SAP の内部形式は YYYYMMDD であるのに対し、MMDDYYYY という形式を指定している場合があります。その場合には、指定した形式を解釈できないため、BAPI は失敗します。
- 各属性のアプリケーション固有の情報が正しいかどうかを検査します。これらの値が正しくない場合、RFC Server Module はオブジェクトを SAP アプリケーションに戻す前に、そのオブジェクトに正しくデータを取り込むことができません。
- I パラメーターおよび E パラメーターが正しく指定されているかどうかを検査します。I パラメーターはインポート・パラメーターを示し、E パラメーターはエクスポート・パラメーターを示します。詳細については、76 ページの『ビジネス・オブジェクトの処理に障害が起こる』を参照してください。

ALE Module のトラブルシューティング

このセクションでは、ALE Module の始動時または実行時に発生する可能性のある問題について説明します。以下の事項について説明します。

始動時の問題

以下のサブセクションでは、共通の始動時の問題に関する推奨事項を示します。

コネクターが SAP アプリケーションにログオンまたは登録できない

コネクターが SAP アプリケーションにログオンまたは登録できない場合には、以下のように対処します。

- SAP アプリケーションが使用可能かどうかを検査します。
- 標準構成プロパティーおよびコネクター固有のコネクター構成プロパティーが正しく設定されているかどうかを検査します。
  - 必要な WebSphere MQ キューが作成されているかどうか、およびそれぞれに対応する構成プロパティーがそれぞれの名前を正しく指定しているかどうかを確認します。
  - 要求処理については、Sysnr、Client、Hostname、および Modules の各プロパティーを検査して下さい。
  - イベント処理については、gwService、Hostname、RfcProgramId、および Modules の各プロパティーを検査して下さい。

  詳細については、13 ページの『第 2 章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 69 ページの『第 5 章 コネクターのトラブルシューティング』 を参照してください。
- コネクターに対してセットアップしたユーザー名とパスワードに、適切なレベルの特権が与えられているかどうかを検査します。

コネクターがポーリング・イベントを選出しない

コネクターが SAP アプリケーションからイベントをポーリングしない場合には、以下のように対処します。

- 必要な動詞の動詞アプリケーション固有の情報が、正しいメッセージ・タイプ、メッセージ・コード、およびメッセージ機能を持つように変更されているかどうかを検査します。
- 動詞 AleOutboundVerbs が存在し、有効な動詞のリストを持っているかどうかを検査します。

コネクターがポーリング中に見えるが、イベントは選出されていない

- イベント・キュー (SAPALE_Event_Queue および SAPALE_Wip_Queue) が正しく作成されているかどうか、およびポーリングが event キューに対して実行されているかどうかを確認します。
- システム上で以下のものが稼働していることを確認します。
  - WebSphere MQ
  - TCP/IP
- SAP アプリケーション内での ALE 構成が正しいかどうかを検査します。詳細については、117 ページの『第 9 章 ALE Module の概要』を参照してください。
- コネクターがこれまでに少なくとも 1 回のポーリング呼び出しを実行しているかどうかを検査します。実行していれば、イベント処理用の機能モジュールがインストールされます。
- メッセージが wip キューに書き込まれ、event キューに移動されたかどうかを検査します。

イベント処理

コネクターは、(SAPALE_Event_Queue 構成プロパティーで指定されたキューの) JMS-MQ イベント・メッセージ内の正常に処理された IDoc に関する情報を、EventState.log ファイルに記録します。このファイルは、AleEventDir構成プロパティーで指定されたディレクトリーに格納されています。

注: それぞれのイベント・メッセージに、それぞれがビジネス・オブジェクトを表す複数の IDoc が含まれている場合があります。

コネクターが現行のイベント・メッセージのすべての IDoc を処理する前にダウンした場合、コネクターは、リカバリーの際に EventState.log ファイルを使用して、各 IDoc が 1 回のみ統合ブローカーに送信されるようにします。

重要: コネクターはイベントを初めて処理する際に自動的にログ・ファイルを作成しません。コネクターを最初に稼働させる前に、このファイルを作成する必要があります。

ログ・ファイルの書式は次のとおりです。

```
TID: OS, 1S, 2F, 3U
```

ここで、<*TID*> は、処理されている現行トランザクション ID であり、各番号はイベント・メッセージ内のすべての作業単位のシーケンス番号を表します。

例えば、コネクターが現行のイベント・メッセージの最初の 4 つの IDoc のうち 3 つを正常に処理し、2 番目の IDoc の処理に失敗した場合は、コネクターは現行のイベント・メッセージの処理を完了しておらず、EventState.log ファイルには以下のように示されます。

```
<TID> :: OS, 1F, 2S, 3S
```

コネクターがイベント・メッセージ全体を処理する前にダウンした場合、コネクターは始動時にログ・ファイルの情報を使用して、メッセージ内のイベントの処理を中断したポイントから再開します。コネクターはログを読み取り、リカバリーするイベントのトランザクション ID、最新の作業単位および各作業単位の状況を取得し

ます。次に、コネクターは統合ブローカーに対して、ログ・ファイル内の最後の番号よりもシーケンス番号が大きいイベント・メッセージ内の各 IDoc を表すビジネス・オブジェクトの送信を開始します。前記の例では、コネクターは現行のイベント・メッセージ内の 5 番目の IDoc を処理します。

コネクターは、パフォーマンスを向上させるため、ログ・ファイルの内容をメモリーに保持します。ディスク上のファイルには、新しいエントリーで更新する場合にのみアクセスします。コネクターは、ログ・ファイルをリカバリー時にのみ読み取ります。

リカバリー・プロセスでコネクターが EventState.log ファイルをどのように使用するかについては、『障害リカバリー』を参照してください。

障害リカバリー

注: 以下のリカバリー・ステップは、ディスクに障害が発生したり、ディスクが満杯の場合には適用されません。

コネクターは、イベント通知中の障害をリカバリーするために、以下の処理を実行します。

1. コネクターは event キュー (SAPALE_Event_Queue 構成プロパティーで指定) の JMS-MQ メッセージ内の IDoc を処理します。IDoc が正常に処理されると、EventState.log ファイルにエントリーを記録します。
   - イベント・メッセージ内の作業単位がすべて正常に処理された場合は、コネクターはイベント・メッセージをアーカイブ・キューに移動し、IDocProcessStatus 値を success にします。
   - イベント・キュー・メッセージ内の作業単位のいずれかの処理に失敗した場合は、コネクターはイベント・メッセージをアーカイブ・キューに移動し、IDocProcessStatus 値を partial に更新します。
2. コネクターはイベント・メッセージ内のすべての IDoc を処理した後、EventState.log ファイルをクリアして、このファイルへのエントリーの書き込みを次のイベント・メッセージから開始します。
3. コネクターがイベント・メッセージのすべての IDoc を処理する前にダウンした場合、コネクターはリカバリー処理中に EventState.log の情報を使用して、処理を開始する位置を決定します。復旧すると、コネクターはログ・ファイル内にエントリーがあるかどうかを検査します。
   - エントリーがない場合は、イベント・メッセージのすべての IDoc を統合ブローカーに送信します。
   - エントリーがある場合は、この情報を使用して、処理が中断された位置から、イベント・メッセージの処理を再開します。コネクターはログを読み取り、リカバリーするイベント・メッセージの名前と、最後の IDoc シーケンス番号を取得します。次に、コネクターは統合ブローカーに対して、ログ・ファイル内の最後の番号よりもシーケンス番号が大きいイベント・メッセージ内の各 IDoc を送信します。この例では、イベント・メッセージはアーカイブ・キューに移動され、IDocProcessStatus は EventState.log 内の各作業単位の状況に従って更新されます。

ログ・ファイルを使用することにより、コネクターが同じ IDoc を複数回統合ブローカーに送信することを防ぐことができます。コネクターは、パフォーマンスを向上させるため、ログ・ファイルをメモリーに保持します。ディスク上のファイルには、新しいエントリーで更新する場合にのみアクセスし、リカバリー時にのみログ・ファイルを読み取ります。

注: イベント・メッセージに、ログ・ファイル内の最後の番号よりも大きいシーケンス番号を持つ IDoc が存在しない場合、コネクターは最後のイベントを処理した後、イベント・ファイルをアーカイブする前にダウンします。この場合、イベント・メッセージはアーカイブ・キューに移動され、`IDocProcessStatus` は `EventState.log` 内の各作業単位の状況に従って更新されます。

ビジネス・オブジェクト作成エラーのリカバリー

コネクターが WIP キューのメッセージのヘッダー部分のみを作成し、データ部分を作成していない場合は、以下の手順によってメッセージのデータ部分をリカバリーします。

1. SAP コネクター・ログを調べて、該当するビジネス・オブジェクトの名前、メッセージ・タイプ、または動詞に関連したエラー・メッセージを見つけます。
2. ビジネス・オブジェクト定義またはコネクター構成に適切な修正を加えます。

   注: 構成変更には WebSphere MQ キューに対する変更も含まれる場合があります。詳細については、 125 ページの『ALE Module 実行の前提条件』を参照してください。
3. コネクターを再始動します。

要求処理

サブスクライブしているビジネス・オブジェクトが ALE Module によって処理されない場合には、以下のように対処します。

- Vision コネクター・フレームワークが ALE Module を呼び出すように設定されているかどうかを検査します。Modules プロパティーには `ALE` を設定してください。
- コネクターがビジネス・オブジェクトにサブスクライブしているかどうかを検査します。

Hierarchical Dynamic Retrieve Module のトラブルシューティング

このセクションでは、Hierarchical Dynamic Retrieve Module の始動時または実行時に発生する可能性のある問題について説明します。以下の項目について説明します。

- 『エラー処理とロギング』
- 84 ページの『SQL SELECT の失敗』

エラー処理とロギング

コネクターは、検索が失敗する原因となる条件が発生すると、エラー・メッセージを記録します。また、そのようなエラーが発生すると、コネクターは失敗したビジ

ネス・オブジェクトのテキスト表現も、統合ブローカーから受け取ったままの状態で出力します。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。このテキストを使用すると、エラーの原因を調べることができます。

エラー・タイプ

表 7 に、Hierarchical Dynamic Retrieve Module 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を示します。これらは、Java コネクター実行ラッパーや WebSphere MQ メッセージ・インターフェースなどの、WebSphere Business Integration システムのアーキテクチャーで出力されるトレース・メッセージに追加されるメッセージです。

表 7. 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、これ以外のトレースは実行されません。
レベル 1	機能モジュールの開始および終了メッセージ。これらのメッセージは、コネクター実行スレッドが、ある機能を開始したり、終了した際に書き込まれます。このメッセージは、コネクターの処理フローをトレースするために役立ちます。
レベル 2	コネクターがビジネス・オブジェクトの処理中に検出または検索した配列や子ビジネス・オブジェクトなどの情報を格納している、ビジネス・オブジェクト・ハンドラー・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。これらのメッセージは、例えばコネクターが複数のビジネス・オブジェクトの間に一致を検出したとき、子ビジネス・オブジェクトの配列内でビジネス・オブジェクトを検出したとき、あるいは検索中に子ビジネス・オブジェクトを削除したときに出力されます。
レベル 4	• アプリケーション固有情報メッセージ。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プロパティーを構文解析する関数から戻された値を示すメッセージ。 • コネクターがいつ Java メソッドを開始または終了したかを識別するメッセージ。これは、コネクターの処理フローをトレースするために役立ちます。 • SQL ステートメント。このレベルおよびこれより上のレベルでは、コネクターは実行したすべての SQL ステートメントを出力します。 • 検索時における属性値に対する変更。このレベルおよびこれより上のレベルでは、コネクターは属性の名前およびその新しい値を出力します。

表 7. 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。 • ビジネス・オブジェクトのダンプからなるメッセージ。 • コネクターがビジネス・オブジェクトの処理を開始する前 (コネクターが統合ブローカーから受け取った際のビジネス・オブジェクトの状態を表示) およびコネクターがビジネス・オブジェクトの処理を完了した後 (コネクターが統合ブローカーに戻す際のビジネス・オブジェクトの状態を表示) のビジネス・オブジェクトの表現からなるメッセージ。

コネクター・メッセージ・ファイル

コネクターが生成したエラー・メッセージは、Connector.txt. という名前のメッセージ・ファイルに保存されます。それぞれのエラーには、エラー・メッセージの前にエラー番号が付けられます。例えば、次のようにします。

```
1210

SAP Hierarchical Dynamic Retrieve Module unable to initialize.

1211

SAP Hierarchical Dynamic Retrieve module failed to locate...
```

RFC_READ_TABLE の呼び出し失敗

SAP RFC_READ_TABLE 関数はキャラクター・ベースのデータ型を処理しません。フィールドで以下のデータ型が使用されている場合には、このモジュールはデータの検索に失敗します。

- CURR
- DEC
- FLTP
- INT1
- INT2
- INT4
- LRAW
- RAW
- RAWSTRING

SQL SELECT の失敗

SELECT ステートメントが失敗した場合には、キーとしてマーク付けされているか、または外部キーとして使われている単純属性に、一重引用符 (') が含まれているかどうかを検査します。含まれていた場合は、一重引用符 (') を 2 つの一重引用符 ('') に変換するように、ビジネス・オブジェクトのマップを修正します。

SAPODA のトラブルシューティング

SAPODA には、使用時に発生する可能性のある既知の問題が 2 つあります。

- SAPODA がメッセージなしに稼働します。

  ODA に指定されたメッセージ・ファイルが存在しない場合、ODA はメッセージなしに稼働します。この問題は Business Object Designer Express でメッセージ・ファイルのデフォルト名が表示される際に、ODA の構成中に発生します。デフォルト名は命名規則に従っています。

  *AgentName*Agent.txt

  実際のメッセージ・ファイルの名前がこの規則に従わず、デフォルト値が実際の値によって上書きされない場合は、Business Object Designer Express は、ODA の起動元となったウィンドウにエラー・メッセージを表示します。このメッセージは、Business Object Designer Express ではポップアップしません。

  詳細については、 38 ページの『エラーおよびトレース・メッセージ・ファイルでの作業』を参照してください。

- Windows システムの場合、Business Object Designer Express が必要なライブラリー・ファイルを Path 環境変数内で見つけることができないか、またはファイルがシステム上に存在しないときには、ツリー・ノードの取得を試行する間に CORBA Exception が表示されます。これらのファイルの詳細については、 36 ページの『SAPODA を使用する前に』を参照してください。

サポートの取得方法

このセクションでは、トラブルシューティングを行う際の技術情報の使用法について説明します。

本書の発行後に公開されたテクニカル・サポートの技術情報や速報に、本書の対象製品に関する重要な情報が記載されている場合があります。これらの技術情報や速報は、WebSphere Business Integration のサポート Web サイトで参照できます。

WebSphere Business Integration のサポート Web サイトで技術情報や速報にアクセスするには、次のようにします。

1. http://www.ibm.com/software/integration/websphere/support/ にアクセスします。
2. 適切なコンポーネント領域を選択し、「Technotes (技術情報)」セクションと「Flashes (速報)」セクションを参照または検索してください。

第 2 部 BAPI Module

第 6 章 BAPI Module の概要

この章では、SAP BAPI Module の概要について説明します。BAPI Module では、統合ブローカーがビジネス・オブジェクトを SAP アプリケーションに送信することができます。

BAPI は SAP の標準化された Business Application Programming Interface で、サード・パーティー製品が SAP アプリケーションと相互作用することを可能にします。BAPI は、SAP ビジネス・オブジェクトのメソッド用の RFC 対応機能モジュールとして実装されています。

この章の内容は以下のとおりです。

注: BAPI は、SAP アプリケーションの RFC 対応機能です。BAPI Module を使用すると、BAPI だけではでなく、任意の RFC 対応機能をサポートできます。

BAPI Module のコンポーネント

BAPI Module は、SAP アプリケーションを直接呼び出すネイティブの BAPI 呼び出しをサポートする、Java で記述されたコネクター・モジュールです。これは、VisionConnectorAgent クラスおよび VisionBOHandler クラスを実装することによって、Vision コネクター・フレームワークを拡張します。BAPI Module は、Java および C で作成された SAP RFC ライブラリーを使用します。これにより、外部プログラムが SAP アプリケーションと通信できます。

図 46 に、BAPI Module 全体のアーキテクチャーを示します。BAPI Module は、SAP RFC ライブラリーのほかに、コネクター・フレームワーク、BAPI に対応するコネクターのアプリケーション固有のコンポーネント、およびすべての BAPI 呼び出しをサポートする 1 つの BAPI ビジネス・オブジェクト・ハンドラーから構成されています。BAPI Module 提供の 1 つの BAPI ビジネス・オブジェクト・ハンドラーのほかに、カスタム・ビジネス・ オブジェクト・ハンドラーを作成することができます。これについては 109 ページの『カスタム・ビジネス・オブジェクト・ハンドラーの使用』で説明します。

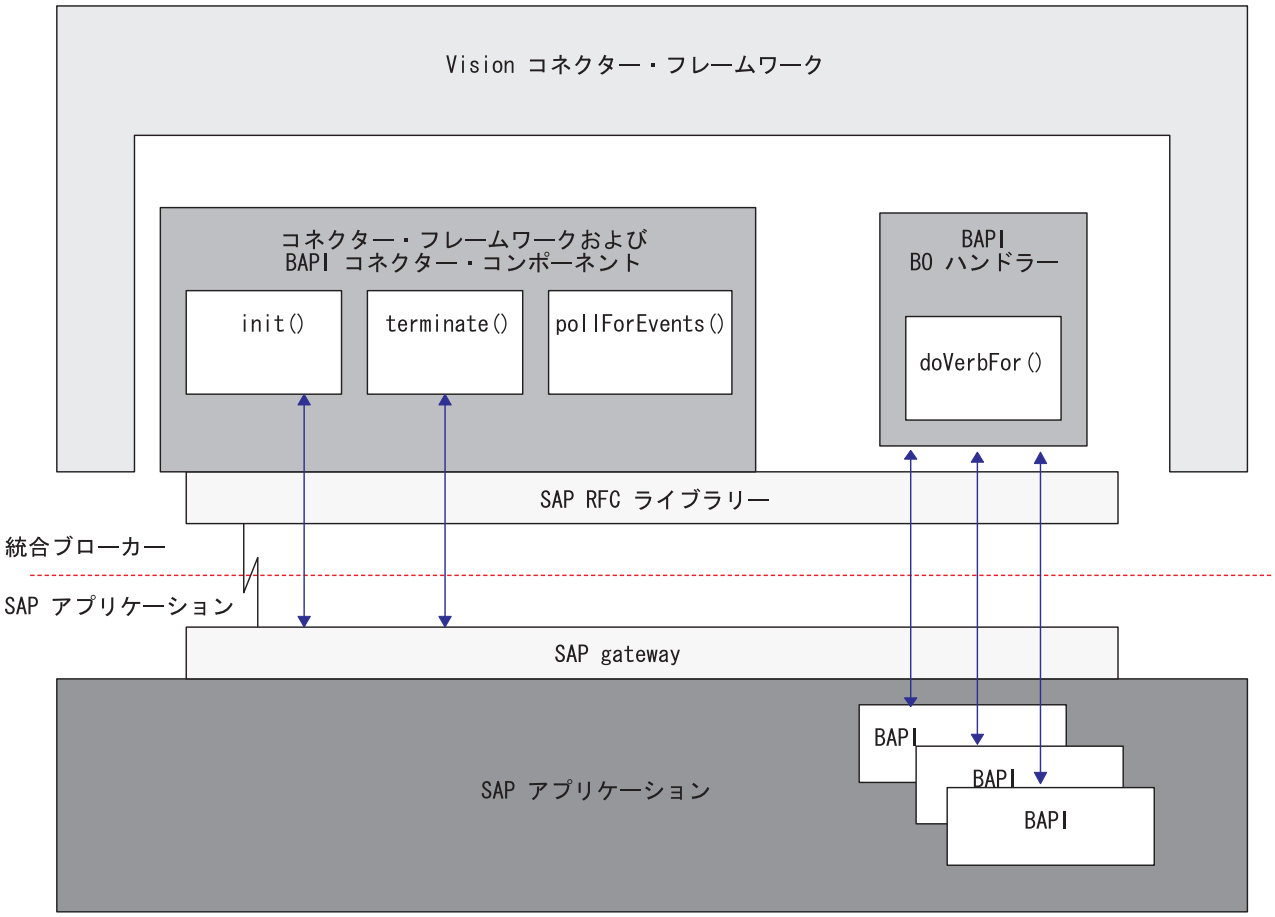


図 46. BAPI Module のアーキテクチャー

BAPI Module のコンポーネントは、以下の動作を実行します。

- SAP RFC ライブラリーと SAP Gateway を使用して、SAP アプリケーションへの RFC 接続をオープンします。
- 統合ブローカーからの要求を処理し、SAP アプリケーションの BAPI を呼び出します。
- SAP アプリケーションへの接続を終了します。

BAPI Module の動作方法

BAPI Module は、`init()`、`terminate()`、`pollForEvents()`、および `doVerbFor()` の各メソッドを実装します。ただし、BAPI Module は要求操作のみをサポートするため、`pollForEvents()` メソッドは使用されません。

初期化と終了

`init()` メソッドは、SAP Gateway を経由した SAP アプリケーションとの RFC 接続を開きます。コネクターは、初期化に失敗すると `terminate()` メソッドを使用して終了します。コネクターは、SAP Gateway への接続を切断することで終了します。

ビジネス・オブジェクトの処理

すべてのビジネス・オブジェクト要求は、Vision コネクター・フレームワークのビジネス・オブジェクト・ハンドラー内にある `doVerbFor()` メソッドの単一の実装で開始されます。 Vision ビジネス・オブジェクト・ハンドラーは、BAPI Module と統合ブローカーとの間で受け渡しされるすべてのビジネス・オブジェクトを処理します。BAPI Module では、1 つの BAPI ビジネス・オブジェクト・ハンドラーですべての BAPI 呼び出しをサポートします。

図 47 に、BAPI Module のビジネス・オブジェクト処理を示します。

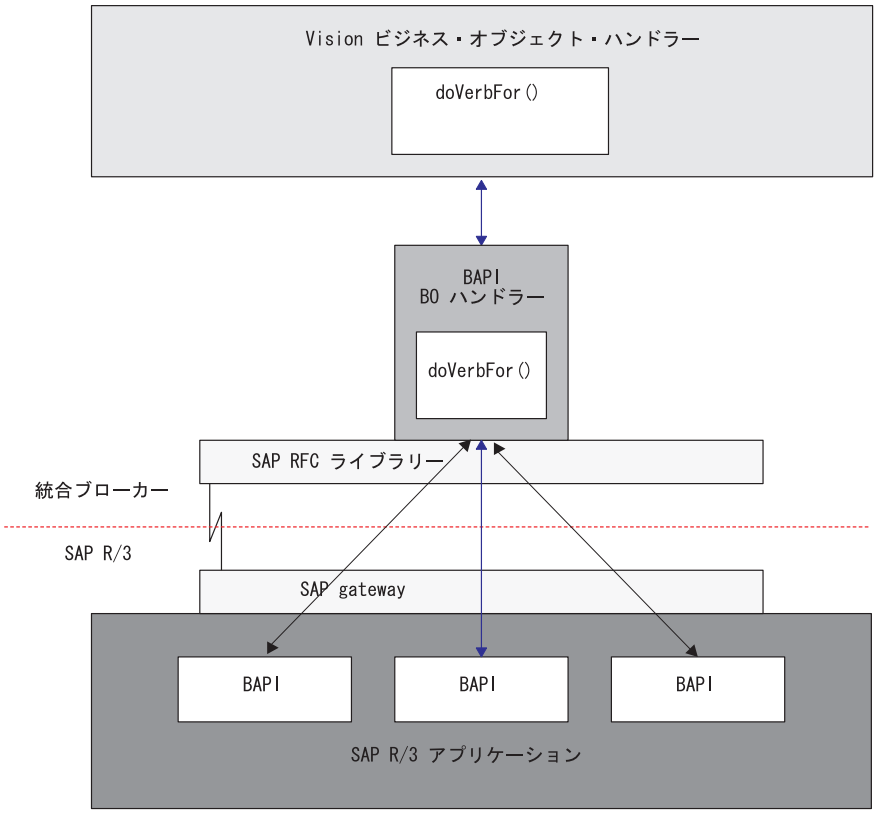


図 47. BAPI Module ビジネス・オブジェクト処理

BAPI ビジネス・オブジェクト・ハンドラーは、Vision ビジネス・オブジェクト・ハンドラーによって呼び出されると、以下の動作を実行します。

1. Vision ビジネス・オブジェクト・ハンドラーから SAP 用の WebSphere ビジネス・オブジェクトを受け取ります。
2. BAPI パラメーターにビジネス・オブジェクト・データを取り込みます。
3. RFC を使用して BAPI 呼び出しを実行し、SAP アプリケーションに BAPI パラメーターを渡します。ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクト・データが戻されるまで待機します。
4. ビジネス・オブジェクト・データ (BAPI パラメーター) を受け取ります。
5. BAPI パラメーターを、WebSphere ビジネス・オブジェクト・データに変換します。

6. ビジネス・オブジェクトを Vision ビジネス・オブジェクト・ハンドラーに渡し、最終的には統合ブローカーに渡します。

注: BAPI Module が Return Structure または Return Table を持つ場合、コネクターはメッセージ・タイプ A (打ち切り) および E (エラー) の有無を調べ、サービス呼び出し要求が正しく処理されたかどうかを判別します。メッセージ・タイプ A または E は、サービス呼び出し要求の処理が失敗したことを示します。BAPI に Return Structure または Return Table がない場合は、独自のエラー処理を実装する必要があります。 Return Structure または Return Table にあるエラー・メッセージ (1 つまたは複数) が、リターン状況記述子の中に返されます。

BAPI のサポート

ビジネス・オブジェクト生成ユーティリティーの SAPODA は、BAPI をサポートするビジネス・オブジェクト定義を生成します。SAPODA は、BAPI のインターフェースを解釈し、そのパラメーターをビジネス・オブジェクト属性にマップし、各属性のアプリケーション固有情報 (ASI) を追加します。

注: 一部の BAPI は、WebSphere ビジネス・オブジェクトの単純属性に対応する単一フィールド・パラメーターを持ちません。コネクターは、すべてのトップレベル・ビジネス・オブジェクトが、キー属性として使用される単純属性を持つことを要求します。そのため、単一フィールド・パラメーターを持たない BAPI からビジネス・オブジェクトおよびビジネス・オブジェクト・ハンドラーを生成すると、SAPODA はトップレベル・ビジネス・オブジェクトに Dummy_key という名前のキー属性を作成し、それにキー属性としてマークを付け、dummy_key をこの属性のアプリケーション固有情報として追加します。Dummy_key は、コネクターにキー属性を提供して、ビジネス・オブジェクトを処理できるようにします。ただしコネクターは、アプリケーション・データを変更する際には Dummy_key 属性の値を無視します。

BAPI トランザクションのサポート

コネクターおよび SAPODA は、SAP BAPI トランザクション (「作業論理単位」とも呼ばれる) をサポートします。BAPI トランザクションは、トランザクション全体を完全に実行するために順番に実行される BAPI の集合から成ります。複数の BAPI のシーケンスを、同じ JCo クライアント接続を使用して呼び出します。

SAPODA は、BAPI トランザクションをサポートするため、トップレベルのビジネス・オブジェクトを生成します。このオブジェクトは、それぞれがトランザクションのシーケンスの中の 1 つの BAPI 呼び出しを表す子ビジネス・オブジェクトの集合のラッパーの役目を果たします。BAPI トランザクションのラッパー・オブジェクトは、完全なトランザクションを表します。第 2 レベルの子ビジネス・オブジェクトはそれぞれ、メソッドの構造体パラメーターまたは表パラメーターを表します。単純属性は、メソッドの単純パラメーターに相当します。

BAPI ビジネス・オブジェクト・ハンドラーは、トランザクション内のすべての BAPI 呼び出しが正常に処理されたとき、SUCCESS を戻します。障害が起きた場合、ビジネス・オブジェクト・ハンドラーは、エラー処理も提供します。トランザクションの BAPI 呼び出しの処理に失敗した場合、エラー・コードによっては、そ

のトランザクションのそれ以降の呼び出しが終了し、BAPI_RETURN によって FAIL または APPRESPONSETIMEOUT が戻されます。

BAPI インターフェースには、トランザクションのロールバック機構はありません。ロールバックは、次のいずれかの方法で実行することができます。

- 以降のすべての BAPI 呼び出しを終了することによって、COMMIT が発生する前にトランザクションを終了する。コネクターは、エラーを検出するとトランザクションを終了します。既に呼び出されている BAPI に組み込みの COMMIT がない場合は、このほかの手順は必要ありません。
- 組み込みの COMMIT がある BAPI の場合、既にコミット済みの作業をロールバックすることが可能な別の BAPI を呼び出す。

ロールバックは、ビジネス・プロセス・ロジックによって処理する必要があります。

BAPI トランザクション・ビジネス・オブジェクトの構造の詳細については、97 ページの『第 8 章 BAPI Module のビジネス・オブジェクトの開発』を参照してください。

第 7 章 BAPI Module の構成

この章では、コネクターをインストールした後、BAPI モジュールを構成する方法について説明します。コネクターのインストールについては、 3 ページの『第 1 章 コネクターのインストール』 を参照してください。

この章の内容は以下のとおりです。

- 『BAPI Module のディレクトリーとファイル』
- 『BAPI Module の構成プロパティー』

BAPI Module のディレクトリーとファイル

BAPI Module のディレクトリーとファイルは、¥connectors¥SAP¥ ディレクトリーに格納されています。表 8 に、BAPI Module で使用されるディレクトリーとファイルを示します。

表 8. BAPI Module のディレクトリーとファイル

ディレクトリー/ファイル名	説明
¥bapi¥client	コネクターのランタイム・ファイルを格納するディレクトリー。このディレクトリーに、独自に作成したすべてのカスタム・ビジネス・オブジェクト・ハンドラーをコピーすることができます。
CWSAP.jar	コネクターのクラス・ファイル。

BAPI Module の構成プロパティー

BAPI Module の運用を開始する前に、構成を行う必要があります。BAPI Module を構成するには、標準およびコネクター固有のコネクター構成プロパティーを設定します。コネクター構成プロパティーの構成の詳細については、 13 ページの『第 2 章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 315 ページの『付録 B. 標準構成プロパティー』 を参照してください。

第 8 章 BAPI Module のビジネス・オブジェクトの開発

この章では、BAPI Module によって処理される 3 種類のビジネス・オブジェクト (単一の BAPI 呼び出し用のビジネス・オブジェクト、BAPI トランザクション用のビジネス・オブジェクト、および BAPI ResultSet オブジェクト) について説明します。

また、ビジネス・オブジェクト生成ユーティリティーの SAPODA が定義を生成する方法についても説明します。この章の読者は、コネクターがビジネス・オブジェクトを処理する方法について十分な知識を持っていることが想定されています。BAPI Module でのビジネス・オブジェクト処理の詳細については、89 ページの『第 6 章 BAPI Module の概要』 を参照してください。

この章の内容は以下のとおりです。

BAPI Module 用のビジネス・オブジェクトを開発するには、ビジネス・オブジェクト・レベルでアプリケーション固有の情報を構成する必要があります。1 つのビジネス・オブジェクト・ハンドラーですべての BAPI をサポートします。 SAPODA では、SAP アプリケーションのネイティブ定義を、統合ブローカー用のビジネス・オブジェクト定義を生成する際のテンプレートとして使用します。

SAP は、コネクターがサポートする標準の動詞 (Create、Update、Delete、および Retrieve) にマップできる、多数のメソッドをサポートします。BAPI で使用される任意のメソッドをサポートする、ビジネス・オブジェクトやビジネス・オブジェクト・ハンドラーを開発できます。

注: SAPODA は、BAPI インターフェースを検索するために、SAP システム内の BAPI にアクセスできる必要があります。

この章では BAPI をサポートするビジネス・オブジェクトを説明しますが、BAPI Module を使用すると、任意の RFC 対応機能をサポートすることができます。

ビジネス・オブジェクトの命名規則

BAPI インターフェースは、単純パラメーター、構造体パラメーター、戻りパラメーター、および表パラメーターから成ります。ここで、

- 構造体パラメーター、単純パラメーター、戻り (インポート) パラメーターは、BAPI に渡されます。

- 構造体パラメーター、単純パラメーター、戻り (エクスポート) パラメーターは、BAPI から渡されます。
- 表 (エクスポート/インポート) パラメーターは両方の方向で渡されます。

BAPI によっては、一部のタイプのパラメーターを持たない場合もあります。例えば、ある BAPI はインポート・パラメーターと表パラメーターのみを持つ、という場合があります。

SAPODA は、表 9に示すように、自動的に BAPI の構造体パラメーターおよび表パラメーターを子ビジネス・オブジェクトにマップし、BAPI の単純パラメーターを、SAP 用の WebSphere ビジネス・オブジェクトの対応する単純属性にマップします。

表 9. 命名規則: SAP 用の WebSphere ビジネス・オブジェクト

ビジネス・オブジェクト	BAPI インターフェース
トップレベル・ビジネス・オブジェクト	*BOprefix_BAPIname*
子ビジネス・オブジェクト	*BOprefix_BAPIParameterName*
属性	*FieldDescription*

注: この章の図では、ビジネス・オブジェクトのプレフィックスとして SAP_ または sap_ を使用します。実際にビジネス・オブジェクト定義を作成する場合は、分かりやすい独自のプレフィックスを指定できます。

SAPODA は、ビジネス・オブジェクト定義内のすべての属性名が固有であることを保証します。1 つの BAPI に同じフィールド記述を持つ複数のパラメーターが存在する場合、SAPODA は生成した属性名にサフィックスとしてカウンターを追加します。

BAPI パラメーターから属性を命名する際に、変更された属性名が次の条件に該当すると、SAPODA はストリングを属性名の前に付加します。

- 数字で開始される場合は、A_ を前に付加します。
- アンダースコアー文字 (_) で開始される場合は、A を前に付加します。

重要: 属性名は、ビジネス・オブジェクト定義を生成した後でいつでも変更できます。しかし、属性名を変更する場合は、アプリケーション固有情報を変更しません。コネクターはこの情報を使用して、属性に対応する BAPI パラメーターを識別します。アプリケーション固有の情報の詳細については、104 ページの『属性レベルの ASI』を参照してください。

ビジネス・オブジェクト構造

コネクターは、単一の BAPI 呼び出し用のビジネス・オブジェクト、BAPI トランザクション用のビジネス・オブジェクト、および BAPI ResultSet オブジェクトの 3 種類の BAPI ビジネス・オブジェクトをサポートします。

単一の BAPI 呼び出し用のビジネス・オブジェクトの構造

単一の BAPI 呼び出し用のビジネス・オブジェクトは、BAPI インターフェースのメソッドを反映します。ビジネス・オブジェクトは、BAPI ビジネス・オブジェク

ト・ハンドラーを使用して、各ビジネス・オブジェクト属性を BAPI パラメーターにマップします。コネクター、各ビジネス・オブジェクト、および BAPI ビジネス・オブジェクト・ハンドラーは、メタデータ主導型です。各ビジネス・オブジェクトのメタデータで提供されるアプリケーション固有情報とビジネス・オブジェクト・ハンドラーを利用することで、コネクターのコードを変更することなく、新しいビジネス・オブジェクトやビジネス・オブジェクト・ハンドラーにコネクターのサポートを追加することができます。その代わりに、次のような処理が実行されます。

- コネクターはトップレベル・ビジネス・オブジェクトの動詞アプリケーション固有情報を使用して、適切なビジネス・オブジェクト・ハンドラーのインスタンスを生成します。
- ビジネス・オブジェクト・ハンドラーは、動詞 ASI を使用して呼び出すべき正しい BAPI を判別します。

ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクト間の単一および複数カーディナリティー関係をサポートします。

BAPI を基にしたビジネス・オブジェクトが格納できる階層のレベルは 2 つまでです。そのため、すべての BAPI 単純パラメーターはトップレベル・ビジネス・オブジェクトの属性に対応し、BAPI 構造体および表パラメーターは子ビジネス・オブジェクトに対応します。

表 10. BAPI と SAP 用のビジネス・オブジェクトの間の対応

BAPI インターフェース・パラメーター	**SAP 用の WebSphere ビジネス・オブジェクト**
単純フィールド	トップレベル・ビジネス・オブジェクトの属性
構造体	単一カーディナリティーの子ビジネス・オブジェクト
テーブル	複数カーディナリティーの子ビジネス・オブジェクト

注: インポート・パラメーターおよびエクスポート・パラメーターは、単純フィールド・パラメーターまたは構造体パラメーターになります。

図 48 に、ビジネス・オブジェクトと BAPI の間の関連を示します。この図には、sap_bapi_salesorder_createfromdat2 ビジネス・オブジェクトのフラグメントが示されています。これは、BAPI_SALESORDER_CREATEFROMDAT2 BAPI に対応します。

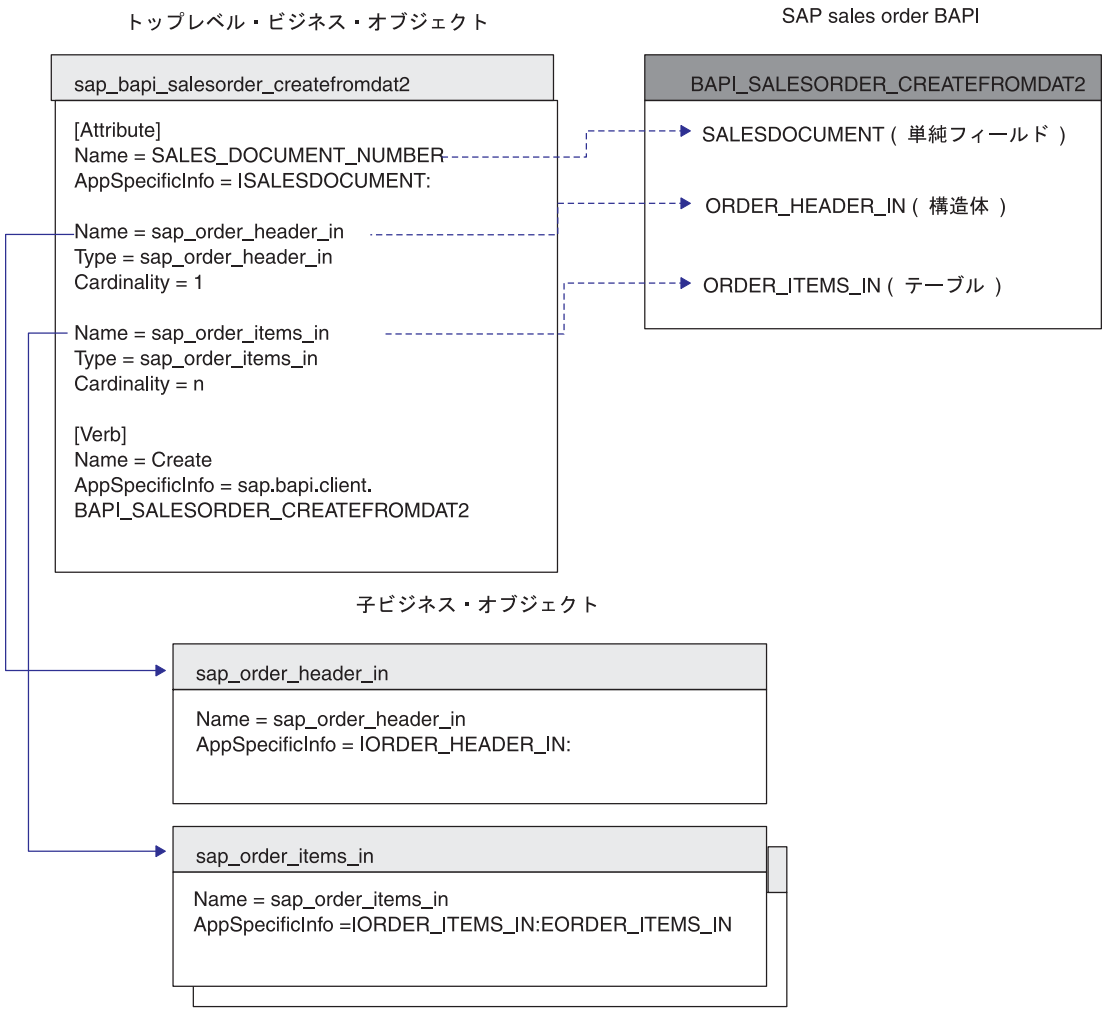


図 48. ビジネス・オブジェクトと BAPI とのマッピング

BAPI トランザクション用のビジネス・オブジェクトの構造

BAPI トランザクションを表すビジネス・オブジェクトは、複数の BAPI オブジェクトを子として含むラッパー・オブジェクトです。ラッパー BAPI トランザクション・オブジェクト内の個別の子 BAPI オブジェクトはそれぞれ、単一の BAPI 呼び出しのパラメーターを表します。SAPODA は、BAPI トランザクション・オブジェクト名に必ず `_txn` というサフィックスを付加します。例えば、`sap_BAPI_salesorder_txn` となります。

サポートされる動詞

BAPI Module は、WebSphere Business Integration システムで使用される標準の動詞 (Create、Update、Delete、および Retrieve) をサポートします。サポートする動詞のそれぞれについて、BAPI にメソッドを関連付けることができます。動詞には、BAPI メソッドによって意味が与えられます。つまり、BAPI 呼び出しには、それに関連付けられる動詞に関係なく、本来備わっている機能性があります。ほとんどの BAPI は、作成、検索、更新、および削除操作のいずれかをサポートします。

ビジネス・オブジェクト属性のプロパティー

トップレベル・ビジネス・オブジェクトの属性のプロパティーは、その属性が単純値を表すか、または子ビジネス・オブジェクトあるいは子ビジネス・オブジェクトの配列を表すかによって異なります。

- 表 11 に、トップレベル・ビジネス・オブジェクトの単純属性のプロパティーを示します。
- 表 12 に、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示します。

SAPODA は、以下の表の説明のように、属性プロパティーを生成します。

表 11. 単純属性プロパティー: トップレベル・ビジネス・オブジェクト

プロパティー名	説明
Name	BAPI パラメーターの記述または名前から派生します。
Type	データのタイプを指定します。SAPODA はこの値を String に設定します。
MaxLength	BAPI パラメーターのフィールド長を指定します。
IsKey	その属性がキーかどうかを指定します。デフォルトでは、ビジネス・オブジェクトの最初の単純属性がキー属性になります。 SAPODA は、単一の BAPI オブジェクトでは、Dummy_key 属性を最初の属性として挿入し、これにキー属性のマークを付けて、適切な値を設定します。BAPI トランザクションと ResultSet の場合、SAPODA はキーとして最初の属性を使用します。
IsForeignKey	SAPODA はこの値を false に設定します。
IsRequired	属性が値を含んでいる必要があるかどうかを指定します。 SAPODA はこの値を false に設定します。
AppSpecificInfo	関連付けられた属性に対応する BAPI パラメーターの名前を格納します。書式は次のとおりです。 I*ABAPFieldName*:E*ABAPFieldName* アプリケーション固有の情報の詳細については、103 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』を参照してください。
DefaultValue	実行時値がない場合にこの属性に割り当てる値を指定します。 SAPODA は、このプロパティーの値を設定しません。

表 12 に、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示します。SAPODA は、以下のプロパティーを生成します。

表 12. 子または子の配列を表す属性のプロパティー

プロパティー名	説明
Name	この値は、構造体または表パラメーターの名前です。書式は BOprefix_BAPIParameterName です。ビジネス・オブジェクト名の中に特殊文字がある場合は、下線文字 _ に置き換えられます。
Type	この値は子ビジネス・オブジェクトのタイプ、つまり BOprefix_BAPIParameterName です。
ContainedObjectVersion	SAPODA はこの値を 3.0.0 に設定します。
Relationship	SAPODA はこの値を containment に設定します。

表 12. 子または子の配列を表す属性のプロパティー (続き)

プロパティー名	説明
IsKey	SAPODA は、BAPI トランザクションまたは ResultSet では、最初の属性の値を true に設定し、その他のすべての属性の値を false に設定します。
IsForeignKey	SAPODA はこの値を false に設定します。
IsRequired	属性が値を含んでいる必要があるかどうかを指定します。SAPODA はこの値を false に設定します。
AppSpecificInfo	関連付けられた属性に対応する BAPI パラメーターの名前を格納します。書式は次のとおりです。 *IBAPIParameterName*:*EBAPIParameterName* アプリケーション固有の情報の詳細については、104 ページの『属性レベルの ASI』を参照してください。
Cardinality	BAPI 構造体パラメーターは単一カーディナリティー (1) を持ち、BAPI 表パラメーターは複数カーディナリティー (n) を持ちます。

重要: 単純属性は、CxIgnore および CxBlank という 2 つの特殊値を持つ場合があります。ビジネス・オブジェクトがサービス呼び出し要求として BAPI Module に送られ、このビジネス・オブジェクト内に CxIgnore または CxBlank に設定された単純属性が存在した場合、それらの属性は BAPI Module から不可視であるのと同じになります。ただし、SAP アプリケーションは、そのような属性をその ABAP データ型に初期化します。BAPI Module は、戻されたすべてのブランク値を CxIgnore に変換します。

初期化属性値

SAP のすべてのフィールドには初期値があります。コネクターがサービス呼び出し要求を受け取ると、ビジネス・オブジェクト・ハンドラーは、ほとんどの BAPI インターフェース・パラメーターに 表 13に示す値を取り込みます。唯一の例外は文字データ型です。ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクト属性での CxIgnore を、SAP フィールドではスペースに変換します。ほかの値を CxIgnore に変換する場合は、ビジネス・オブジェクトを作成するコンポーネントが変換を実行する必要があります。例えば、統合ブローカーが Inter Change Server Express である場合は、この変換を処理するようにマップを変更します。

表 13 に、ビジネス・オブジェクト・ハンドラーが設定する初期値を示します。

表 13. SAP 内のフィールド初期値

データ型	説明	ビジネス・オブジェクト・ハンドラーによって設定される初期値
C	文字	space
N	数字ストリング	*000...*
D	日付 (YYYYMMDD)	00000000
T	時間 (HHMMSS)	000000
X	バイト (16 進数)	X00
I	整数	0
P	パック 10 進数	0

表 13. SAP 内のフィールド初期値 (続き)

データ型	説明	ビジネス・オブジェクト・ハンドラーによって設定される初期値
F	浮動小数点数	0.0

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・オブジェクト定義のアプリケーション固有情報 (ASI) は、ビジネス・オブジェクトの処理方法に関する、アプリケーションに依存した指示を BAPI Module に提供します。この指示は、以下のレベルで指定されます。

- 単一 BAPI 呼び出しオブジェクト、BAPI トランザクション・オブジェクト、および ResultSet オブジェクトの場合は、ビジネス・オブジェクト・レベル
- 単一 BAPI 呼び出しオブジェクト、BAPI トランザクション・オブジェクト、およびカスタム・ビジネス・オブジェクト・ハンドラー (CBOH) の場合は、動詞レベル
- 以下の場合は、属性レベル
  - 単純属性
  - 子オブジェクトを表す属性
  - 子オブジェクト (ResultSet オブジェクト) の配列を表す属性

ビジネス・オブジェクト・レベルの ASI

ビジネス・オブジェクト・レベルの ASI は、表 14 に示すように、オブジェクトのタイプごとに設定する必要があります。

表 14. ビジネス・オブジェクト・レベルの ASI

オブジェクト・タイプ	アプリケーション固有の情報
単一 BAPI 呼び出しを表すオブジェクト	type=bapi
BAPI トランザクションを表すオブジェクト	type=bapi_txn
BAPI ResultSet を表すオブジェクト	type=bapi_resultset

動詞レベルの ASI

動詞レベルの ASI は、表 15 に示すように、オブジェクトのタイプごとに設定する必要があります。

表 15. 動詞レベルの ASI

オブジェクト・タイプ	動詞 ASI
単一 BAPI 呼び出しを表すオブジェクト	verb ASI=*NameOfBAPI*
BAPI トランザクションを表すオブジェクト	verb ASI=*NameOfBAPI1;NameOfBAPI2;NameOfBAPI3*
カスタム・ビジネス・オブジェクト・ハンドラーを表すオブジェクト	CBOH=*bapi.client.customBOHandlerName*

単一の BAPI 呼び出しを表すオブジェクトの後方互換性

コネクターは以前のリリースのビジネス・オブジェクトの動詞 ASI の形式をサポートします。ここでは、AppSpecific プロパティーの値により、BAPI 固有のビジネ

ス・オブジェクト・ハンドラーのクラス名 (verb ASI= bapi.client.*BOHandlerName*。 bapi.client は WebSphere Business Integration の BAPI 固有のビジネス・オブジェクト・ハンドラー名の修飾子で、 *BOHandlerName* は クラスの名前) を取り込みます。ビジネス・オブジェクト・ハンドラーがクライアントとして動作することを示すために、ビジネス・オブジェクト・ハンドラー名の前に値 client を組み込む必要があります。コネクターはこのような以前の形式をサポートしますが、SAPODA は以前の形式を自動的に生成しないため、これを動詞 ASI の中で名前で明示的に指定する必要があります。

例えば、以前のリリースの SALES_ORDER_CREATEFROMDAT2 BAPI をサポートしている場合、アプリケーション固有の情報は次のようになります。

```
AppSpecificInfo = bapi.client.sales_order_createfrom dat2
```

カスタム・ビジネス・オブジェクト・ハンドラーを表すオブジェクトの動詞 ASI

カスタム・ビジネス・オブジェクト・ハンドラーでは、動詞 ASI を明示的に設定 (SAPODA では生成されないため) し、パッケージ名で完全に修飾する必要があります (*bapi.client* がパッケージ名を表します)。

属性レベルの ASI

コネクターは、属性のアプリケーション固有情報の値を使用して、どのインポート・パラメーター、エクスポート・パラメーター、および表パラメーターを使用するかを決定します。このプロパティーの値には、プレフィックス I (インポート・パラメーターの場合) または E (エクスポート・パラメーターの場合) が含まれています。このプレフィックスは、属性値がデータを SAP アプリケーションに渡すために使われるのか、SAP アプリケーションからデータを渡すために使われるのかを示します。

構造体パラメーターはインポートとエクスポートの場合があるため、パラメーター値の前に I または E が使用されます。表パラメーターは、BAPI にデータを渡したり、BAPI からデータを戻したりすることができるため、I と E の両方のパラメーター値を持つことができます。

重要: I および E を使用してパラメーター値を指定するときには、区切り文字としてコロン (:) を必ず使用します。インポート値のみを指定する場合は、値の後にコロンを付ける必要があります。エクスポート値のみを指定する場合は、値の前にコロンを付ける必要があります。両方の値を指定する場合には、インポート値を先に、エクスポート値を後に指定し、両者をコロンで区切ります。

105 ページの図 49 に、ビジネス・オブジェクトと BAPI_EXAMPLE という名前のサンプル BAPI の間の対応を示します。この例では、単純属性 (Attribute_1、Attribute_2、 および Attribute_3) は、インポート・パラメーターまたはエクスポート・パラメーターのみを指定しています。子ビジネス・オブジェクトを表す属性 (Child_1) は、エクスポート構造体パラメーターに対応します。子ビジネス・オブジェクトの配列を表す属性 (Child_2) は、表パラメーターに対応します。

各子ビジネス・オブジェクトには、対応する構造体またはテーブルのフィールドに対応する単純属性 (それぞれ Attribute_11 と Attribute_14) があります。これらのフィールドは、BAPI の詳細を参照することで確認できます。

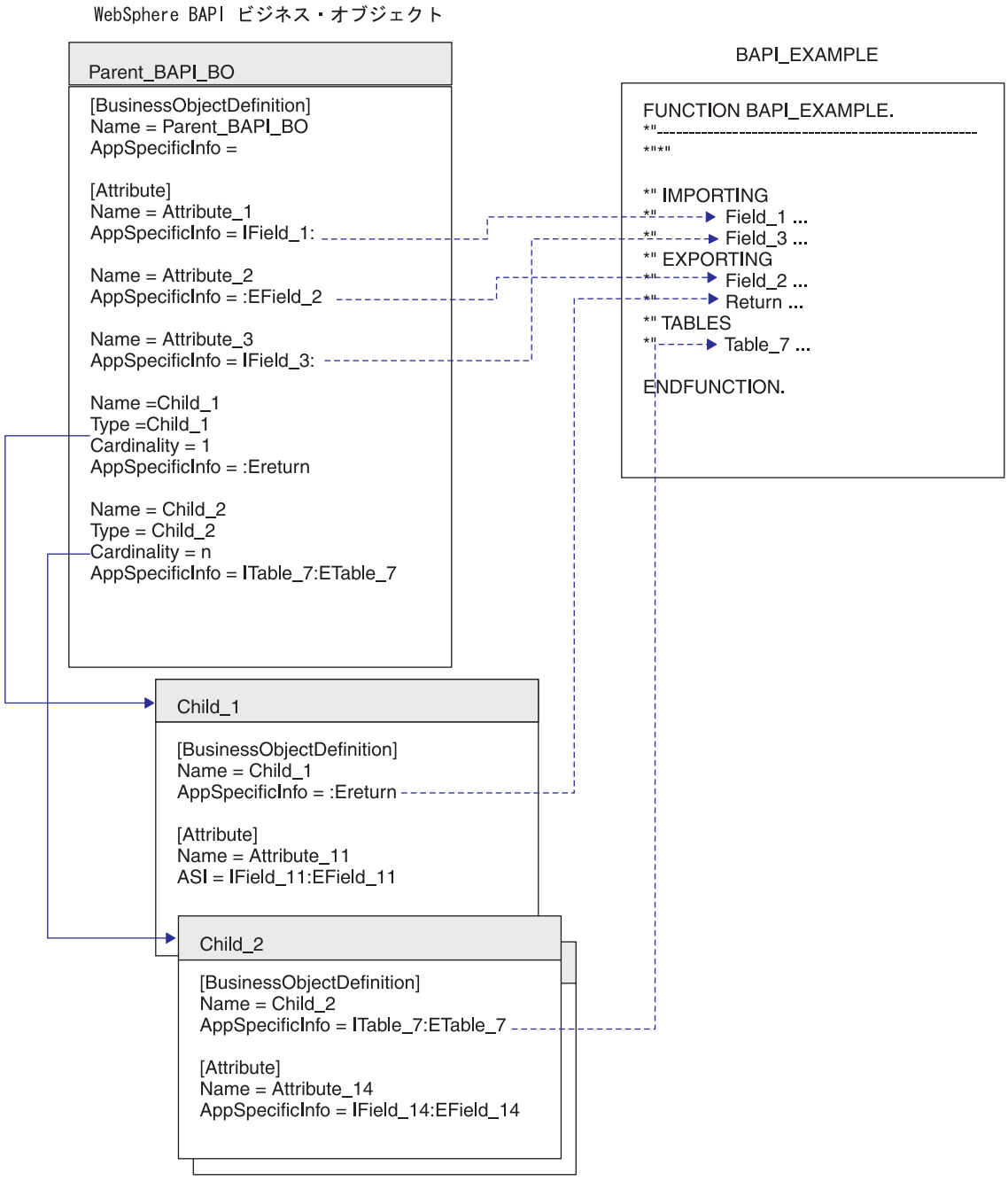


図 49. ビジネス・オブジェクトとサンプル BAPI との対応

表 16 に、特定の種類の属性に対するアプリケーション固有情報の書式を示します。

表 16. 特定の種類の属性に対する AppSpecificInfo の書式

AppSpecificInfo の書式	**属性のタイプ**
IParameterName:*EParameterName*	単純

表 16. 特定の種類の属性に対する AppSpecificInfo の書式 (続き)

AppSpecificInfo の書式	属性のタイプ
I*TableName*:E*TableName*	表パラメーターにマップされる、子ビジネス・オブジェクトを表します。
I*StructureName*:E*StructureName*	構造体パラメーターにマップされる、子ビジネス・オブジェクトを表します。
I*FieldName*:E*FieldName*	表パラメーターまたは構造体パラメーターのフィールドにマップされる、子ビジネス・オブジェクトの属性を表します。

SAPODA は、ビジネス・オブジェクト定義に対する適切なアプリケーション固有情報を自動的に生成します。生成されたアプリケーション固有情報のパラメーター名は、変更しないようにお勧めします。

生成したビジネス・オブジェクト定義の使用

SAPODA は、サポートする必要のある各 RFC 対応機能に対して、ビジネス・オブジェクト定義を生成するために使用します。生成されたオブジェクトは、変更せずにそのまま使用できます。ただし、機能を洗練されたものにするために、これらのオブジェクトを手動で編集できます。

オブジェクトが生成されたら、ビジネス・オブジェクト定義を WebSphere Business Integration システムの実行時環境に追加する必要があります。

- Business Object Designer Express ビジネス・オブジェクト定義をリポジトリーにコピーします。

  注: 統合ブローカーが InterChange Server Express である場合は、`repos_copy` コマンドを使用して、定義をリポジトリーにロードすることができます。

重要: 生成されたビジネス・オブジェクトの名前や、その子ビジネス・オブジェクトの名前を変更することができます。それには、定義を Business Object Designer Express によってではなく、テキスト・ファイルとして編集する必要があります。ビジネス・オブジェクトの名前を変更する場合は、変更する名前に対するすべての参照も必ず変更してください。

注: 開発ネーム・スペースで開発された BAPI や RFC 対応 ABAP 機能の場合、SAPODA は、ビジネス・オブジェクト定義、`.java` ファイル、および `.class` ファイルに名前を付ける際に、機能名の「/」文字を削除するか、または「_」で置き換えます。SAPODA は、名前の先頭の文字である場合に限り、「/」文字を削除します。定義名やファイル名に「/」文字が含まれていなくても、コードは指定した機能を、「/」文字を含んだ正しい名前で正確に呼び出します。また、機能名が数字で開始される場合、SAPODA はストリング `Rfm_` を名前の前に付加します。

ヒントとテクニック

このセクションでは、ビジネス・オブジェクトを開発する際に役立つ、いくつかのヒントとテクニックについて説明します。内容は以下のとおりです。

- 107 ページの『複数のビジネス・オブジェクトに同じリターン・ビジネス・オブジェクトが含まれる場合』

- 107 ページの『生成したビジネス・オブジェクト定義に不要な属性や子ビジネス・オブジェクトが含まれる場合』
- 108 ページの『生成したビジネス・オブジェクト名が長すぎて、命名規則に障害が起こる場合』
- 108 ページの『表パラメーターについて生成された AppSpecificInfo で不要なパラメーターが指定されている場合』

複数のビジネス・オブジェクトに同じリターン・ビジネス・オブジェクトが含まれる場合

ほとんどの BAPI では、リターン・オブジェクトに同じ名前を使用します。SAPODA は、ビジネス・オブジェクト定義を生成する際に、このリターン・オブジェクトを表す子ビジネス・オブジェクトを作成します。複数のビジネス・オブジェクト定義に同じ名前の子ビジネス・オブジェクトが含まれている場合は、その子ビジネス・オブジェクトをリポジトリーに追加できるのは 1 回のみです。つまり、リポジトリー・ディレクトリーにコピーできる定義ファイルは 1 つのみです。

複数のビジネス・オブジェクトにリターン・ビジネス・オブジェクトを含めることができるようにするには、リターン・ビジネス・オブジェクトの名前を各ビジネス・オブジェクトに固有になるように変更する必要があります。

リターン・ビジネス・オブジェクトの名前を変更するには、それを含んでいる各ビジネス・オブジェクト定義で、以下の手順を実行してその定義を変更します。子ビジネス・オブジェクトの定義は、その親と同じ定義ファイル内に含まれています。

子の名前を変更するには、以下の手順を行います。

1. テキスト・エディターで、トップレベル・ビジネス・オブジェクトの定義ファイルを開きます。
2. `BOprefix_return` 子ビジネス・オブジェクトの定義を見つけます。
3. 子の名前を固有な名前に変更します。例えば、テキストに番号を付加します (`sap_return_2`)。
4. 定義内のすべての参照を、その子を新しい名前で参照するように変更します。例えば、子ビジネス・オブジェクトを表すすべての属性の Type プロパティーの値を変更します。
5. 変更した定義ファイルを保存します。
6. System Manager (CSM) を使用して、新しく名前を付けた子ビジネス・オブジェクトをリポジトリーにロードします。

   注: 統合ブローカーが InterChange Server Express である場合は、`repos_copy` コマンドを使用して、定義をリポジトリーにロードすることができます。

生成したビジネス・オブジェクト定義に不要な属性や子ビジネス・オブジェクトが含まれる場合

SAPODA は、すべての BAPI インターフェース・パラメーターを解釈し、それぞれに対応するビジネス・オブジェクト属性または子ビジネス・オブジェクトを作成します。ビジネス・オブジェクト処理のパフォーマンスを向上させるには、すべての不要な属性やビジネス・オブジェクトをビジネス・オブジェクト定義から削除します。

注: SAPODA では、定義を生成する前に、すべてのオプションの属性および子ビジネス・オブジェクトをグラフィカルに削除できます。詳細については、97 ページの『第 8 章 BAPI Module 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・オブジェクト処理のパフォーマンスを向上させるために、アプリケーション固有情報から、すべての不要なインポートおよびエクスポート表パラメーター値を削除することもできます。

その他の変更が必要な場合は、定義が生成された後、Business Object Designer Express を使用して手動でビジネス・オブジェクト定義を編集できます。ただし、使用されないことが確実な属性のみを削除するように十分注意してください。

生成したビジネス・オブジェクト名が長すぎて、命名規則に障害が起こる場合

SAPODA は、BAPI 機能モジュールの名前を使用して、ビジネス・オブジェクト定義の名前を生成します。ビジネス・オブジェクトの名前を変更するには、テキスト・エディターを使用します。

重要: 名前を変更する場合は、その名前に対するすべての参照も必ず変更してください。ただし、生成されたアプリケーション固有情報のパラメーター名は変更しないでください。

生成されたビジネス・オブジェクトの名前を変更するには、以下の手順を行います。

1. 定義をファイルに保管します。
2. テキスト・エディターを使用して、名前を短縮または変更します。
3. System Manager (CSM) を使用して、新しく名前を付けた子ビジネス・オブジェクトをリポジトリーにロードします。

   注: 統合ブローカーが InterChange Server Express である場合は、`repos_copy` コマンドを使用して、定義をリポジトリーにロードすることができます。

表パラメーターについて生成された AppSpecificInfo で不要なパラメーターが指定されている場合

表パラメーターは、インポート・パラメーターとエクスポート・パラメーターの両方になることができます。表パラメーターの値のインポートまたはエクスポートが不要な場合は、アプリケーション固有情報から削除できます。

例えば、作成操作の場合、作成操作が完了した後に SAP アプリケーションから表データを戻す必要がないときには、エクスポート・パラメーター値 (例えば E*table name*) を削除できます。

検索操作の場合は、インポート表パラメーターを指定する必要はありません。したがって、インポート・パラメーター値 (例えば I*table name*) を削除できます。

注: 親ビジネス・オブジェクト内で子を表す属性の AppSpecificInfo や、子ビジネス・オブジェクトのビジネス・オブジェクト・レベルにある AppSpecificInfo からは、不要な値を削除する必要があります。コロン (:) を削除しないでください。

例えば、 105 ページの図 49 の ETable_7 エクスポート・パラメーターを削除するには、以下の手順を行います。

1. Top_Level_BusObj ビジネス・オブジェクトの Child_2 属性で、属性の AppSpecificInfo 値を次のように変更します。

   ```
   ITable_7:
   ```

2. Child_2 ビジネス・オブジェクトのビジネス・オブジェクト・レベルにある AppSpecificInfo で、値を次のように変更します。

   ```
   ITable_7:
   ```

3. 子ビジネス・オブジェクトの各属性の AppSpecificInfo で、例として Attribute_14 を使用するとすれば、値を次のように変更します。

   ```
   IField_14:
   ```

カスタム・ビジネス・オブジェクト・ハンドラーの使用

ビジネス・オブジェクト・ハンドラーを独自に作成するよりも、コネクターにテンプレートとして付属している BAPI ビジネス・オブジェクト・ハンドラーを使用することをお勧めします (詳細については、 91 ページの『ビジネス・オブジェクトの処理』を参照してください)。カスタム・ビジネス・オブジェクト・ハンドラーを使用する理由は以下のとおりです。

- カスタム・エラー処理を実装する。
- このリリースの SAPODA でビジネス・オブジェクトを再生成するのではなく、以前のリリースで作成したビジネス・オブジェクトをサポートする。ビジネス・オブジェクトの後方互換性の詳細については、 103 ページの『単一の BAPI 呼び出しを表すオブジェクトの後方互換性』を参照してください。

カスタム・ビジネス・オブジェクト・ハンドラーの作成

カスタム・ビジネス・オブジェクト・ハンドラーを作成することを決めた場合は、次に示す例をコーディング方法のガイドとして検討してください。 SAPODA は自動的にコードを生成しないことに注意してください。

```
package bapi.client ;


import AppSide_Connector.JavaConnectorUtil;
import CxCommon.BusinessObjectInterface;
import CxCommon.CxMsgFormat;
import CxCommon.CxStatusConstants;
import CxCommon.ReturnStatusDescriptor;
import com.crossworlds.connectors.sap.codegen.BapiBOHandlerBase;
import com.crossworlds.connectors.sap.codegen.exception.CwBoHandlerAppResponseTimeout;
import com.crossworlds.connectors.sap.codegen.exception.CwBoHandlerProcessingFailed;
import com.crossworlds.connectors.sap.visionframework.VisionBOHandlerInterface;
import com.crossworlds.connectors.sap.visionframework.VisionConnectorAgent;
import com.sap.mw.jco.IRepository;
import com.sap.mw.jco.JCO;


public class Bapi_customer_getlist extends BapiBOHandlerBase implements VisionBOHandlerInterface {

  private static VisionConnectorAgent vca = VisionConnectorAgent.getCWSapConnManager();
  private IRepository repository = getRepository();
  private JCO.Function bapicommit = new JCO.Function(repository.
```

```
    getFunctionTemplate("BAPI_TRANSACTION_COMMIT"));
 private int checkBapiReturn = CxStatusConstants.SUCCEED;

 public Bapi_customer_getlist()
 {
 }


 public int doVerbForVision(BusinessObjectInterface theObj, ReturnStatusDescriptor rtn)
 {
    JCO.Function function = new JCO.Function(repository.getFunctionTemplate("BAPI_CUSTOMER_GETLIST"));

    //Default processing to failure
    int    mReturnCode = CxStatusConstants.FAIL;
    JCO.Client theClient = null;

    try
    {
    traceMessage(JavaConnectorUtil.LEVEL1, 27025, CxMsgFormat.XRD_INFO, theObj.getName(), theObj.getVerb(),
null, null);

    // get defaults and check for required fields for Create and Update only
    if ((theObj.getVerb().equalsIgnoreCase("Create")) || (theObj.getVerb().equalsIgnoreCase("Update")))
    {
      traceMessage(JavaConnectorUtil.LEVEL4, 27021, CxMsgFormat.XRD_INFO, theObj.getName(),
  null, null, null);
      JavaConnectorUtil.initAndValidateAttributes(theObj);
    }

    // get a new connection to Sap
    theClient = this.getClient();

    //   populate Rfc interface parameters
    mReturnCode = doBusObjtoRfcData(theObj, function, function.getImportParameterList(),"I",false);
    if (mReturnCode != CxStatusConstants.SUCCEED)
    {
      if (theClient != null) vca.releaseClient(theClient);
      return CxStatusConstants.FAIL;
    }

    // Execute Rfc Call
    this.callBapi(theClient,function);

    try {
    // After successful RfcCall: check Structure/Table RETURN for Bapi Return Codes E or A that
    // indicate failure
      this.checkBapiRc(function);

    // After successful Bapi Call: call BAPI_TRANSACTIOM_COMMIT
      this.callBapiCommit(theClient);

    } catch (CwBoHandlerProcessingFailed cwpf) {
      rtn.setErrorString(cwpf.getMessage());
      rtn.setStatus(CxStatusConstants.FAIL);
      checkBapiReturn = CxStatusConstants.FAIL;
    }
    // Now create CW Business Object
    mReturnCode = doRfcDatatoBusObj(theObj, function , function.getExportParameterList(), "E");
    if (mReturnCode != CxStatusConstants.SUCCEED || checkBapiReturn != CxStatusConstants.SUCCEED)
    {
    if (theClient != null) vca.releaseClient(theClient);
      return CxStatusConstants.FAIL;
    }

    // Clean up
    if (theClient != null) vca.releaseClient(theClient);

    // Finally, return success to ICS
    traceMessage(JavaConnectorUtil.LEVEL1, 27034, CxMsgFormat.XRD_INFO, theObj.getName(),
       null, null, null);
    return CxStatusConstants.VALCHANGE;
    }//end of try

    catch (CxCommon.Exceptions.SetDefaultFailedException sdfe)
    {
      if (theClient != null) vca.releaseClient(theClient);
      String msg = logMessage(20059, CxMsgFormat.XRD_INFO, theObj.getName(), null, null, null);
      rtn.setErrorString(msg);
      rtn.setStatus(CxStatusConstants.FAIL);
      return CxStatusConstants.FAIL;
    }
    catch (CxCommon.Exceptions.BusObjSpecNameNotFoundException c)
    {
      if (theClient != null) vca.releaseClient(theClient);
      String msg = logMessage(20059, CxMsgFormat.XRD_INFO, theObj.getName(), null, null, null);
      rtn.setErrorString(msg);
      rtn.setStatus(CxStatusConstants.FAIL);
      return CxStatusConstants.FAIL;
```

```
    }
    catch (CwBoHandlerAppResponseTimeout art)
    {
      if (theClient != null) vca.releaseClient(theClient);
      rtn.setErrorString(art.getMessage());
      rtn.setStatus(CxStatusConstants.APPRESPONSETIMEOUT);
      return CxStatusConstants.APPRESPONSETIMEOUT;
    }
    catch (CwBoHandlerProcessingFailed cwpf)
    {
      if (theClient != null) vca.releaseClient(theClient);
       rtn.setErrorString(cwpf.getMessage());
      rtn.setStatus(CxStatusConstants.FAIL);
      return CxStatusConstants.FAIL;
    }
    catch (Exception e)
    {
      if (theClient != null) vca.releaseClient(theClient);
       rtn.setErrorString(e.getMessage());
      rtn.setStatus(CxStatusConstants.FAIL);
      return CxStatusConstants.FAIL;
    }
    catch ( OutOfMemoryError merr ){                // CR 30185
      String msg = logMessage(23060, JavaConnectorUtil.XRD_ERROR,
        "doVerbForVision()", merr.toString(), null, null);
      rtn.setErrorString(msg);
      rtn.setStatus(CxStatusConstants.APPRESPONSETIMEOUT);
      return CxStatusConstants.APPRESPONSETIMEOUT;
    }
    catch ( StackOverflowError serr ){                // CR 30185
      String msg = logMessage(23060, JavaConnectorUtil.XRD_ERROR,
         "doVerbForVision()", serr.toString(), null, null);
      rtn.setErrorString(msg);
      rtn.setStatus(CxStatusConstants.APPRESPONSETIMEOUT);
      return CxStatusConstants.APPRESPONSETIMEOUT;
    }
    catch (Throwable ex) {
      // 23046 EXception raised in: {1} : ErrorMessage: {2}.
      String msg = logMessage(23046, JavaConnectorUtil.XRD_ERROR,
        "doVerbForVision()",ex.toString(), null, null);
      return CxStatusConstants.FAIL;
    }
  }// end of doVerbforVision

public int callBapiCommit(JCO.Client theClient) throws
    CwBoHandlerProcessingFailed, CwBoHandlerAppResponseTimeout
{

  int mStatus = CxStatusConstants.FAIL;
  try {
  traceMessage(JavaConnectorUtil.LEVEL3, 27032, CxMsgFormat.XRD_INFO, null, null, null, null);
  try
  {
     theClient.execute(bapicommit);
  }
  catch (JCO.Exception e)
  {
     String msg = logMessage(27019, CxMsgFormat.XRD_ERROR, null, null,
      null, null);
       throw new CwBoHandlerProcessingFailed(msg);
  }
  //  Read Return
  readBapiRc(bapicommit.getExportParameterList().getStructure("RETURN"));
  }// end of try
  catch (Exception e)
  {
     String msg = logMessage(27019, CxMsgFormat.XRD_ERROR, null, null,
       null, null);
     throw new CwBoHandlerProcessingFailed(e.getMessage());
  }

  // Return Success
  return CxStatusConstants.SUCCEED;
}


public void checkBapiRc(JCO.Function function) throws CwBoHandlerProcessingFailed
{

  try
  {
     JCO.ParameterList p = function.getTableParameterList();
       this.readBapiRc(p.getTable("RETURN"));
     }
     catch (CwBoHandlerProcessingFailed cw)
     {
       throw cw;
     }
```

```
    catch (Exception e)
    {
      try
      {
      JCO.ParameterList p = function.getExportParameterList();
      if (p != null)
      this.readBapiRc(p.getStructure("RETURN"));
      else
      {
      String msg = logMessage(27045, CxMsgFormat.XRD_INFO, null,null, null, null);
      throw new CwBoHandlerProcessingFailed(msg);
      }
      }//end try
    catch (JCO.Exception o)
      {
      String msg = logMessage(27045, CxMsgFormat.XRD_INFO, null,null, null, null);
      throw new CwBoHandlerProcessingFailed(msg);
      }
  }

}// end of checkBapiRc


public void readBapiRc(JCO.Structure Return) throws CwBoHandlerProcessingFailed
{
  String type = null;
  String no = null;
  String message = null;
  traceMessage(JavaConnectorUtil.LEVEL4, 27033, CxMsgFormat.XRD_INFO, null, null, null, null);
  if (Return.getString("TYPE") != null)
  {
     // Depending on RETURN ddic structure, number field is either "NUMBER" or "CODE".
     try
     {
       no = Return.getString("NUMBER");
     }
     catch (JCO.Exception o)
     {
       no = Return.getString("CODE");
     }
     message = Return.getString("MESSAGE");
     type = Return.getString("TYPE");
     if ((type.equalsIgnoreCase("A")) || (type.equalsIgnoreCase("E")))
     {
       String msg = logMessage(27015, CxMsgFormat.XRD_ERROR, type, no,
         message, null);
       throw new CwBoHandlerProcessingFailed(msg);
     }
     else
     {
     traceMessage(JavaConnectorUtil.LEVEL1, 27016, CxMsgFormat.XRD_INFO,
       type, no, message, null);
     return;
     }
  }
  // If structure Return is empty, it will be assumed that processing was
     successful
     traceMessage(JavaConnectorUtil.LEVEL1, 27036, CxMsgFormat.XRD_INFO, type, no, message, null);
}



public void readBapiRc(JCO.Table Return) throws CwBoHandlerProcessingFailed
{
  String type = null;
  String no = null;
  String message = null;
  String msg = null;
  String errorMsg = "";
  int mStatus = CxStatusConstants.SUCCEED;

  traceMessage(JavaConnectorUtil.LEVEL4, 27033, CxMsgFormat.XRD_INFO, null, null, null, null);
  for (int i=0; i<Return.getNumRows(); i++)
  {
  try
  {
  Return.setRow(i);
  if (Return.getString("TYPE") != null)
  {
     type = Return.getString("TYPE");
     // Depending on RETURN ddic structure, number field is either "NUMBER" or "CODE".
     try
     {
       no = Return.getString("NUMBER");
     }
     catch (JCO.Exception e)
     {
```

```
      no = Return.getString("CODE");
    }
    message = Return.getString("MESSAGE");
    }//end if
    }//end try
    catch (Exception o)
    {
  // Could not interprete Return structure ==> Failing event
      msg = logMessage(27043, CxMsgFormat.XRD_ERROR, type, no,
       message, null);
      throw new CwBoHandlerProcessingFailed(msg);
    }
    if  (type != null)
    {
    if ((type.equalsIgnoreCase("A")) || (type.equalsIgnoreCase("E")))
    {
      msg = logMessage(27015, CxMsgFormat.XRD_ERROR, type, no,
       message, null);
      mStatus = CxStatusConstants.FAIL;
    }//end if
    else
    {
    traceMessage(JavaConnectorUtil.LEVEL1, 27044, CxMsgFormat.XRD_INFO,
        type, no, message, null);
    }
    }//end if type != null
  }//end of for
  if (mStatus == CxStatusConstants.FAIL)
    {
      logMessage(27046, CxMsgFormat.XRD_ERROR, msg, null, null, null);
      throw new CwBoHandlerProcessingFailed(msg);
    }
  // If Return is empty, it will be assumed that processing was successful
  if (Return.getNumRows() <= 0)
     traceMessage(JavaConnectorUtil.LEVEL1, 27036, CxMsgFormat.XRD_INFO, type, no, message, null);
  else
  traceMessage(JavaConnectorUtil.LEVEL1, 27011, CxMsgFormat.XRD_INFO, null, null, null, null);
}//end of ReadBapiRc


};
```

以下の例では、Windows プラットフォームでカスタム BOHandler をコンパイルするスクリプトを示しています。このスクリプトを実行するためには、マシン上に JDK をインストールしておく必要があります。

```
REM @echo off REM

REM init environment
call "%CROSSWORLDS%"¥bin¥CWODAEnv.bat
setlocal

REM set classpaths
set WBIA=%CROSSWORLDS%"¥lib¥WBIA¥4.2.0¥WBIA.jar
set CWLIB=%CROSSWORLDS%"¥lib¥CrossWorlds.jar
set AGENT="%CROSSWORLDS%"¥ODA¥SAP¥SAPODA.jar
set JCO_JAR="%CROSSWORLDS%"¥ODA¥SAP¥sapjco.jar

set JCLASSES=%AGENT%;%JCO_JAR%;%CWLIB%;%WBIA%
echo classpath = %JCLASSES%

REM compile the BAPI BOHandler passed as argument
javac -verbose -classpath %JCLASSES% %1

endlocal
pause
```

注: コネクターには、生成済み (カスタムでない) ビジネス・オブジェクト・ハンドラーを使用する既存のインストール済み環境用に、完全な後方互換性があります。

第 3 部 ALE Module

第 9 章 ALE Module の概要

この章では、Connector for mySAP.com. の ALE (Application Link Enabling) Module について説明します。ALE は、SAP のビジネス・フレームワーク内の統合層の一部です。ALE Module により、ビジネス・プロセスの統合、および複数の SAP システム間または SAP と外部システムの間での非同期データ通信が可能になります。

この章の内容は以下のとおりです。

ALE テクノロジーの概要

ALE Module は、本質的に非同期であるバッチ・オブジェクトなどのオブジェクトに最も使用されています。プッシュ・テクノロジーを使用するため、サーバーがイベントを listen している必要があります。登録およびインストールという処理により、listen する対象と情報の送信元がサーバーに通知されます。登録時には、プログラム ID を使用し、リスナー・スレッドとの通信点 (サーバー) を SAP Gateway に指定します。サーバー内の機能モジュール定義は、このデータをテンプレートとして提供することにより、SAP からプッシュされたデータを解釈します。

ALE モジュールは、イベント処理に RFC Server モジュールを使用します。ALE Module は、トランザクション ID (TID) および IDoc 管理に WebSphere MQ キューを使用します。コネクターは、SAP からコネクターへのデータを処理するときにサブスクリプションを検査するため、トランザクションはコラボレーションが開始されるまで SAP に残ります。

- 統合ブローカーは SAP に対してアダプター・ビジネス・オブジェクトを送信します。ビジネス・オブジェクトのデータは、コネクターに対する処理要求を表します。コネクターは、ビジネス・オブジェクトを SAP Intermediate Document (IDoc) 形式と互換性のある表形式に変換します。コネクターでは、ALE インターフェースへの Remote Function Call (RFC) を使用して、IDoc データを SAP システムに渡します。
- コネクターは、アプリケーション・イベントを表すデータを IDoc 表形式で SAP から受け取ります。コネクターは統合ブローカーへ送信する前に、データを SAP 用の アダプター・ビジネス・オブジェクトに変換します。コネクターは、ALE Module への RFC を使用して、ALE インターフェースからのデータを受け取ります。

重要: バージョン 4.8.2 以前のコネクターのリリースでは、コネクターはコラボレーション、ビジネス・オブジェクトおよびマップを使用して Transaction ID (TID) とその状況をリポジトリーに格納し、ローカル・ファイル・システムを使用して IDoc データを格納します。バージョン 4.8.2 のコネクターでは、TID と IDoc データを使用する以前の管理方法から WebSphere MQ キューを使用する方法に変更しています。

注: ALE Module は非同期通信を使用するため、相互参照が必要な場合には使用できません。

ALE Module のコンポーネント

ALE Module は Java で記述されており、Vision コネクター・フレームワークを拡張します。このモジュールは、以下のものから構成されます。

- コネクター・フレームワーク
- コネクターの ALE 用アプリケーション固有コンポーネント
- ALE ビジネス・オブジェクト・ハンドラーの 2 つのクラス (イベント処理用と要求処理用)
- SAP RFC ライブラリー
- SAP SAP JCo コネクター
- RFC Server 用アプリケーション固有コンポーネント (イベント処理用のみで使用)

  ALE Module と RFC Server コネクター・コンポーネントは、どちらも SAP アプリケーションからの直接の RFC 呼び出しをサポートし、イベント処理の方法が類似していることから、ALE Module では RFC Server コネクター・コンポーネントを使用しています。

SAP では、Java および C で記述された RFC ライブラリーを提供しています。コネクターは、JAR ファイルとして提供され、実行されます。

119 ページの図 50 に、ALE Module のアーキテクチャーを示します。

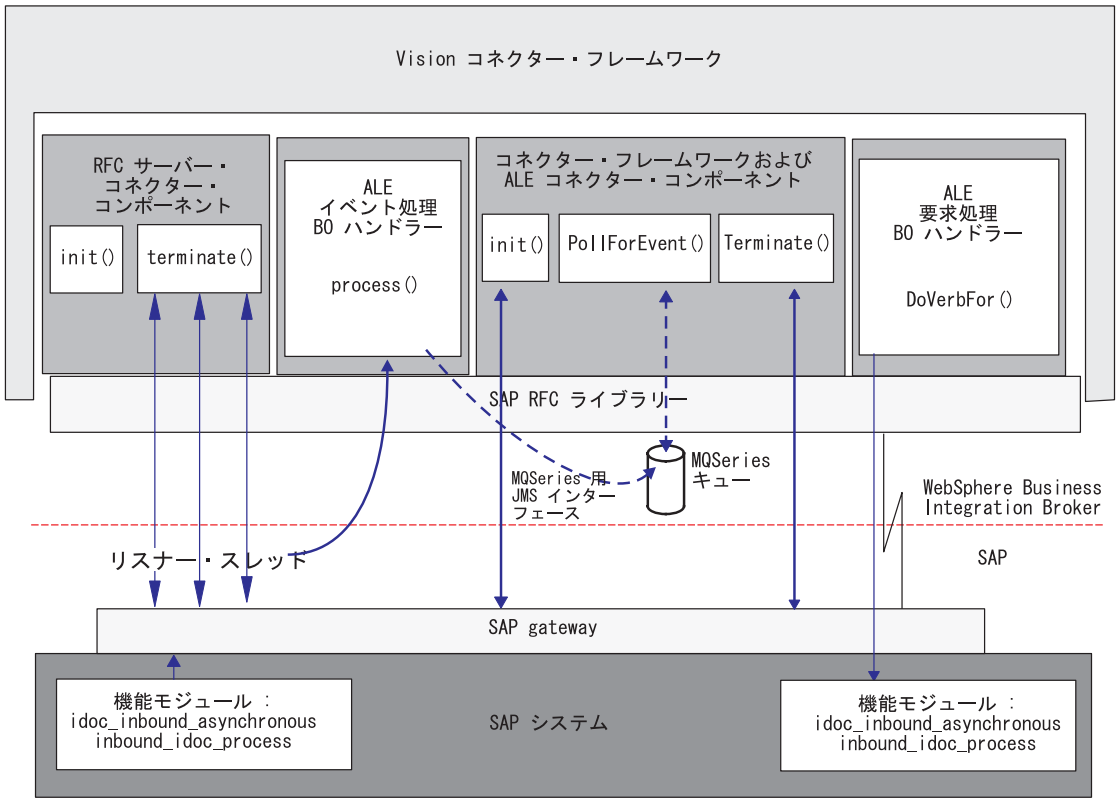


図 50. ALE Module のアーキテクチャー

イベント処理コンポーネント

SAP からのイベントを処理する場合、コネクターは 図 50 に示したコンポーネントを次のように使用します。

- Vision コネクター・フレームワークは RFC Server コネクター・コンポーネントを始動します。これにより、リスナー・スレッドが作成されます。各リスナー・スレッドは RFC ライブラリーと SAP Gateway を使用して、SAP アプリケーションへの単一のハンドルを登録します。
- リスナー・スレッドは、SAP アプリケーションからのイベントを処理します。

  イベントとは、データをリスナーに転送する ABAP 機能を実行することです。SAP によって送信されるイベント・データは、1 つ以上のそのような異種の実行を表す場合があります。

  SAP からの各イベントは、トランザクションと見なされます。コネクターは、各イベントを処理するために、Transaction ID (TID) を利用した 2 ステップのプロセスを使い、データが SAP からコネクターに 1 回に限ってデリバリーされることを保証します。
- WebSphere MQ キューは各イベントに対する JMS-MQ メッセージを永続的に保管します。それぞれの JMS-MQ メッセージでは、イベントを識別する TID、TID の状況、イベントに関連した IDoc データ、および IDoc の処理状況を保管します。

- コネクターのポーリング・プロセスでは、保管されたイベント・メッセージから WebSphere ビジネス・オブジェクトを作成し、そのビジネス・オブジェクトを統合ブローカーへ送信します。コネクターは、ラージ・ビジネス・オブジェクトとなる大きな IDoc メッセージをサポートします。

  より効果的な処理パフォーマンスのために、コネクターは大きな IDoc をそれより小さいパーツに分割します。それぞれのパーツは JMS-MQ メッセージとして、より小さいビジネス・オブジェクトに変換されます。これらのメッセージはそれぞれ、`MultiPartMessage` プロパティーを持っており、このプロパティーによって、そのメッセージがそれより大きなメッセージの対応するパーツであることが識別されます。例えば、元の大きな IDoc を 8 つの JMS-MQ メッセージに分割するとすれば、メッセージごとの各パーツの `MultiPartMessage` プロパティーの値は `1/8`、`2/8` などとなります。すべてのメッセージ・パーツを相互に関連付けるため、コネクターは、最初のメッセージ以外の各メッセージ・パーツの `CorrelationID` ヘッダー・プロパティーを最初のパーツの `JMSMessageID` プロパティーの値に設定します。さらに、最初のパーツの `CorrelationID` プロパティーを、必ず元の大きな IDoc に対応付けられている最初の JMS-MQ メッセージの CorrelationID プロパティーの値に設定します。JMS-MQ メッセージ・プロパティーの詳細については、132 ページの表 19 を参照してください。
- Business Integration システムは、未処理のイベントを追跡して、統合ブローカーまたはコネクターがダウンした場合にそれらのイベントをリカバリーできるようにします。統合ブローカーまたはコネクターがリストアされると、コネクターはこれらのイベントを自動的に再サブミットします。

要求処理コンポーネント

統合ブローカーからの要求を処理する場合、コネクターは 図 50 に示したコンポーネントを次のように使用します。

- ALE Module は SAP RFC ライブラリーおよび SAP Gateway を使用して、SAP アプリケーションへの RFC 接続を開きます。
- ALE 要求処理ビジネス・オブジェクト・ハンドラーは、統合ブローカーからの要求を、ビジネス・オブジェクト形式から SAP IDoc 形式に基づく IDoc データに変換する処理を行います。
- アプリケーションに送られるすべての要求について、ALE Module は Transaction ID (TID) を JMS-MQ メッセージの中の TID キューに永続的に保管します。TID は、要求が 1 回に限ってデリバリーされることを保証しています。ただし、統合ブローカーが、トランザクション ID 属性の中で同じ値を持つオブジェクトを送信する場合は、このオブジェクトは再度処理されることになります。オブジェクトがいったん正常に送信されたあとは、統合ブローカーが再度オブジェクトを送信することはないと予期されます。
- ALE Module は、SAP アプリケーションへの接続を解放します。

リスナー・スレッド

リスナー・スレッドは、以下の動作を行います。

- プログラム ID を使用して、SAP Gateway に登録します。

- サポートする ALE 固有の RFC 対応機能を、SAP Gateway に対して識別します。これらの機能は、idoc_inbound_asynchronous および inbound_idoc_process です。
- ALE 固有機能からイベントを受け取ります。
- イベント処理 ALE ビジネス・オブジェクト・ハンドラーのインスタンスを生成します。

スレッドは、それがサポートする ALE 固有機能からのイベントを、同期的な方法で連続的に listen します。

Transaction IDs

SAP は、トランザクションとそれに対応する ID を使用してイベントを作成し、個々のデータが 1 回に限って SAP からデリバリーされることを保証します。SAP は、イベント・データとともに Transaction ID (TID) を送信します。イベント処理と要求処理に対する TID を集中管理するために、コネクターは各 TID を JMS-MQ メッセージとして WebSphere MQ キューに格納します。イベントを処理する場合、コネクターは関連する IDoc データをメッセージ・ボディとしても格納します。コネクターは、TID、TID 状況、および IDoc の処理状況をメッセージ・ヘッダーに格納します。

ALE 固有ビジネス・オブジェクト・ハンドラー

ALE 固有ビジネス・オブジェクト・ハンドラーは、イベント処理用と、要求処理用の 2 種類が提供されています。

イベント処理ビジネス・オブジェクト・ハンドラー

リスナー・スレッドは、以下を行うイベント処理ビジネス・オブジェクト・ハンドラーのインスタンスを生成します。

- SAP から RFC イベント・データを検索します。
- JMS-MQ メッセージを作成し、SAP からイベントとともに送信されたトランザクション ID を永続的に保管して管理します。
- SAP から JMS-MQ メッセージ内に受信した 1 つ以上の IDoc のデータを保管します。
- SAP Gateway を経由して、ALE 固有機能に応答を戻します。この応答は、トランザクションが完了したことを示します。

要求処理ビジネス・オブジェクト・ハンドラー

Vision コネクター・フレームワークは ALE 要求処理ビジネス・オブジェクト・ハンドラーをインスタンス化します。このハンドラーは、SAP 用の WebSphere ビジネス・オブジェクトにある TransactionId 属性の値をチェックします。この値が存在する場合は、以下のステップを続行します。

1. JMS-MQ メッセージまたは SAP のいずれかから TID を取得します。
2. ビジネス・オブジェクト・データを、SAP への RFC 呼び出しに必要な機能モジュール・インターフェースによって定義された IDoc データ形式に変換します。
3. ALE インターフェースへの RFC 呼び出しを実行します。
4. JMS-MQ メッセージ内でこの要求に対する TID の状況を更新します。

5. 成功応答を統合ブローカーに戻します。

SAP 用のビジネス・オブジェクトの構造

SAP 用 WebSphere ビジネス・オブジェクトはそれぞれの IDoc を、制御レコード・ビジネス・オブジェクトとデータ・レコード・ビジネス・オブジェクトという 2 つの子ビジネス・オブジェクトを含む親ラッパー・ビジネス・オブジェクトとして表します。制御レコード・ビジネス・オブジェクトには、コネクターがビジネス・オブジェクトを処理するために必要とするメタデータが含まれています。データ・レコード・ビジネス・オブジェクトには、SAP アプリケーションによって処理される実際のビジネス・オブジェクト・データと、コネクターがそれを RFC 呼び出し用の IDoc 構造体に変換するために必要なメタデータが含まれています。

コネクターには、制御レコード用のビジネス・オブジェクト定義が含まれています。定義ファイル sap_idoccontrol.xsd は、¥repository¥SAP ディレクトリーに配置されます。

制御レコード・ビジネス・オブジェクトの TABNAM 属性は、以下のように、親ラッパー・ビジネス・オブジェクトがどの SAP 機能モジュールを呼び出すかを示します。

- EDI_DC40 の値は、idoc_inbound_asynchronous 機能モジュールを示します。これは、コネクターが SAP 4x 用にのみ使用するものです。
- EDI_DC の値は、inbound_idoc_process 機能モジュールを示します。これは、SAP 3x との後方互換性のために提供されているものです。

さらに、SAP が ALE のオブジェクトを正しく処理するためには、以下の属性に値を設定する必要があります。これらの値は ALE 構成に基づいて設定します。

- Name_of_table_structure
- Client
- Name_of_basic_type
- Logical_message_type
- Partner_type_of_sender
- Partner_number_of_sender
- Partner_type_of_recipient
- Partner_number_of_recipient

両方のビジネス・オブジェクトの DOCNUM 属性は、データ・レコード・ビジネス・オブジェクトと制御レコード・ビジネス・オブジェクトとの関係を確立します。

サービス呼び出し要求を処理する場合には、ALE Module は単一のビジネス・オブジェクト内の複数の IDoc を処理できます。ただし、この処理を行えるようにするには、2 つ以上の親ラッパー・ビジネス・オブジェクトに対して別の複数 IDoc ラッパー・ビジネス・オブジェクトを追加する必要があります。このトップレベルの複数 IDoc ラッパー・ビジネス・オブジェクトには、親ラッパー・ビジネス・オブジェクトの配列を表す属性が含まれています。詳細については、145 ページの『親ラッパー・ビジネス・オブジェクト』を参照してください。

コネクターには、ビジネス・オブジェクトの生成ツールである SAPODA が組み込まれています。このツールは、IDoc 定義テキスト・ファイルを使用して ALE Module 用のビジネス・オブジェクト定義を生成します。ALE Module 用のビジネス・オブジェクトの開発の詳細については、 141 ページの『第 11 章 ALE Module のビジネス・オブジェクトの開発』および 35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

第 10 章 ALE Module の使用

この章では、ALE Module の構成および使用について説明します。この章で説明する構成作業を行う前に、Adapter for mySAP.com のコネクター・コンポーネントをインストールしておく必要があります。

コネクターのインストールの詳細については、136 ページの『ALE Module Queue ユーティリティーのインストールおよび構成』を参照してください。

この章の内容は以下のとおりです。

ALE Module 実行の前提条件

コネクターがイベント処理中に TID および IDoc データを永続的に保管でき、要求処理中に TID を永続的に保管できるようにするには、以下のことを行う必要があります。

- システム上に以下のものがインストールされており、稼働していることを確認します。
  - WebSphere MQ (含まれません)
  - TCP/IP
- イベント処理の場合、以下の WebSphere MQ キューを作成します。キューの名前は、対応するコネクター固有の構成プロパティーで指定します。
  - アーカイブ (SAPALE_Archive_Queue プロパティー)
  - イベント (SAPALE_Event_Queue プロパティー)
  - 処理中の作業 (WIP) (SAPALE_Wip_Queue プロパティー)
  - エラー (SAPALE_Error_Queue プロパティー)
  - アンサブスクライブ (SAPALE_UnSubscribed_Queue プロパティー)
  - TID (SAPtid_Queue プロパティー)

  コネクターがこれらのキューを使用する方法の詳細については、129 ページの『ALE Module の実行』を参照してください。
- ALE Module を使用して大規模な IDoc または IDoc パケットを処理するには、以下のようにします。
  - WebSphere MQ キュー・マネージャーの最大メッセージ長とそのキューを増やします。この長さのデフォルト値は 4194304 バイトです。
  - キュー・マネージャーを作成する際に、ログ・ファイルのサイズと数を増やします。

– チャネルを WebSphere MQ キュー・マネージャー用に使用する場合は、チャネルの最大メッセージ長を増やしてください。

ログ・ファイルの構成の詳細については、WebSphere MQ システム管理の資料を参照してください。

ALE Module のディレクトリーとファイル

表 17 に、ALE Module が使用するディレクトリーとファイルを示します。

表 17. ALE Module のディレクトリーとファイル

ファイル名	イベント	要求	説明
sap_idoccontrol.xsd	可	可	制御レコード・ビジネス・オブジェクト定義ファイル。¥repository¥SAP ディレクトリーに格納されています。
EventState.log ファイル	可	なし	AleEventDir 構成プロパティーで指定されたディレクトリーに格納されており、コネクターは、JMS-MQ イベント・メッセージ内の正常に処理された IDoc についての情報をこのファイルに記録します。注: コネクターは、最初にイベントを処理する時点でログ・ファイルを自動的に作成するわけではありません。コネクターを最初に稼働させる前に、このファイルを作成する必要があります。

注: 本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。Linux のインストールの場合には、円記号 (¥) をスラッシュ (/) に置き換えてください。ファイルのパス名はすべて、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

ALE Module の構成

ALE Module を使用する前に、以下の作業を行う必要があります。

- モジュールのプロパティーに ALE Module のモジュール名を追加します。モジュール名は ALE です。
- TID 管理でイベント処理を使用可能にするには、適切なコネクター固有のプロパティーを構成する必要があります。
- ALE Module がイベント処理のために IDoc を検索した後で、コネクターによって標準の SAP 状況コードが更新されるようにするためには、特定のプロパティーを構成し、SAP の Logical System にある Partner Profile のインバウンド・パラメーターを構成して ALEAUD メッセージ・タイプを受信します。詳細な情報と、関係のあるプロパティーの完全なリストについては、127 ページの『IDoc の状況を更新するための SAP の構成』を参照してください。
- 残りの必要な標準構成プロパティーおよびコネクター固有の構成プロパティーを設定します。

  コネクター構成プロパティーを設定するには、Connector Designer を使用します。コネクター構成プロパティーの設定の詳細については、13 ページの『第 2

章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 315 ページの『付録 B. 標準構成プロパティー』 を参照してください。

重要: アプリケーション・イベントの処理時に適切なエラー管理を行うために、このモジュールではコネクター・ポーリングが必要とされます。したがって、コネクターの PollFrequency プロパティーの値を key または no に設定しないでください。コネクターのログに必須 RFC 機能のインストールが表示されていることを検証するまで、SAP アプリケーションによってコネクターへのイベントが起動されないようにしてください。

SAP の構成の検査

ALE Module を実行する前に、SAP システムがビジネス・オブジェクトを処理するために正しく構成されているかどうかを確認します。

- 論理システムが定義され、SAP システムおよび外部システムに割り当てられているかどうかを検査します (トランザクション・コード SALE)。
- 分散モデルが保守されていること、および必要なメッセージ・タイプがモデルに追加されているかどうかを検査します (トランザクション・コード BD64)。
- 論理システムまたは分散モデルにパートナー・プロファイルが存在するかどうかを検査します (トランザクション・コード WE20)。

MQ 構成の検査

メッセージ・キューが正しく構成されていることを検査します。

イベント処理の場合:

- SAP アプリケーション (トランザクション・コード SM59) が、RfcProgramId 構成プロパティーで指定されているプログラム ID に一致しているかどうかを確認します。TCP/IP ポートのセットアップの詳細については、 165 ページの『RFC Server Module の SAP gateway への登録』を参照してください。
- WIP (SAP_Wip_Queue)、イベント (SAP_Event_Queue)、エラー (SAP_Error_Queue)、アンサブスクライブ (SAP_Unsubscribed_Queue)、およびアーカイブ・キュー (SAP_Archive_Queue) が定義されていて WebSphere MQ で稼動していることを確認します。

要求処理の場合は、要求キュー (SAPtid_Queue) が定義されていて WebSphere MQ で稼働していることを確認します。

IDoc の状況を更新するための SAP の構成

ALE Module がイベント処理のために IDoc を検索した後で、コネクターによって標準の SAP 状況コードが更新されるようにするには、以下の作業を行います。

- AleUpdateStatus 構成プロパティーを true に設定し、AleSuccessCode および AleFailureCode 構成プロパティーの値を設定します。
- SAP の Logical System にある Partner Profile のインバウンド・パラメーターを、ALEAUD メッセージ・タイプを受け取るように構成します。

詳細については、 134 ページの『SAP での IDoc の状況の更新』を参照してください。

SAP の構成

Logical System にある partner profile のインバウンド・パラメーターを、ALEAUD メッセージ・タイプを受け取るように構成します。以下のプロパティーを、指定した値に設定します。

表 18. IDoc の状況を受け取るための SAP の構成

SAP プロパティー	値
Basic Type	ALEAUD01
Logical Message Type	ALEAUD
Function module	IDOC_INPUT_ALEAUD
Process Code	AUD1

コネクター固有構成プロパティーの設定

IDoc 状況を戻すために必要な以下のコネクター固有の構成プロパティーを設定します。

- 344 ページの『AleUpdateStatus』
- 345 ページの『AleSuccessCode』
- 345 ページの『AleFailureCode』

イベントおよび要求の処理に必要な以下のコネクター固有の構成プロパティーを設定します。

- 349 ページの『SAPtid_MQChannel』
- 350 ページの『SAPtid_MQPort』
- 350 ページの『SAPtid_QueueManager』
- 350 ページの『SAPtid_QueueManagerHost』
- 350 ページの『SAPtid_QueueManagerLogin』
- 350 ページの『SAPtid_QueueManagerPassword』

以下のオプションのコネクター固有構成プロパティーも設定できます。

- 345 ページの『AleSelectiveUpdate』
- 345 ページの『AleStatusMsgCode』
- 346 ページの『AleSuccessText』
- 346 ページの『AleFailureText』

キュー・マネージャーをリモートするための接続

リモート・キュー・マネージャーに必要な以下のコネクター固有の構成プロパティーを設定します。

- 349 ページの『SAPtid_MQChannel』
- 350 ページの『SAPtid_MQPort』
- 350 ページの『SAPtid_QueueManager』
- 350 ページの『SAPtid_QueueManagerHost』

- 350 ページの『SAPtid_QueueManagerLogin』
- 350 ページの『SAPtid_QueueManagerPassword』

ALE Module の実行

アプリケーション・イベントの処理時に、ALE Module は SAP アプリケーションがコネクターにプッシュしたイベントを受信します。要求の処理時には、ALE Module はビジネス・オブジェクト要求を統合ブローカーから受け取って SAP アプリケーションに送信します。

初期化と終了

アプリケーション・イベントまたはビジネス・オブジェクト要求を処理する場合、コネクターの初期化プロセスでは以下の処理が実行されます。

1. SAP Gateway に、`RfcProgramID` コネクター構成プロパティーで指定されたプログラム ID を登録します。Program ID を TCP/IP ポートとして設定する方法の詳細については、165 ページの『RFC Server Module の SAP gateway への登録』を参照してください。
2. コネクター用に構成されたキューに対する WebSphere MQ セッションを開きます。
3. イベントと要求の処理に必要な WebSphere MQ キューが作成済みであることを確認します。作成済みでない場合、その処理はコネクターを終了します。

コネクターはマルチスレッド化をサポートしているため、ALE Module は、統合ブローカーからの要求を処理する場合に当該ハンドルの SAP の Java コネクター (SAP JCo) 接続プールを使用します。

重要: ALE モジュールを使用してアプリケーション・イベントを処理する場合、このモジュールを正しく初期化 (サーバーに RFC 機能をインストールするため) したり、このモジュールがエラーを正しく管理したりするためには、コネクターによるポーリングが必要です。したがって、PollFrequency プロパティーの値を `key` または `no` に設定しないでください。コネクターのログに必須 RFC 機能のインストールが表示されていることを検証するまで、SAP アプリケーションによってコネクターへのイベントが起動されないようにしてください。

ビジネス・オブジェクトの処理

ALE Module が SAP 用の WebSphere ビジネス・オブジェクトを処理する場合は、イベント処理または要求処理のいずれかによって開始されます。

ビジネス・オブジェクト・データが SAP の Java コネクター (SAP JCo) API から戻されると、ALE Module は、DATS および TIMS フィールドの値を、DATS データ・エレメントの場合は `YYYY-MM-DD` (ハイフンを含む)、TIMS データ・エレメントの場合は `HH:mm:ss` (コロンを含む) という形式で受け取ります。大文字の `HH` は、12 時間制ではなく 24 時間制の時刻を表します。イベントを処理する場合、ALE Module はこれらの形式を、対応するビジネス・オブジェクト属性の 8 文字および 6 文字の最大サイズに合わせて変更します。コネクターは、日付データのハイフンや時刻データのコロンを除去して、値の長さを短縮します。

イベント処理

ALE Module のすべてのイベント処理は、SAP アプリケーションの 2 つの RFC 対応機能が開始します。ALE のイベント処理用のビジネス・オブジェクト・ハンドラーでは、機能 idoc_inbound_asynchronous および inbound_idoc_process をサポートしています。

イベントを処理する場合、このビジネス・オブジェクト・ハンドラーはビジネス・オブジェクトを WebSphere MQ キューに永続的に保管します。コネクターは RFC 呼び出しに関連付けられた Transaction ID (TID) を保守することで、個々のデータが 1 回に限ってデリバリーされることを保証します。

重要: 1 回の RFC 呼び出しで、1 つ以上の IDoc に対するデータを送信できます。この場合、WebSphere MQ キューには複数の IDoc を表す JMS-MQ メッセージが格納され、それぞれの IDoc が 1 つのビジネス・オブジェクトを表すことになります。各 RFC 呼び出しは、1 つの TID に関連付けられています。

WebSphere MQ キューにおけるイベント処理: 図 51 では、ALE Module が WebSphere MQ キューを処理する方法を説明しています。

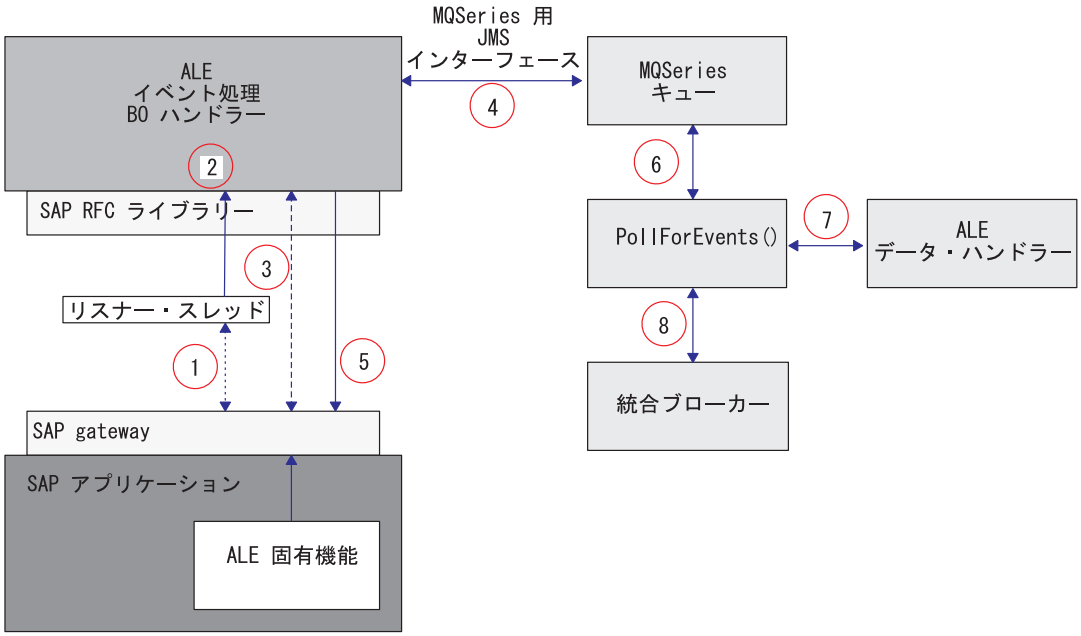


図 51. ビジネス・オブジェクト・イベント処理

ALE Module のビジネス・オブジェクト・イベント処理は、以下の方法で実行されます。

1. RFC 機能はイベント・データを SAP Gateway にプッシュし、そこでリスナー・スレッドがイベントを選出します。スレッドはイベントに関連した TID を検査し、以下のようにその TID に対応する JMS-MQ メッセージが存在するかどうかを判別します。
   - TID が事前に送信されていない場合、コネクターは 2 の手順に進みます。

- TID が事前に送信済みの場合、コネクターの振る舞いは直前のトランザクションの状態によって異なります。TidStatus が CREATED の場合、コネクターは IDoc データをメッセージから除去します。状況が ROLLBACK の場合、コネクターは状況を CREATED に変更し、さらに、IDoc データがメッセージ内に存在する場合にはコネクターは IDoc データをメッセージから除去します。状況が EXECUTED の場合、コネクターは制御を SAP に戻します。

2. リスナー・スレッドは、SAP Gateway から RFC インターフェース・データを検索する ALE イベント処理ビジネス・オブジェクト・ハンドラーのインスタンスを生成します。
3. ビジネス・オブジェクト・ハンドラーは、各トランザクションを JMS-MQ メッセージの形式に整え、SAPALE_Wip_Queue 構成プロパティーによって指定されたキューの中に永続的に保管します。

   各 JMS-MQ メッセージは単一の RFC 呼び出しを表します。各 RFC 呼び出しは、単一の TID に関連した 1 つ以上のビジネス・オブジェクトを表すことができます。コネクターは TID をそのメッセージの CorrelationID プロパティーに格納し、TidStatus を CREATED に設定し、IDocProcessStatus を unknown に設定します。コネクターはメッセージ・ボディを使用して IDoc データを格納します。

   ラージ・オブジェクトの場合、コネクターは処理をより効率化できるように、オブジェクトを複数のメッセージに分割します。このサポートを使用可能にする方法については、119 ページの『イベント処理コンポーネント』および 132 ページの表 19 を参照してください。
4. 各トランザクションの完了後、コネクターは TidStatus の値を変更し、トランザクションが完了したことを示す確認を SAP に返送します。SAP は確認を受信した後で、SAP アプリケーションから TID とその関連データを除去します。

   AleUpdateStatus 構成プロパティーの値が true の場合、コネクターは SAP 内の IDoc の状況を更新します。コネクターは、複数の IDoc を含んだパケットを検索すると、パケット内のすべての IDoc の状況を更新します。詳細については、134 ページの『SAP での IDoc の状況の更新』を参照してください。
5. コネクターは JMS-MQ メッセージを WIP キューから SAPALE_Event_Queue 構成プロパティーで指定されたキューに移動させます。
6. ALE Module のポーリング・スレッドが、イベント・キューからイベント・メッセージを選出します。
7. コネクターは、ALE データ・ハンドラーをインスタンス化します。このハンドラーによって、メッセージ・ボディーにあるデータがビジネス・オブジェクトに変換されて、統合ブローカーにポストされます。
8. コネクターは、それぞれのビジネス・オブジェクトを統合ブローカーにポストしようとします。コネクターは、まず最初に、そのビジネス・オブジェクトに対するサブスクリプションがあるかチェックします。メッセージ・ボディにあるすべてのビジネス・オブジェクトを処理したあと、そのメッセージの IDocProcessingStatus と BOProcessingStatus が更新され、メッセージは、SAPALE_Archive_Queue のプロパティーで指定されたキューに移動されます。IDocProcessingStatus の詳細については、『アーカイブ・メッセージの作成』を、BOProcessingStatus については、『イベント処理およびアーカイブ処理用の JMS-MQ メッセージの構造』を参照してください。

ALE Module は、イベント・キューからメッセージを読み取る際の処理の順序を FIFO (先入れ先出し法) を使用することで維持します。

重要: アプリケーション・イベントの処理時に適切なエラー管理を行うために、このモジュールではコネクター・ポーリングが必要とされます。したがって、コネクターの PollFrequency プロパティーの値を key または no に設定しないでください。コネクターのログに必須 RFC 機能のインストールが表示されていることを検証するまで、SAP アプリケーションによってコネクターへのイベントが起動されないようにしてください。

イベントの再サブミット: SAPALE_Unsubscribed_Queue および SAPALE_Error_Queue の状態にあったイベントを、コマンド行ユーティリティーを使用して再サブミットすることができます。このコマンド行ユーティリティーは、Windows では BIA_AleEventUtil.bat、Linux では、BIA_AleEventUtil.sh で、ディレクトリー *ProductDir*/connectors/SAP/utilities/ALEEventUtil/ (*ProductDir* はコネクターがインストールされているディレクトリー) 内にあります。詳細については、135 ページの『イベント処理用の ALE Module キュー管理ユーティリティー』を参照してください。

イベント処理およびアーカイブ処理用の JMS-MQ メッセージの構造: 表 19 では、コネクターがイベント・キューおよびアーカイブ・キューに送信するメッセージの構造を説明しています。

表 19. イベント処理およびアーカイブ処理用の JMS-MQ メッセージの構造

JMS メッセージ・ヘッダー・プロパティー	説明
CorrelationId	コネクターは SAP が送信するトランザクション ID (TID) からこのプロパティーの値を設定します。 大きな IDoc をそれより小さいメッセージのパーツに分解するとき、このプロパティーはそのパーツが属する大きなメッセージを識別します。コネクターは、この値をこの集合の最初のパーツの JMSMessageID に設定します。最初のパーツの CorrelationID は必ず、大きな IDoc に関連付けられている最初の JMS-MQ メッセージの CorrelationID であることに注意してください。IDoc をより小さいメッセージのパーツに分割する方法の詳細については、119 ページの『イベント処理コンポーネント』を参照してください。
JMSMessageID	メッセージの固有 ID。大きな IDoc をそれより小さいメッセージのパーツに分割するとき、コネクターは最初のパーツ以外のすべてのパーツのこのプロパティーの値を、最初のパーツの JMSMessageID に設定します。IDoc をより小さいメッセージのパーツに分割する方法の詳細については、119 ページの『イベント処理コンポーネント』を参照してください。

表 19. イベント処理およびアーカイブ処理用の JMS-MQ メッセージの構造 (続き)

JMS メッセージ・ヘッダー・プロパティー	説明
MutliPartMessage	大きな IDoc をそれより小さいメッセージのパーツに分割するとき、コネクターはこのプロパティーを使用して、そのメッセージがそれより大きなメッセージの対応するパーツであることを識別します。例えば、元の大きな IDoc メッセージを 8 つの JMS-MQ メッセージに分割するとすれば、メッセージごとの各パーツの MultiPartMessage プロパティーの値は 1/8、2/8 などとなります。IDoc をより小さいメッセージのパーツに分割する方法の詳細については、 119 ページの『イベント処理コンポーネント』を参照してください。
TidStatus	TID の状況を保持します。
IDocProcessStatus	イベント処理中に IDoc オブジェクトの状況を保持します。
BOProcessingStatus	メッセージ内のすべての IDoc の状況を保持します。フォーマットは <CID> :: <IDoc sequence number><Status symbol> です。状況記号の種類は、S (成功)、F (失敗) および U (アンサブスクライブされた) です。例えば「<CID> :: 0S, 1F, 2U」は、最初の IDoc は成功、2 番目は失敗、3 番目は CorrelationId = <CID> に対してアンサブスクライブされたことを表します。

表 20 では、イベントがアーカイブ・キューに移された後で、IDocProcessStatus プロパティーが取り得る値について説明しています。

表 20. IDocProcessStatus メッセージ・プロパティーに対するアーカイブ・キューの値

IDocProcessStatus プロパティー値	イベント状況	説明
success	成功	メッセージにあるすべてのビジネス・オブジェクトが、エラーなしでポストされました。
partial	一部成功	メッセージにあるすべてのビジネス・オブジェクトではないが、1 つ以上のものがエラーなしでポストされました。メッセージにあるすべてのビジネス・オブジェクトではないが、1 つ以上のものがエラーなしでポストされたか、アンサブスクライブされています。
unsubscribed	アンサブスクライブ	メッセージにあるすべてのビジネス・オブジェクトがアンサブスクライブされています。
fail	失敗	メッセージにあるすべてのビジネス・オブジェクトが、エラーを示してポストされました。

アーカイブ・メッセージの作成: メッセージがイベント・キューからアーカイブ・キューに移されると、IDocProcessingStatus および BOProcessingStatus が更新されます。メッセージ・ボディはそのまま変更されません。

例えば、コネクターが 4 つの IDoc を含んだイベント・メッセージを処理し、各 IDoc がビジネス・オブジェクトに変換されるか、または変換を試行された場合、結果は表 21 のようになります。

表 21. アーカイブ・メッセージの作成

IDoc またはビジネス・オブジェクトの状況	結果として得られるアーカイブ・メッセージ
最初の IDoc は正常に変換され、ビジネス・オブジェクトは統合ブローカーにポストされました。 2 番目の IDoc のビジネス・オブジェクトへの変換に失敗しました。	IDocProcessStatus は success に更新され、BOProcessingStatus は `<CID> :: 0S` になります。 IDocProcessStatus は partial に更新され、BOProcessingStatus は `<CID> :: 0S, 1F` なります。
3 番目の IDoc は正常に変換され、ビジネス・オブジェクトは統合ブローカーにポストされました。 4 番目の IDoc は正常に変換されましたが、作成されたビジネス・オブジェクトは統合ブローカーにサブスクライブされませんでした。	IDocProcessStatus は partial に設定され、BOProcessingStatus は `<CID> :: 0S, 1F, 2S` になります。 • IDocProcessStatus は partial に設定され、BOProcessingStatus は `<CID> :: 0S, 1F, 2S, 3U` になります。 • 最後の IDoc を処理した後で、メッセージをイベント・キューからアーカイブ・キューに移動し、IDocProcessStatus は partial に、BOProcessingStatus は `<CID> :: 0S, 1F, 2S, 3U` に設定されます。

SAP での IDoc の状況の更新: ALE Module がイベント処理のために IDoc を検索した後で、コネクターによって標準の SAP 状況コードが更新されるようにするためには、以下の作業を行う必要があります。

- `AleUpdateStatus` 構成プロパティーを `true` に設定し、`AleSuccessCode` および `AleFailureCode` 構成プロパティーの値を設定します。
- SAP の Logical System にある Partner Profile のインバウンド・パラメーターを、`ALEAUD` メッセージ・タイプを受け取るように構成します。

`AleUpdateStatus` の値が `true` に評価された場合、コネクターは `ALEAUD` IDoc を状況コード情報および記述テキストとともに SAP に送信します。`ALEAUD` IDoc は `IDOC_INPUT_ALEAUD` 機能モジュールを呼び出します。コネクターは、この機能モジュールに対する以下の状況コードの送信をサポートします。

- IDoc は Business Integration システムに完全にポストされています。

  `AleSuccessCode` コネクター固有の構成プロパティーは、値 `52` または `53` に設定できます。SAP はこの値を `41` に変換します。

- IDoc は Business Integration システムで処理できません。

  `AleFailureCode` コネクター固有の構成プロパティーは、値 `68` に設定できます。SAP はこの値を `40` に変換します。

上記のどちらの場合でも、Business Integration システムはそれ以降、後続の処理を表す状況コードを送信しません。

IDoc の状況を戻すために必要なコネクター固有構成プロパティーの設定の詳細については、以下の各セクションを参照してください。

- 344 ページの『AleUpdateStatus』
- 345 ページの『AleSuccessCode』

- 345 ページの『AleFailureCode』

IDoc の状況を戻すための、オプションのコネクター固有構成プロパティーの設定の詳細については、以下の各セクションを参照してください。

- 345 ページの『AleSelectiveUpdate』
- 345 ページの『AleStatusMsgCode』
- 346 ページの『AleSuccessText』
- 346 ページの『AleFailureText』

イベント処理用の ALE Module キュー管理ユーティリティー

このコマンド行ユーティリティーは、Adapter for mySAP.com の (v. 5.3.2) ALE Module が使用する MQ キューの保守に使用します。このユーティリティーはイベント・メッセージを再サブミットし、表示用にイベント・メッセージをファイル・システムにダンプし、メッセージをファイル・システムにアーカイブします。

IDoc は、トランザクションという作業単位で処理されます。複数の IDoc を含む SAP トランザクションをトランザクション・パケットと呼びます。アダプターは、1 つ以上の IDoc を保持する MQ メッセージを使用してトランザクションおよびトランザクション・パケットを処理します。アダプターは、IDoc を対応するビジネス・オブジェクトに変換します。ALE モジュールは、SAP からアダプター、アダプターからブローカーの順に 2 段階で IDoc を処理します。例外は、各ステップで個別に処理されます。

MQ メッセージについて詳しくは、WebSphere Business Integration Library: http://www.ibm.com/software/integration/wmq/library/ を参照してください。

SAP からアダプターへの IDoc の処理: アダプターがアンサブスクライブされたビジネス・オブジェクトまたはサポートされないビジネス・オブジェクトを検出した場合、または IDoc の送信中に例外が発生した場合は、アダプターは SAP トランザクションの処理に失敗します。失敗したトランザクションは、SAP トランザクション SM58 から表示および再サブミットできます。トランザクションを再サブミットする前に、以下の例外に対処してください。

- `Unsupported`: ビジネス・オブジェクト用のエージェント・サポートを追加します。
- `Unsubscribed`: ビジネス・オブジェクトのコラボレーションを再始動します。
- その他の例外: アダプター・ログを参照して例外を判別し、必要な訂正を行います。

上記のステップが正常に実行されると、SAP でのトランザクションは完了します。

重要: イベントのデリバリーが重複しないようにするために、訂正した IDoc トランザクションまたはトランザクション・パケット内の個々の IDoc は再サブミットしないでください。

アダプターからブローカーへの IDoc の処理: MQ メッセージに単一のビジネス・オブジェクトが含まれ、そのビジネス・オブジェクトがアンサブスクライブされている場合は、MQ メッセージはアンサブスクライブされたキューに移動されます。トランザクション・パケット内の各アンサブスクライブされたビジネス・オブジェクトは、アンサブスクライブされたキューに専用の MQ メッセージとして存続しま

す。元の MQ メッセージは変更されず、個々の IDoc の処理状況が格納されます。MQ メッセージのトランザクション・パケットが完全に処理されると、アーカイブ・キューに移動されます。

トランザクションを再サブミットする前に、以下の例外に対処してください。

- Unsubscribed: ビジネス・オブジェクトのコラボレーションを再始動します。
- その他の例外: アダプター・ログを参照して例外を判別し、必要な訂正を行います。

訂正が完了したら、コマンド・ユーティリティー AleEventUtil を使用して MQ メッセージをイベント・キューに戻し、イベントを再サブミットします。

IDoc に誤った形式のデータが含まれる場合、または「nodata」が含まれる場合は、IDoc は独自のメッセージとしてエラー・キューに移動されます。

ALE Module Queue ユーティリティーのインストールおよび構成: ALE Module Queue ユーティリティーは SAP アダプターにパッケージされています。インストールすると、以下のディレクトリー構造が作成されます。

```
¥Connectors¥SAP¥BIA_AleEventUtil.jar

¥Connectors¥SAP¥BIA_AleEventUtil.bat

¥Connectors¥SAP¥BIA_AleEventUtil_readme.txt
```

以下のパラメーターを収集するように始動スクリプト・ファイル BIA_AleEventUtil.bat を変更します。ローカル・キュー・マネージャーにアクセスするために構成する必要があるのは MQQueueManager のみです。

変数	説明	コメント
MQQueueManager	Queue Manager の名前	必須パラメーター
MQChannel	サーバー接続チャネル名	リモート・キュー・マネージャーにアクセスするために必要です。
MQPort	チャネルが listen するポート	リモート・キュー・マネージャーにアクセスするために必要です。
MQHost	キュー・マネージャーが稼動するホスト名または IP アドレス	リモート・キュー・マネージャーにアクセスするために必要です。
MQUser	MQHost の有効なユーザー名	リモート・キュー・マネージャーにアクセスするために必要です。
MQPassword	ユーザー・パスワード	リモート・キュー・マネージャーにアクセスするために必要です。値は暗号化されません。

MQ 管理ユーティリティーの実行: ユーティリティーをインストールして構成したら、ALE Module キュー管理ユーティリティーのインストール先ディレクトリーにナビゲートします。ユーティリティーに有効なコマンドは以下のとおりです。

```
-c <choice> (有効なオプションは [move、archive、dump、replicate])

-i <inputq>

-o <outputq>

-f <outputfile>

-d <date>

-u <unique message ID>

-n <replication count>
```

注: 同名のファイルが既に存在する場合、archive コマンドは例外を発生させますが、dump コマンドはファイルを上書きします。

メッセージの内容をファイルにダンプするには、コマンド・プロンプトで、ユーティリティーのインストール先ディレクトリーに切り替えて以下のコマンドを実行します。

```
BIA_AleEventUtil -cdump -i<QueueName> -f<OutputFileName>
```

あるキューから別のキューにメッセージを移動するには、以下のコマンドを実行します。以下のコマンドは、キューにあるメッセージをすべて移動します。

```
BIA_AleEventUtil -cmove -i<FromQueue> -o<ToQueue>
```

単一のメッセージを移動するには、以下のように、該当するメッセージのメッセージ ID に対応する MessageIdByte の追加のパラメーターを使用します。

```
BIA_AleEventUtil -cmove -i<FromQueue> -o<ToQueue> -u<MessageIdByte>
```

指定の日付以前のメッセージをすべて移動するには、以下のように Date パラメーターを追加します。

```
BIA_AleEventUtil -cmove -i<FromQueue> -o<ToQueue> -d<date(YYYYMMDD)>
```

キューからファイルにメッセージをアーカイブし、指定の日付以前のメッセージをすべて除去するには、以下のコマンドを使用します。

```
BIA_AleEventUtil -carchive -i<QueueName> -f<ArchiveFileName>
-d<date(YYYYMMDD)>
```

要求処理

Vision コネクター・フレームワークは、トップレベルのビジネス・オブジェクトの動詞 AppSpecificInfo プロパティーの値を使用して、ALE 要求処理ビジネス・オ

ブジェクト・ハンドラーをインスタンス化します。要求処理ビジネス・オブジェクト・ハンドラーの doVerbFor() メソッドでは、すべてのビジネス・オブジェクト要求を開始します。

ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクト・データを、IDoc 形式と、そのメタデータ・コンポーネントである制御レコードを表す 2 つの表に変換します。データを IDoc 形式に変換すると、ビジネス・オブジェクト・ハンドラーは idoc_inbound_asynchronous または inbound_idoc_process のいずれか適切な SAP 機能モジュールに対して RFC 呼び出しを行います。ALE は非同期なので、コネクターは応答が戻されるのを待機しません。

重要: デフォルトでは、SAPODA で生成された親ラッパー・ビジネス・オブジェクトには、TransactionId 属性が含まれています。この属性の値に応じて、コネクターはサービス呼び出し要求を処理する際に TID の管理を行います。要求処理の際に TID 管理を行わないようにしたい場合は、この属性の値を設定しないでください。詳細については、145 ページの『親ラッパー・ビジネス・オブジェクト』を参照してください。

注: TransactionId 属性の値は、固有の ID でなければなりません。値は SAP アプリケーション内の TID と等価ではありません。これらの値は、SAPtid_Queue 構成プロパティーで指定されたキューの中の JMS_MQ メッセージにあるテーブルに保管されます。

TransactionId 属性が値を持っていない場合は、ALE Module は要求を直接 SAP に送信します。TransactionId 属性が値を持っている場合は、ALE Module は以下のいずれかを行います。

1. コネクターは、SAPtid_Queue 構成プロパティーで指定されたキューにある JMS-MQ メッセージがこの値を持っているかどうか、チェックします。
   - ビジネス・オブジェクトの TransactionId 属性、ObjectID の値が JMS_MQ メッセージのテーブルにない場合は、新しいエントリーがテーブルの中に作成されます。ObjectID は、テーブル・エントリーへのキーになります。このとき、コネクターは SAP から新規の TID を検索し、その TID がこの ObjectID に割り当てられます。また、コネクターは、この ObjectID 用の TidStatus も CREATED に設定します。
   - ObjectID がテーブルに存在する場合のコネクターの振る舞いは、この ObjectID 用の TidStatus によって決まります。TidStatus が CREATED の場合、コネクターは 2 に進みます。TidStatus が ROLLBACK の場合、コネクターは、値を CREATED に変更してから、2 に進みます。TidStatus が EXECUTED の場合は、キーが除去され、アーカイブされます。
2. コネクターはビジネス・オブジェクトを RFC 表に変換し、SAP への RFC 呼び出しを行います。
   - 呼び出しが正常にポストされた場合、コネクターはキーの TidStatus を EXECUTED に更新します。
   - 呼び出しが SAP へのポストに失敗した場合、あるいは例外が発生した場合、コネクターはキーの TidStatus を ROLLBACK に更新します。
3. SAP が RFC 呼び出しの受け取りを確認したあと、コネクターは テーブルからキーを除去し、キーをアーカイブしてから、統合ブローカーに成功状況を戻します。

アーカイブ: サービス呼び出し要求の処理が正常に行なわれると、SAPtid_Queue にある JMS-MQ メッセージのテーブルの中で該当するエントリーが除去され、ディレクトリーにアーカイブされます。ファイルが、WINNT システムの場合は ¥ale¥request サブディレクトリーに、Linux システムの場合は /ale/request に作成されます。ale サブディレクトリーは、アダプターが開始されるディレクトリーに置かれます。テーブルから除去されたエントリーは、新しいファイルを作成するのに使用されます。ファイル名は、<ObjectID>_<TID><timestamp>.executed という形式になります。ここで、ObjectID は TransactionId 属性からの値、TID は SAP からのトランザクション ID、timestamp はファイルが作成されたときのタイム・スタンプです。

アダプター自体が、コネクターの構成プロパティー ArchiveDays を用いて、これらのアーカイブ・ファイルの削除を管理します。コネクターの構成プロパティー、ArchiveDays の値によって、これらのアーカイブ・ファイルが ale¥request サブディレクトリーに存続する日数が決まります。ArchiveDays で指定された日数より古いファイルはいずれも、削除されます。このプロパティーが構成されていない場合の ArchiveDays のデフォルト値は 7 日です。これらのアーカイブ・ファイルは、ファイルを自分で削除することによって手作業で管理することもできます。

失敗した要求の再サブミット: 統合ブローカーによって示されている、失敗に終ったすべての要求について、その要求に関するアーカイブ・ファイルが作成されたかどうかチェックしてください。要求にあるオブジェクト ID に対するアーカイブ・ファイルが存在する場合は、統合ブローカーから要求を再サブミットしないでください。その ObjectID に対するアーカイブ・ファイルがない場合は、要求を再サブミットしてください。ArchiveDays コネクター構成のプロパティーが、再サブミットされた要求の検査の許容範囲の値に設定されていることを確認してください。

要求処理のための JMS WebSphere MQ メッセージ・テーブルの欄: 表 22 では、コネクターが SAPtid_Queue から取得する JMS-WebSphere MQ メッセージの欄について説明します。

表 22. 要求処理用の JMS-MQ メッセージの欄

列名	説明
ObjectID	要求されたビジネス・オブジェクトの TransactionID 属性にある値。この値は、テーブルのキーとして使用されます。
TID	SAP から取得されたトランザクション ID
TidStatus	トランザクションの状況

要求処理のための複数のメッセージ・タイプのサポート: 要求処理のために、1 つの SAP コネクターのインスタンスが同じ IDoc タイプを参照する複数のメッセージ・タイプを処理することができます。イベントの処理では、メッセージのタイプごとに異なるビジネス・オブジェクト定義が必要になることがありますが、要求の処理では複数のメッセージ・タイプに対して 1 つのビジネス・オブジェクト定義を使用することができます。

制御レコード・オブジェクトには、該当するメッセージ・タイプ (MESTYP) を設定します。さらに、動詞はメッセージ・タイプに対して影響を与えないため、異なるメッセージ・タイプを持つ別々の動詞に対して同じビジネス・オブジェクト・タイプを使用することができます。

イベント処理のための複数のメッセージ・タイプのサポート: イベント処理では、以下の機構を使用することができます。

- 1 つの IDoc タイプを表す同じビジネス・オブジェクトを使用して、動詞 ASI メタデータを MsgType / MsgCode / MsgFunction の異なる組み合わせで構成します。各動詞に指定された値の組み合わせが異なっている必要があります。例えば、異なる動詞に対して、ASI を以下のように構成します。

  ```
  Verb=Create   VerbASI : MsgType=ORDERS; MsgCode=MC01;MsgFunction=MF01

  Verb=Update   VerbASI : MsgType=ORDERS;MsgCode=MC02;MsgFunction=MF02

  Verb=Delete    VerbASI : MsgType=ORDERS;MsgCode=MC03;MsgFunction=MF03
  ```

  動詞が異なれば、同じ組み合わせの MsgType/MsgCode/MsgFunction 値を使用できないことに注意してください。

  あるいは、動詞ごとに異なるメッセージ・タイプを使用することができます。

  ```
  Verb=Create  VerbASI : MsgType=ORDERS;MsgCode=;MsgFunction=

  Verb=Update   VerbASI :  MsgType=ORDCHG;MsgCode=;MsgFunction=

  Verb=Delete    VerbASI :  MsgType=;MsgCode=;MsgFunction=
  ```

- 異なるメッセージ・タイプに対して、同じビジネス・オブジェクトと動詞の組み合わせを使用する必要がある場合は、同じ IDoc タイプのビジネス・オブジェクトを別の名前でコピーしてください。例えば、ビジネス・オブジェクト sap_orders_05_ORDERS と sap_orders_05_QUOTES の両方が同じ IDoc タイプの定義を参照し、同じビジネス・オブジェクトのコピーであるとします。オブジェクトごとの ASI は以下のように構成されます。

  sap_orders_05_ORDERS の動詞 ASI

  ```
  Verb=Create  VerbASI : MsgType=ORDERS;MsgCode=;MsgFunction=
  ```

  sap_orders_05_QUOTES の動詞 ASI

  ```
  Verb=Create  VerbASI :  MsgType=QUOTES;MsgCode=;MsgFunction=
  ```

第 11 章 ALE Module のビジネス・オブジェクトの開発

この章では、Adapter for mySAP.com の ALE Module のために必要なビジネス・オブジェクトについて説明します。また、ビジネス・オブジェクト生成ユーティリティーの SAPODA が定義を生成する方法についても説明します。この章の読者は、コネクターがビジネス・オブジェクトを処理する方法について十分な知識を持っていることが想定されています。ALE Module の詳細については、117 ページの『第 9 章 ALE Module の概要』 を参照してください。

SAPODA を使用して、このモジュール用のビジネス・オブジェクト定義を生成します。SAPODA は SAP アプリケーションのネイティブ IDoc (Intermediate Document) 定義を、ALE Module 用のビジネス・オブジェクト定義のテンプレートとして使用します。いったん作成した定義を変更するには、Business Object Designer Express またはテキスト・エディターを使用します。SAPODA を利用して、以下のような IDoc に基づく ALE Module 用のビジネス・オブジェクト定義を生成することができます。

- ファイルに抽出された IDoc
- SAP システム内で定義された IDoc

IDoc は、SAP で正しく処理されるためには、特定のフォーマットに準拠している必要があります。そのため、ALE Module 用のビジネス・オブジェクト定義を開発する場合、その定義が SAP に定義されている IDoc の構造体に従っていることを確認してください。

SAPODA の詳細については、35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

この章の内容は以下のとおりです。

IDoc 定義ファイルの作成

SAPODA を使用して IDoc 定義ファイルからビジネス・オブジェクト定義を生成する前に、サポートする必要のある各 IDoc に対して IDoc 定義ファイルを作成する必要があります。SAPODA では、このファイルを入力として使用します。SAP でトランザクション WE63 を使用して IDoc 定義ファイルを作成します。SAPODA を使用して SAP システムに定義された IDoc から定義を生成する場合、この IDoc 定義ファイルを作成する必要はありません。

重要: IDoc ファイルからビジネス・オブジェクト定義を生成するには、SAP システムに英語でログオンする必要があります。SAPODA は IDoc 定義内のテキス

ト・フィールドを使用して属性名を生成し、属性名は英語でなければならないため、定義は英語のファイルから生成することが重要です。

mySAP.com V.4.6 用の IDoc 定義ファイルの作成

mySAP.com V.4.6 用の IDoc 定義ファイルを作成するには、以下の手順を実行します。

1. SAP で /oWE63 と入力して、トランザクション WE63 を選択します。
2. **IDoc レコード**・タイプのチェック・ボックスを選択解除します。
3. 「Basic type」フィールドのチェック・ボックスを選択します。
4. 「**Basic type**」フィールドに、基本 IDoc タイプを入力します。
5. 「**Output From Segment Fields**」チェック・ボックスを選択します。
6. 画面の一番上にある「**Execute**」アイコンをクリックします。IDoc 定義が画面に表示されます。
7. 定義をローカル・ディレクトリーに保管します。
   a. 「**List**」>「**Download**」を選択します。
   b. 定義を保管するためのファイル・タイプをクリックします。
   c. 「Continue」チェック・マークをクリックします。
   d. 保管するファイルのパス名を入力します。
   e. 「**Transfer**」をクリックします。

注: ビジネス・オブジェクトが IDoc 拡張に基づいている場合は、Extended Basic Types グループ化を使用します。

mySAP.com V.4.7 用の IDoc 定義ファイルの作成

mySAP.com V.4.7 用の IDoc 定義ファイルを作成するには、以下の手順を実行します。

1. SAP で /oWE63 と入力して、トランザクション WE63 を選択します。
2. 「**Basic type**」ラジオ・ボタンを選択します。
3. 「**Basic type**」フィールドに、基本 IDoc タイプを入力します。
4. 「**Output From Segment Fields**」チェック・ボックスを選択します。
5. 「**Documentation**」>「**Parser**」(F9) を選択します。
6. 「**System**」>「**List**」>「**Save**」> 「**Local File**」を選択して、定義をローカル・ディレクトリーに保管します。

ビジネス・オブジェクト構造

ALE Module 用の SAP の WebSphere ビジネス・オブジェクトは、トップレベルの親ラッパー・オブジェクトと、制御レコード・オブジェクトとデータ・レコード・オブジェクトという 2 つの子オブジェクトから構成されます。このセクションでは、以下について説明します。

- 143 ページの『ビジネス・オブジェクトの構造図』
- 144 ページの『ビジネス・オブジェクトの命名規則』
- 145 ページの『親ラッパー・ビジネス・オブジェクト』

- 147 ページの『制御レコード・ビジネス・オブジェクト』
- 148 ページの『データ・レコード・ビジネス・オブジェクト』

ビジネス・オブジェクトの構造図

図 52 では、ALE Module 用の WebSphere ビジネス・オブジェクトの構造を説明しています。

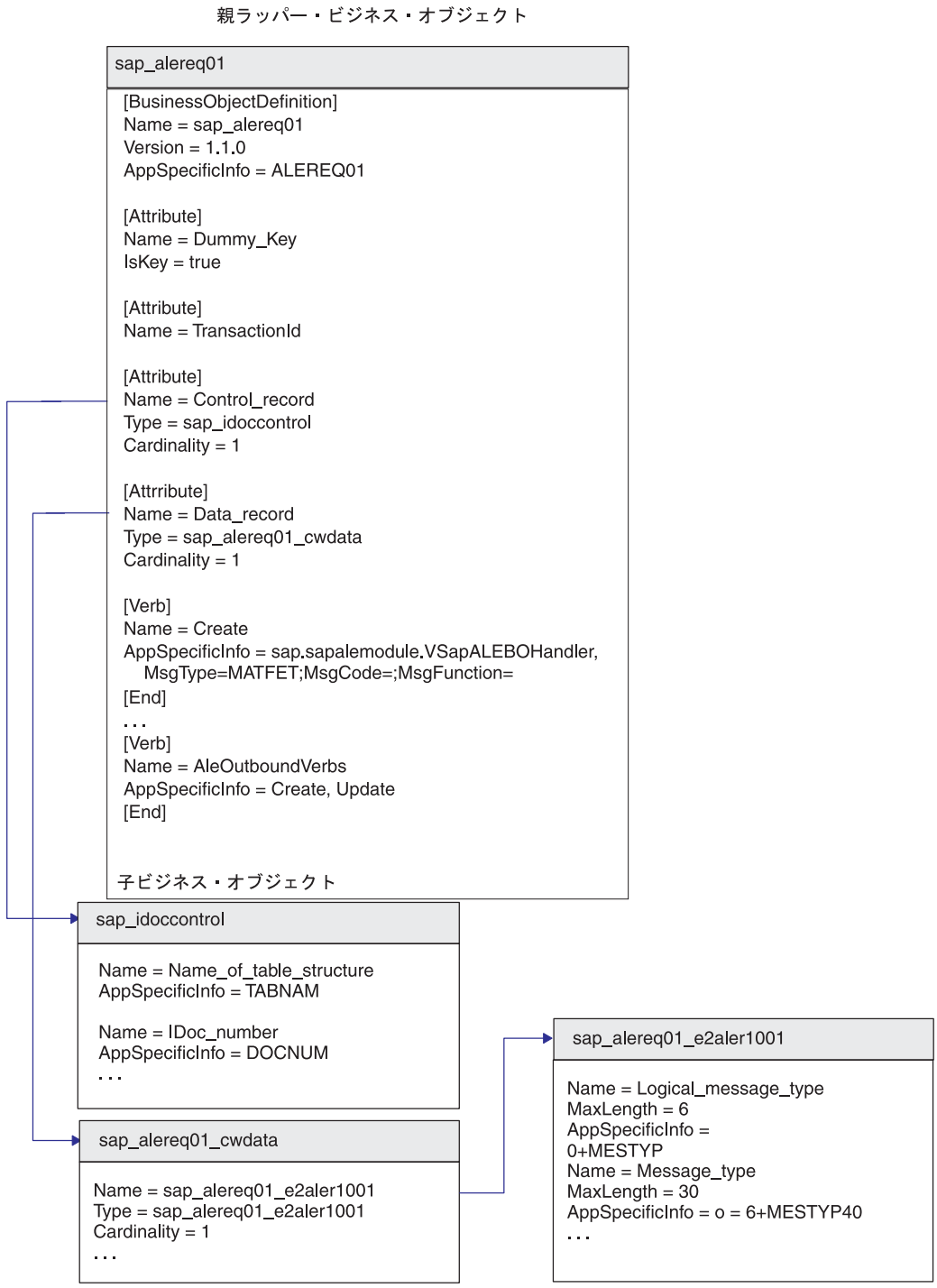


図 52. SAP の WebSphere ビジネス・オブジェクトと IDoc の関係

ビジネス・オブジェクトの命名規則

このセクションでは、以下について説明します。

- 『標準命名規則』
- 145 ページの『IDoc 拡張に対応した命名規則』

標準命名規則

ALE Module では、ビジネス・オブジェクトは、表 23 に示す命名規則に従っている必要があります。制御レコード・ビジネス・オブジェクト以外のすべてを生成する SAPODA は、これらの規則に従って、IDoc 定義からビジネス・オブジェクト名と属性名を導出します。

表 23. IBM WebSphere SAP ビジネス・オブジェクトの命名規則

IBM WebSphere のビジネス・オブジェクトまたは属性	名前	タイプ
親ラッパー・ビジネス・オブジェクト	*BOprefix_BasicIDocType* **注:** この章の図では、ビジネス・オブジェクトのプレフィックスとして SAP_ または sap_ を使用します。実際にビジネス・オブジェクト定義を作成する場合は、分かりやすい独自のプレフィックスを指定できます。	なし
制御レコード・ビジネス・オブジェクト	Control_record	sap_idoccontrol
データ・レコード・ビジネス・オブジェクト	Data_record	*BOprefix_BasicIDocType*_cwdata
データ・レコード子ビジネス・オブジェクト	*BOprefix_BasicIDocType_IDocSegmentName*	*BOprefix_BasicIDocType_IDocSegmentName*
データ・レコード属性	*IDocFieldName* または IDoc Field Description	ビジネス・オブジェクトを生成する際、ユーザーはビジネス・オブジェクト属性名として IDoc セグメントのフィールド名またはフィールド記述を選択できます。

WebSphere Business Integration システムのコンポーネント名では、英数字とアンダースコアー文字 (_) のみをサポートします。したがって、生成されたビジネス・オブジェクト定義でコンポーネントを命名する際、SAPODA によって、IDoc セグメントのフィールド記述またはフィールド名内の特殊文字は、アンダースコアー文字に置き換えられます。例えば、SAPODA は次に示す SAP 記述内のスペース、括弧、およびピリオドを、対応する属性名ではアンダースコアーに変更します。

```
Partner function (e.g. sold-to party, ship-to party)
```

SAPODA は生成されたビジネス・オブジェクト定義内で上記の記述を以下のように表します。

```
Partner_function__e_g__sold_to_party__ship_to_party__
```

SAPODA は、ビジネス・オブジェクト定義内のすべての属性名が固有であることを保証します。1 つの IDoc に同じフィールド記述を持つ複数のフィールドが存在する場合、SAPODA は生成した属性名にカウンター・サフィックスを追加します。

属性を命名する際に、変更された属性名が次の条件に該当すると、SAPODA はストリングを属性名の前に付加します。

- 数字で開始される場合は、A_ を前に付加します。
- アンダースコアー文字 (_) で開始される場合は、A を前に付加します。

重要: 属性名は、ビジネス・オブジェクトを生成した後でいつでも変更できます。しかし、属性名を変更する場合は、そのアプリケーション固有情報を変更しません。コネクターはこのテキストを使用して、ビジネス・オブジェクト属性に対応する IDoc フィールドを識別します。詳細については、 150 ページの『アプリケーション固有情報: データ・レコード・ビジネス・オブジェクト』を参照してください。

IDoc 拡張に対応した命名規則

SAPODA が IDoc 拡張に基づいてビジネス・オブジェクト定義を生成する場合は、144 ページの『ビジネス・オブジェクトの命名規則』に示した命名規則とはやや異なる命名規則が使用されます。この場合、表 24 に示すように、拡張名が含まれます。

表 24. IDoc 拡張に対応した命名規則

IBM WebSphere のビジネス・オブジェクトまたは属性	**名前**	**タイプ**
親ラッパー・ビジネス・オブジェクト	*BOprefix_BasicIDocType_ExtensionName*	なし
制御レコード・ビジネス・オブジェクト	Control_record	sap_idoccontrol
データ・レコード・ビジネス・オブジェクト	Data_record	*BOprefix_BasicIDocType*_cwdata
データ・レコード子ビジネス・オブジェクト	*BOprefix_BasicIDocType_ ExtensionName_IDocSegmentName*	*BOprefix_BasicIDocType_ ExtensionName_IDocSegmentName*
データ・レコード属性	*IDocFieldText* または *IDocFieldName*	String

拡張を指定する AppSpecificInfo プロパティーの構文については、『親ラッパー・ビジネス・オブジェクト』を参照してください。

重要: 統合ブローカーが InterChange Server Express である場合、IDoc 拡張子に対応するビジネス・オブジェクト定義をリポジトリーにロードする際には注意が必要です。基本 IDoc Type に対応するビジネス・オブジェクト定義がリポジトリー内に既に存在し、その名前が基本 IDoc Type に拡張を追加した名前と一致する場合、競合が発生することがあります。この競合は、手動で解決する必要があります。

親ラッパー・ビジネス・オブジェクト

親ラッパー・ビジネス・オブジェクトの名前は、基本 IDoc タイプの前に、ユーザー定義のプレフィックスとアンダースコアー (_)、例えば sap_ を付けたものです。

親ラッパー・ビジネス・オブジェクトには、Dummy_key、Control_record、Data_record、および TransactionId の 4 つの属性が含まれます。

IDoc のトップレベル・オブジェクトの Dummy_key 属性は、制御レコードおよびデータ・レコードのキー・フィールドをトップレベルのオブジェクトの Dummy_key にマップする目的で使用します。コネクターは、Dummy_key のマッピングを次のように処理します。

1. Dummy_key 属性の属性レベルの ASI は、値の設定に使用される属性のパスとして構成されます。つまり、属性レベルの ASI は、トップレベルのオブジェクトにマップされる属性のビジネス・オブジェクト・ツリー内でのパスに設定されます。値のペアの区切り文字は ; (セミコロン) です。子からのキー属性のパスの区切り文字は : (コロン) です。外部キー (FK) には、絶対パスを指定する必要があります。

   例えば、
   DummyKey;FK=Data_record:sap_orders05_e2edk01005:IDOC_document_number" となります。
2. コネクターは、このパスの中で複数カーディナリティーのオブジェクトを検出した場合、このコンテナーから最初の子インスタンスを使用します。これは、(ビジネス・オブジェクト・ツリーのどこに存在する複数カーディナリティーのオブジェクトであろうと) すべての複数カーディナリティーのオブジェクトに当てはまります。
3. ASI が正しくない場合、またはマップされた属性値が空の場合、コネクターはイベントに失敗し、イベントを SAPALE_Error_Queue の状態にします。これは、ASI がオブジェクト・タイプ値を Dummy_key として設定するように構成されている場合にも当てはまります。Dummy_key 属性は単純タイプの属性の値のみを格納することができます。

Control_record 属性および Data_record 属性は、単一カーディナリティーの子ビジネス・オブジェクトを表します。

Control_record 属性のタイプは sap_idoccontrol です。このビジネス・オブジェクト定義は ALE Module とともに提供されます。

Data_record 属性のタイプは *BOprefix_BasicIDocType*_cwdata です。このビジネス・オブジェクト定義には、SAP アプリケーションの基本 IDoc タイプの IDoc セグメント定義に応じて、1 つ以上の子ビジネス・オブジェクトが含まれます。

TransactionId 属性の値に応じて、サービス呼び出し要求を処理する際にコネクターが TID を管理するかどうかが決まります。要求処理の際に TID 管理を行いたくない場合は、TransactionID 属性の値を設定しないでください。

親ラッパー・ビジネス・オブジェクトのアプリケーション固有情報は、以下の項目を表します。

- 作成される IDoc のタイプ。
- IDoc 拡張 — ビジネス・オブジェクトが基本 IDoc タイプのカスタマイズによって生成される場合のみ設定します。 IDoc 定義ファイルの生成の詳細については、36 ページの『SAPODA を使用する前に』を参照してください。

- ALE Communication Partner 情報 — 使用するデータで複数の Partner タイプ、Partner 番号、または Partner 機能が必要な場合のみ設定します。

構文

親ラッパー・オブジェクトの AppSpecificInfo プロパティーの構文は次のとおりです。

```
BasicIDocType [,Ext=ExtensionName
[,Pn=PartnerNumberOfRecipient [,Pt=
PartnerTypeOfRecipient[,Pf=PartnerFunctionOfRecipient
]]
```

構文の説明

BasicIDocType
: 基本 IDoc タイプを指定

Ext
: 拡張タイプを指定

Pn
: 受信側のパートナーの数を指定

Pt
: 受信側のパートナーのタイプを指定

Pf
: 受信側のパートナーの機能を指定

例

```
AppSpecificInfo = ALEREQ01,Pn=ALESYS2,Pt=LS,Pf=EL
```

制御レコード・ビジネス・オブジェクト

ALE Module では、すべての IDoc に対して汎用の制御レコード・ビジネス・オブジェクト定義が使用されます。これには、制御レコードの 3.x バージョン (SAP 構造体 EDI_DC) および 4.x バージョン (SAP 構造体 EDI_DC40) で使用される属性のスーパーセットが含まれています。制御レコード・ビジネス・オブジェクト定義は、ALE Module とともに提供されます。また、ビジネス・オブジェクト・リポジトリーにロードする必要があります。Business Object Designer Express を使用して、ビジネス・オブジェクトをリポジトリーにロードします。

注: repos_copy コマンドも使用できます。

表 25 に、制御レコード・ビジネス・オブジェクトの単純属性プロパティーのリストを示します。

表 25. 制御レコード・ビジネス・オブジェクトの単純属性のプロパティー

プロパティー名	説明
Name	Name プロパティーの値は、IDoc 定義の TEXT フィールドの値を変更したものです。 SAPODA は、 144 ページの『ビジネス・オブジェクトの命名規則』で説明したとおり、名前に英数字とアンダースコアー文字 (_) のみが含まれるようにするために、特殊文字 (ピリオド、スラッシュ、スペースなど) をアンダースコアーに置き換えます。
Type	データのタイプを指定します。SAPODA はこの値を String に設定します。
MaxLength	SAPODA は MaxLength の値を IDoc 定義内の LENGTH フィールドから導出します。

表 25. 制御レコード・ビジネス・オブジェクトの単純属性のプロパティー (続き)

プロパティー名	説明
IsKey	SAPODA は、ビジネス・オブジェクトの最初の属性について、このプロパティーを `true` に設定します。
IsForeignKey	SAPODA はこの値を `false` に設定します。
IsRequired	`IsRequired` プロパティーは、属性が値を含んでいる必要があるかどうかを指定します。SAPODA は、制御レコード・オブジェクト内の `Name_of_table_structure` 属性の場合のみ、このプロパティーを `true` に設定します。
AppSpecificInfo	SAPODA は、この値を IDoc 定義の NAME フィールドから導出します。
DefaultValue	実行時値がない場合にこの属性に割り当てる値を指定します。SAPODA は、このプロパティーの値を設定しません。

重要: 属性の値が制御レコード・ビジネス・オブジェクト内で `CxIgnore` または `CxBlank` のいずれかに設定されている場合、その値はコネクターによって IDoc 制御レコードでのブランク・スペースとして設定されます。

データ・レコード・ビジネス・オブジェクト

IDoc 定義ファイルには、IDoc の構造、IDoc セグメント階層、およびセグメントを構成するフィールドに関する情報が含まれています。SAPODA は、データ・レコード・ビジネス・オブジェクトとその子ビジネス・オブジェクトを生成するための入力として IDoc を使用します。子の数は、SAP アプリケーションの基本 IDoc タイプの IDoc セグメント定義に依存します。

データ・レコード・ビジネス・オブジェクトのトップレベルは、基本 IDoc タイプに対応します。このトップレベル・ビジネス・オブジェクトには、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 (各要素が各 IDoc セグメントに対応) を表す属性が含まれています。子ビジネス・オブジェクトの構造と階層は、基本 IDoc タイプの IDoc セグメントの構造と階層に一致します。

SAPODA を用いてシステムから IDoc を生成すると、SAP システム自体に呼び出しを行なうことによって、データ・レコード・オブジェクトおよびその子ビジネス・オブジェクトが作成されます。ビジネス・オブジェクトのさまざまなプロパティーの設定方法を示す上で役立つように、IDoc 定義ファイルからのフィールドがこのセクションで使用されます。システムからの IDoc の生成には、SAP システムに対して行なわれた呼び出しからの該当フィールドが使用されます。

このセクションで説明する内容は次のとおりです。

- 『属性: データ・レコード・ビジネス・オブジェクト』
- 150 ページの『アプリケーション固有情報: データ・レコード・ビジネス・オブジェクト』
- 151 ページの『ビジネス・オブジェクトと IDoc との関係図』

属性: データ・レコード・ビジネス・オブジェクト

表 26 は、データ・レコード・ビジネス・オブジェクトの各単純属性のプロパティーを説明したものです。SAPODA は、以下のプロパティーを生成します。

表 26. 単純属性: データ・レコード・ビジネス・オブジェクト

プロパティー名	説明
Name	Name プロパティーの値は IDoc 定義の中の NAME または TEXT フィールドの修正された値です。 SAPODA は、 144 ページの『ビジネス・オブジェクトの命名規則』で説明したとおり、名前に英数字とアンダースコアー文字 (_) のみが含まれるようにするために、特殊文字 (ピリオド、スラッシュ、スペースなど) をアンダースコアーに置き換えます。
Type	データのタイプを指定します。SAPODA はこの値を String に設定します。
MaxLength	SAPODA は MaxLength の値を IDoc 定義内の LENGTH フィールドから導出します。
IsKey	SAPODA は、各ビジネス・オブジェクトの最初の属性について、このプロパティーを true に設定します。その他すべての属性については、SAPODA はこの値を false に設定します。
IsForeignKey	SAPODA はこの値を false に設定します。
IsRequired	属性が値を含んでいる必要があるかどうかを指定します。 SAPODA はこの値を false に設定します。
AppSpecificInfo	SAPODA は AppSpecificInfo プロパティーの値を、IDoc 定義内の Name フィールドの値の前にオフセット値と + 文字を付加して設定します。例えば、40 のオフセットを持つ SIGN という名前のセグメント・フィールドの場合、 AppSpecificInfo に対して 40+SIGN という値を設定します。 詳細については、 150 ページの『単純属性のアプリケーション固有情報』を参照してください。
DefaultValue	実行時値がない場合にこの属性に割り当てる値を指定します。 SAPODA は、このプロパティーの値を設定しません。

重要: データ・レコード・ビジネス・オブジェクトの単純属性は、CxIgnore および CxBlank という 2 つの特殊値を持つ場合があります。CxIgnore または CxBlank に対応する単純属性セットはセグメント・データ・ストリング内のブランク・スペースで表されます。SAP はこれらの属性を、アプリケーション・フィールドにスペース文字を 1 つ置くことで処理します。

表 27 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、データ・レコード・ビジネス・オブジェクトの各属性のプロパティーを説明したものです。SAPODA は、以下のプロパティーを生成します。

表 27. 子ビジネス・オブジェクトを表す属性

プロパティー名	説明
Name	SAPODA はこの値を BOprefix_BasicIDocTypeIdocSegmentName に設定します。例えば、SAP_E2ALER1001 のようになります。
Type	SAPODA はこの値を以下のように設定します。 BOprefix_BasicIDocTypeIdocSegmentName
ContainedObjectVersion	SAPODA はこの値を 1.0.0 に設定します。
Relationship	SAPODA はこの値を containment に設定します。
IsKey	SAPODA はこの値を false に設定します。
IsForeignKey	SAPODA はこの値を false に設定します。

表 27. 子ビジネス・オブジェクトを表す属性 (続き)

プロパティー名	説明
IsRequired	IsRequired プロパティーでは、子ビジネス・オブジェクトが必ず存在しなければならないかどうかを指定します。SAPODA は、IDoc 定義の対応するセグメントの STATUS フィールドの値が OPTIONAL である場合には、この値を false に設定します。SAPODA は、IDoc 定義の STATUS フィールドの値が MANDATORY である場合には、このプロパティーを true に設定します。
AppSpecificInfo	AppSpecificInfo プロパティーには、階層レベルや、許容されるセグメントの最小数および最大数についての情報が格納されます。詳細については、 151 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。
Cardinality	IDoc 定義内の LOOPMAX フィールドの値が 1 である場合、SAPODA はこの値を 1 に設定します。LOOPMAX の値が 1 より大きい場合、SAPODA はこの値を n に設定します。

アプリケーション固有情報: データ・レコード・ビジネス・オブジェクト

このセクションでは、コネクターが AppSpecificInfo プロパティーの値を使用する方法を、次の 3 つの場合について説明します。

- 『ビジネス・オブジェクト・レベルでのアプリケーション固有情報』
- 『単純属性のアプリケーション固有情報』
- 151 ページの『子を表す属性のアプリケーション固有情報』

ビジネス・オブジェクト・レベルでのアプリケーション固有情報: コネクターは、AppSpecificInfo プロパティーの値をデータ・レコードおよびそのそれぞれの子のビジネス・オブジェクト・レベルで使用し、関係付けられている IDoc およびそのセグメントの名前を取得します。

- データ・レコード・ビジネス・オブジェクトでのアプリケーション固有情報の構文は次のとおりです。

  *IDocType*_CWDATA

  例えば、ALERQ01 という名前の IDoc である場合、SAPODA は AppSpecificInfo プロパティーの値を ALERQ01_CWDATA として作成します。
- データ・レコード・ビジネス・オブジェクトの子のアプリケーション固有情報の値は、対応するセグメント名です。例えば、E2ALER1001 および E2ALEQ1 という名前の 2 つのセグメントを持つ IDoc ALERQ01 の場合、SAPODA は 2 つの子ビジネス・オブジェクトに対して AppSpecificInfo プロパティーの値を自動的に作成します。
  - 最初の子オブジェクト: E2ALER1001
  - 2 番目の子オブジェクト: E2ALEQ1

単純属性のアプリケーション固有情報: コネクターは、単純属性の AppSpecificInfo プロパティーの値を使用して、SAP でのフィールド名と、データ・ストリングでのその位置 (オフセット) を取得します。

オフセット値は、データ・ストリングにおける、属性値の先頭の文字の位置です。オフセット値は、IDoc 定義の最初のフィールドの BYTE_FIRST 値の値を、特定の属性の BYTE_FIRST 値から差し引くことで算出されます。この値は、MaxLength プロパティーとともに、IDoc セグメントのデータ・ストリングを作成するために使用されます。

単純属性の AppSpecificInfo プロパティーの構文は以下のようになります。

OffsetNumber+IDoc*FieldName*

例えば、40 のオフセットを持つ SIGN という名前のセグメント・フィールドは AppSpecificInfo に対して以下の値を持ちます。

```
40+SIGN
```

子を表す属性のアプリケーション固有情報: 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の AppSpecificInfo プロパティーの値を使用して、階層レベルや、許容されるセグメントの最小数および最大数についての情報を取得します。SAPODA は、IDoc 定義内の LEVEL、LOOPMIN および LOOPMAX フィールドから情報を取得することによって、これらの属性に対して AppSpecificInfo プロパティーを設定します。

ビジネス・オブジェクトと IDoc との関係図

図 53 に、WebSphere データ・レコード・ビジネス・オブジェクトと、SAP アプリケーションの IDoc 定義との関係を示します。

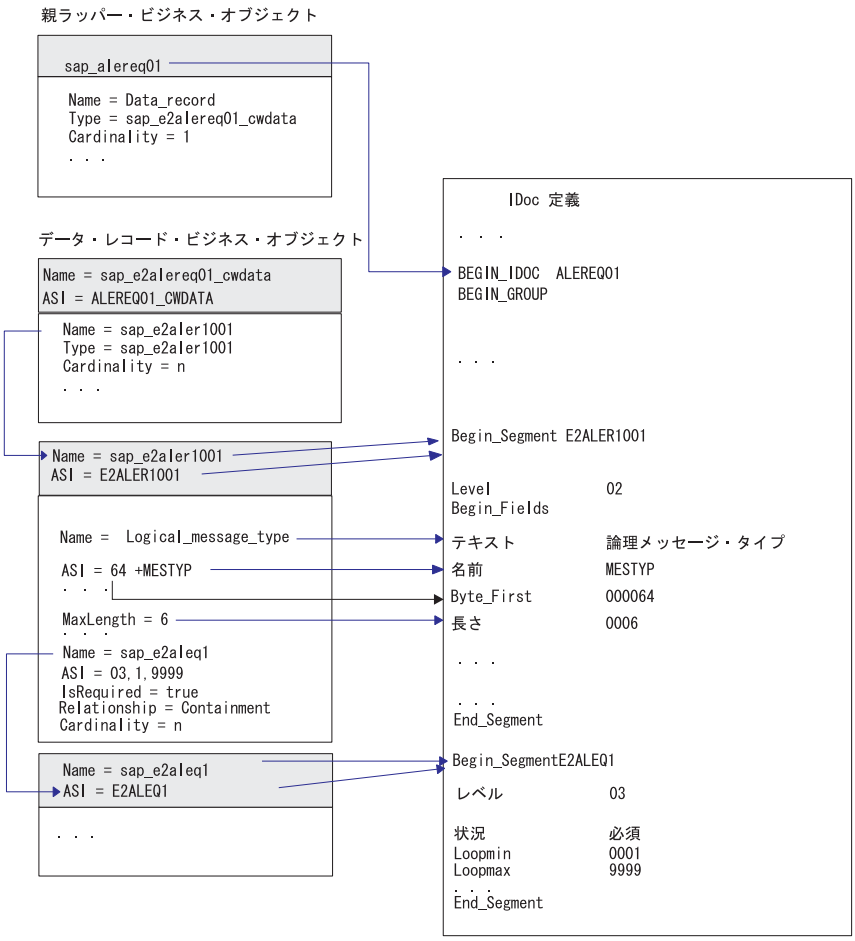


図 53. データ・レコード・ビジネス・オブジェクトと IDoc 定義フィールドとの関係

サポートされる動詞

ALE Module についてサポートされる動詞は、SAP がその ALE インターフェースを通じてサポートする動詞によって制限されます。SAPODA はビジネス・オブジェクト定義で、Create、Update、Delete、および Retrieve 動詞を生成します。各動詞を実装するには、SAP での ALE 構成に関する知識が必要です。

SAPODA は動詞の `AppSpecificInfo` と、親ラッパー・ビジネス・オブジェクトの `AleOutboundVerbs` メタ動詞を生成します。しかし、`AppSpecificInfo` の 1 つのパラメーターにのみ値を取り込みます。これはサービス呼び出し要求処理に使用する

ビジネス・オブジェクト・ハンドラーを指定します。その他すべての処理については、ビジネス・オブジェクト定義を手動で変更して、次のように特定の情報を追加または削除する必要があります。

- ビジネス・オブジェクトをイベント処理に使用する場合は、以下の AppSpecificInfo プロパティーの値を指定する必要があります。
  - 親ラッパー・ビジネス・オブジェクトの動詞 — 動詞を一意的に識別するパラメーターの値。使用する ALE 構成の要件に応じて、メッセージ・タイプ、メッセージ・コード、およびメッセージ機能を指定します。これらの変更は、ビジネス・オブジェクト定義をリポジトリーにインポートした後で行ってください。

    重要: SAPODA は、ビジネス・オブジェクト・ハンドラーを指定する AppSpecificInfo 値を挿入します。これは、コネクターが要求を処理するためにのみ使用します。SAPODA は、メッセージ・パラメーターの値は挿入しません。イベント処理に ALE Module を使用している場合は、メッセージ・パラメーターの値を手動で追加する必要があります。
  - 親ラッパー・ビジネス・オブジェクトの AleOutboundVerbs メタ動詞 — イベント処理でサポートされる動詞の、コンマで区切られたリスト。
- ビジネス・オブジェクトを要求処理に使用する場合は、以下の AppSpecificInfo プロパティーの値を指定する必要があります。
  - 親ラッパー・ビジネス・オブジェクトの動詞 — コネクターが適切なビジネス・オブジェクト・ハンドラーを判別できるように、ビジネス・オブジェクト・ハンドラーのパッケージおよびクラス名を指定します。SAPODA は各標準動詞の AppSpecificInfo プロパティーに、AppSpecificInfo = sap.sapalemodule.VSapALEBOHandler という値を挿入します。
  - ラッパー・ビジネス・オブジェクトを使用して複数の IDoc 親ビジネス・オブジェクトを処理する場合は、その複数の IDoc ラッパー・ビジネス・オブジェクトの各動詞の AppSpecificInfo プロパティーに、ビジネス・オブジェクト・ハンドラーのパッケージおよびクラス名を追加する必要があります。

各親ラッパー・ビジネス・オブジェクトについて、SAPODA は Create、Retrieve、Update、および Delete 動詞を生成します。これらの動詞のそれぞれについて、次に示す AppSpecificInfo 値が生成されます。

```
sap.sapalemodule.VSapALEBOHandler,MsgType=;MsgCode=;MsgFunction=
```

AppSpecificInfo プロパティー: 親ラッパー動詞

親ラッパー・ビジネス・オブジェクトの動詞の AppSpecificInfo プロパティーの構文は、そのビジネス・オブジェクトがアプリケーション・イベントを表すか、サービス呼び出し要求を表すかによって、次のように異なります。

アプリケーション・イベントの構文

[*BOHandler*],MsgType=*messageType*;MsgCode=[*messageCode*];MsgFunction=[*messageFunction*]

注: コネクターは動詞を判別するために、制御レコード内の値を、動詞の AppSpecificInfo プロパティーで指定された値と照合します。

サービス呼び出し要求の構文

BOHandler[,MsgType=*messageType*;MsgCode=[*messageCode*];MsgFunction=[*messageFunction*]]

構文の説明

BOHandler 要求処理ビジネス・オブジェクト・ハンドラーを指定します。デフォルト値は次のとおりです。sap.sapalemodule.VSapALEBOHandler

MsgType ALE で IDoc に対して構成されるメッセージ・タイプを指定します。

MsgCode ALE の IDoc のために構成されたメッセージ・コードを指定します。コネクターは、MsgType が動詞を一意的に識別しない場合のみ値を必要とします。ただし、ALE 構成で必要とされる場合は値を指定してください。

MsgFunction ALE の IDoc のために構成されたメッセージ機能を指定します。コネクターは、MsgType および MsgCode が動詞を一意的に識別しない場合のみ値を必要とします。ただし、ALE 構成で必要とされる場合は値を指定してください。

AppSpecificInfo プロパティー: 親ラッパー動詞

親ラッパー・ビジネス・オブジェクトの AleOutboundVerbs 動詞の AppSpecificInfo プロパティーで、コネクターがアプリケーション・イベント処理のためにサポートする必要のある各動詞を、コンマで区切って指定します。

重要: SAPODA は、Create、Retrieve、Update、および Delete 動詞の値を生成します。定義が生成された後、コネクターでサポートする必要のない動詞を手動で削除する必要があります。

次に示す例では、コネクターに対して、アプリケーション・イベントを処理するために Create および Update 動詞をサポートするように指示しています。

```
[Verb]
Name = AleOutboundVerbs
AppSpecificInfo = Create, Update
[End]
```

ラッパー・ビジネス・オブジェクトがある複数の IDocs の処理

注: このセクションは、サービス呼び出し要求処理のみを対象とします。

複数の IDoc を処理する場合、ALE Module はトップレベル・ビジネス・オブジェクトとしてラッパー・ビジネス・オブジェクトを必要とします。複数 IDoc ラッパー・ビジネス・オブジェクトには、IDoc 親ラッパー・ビジネス・オブジェクトの配列を表す属性が含まれています。

各親ラッパー・ビジネス・オブジェクトについて、SAPODA は Create、Retrieve、Update、および Delete 動詞を生成します。これらの動詞のそれぞれについて、次に示す AppSpecificInfo 値が生成されます。

```
sap.sapalemodule.VSapALEBOHandler,MsgType=;MsgCode=;MsgFunction=
```

図 54 に、トップレベル・ラッパー・オブジェクトと、その子 IDoc ビジネス・オブジェクトとの関係を示します。

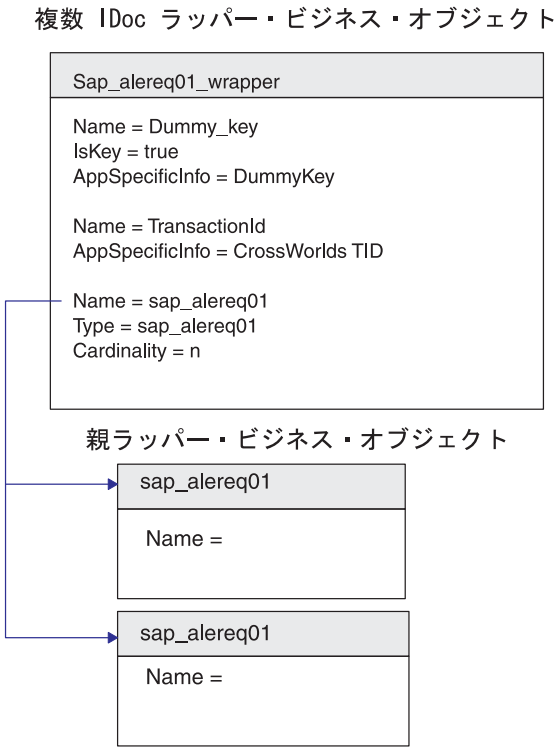


図 54. 子ビジネス・オブジェクトを含んでいるラッパー・ビジネス・オブジェクト

複数 IDoc ラッパー・オブジェクトの例

複数 IDoc ラッパー・ビジネス・オブジェクトのサンプル定義を次に示します。

```
[BusinessObjectDefinition]
Name = sap_alereq01_wrapper
Version = 1.0.0
AppSpecificInfo =

[Attribute]
Name = Dummy_key
Type = String
Cardinality = 1
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = true
AppSpecificInfo = DummyKey
DefaultValue =
[End]

[Attribute]
Name = TransactionId
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
AppSpecificInfo = CrossWorlds TID
DefaultValue =
[End]

[Attribute]
Name = sap_alereq01
Type = sap_alereq01
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
DefaultValue =
[End]

[Verb]
Name = Create
AppSpecificInfo = sap.sapalemodule.VSapALEBOHandler,MsgType=;MsgCode=;MsgFunction=
[End]

[Verb]
Name = Retrieve
AppSpecificInfo = sap.sapalemodule.VSapALEBOHandler,MsgType=;MsgCode=;MsgFunction=
[End]

[Verb]
Name = Update
AppSpecificInfo = sap.sapalemodule.VSapALEBOHandler,MsgType=;MsgCode=;MsgFunction=
[End]

[Verb]
Name = Delete
AppSpecificInfo = sap.sapalemodule.VSapALEBOHandler,MsgType=;MsgCode=;MsgFunction=
[End]
```

複数 IDoc ラッパー: 子ビジネス・オブジェクトを表す属性

表 28 に、複数 IDoc ラッパー・ビジネス・オブジェクトで子ビジネス・オブジェクトを表す属性のプロパティーについてリストし、説明します。

表 28. 複数 IDoc ラッパー: 子ビジネス・オブジェクトを表す属性

プロパティー名	説明
Name	値を、SAPODA が生成した親ビジネス・オブジェクトの名前に設定します。
Type	値を、SAPODA が生成した親ビジネス・オブジェクトの名前に設定します。
ContainedObjectVersion	この値を `1.0.0` に設定します。
Relationship	子ビジネス・オブジェクトは親ビジネス・オブジェクトに包含されているため、この値は `containment` です。
IsKey	この値を `false` に設定します。
IsForeignKey	この値を `false` に設定します。
IsRequired	この値を `false` に設定します。
AppSpecificInfo	このプロパティーは、ALE Module で子ビジネス・オブジェクトを表す属性に対しては使用されていません。
Cardinality	トップレベル・ラッパー・ビジネス・オブジェクトで IDoc 親ビジネス・オブジェクトを表す属性の値をカーディナリティー `n` に設定します。

第 4 部 RFC Server Module

第 12 章 RFC Server Module の概要

この章では、Adapter for mySAP.com の RFC Server Module の概要について説明します。RFC Server Module を使用すると、統合ブローカーは、RFC 呼び出しをサポートする SAP アプリケーションからビジネス・オブジェクトを受け取ることが可能になります。RFC Server Module は、RFC 対応機能を使用するすべての SAP アプリケーションを、それらに対するサーバーとして動作することによってサポートします。

この章の内容は以下のとおりです。

RFC Server Module のコンポーネント

RFC Server Module は、SAP アプリケーションからの直接の RFC 呼び出しをサポートする、Java で記述されたコネクター・モジュールです。これは、VisionConnectorAgent クラスを実装することで、Vision コネクター・フレームワークを拡張します。RFC Server Module は、Java および C で記述された SAP RFC ライブラリーを使用します。これにより、外部プログラムは SAP アプリケーションと通信できます。

160 ページの図 55 に、RFC Server Module の全体的なアーキテクチャーを示します。RFC Server Module は、コネクター・フレームワーク、RFC Server に対応するコネクターのアプリケーション固有のコンポーネント、RFC Server 固有ビジネス・オブジェクト・ハンドラー、リスナー・スレッド、および SAP RFC ライブラリーから構成されています。

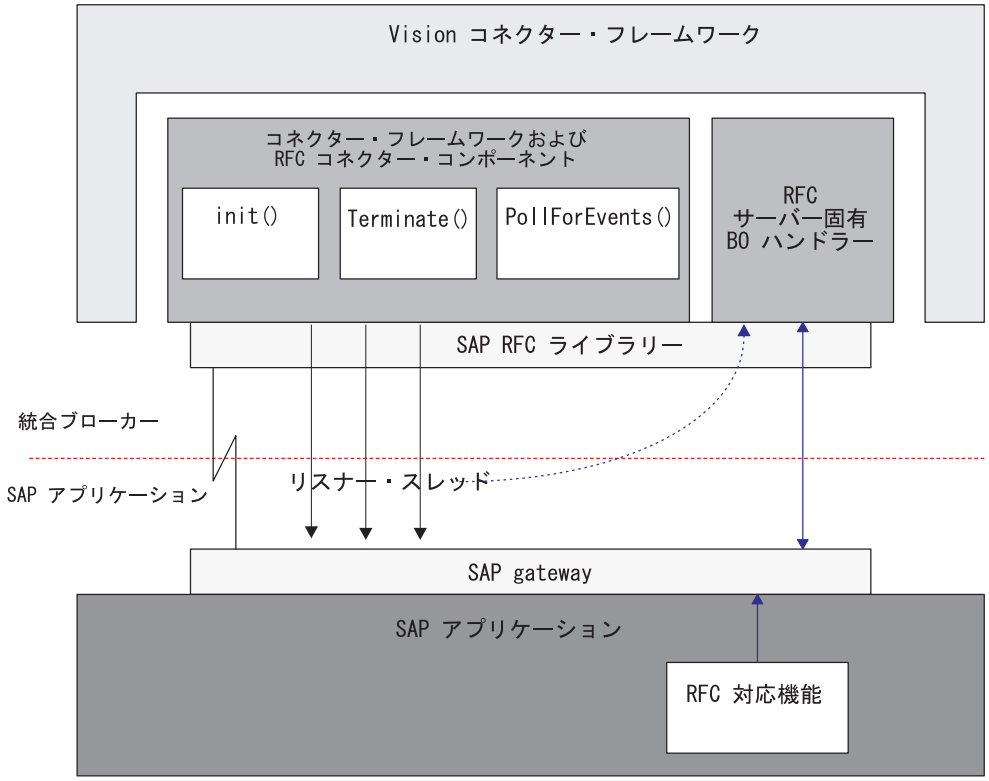


図 55. RFC Server Module のアーキテクチャー

RFC Server Module のコンポーネントは、以下の動作を実行します。

- SAP RFC ライブラリーと SAP Gateway を使用して、SAP アプリケーションへのハンドルを開くリスナー・スレッドを作成します。各リスナー・スレッドは、SAP アプリケーションへの単一のハンドルを開きます。
- SAP アプリケーション内の RFC 対応機能からの要求を処理します。
- SAP アプリケーションへの接続を終了します。

リスナー・スレッド

リスナー・スレッドは、RFC Server Module と SAP アプリケーションとの間のすべての RFC 呼び出しを処理します。コネクターは始動時に、構成可能な数のリスナー・スレッドを作成します。各リスナー・スレッドは、SAP Gateway へのハンドルを開きます。

リスナー・スレッドは、以下の動作を実行します。

- プログラム ID を使用して SAP Gateway に登録します。
- サポートする RFC 対応機能を、SAP Gateway に対して識別します。
- 利用可能な最初のスレッドを使用して、サポートする RFC 対応機能からのイベントを選出します。
- 対応するビジネス・オブジェクトの Server 動詞に基づいて RFC Server 固有ビジネス・オブジェクト・ハンドラーのインスタンスを生成した後、SAP Gateway でイベント・データを検索します。
- ビジネス・オブジェクトに RFC イベント・データを取り込み、戻されたビジネス・オブジェクト・データを RFC イベント・データに変換します。

- SAP Gateway を経由して、RFC 対応機能に応答を戻します。

注: スレッドは、それがサポートする RFC 対応機能からのイベントを、同期的な方法で連続的に listen します。

RFC server 固有ビジネス・オブジェクト・ハンドラー

RFC Server 固有ビジネス・オブジェクト・ハンドラーは、SAP アプリケーション内の各 RFC 対応機能に固有です。各ビジネス・オブジェクト・ハンドラーは、リスナー・スレッドによってインスタンス化され、関連するビジネス・オブジェクトを呼び出します。

RFC Server Module は、SAP アプリケーションに対するサーバーとして動作するため、イベントを SAP アプリケーションから統合ブローカーに「プッシュ」(送信) します。この動作は、アプリケーションに対するイベント・ポーリングを実行するほかのモジュールとは大きく異なります。この違いがあるため、RFC Server 固有ビジネス・オブジェクト・ハンドラーが実行するタスクは、ほかのビジネス・オブジェクト・ハンドラーが実行するタスクとは異なります。

インスタンス化された RFC Server 固有ビジネス・オブジェクト・ハンドラーは、以下の動作を実行します。

- RFC イベント・データを検索し、関連する SAP 用の WebSphere ビジネス・オブジェクトに取り込みます。
- ビジネス・オブジェクトを統合ブローカーに渡し、リターンとしてビジネス・オブジェクトを受け取ります。

  ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクトの Server 動詞のアプリケーション固有情報を使用して、どのコラボレーションがビジネス・オブジェクト・データを処理するかを決定します。

  – ビジネス・オブジェクトの Server 動詞で有効なコラボレーションを指定する必要があります。コネクターにプッシュされるイベントに対してコラボレーションを明示的にサブスクライブさせることはできないため、RFC Server 固有ビジネス・オブジェクト・ハンドラーは、適切なコラボレーションを判別した後で、そのコラボレーションのインスタンスを生成する必要があります。
- 戻されたビジネス・オブジェクト・データを RFC イベント・データに変換します。
- RFC イベント・データを SAP アプリケーションに戻します。

RFC Server Module の動作方法

RFC Server Module は、`init()`、`terminate()`、`pollForEvents()`、および `process()` の各メソッドを実装します。

このセクションで説明する内容は次のとおりです。

- 162 ページの『初期化と終了』
- 162 ページの『ビジネス・オブジェクトの処理』
- 163 ページの『RFC 対応機能のサポート』

初期化と終了

init() メソッドは、SAP Gateway へのハンドルを開く構成可能な数のリスナー・スレッドを作成するメイン・スレッドを作成します。コネクターは、初期化に失敗すると terminate() メソッドを使用して終了します。コネクターは、SAP Gateway への接続を切断することで終了します。

初期化プロセス中に、RFC Server Module は指定された Program ID を使用して SAP Gateway に登録します。この Program ID は、RfcProgramID コネクター構成プロパティーを使用して設定し、SAP アプリケーション内の TCP/IP ポートとしてセットアップする必要があります。TCP/IP ポートのセットアップの詳細については、165 ページの『RFC Server Module の SAP gateway への登録』を参照してください。

ビジネス・オブジェクトの処理

RFC Server Module の WebSphere ビジネス・オブジェクトのすべての処理は、SAP アプリケーションの RFC 対応機能によって開始されます。RFC Server Module では、1 つの RFC Server 固有ビジネス・オブジェクト・ハンドラーは 1 つの RFC 対応機能だけをサポートするため、SAP アプリケーションでサポートされる機能ごとに、関連付けられた RFC Server 固有ビジネス・オブジェクト・ハンドラーが必要になります。さらに、それぞれの RFC Server 固有ビジネス・オブジェクト・ハンドラーについて、関連付けられたビジネス・オブジェクトが必要になります。

図 56 に、RFC Server Module のビジネス・オブジェクト処理を示します。

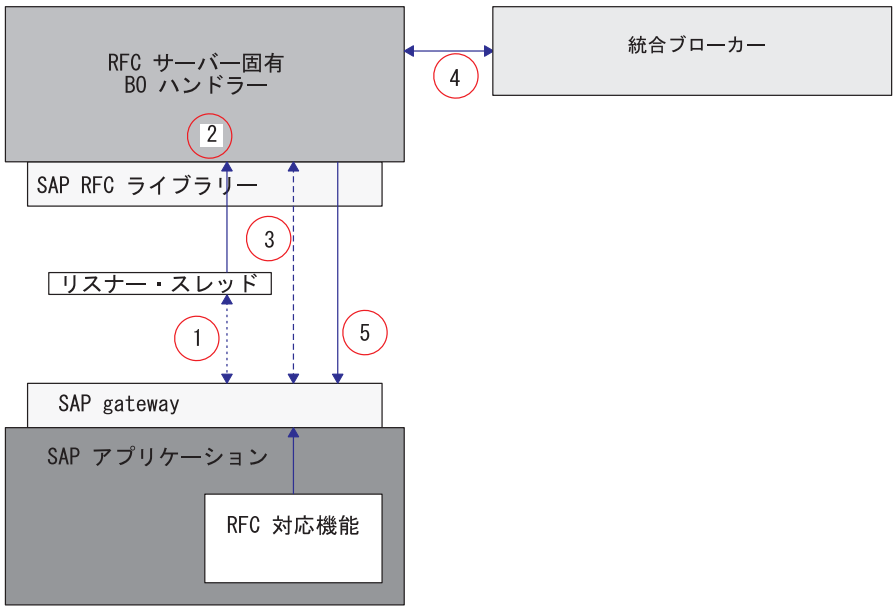


図 56. ビジネス・オブジェクトの処理

RFC Server Module のビジネス・オブジェクト処理は、以下の方法で実行されます。

1. リスナー・スレッドは、サブスクライブされたイベントを SAP Gateway から選出し、対応する RFC 対応機能の名前を RFC Server 固有ビジネス・オブジェクト・ハンドラーと突き合わせます。

2. リスナー・スレッドは、SAP Gateway での RFC イベントからのデータに基づいて適切な RFC Server 固有ビジネス・オブジェクト・ハンドラーのインスタンスを生成した後、対応するビジネス・オブジェクトのインスタンスを生成します。
3. RFC Server 固有ビジネス・オブジェクト・ハンドラーは、SAP Gateway で RFC インターフェース・データを検索し、SAP 用の WebSphere ビジネス・オブジェクトに取り込みます。
4. RFC Server 固有ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクトを統合ブローカーに渡します。
5. ビジネス・オブジェクト・ハンドラーは、統合ブローカーから戻されたビジネス・オブジェクトを受け取り、RFC インターフェースに変換して、SAP Gateway に戻します。

RFC Server Module は SAP Gateway を使用して、イベントを処理する順序やイベントの状況を保守します。リスナー・スレッドは同期呼び出しを行うため、イベントが SAP Gateway に戻された場合にのみ、そのイベントは正常に処理されたと見なされます。

注: RFC 対応モジュールが Return Structure または Return Table を持つ場合、コネクターはメッセージ・タイプ A (打ち切り) および E (エラー) の有無を調べ、イベントが正常に処理されたかどうかを判別します。メッセージ・タイプ A または E は、イベントの処理が失敗したことを示します。RFC 対応機能モジュールに Return Structure または Return Table がない場合は、独自のエラー処理を実装する必要があります。 Return Structure または Return Table にあるエラー・メッセージ (1 つまたは複数) が、リターン状況記述子の中に返されます。

RFC 対応機能のサポート

開発環境では、RFC 対応機能に基づいたビジネス・オブジェクト定義を生成するためのユーティリティー、SAPODA が提供されています。SAPODA は、RFC 対応機能のインターフェースを解釈し、そのインターフェース・パラメーターをビジネス・オブジェクト属性にマップし、各属性のアプリケーション固有情報を追加します。

各ビジネス・オブジェクト定義に対して、関連する RFC Server 固有ビジネス・オブジェクト・ハンドラーを生成する必要があります。これは、対応するビジネス・オブジェクトを呼び出します。ビジネス・オブジェクトと RFC Server 固有ビジネス・オブジェクト・ハンドラーの開発の詳細については、167 ページの『第 14 章 RFC Server Module のビジネス・オブジェクトの開発』を参照してください。

注: 一部の RFC 対応機能は、WebSphere ビジネス・オブジェクトの単純属性に対応する単一フィールド・パラメーターを持ちません。コネクターは、すべてのトップレベル・ビジネス・オブジェクトが、キー属性として使用される単純属性を持つことを要求します。そのため、単一フィールド・パラメーターを持たない RFC 対応機能からビジネス・オブジェクトおよびビジネス・オブジェクト・ハンドラーを生成すると、SAPODA はトップレベルのビジネス・オブジェクトに Dummy_key という名前のキー属性を作成し、それにキー属性としてマークを付け、dummy_key をこの属性のアプリケーション固有情報として追加します。Dummy_key は、コネクターにキー属性を提供して、ビジネス・オブジェ

クトを処理できるようにします。ただしコネクターは、アプリケーション・データを変更する際には Dummy_key 属性の値を無視します。

イベントの起動

RFC Server Module 用のイベントを起動するには、リモート関数呼び出しで RFC 宛先を指定する必要があります。リモート関数呼び出しの実行には、プログラマチックに実行するか、トランザクション SE37 を使用して実行するという 2 つの方法があります。プログラマチックに実行する場合は、宛先を指定する CALL FUNCTION コマンドのバリエーションを使用する必要があります。宛先に指定する値は、RFC Server Module を登録するために作成した値です。詳細については、『RFC Server Module の SAP gateway への登録』を参照してください。トランザクション SE37 を使用して、RFC ターゲット・システムを RFC 宛先を一致させる必要があります。RFC Server Module 用の RFC 宛先の作成と登録の詳細については、『RFC Server Module の SAP gateway への登録』を参照してください。

第 13 章 RFC Server Module の構成

この章では、RFC Server Module の構成について説明します。なお、必要なすべてのファイルは、Adapter for mySAP.com をインストールした際にインストール済みであることが想定されています。コネクターのインストールの詳細については、『RFC Server Module の構成プロパティー』を参照してください。

この章の内容は以下のとおりです。

- 『RFC Server Module のディレクトリーとファイル』
- 『RFC Server Module の構成プロパティー』
- 『RFC Server Module の SAP gateway への登録』

RFC Server Module のディレクトリーとファイル

RFC Server Module のディレクトリーとファイルは、¥connectors¥SAP¥ ディレクトリーに格納されています。表 29 に、RFC Server Module で使用されるディレクトリーとファイルを示します。

表 29. RFC Server Module のディレクトリーとファイル

ディレクトリー/ファイル名	説明
¥bapi¥server	コネクターのランタイム・ファイルを格納するディレクトリー。すべての RFC Server 固有 BO Handler クラス・ファイルは、このディレクトリーにコピーする必要があります。
CWSAP.jar	コネクターのクラス・ファイル。

RFC Server Module の構成プロパティー

RFC Server Module の運用を開始する前に、構成を行う必要があります。RFC Server Module を構成するには、標準構成プロパティーおよびコネクター固有のコネクター構成プロパティーを設定します。コネクター構成プロパティーの構成の詳細については、13 ページの『第 2 章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 315 ページの『付録 B. 標準構成プロパティー』 を参照してください。

RFC Server Module の SAP gateway への登録

初期化中に、RFC Server Module は SAP Gateway に登録を行います。その際には、RfcProgramId コネクター固有構成プロパティーに対して設定した値が使用されます。この値は、SAP アプリケーションで設定した値と一致している必要があります。SAP アプリケーションは、RFC Server Module がそれに対してハンドルを作成できるように構成する必要があります。

RFC Server Module を RFC 宛先として登録するには、以下の手順を行います。

1. SAP アプリケーションで、トランザクション SM59 に移動します。

2. TCP/IP 接続ディレクトリーを展開します。
3. 「Create」をクリックします (F8)。
4. 「RFC destination」フィールドに、RFC 宛先システムの名前を入力します。RFCSERVER を使用することをお勧めします。
5. 接続タイプを T (外部プログラムを TCP/IP 経由で始動) に設定します。
6. 新しい RFC 宛先の説明を入力して、「Save」をクリックします。
7. 「Activation Type」の「Registration」ボタンをクリックします。
8. Program ID を設定します。RFC 宛先 (RFCSERVER) と同じ値を使用することをお勧めします。設定が終わったら「Enter」をクリックします。

重要: コネクター固有構成プロパティー RfcProgramID は、必ず SAP アプリケーションでの Program ID 値と同じ値に設定してください。両者の値が一致しないと、ビジネス・オブジェクト処理は失敗します。

第 14 章 RFC Server Module のビジネス・オブジェクトの開発

この章では、RFC Server Module のために必要なビジネス・オブジェクトおよびビジネス・オブジェクト・ハンドラーについて説明します。また、背景情報を提供し、ビジネス・オブジェクト生成ユーティリティーの SAPODA が定義を生成する方法についても説明します。この章の読者は、コネクターがビジネス・オブジェクトを処理する方法について十分な知識を持っていることが想定されています。RFC Server Module でのビジネス・オブジェクト処理の詳細については、 159 ページの『第 12 章 RFC Server Module の概要』を参照してください。

この章の内容は以下のとおりです。

RFC Server Module 用のビジネス・オブジェクト開発は、アプリケーション固有ビジネス・オブジェクト定義と、それに関連付けられた、サポートする必要のある各 RFC 対応機能用の RFC Server 固有ビジネス・オブジェクト・ハンドラーの作成から成り立っています。SAPODA では、SAP アプリケーションのネイティブ定義を、これらについての定義を生成する際のテンプレートとして使用するため、これらの定義を生成する場合には SAPODA を使用することをお勧めします。

注: SAP は、コネクターがサポートする標準の動詞 (Create、Update、Delete、および Retrieve) にマップできる、多数のメソッドをサポートします。RFC 対応機能で使用される任意のメソッドをサポートする、ビジネス・オブジェクトや RFC Server 固有ビジネス・オブジェクト・ハンドラーを開発できます。

注: ビジネス・オブジェクトおよび RFC Server 固有ビジネス・オブジェクト・ハンドラーを作成したら、必ず RFC Server Module を SAP Gateway に登録してください。詳細については、 165 ページの『RFC Server Module の SAP gateway への登録』を参照してください。

ビジネス・オブジェクトの命名規則

RFC 対応機能インターフェースは、インポート・パラメーター、エクスポート・パラメーター、および表パラメーターから成り立っています。

- インポート・パラメーターは RFC 対応機能に渡されます。
- エクスポート・パラメーターは RFC 対応機能から戻されます。
- 表パラメーターは両方の方向で渡されます。

RFC 対応機能によっては、一部のタイプのパラメーターを持たない場合もあります。例えば、ある RFC 対応機能はインポート・パラメーターと表パラメーターのみを持つ、という場合もあります。

SAPODA は、表 30 に示すように、RFC 対応機能のインポート・パラメーター、エクスポート・パラメーター、および表パラメーターを、IBM WebSphere 属性に自動的にマップします。

表 30. 命名規則: SAP 用の WebSphere ビジネス・オブジェクト

ビジネス・オブジェクト	RFC 対応機能インターフェース
トップレベル・ビジネス・オブジェクト	*BOprefix_FunctionName* 注: この章の図では、ビジネス・オブジェクトのプレフィックスとして SAP_ または sap_ を使用します。実際にビジネス・オブジェクト定義を作成する場合は、分かりやすい独自のプレフィックスを指定できます。
属性	*フィールド記述またはフィールド名*
子ビジネス・オブジェクト	*BOprefix_FunctionParameterName*

SAPODA は、ビジネス・オブジェクト定義内のすべての属性名が固有であることを保証します。 1 つの RFC 対応機能に同じフィールド記述を持つ複数のパラメーターが存在する場合、SAPODA は生成した属性名にサフィックスとしてカウンターを追加します。

RFC 対応の機能パラメーターから属性を命名する際に、変更された属性名が次の条件に該当すると、SAPODA はストリングを属性名の前に付加します。

- 数字で開始される場合は、A_ を前に付加します。
- アンダースコアー文字 (_) で開始される場合は、A を前に付加します。

重要: 属性名は、ビジネス・オブジェクト定義を生成した後でいつでも変更できます。しかし、属性名を変更する場合は、アプリケーション固有情報を変更しません。コネクターはこの情報を使用して、属性に対応する RFC 対応機能パラメーターを識別します。アプリケーション固有の情報の詳細については、173 ページの『属性の AppSpecificInfo』を参照してください。

ビジネス・オブジェクト構造

コネクターは、各ビジネス・オブジェクト属性を RFC 対応機能のパラメーターにマップするために、RFC Server 固有ビジネス・オブジェクト・ハンドラーを使用します。コネクター、各ビジネス・オブジェクト、および各 RFC Server 固有ビジネス・オブジェクト・ハンドラーは、メタデータ主導型です。各ビジネス・オブジェクトおよびビジネス・オブジェクト・ハンドラーのメタデータで提供されるアプリケーション固有情報を利用することで、コネクターのコードを変更する必要なしに、新しいビジネス・オブジェクトやそのハンドラーに対するコネクター・サポートを追加できます。その代わりに、次のような処理が実行されます。

- コネクターはトップレベル・ビジネス・オブジェクトの動詞アプリケーション固有情報を使用して、適切な RFC Server 固有ビジネス・オブジェクト・ハンドラーのインスタンスを生成します。

重要: RFC Server Module は、SAP でイベントをポーリングしない点で他のモジュールと異なります。その代わりに、SAP がイベント・データをコネクターにプッシュします。このモジュールは標準のポーリング・プロシージャーを使用しないため、RFC Server 固有ビジネス・オブジェクト・ハンドラーは、イベントを表す各ビジネス・オブジェクトを検査して、それを処理するコラボレーションの名前を取得します。RFC Server 固有のビジネス・オブジェクト・ハンドラーは、取得した値を使用して、該当するコラボレーションのインスタンスを生成します。

- ビジネス・オブジェクト・ハンドラーは、各ビジネス・オブジェクトの属性アプリケーション固有情報を使用して、各属性とそのパラメーターとの間でのマップを実行します。

各 RFC Server 固有ビジネス・オブジェクト・ハンドラーは、ビジネス・オブジェクト間の単一および複数カーディナリティー関係をサポートします。

RFC 対応機能を基にした WebSphere ビジネス・オブジェクトが格納できる階層のレベルは 2 つまでです。そのため、すべての単純パラメーターはトップレベル・ビジネス・オブジェクトの属性に対応し、構造体および表パラメーターは子ビジネス・オブジェクトに対応します。

表 31. RFC 対応機能とビジネス・オブジェクト間の対応

RFC 対応機能インターフェース・パラメーター	**SAP 用の WebSphere ビジネス・オブジェクト**
単純フィールド	トップレベル・ビジネス・オブジェクトの属性
構造体	単一カーディナリティーの子ビジネス・オブジェクト
テーブル	複数カーディナリティーの子ビジネス・オブジェクト

注: インポート・パラメーターおよびエクスポート・パラメーターは、単純フィールド・パラメーターまたは構造体パラメーターになります。

図 57 に、WebSphere ビジネス・オブジェクトと RFC 対応機能、この例では BAPI との関連を示します。この図には、ユーザー定義の sap_bapi_po_create ビジネス・オブジェクトのフラグメントが示されています。これは、BAPI_PO_CREATE BAPI に対応します。

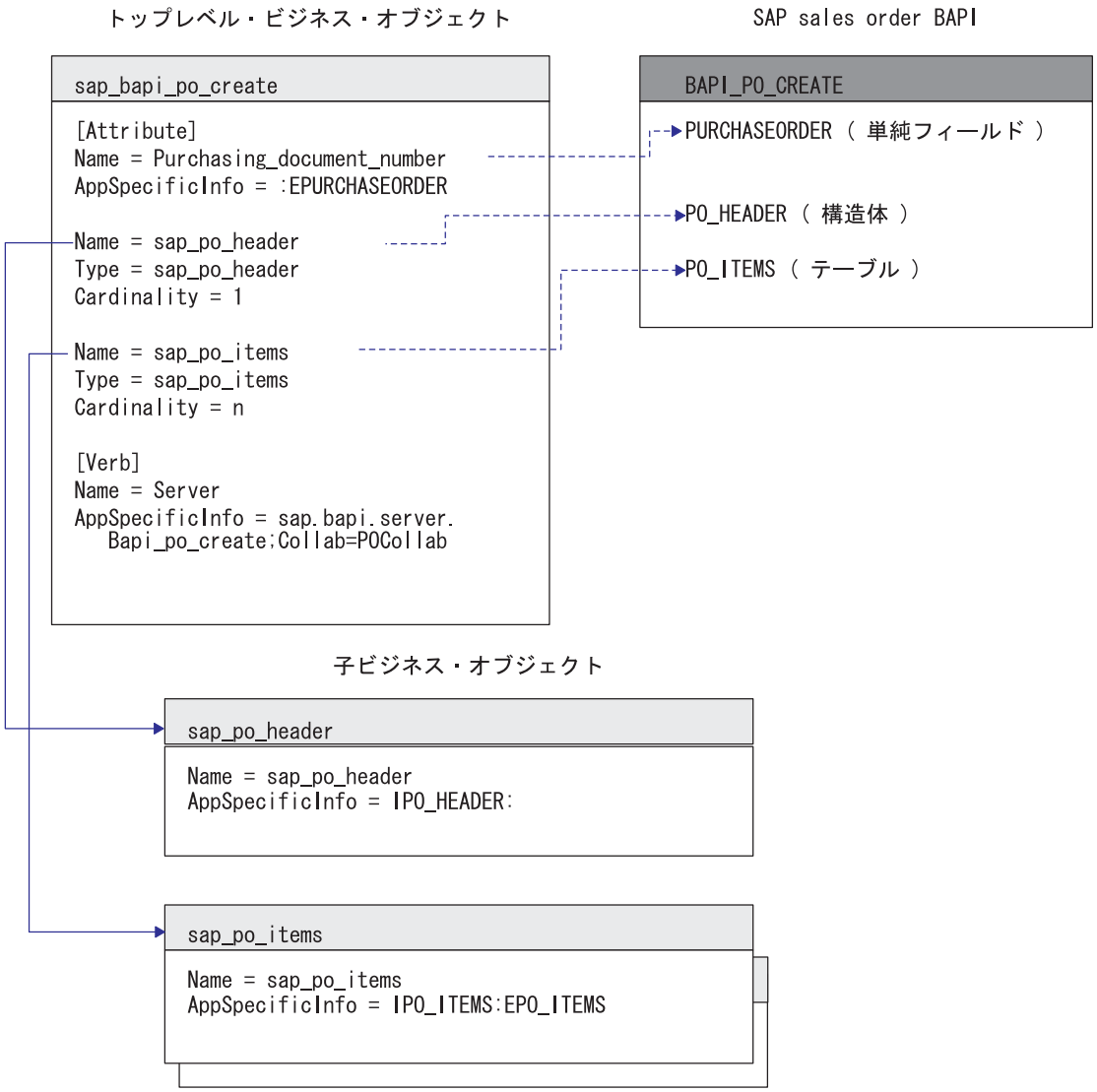


図 57. ビジネス・オブジェクトと BAPI とのマッピング

サポートされる動詞

RFC Server Module は、WebSphere Business Integration システムで使用される標準の動詞 (Create、Update、Delete、および Retrieve) をサポートします。サポートされる動詞のそれぞれについて、RFC 対応機能にメソッドを関連付けることができます。ほとんどの RFC 対応機能は、作成、検索、更新、および削除操作のいずれかをサポートします。

ビジネス・オブジェクト属性のプロパティー

トップレベル・ビジネス・オブジェクトの属性のプロパティーは、その属性が単純値を表すか、または子ビジネス・オブジェクトあるいは子ビジネス・オブジェクトの配列を表すかによって異なります。

- 表 32 に、トップレベル・ビジネス・オブジェクトの単純属性のプロパティーを示します。
- 表 33 に、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示します。

SAPODA は、以下の表の説明のように、属性プロパティーを生成します。

表 32. 単純属性: トップレベル・ビジネス・オブジェクト

プロパティー名	説明
Name	RFC 対応機能パラメーターの記述または名前から派生します。SAPODA は、特殊文字 (ピリオド、スラッシュ、およびスペースなど) をアンダースコアーで置き換えます。
Type	データのタイプを指定します。SAPODA はこの値を `String` に設定します。
MaxLength	RFC 対応機能パラメーターのフィールド長を指定します。
IsKey	その属性がキーかどうかを指定します。デフォルトでは、ビジネス・オブジェクトの最初の単純属性がキー属性になります。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をキー属性として使用することはサポートしていません。 そのため、その機能が構造体パラメーターと表パラメーターのみを提供する場合は、単純属性を最初の属性として挿入する必要があります。SAPODA は、Dummy_key 属性を最初の属性として挿入し、それにキー属性としてマークを付け、適切な値を設定します。これらの値は変更しないでください。詳細については、 92 ページの『BAPI のサポート』を参照してください。
IsForeignKey	SAPODA はこの値を `false` に設定します。
IsRequired	属性が値を含んでいる必要があるかどうかを指定します。 SAPODA はこの値を `false` に設定します。
AppSpecificInfo	関連付けられた属性に対応する RFC 対応機能の名前を格納します。書式は次のとおりです。 I*RFCFunctionParameterName*:E*RFCFunctionParameterName* アプリケーション固有の情報の詳細については、 172 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』を参照してください。
Default Value	実行時値がない場合にこの属性に割り当てる値を指定します。SAPODA は、このプロパティーの値を設定しません。

表 33 に、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示します。SAPODA は、次の表に示すプロパティーを生成します。

表 33. 子または子の配列を表す属性のプロパティー

プロパティー名	説明
Name	この値は、構造体の名前または表パラメーター名です。書式は `BOprefix_FunctionParameterName` です。
Type	この値は子ビジネス・オブジェクトのタイプ、つまり `BOprefix_FunctionParameterName` です。
ContainedObjectVersion	SAPODA はこの値を `1.0.0` に設定します。

表 33. 子または子の配列を表す属性のプロパティー (続き)

プロパティー名	説明
Relationship	SAPODA はこの値を containment に設定します。
IsKey	SAPODA はこの値を false に設定します。
IsForeignKey	SAPODA はこの値を false に設定します。
IsRequired	属性が値を含んでいる必要があるかどうかを指定します。 SAPODA はこの値を false に設定します。
AppSpecificInfo	関連付けられた属性に対応する RFC 対応の機能パラメーターの名前が入っています。書式は次のとおりです。 I*FieldName*:E*FieldName* アプリケーション固有の情報の詳細については、『ビジネス・オブジェクトのアプリケーション固有情報』を参照してください。
Cardinality	構造体パラメーターは単一カーディナリティー (1) を持ち、表パラメーターは複数カーディナリティー (n) を持ちます。

初期化属性値

表 34 に示すように、SAP のすべてのフィールドには初期値があります。コネクターがイベントを受け取ると、RFC Server 固有ビジネス・オブジェクト・ハンドラーは、これらの値を各 SAP フィールドからそれに対応するビジネス・オブジェクト属性に移動します。ビジネス・オブジェクト・ハンドラーは、文字データ型の場合を唯一の例外として、SAP からの初期値を保持します。ビジネス・オブジェクト・ハンドラーは、SAP フィールドでのスペースを、ビジネス・オブジェクト属性では CxIgnore に変換します。ほかの値を CxIgnore に変換する場合は、ビジネス・オブジェクトを作成するコンポーネントが変換を実行する必要があります。例えば、統合ブローカーが InterChange Server Express である場合は、この変換を処理するようにマップを変更します。

表 34. SAP 内のフィールド初期値

データ型	説明	ビジネス・オブジェクト・ハンドラーによって設定される初期値
C	文字	space
N	数字ストリング	*000...*
D	日付 (YYYMMDD)	00000000
T	時間 (HHMMSS)	000000
X	バイト (16 進数)	X00
I	整数	0
P	パック 10 進数	0
F	浮動小数点数	0.0

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・オブジェクト定義のアプリケーション固有情報は、ビジネス・オブジェクトの処理方法に関する、アプリケーションに依存した指示を RFC Server Module に提供します。これらの指示は、ビジネス・オブジェクト・レベルや属性レベル (単純属性と、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の両方) で指定されたり、動詞に対して指定されます。

トップレベル・ビジネス・オブジェクトの Server 動詞の AppSpecificInfo

コネクターは、トップレベル・ビジネス・オブジェクトの Server 動詞のアプリケーション固有情報の値を使用して、適切な RFC Server 固有ビジネス・オブジェクト・ハンドラーを呼び出したり、イベント処理のための宛先コラボレーションを決定したりします。Server 動詞の AppSpecificInfo プロパティーの値は、以下の情報を指定します。

- RFC Server 固有ビジネス・オブジェクト・ハンドラーのパッケージおよびクラス名
- 宛先コラボレーション

書式は次のとおりです。

```
AppSpecificInfo = bapi.server.BOHandler;Collab=CollaborationName
```

ここで、BOHandler はクラスの名前、*CollaborationName* は宛先コラボレーションの名前です。

SAPODA は、Server 動詞のアプリケーション固有情報をトップレベル・ビジネス・オブジェクトに自動的に追加します。ビジネス・オブジェクト・ハンドラーのクラス名の値には、RFC 対応機能の名前を使用します。コラボレーション名パラメーターの値は提供しません。そのため、コラボレーションの名前を手動で追加する必要があります。

注: SAP 用の WebSphere ビジネス・オブジェクトと、RFC Server 固有ビジネス・オブジェクト・ハンドラーとの間には、1 対 1 の関係があります。ビジネス・オブジェクト・ハンドラーのクラス・ファイルは、¥connectors¥SAP¥bapi¥server ディレクトリーに配置する必要があります。

重要: RFC Server 固有ビジネス・オブジェクト・ハンドラーがサーバーとして動作することを示すために、ビジネス・オブジェクト・ハンドラー名の前に値 server を組み込む必要があります。

例えば、BAPI_PO_CREATE RFC 対応機能をサポートしており、宛先コラボレーションの名前が POCollab である場合、動詞のアプリケーション固有情報は次のようになります。

```
AppSpecificInfo =bapi.server.Bapi_po_create;Collab=POCollab
```

属性の AppSpecificInfo

コネクターは、属性のアプリケーション固有情報の値を使用して、どのインポート・パラメーター、エクスポート・パラメーター、および表パラメーターを使用するかを決定します。このプロパティーの値には、プレフィックス I (インポート・パラメーターの場合) または E (エクスポート・パラメーターの場合) が含まれています。このプレフィックスは、属性値がデータを SAP アプリケーションに渡すために使われるのか、SAP アプリケーションからデータを渡すために使われるのかを示します。

構造体パラメーターはインポートとエクスポートの場合があるため、パラメーター値の前に I または E が使用されます。表パラメーターは、RFC 対応機能にデータ

を渡したり、RFC 対応機能からデータを戻したりすることができるため、I と E の両方のパラメーター値を持つことができます。

重要: I および E を使用してパラメーター値を指定するときには、区切り文字としてコロン (:) を必ず使用します。インポート値のみを指定する場合は、値の後にコロンを付ける必要があります。エクスポート値のみを指定する場合は、値の前にコロンを付ける必要があります。両方の値を指定する場合には、インポート値を先に、エクスポート値を後に指定し、両者をコロンで区切ります。

図 58 に、1 つのビジネス・オブジェクトとサンプル RFC 対応機能 (BAPI_EXAMPLE) の間のマッピングを示します。この例では、単純属性 (Attribute_1、Attribute_2、 および Attribute_3) は、インポート・パラメーターまたはエクスポート・パラメーターのみを指定しています。子ビジネス・オブジェクトを表す属性 (Child_1) は、エクスポート構造体パラメーターにマップされます。子ビジネス・オブジェクトの配列を表す属性 (Child_2) は、表パラメーターにマップされます。

各子ビジネス・オブジェクトには、対応する構造体またはテーブルのフィールドにマップされる単純属性 (それぞれ Attribute_11 と Attribute_14) があります。これらのフィールドは、BAPI の詳細を参照することで確認できます。

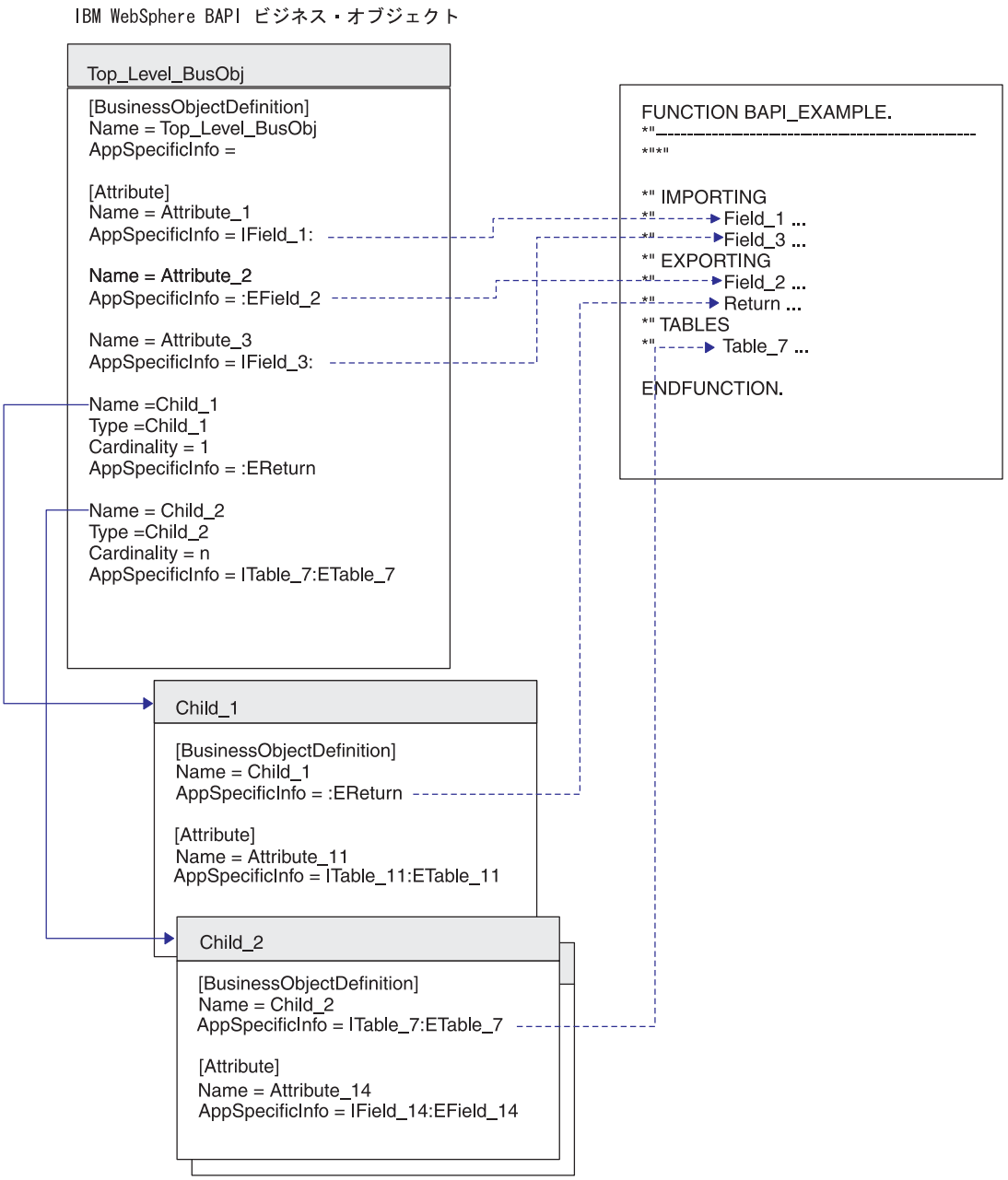


図 58. ビジネス・オブジェクトと BAPI 例とのマッピング

表 35 に、特定の種類の属性に対するアプリケーション固有情報の書式を示します。

表 35. 特定の種類の属性に対する AppSpecificInfo の書式

AppSpecificInfo の書式	**属性のタイプ**
I*ParameterName*:E*ParameterName*	単純
I*TableName*:E*TableName*	表パラメーターにマップされる、子ビジネス・オブジェクトを表します。
I*StructureName*:E*StructureName*	構造体パラメーターにマップされる、子ビジネス・オブジェクトを表します。
I*FieldName*:E*FieldName*	表パラメーターまたは構造体パラメーターのフィールドにマップされる、子ビジネス・オブジェクトの属性を表します。

SAPODA は、ビジネス・オブジェクト定義に対する適切なアプリケーション固有情報を自動的に生成します。生成されたアプリケーション固有情報のパラメーター名は、変更しないようにお勧めします。

生成したビジネス・オブジェクトおよびビジネス・オブジェクト・ハンドラーの使用

SAPODA は、サポートする必要のある各 RFC 対応機能に対して、RFC 対応機能固有ビジネス・オブジェクト定義と、RFC Server 固有ビジネス・オブジェクト・ハンドラーを生成するために使用します。生成されたファイルは、最小限の変更を加えるだけで利用できます。

必要な編集作業は、Server 動詞の動詞アプリケーション固有情報に宛先コラボレーションの名前を指定することのみです。

- 統合ブローカーが InterChange Server Express である場合は、コネクターにプッシュされるイベントに対してコラボレーションを明示的にサブスクライブすることはできないため、この情報が必要になります。したがって、RFC Server 固有オブジェクト・ハンドラーは、ビジネス・オブジェクトのメタデータから適切な宛先コラボレーションを判別した後で、そのコラボレーションのインスタンスを生成する必要があります。

重要: 使用している RFC 対応機能に単純フィールド属性が含まれておらず、SAPODA によって Dummy_key 属性がキー属性として作成されている場合は、この属性の値を変更しないでください。

ビジネス・オブジェクト定義とそれに対応する RFC Server 固有ビジネス・オブジェクト・ハンドラーが生成されたら、ビジネス・オブジェクト定義を、WebSphere Business Integration システムの実行時環境に追加する必要があります。

- Business Object Designer Express を使用して、ビジネス・オブジェクト定義をリポジトリーにロードします。

  注: `repos_copy` コマンドを使用して、定義をリポジトリーにロードすることもできます。

- システム・コマンドを使用して、RFC Server 固有ビジネス・オブジェクト・ハンドラーのファイルを、製品ディレクトリーの下にある次のディレクトリーにコピーします。

  ```
  ¥connectors¥SAP¥bapi¥server
  ```

RFC Server 固有ビジネス・オブジェクト・ハンドラーのファイルは、以下のとおりです。

- *RFC-EnabledFunctionName*`.java`
- *RFC-EnabledFunctionName*`.class`

例えば、`BAPI_PO_CREATE` RFC 対応機能とユーザー指定プレフィックス `sap_` が指定されると、SAPODA は以下のオブジェクトおよびファイルを生成します。

- `sap_bapi_po_create` (すべての子ビジネス・オブジェクトを含むビジネス・オブジェクト定義)
- `Bapi_po_create.java`

- Bapi_po_create.class

重要: 生成されたビジネス・オブジェクトの名前や、その子ビジネス・オブジェクトの名前を変更することができます。それには、定義を Business Object Designer Express によってではなく、テキスト・ファイルとして編集する必要があります。ビジネス・オブジェクトの名前を変更する場合は、変更する名前に対するすべての参照も必ず変更してください。また、ビジネス・オブジェクト・ハンドラーに対して生成された .class ファイルの名前を変更する場合は、関連するビジネス・オブジェクトの Server 動詞アプリケーション固有情報についても、同じように変更する必要があります。

注: 開発ネーム・スペースで開発された RFC 対応 ABAP 機能や BAPI の場合、SAPODA は、ビジネス・オブジェクト定義、.java ファイル、および .class ファイルに名前を付ける際に、機能名の「/」文字を削除するか、または「_」で置き換えます。SAPODA は、名前の先頭の文字である場合に限り、「/」文字を削除します。定義名やファイル名に「/」文字が含まれていなくても、コードは指定した機能を、「/」文字を含んだ正しい名前で正確に呼び出します。また、機能名が数字で開始される場合、SAPODA はストリング Rfm_ を名前の前に付加します。

ヒントとテクニック

ビジネス・オブジェクトや RFC Server 固有ビジネス・オブジェクト・ハンドラーを開発する際に役立つ、いくつかのヒントとテクニックを以下に示します。

- 『複数のビジネス・オブジェクトに同じリターン・ビジネス・オブジェクトが含まれる場合』
- 178 ページの『生成したビジネス・オブジェクト定義に不要な属性や子ビジネス・オブジェクトが含まれる場合』
- 178 ページの『生成したビジネス・オブジェクト名が長すぎて、命名規則に障害が起こる場合』
- 179 ページの『表パラメーターについて生成された AppSpecificInfo で不要なパラメーターが指定されている場合』

複数のビジネス・オブジェクトに同じリターン・ビジネス・オブジェクトが含まれる場合

ほとんどの RFC 対応機能では、リターン・オブジェクトに同じ名前を使用します。SAPODA は、ビジネス・オブジェクト定義を生成する際に、このリターン・オブジェクトを表す子ビジネス・オブジェクトを作成します。複数のビジネス・オブジェクト定義に同じ名前の子ビジネス・オブジェクトが含まれている場合は、その子ビジネス・オブジェクトの定義をリポジトリーに追加できるのは 1 回のみです。

複数のビジネス・オブジェクトにリターン・ビジネス・オブジェクトを含めることができるようにするには、リターン・ビジネス・オブジェクトの名前を各ビジネス・オブジェクトに固有になるように変更する必要があります。

リターン・ビジネス・オブジェクトの名前を変更するには、それを含んでいる各ビジネス・オブジェクト定義で、以下の手順を実行してその定義を変更します。子ビジネス・オブジェクトの定義は、その親と同じ定義ファイル内に含まれています。

子の名前を変更するには、以下の手順を行います。

1. テキスト・エディターで、トップレベル・ビジネス・オブジェクトの定義ファイルを開きます。
2. BOprefix_return 子ビジネス・オブジェクトの定義を見つけます。
3. 子の名前を固有な名前に変更します。例えば、テキストに番号を付加します (sap_return_2)。
4. 定義内のすべての参照を、その子を新しい名前で参照するように変更します。例えば、子ビジネス・オブジェクトを表すすべての属性の Type プロパティーの値を変更します。
5. 変更した定義ファイルを保存します。
6. Business Object Designer Express を使用して、新しく名前を付けた子ビジネス・オブジェクトをリポジトリーにロードします。

   注: repos_copy コマンドを使用して、定義をリポジトリーにロードすることもできます。

生成したビジネス・オブジェクト定義に不要な属性や子ビジネス・オブジェクトが含まれる場合

SAPODA は、すべての RFC 対応機能インターフェース・パラメーターを解釈し、それぞれに対応する WebSphere ビジネス・オブジェクト属性または子ビジネス・オブジェクトを作成します。ビジネス・オブジェクト処理のパフォーマンスを向上させるために、すべての不要な属性やビジネス・オブジェクトをビジネス・オブジェクト定義から削除する必要があります。

注: SAPODA では、定義を生成する前に、すべてのオプションの属性および子ビジネス・オブジェクトをグラフィカルに削除できます。詳細については、48 ページの『追加情報の指定』を参照してください。

ビジネス・オブジェクト処理のパフォーマンスを向上させるために、アプリケーション固有情報から、すべての不要なインポートおよびエクスポート表パラメーター値を削除することもできます。

その他の変更が必要な場合は、定義が生成された後、Business Object Designer Express を使用して手動でビジネス・オブジェクト定義を編集できます。ただし、使用されないことが確実な属性のみを削除するように十分注意してください。

生成したビジネス・オブジェクト名が長すぎて、命名規則に障害が起こる場合

SAPODA は、RFC 対応機能モジュールの名前を使用して、生成されたビジネス・オブジェクトに名前を付けます。ビジネス・オブジェクトの名前を変更するには、テキスト・エディターを使用します。

重要: 名前を変更する場合は、その名前に対するすべての参照も必ず変更してください。ただし、生成されたアプリケーション固有情報のパラメーター名は変更しないでください。

生成されたビジネス・オブジェクトの名前を変更するには、以下の手順を行います。

1. 定義をファイルに保管します。
2. テキスト・エディターを使用して、名前を短縮または変更します。
3. Business Object Designer Express を使用して、新しく名前を付けた子ビジネス・オブジェクトをリポジトリーにコピーします。

   注: `repos_copy` コマンドを使用して、定義をリポジトリーにロードすることもできます。

表パラメーターについて生成された AppSpecificInfo で不要なパラメーターが指定されている場合

表パラメーターは、インポート・パラメーターとエクスポート・パラメーターの両方になることができます。表パラメーターの値のインポートまたはエクスポートが不要な場合は、アプリケーション固有情報から削除できます。

例えば、作成操作の場合、作成操作が完了した後に SAP アプリケーションから表データを戻す必要がないときには、エクスポート・パラメーター値 (例えば E*table name*) を削除できます。

検索操作の場合は、インポート表パラメーターを指定する必要はありません。したがって、インポート・パラメーター値 (例えば I*table name*) を削除できます。

注: 親ビジネス・オブジェクト内で子を表す属性の `AppSpecificInfo` や、子ビジネス・オブジェクトのビジネス・オブジェクト・レベルにある `AppSpecificInfo` からは、不要な値を削除する必要があります。コロン (`:`) を削除しないでください。

例えば、図 58 の `ETable_7` エクスポート・パラメーターを削除するには、以下の手順を行います。

1. Top_Level_BusObj ビジネス・オブジェクトの Child_2 属性で、属性の AppSpecificInfo 値を次のように変更します。

   ```
   ITable_7:
   ```

2. Child_2 ビジネス・オブジェクトのビジネス・オブジェクト・レベルにある AppSpecificInfo で、値を次のように変更します。

   ```
   ITable_7:
   ```

3. 子ビジネス・オブジェクトの各属性の AppSpecificInfo で、例として Attribute_14 を使用するとすれば、値を次のように変更します。

   ```
   IField_14:
   ```

第 5 部 Hierarchical Dynamic Retrieve Module

第 15 章 Hierarchical Dynamic Retrieve Module の概要

この章では、Hierarchical Dynamic Retrieve Module について説明します。Hierarchical Dynamic Retrieve Module は、階層型またはフラットなビジネス・オブジェクトを処理します。これらの要求を処理するために、コネクターはデータを SAP アプリケーションから検索します。

この章の内容は以下のとおりです。

Hierarchical Dynamic Retrieve Module のコンポーネント

Hierarchical Dynamic Retrieve Module は Java で記述されており、Vision コネクター・フレームワークを拡張します。このモジュールはそれ自身のアプリケーション固有のコンポーネントを持たないため、BAPI 用のアプリケーション固有のコンポーネントを使用します。そのため、このモジュールはコネクター・フレームワーク、BAPI 用のアプリケーション固有のコンポーネント、DynRetBOH ビジネス・オブジェクト・ハンドラー、および SAP RFC ライブラリーから構成されています。 SAP では、Java および C で記述された RFC ライブラリーを提供しています。コネクターは、Java アーカイブ (JAR) ファイルとして提供され、実行されます。

184 ページの図 59 は、Hierarchical Dynamic Retrieve Module のアーキテクチャーを示すものです。

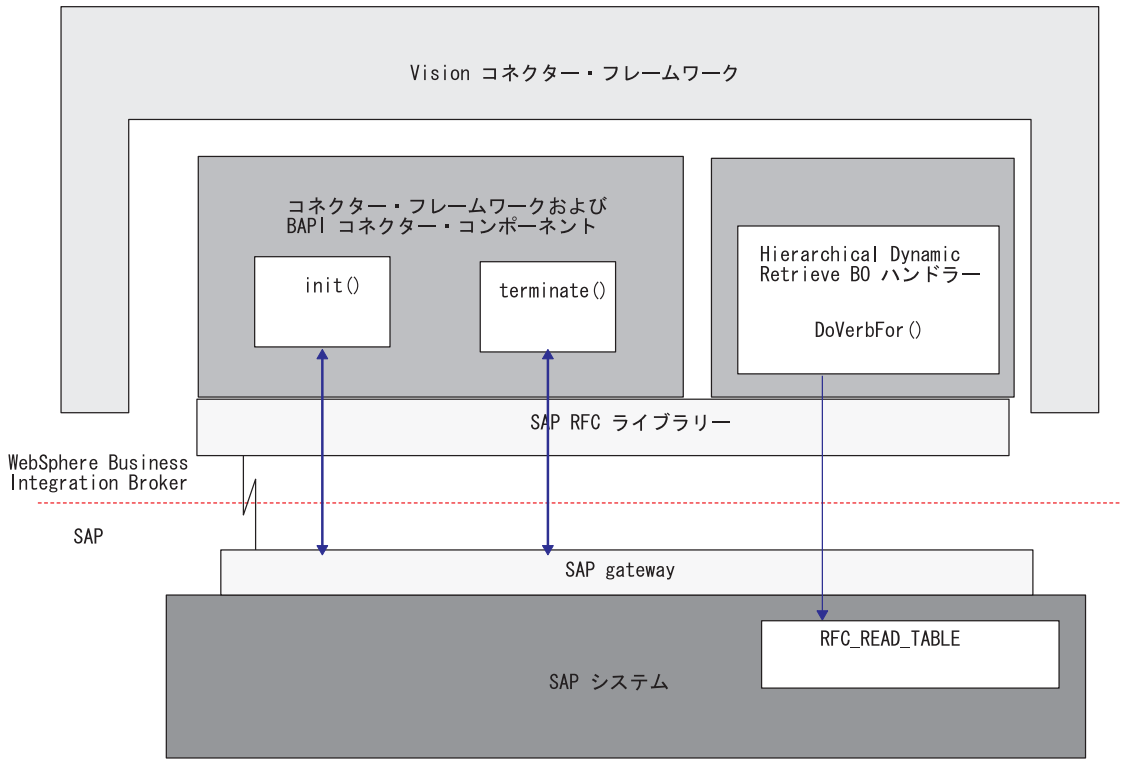


図 59. Hierarchical Dynamic Retrieve Module のアーキテクチャー

コネクターの動作方法

コネクターは、ビジネス・オブジェクトの処理情報を、コネクター内にハードコーディングされた情報からではなくビジネス・オブジェクト内で指定されたメタデータから取得します。コネクターは、ビジネス・オブジェクトから処理情報を取得するために、以下の情報を想定します。

- 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
- ビジネス・オブジェクトの可能なデータベース表現

詳細については、 xxii ページの『ビジネス・オブジェクトの処理』、および 189 ページの『第 17 章 Hierarchical Dynamic Retrieve Module のビジネス・オブジェクトの開発』を参照してください。

コネクターは、統合ブローカーからアプリケーション操作の実行要求を受け取ると、トップレベル・ビジネス・オブジェクトで指定されている動詞から処理情報を取得します。

コネクターは階層型ビジネス・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、各子ビジネス・オブジェクトに対して同じステップを実行します。

注: **階層型**ビジネス・オブジェクトという用語は、その任意のレベルに格納されているすべての子ビジネス・オブジェクトを含めた、ビジネス・オブジェクトの全体のことを表します。**個別**ビジネス・オブジェクトという用語は、それが格納している、あるいはそれが格納されている子ビジネス・オブジェクトにはかかわりなく、単一のビジネス・オブジェクトのことを表します。**トップレベル**・ビジネス・オブジェクトという用語は、階層のトップレベルにあって、それ自身は親ビジネス・オブジェクトを持たない個別ビジネス・オブジェクトのことを表します。

統合ブローカーが Retrieve 動詞を使用する階層型ビジネス・オブジェクトを送信すると、コネクターは、そのビジネス・オブジェクトの現在のデータベース表現に正確に一致するビジネス・オブジェクトを統合ブローカーに返送します。つまり、コネクターが戻すすべての個別ビジネス・オブジェクトの各単純属性の値は、データベース内でそれに対応するフィールドの値に一致します。また、戻されるビジネス・オブジェクトの各配列内の個別ビジネス・オブジェクトの数は、その配列に対応するデータベース内の子の数に一致します (アプリケーション固有の情報で子が一部に制限されている場合を除く)。

そのような検索を実行する場合、コネクターはトップレベル・ビジネス・オブジェクトの基本キーの値を使用して、データベース内の対応するデータを減少させて再帰的に処理を行います。

第 16 章 Hierarchical Dynamic Retrieve Module の構成

この章では、Adapter for mySAP.com の Hierarchical Dynamic Retrieve Module の構成について説明します。この章で説明する構成作業を行う前に、SAP コネクターをインストールしておく必要があります。

この章の内容は以下のとおりです。

- 『Hierarchical Dynamic Retrieve Module のディレクトリーとファイル』
- 『Hierarchical Dynamic Retrieve Module の構成プロパティー』

Hierarchical Dynamic Retrieve Module のディレクトリーとファイル

表 36 に、Hierarchical Dynamic Retrieve Module で使用するディレクトリーとファイルのリストがあります。

表 36. Hierarchical Dynamic Retrieve Module のディレクトリーとファイル

ファイル名	説明
CWSAP.jar	コネクターのクラス・ファイル。

注: 本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。Linux のインストールの場合には、円記号 (¥) をスラッシュ (/) に置き換えてください。ファイルのパス名はすべて、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

Hierarchical Dynamic Retrieve Module の構成プロパティー

Hierarchical Dynamic Retrieve Module を実行する前に、標準構成プロパティーおよびコネクター固有の構成プロパティーを設定する必要があります。少なくとも、Modules プロパティーに BAPI Module の名前を追加する必要があります。 BAPI Module の名前は Bapi です。

Hierarchical Dynamic Retrieve Module はサービス呼び出し要求を実行するのみであるため、ビジネス・オブジェクト・ハンドラーは、送信されるビジネス・オブジェクト内のメタデータを通して呼び出されます。ただし、Modules の値を Bapi に設定すると、コネクター・スレッドが設定され、これによってコネクターの初期化と終了を行うことができます。この設定では、Hierchical Dynamic Retrieve Module の処理中に問題が起きた場合、実行中のコネクター・スレッドの terminate() メソッドを呼び出すことにより、コネクターを簡単にシャットダウンすることができます。

コネクター構成プロパティーの構成の詳細については、13 ページの『第 2 章 コネクターの構成』、 341 ページの『付録 C. コネクター固有の構成プロパティー』、および 315 ページの『付録 B. 標準構成プロパティー』 を参照してください。

第 17 章 Hierarchical Dynamic Retrieve Module のビジネス・オブジェクトの開発

この章では、Hierarchical Dynamic Retrieve Module がビジネス・オブジェクトを処理する方法と、コネクターがデータを検索する際の前提事項について説明します。この情報は、既存のビジネス・オブジェクトを変更する際のガイドとして、あるいは新しいビジネス・オブジェクトを実装する際の提案として利用できます。

この章では、ビジネス・オブジェクトとその処理に関する背景情報を記載するほか、SAPODA (Object Discovery Agent の 1 つ) を使用して Hierarchical Dynamic Retrieve Module 用のビジネス・オブジェクトを開発する方法について説明します。SAPODA は、ユーザーがグラフィカル・インターフェースを使用して指定したテーブルからビジネス・オブジェクト定義を生成します。このユーティリティーは、階層型ビジネス・オブジェクト定義を作成するときよりも、個別ビジネス・オブジェクト定義を作成するときに非常に役立ちます。

Hierarchical Dynamic Retrieve Module の詳細については、 183 ページの『第 15 章 Hierarchical Dynamic Retrieve Module の概要』を参照してください。

この章の内容は以下のとおりです。

ビジネス・オブジェクト開発ユーティリティー

Hierarchical Dynamic Retrieve Module 用のビジネス・オブジェクトを開発するには、コネクターで処理する必要のあるオブジェクトの各タイプについて、アプリケーション固有ビジネス・オブジェクト定義を作成する必要があります。Adapter for mySAP.com には以下のコンポーネントがあります。

- vDynRetBOH ビジネス・オブジェクト・ハンドラー。これは、コネクターがアプリケーションでデータを検索する際に使用します。
- SAPODA

Business Object Designer Express またはテキスト・エディターを使用して、コネクター用のビジネス・オブジェクト定義を作成することもできますが、SAPODA では SAP アプリケーションのネイティブ定義をテンプレートとして使用するため、最初は SAPODA を使用することをお勧めします。

ビジネス・オブジェクト名

SAPODA は、ビジネス・オブジェクト定義内のすべての属性名が固有であることを保証します。フィールド名と記述を付加することによって、SAP のデータ・ディクショナリーから名前を派生させます。SAP テーブルから属性を命名する際に、変更された属性名が次の条件に該当すると、SAPODA はストリングを属性名の前に付加します。

- 数字で開始される場合は、A_ を前に付加します。
- アンダースコアー文字 (_) で開始される場合は、A を前に付加します。

重要: 属性名は、ビジネス・オブジェクトを生成した後で、いつでも変更できます。ビジネス・オブジェクトの名前や属性名を変更しても、ビジネス・オブジェクトの処理には影響を与えません。ただし、アプリケーション固有の情報では属性に対応する SAP テーブルおよび列が識別されるため、アプリケーション固有の情報を変更すると、ビジネス・オブジェクトの処理に影響を**与えます**。

アプリケーション固有の情報の詳細については、 199 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』を参照してください。

ビジネス・オブジェクト構造

コネクターは、すべての個別ビジネス・オブジェクトが 1 つ以上のデータベース表によって表現されており、かつ、そのビジネス・オブジェクト内の各**単純属性** (String または Integer または Date などの単一値を表す属性) が、データベース表のうち 1 つの中にある列によって表現されていることを想定しています。以下の状態は有効です。

- データベース表に、対応する個別ビジネス・オブジェクトが持つ単純属性よりも多くの列が存在する (つまり、データベース内の一部の列がビジネス・オブジェクト内で表現されていない) 場合。設計には、ビジネス・オブジェクト処理に必要な列のみを含めるようにしてください。
- 個別ビジネス・オブジェクトに、対応するデータベース表内の列よりも多くの単純属性が存在する (つまり、ビジネス・オブジェクト内の一部の属性がデータベース内で表現されていない) 場合。データベース内で表現されていない属性は、アプリケーション固有情報を持ちません。
- SAP API の制限のために、1 つのビジネス・オブジェクトで表現される各テーブルで必要とされるすべての列のバイトの合計数は 512 バイトを超えることができません。詳細については、 194 ページの『長いデータ行の取り扱い』を参照してください。
- SAP API の制限のため、ランタイム HDR Modules は一部の非キャラクター・ベースのデータ型を解析できません。82 ページの『Hierarchical Dynamic Retrieve Module のトラブルシューティング』を参照してください。

SAP 用の WebSphere ビジネス・オブジェクトは、フラット (階層なし) または階層型にすることができます。**フラット**なビジネス・オブジェクトのすべてとの属性は単純属性で、単一の値を表します。

階層型ビジネス・オブジェクトは、1 つの子ビジネス・オブジェクト、子ビジネス・オブジェクトの配列、または両者の組み合わせを表す属性を持ちます。さら

に、それぞれの子ビジネス・オブジェクトも、1 つの子ビジネス・オブジェクトやビジネス・オブジェクトの配列を格納できます。これらの子についても同様です。

ビジネス・オブジェクトの関係

子または配列を表す属性の Cardinality プロパティーは、親と子の関係のタイプを次のように決定します。

- **単一カーディナリティー関係**は、親ビジネス・オブジェクトの属性が、カーディナリティー 1 の子ビジネス・オブジェクトを表す場合に生じます。
- **複数カーディナリティー関係**は、親ビジネス・オブジェクトの属性が、カーディナリティー n の子ビジネス・オブジェクトの配列を表す場合に生じます。

コネクターが単一カーディナリティー関係を処理する方法と、複数カーディナリティー関係を処理する方法には、違いはありません。ただし、データベース表が単一カーディナリティー関係を持つ場合と、複数カーディナリティー関係を持つ場合とでは、外部キー関係に構造的な相違があります。この相違は、次の理由で重要です。

- 単一カーディナリティー関係の場合、外部キーは、親の**非キー**属性を外部キーとして参照する子の基本キーによって決定されます。それぞれの子は、親の非基本キー属性を外部キーとして参照する単純属性を少なくとも 1 つ持ちます。192 ページの図 60 に例を示します。
- 複数カーディナリティー関係の場合、外部キーは、親の**基本キー**属性を参照する子の基本キーによって決定されます。それぞれの子は、親の基本キーを外部キーとして格納する単純属性を少なくとも 1 つ持ちます。子は、親が持つ基本キー属性と同じ数の外部キー属性を持ちます。193 ページの図 62 に例を示します。

どちらの場合にも、親ビジネス・オブジェクトと子ビジネス・オブジェクトの外部キー関係は、子ビジネス・オブジェクトのキー属性のアプリケーション固有の情報によって指定されます。詳細については、197 ページの『ビジネス・オブジェクト属性のプロパティー』および 200 ページの『単純属性のアプリケーション固有の情報』を参照してください。

単一カーディナリティー関係の例

192 ページの図 60 に、SAP でカスタマー・オブジェクトを処理するために開発された、単純な WebSphere ビジネス・オブジェクトの例を示します。このサンプルの SAP_Customer は、それが格納するサンプルのアドレス・オブジェクトに対して、単一カーディナリティー関係を持ちます (`addr_data[1]` 属性がカーディナリティー 1 を持ちます)。子ビジネス・オブジェクトの基本キー属性 (`address_id`) は、親ビジネス・オブジェクトの非基本キー (`address_id`) を参照します。

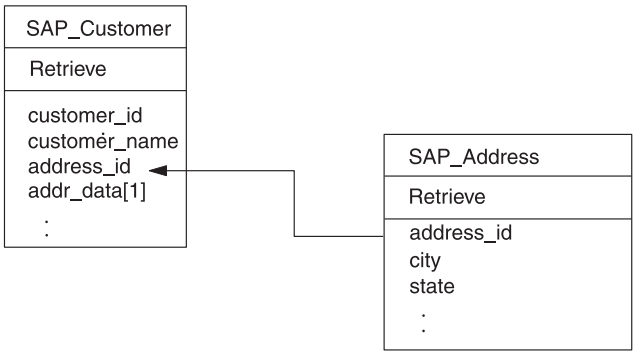


図 60. カスタマーとアドレスの関係の例

次に示す SELECT ステートメントとそれらの出力は、上に示したビジネス・オブジェクトによって表されるテーブルでのデータ検索を示したものです。

```
SELECT * FROM KNA1

KUNNR          NAME1          ADRNR
-----          -------------  ----------
10254          JOE'S PIZZA          2208
10255          LARRY'S HARDWARE     2209

SELECT * FROM ADRC

ADDRNUMBER          CITY1          REGION
----------          ----           -----
2208                BURLINGAME     CA
2209                SAN FRANCISCO  CA
```

上の例では、各カスタマー (Joe's Pizza および Larry's Hardware) は単一アドレスを持ちます。KUNNR 列および ADDRNUMBER 列がそれぞれのテーブルに対する基本キー制約として定義されている場合には、上に示した構造によって、各カスタマーは関連するアドレスを 1 つのみ持つことができるようになります。

注: 簡潔にするために、本書に掲載されている図には、SAP アプリケーションのデータベース内のテーブルやフィールドを決定するためにコネクターが使用するアプリケーション固有の情報は表示されていません。

複数カーディナリティー関係の例

193 ページの図 61 に、複数カーディナリティー関係を示します。この例では、ID=ABC は親の基本キーを持つ単純属性で、child[n] は子ビジネス・オブジェクトの配列を表す属性です。

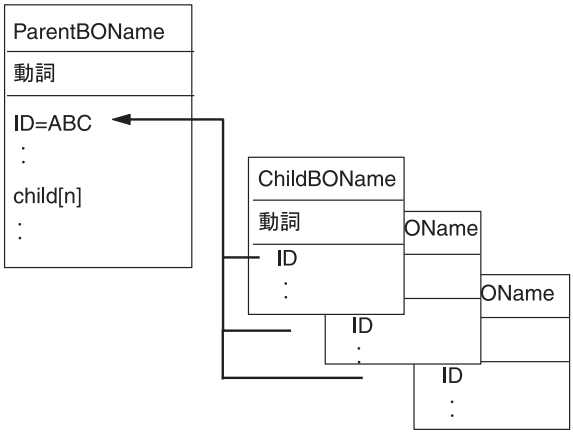


図 61. 複数カーディナリティーのビジネス・オブジェクト関係

図 62 に、SAP でカスタマー・オブジェクトを処理するために開発された、別の WebSphere ビジネス・オブジェクトの例を示します。このサンプルの SAP_Customer は、それが格納するサンプルの販売ビュー・オブジェクトに対して、複数カーディナリティー関係を持ちます (`sales_view_data[n]` 属性がカーディナリティー `n` を持ちます)。子ビジネス・オブジェクトの基本キー属性 (`customer_id`) は、親ビジネス・オブジェクトの基本キー (`customer_id`) を参照します。

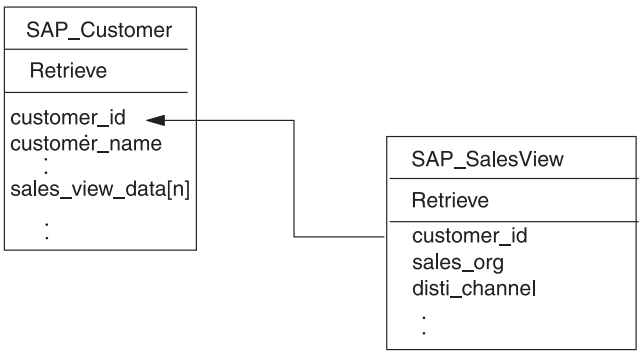


図 62. カスタマーと販売ビュー関係の例

次に示す `SELECT` ステートメントとそれらの出力は、それぞれのテーブルでのデータ検索を示したものです。

```
SELECT * FROM KNA1

KUNNR          NAME1
-----          -------------
10254          JOE'S PIZZA
10255          LARRY'S HARDWARE

SELECT * FROM KNVV

KUNNR          VKORG          VTWEG          SPART
----------     -----          -----          -----
10254          EURP           01             12
10255          EURP           01             09
10255          USA            01             13
10255          USA            01             14
```

この例では、Joe's Pizza に関連付けられた販売ビュー・レコードが 1 件あるのに対して、Larry's Hardware に関連付けられた販売ビュー・レコードは 3 件あります。上に示した構造により、各カスタマーは関連付けられた販売ビュー・レコードを 0 件以上持つことができます。

長いデータ行の取り扱い

SAP の RFC_READ_TABLE 機能では、データ検索は 1 データ行当たり 512 バイトに制限されています。多くの SAP テーブルには、1 行当たり 512 バイトを超えるデータが格納されています。しかし、ほとんどのビジネス・オブジェクトは、すべてのデータベース・フィールドの小さな一部を表しています。そのため、ビジネス・オブジェクト内のすべての属性の合計長が、最大値の 512 バイトを超えることはほとんどありません。

コネクターが 1 つのデータベース表で 512 バイトを超えるデータを検索しなければならない場合、追加のフィールドは、別個の単一カーディナリティー子ビジネス・オブジェクトとして表される必要があります。例えば、あるビジネス・オブジェクトが 1 つのテーブルの 1500 バイトのデータを表す必要がある場合、トップレベル・ビジネス・オブジェクトには、少なくとも 2 つの単一カーディナリティー子ビジネス・オブジェクトが含まれています。親にも、またどちらの子にも、全長 (つまり属性の最大長の合計) が 512 バイトを超える属性は含まれていません。

注: ビジネス・オブジェクトが複数のデータベース表を表す場合、各テーブルを表す属性内の値の合計長は 512 バイトを超えることはできません。しかし、この制限はすべての属性の値の合計長には適用されません。例えば、あるビジネス・オブジェクトが Customers および CustomerPartners についての情報を格納しているテーブルのデータを表す場合、Customers を表す属性の値は 512 バイトを超えることができず、CustomerPartners を表す属性の値も 512 バイトを超えることはできませんが、これらの属性を結合した値は 512 バイトを超えることができます。

ビジネス・オブジェクト動詞の処理

このセクションでは、Retrieve 動詞を使用するビジネス・オブジェクト要求を処理するためにコネクターが実行するステップの概要を説明します。コネクターは階層型ビジネス・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、各子ビジネス・オブジェクトに対して同じステップを実行します。

ビジネス・オブジェクトの比較

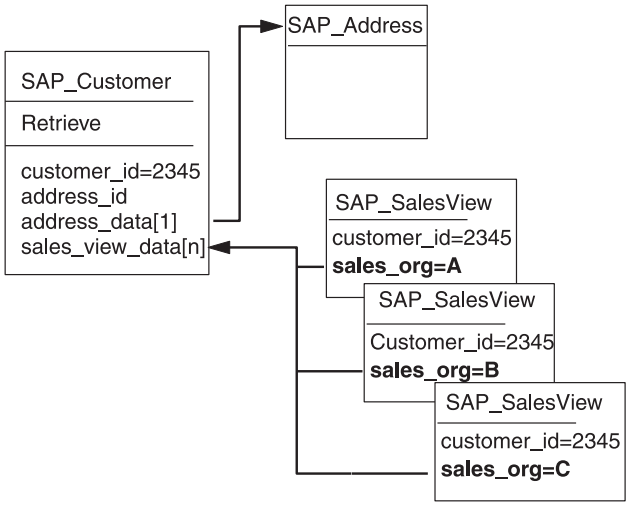
統合ブローカーからの検索要求を処理する場合、コネクターは、そのオブジェクトの現在のデータベース表現に一致するビジネス・オブジェクトの返送を試行します。詳細については、次のようになります。

- 統合ブローカーに戻されるすべての個別ビジネス・オブジェクト内の各単純属性の値は、データベース内でそれに対応するフィールドの値に一致します。
- 戻されるビジネス・オブジェクトの各配列内の個別ビジネス・オブジェクトの数は、データベース内でそれに対応する子の数に一致します。

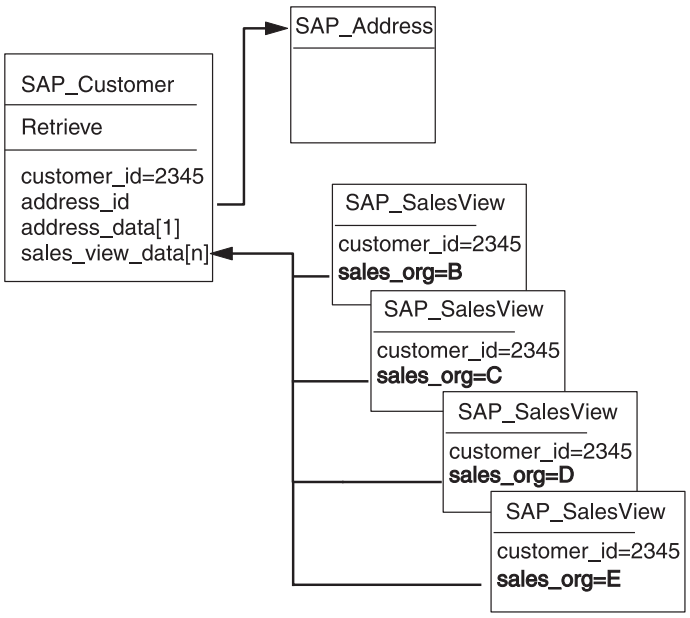
したがって、Hierarchical Dynamic Retrieve Module が Retrieve 動詞を使用するビジネス・オブジェクト要求を受け取ると、アプリケーションのオブジェクト全体を再

帰的に降順にたどり、現在のデータベース表現を検索することによって、応答ビジネス・オブジェクトを作成します。検索を行うために、コネクターは要求の際のトップレベル・ビジネス・オブジェクトで指定されたキー値を使用します。そのため、そのトップレベル親のすべての子を含んだ応答ビジネス・オブジェクトは、要求ビジネス・オブジェクトとは異なる単純属性の値や、異なる子ビジネス・オブジェクトを持つ場合があります。

例えば、統合ブローカーが次に示すような SAP_Customer ビジネス・オブジェクトを Hierarchical Dynamic Retrieve Module に渡したとします。



現在のデータベース表現で、SAP_Customer 2345 に格納されている SAP_SalesView 子ビジネス・オブジェクトの配列に sales_org A が含まれていない場合、コネクターの応答ビジネス・オブジェクトには、この子は含まれません。また、SAP_Customer 2345 の現在のデータベース表現に sales_org D および sales_org E が含まれている場合、コネクターはこれらの子を応答ビジネス・オブジェクトに含めます。SAP Hierarchical Dynamic Retrieve Module が Retrieve 処理の最後に統合ブローカーに戻すビジネス・オブジェクトは、次のようになります。



注: コネクターが、特定の応答ビジネス・オブジェクトを作成する際に、複数のテーブルから読み取りを行った場合、そのビジネス・オブジェクトは単一のデータベース・オブジェクトには一致しません。その代わりに、指定したテーブルの選択したフィールドに一致します。

検索操作

ビジネス・オブジェクトを検索すると、コネクターは、操作が成功した場合には (操作によってビジネス・オブジェクトが変更されたかどうかには関係なく) VALCHANGE という状況を、操作が失敗した場合には FAIL という状況を戻します。

コネクターは、階層型ビジネス・オブジェクトを検索する際に、以下のステップを実行します。

1. 統合ブローカーから受け取ったトップレベル・ビジネス・オブジェクトから、すべての子ビジネス・オブジェクトを除去します。
2. RFC_READ_TABLE 関数を呼び出して、データベースで**トップレベル・ビジネス・オブジェクト**を検索します。

   コネクターは、要求ビジネス・オブジェクト内のキー値を使用して、SELECT ステートメントの WHERE 文節を作成します。検索の結果、次のいずれかのアクションが実行されます。

   - SELECT ステートメントが 1 件のレコードを戻した場合、コネクターは子の処理を継続し、VALCHANGE を (属性値が変更されたかどうかには関係なく) 戻します。
   - SELECT ステートメントがレコードを戻さず、データベースにトップレベル・ビジネス・オブジェクトが存在しないことを通知した場合、コネクターは BO_DOES_NOT_EXIST を戻します。
   - SELECT ステートメントが複数のレコードを戻した場合、コネクターは子の処理を継続し、VALCHANGE を戻します。

3. すべての**子ビジネス・オブジェクト** (単一カーディナリティーまたは複数カーディナリティー) を再帰的に検索します。

   コネクターは RFC_READ_TABLE 機能を呼び出します。この機能は、適切な外部キー値を使用して SELECT ステートメントの WHERE 文節を作成します。コネクターは、Required としてマークされた属性を、次の方法で処理します。

   - ビジネス・オブジェクトの定義で子が必須であることが指定されている場合には、検索はレコードを戻す必要があります。検索がレコードを戻さない場合、コネクターは FAIL を戻します。
   - 子が必須でなく、かつ、検索がレコードを戻さず、アプリケーションに子が存在しないことを通知した場合、コネクターは親の属性を空のままにします。

   レコードが戻されるたびに、コネクターは以下のアクションを実行します。

   a. 正しいタイプの新しい個別ビジネス・オブジェクトを作成します。
   b. 現在のビジネス・オブジェクトのすべての属性を、戻された行内の値に基づいて設定します。
   c. 現在のビジネス・オブジェクトのすべての子を、再帰的に検索します。

      重要: 単一カーディナリティーの子の検索で複数のレコードが戻された場合、コネクターは最初のレコードのみを戻します。
   d. 現在のビジネス・オブジェクトとそのすべての子を、親の適切な単一カーディナリティー属性または配列属性に挿入します。

注: ビジネス・オブジェクトには、データベース列に対応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。検索の際には、コネクターはトップレベル・ビジネス・オブジェクトのそのような属性については変更せず、統合ブローカーから受け取ったときの値に設定されたままにします。これらの属性のアプリケーション固有の情報は、ブランクにしておく必要があります。

ビジネス・オブジェクト属性のプロパティー

ビジネス・オブジェクト・アーキテクチャーでは、属性に適用されるさまざまなプロパティーを定義しています。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これらを設定する方法について説明します。

Name プロパティー

各ビジネス・オブジェクト属性は、固有の名前を持つ必要があります。

Type プロパティー

各ビジネス・オブジェクト属性は、String 型か、または子ビジネス・オブジェクトあるいは子ビジネス・オブジェクトの配列のタイプである必要があります。

Cardinality プロパティー

各ビジネス・オブジェクト属性は、このプロパティーの値として 1 または n を持ちます。子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す

すべての属性は、ContainedObjectVersion プロパティー (子のバージョン番号を指定します) および Relationship プロパティー (値 Containment を指定します) も持ちます。

Max length プロパティー

コネクターでは、このプロパティーは使用されません。

Key プロパティー

各ビジネス・オブジェクトで、少なくとも 1 つの単純属性がキーとして指定される必要があります。ある属性をキーとして定義するには、このプロパティーを true に設定します。

重要: 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に指定することはサポートしていません。

ある単純属性の Key プロパティーが true に設定されている場合、コネクターはその属性を、ビジネス・オブジェクトの処理中に生成する SELECT SQL ステートメントの WHERE 文節に追加します。

パフォーマンスを最大化するために、可能な限り多くのキー・フィールドにデータを指定することをお勧めします。

ビジネス・オブジェクトの配列で 1 つまたは複数の子ビジネス・オブジェクトを検索するために、コネクターは SELECT ステートメントの WHERE 文節にある外部キーを使用します。子ビジネス・オブジェクトの属性の Key プロパティーは使用しません。子ビジネス・オブジェクトの属性を外部キーとして指定する方法については、200 ページの『単純属性のアプリケーション固有の情報』を参照してください。

Foreign key プロパティー

コネクターでは、このプロパティーは使用されません。コネクターは外部キー情報を、アプリケーション固有の情報から取得します。詳細については、200 ページの『単純属性のアプリケーション固有の情報』を参照してください。

Required プロパティー

Required プロパティーは、属性が値を含んでいる必要があるかどうかを指定します。

- 子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す属性が Required としてマーク付けされていて、コネクターがアプリケーションで子の検索に失敗すると、検索操作は失敗します。
- 単純属性が Required としてマーク付けされていて、コネクターがデータベースで対応する行の検索に失敗すると、検索操作は失敗します。例えば、コネクターが、あるビジネス・オブジェクトに関する複数のテーブルから読み取りを行い、これらのテーブルの 1 つに含まれている値を表す Required の単純属性に対応する行の検索に失敗すると、検索全体が失敗します。

AppSpecificInfo

このプロパティーの詳細については、 200 ページの『単純属性のアプリケーション固有の情報』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、コネクターが SELECT ステートメントの WHERE 文節を生成するときに使用するデフォルト値を指定します。このプロパティーは、キーとして指定されている単純属性のみに関係します。例えば、コネクターが Language 属性に対して指定されたデフォルト値を使用するには、Language 属性をキーとして指定する必要があります。

単純属性の特殊値

ビジネス・オブジェクトの単純属性は、CxIgnore という特殊値を持つことができます。コネクターは統合ブローカーからビジネス・オブジェクトを受け取ると、CxIgnore という値を持つすべての属性を無視します。これは、これらの属性がコネクターに対して不可視になったのと同様です。

コネクターがデータベースでデータを検索し、SELECT ステートメントが属性についてブランクの値を戻した場合、コネクターはこの属性の値をデフォルトで CxBlank に設定します。

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 つのキー属性を持つことを要求するため、コネクターに渡されるビジネス・オブジェクトが、CxIgnore に設定されない基本キーまたは外部キーを少なくとも 1 つ持つようにする必要があります。

ビジネス・オブジェクトのアプリケーション固有情報

ビジネス・オブジェクト定義のアプリケーション固有の情報は、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に依存した指示をコネクターに提供します。この情報には、以下の内容が含まれます。

- vDynRetBOH ビジネス・オブジェクト・ハンドラーのクラス。これは、トップレベル・ビジネス・オブジェクトの動詞のアプリケーション固有の情報で提供されます。この値は、このモジュールが処理するすべてのビジネス・オブジェクトについて同一です。
- データベースおよび照会情報。これは、単純属性のアプリケーション固有の情報で提供されます。コネクターは、この情報を解析して SELECT 照会を生成します。

アプリケーション固有ビジネス・オブジェクトを拡張または変更する場合は、ビジネス・オブジェクト定義のアプリケーション固有の情報が、コネクターで予期されている構文に一致するようにしてください。

次のセクションでは、この機能をより詳細に説明します。

トップレベル・ビジネス・オブジェクトの動詞のアプリケーション固有情報

トップレベル・ビジネス・オブジェクトの動詞は、vDynRetBOH ビジネス・オブジェクト・ハンドラーのクラスを指定します。このアプリケーション固有の情報は、必ず次に示すものである必要があります。

```
sap.bapimodule.vDynRetBOH
```

単純属性のアプリケーション固有の情報

属性のアプリケーション固有の情報は、以下の情報を指定します。

- 対応するデータベース表の名前。
- 対応するデータベース列の名前。
- 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・オブジェクトとの外部キー関係。
- オペランド。

アプリケーション固有情報の書式は、4 つの名前 = 値パラメーターから構成されています。各パラメーターには、パラメーター名とその値が含まれています。各パラメーター・セットは、後続のセットとコロン (:) で区切られます。

属性アプリケーション固有の情報の書式を次に示します。大括弧 ([ ]) で囲まれているのは、オプショナル・パラメーターです。オプションのセットの各メンバーは、縦棒 (|) で区切られています。コロンは区切り文字として予約されています。

```
TN=TableName:CN=ColumnName:[FK=[..]fk_attributeName]:[OP=GT|GE|EQ|NE|LE|LT|LIKE]
```

表 37 に、それぞれの名前 = 値パラメーターの説明を示します。

表 37. 属性アプリケーション固有情報内の名前 = 値パラメーター

パラメーター	説明
TN=*TableName* CN=*ColumnName* FK=[..]fk_*attribute Name*	データベース表の名前。 データベース表列 (フィールド) の名前。 このプロパティーの値は、外部キー関係が親ビジネス・オブジェクトに格納されているか、現在のビジネス・オブジェクトに格納されているかによって異なります。 • *attributeName* — 現在のビジネス・オブジェクト内の属性を指定します。詳細については、 201 ページの『例: 現在のビジネス・オブジェクトが外部キーを格納している場合』を参照してください。 • ..attributeName — 親ビジネス・オブジェクト内の属性を指定します。 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の情報に含めないでください。

表 37. 属性アプリケーション固有情報内の名前 = 値パラメーター (続き)

パラメーター	説明
OP=GT|GE|EQ|NE|LE|LT| LIKE	オペランド・オプションは次のとおりです。 • GT— より大 • GE— 以上 • EQ— 等しい (デフォルト・オプション) • NE— 等しくない • LE— 以下 • LT— より小 • LIKE— 類似 パフォーマンスを最大化するために、EQ を指定することをお勧めします。オペランドが指定されていない場合、コネクターは EQ を使用します。

各単純属性に必須のパラメーターは、テーブル名と列名です。オペランドには、デフォルトで EQ (等しい) が使用されます。基本的な書式を次の例に示します。

```
TN=KNA1:CN=KUNNR
```

重要: これらのパラメーターに値を指定する場合は、大文字と小文字の区別に注意してください。

ビジネス・オブジェクト内の単純属性の場合、アプリケーション固有の情報フィールドに値を指定しない (つまり長さを 0 にする) ことができます。コネクターでは、そのような属性は無視されます。これは、コネクターが子ビジネス・オブジェクトの隣接する配列を区切るために使われているプレースホルダー属性を処理しないようにするための便利な方法です。

ビジネス・オブジェクトのすべての属性のすべてのアプリケーション固有情報が、コネクターで照会を作成または実行するために十分な情報を提供しない場合、コネクターは失敗を戻します。

例: 現在のビジネス・オブジェクトが外部キーを格納している場合

202 ページの図 63 に、ビジネス・オブジェクト内の属性を参照する 2 つの外部キーを持つ WebSphere ビジネス・オブジェクトの例を示します。この例では、ビジネス・オブジェクトのデータは 2 つのテーブルで表されています。一方のテーブルにはアドレス・データが、他方のテーブルには都道府県名および国名の省略語に関するルックアップ・データが格納されています。このデータを処理するために、コネクターは 2 回のテーブル読み込みを実行します。

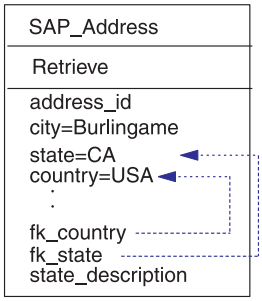


図 63. 例: 現在のビジネス・オブジェクトが外部キーを格納している場合

属性情報: 表 38 に、この例の SAP_Address にある各属性のテーブル名、列名、キー、および外部キーを示します。

表 38. ビジネス・オブジェクト属性の例の説明

属性	テーブル名	列名	キー	外部キー	デフォルト
address_id	ADRC	ADDRNUMBER	true		
city	ADRC	CITY1	false		
state	ADRC	REGION	false		
country	ADRC	LAND1	false		
language	T005U	SPRAS	true		E
fk_country	T005U	LAND1	false	FK=country	
fk_state	T005U	BLAND	false	FK=state	
state_description	T005U	BEZEI	false		

属性アプリケーション固有情報: 表 38 の情報に基づいた fk_state 属性のアプリケーション固有情報は次のとおりです。

```
TN=T005U:CN=BLAND:FK=state
```

fk_country 属性のアプリケーション固有情報は次のとおりです。

```
TN=T005U:CN=LAND1:FK=country
```

SQL 照会: 次の SELECT ステートメントは、コネクターが SAP_Address で表されるテーブルでデータを検索するために作成する WHERE 文節を示したものです。

```
SELECT * FROM ADRC WHERE ADDRNUMBER = address_idValue
SELECT * FROM T005U WHERE SPRAS = 'E' AND LAND1 = countryValue AND
BLAND = stateValue
```

ビジネス・オブジェクトの生成

WebSphere Business Integration システムは SAPODA を備えています。SAPODA を使用すると、ビジネス・オブジェクトと、このビジネス・オブジェクトの SAP アプリケーションでの処理をサポートするために必要なメタデータを定義することができます。SAPODA は、ユーザーがグラフィカル・インターフェースを使用して指定したテーブルからビジネス・オブジェクト定義を生成します。このユーティリティーは、階層型ビジネス・オブジェクト定義を作成するときよりも、個別ビジネス・オブジェクト定義を作成するときに非常に役立ちます。親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は、手動で定義する必要があります。

ビジネス・オブジェクトの生成: SAPODA

SAPODA は、Hierarchical Dynamic Retrieve Module の個別ビジネス・オブジェクト定義を生成します。このユーティリティーを使用して階層型ビジネス・オブジェクト定義を作成する場合は、生成された親ビジネス・オブジェクト定義と子ビジネス・オブジェクト定義の間の関係を手作業で指定する必要があります。

注: テーブル定義に変則的な点がある場合は、ニーズを完全に満たすために、生成されたビジネス・オブジェクトを手動で変更しなければならないことがあります。

SAPODA を使用してビジネス・オブジェクト定義を作成するためのステップ

SAPODA を使用してこのモジュールのビジネス・オブジェクト定義を生成するには、以下のステップを実行します。

1. SAPODA を起動します。
2. Business Object Designer Express を起動します。このユーティリティーは、(ODA へのアクセスを提供することにより) ビジネス・オブジェクト定義の手作業および自動での開発を支援します。
3. Business Object Designer Express の 6 つのステップの処理を実行して、ODA を構成し、実行します。
4. Business Object Designer Express を使用して、生成された定義を手動で変更します。
   - 不要な属性を除去します。

     重要: 1 つのビジネス・オブジェクトで表現される各テーブル内のすべての列について、バイトの合計数は 512 バイトを超えることができません。したがって、定義の長さがこの制限を超えないように、不要な属性を除去する必要があります。詳細については、194 ページの『長いデータ行の取り扱い』を参照してください。
   - 階層型ビジネス・オブジェクト定義を作成する場合は、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を指定します。
   - 必要に応じて変則的な点を修正します。

SAPODA の詳細については、35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』を参照してください。Business Object Designer Express の起動および Business Object Designer を使用したビジネス・オブジェクト定義の手動での変更については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

テーブル間の関係の作成

SAPODA は、指定されたすべてのテーブルのビジネス・オブジェクト定義を生成します。生成が完了した後で、Business Object Designer Express ですべてのテーブルを開いて編集することができます。

SAPODA が生成した複数の個別ビジネス・オブジェクト定義から 1 つの階層型ビジネス・オブジェクト定義を作成するには、以下のステップを実行します。

1. 階層のトップレベルにあるテーブルを判別します。

例えば、トップレベル・ビジネス・オブジェクトが SAP_Customer であるとします。このビジネス・オブジェクトは、Customer_KUNNR という単一のキーを持っています。SAPODA では、この属性に対して次に示すアプリケーション固有の情報が指定されています。

```
TN=KNA1:CN=KUNNR
```

2. すべての子ビジネス・オブジェクトおよび孫ビジネス・オブジェクトを見つけ出し、識別します。
3. トップレベル・ビジネス・オブジェクトおよびその下位階層にある各親ビジネス・オブジェクトに対して、個別の子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表現する属性を追加します。
   - 属性のタイプとして子ビジネス・オブジェクトの名前を指定します。
   - 関係として包含関係を指定します。
   - 該当するカーディナリティーとして 1 または n を指定します。
4. 親のキーに対応する属性を含む個別の子ビジネス・オブジェクト定義に対して、その属性のアプリケーション固有の情報内で外部キー関係を指定します。

   例えば、SAP_Customer の直接の子であるビジネス・オブジェクトには、多くの場合に Customer_KUNNR 属性が含まれています。Customer_KUNNR のアプリケーション固有の情報内で、次の行を指定します。

   ```
   TN=KNVI:CN=KUNNR:FK=..Customer_KUNNR
   ```

   外部キーの指定の詳細については、200 ページの表 37 を参照してください。
5. 対応するテーブルに親オブジェクトのキーが含まれていない子ビジネス・オブジェクト定義を見つけ出します。これらの定義内で、子の基本キーに一致する親の非キー・フィールドを見つけ出します。

   例えば、SAP_Customer_ADRC は親と同じ非キーを持つ第 2 レベル・ビジネス・オブジェクトです。SAPODA は、SAP_Customer 内の非キー・フィールドである Address_number_ADDRNUMBER 属性を使用して、このビジネス・オブジェクト定義を生成します。

   この属性のアプリケーション固有の情報内で、外部キー関係を次のように指定します。

   ```
   TN=ADRC:CN=ADDRNUMBER:FK=..Address_ADRNR
   ```

   注: SAP では、SAP バージョン 3x で作成されたテーブル (例えば KNA1) で使用されている ADDNR フィールドの名前を、SAP バージョン 4x で作成されたテーブル (例えば ADRC) では ADDRNUMBER に変更したため、これら 2 つのフィールド間の関係を認識することは、比較的困難です。

第 6 部 ABAP Extension Module

第 18 章 ABAP Extension Module の概要

この章では、Adapter for mySAP.com の ABAP Extension Module について説明します。ABAP Extension Module により、統合ブローカーは SAP アプリケーションからビジネス・オブジェクトを送受信することができます。

この章の内容は以下のとおりです。

ABAP Extension Module のコンポーネント

ABAP Extension Module は、Java および ABAP で記述されたコンポーネントから構成されています。Java コンポーネントは、コネクター・モジュールと SAP RFC ライブラリーから構成されています。SAP では、Java および C で記述した RFC ライブラリーを提供しています。ABAP コンポーネントは、各種の SAP アプリケーション機能モジュール、データベース表、およびプログラムから構成されています。これらの ABAP コンポーネントは、アダプターの一部として開発、提供されるものと、インストールされた各 SAP システムに固有なものとがあります。

208 ページの図 64 に、ABAP Extension Module の全体的なアーキテクチャーを示します。

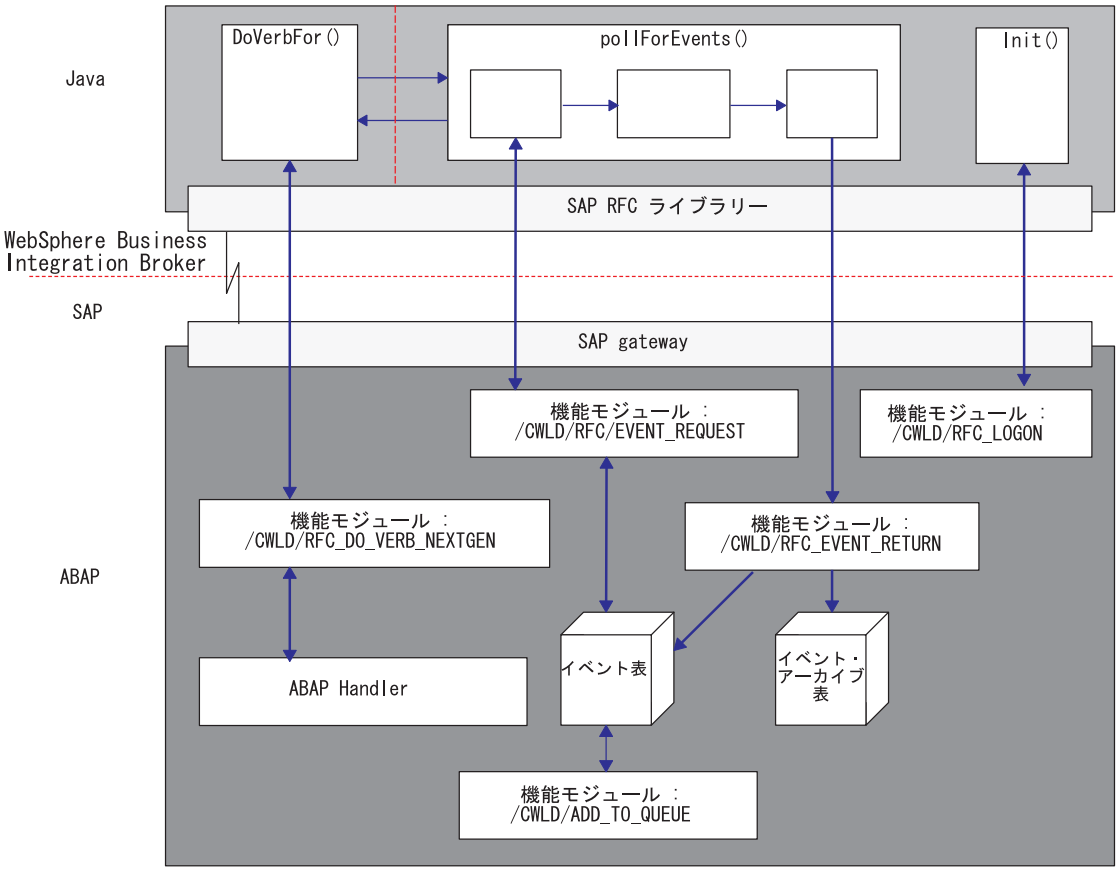


図 64. ABAP Extension Module のアーキテクチャー

Java コンポーネント

コネクターは、JAR (Java Archive) ファイルとして提供され、実行されます。コネクターは、イベント・デリバリーおよびイベント・ビジネス・オブジェクト要求プロセスを処理します。SAP RFC ライブラリーも、JAR ファイルとして納入され、実行されます。このライブラリーは、外部プログラムから SAP アプリケーション内の ABAP 機能モジュールを実行できるようにします。

Java コンポーネントは、以下の処理を行います。

- SAP RFC ライブラリーと SAP Gateway を使用して、SAP アプリケーションへの RFC 接続をオープンします。
- 統合ブローカーからの要求を処理し、その要求をコネクターの ABAP コンポーネントに渡します。
- SAP アプリケーションでイベントをポーリングします。

ABAP コンポーネント

コネクターの ABAP コンポーネントは、機能モジュール、プログラム、およびデータベース表です。これらの要素は、Java コンポーネントによって開始されたイベント・デリバリーおよびビジネス・オブジェクト要求プロセスを処理します。ABAP

コンポーネントは、コネクター・トランスポート・ファイルでデリバリーされて SAP アプリケーションにロードされ、ロードされた後には ABAP リポジトリー・オブジェクトとして実行されます。

ABAP コンポーネントは、以下の処理を行います。

- Java コンポーネントからのビジネス・オブジェクト要求を、特定のビジネス・オブジェクト・タイプおよび動詞を処理するために設計された適切な機能モジュールを呼び出して処理します。
- イベント表内のイベントを検出または起動したり、イベント表にイベントを格納したりします。
- Java コンポーネントからのイベント要求と、そのイベントの戻り (イベント・ステータス更新) を処理します。

ABAP Extension Module の動作方法

ABAP Extension Module が提供するほとんどの機能は、SAP アプリケーションの内部で実行されます。SAP アプリケーションには、すべてのコネクターが実装する必要のある仮想関数のほとんどについて、それに対応する ABAP 機能モジュールがあります。ただし、`init()`、`doVerbFor()`、および `pollForEvents()` の各メソッドに固有な要件に対応する ABAP 機能モジュールは SAP から提供されていないため、これらの機能モジュールはコネクター・モジュールの一部として開発され、提供されています。Java コンポーネントもいくつかの機能を提供しますが、これらのメソッドに関する処理の大部分は、SAP アプリケーション内で ABAP コンポーネントによって行われます。

表 39 に、コネクター・モジュールが実装している仮想 Java メソッドと、それに対応する ABAP コンポーネントを示します。これは、コネクターで使用される ABAP コンポーネントの完全なリストでないことに注意してください。

表 39. Java コンポーネントとそれに対応する ABAP コンポーネント

Java コンポーネント	ABAP コンポーネント
`doVerbFor()`	`/CWLD/RFC_DO_VERB_NEXTGEN`
`getVersion()`	実装不要
`getBOHandlerForBO`	実装不要
`init()`	`/CWLD/RFC_LOGON`
`pollForEvents()`	`/CWLD/RFC_EVENT_REQUEST` `/CWLD/RFC_EVENT_RETURN`
`terminate()`	実装不要

これらの ABAP 機能モジュールは、ABAP Extension Module のコアを構成しています。以下のセクションでは、コネクターの初期化、ビジネス・オブジェクトの処理、およびコネクターでのイベント通知の処理方法について説明します。

実装されている機能については、この章の残りの部分で説明します。

初期化

`init()` メソッドは、宛先の SAP アプリケーションが実行されているかどうか、および ABAP 機能モジュールを実行するために RFC ライブラリーを使用できるかど

うかを確認するために、ABAP 機能モジュールの /CWLD/RFC_LOGON を呼び出します。また、/CWLD/RFC_LOGON 機能モジュールは、進行中のすべてのイベントを処理するときにも呼び出されます。イベント検索済みの状況 (イベント表に R とマークされる状況) がマークされたイベント表にあるイベントはすべて、InDoubtEvents コネクター・プロパティーに基づいて処理されます。デフォルトのプロパティー値は、Ignore です。イベント分配が使用されているときは、その特定のコネクターおよびサーバーに属する、状況が 'R' のイベントだけ、そのコネクター・プロパティーに従って処理されます。イベント分配が使用されていないときは、状況が 'R' のすべてのイベントが、そのコネクター・プロパティーに従って処理されます。コネクター・プロパティーが reprocess であるときは、これらのイベントは、キュー (イベント表で Q とマークされる) の状況に変更されます。コネクターでイベントのポーリングを行うときは、'Q' 状況のすべてのイベントが、/CWLD/RFC_EVENT_REQUEST機能モジュールを使用して処理されます。コネクター・プロパティーが FailOnStartUp に設定されている場合は、SAP ログおよびローカル・ログ・ファイル内に致命的エラーが記録され、コネクターがシャットダウンされます。また、致命的エラーが発生したことを通知する E メールもユーザーに送信されます。コネクター・プロパティーが LogError に設定されている場合は、SAP ログおよびローカル・ログ・ファイルの両方にエラーが記録されます。進行中のイベントは処理されず、またコネクターはシャットダウンされません。コネクター・プロパティーが Ignore に設定されている場合、イベント表に進行中のイベントがないかのように、進行中イベントは無視され、コネクターはポーリングします。

機能モジュールが正常に実行されない場合には、コネクターは終了します。

ビジネス・オブジェクトの処理

SAP 用のすべてのサービス呼び出し要求は、コネクター・モジュールの Java コンポーネントに属する doVerbFor() メソッドにより開始されます。コネクターの ABAP 機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN およびコネクター・モジュールの ABAP コンポーネントに属する ABAP Handler が要求を処理します。

211 ページの図 65 に、ビジネス・オブジェクト処理を示します。

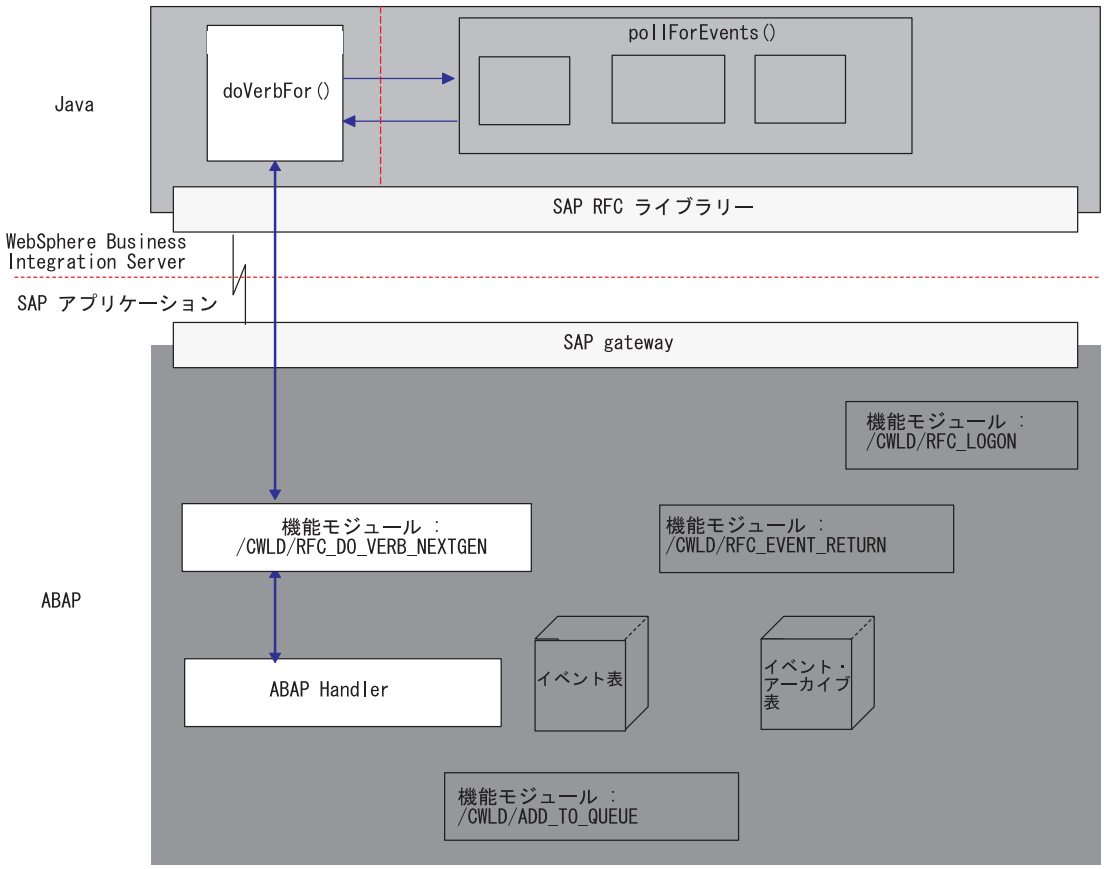


図 65. doVerbFor() のビジネス・オブジェクト処理

doVerbFor()

コネクター・モジュールの Java コンポーネントの中で、実装された単一のビジネス・オブジェクト・ハンドラーの doVerbFor() メソッドが、統合ブローカーからのすべてのビジネス・オブジェクト要求および pollForEvents() メソッドからのすべてのビジネス・オブジェクト・イベントを処理します。いずれの場合も、doVerbFor() は次のように実行されます。

1. SAP 用の WebSphere ビジネス・オブジェクトのインスタンスを、このビジネス・オブジェクト・データを格納している単一の事前定義フラット構造に変換します。
2. ABAP 機能モジュールの /CWLD/RFC_DO_VERB_NEXTGEN を呼び出し、ビジネス・オブジェクト・データを渡してから、ビジネス・オブジェクト・データが戻されるまで待機します。
3. 返されたビジネス・オブジェクト・データを、WebSphere ビジネス・オブジェクトに変換します。

doVerbFor() メソッドは、ビジネス・オブジェクト・データを機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN に渡した後、戻されたビジネス・オブジェクト・データから新たなビジネス・オブジェクト構造を作成します。

/CWLD/RFC_DO_VERB_NEXTGEN

コネクター・モジュールの ABAP コンポーネントでは、コネクターの ABAP 機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN が、SAP アプリケーションでのすべての WebSphere ビジネス・オブジェクト処理を実行します。具体的には、ビジネス・オブジェクト・データを適切な ABAP Handler に発送します。この意味で、機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN は、ビジネス・オブジェクト・ルーターと考えることができます。これは次のように実行されます。

1. ビジネス・オブジェクトを受け取ります。
2. ビジネス・オブジェクト・データを処理する ABAP Handler を動的に呼び出し、ビジネス・オブジェクト・データをパラメーターとして渡します。
3. ビジネス・オブジェクト・データを ABAP Handler から受け取り、要求呼び出しに戻します。

/CWLD/RFC_DO_VERB_NEXTGEN は ABAP Handler を使用して、各オブジェクト・タイプおよび動詞固有の要求を実行します。/CWLD/RFC_DO_VERB_NEXTGEN は、ビジネス・オブジェクトの動詞のアプリケーション固有情報の値を使用して、どの ABAP Handler を呼び出すかを決定します。また、アーカイブ状況についても検査します。/CWLD/RFC_DO_VERB_NEXTGEN は、doVerbFor() メソッドから ABAP Handler へのルーターと見なすことができます。

ABAP Handler

ABAP Handler は、コネクター・モジュールの Java コンポーネントに含まれているビジネス・オブジェクト・ハンドラーの機能を拡張するため、コネクター・モジュールに固有です。ABAP Handler は、SAP アプリケーション内に ABAP 機能モジュールとして存在し、/CWLD/RFC_DO_VERB_NEXTGEN と直接通信します。ABAP Handler は、ビジネス・オブジェクト・データを SAP アプリケーション・データベースとの間で出し入れするために必要です。

213 ページの図 66 に、ABAP Extension Module のビジネス・オブジェクト処理コンポーネントと、それらの相互関係を示します。この図から分かるように、1 つのビジネス・オブジェクト・ハンドラー (doVerbFor()) および ビジネス・オブジェクト・ルーター (/CWLD/RFC_DO_VERB_NEXTGEN) に対して、複数の ABAP Handler が存在します。

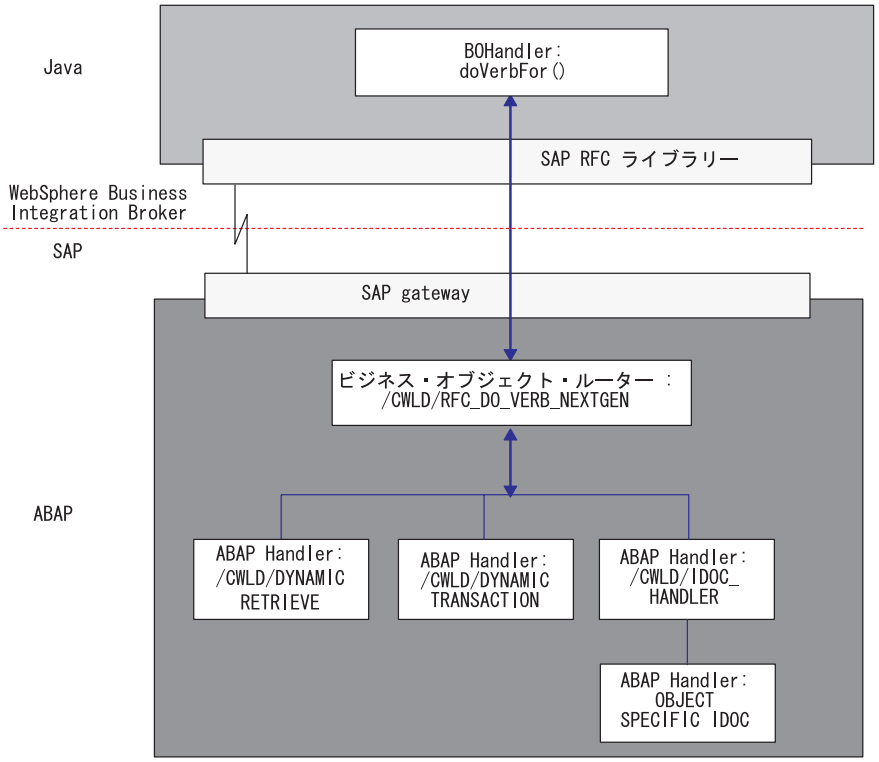


図 66. Adapter 提供のビジネス・オブジェクト処理コンポーネント

ABAP Handler は、ビジネス・オブジェクト・データを SAP アプリケーション・データベースに追加 (Create、Update、Delete) したり、SAP アプリケーション・データベースからデータを検索するためのキーとしてビジネス・オブジェクト・データを使用 (Retrieve) する処理を行います。

アダプターは、汎用の ABAP Handler を提供します。例えば、機能モジュールの /CWLD/DYNAMIC_TRANSACTION は、Create、Update、Delete、および Retrieve 処理について、フラットなビジネス・オブジェクトをサポートします。

WebSphere Business Integration システムはメタデータ・リポジトリーを提供し、アダプターは、フラット・ビジネス・オブジェクトをサポートするため、汎用 ABAP Handler を提供します。さらにアダプターは、階層型ビジネス・オブジェクトをサポートするため、ABAP Handler (/CWLD/IDOC_HANDLER) も提供します。ただし、サポートが必要な各階層型ビジネス・オブジェクトについて、ビジネス・オブジェクト固有の ABAP Handler を追加開発することが必要です。

WebSphere Business Integration システムは、この開発プロセスを容易にするツールを提供しています。ビジネス・オブジェクトおよび ABAP Handler の開発の詳細については、247 ページの『第 21 章 ABAP Extension Module のビジネス・オブジェクトの開発』および 35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

イベント通知

イベント通知とは、コネクターに SAP アプリケーション・オブジェクト・イベントを通知する処理の集合のことです。通知には、イベントのタイプ (オブジェクト

および動詞)、および外部システムが関連データを検索するために必要とするデータ・キーが含まれますが、それだけに限定されません。

図 67 に、pollForEvents() メソッドを使用したイベント通知プロセスを示します。

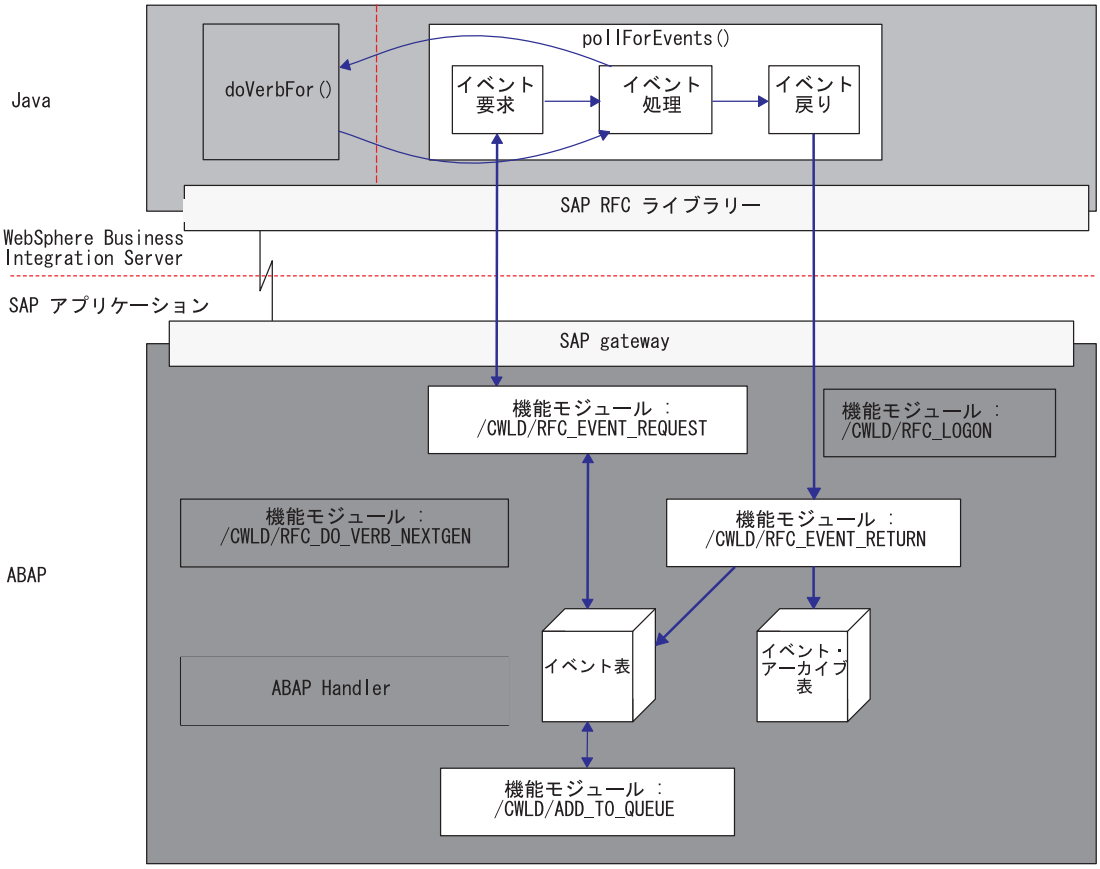


図 67. イベント通知プロセス

コネクターに対するイベント通知は、次の 2 つの機能から構成されています。

- 『イベント・ポーリング』
- 217 ページの『イベント・トリガー』

イベント・ポーリング

イベント・ポーリングは、pollForEvents() メソッドにより実行される次の 3 つの機能から構成されます。

- 215 ページの『イベント要求』
- 216 ページの『イベント処理』
- 216 ページの『イベント戻り』

注: これらの機能の役割は、Java コンポーネントおよび ABAP コンポーネントでは分散されています。ただし、イベント・ポーリングでは常に Java コンポーネントが開始します。

イベント要求: イベント要求とは、SAP アプリケーションのイベント表に対してイベントをポーリングおよび検索する処理です。Java コンポーネントのイベント要求機構に対して、SAP アプリケーション側には、それに対応する機能モジュール /CWLD/RFC_EVENT_REQUEST があります。この機能は、コネクターの ABAP イベント表 /CWLD/EVT_CUR からイベントを検索します。

起動されたすべてのイベントは、初期状況がプレキュー (イベント表で P としてマークされた状況)、イベント優先順位がデフォルトの 0 という状態でイベント表に入れられます。イベントを処理する前に、イベントの状況をキュー (イベント表では Q) に変更する必要があります。イベントの優先順位は、イベントが表す完全なオブジェクトをコネクターが検索する前には 0 である必要があります。イベントの優先順位の詳細については、220 ページの 『イベント優先順位』を参照してください。

イベントを作成したユーザーとイベントのキーとの組み合わせについてデータベース・ロックが存在しなければ、イベントの状況がプレキューからキューに変わります。イベント表からのイベントの検索後、イベントの状況はイベント検索済み (イベント表では R) に更新されます。ロックが存在している場合には、イベントの状況はロック (イベント表では L) に設定され、そのイベントは再びキューに入れられます。ABAP 定数 C_MAXIMUM_REQUEUE は、1 つのイベントを再びキューに入れられる回数を定義します。最大数 (デフォルトでは 100) に達すると、イベントはイベント・アーカイブ表にアーカイブされます。

注: 状況がプレキューまたはロックのイベントは、ポーリングのたびに更新されます。複数のイベントが同時に起動されると、パフォーマンス上の問題が発生する可能性があります。PollFrequency 構成プロパティーを使用してポーリングの頻度を構成できます。詳細については、315 ページの『付録 B. 標準構成プロパティー』を参照してください。

ABAP 機能モジュール /CWLD/RFC_EVENT_REQUEST は、すべてのプレキュー・イベントを前処理した後、コネクター・モジュールの Java コンポーネント内のイベント要求メソッドに戻すイベントを選択します (状況がキューのイベントだけが選択されます)。コネクター固有の構成プロパティー PollQuantity (デフォルトは 20) によって、単一のポーリングで戻されるイベントの最大数が決定されます。

イベント要求機構は、イベント選択プロセスは 2 つのステップとして実行します。

1. コネクターおよび統合ブローカー専用のイベントを選択します。

   イベント分配表 (/CWLD/EVT_DIS) の中で、イベントは特定の統合ブローカーに専属しています。この表に指定された統合ブローカーの名前は、コネクターを始動するショートカットに指定された名前と一致していることが必要です。例えば、Windows 上で稼働する SAP コネクター用の標準のショートカットは次のフォーマットです。

   ```
   ...¥start_SAP.bat SAPconnectorName integrationBrokerName -cConfigFileName
   ```

2. 選択されたイベント数が最大イベント数より小さい場合には、イベント配布用としては構成されていないイベントから、残りのイベントが補てんされます。

例えば、コネクター固有の構成プロパティー PollQuantity が 20 に保持され、特定のコネクターおよび統合ブローカーに専属するイベントが 8 つある場合には、イベント要求機構は 12 の追加イベントを選択します。

イベント処理: イベント要求機能は、処理されるイベントの配列を /CWLD/EVT_CUR イベント表から作成します。イベント要求機能は、これらのイベントをイベント処理機能に渡します。イベント処理機能は、これらのイベントを次の方法で一度に 1 つずつ処理します。

1. object.verb 値を使用して、そのイベントがコネクター・サブスクリプション・リストにあるかどうかを評価します。

   イベントがサブスクリプション・リストにない場合は、イベントの状況を not subscribed に設定します。
2. イベントがサブスクリプション・リストにあれば、parentObjectOnly.Retrieve ビジネス・オブジェクトを作成します。イベント処理機能は、次のいずれかの方法でキー値を設定します。
   - イベントのキー値に |Cx| 区切り文字が含まれていない場合、コネクターは最初のキー属性の値をイベントのキーで指定された値に設定します。この場合、複合キーは、singleton として扱われます。また、ABAP ビジネス・オブジェクト処理機能モジュールで解釈される必要があります。
   - イベントのキー値に |Cx| 区切り文字のインスタンスが 1 つ以上含まれている場合、コネクターは指定された各属性の値をそれぞれの指定された値に設定します。

   イベントの複合キーを指定する方法の詳細については、276 ページの『名前と値のペアとしての複合キーのコーディング』を参照してください。
3. doVerbFor() を起動し、ビジネス・オブジェクト・データをこの関数に渡します。ビジネス・オブジェクトを渡すと、イベント処理はビジネス・オブジェクト・データが戻されるまで待機します。
4. doVerbFor() の処理に基づいて、イベント配列の状況を更新します。
5. ビジネス・オブジェクト・データが正常に検索された場合は、ビジネス・オブジェクト・データを統合ブローカーに送信します。

イベント戻り: 各イベントは、イベント要求により処理された後、機能モジュール /CWLD/RFC_EVENT_RETURN を使用して、SAP アプリケーションに戻ります。この機能モジュールは、処理されたイベントのコピーを作成し、イベント・アーカイブ表 (/CWLD/EVT_ARC) に追加した後、イベント表から元のエントリーを削除します。

注: 状況が新しくなったイベントは、個々のイベントが処理された後、すべて更新されます。

アーカイブされたイベントには、正常に処理されたイベント、処理されたがエラーで終了したイベント、およびアンサブスクライブされたイベントが含まれます。各イベントの状況は、次のいずれかの状態を示すことができます。

- このビジネス・オブジェクトは統合ブローカーに正常に送信されました。
- このイベントはコネクターからの不明な Java 戻りコードを生成しました。
- このイベントは SAP アプリケーションからデータを検索しようとして失敗しました。

- このイベントはビジネス・オブジェクトがロックされていたためタイムアウトになりました。
- イベントにサブスクライブするコラボレーションはありません。

イベント・アーカイブ表を管理するには、SAP アプリケーション内で IBM WebSphere BI Station ツールを使用します。IBM WebSphere BI Station を使用すると、管理者はアーカイブ表を表示して切り捨てたり、処理するためにイベントを再サブミットすることができます。アーカイブ表の保守およびログ切り捨てのセットアップの詳細については、283 ページの『第 23 章 ABAP Extension Module の管理』を参照してください。

イベント・トリガー

コネクターはイベント・ドリブンです。SAP アプリケーションからイベントを取り出すには、IBM WebSphere でサポートされる各オブジェクトについて、イベント・トリガー機構を実装する必要があります。コネクターのイベント・トリガーは、次の 3 つの機能から構成されています。

- 『イベント検出』
- 218 ページの『イベント・トリガー』
- 221 ページの『イベント永続性』

イベント検出: イベント検出とは、あるイベントが SAP アプリケーション内で生成されたことを識別する処理です。通常、コネクターではイベントを検出するためにデータベース・トリガーを使用します。ただし、SAP アプリケーションは SAP データベースと緊密に統合されているため、SAP ではそのデータベースの直接変更を目的としたアクセスは非常に制限されます。そのため、イベント検出機構は、データベースの上のアプリケーション・トランザクション層に実装されています。

SAP アプリケーション内でのイベントを検出するために、一般に次の 4 つの機構が使用されています。

- コード拡張
- バッチ・プログラム
- ビジネス・ワークフロー
- 変更ポインター

これらのイベント検出機構は、すべてオブジェクトのリアルタイム・トリガーおよび検索をサポートしています。さらに、コード拡張およびバッチ・プログラムは、イベントの検索を遅らせる機能を提供します。検索が遅延されるイベントは、将来のイベントと呼ばれます。将来のイベントのトリガーの詳細については、**『イベント・トリガー』**を参照してください。

注: それぞれのイベント検出機構には、ビジネス・オブジェクト・トリガーを設計および開発する際に考慮する必要のある利点と欠点があります。イベント検出機構の実装の詳細については、269 ページの『第 22 章 ABAP Extension Module のイベント検出の開発』を参照してください。

これらはイベント検出機構のごく一部の例に過ぎません。イベントを検出するには、多くの方法があります。

イベント・トリガー: イベントは、イベント検出機構の 1 つによって識別されると、アダプターから送出されるイベント・トリガーの 1 つを使用して起動されます。

- /CWLD/ADD_TO_QUEUE — 即時に処理するためにイベントを現在のイベント表に起動する機能モジュール。
- /CWLD/ADD_TO_QUEUE_IN_FUTURE — 後で処理される将来のイベント表にイベントを起動する機能モジュール。

注: どちらの機能も、リアルタイム・トリガー用です。/CWLD/ADD_TO_QUEUE はイベントを即時に処理し、/CWLD/ADD_TO_QUEUE はイベントを後で処理します。

イベントがリアルタイムに起動される場合、/CWLD/ADD_TO_QUEUE はイベントを現在のイベント表 (/CWLD/EVT_CUR) にコミットします。具体的には、オブジェクト名、動詞、およびそのイベントを示すキーを含んだデータ行を追加します。

図 68 に、/CWLD/ADD_TO_QUEUE により起動されるイベントを示します。

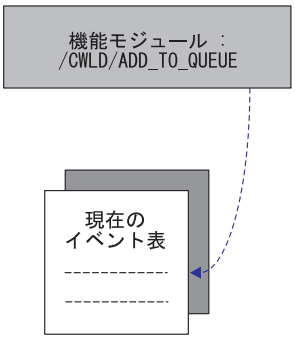


図 68. /CWLD/ADD_TO_QUEUE

イベントが将来の時点で処理される必要がある場合、/CWLD/ADD_TO_QUEUE_IN_FUTURE はイベントを将来のイベント表 (/CWLD/EVT_FUT) にコミットします。具体的には、オブジェクト名、動詞、およびそのイベントを示すキーを含んだデータ行を追加します。さらに、アダプターから提供されるバッチ・プログラム /CWLD/SUBMIT_FUTURE_EVENTS により読み取られるデータ行を追加します。このバッチ・プログラムは、将来のイベント表からイベントを検索するようにスケジュールできます。イベントを検索すると、/CWLD/ADD_TO_QUEUE を呼び出して、イベントを現在のイベント表に起動します。

注: /CWLD/ADD_TO_QUEUE_IN_FUTURE は、将来のイベント表の日付行に日付を取り込む際に、現在日付としてシステム日付を使用します。

219 ページの図 69 に、/CWLD/ADD_TO_QUEUE_IN_FUTURE により起動されるイベントを示します。

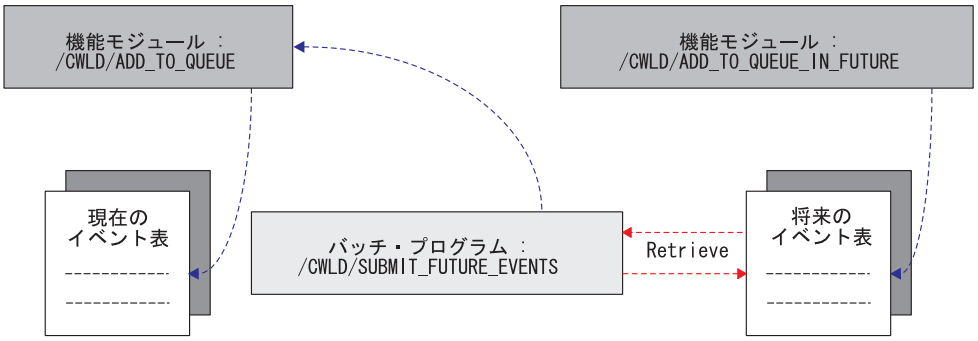


図 69. /CWLD/ADD_TO_QUEUE_IN_FUTURE

将来のイベント表のイベントのトリガーの詳細については、269 ページの『第 22 章 ABAP Extension Module のイベント検出の開発』を参照してください。

すべてのイベントは、/CWLD/ADD_TO_QUEUE を使用して現在のイベント表に追加されます。/CWLD/ADD_TO_QUEUE は、現在のイベント表にデータ行を追加するほかに、次に示す目的のためにセットアップすることもできます。

- イベントのフィルター操作
- イベント分配
- イベント優先順位

イベントのフィルター操作、イベント分配、およびイベント優先順位は、イベント・トリガーの一部として実行され、他のプログラムから実行されることはありません。これらを実行した結果は、イベントの制限 (フィルター操作)、または変更 (イベント分配またはイベント優先順位付け) になります。

イベントのフィルター操作

イベント・トリガーを使用すると、イベントをフィルターに掛けて特定のイベントをイベント表に追加しないようにすることができます。アダプターは、この目的のために特定のイベントを制限することのできる ABAP 組み込みプログラム (/CWLD/TRIGGERING_RESTRICTIONS) が提供されます。

イベント分配

ロード・バランシングを使用することで、イベント処理を複数のコネクターに分配できます。これにより、複数のイベントを同時に処理できます。イベント・トリガーは、この機能をイベント分配表 (/CWLD/EVT_DIS) を通じて提供します。ビジネス・オブジェクトは、特定のコネクターによってのみ検索されるようにすることができます。また、イベント分配は 1 つのイベントを取得して、コネクターと統合ブローカーのサブスクライブされた組み合わせごとに 1 回以上複製することができます。

重要: 複数のコネクターを使用してポーリングしている場合は、サブスクライブされた各イベントを特定のコネクター専用にする必要があります。そのようにしない場合、イベントが重複してデリバリーされる可能性があります。異なるコネクター専用となっている複数のオブジェクトが相互に依存関係を持たないようにする必要があります。これは、依存関係があると、イベントが順序どおりにデリバリーされない可能性があるためです。

例えば、CrossWorlds1 という名前の統合ブローカーが 1 つあり、これが 2 つの異なるビジネス・オブジェクト BO_A および BO_B にサブスクライブされていると仮定します。BO_A ビジネス・オブジェクトは小さくて、素早く取り込むことができますが、BO_B は大きく、取り込むのに時間がかかるとします。SAP1connector と SAP2connector という 2 つのコネクターがポーリングしている場合、SAP1connector が BO_A を検索し、SAP2connector が BO_B を検索するようにイベント分配表をセットアップすることができます。SAP1connector はタイプ A の小さなオブジェクトを継続してポーリングすることができ、SAP2connector はタイプ B の大きなオブジェクトにフォーカスします。

重要: イベント分配表が特定のオブジェクト用に構成されていない場合には、そのオブジェクトに対して起動された各イベントは、コネクターと統合ブローカーのどの組み合わせでも処理できます。

イベント優先順位

イベント優先順位は、イベントのリトリーブを遅らせることにより、ビジネス・オブジェクト、コネクター、および統合ブローカーの組み合わせそれぞれについて設定することができます。イベントの優先順位は、そのイベントがデリバリーに選出されるまでに必要なポーリングの回数を示します。例えば、あるイベントの優先順位を 10 に設定すると、コネクターはそのイベントが検索されるまでに、イベント表を 10 回ポーリングします。コネクターがポーリングするたびに、優先順位の値は 0 になるまで 1 ずつ削減されます。

デフォルトでは、すべてのイベントの優先順位は 0 に設定されます。オブジェクトの優先順位は、イベント分配と同じ ABAP 表で構成します。

221 ページの図 70 に、SAP アプリケーション内部でのイベント・トリガー機能を示します。イベント E1、E2、および E3 は、イベント・トリガー /CWLD/ADD_TO_QUEUE が受け取ります。E1 は Customer イベントを表し、E3 は Order イベントを表します。イベント分配は、すべての Customer オブジェクトが SAP1connector によって処理され、すべての Order オブジェクトが SAP2connector によって処理されるようにセットアップされています。この環境では、両方のコネクターが同じ統合ブローカーを使用します。E1 は Customer オブジェクトなので SAP1connector によってポーリングされ、E3 は Order オブジェクトなので SAP2connector によってポーリングされます。 E2 は Inventory オブジェクトで、特定の倉庫への在庫オブジェクトを制限する制限プログラム /CWLD/TRIGGERING_RESTRICTIONS 内のコードによってフィルターに掛けられて除外されます。

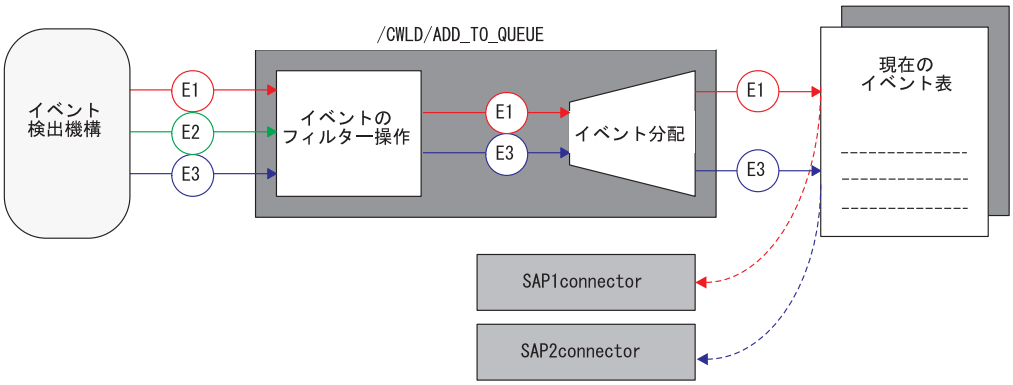


図 70. 機能モジュール /CWLD/ADD_TO_QUEUE によるイベント優先順位の設定

イベント永続性: イベント・トリガーがイベントをイベント表に挿入すると、そのイベントはイベント分配とイベント優先順位の値を設定された状態でデータベースにコミットされます。この時点で、イベントを変更できるのはポーリングだけです。イベントのポーリング・プロセスが完了した時点、つまりイベントが SAP アプリケーションから検索され、コネクターの Java コンポーネントによって処理された時点で、処理されたイベントのコピーがイベント・アーカイブ表 (/CWLD/EVT_ARC) に追加されます。その後、元のイベントはイベント表から削除されます。

注: イベントは、アーカイブ表から再サブミットすることができます。イベントはイベント表に移動されるだけで、再び起動されるのではないことに注意する必要があります。具体的に言えば、イベントのフィルター操作、イベント分配、およびイベント優先順位を通じて戻されるのではありません。

第 19 章 ABAP Extension Module のインストールとカスタマイズ

この章では、ABAP Extension Module のインストールとカスタマイズについてのみ説明します。コネクターのインストールと構成は既に完了していることが前提とされています。コネクターのインストールと構成の詳細については、315 ページの『付録 B. 標準構成プロパティー』を参照してください。コネクターのカスタマイズはオプションですが、推奨されています。

この章の内容は以下のとおりです。

コネクターのコンポーネントはすべて、Windows では `¥connectors¥SAP` ディレクトリーに、Linux では `/connector/SAP` および `/bin` ディレクトリーにあります。トランスポートは、後出の『コネクター・トランスポート・ファイルのインストール』で説明するように、SAP アプリケーションまたはデータベース・サーバーにインストールされます。

注: 本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。Linux のインストールの場合には、円記号 (¥) をスラッシュ (/) に置き換えてください。ファイルのパス名はすべて、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。

コネクター・トランスポート・ファイルのインストール

Adapter for mySAP.com 用のトランスポート・ファイルには、表構造、関数、データなど各種のオブジェクトが格納されています。ABAP Extension Module で必要な特定の機能を提供するには、これらの開発オブジェクトを SAP システムにインポートする必要があります。

各トランスポート・ファイルは、`.zip` ファイルに入っています。例えば、SAP バージョン 4.x Primary トランスポート用のトランスポート・ファイルは、`4_Primary.zip` ファイルに含まれています。

必要なトランスポート・ファイルが正常にロードされたら、ビジネス・オブジェクト固有のトランスポートを任意の順序でロードできます。トランスポート・ファイルの詳細については、各トランスポートの `.zip` ファイルに含まれているトランスポート・メモを参照してください。

コネクター・トランスポートのインストールのためのネーム・スペースの作成

コネクター・トランスポート・ファイルをインストールする前にコネクター用のネーム・スペースを作成します。ネーム・スペースを作成していないと一部のトランスポートが失敗する可能性があるため、このステップは SAP バージョン 4.0 では必須です。

注: SAP アプリケーションのどの 4.x バージョンでも、コネクターの ABAP オブジェクトの 1 つを変更する前に、コネクターのネーム・スペースを作成することが必要です。

/CWLD/ ネーム・スペースの作成

1. 「Workbench Organizer: Tools」ウィンドウを開きます (トランザクション SE03)。
2. 「Administration」メニューを展開し、「Display/change namespaces」オプションをダブルクリックします。
3. 「Display」->「Change」ボタンをクリックします (Ctrl+F1)。
4. 「Continue」ボタンをクリックして、「Information」ウィンドウを閉じます。
5. 「New entries」ボタン (F5) をクリックし、「Namespace」フィールドに `/CWLD/` と入力します。
6. 「Namespace role」フィールドを選択し、それを展開して (F4) オプションを表示してから、「Recipient」を選択します (C)。
7. 「Short text」フィールドに `CrossWorlds Namespace` と入力し、「Owner」フィールドに `CrossWorlds` と入力します。「Save」ボタンをクリックします (Ctrl+S)。システムがカスタマイズの変更を追跡するようにセットアップされている場合は、変更要求のためのプロンプトが出されます。これにより、ネーム・スペースを別のシステムに移送することができます。

ネーム・スペースの変更可能化

コネクターのネーム・スペース内の ABAP オブジェクトは、ネーム・スペースを変更可能にするまでは変更できません。SAP4.x 提供の ABAP オブジェクトを更新するには、オブジェクトを変更するための修理ライセンスが必要です。このライセンスを取得するには、IBM ソフトウェア・サポートにお問い合わせください。

1. 「Workbench Organizer: Tools」ウィンドウを開きます (トランザクション SE03)。
2. 「Administration」メニューを展開し、「Display/change namespaces」オプションをダブルクリックします。
3. 「Display」->「Change」ボタンをクリックします (Ctrl+F1)。
4. 「Continue」ボタンをクリックして、「Information」ウィンドウを閉じます。
5. 「`/CWLD/`」をダブルクリックし、修理ライセンスを入力します。「Save」ボタンをクリックします (Ctrl+S)。
6. 「Back」ボタン (F3) を 2 回クリックし、「Administration」メニューを展開して、「`Set system change`」オプションをダブルクリックします。
7. 「Namespace」行の「Modifiable」列にチェック・マークを付けます。「Save」ボタンをクリックします (Ctrl+S)。

コネクター・トランスポート・ファイル

コネクターには、2 つのコネクター・トランスポート・ファイルが格納されています。アダプターで必要な変更は、これらのコネクター・トランスポート・ファイルで処理されます。

これらの表のデータが追加される前に必要な表が作成されるようにするため、トランスポートはこのリストの順序でインストールする必要があります。これらのファイルは、*ProductDir*¥connectors¥SAP¥dependencies (*ProductDir* はコネクターがインストールされているディレクトリー) にあります。

表 40. 各バージョンのコネクター・トランスポート・ファイル

SAP のバージョン	トランスポート・ファイル
V.4.0, V.4.5, V.4.6	• ¥connectors¥SAP¥dependencies¥transports_40_45_46¥40_45_46_Primary.zip • ¥connectors¥SAP¥dependencies¥transports_40_45_46¥ 40_45_46_Infrastructure.zip
V.4.7	• ¥connectors¥SAP¥dependencies¥transports_47¥47_Primary.zip • ¥connectors¥SAP¥dependencies¥transports_47¥47_Infrastructure.zip

Primary ファイルと Infrastructure ファイルによって提供される機能を次に示します。

表 41. コネクター・トランスポート・ファイルの機能

ファイル	機能
Primary	このトランスポート・ファイルには、次の要素が含まれています。 • システムに 1 回だけロードする必要がある開発オブジェクト。このファイルには、数値範囲オブジェクト、開発クラス、動的トランザクション宣言組み込みプログラム、および制限組み込みプログラムが含まれています。これらを利用することで、トリガー・ロジックにカスタマー独自の変更を行うことができます。このトランスポート・ファイルを既にコネクターが稼働しているシステムに適用すると、このトランスポート・ファイルの内容で、既存の環境にあるすべてのオブジェクトが上書きされます。 • 4 つの数値範囲が初期状態で含まれています。Primary トランスポート・ファイルを再インポートすると、コネクターの既存の数値範囲間隔が初期化されます。この結果、コネクターのログ表、現在のイベント表、将来のイベント表、およびアーカイブ表を再使用前にリフレッシュしなければ、これら表内のデータは破壊されます。
Infrastructure	このクライアントから独立したトランスポート・ファイルには、次の要素が含まれています。 • 要求、デリバリー、開発、および保守コンポーネントで共用されるオブジェクトおよび機能。例えば、このファイルにはログおよびデータ・エレメントが含まれています。 • ビジネス・オブジェクト要求操作をサポートするために必要な機能。 • イベントの起動やイベントのポーリングなどのイベント・デリバリー操作をサポートするために必要な機能。 • ログ統計やイベント表の表示などの保守操作をサポートするために必要な機能。 • オブジェクトの開発をサポートするために必要な機能。

コネクター・トランスポート・ファイルのインストール

コネクター・トランスポート・ファイルは、コネクターとともに提供されるプログラムおよびその他の開発オブジェクトをインポートすることにより、SAP に対する必要な変更をすべて行います。これらのファイルは、SAP プログラムやユーザー出口を変更しません。

重要: トランスポートを再適用すると、環境がリセットされることに注意してください。トランスポート・ファイルを再適用する前に行ったすべての開発は上書きされます。

次の手順で、SID は SAP システムの ID を示し、*<TransportFileName>* はトランスポート・ファイルの名前を示します。ただし、インストール・ディレクトリーでは、トランスポート・ファイル名を構成する文字は、そのファイル名が各種の tp コマンドにパラメーターとして渡される仕方とは異なる順序で表示されます。¥usr¥sap¥trans¥cofiles ディレクトリーの中で、トランスポート・ファイル名のフォーマットは、K9*xxxxx*.*SID* ですが、ファイル名をパラメーターとして渡すときには、*SID*K9*xxxxx* のフォーマットとなります。例えば、ファイル名 K912345.D30 の場合、D30 がソース・システムの SID であるため、パラメーターとして渡される際には D30K912345 となります。

重要: コネクター・トランスポート・ファイルの名前は変更しないでください。

トランスポートをインストールするには、以下の手順を行います。

1. SAP 管理者 <SID>adm としてログインします。
2. トランスポートを SAP データベース・サーバーにコピーします。トランスポート・ファイルには、次の 2 種類があります。
   a. K で始まる名前のファイルを ¥usr¥sap¥trans¥cofiles ディレクトリーにコピーします。
   b. 他のファイルを ¥usr¥sap¥trans データ・ディレクトリーにコピーします。
3. 次の tp connect コマンドを実行することにより、データベースとの接続を確認し、tpparam ファイルのパスを決定します。

   ```
   tp connect <SID>
   ```

   このコマンドが失敗した場合は、第 2 のパラメーターとして tpparam ファイルのパスを次のように追加してください。

   ```
   tp connect <SID> pf = <path of tpparam>
   ```

   例えば、SID が P11 で、tpparam ファイルのパスが ¥usr¥sap¥trans¥bin¥tpparam の場合、コマンドは次のようになります。

   ```
   tp connect P11 pf = ¥usr¥sap¥trans¥bin¥tpparam
   ```

   tpparam ファイルのパスを指定した場合に tp connect が成功し、指定しなかった場合に失敗する場合には、コマンドにステップ 3 で説明したオプショナルの tpparam パスを指定してください。
4. 次の 2 つの方法のうちのいずれかで、トランスポート・ファイルをインポートします。
   - 227 ページの『アダプター配布のコマンドの使用』

- 『SAP トランザクション・コードの使用』

アダプター配布のコマンドの使用

¥usr¥sap¥trans¥bin で、各トランスポートに対して次のコマンドを、指定された順序で実行します。

```
tp addtobuffer <TransportFileName> <SID> pf = tpparamFilePath
tp import <TransportFileName> <SID> u023689 CLIENT=<CLIENT#> pf = tpparamFilePath
```

SAP トランザクション・コードの使用

トランスポート管理で、 システム (トランザクション STMS):

1. 「Import overview」アイコンをクリックします (F5)。
2. 更新する適切なキューをダブルクリックします。
3. メニュー・バーで、「Extras」をクリックし、次に「Other requests」をクリックしてから、「追加」をクリックします。
4. トランスポート要求フィールドに取り込み、チェック・マークをクリックします (Enter)。
5. 「Add Transport Request confirmation」ウィンドウが表示されたら、「Yes」をクリックしてインポートをキューに付加します。
6. 追加されたトランスポートにカーソルを置きます。
7. メニュー・バーで、「Request」をクリックし、次に「Import」をクリックします。
8. 「Target client」フィールドに取り込み、チェック・マークをクリックして、トランスポート・ファイルをインポートします。

トランスポートは、 223 ページの『コネクター・トランスポート・ファイルのインストール』にリストされている順序でインストールする必要があります。

トランスポートをインストールしたら、ユーザーの開発クラスのマイグレーション・パスに従って、開発クラスを変更します。IBM WebSphere BI Station (トランザクション /n/CWLD/HOME) を使用します。

1. 「Tools」タブをクリックし、次に「Transport Layer」ボタンをクリックします。
2. 適切な「Transport layer」エントリーを選択し、「Save」ボタンをクリックします。

重要: コネクター・トランスポートに含まれていた開発オブジェクトに変更を加える場合は、SAP の外部で詳細に文書化しておく必要があります。トランスポート・ファイルの次のリリースにより、変更が上書きされる可能性があります。この場合には、変更を手動で再適用することが必要です。アップグレードの問題について詳しくは、 293 ページの『第 24 章 ABAP Extension Module のアップグレード』を参照してください。

コネクター・トランスポート・ファイルのインストールの確認

コネクター・トランスポート・ファイルが正常にインストールされたことを確認する手順は次のとおりです。

- 228 ページの『トランスポート・ファイルが SAP アプリケーションに移動したことの確認』

- 『SAP がオブジェクトを正常に生成したことの確認』

トランスポート・ファイルが SAP アプリケーションに移動したことの確認

コネクター・トランスポート・ファイルが SAP アプリケーションに物理的に移動されていることを検証するには、次のいずれかの方法で、トランスポート・ログを検査します。

- トランスポート・オーガナイザー (トランザクション SE01) を使用します。
- トランスポート管理システムのグラフィック・インターフェース (トランザクション STMS) を使用します。

トランスポート・オーガナイザー (トランザクション SE01) の使用

トランスポート・オーガナイザー (トランザクション SE01) を使用するには、以下の手順を行います。

1. 番号フィールドにトランスポート・ファイルの名前を取り込みます。
2. 「Display」をクリックして、ログを表示します。

トランスポート管理システムのグラフィック・インターフェース (トランザクション STMS) の使用

トランスポート管理システムのグラフィック・インターフェース (トランザクション STMS) を使用する手順は次のとおりです。

1. 「Import overview」アイコンをクリックします (F5)。
2. 適切なキューをダブルクリックします。
3. トランスポート番号を右クリックして、「Logs」を選択します。
4. ログを調べて、インストールが成功しているかどうかを確認します。

SAP がオブジェクトを正常に生成したことの確認

SAP でオブジェクトが正常に生成されていることを検証するには、以下の手順を行います。

1. トランザクション SE38 に移動します。
2. プログラム `/CWLD/CONSTANTS` を入力します。
3. 「Source Code」を選択して、「Display」をクリックします。
4. 「Program」メニューで、「Generate」をクリックします。
5. 「Select All」をクリックし、次に「Continue」をクリックします (F2)。

これにより、これらのプログラムを含むすべてのアダプター・プログラムが生成されます。

応答が `Programs successfully generated` の場合には、トランスポートに成功したと想定できます。

ABAP Extension Module のアップグレード

コネクターのアップグレードには、最新のアダプター・ファイルをインストールしたり、ABAP Extension Module 用の最新の ABAP トランスポート・ファイルをロードしたりすることが含まれます。アップグレードに関する追加情報については、「*システム・インストール・ガイド (Linux 版)* 」または「*システム・インストール・ガイド (Windows 版)*」を参照してください。

始動前に現在のコネクター・ファイル (構成ファイルおよびメッセージ・ファイル `CN_SAP.txt` および `SAPConnector.txt` など) をバックアップしておくことをお勧めします。コネクター定義をリポジトリーにロードする前に、サポートされているすべてのオブジェクト参照 (ただし必要なものは除く) を削除することも可能です。

重要: 最新の Primary および Number Range トランスポート・ファイルをインストールすると、既存の数値範囲間隔情報が上書きされます。数値範囲間隔が上書きされると、数値範囲間隔が 0 にリセットされるため、既存のイベントおよびオブジェクトが同期しなくなる可能性があります。

最新バージョンのコネクターをインストールした後、これから使用する SAP バージョンに対応する最新の ABAP トランスポート・ファイルをインストールします。最新ファイルがないと、ABAP Extension Module の既存コンポーネントをトランスポートできません。トランスポートのインストールは、223 ページの『コネクター・トランスポート・ファイルのインストール』で説明しています。新しいトランスポート・ファイルをインストールするとアダプター提供のコードに対する変更がすべて上書きされることに注意してください。

使用する環境に対応した正しいトランスポート・ファイルを使用してください。例えば、SAP バージョン 4.6 の環境の場合は、アダプターの 4.x トランスポート・ファイルをインストールします。これにより、オブジェクトをロードしたときに出力される警告やエラーのメッセージは、SAP バージョン 4.x 環境に関するものとなり、SAP バージョン 3.x に起因するものは除外されます。この結果、SAP バージョン 4.x への移行に伴って発生する問題を解決することができます。

最新バージョンのコネクターをインストールし、これから使用する SAP バージョンに対応する最新の ABAP トランスポート・ファイルをインストールした後、新しいコネクターを構成します。

コネクターを SAP 3.x バージョンからこのバージョンにアップグレードする場合は、新しいコネクター固有の構成プロパティーである 347 ページの『Modules』と 348 ページの『Namespace』を構成する必要があります。これらのプロパティーの詳細については、341 ページの『コネクター固有の構成プロパティー』を参照してください。

コネクターに対する SAP アプリケーションの使用可能化

コネクターのインストールと、標準およびコネクター固有の構成プロパティーの構成が終わったら、オプションとして、SAP アプリケーション内からコネクターのイベント処理およびロギング機能を変更することができます。

イベント分配のセットアップ

ロード・バランシングを使用すると、イベントおよびビジネス・オブジェクト要求の処理を、複数のコネクターに分配できます。コネクターが同時に処理できるトランザクションは 1 つのみです。ただし、特定のビジネス・オブジェクトを処理するように複数のコネクターを設定しておけば、複数のイベントと複数のビジネス・オブジェクトを同時に処理できます。複数コネクターのセットアップの詳細については、4 ページの『複数のコネクター・インスタンスの作成』を参照してください。

複数のコネクターに対するイベント分配をセットアップするには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。
2. 「Configuration」タブをクリックし、次に「Event Distribution」ボタンをクリックします。
3. 「New Entries」ボタンをクリックし (F5)、「New Entries」ウィンドウで、ビジネス・オブジェクト名、コネクター名、および統合ブローカー名を入力します。
4. 各ビジネス・オブジェクトについて、カウンター・フィールドに数値を入力します。ビジネス・オブジェクトとカウンターの組み合わせは、イベント分配表に対する固有キーを提供します。カウンターには、6 桁までの任意の数値を入力できます。

注: テスト環境では、複数のユーザーが、複数のコネクターによってサブスクライブされた同じビジネス・オブジェクトをテストする場合があります。そのビジネス・オブジェクトに対して各ユーザーが特定のイベントを必要とする場合には、コネクターと統合ブローカーのどの組み合わせに対してどのイベントを渡すか区別するためにユーザー名を指定することができます。「User (Event Trigger)」フィールドで、そのビジネス・オブジェクトに対して適切なユーザー名を入力してください。

イベント・フィルター操作のセットアップ

SAP アプリケーション内の構成表にすべての変更を収容することはできません。したがって、アダプターは、イベントにフィルターをかけるために変更できる ABAP 組み込みプログラムを提供しています。このプログラム /CWLD/TRIGGERING_RESTRICTIONS は、イベントに追加のフィルターを掛けることができるように、イベント・トリガー /CWLD/ADD_TO_QUEUE 内から呼び出されます。

注: 変更を行う場合は、コードの再コンパイルが必要になるため、開発者特権が必要になります。

組み込みプログラム /CWLD/TRIGGERING_RESTRICTIONS を表示または変更するには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。
2. 「Configuration」タブをクリックし、次に「Event Restriction」ボタンをクリックします。

イベント優先順位のセットアップ

イベントの重要性に基づいて、処理されるイベントの優先順位を設定できます。各ビジネス・オブジェクト、統合ブローカー、およびコネクターの組み合わせの優先

順位を設定することで、コネクターによるイベントの検索を遅らせることができます。例えば、あるイベントの優先順位を 10 に設定すると、コネクターはそのイベントを検索するまでに、イベント表を 10 回ポーリングします。そのため、コネクターがイベント表を 5 秒おきにポーリングする場合、コネクターはイベントを 50 秒後に選出します。コネクターがポーリングするたびに、優先順位の値は、そのイベントが検索され、処理されるまで、1 ずつ削減されます。

イベントの優先順位を設定するには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。
2. 「Configuration」タブをクリックし、次に「Event Distribution」ボタンをクリックします。
3. 該当するビジネス・オブジェクトに対して、「Priority」列に 1 から 99 の値を入力します。

ログ表スペース・サイズの拡張

コネクターのログ表は、デフォルトでは PSAPUSER1D という名前の表スペースにあり、索引は表スペース PSAPUSER1I にあります。PSAPUSER1D および PSAPUSER1I という SAP アプリケーション表スペースは、カスタマーが使用するために予約されており、一般には小容量です。このデフォルトのサイズのため、アクティビティーのレベルやアダプターのためにインストールされた SAP システムのロギング・レベルにもよりますが、これらの表スペースはすぐに満杯になってしまうことがあります。

これらの表スペースの現在のサイズを表示するには、トランザクション DB02 に移動し、「Current Sizes」ボタンをクリックします。これらの表スペースに必要なサイズは、WebSphere Business Integration システムでキャプチャーされるイベントのボリュームによって決まります。

デフォルトのサイズでは小さすぎる場合は、SAP データベース管理者に問い合わせてサイズを変更してください。

トランスポート・オブジェクトの数値範囲の検証

WebSphere Business Integration システム用として 4 つのオブジェクトがありますが、これらは、SAP アプリケーションの中に十分な数値範囲を持っていることが必要です。トランスポートをインストールすると、以下のオブジェクトとそれらのデフォルトの数値範囲が設定されます。

- /CWLD/EVT
- /CWLD/IDOC
- /CWLD/LOG
- /CWLD/OBJA

関連付けられている数値範囲が正しく設定されているかどうかを検証します。数値範囲を表示するには、以下の手順を行います。

1. トランザクション SNRO に移動します。
2. 「Object」フィールドにオブジェクト名を取り込みます (例えば、/CWLD/EVT)。
3. 「Number Ranges」をクリックし、次に「Intervals」をクリックします。

重要: 既にイベントが生成されているインストール済み環境で、トランスポート・ファイルを再インストールすると、既存のイベント ID を使用して新しいイベントが作成される場合があります。この問題を回避するには、IBM WebSphere BI Station の「Configuration」タブに移動してロギングをオフにし、ログを完全に切り捨ててから、コネクター・トランスポート・ファイルを再インポートします。コネクター・トランスポート・ファイルが正常にロードされたら、ロギングを元どおりオンにします。イベント・ログの切り捨ての詳細については、287 ページの『イベント・ログの切り捨てのセットアップ』を参照してください。

アダプターから提供される ABAP オブジェクトの変更

表、機能モジュール、プログラムなどすべてのアダプター・オブジェクトは、製品のネーム・スペース /CWLD/ を使用します。アダプターから提供される ABAP コードの変更が必要な場合には、IBM ソフトウェア・サポートに連絡して、変更キーを取得する必要があります。

イベントのピンポンの防止

ピンポンは、アプリケーションに対する要求の正常な実行が、そのアプリケーションでのイベントを起動し、イベントがイベント表に作成される場合に発生します。コネクターがイベント表をポーリングするように設定されている場合、コネクターは新しいイベントを選出し、元のソース・アプリケーションに返送しますが、それによってさらにイベントが起動されます。このソース・アプリケーションでの新しいイベントにより、同じことが繰り返されます。

注: 統合ブローカーが InterChange Server Express であるため、コラボレーション・ルックアップおよび相互参照マッピングにより、ソース・アプリケーション内部での重複レコードの発生を防止することができます。ただし、下記の追加処理は不要です。

コネクターとのピンポンを防止するには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。
2. 「Tools」タブをクリックし、次に「Config Objects」ボタンをクリックします。
3. 「New Entries」ボタンをクリックし、次の情報を入力します。

   構成名 — `Trigger: NoEventForUser`

   テキスト — `Prevent triggering for certain users`
4. IBM WebSphere BI Station の「Configuration」タブに戻り、「Configuration Values」ボタンをクリックします。
5. 「New Entries」ボタンをクリックし、イベントの起動を防止する各 *User Id* について、以下のエントリーを追加します。
   - 構成名 — `Trigger: NoEventForUser`
   - カウンター — *任意の数*
   - 構成値 — *ユーザー ID (コネクター名)*

注: これにより、コネクターによるイベントの起動が防止されることに注意する必要があります。

第 20 章 ABAP Extension Module でのビジネス・オブジェクトの処理

この章では、ABAP Extension Module のビジネス・オブジェクト処理について説明します。コネクターがビジネス・オブジェクトを処理する方法の詳細を示します。この章は、コネクターの Java コンポーネントおよび ABAP コンポーネントでのビジネス・オブジェクトの処理過程を示すことを目的としています。

この章の内容は以下のとおりです。

Adapter for mySAP.com の Extension Module のビジネス・オブジェクト処理は、特定のネイティブ SAP API が使用されているかどうかに関係なく、すべてのビジネス・オブジェクトについて同じです。例えば、呼び出しトランザクションまたは IDoc に基づいてビジネス・オブジェクトを開発する場合、ビジネス・オブジェクト・データは同じ方法で処理されます。この処理は、ビジネス・オブジェクトがイベント通知の一部として実行された検索として SAP アプリケーションに送信される場合でも、ビジネス・オブジェクト要求として送信される場合でも同じです。ビジネス・オブジェクトの動詞も処理を変更しません。

図 71 に、アプリケーション固有ビジネス・オブジェクトからフラット (階層なし) 構造への変換および処理と、アプリケーション固有ビジネス・オブジェクトへの逆方向の変換および処理を示します。SAP アプリケーションから渡されるビジネス・オブジェクト・データは、このアプリケーションに渡されるデータと同じ構造を持っている必要がありますが、値は異なっている可能性があります。

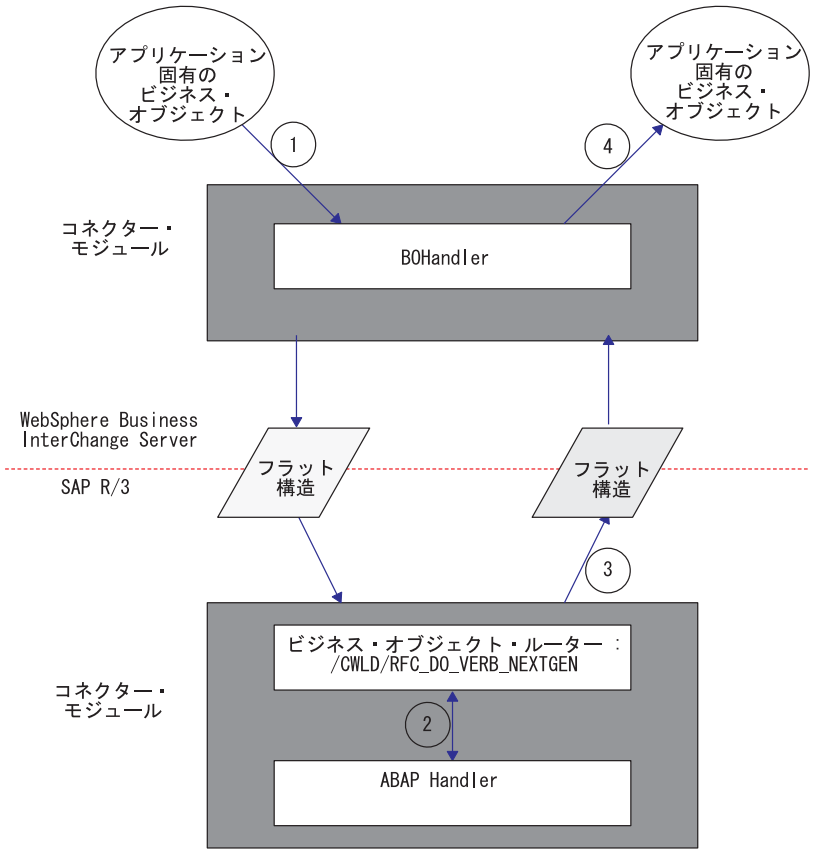


図 71. ビジネス・オブジェクトの処理

ビジネス・オブジェクト処理は、4 つのステップから構成されます。以下に示す 4 つのステップは、図 71 の番号に対応しています。

1. コネクターはアプリケーション固有ビジネス・オブジェクトを、ビジネス・オブジェクト・データを含んだフラットな構造に変換し、そのデータを SAP アプリケーションに渡します。
2. コネクターの機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN は、ビジネス・オブジェクト・データを動的に ABAP Handler に転送します。
3. ABAP Handler は、ビジネス・オブジェクト・データを処理し、ビジネス・オブジェクト応答データを生成し、/CWLD/RFC_DO_VERB_NEXTGEN を介して新規ビジネス・オブジェクト・データをコネクターに戻します。
4. コネクターは新しいビジネス・オブジェクト・データを受け取り、このデータと、アプリケーション固有ビジネス・オブジェクトのビジネス・オブジェクト定義を使用して新しいビジネス・オブジェクトを作成し、統合ブローカーに渡します。

ビジネス・オブジェクトのフラット構造への変換

コネクターは、ビジネス・オブジェクト処理の最初のステップとして、ビジネス・オブジェクトを、SAP アプリケーションで処理できるフラット (階層なし) 構造に変換します。フラットな構造の形式は、ビジネス・オブジェクトのすべてのタイプ (例えば呼び出しトランザクション・ベースまたは IDoc ベースのビジネス・オブジ

ェクト) で共通です。フラット構造は、アプリケーション固有ビジネス・オブジェクトから再フォーマットされたデータです。2 つの形式のデータの相違は、フラット構造が、ビジネス・オブジェクトの親子関係を保持していないことのみです。そのため、コネクターでは一連のルールに従ってフラットな構造を作成します。

コネクターは、ビジネス・オブジェクトをフラット構造に変換するとき、構造をメモリー内に作成し、ビジネス・オブジェクトからのデータをこの構造に組み入れます。それにより、コネクターはビジネス・オブジェクトから、以下に示すデータを SAP アプリケーションに渡します。

- ビジネス・オブジェクト名
- 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・オブジェクトの動詞
- ビジネス・オブジェクト動詞のアプリケーション固有情報
- 属性名
- 属性プロパティー `IsKey`
- 属性プロパティー `AppText`
- 属性値

表 42 に、ビジネス・オブジェクトの汎用フラット構造を示します。コネクターは、WebSphere ビジネス・オブジェクトからビジネス・オブジェクト・データを追加する際に、このフラットな構造を使用します。

表 42. SAP 用 WebSphere ビジネス・オブジェクトの汎用フラット構造表現

フィールド名	データ型	長さ	説明
ATTR_NAME	CHAR	32	属性名 (例えば CustomerId)
BLANK1	CHAR	1	区切り文字
ATTR_VALUE	CHAR	200	属性値 (例えば 00000103)
BLANK2	CHAR	1	区切り文字
ISKEY	CHAR	1	1= true、0 = false、属性のみ
BLANK3	CHAR	1	区切り文字
ISNEW	CHAR	1	1 = BO、0 = 動詞または属性
BLANK4	CHAR	1	区切り文字
PEERS	CHAR	6	ビジネス・オブジェクト配列のピア数を示します。
BLANK5	CHAR	1	区切り文字
OBJ_NUMBER	CHAR	6	未使用
BLANK6	CHAR	1	区切り文字
APPTEXT	CHAR	120	オブジェクト、動詞、または属性のアプリケーション固有情報
BLANK7	CHAR	1	区切り文字

注: BLANK*n* フィールド名は、常に単一の文字 (CHAR) スペースを含みます。また、値を取り込むことはできません。

データ変換が正しく実行されるためには、フラットな構造のビジネス・オブジェクト・データは一連のルールに厳格に従う必要があります。これらのルールは、この初期データ変換ステップで定義されます。

- 各ビジネス・オブジェクト属性はフラットな構造に、1 行が 1 つの属性に対応するように連続的に配置されます。

- 階層型のビジネス・オブジェクトは、最初に縦方向に変換され、次に横方向に変換されます。

コネクターは、フラットな構造にビジネス・オブジェクト・データを取り込む際に、トップレベルのビジネス・オブジェクトを開始位置として、各ビジネス・オブジェクトを 2 回通してループします。

1. 最初のパスでは、コネクターはすべての単純な属性を設定します。各属性は、フラットな構造の 1 行に相当します。
2. 2 回目のパスでは、それぞれの子ビジネス・オブジェクトについて、ステップ 1 と同じ処理を再帰的に実行します。

子ビジネス・オブジェクトを表す属性は親に含まれません。その代わりに、データを格納している各子ビジネス・オブジェクトは、完全なビジネス・オブジェクトとして作成されています。この結果、最初に縦方向、次に横方向の順に並べられた、属性の単一のリストができあがります。

図 72 に、SAP 用 WebSphere ビジネス・オブジェクトからフラット・データ構造へのデータ変換を示します。データの変換は、最初に縦方向、次に横方向というルールに常に従って行われます。この例では、トップレベルの親ビジネス・オブジェクト SAP_Order には、SAP_LineItem (1) および SAP_LineItem (2) という 2 つの子があり、これらはピアと見なされます。SAP_LineItem (1) には、SAP_ScheduleLines という 1 つの子ビジネス・オブジェクトがあります。

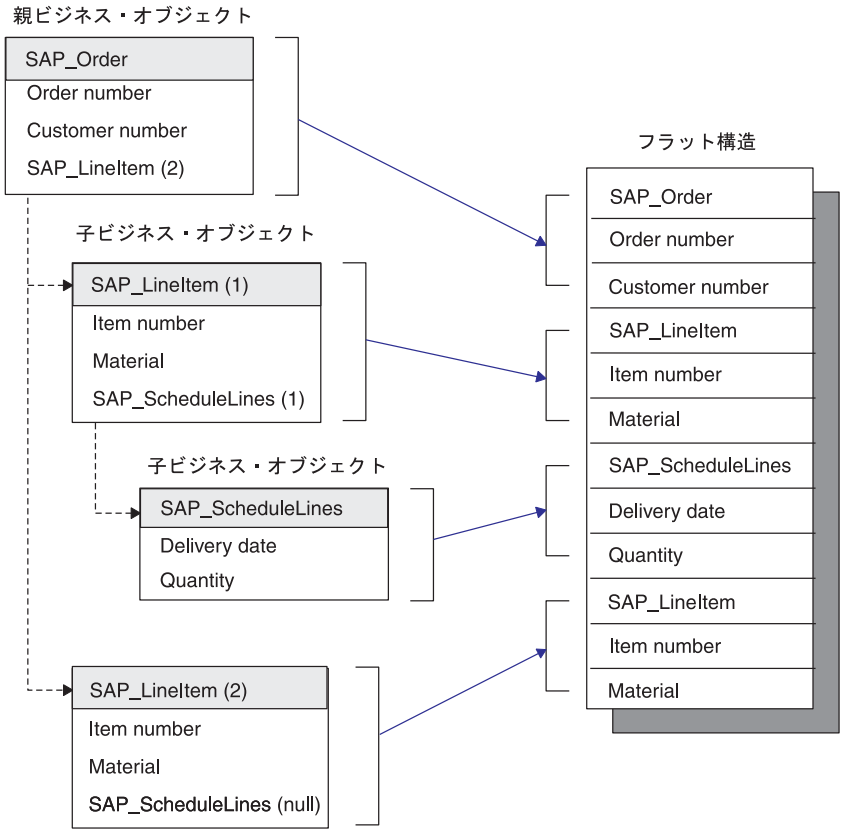


図 72. ビジネス・オブジェクトからフラットな構造への変換

ビジネス・オブジェクト定義を設計するときには、ビジネス・オブジェクトの順序および属性を把握することが重要です。次の表に、WebSphere ビジネス・オブジェクトからフラット構造への変換の結果を示します。 表 43 に、フラット・ビジネス・オブジェクト SAP_Material (キー値は ItemID) に対応するフラット構造を示します。この例では、ビジネス・オブジェクトや属性に、アプリケーション固有情報はありません。 表 44 に、IDoc Sales Order に基づく階層型ビジネス・オブジェクトのフラット構造を示します。

表 43. フラットなビジネス・オブジェクト SAP_Material

ATTR_NAME	ATTR_VALUE	ISKEY	ISNEW	PEERS	OBJ_ NUMBER	APPTEXT
BoName	SAP_Material	0	1	1	(ブランク)	(ブランク)
BoVerb	Retrieve	0	0	1	(ブランク)	:/CWLD /DYNAMIC_RETRIEVE
ItemID	000000000000001179	1	0	1	(ブランク)	(ブランク)
ShortDesc	CxIgnore	0	0	1	(ブランク)	(ブランク)
ObjectEventID	SAP_124	0	0	1	(ブランク)	(ブランク)

この例では、ビジネス・オブジェクトや属性に、アプリケーション固有情報はありません。

表 44. IDoc Sales Order に基づく階層型ビジネス・オブジェクト

ATTR_NAME	ATTR_VALUE	ISKEY	ISNEW	PEERS	OBJ_ NUMBER	APPTEXT
BoName	SAP_Order	0	1	1	(ブランク)	YXRV4B01
BoVerb	Create	0	0	1	(ブランク)	[*archive:methods*]
Currency	USD	0	0	1	(ブランク)	E1EDK01:CURCY
OrderId	CxIgnore	1	0	1	(ブランク)	E1EDK01:BELNR
ObjectEventId	SAP_124	0	0	1	(ブランク)	E1EDK01: ObjectEventId
BoName	SAP_LineItem	0	1	2	(ブランク)	Z1XRV40
BoVerb	Create	0	0	2	(ブランク)	(ブランク)
Createdby	User1	1		2	(ブランク)	Z1XRV40:ERNAM
ObjectEventId	SAP_125	0	0	2	(ブランク)	Z1XRV40: ObjectEventId
BoName	SAP_ScheduleLines	0	1	1	(ブランク)	E1EDK14
BoVerb	Create	0	0	1	(ブランク)	(ブランク)
Qualifier	001	1	0	1	(ブランク)	Z1XRV40:QUALF
OrganizationId	1000	0	0	1	(ブランク)	E1EDK14:ORGID
ObjectEventId	SAP_126	0	0	1	(ブランク)	E1EDK14: ObjectEventId
BoName	SAP_LineItem	0	1	2	(ブランク)	Z1XRV40
BoVerb	Create	0	0	2	(ブランク)	(ブランク)
Createdby	User1	1	0	2	(ブランク)	Z1XRV40:ERNAM
ObjectEventId	SAP_127	0	0	2	(ブランク)	Z1XRV40: ObjectEventId

最初の 2 つの行 BoName および BoVerb が各ビジネス・オブジェクトについてコネクターにより追加されています。BoName と BoVerb は、ビジネス・オブジェクト属性として使用できないキーワードです。

ABAP Handler へのビジネス・オブジェクト・データ発送

ビジネス・オブジェクト・データは、フラットな構造に変換されると、アダプターの ABAP 機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN が呼び出されることにより、SAP メモリーに渡されます。/CWLD/RFC_DO_VERB_NEXTGEN は、ビジネス・オブジェクト・データを操作せず、後続の処理を行う適切な ABAP Handler に発送するのみです。/CWLD/RFC_DO_VERB_NEXTGEN は、ビジネス・オブジェクト・データを ABAP Handler に渡すと、ビジネス・オブジェクト・データが戻されるまで待機します。

注: すべてのビジネス・オブジェクト検索および要求は、/CWLD/RFC_DO_VERB_NEXTGEN を通じて処理されることに注意してください。

/CWLD/RFC_DO_VERB_NEXTGEN は、ビジネス・オブジェクトの動詞のアプリケーション固有情報を使用して、どの ABAP Handler がビジネス・オブジェクト・データを処理するかを決定します。実行時には、/CWLD/RFC_DO_VERB_NEXTGEN は動詞のアプリケーション固有情報を読み取り、指定された ABAP Handler にビジネス・オブジェクト・データを渡します。

すべての ABAP Handler は、動詞アプリケーション固有情報の使用をコネクターに対して予約する必要があります。動詞のアプリケーション固有情報の書式は、次のとおりです。

:function1:function2:function3

ここで、/CWLD/RFC_DO_VERB_NEXTGEN は *function1* を実行し、*function2* と *function3* をパラメーターとして渡します。例えば、Customer Update および Material Retrieve では、function1 のみを実行します。

Create、Update、および Delete 動詞では、:/CWLD/RFC_DYNAMIC_TRANSACTION を指定します。

Retrieve 動詞では、:/CWLD/RFC_DYNAMIC_RETRIEVE: を指定します。

アダプターが提供する ABAP Handler の 1 つに、機能モジュール /CWLD/IDOC_HANDLER があります。この ABAP Handler は、フラットな構造のデータを IDoc 定義のインスタンスに再フォーマットし、再フォーマットされたデータを、IDoc のその特定のタイプを処理するように記述された別の ABAP Handler に渡します。IDoc Handler API の使用方法の例を以下に示します。

Sales Order Update = :/CWLD/IDOC_HANDLER:Y_XR_ORDER_C2

Sales Order Retrieve = :/CWLD/IDOC_HANDLER:Y_XR_ORDER_C4

この例では、/CWLD/IDOC_HANDLER が実行され、ビジネス・オブジェクト・データとともに第 2 の機能モジュール名を渡します。/CWLD/IDOC_HANDLER は第 2 の ABAP Handler への呼び出しを実行し、IDoc 形式のビジネス・オブジェクト・データを、Order オブジェクトを処理するように記述された Y_XR_ORDER_C2 または Y_XR_ORDER_C4 機能モジュールに渡します。

注: /CWLD/RFC_DO_VERB_NEXTGEN は、*function1* の値のみを使用します。*function2* と *function3* は、ABAP Handler により使用されることがあります。

ABAP Handler を動的に呼び出すために、/CWLD/RFC_DO_VERB_NEXTGEN は、すべての ABAP Handler のインターフェースが正確に同じであることを必要とします。これにより、/CWLD/RFC_DO_VERB_NEXTGEN は任意の ABAP Handler との間で、戻りコードや戻りテキスト・メッセージとともに、ビジネス・オブジェクト・データを送受信できます。機能モジュール・インターフェースの詳細については、250 ページの『IBM WebSphere 機能モジュール・インターフェース』を参照してください。

ABAP Handler によるビジネス・オブジェクト・データの処理

ABAP Handler の機能は、SAP アプリケーション・データベースとの間でビジネス・オブジェクト・データを出し入れすることです。ビジネス・オブジェクト・データを処理する際に、ABAP Handler は以下の処理を行います。

1. ビジネス・オブジェクト・データを解釈します。
2. データを SAP ネイティブ API と統合します。
3. ネイティブ API から戻されたすべてのデータを再フォーマットします。

ビジネス・オブジェクト・データおよび ABAP Handler

すべての ABAP Handler は、ビジネス・オブジェクト・データを同じ形式 (フラットな構造) で受け取ります。しかし、それぞれの ABAP Handler はビジネス・オブジェクトについて、WebSphere ビジネス・オブジェクト定義の複雑さ、SAP が提供するネイティブ API、および ABAP Handler が提供する機能のレベルによって決定される固有の要件を持っています。これらの理由から、ABAP Handler は、そのビジネス・オブジェクトに固有の構造に構文解析することで、ビジネス・オブジェクト・データを解釈することができます。これにより、ABAP Handler はより容易にデータを操作できます。

注: データの構文解析は必須ではありません。しかし、これを行うことで、ABAP Handler でのビジネス・オブジェクトの処理が単純化されます。

アダプターでは、IDoc Handler などのいくつかの ABAP Handler が提供されます。IDoc Handler は、ビジネス・オブジェクト・データを ABAP Handler で使用される IDoc ベースの構造に再フォーマットすることで解釈する ABAP Handler を提供することによって、SAP の IDoc テクノロジーを強化しています。

ビジネス・オブジェクト・データおよび SAP ネイティブ API

ABAP Handler は、ビジネス・オブジェクト・データを解釈したら、そのデータを SAP アプリケーション・データベースに統合する必要があります。Handler は、呼び出しトランザクションや ABAP SQL などの SAP ネイティブ API を使用して、アプリケーション・データベースとの間でデータを出し入れするには、ビジネス・オブジェクト・データを操作する必要があります。

Create、Update および Delete の処理

Create、Update、または Delete 操作の目的は、SAP アプリケーション・データベースを変更することです。特定のビジネス・オブジェクトの SAP アプリケーション・データベース・スキーマがデータの構造を定義するのに対して、そのデータを

変更する SAP 提供のトランザクションは、はるかに幅広い作用範囲を持っています。その結果、SAP アプリケーションのアプリケーション・データベース表を直接変更することは、アプリケーションのデータ保全性に対して危険な結果をもたらす可能性があります。

SAP では、データベース表を直接変更する代わりとして、Create、Update、および Delete 操作用の柔軟な ABAP API (呼び出しトランザクション) を提供しています。呼び出しトランザクションは、SAP アプリケーションにデータを入力するための SAP 提供の機能です。この機能は、オンライン・ユーザーがトランザクションで使用するのと同じ画面を使用することで、データが SAP のデータ・モデルに準拠することを保証します。この処理は、一般に screen scraping (画面を通した情報のやり取り) と呼ばれます。

Retrieve 処理

動詞が Retrieve の場合、コネクターは ABAP SQL ステートメントを使用して、SAP アプリケーション・データベースからデータを検索します。ビジネス・オブジェクト・データは、データをプルする際に、where 文節のキーを提供します。この方法によるデータ検索の問題点は、検索されたデータがビジネス・オブジェクト構造を表す形式で表される必要がある点です。この処理は、ABAP Handler の ABAP コードで行われます。

コネクターは、基本キーが指定されている場合にのみ、Retrieve 処理をサポートします。

戻りコードおよび戻されたビジネス・オブジェクト・データ

ビジネス・オブジェクトの動詞に関係なく、コネクターは次の 2 種類の確認を待機します。

- 戻りコード
- 戻されたビジネス・オブジェクト・データ (成功時のみ、戻りコード = 0)

ABAP Extension Module では、ビジネス・オブジェクト・データを処理する際に、0、21、および任意のゼロ以外のコード (21 以外) という 3 種類の戻りコードを使用します。戻りコードは、機能モジュール・インターフェースで設定します。機能モジュール・インターフェースの詳細については、250 ページの『IBM WebSphere 機能モジュール・インターフェース』を参照してください。

戻りコード 0

戻りコード 0 は、ビジネス・オブジェクトが正常に処理され、コネクター・インフラストラクチャーに VALCHANGE が戻ったことを示します。ABAP Handler での処理が正常に終了した場合、コネクターは、実行された操作を反映した新しいビジネス・オブジェクト・データを予期します。例えば、Create 処理が正常に終了すると、戻されるビジネス・オブジェクトは、キーが更新されていることを除いて、始めに渡されたビジネス・オブジェクトの正確なコピーです。同様に、Retrieve 処理が正常に終了すると、そのビジネス・オブジェクトの完全にフォーマットされたインスタンスが戻されます。ただし、Create、Update、および Delete 操作は、戻されるビジネス・オブジェクトについて、検索操作とは異なる要件を持っています。

統合ブローカーが IBM InterChange Server Express であるため、要件の違いは、WebSphere Business Integration システムによるビジネス・オブジェクトの処理方法、特にマッピング中のオブジェクト ID の動的相互参照の方法によって生じます。Create または Update 操作の後でコネクターが InterChange Server Express にビジネス・オブジェクトを戻す場合、マッピング・インフラストラクチャーは、相互参照表を新しく取得したオブジェクト ID で更新します。これは、ビジネス・オブジェクトが最初にコネクターに送信されたときに設定された、ビジネス・オブジェクトの ObjectEventId 属性の値をルックアップすることで行われます。

ABAP Handler は、オブジェクト ID をコネクターに戻ったビジネス・オブジェクトに「添付」する任務があるため、このことは ABAP Handler にとっては重要です。Retrieve 操作では、これに対応する動的な相互参照が行われないため、これは通常、Retrieve 操作には関係ありません。Retrieve 操作では、まったく新しいビジネス・オブジェクトが生成され、コネクターに戻されます。このビジネス・オブジェクトは、始めに送信されたビジネス・オブジェクト構造とは、直接的な関係はありません。

ABAP Handler から戻されるビジネス・オブジェクト・データは、そのデータが始めに機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN に渡されたときと同じフラットな構造形式である必要があります。ABAP Handler は、次の情報を持った単純なタイプ属性のみを送信する必要があります。

- 値
- ピア関係
- アプリケーション固有の情報

コネクターがこのデータからビジネス・オブジェクトを作成する場合にはアプリケーション固有情報のみを使用するため、この時点では属性名は必要ありません。ビジネス・オブジェクトまたはオブジェクト・タイプ属性の先頭と末尾を示す ID は使用されません。また、追加してはいけません。例えば、BoName および BoVerb 行は、ABAP Handler から戻されるビジネス・オブジェクトでは使用されません。これらは、処理を容易にするためにのみ、始めに ABAP Handler に渡されます。

ABAP Handler は、フラットな構造に WebSphere ビジネス・オブジェクトを表すビジネス・オブジェクト応答データを取り込む場合、次の一連のルールに従う必要があります。

- オブジェクト・タイプではなく、単純な属性のみを送信します。
- すべての属性は、WebSphere ビジネス・オブジェクト定義内に存在する必要があります。
- すべての属性は、WebSphere ビジネス・オブジェクト定義にリストされている順序で送信される必要があります。
- 子ビジネス・オブジェクトの属性は、その親ビジネス・オブジェクトについて少なくとも 1 つの属性が送信されなければ送信できません。
- 包含されているビジネス・オブジェクトは、それが持っているピアの数を通知する必要があります。
- 属性名 (フィールド ATTR_NAME) は必須ではありません。

図 73 に、フラットなビジネス・オブジェクト (オブジェクト・タイプ属性を含まない) を示します。

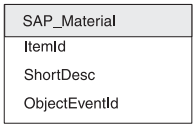


図 73. フラットなビジネス・オブジェクト SAP_Material

表 45 に、フラットなビジネス・オブジェクト SAP_Material (キー値は ItemID) の構造を示します。ATTR_NAME フィールドは必須でないこと、APPTEXT は各属性について固有であること、およびこのビジネス・オブジェクトがフラットであるため、PEERS フィールドはブランクのままにしてもよいことに注意する必要があります。

表 45. フラットなビジネス・オブジェクト SAP_Material

ATTR_NAME	ATTR_VALUE	ISKEY	ISNEW	PEERS	OBJ_ NUMBER	APPTEXT
(ブランク)	000000000000001179	(ブランク)	(ブランク)	(ブランク)	(ブランク)	ItemId
(ブランク)	Toaster 6000	(ブランク)	(ブランク)	(ブランク)	(ブランク)	ShortDesc
(ブランク)	SAP_124	(ブランク)	(ブランク)	(ブランク)	(ブランク)	ObjectEventId

図 74 に、階層型ビジネス・オブジェクト (オブジェクト・タイプを含む) を示します。

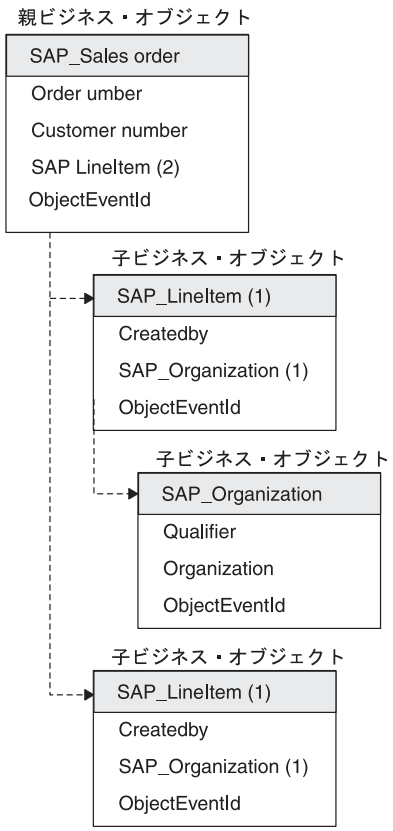


図 74. 階層型ビジネス・オブジェクト SAP sales order (IDoc)

表 46 に、IDoc Sales Order に基づいた階層型ビジネス・オブジェクトのフラットな構造の表現を示します。ATTR_NAME フィールドは必須でないこと、APPTEXT は各属性について固有であること、およびこのビジネス・オブジェクトが階層型であるため、PEERS フィールドに適切な関係がリストされていることに注意する必要があります。

表 46. IDoc Sales Order に基づく階層型ビジネス・オブジェクト

ATTR_NAME	ATTR_VALUE	ISKEY	ISNEW	PEERS	OBJ_ NUMBER	APPTEXT
(ブランク)	USD	0	0	1	(ブランク)	E1EDK01:CURCY
(ブランク)	0000000101	0	0	1	(ブランク)	E1EDK01:BELNR
(ブランク)	SAP_124	0	0	1	(ブランク)	E1EDK01: ObjectEventId
(ブランク)	User1	0	0	2	(ブランク)	Z1XRV40:ERNAM
(ブランク)	SAP_125	0	0	2	(ブランク)	Z1XRV40: ObjectEventId
(ブランク)	001	0	0	1	(ブランク)	Z1XRV40:QUALF
(ブランク)	1000	0	0	1	(ブランク)	E1EDK14:ORGID
(ブランク)	SAP_126	0	0	1	(ブランク)	E1EDK14: ObjectEventId
(ブランク)	User1	0	0	2	(ブランク)	Z1XRV40:ERNAM
(ブランク)	SAP_127	0	0	2	(ブランク)	Z1XRV40: ObjectEventId

戻りコード 21

戻りコード 21 は、ビジネス・オブジェクトが正常に処理され、コネクター・インフラストラクチャーに SUCCESS が戻ったことを示します。このコードは成功のみをコネクターに戻し、ビジネス・オブジェクト・データを戻すことはありません。ビジネス・オブジェクト・データを処理するオブジェクト固有の IDoc Handler は、そのビジネス・オブジェクト・データが正常に SAP アプリケーションに入力されると、戻りコードとして 21 を戻します。この戻りコードは /CWLD/RFC_DO_VERB_NEXTGEN 機能モジュールに渡され、このモジュールが成功をコネクターに戻します。コネクターがビジネス・オブジェクト・データを受け取ることはありません。

このことは、大きなオブジェクト (複数の行項目を持つ IDoc など) を渡すときに有用です。ビジネス・オブジェクト・データが正常に SAP アプリケーションに渡されたことを確認するだけで十分です。コードのみを戻せばよく、ビジネス・オブジェクトを戻す必要がないので、パフォーマンスが大幅に向上します。

統合ブローカーが InterChange Server Express であるため、ビジネス・オブジェクトに相互参照が必要でなく、単にデータを SAP アプリケーションに渡している場合にだけ、戻りコード 21 を使用してください。検索操作では、戻りコード 21 は使用しないでください。SUCCESS 戻りコードの動作は、相互参照や後続の処理のために InterChange Server Express に戻されるビジネス・オブジェクトが存在しないことを意味しています。

ゼロ以外の戻りコード

ゼロ以外の戻りコード (21 以外) は、オブジェクトが正常に処理されず、コネクターに FAIL を戻すことを示します。ABAP Handler がゼロ以外のコード (21 以外) を戻した場合、ビジネス・オブジェクトはコネクターに戻されません。

フラット構造のビジネス・オブジェクトへの変換

フラットな構造に新しいビジネス・オブジェクト・データが再び取り込まれると、/CWLD/RFC_DO_VERB_NEXTGEN はそのビジネス・オブジェクト・データを呼び出し側のコネクターに戻します。コネクターは単一スレッドのため、一度に 1 つのビジネス・オブジェクトのみを渡すことに注意してください。コネクターはこの時点で、ビジネス・オブジェクト・データをフラットな構造からビジネス・オブジェクトに変換する必要があります。フラットな構造のデータをビジネス・オブジェクトへと処理する場合、コネクターは以下の処理を行う必要があります。

1. 元のビジネス・オブジェクトを初期化します。
2. ビジネス・オブジェクト・データをフラットな構造からビジネス・オブジェクトに転送します。
3. ビジネス・オブジェクトを InterChange Server Express (コネクター・コントローラー) にデリバリーします。

ビジネス・オブジェクトの初期化

コネクターは、統合ブローカーから受け取った元のビジネス・オブジェクトを、値を取り込む前に初期化します。ビジネス・オブジェクトを初期化する際、コネクターはトップレベル・ビジネス・オブジェクトのすべての属性をヌルに設定します。

オブジェクト・タイプ属性については、このアクションは包含されているすべてのビジネス・オブジェクトを再帰的に削除し、トップレベル・ビジネス・オブジェクトのみが残されます。

コネクターによるビジネス・オブジェクトの再ビルドの仕方

コネクターが元のビジネス・オブジェクトを初期化した後には、ビジネス・オブジェクト名とビジネス・オブジェクト動詞を含んだトップレベル・ビジネス・オブジェクトのみが残され、属性値データは残されません。属性値データは、ABAP Handler からのフラットな構造から転送される必要があります。戻されたデータを転送するためのロジックは単純ですが、データはコネクターが予期する正確な順序で転送される必要があります。

コネクターは、戻されたデータ内のアプリケーション固有情報を、ビジネス・オブジェクト定義にある属性のアプリケーション固有情報と突き合わせます。コネクターは、戻されたビジネス・オブジェクト・データにあるすべての属性を設定します。どのような属性も設定されていないと、コネクターはそのコネクター・インフラストラクチャーに `FAIL` を戻します。

戻されたデータの転送を正常に実行するために、コネクターは戻されたデータについて、以下の項目が真であることを予期します。

- 単純な属性のみを含んでおり、1 行が 1 つの属性に対応する。
- 属性は、WebSphere ビジネス・オブジェクト定義内に存在する必要がある。
- 属性は、WebSphere ビジネス・オブジェクト定義と同じ順序で配列されている必要がある (最初に縦方向、次に横方向)。
- 属性のアプリケーション固有情報により、そのオブジェクトのアプリケーション固有情報が、ビジネス・オブジェクト定義内の属性を一意的に識別している別の値とリンクしている。
- 子の属性は、その親オブジェクトの属性の後 (親の前ではなく、祖父母の後でもなく) に出現する必要がある。
- 属性は、そのビジネス・オブジェクトのピアの数を通知する必要がある。

コネクターは、アプリケーション固有ビジネス・オブジェクトを再作成する際に、トップレベルのビジネス・オブジェクトを開始位置として、そのビジネス・オブジェクトを 2 回通してループします。

1. 最初のパスでは、コネクターはすべての単純な属性を設定します。
2. 2 回目のパスでは、子オブジェクト内にフラットな属性が存在するかどうかを検査します。存在する場合、コネクターは子オブジェクトに対して同じ処理を再帰的に実行します。

重要: フラットな構造をビジネス・オブジェクトへ変換するのに失敗すると、コネクターは失敗を統合ブローカーに報告します。しかし、データは既に SAP アプリケーション内に送られているため、この段階でロールバックすることはできません。ルールは単純ですが、多数の属性を持つ、複雑な階層型ビジネス・オブジェクトを実装することは管理するのが困難である可能性があります。

ビジネス・オブジェクトが新しいビジネス・オブジェクト・データを使用して正常に再作成されると、コネクターはそのオブジェクトを統合ブローカーに戻します。

第 21 章 ABAP Extension Module のビジネス・オブジェクトの開発

この章では、ABAP Extension Module のビジネス・オブジェクトの開発について説明します。また、ビジネス・オブジェクトおよび ABAP Handler を開発するためのステップや、背景情報を提供します。本書の読者は、コネクターがビジネス・オブジェクトを処理する方法について、十分な知識と経験を持っている必要があります。

この章の内容は以下のとおりです。

背景情報

ABAP Extension Module 用のビジネス・オブジェクトの開発は、アプリケーション固有ビジネス・オブジェクト定義と、それに関連付けられた、サポートする必要のある各動詞用の ABAP Handler の作成から成り立っています。

アプリケーション固有のビジネス・オブジェクトを開発するには、ビジネスのニーズをサポートするビジネス・オブジェクト定義を作成する必要があります。Adapter for mySAP.com では、SAP アプリケーション内でのビジネス・オブジェクト定義の開発過程を容易にするツールを提供します。ABAP Extension Module のビジネス・オブジェクト定義は、Business Object Designer Express またはテキスト・エディターを使用して作成することもできますが、最初はアダプターのビジネス・オブジェクト開発ツールを使用することを推奨します。これらのツールでは、SAP アプリケーションのネイティブ定義をテンプレートとして使用します。

開発する各アプリケーション固有のビジネス・オブジェクト定義は、アダプター提供の ABAP Handler を使用するか、またはカスタム ABAP Handler を開発してサポートする必要があります。ABAP Handler は、SAP アプリケーション・データベースとの間でデータを出し入れする機構です。

注: アプリケーション固有のビジネス・オブジェクトおよび ABAP Handler は、お互いの整合性に依拠して SAP アプリケーションとの間でデータを受け渡しします。このため、ビジネス・オブジェクト定義を変更すると、ABAP Handler も変更する必要があります。

コネクター用の ABAP Handler は ABAP 機能モジュールとして実装されています。ABAP Handler は、ビジネス・オブジェクト・ルーター `/CWLD/RFC_DO_VERB_NEXTGEN` からのビジネス・オブジェクト要求を実行するために

共同して動作する、1 つ以上の機能モジュールです。ABAP Handler は、ビジネス・オブジェクト・データを SAP アプリケーションとの間で出し入れします。

注: SAP は、WebSphere Business Integration システムがサポートする動詞 (Create、Retrieve、Update、および Delete) のほかにも多くの動詞をサポートします。ABAP Handler は、任意の動詞をサポートするように開発できます。

ABAP Handler を開発するには、コネクターが SAP アプリケーションとの間でどのようにデータを出し入れするか、およびこのプロセス中にデータがどのような形式になるかについて理解する必要があります。ビジネス・オブジェクト処理についての高度な説明は、207 ページの『第 18 章 ABAP Extension Module の概要』を参照してください。ビジネス・オブジェクト処理の詳細については、233 ページの『第 20 章 ABAP Extension Module でのビジネス・オブジェクトの処理』を参照してください。

注: ビジネス・オブジェクトを開発する際には、そのオブジェクトを SAP アプリケーションにあるコネクターの表の `/CWLD/OBJECTS` に追加する必要があります。追加しないと、オブジェクトをカスタマイズ (例えばオブジェクトをイベント分配用にセットアップする場合) する際にアクセスできなくなります。

SAP ネイティブ API

アダプターの提供する ABAP Handler は、SAP ネイティブ API を使用することで、SAP アプリケーションとの間でデータを出し入れできます。WebSphere Business Integration システムには、以下のネイティブ API が実装されています。

- 『ABAP SQL』
- 249 ページの『呼び出しトランザクション』
- 249 ページの『バッチ・データ通信 (BDC)』
- 250 ページの『ビジネス・アプリケーション・プログラミング・インターフェース (BAPI)』

ABAP SQL

ABAP SQL は、SQL の SAP 所有バージョンです。これはデータベースおよびプラットフォームに非依存なため、どのような SQL コードを記述しても、SAP がサポートするデータベースとプラットフォームの任意の組み合わせの上で実行できます。ABAP SQL は SQL のほかのバージョンと同様の構文を持っており、update、insert、modify、select、delete などの基本的なデータベース表コマンドをすべてサポートします。ABAP SQL とその使用方法、構文、および機能についての完全な説明は、SAP の資料を参照してください。

ABAP SQL を使用すると、ABAP Handler は作成、更新、および削除操作のために、ビジネス・オブジェクト・データを持つ SAP データベース表を変更できます。ABAP の SELECT ステートメントの `where` 文節内のビジネス・オブジェクト・データをキーとして使用することもできます。

注: WebSphere Business Integration システムでは、データベースの整合性を破壊する可能性があるため、ABAP SQL を使用して SAP 表を変更することはありません。コネクターでは、ABAP SQL はデータを検索する場合、およびアダプター提供のデータベース表を変更する場合にのみ使用されます。

呼び出しトランザクション

呼び出しトランザクションは、SAP システムにデータを入力するための SAP 提供の機能です。呼び出しトランザクションは、トランザクション時にオンライン・ユーザーに対して表示されるのと同じ画面を使用することで、データが SAP のデータ・モデルに準拠することを保証します。この処理は、一般に screen scraping (画面を通した情報のやり取り) と呼ばれます。呼び出しトランザクションを使用するには、次のタイプの命令を指定します。

- Initiation — 呼び出すトランザクション
- Navigation — 処理する画面の順序
- Mapping — 画面上の各フィールドに入力する入力データ

Initiation は、「呼び出しトランザクション」呼び出しでは、単一値パラメーターとして渡されます。Navigation および Mapping 命令は、表に特定の形式で同時に渡されます。この形式は、任意の SAP トランザクションについて呼び出しトランザクションを呼び出すために使用できます。この形式では、これらの命令は BDC データ、BDC 表、または BDC セッションと呼ばれます。

バッチ・データ通信 (BDC)

バッチ・データ通信 (Batch Data Communication、BDC) は、ユーザー介入なしにトランザクションを実行するために SAP が従うことのできる命令セットです。これらの命令は、トランザクションの画面が処理される順序や、どの画面のどのフィールドにデータを取り込むかを指示します。オンライン・ユーザーに対して公開される SAP トランザクションのすべての要素は、BDC で使用できる識別情報を持っています。それらの要素を次に示します。

- 画面 —プログラム名および画面番号によって識別されます。
- 入力フィールド — 通常は、それが参照するデータベース表およびフィールド名によって識別されます。
- トランザクション内のコマンド — save、new items、details、および exit などのコマンド (1 文字から 8 文字のコードで識別されます)。

画面の BDC 識別情報を取得するには、画面上の任意のフィールドにカーソルを置きます。F1 キーを押してヘルプを表示します。次に F9 キーを押すと、技術情報が表示されます。プログラム名および画面番号は、「Screen Data」の下にリストされています。

入力フィールドの BDC 識別情報を取得するには、画面上でデータを入力する各フィールドにカーソルを置きます。F1 キーを押してヘルプを表示します。次に F9 キーを押すと、技術情報が表示されます。「Field Description for Batch Input」というボックスが存在する場合は、「Screen Field」フィールドの情報を使用します。このボックスが存在しない場合は、「Field Data」ボックスの「Table Name」と「Field Name」の情報をハイフンで連結します。

コマンドの BDC 識別情報を取得するには、メニューでそのコマンドを強調表示し、F1 キーを押してヘルプを表示します。「Function」フィールドの値を使用します。

ビジネス・アプリケーション・プログラミング・インターフェース (BAPI)

BAPI Module は、BAPI をサポートするために使用します。詳細については、 89 ページの『第 6 章 BAPI Module の概要』を参照してください。

IBM WebSphere 機能モジュール・インターフェース

すべての ABAP Handler は、同じ機能モジュール・インターフェースを実装する必要があります。この機能モジュール・インターフェースは、ビジネス・オブジェクト・ルーター /CWLD/RFC_DO_VERB_NEXTGEN が ABAP Handler との間でビジネス・オブジェクト・データを受け渡しできることを保証します。インターフェースは次のとおりです。

```
*"*"Local interface:
*"  IMPORTING
*"        VALUE(PROC_FUNC_1) LIKE  RS38L-NAME OPTIONAL
*"        VALUE(PROC_FUNC_2) LIKE  RS38L-NAME OPTIONAL
*"        VALUE(OBJECT_NAME) LIKE  /CWLD/LOG_HEADER-OBJ_NAME OPTIONAL
*"        VALUE(OBJECT_VERB) LIKE /CWLD/WIZ_IN-OBJ_VERB OPTIONAL
*"        VALUE(ARCHIVE) OPTIONAL
*"        VALUE(TEXT) LIKE  T100-TEXT OPTIONAL
*"  EXPORTING
*"        VALUE(RETURN_TEXT) LIKE /CWLD/LOG_HEADER-OBJ_KEY
*"        VALUE(RFCRC) LIKE /CWLD/RFCRC_STRU-RFCRC
*"  TABLES
*"        RFC_STRUCTURE STRUCTURE /CWLD/OBJ_STRU
*"  EXCEPTIONS
*"        NOT_FOUND
*"        ERROR_PROCESSING
```

インターフェースのインポート・セクションでは、ABAP Handler 名、ビジネス・オブジェクト名、およびビジネス・オブジェクトなどの値を通知できます。

インターフェースのエクスポート・セクションは、ABAP Handler の処理の結果を通知するために使用します。戻りコードである RFCRC パラメーターは、コネクターが戻すコードを判別するために使用される単一フィールドです。可能な値は次のとおりです。

RC = 0 (正常、VALCHANGE)

RC = 1 (失敗、FAIL)

RC = 21 (正常、SUCCESS)

RETURN_TEXT パラメーターは 120 文字のフリー・テキスト・フィールドで、コネクターによって書き込まれたり、また戻り状況記述子にエラー・メッセージとして記録されたりします。ABAP Handler がこのパラメーターの値を提供しない場合は、/CWLD/RFC_DO_VERB_NEXTGEN が戻りコードに応じてデフォルトのテキストを提供します。

注: インターフェースの例外セクションでは、2 つの例外が定義されています。その代わりに、エクスポート・パラメーターを使用することをお勧めします。

IBM WebSphere ABAP Handler API

アダプターは、SAP 用の WebSphere ビジネス・オブジェクトのための ABAP Handler 開発に役立ついくつかの API を提供しています。これらの API は、任意のタイプの追加ビジネス・オブジェクトをサポートするためにメタデータのみを必要とするため、「汎用の」ABAP Handler として開発されています。アダプターは、以下の ABAP Handler API を提供します。

- 動的検査 — `/CWLD/DYNAMIC_RETRIEVE`
- 動的トランザクション — `/CWLD/DYNAMIC_TRANSACTION`
- IDoc Handler — `/CWLD/IDOC_HANDLER`

アダプターでは、これらの API をサポートする一連のツールを提供しています。この 3 つの ABAP Handler API のツールは、すべて IBM WebSphere BI Station にあります (トランザクション `/n/CWLD/HOME`)。詳細については、353 ページの『付録 D. IBM WebSphere BI Station のサポート・レベル』を参照してください。アダプターは SAPODA も提供しています。詳細については、35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

以下のセクションでは、アダプター提供の API について説明し、IBM WebSphere BI Station ツールと SAPODA を使用してこれらに対応するビジネス・オブジェクトを開発するためのステップを示します。

重要: IBM WebSphere BI Station ツールを使用して、ビジネス・オブジェクト定義または ABAP Handler を生成するためには、SAP システムに英語でログオンする必要があります。WebSphere BI Station ログは英語でのみ、使用可能です。また、SAPODA 用として SAP システムに英語でログオンする必要があります。

ビジネス・オブジェクト属性のプロパティー

ビジネス・オブジェクト・アーキテクチャーでは、属性のためのさまざまなプロパティーを定義しています。このセクションでは、これらのプロパティーのいくつかをコネクターが解釈する方法と、ビジネス・オブジェクトを変更する際にこれらのプロパティーを設定する方法について説明します。表 47 に、ABAP Extension Module のビジネス・オブジェクト属性プロパティーをリストしています。

表 47. ABAP Extension Module のビジネス・オブジェクト属性プロパティー

プロパティー名	説明
Name	各ビジネス・オブジェクト属性は、固有の名前を持つ必要があります。
Type	値は `String` です。
MaxLength	このプロパティーは使用されていません。
IsKey	ビジネス・オブジェクトの最初の単純な属性は、キー属性として設定されます。すべてのキー属性は String タイプである必要があります。子オブジェクトをキー属性として設定することはサポートされていません。
IsForeignKey	このプロパティーは使用されていません。
IsRequired	このプロパティーは、属性が値を含んでいる必要があるかどうかを指定します。

表 47. *ABAP Extension Module* のビジネス・オブジェクト属性プロパティー *(続き)*

プロパティー名	説明
AppSpecificInfo	このプロパティーの値は、そのビジネス・オブジェクトをどの ABAP Handler がサポートするかによって異なります。アダプターでは、この値を自動的に設定するビジネス・オブジェクト生成ツールを提供しています。生成された値を変更すると、ビジネス・オブジェクトは正しく処理されなくなる場合があります。
DefaultValue	このプロパティーは、実行時値がない場合にこの属性に割り当てる値を指定します。

アダプター開発ツール

アダプターでは、SAP アプリケーション内から WebSphere ビジネス・オブジェクト定義ファイルを生成できる、ビジネス・オブジェクト開発ツールを提供しています。このビジネス・オブジェクト定義ファイルは、それを生成した SAP ビジネス・プロセスおよび API に直接対応します。

注: 統合ブローカーが IBM InterChange Server Express であるため、最後のビジネス・オブジェクト定義ファイルの先頭にバージョンが含まれていることを確認してください。InterChange Server Express の初期のバージョンでは、バージョン・テキストが、製品ディレクトリーの下の `¥repository¥ReposVersion.txt` ファイル内に置かれていることが必要とされます。定義に、すべての必須ビジネス・オブジェクトおよび属性 (`ObjectEventID` 属性を含む) が含まれていることも確認してください。

IBM WebSphere BI Station には、Inbound Wizard ツールがあります。

重要: IBM WebSphere BI Station を使用して、ビジネス・オブジェクト定義または ABAP Handler を生成するためには、SAP システムに英語でログオンする必要があります。WebSphere BI Station ログは英語でのみ、使用可能です。また、SAPODA 用として SAP システムに英語でログオンする必要があります。

Inbound wizard

Inbound Wizard ツールを使用すると、必要な機能をサポートする SAP トランザクションを 1 段階ずつたどっていく際のアクションを記録することにより、ビジネス・オブジェクトおよびその処理に必要なメタデータを定義することが可能です。ABAP コードを記述したり、ビジネス・オブジェクトの基礎となっているデータベース・スキーマについて知っておく必要はありません。

Inbound Wizard は、SAP トランザクションでのユーザーのアクションを記録および解釈することで、動的トランザクション表のデータを生成します。このウィザードは、フラットな (非階層型の) ビジネス・オブジェクトの定義をサポートします。つまり、子ビジネス・オブジェクトを含んだビジネス・オブジェクトはサポートされません。Inbound Wizard は、静的コードを必要とする、より複雑なオブジェクトを開発するためのコード生成プログラムとして利用できます。

注: 動的トランザクション表でエントリーを追加したり、変更することにより、手動で新しいビジネス・オブジェクトを開発したり、既存のビジネス・オブジェクトを変更することができます。

ビジネス・オブジェクト要求に対応するビジネス・オブジェクトの開発の詳細については、『動的トランザクションを使用したビジネス・オブジェクトの開発』を参照してください。

SAPODA を使用したビジネス・オブジェクト定義の開発

SAPODA を使用すると、IDoc、または動的検索と動的トランザクションが使用する表に基づいて、WebSphere ビジネス・オブジェクト定義を作成することができます。SAPODA を使用したビジネス・オブジェクトの開発の詳細については、35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

動的トランザクションを使用したビジネス・オブジェクトの開発

動的トランザクション機能モジュールは、マッピング・ツールおよび動的コード生成プログラムです。これは、SAP の呼び出しトランザクション API を使用して、SAP アプリケーションにデータを渡します。また、バッチ・データ通信 (BDC) セッションの静的定義を、オブジェクトと動詞の組み合わせごとに保管します。BDC データが呼び出しトランザクションに渡される前に、ビジネス・オブジェクト属性値が BDC セッションにマップされます。呼び出しトランザクションが完了すると、ビジネス・オブジェクトの適切な値に結果であるキー値が設定され、呼び出しトランザクションからのすべてのメッセージが記録されます。

動的トランザクション機能モジュールは BDC セッションを作成し、動的トランザクション表 `/CWLD/WIZ_IN` で定義されている BDC と、送られてきたビジネス・オブジェクトの値とを組み合わせることで、呼び出しトランザクションを実行します。動的トランザクション機能モジュールが呼び出されると、以下のステップが実行されます。

1. すべてのエントリーは `/CWLD/WIZ_IN` から検索されます。各項目の説明は以下の通りです。

   `object name = objectName and verb = objectVerb`
2. フィールド入力値が、属性名に基づいてビジネス・オブジェクトから BDC セッションにマップされます。
3. BDC セッションが、呼び出しトランザクションを使用して処理されます。
4. キー値が取り込まれ、呼び出しトランザクション・メッセージが記録され、ビジネス・オブジェクトにキーが設定されます。

ヒント

- 初期画面で入力したデータがすべての行項目のデフォルトになり、入力しなければならない行項目が少なくなる場合があります。
- 追加の入力が必要な詳細画面にドリルダウンしなくても、行項目の概要画面で必要な項目を入力できる場合があります。
- 通常 BDC では、例えば、`Are you sure you want to save?` のような確認メッセージに応答する必要はありません。

- カウンターは、オブジェクトと動詞の各組み合わせについて、変更モードで表の保守を開始および終了するたびに、10 ずつ増やされます。
- 実行中、呼び出しトランザクションでは日付の書式設定にユーザーの設定が使用されます。WebSphere Business Integration システムでは YYYY-MM-DD が標準の日付形式であるため、コネクターのユーザーが必ずこの日付形式のバリエーションを使用するようにセットアップしてください。同様に、トランザクションを段階ごとにたどることでビジネス・オブジェクトを再処理するには、自分のユーザー設定を変更してください。

ビジネス・オブジェクト用の BDC セッションの作成

BDC セッションを作成するには、SAP トランザクションの設計を理解する必要があります。SAP トランザクションでは、同じデータをさまざまな順序で異なる画面から入力できます。通常、各順序またはフローは、追加機能を公開しています。その結果、一定のデータ妥当性検査および入力フィールドの要件が一部の画面には存在し、他の画面には存在しません。必要な作業を最も少ない労力で行うことのできる順序を見つけ出すことは、とても大変です。単純な BDC セッションは、複雑な BDC セッションよりも安定しています。

SAP トランザクションは、オンラインで実行する代わりにバックグラウンド・プロセスで呼び出しトランザクション方式を使用してアクセスすると、異なる動作を示す場合があります。例えば、異なる画面や追加の画面が表示されたり、オンラインで調査している際に表示されるものとは異なる入力フィールドが画面に表示されたりする場合があります。この矛盾が発生するのは、トランザクションの制御コードが、バックグラウンドで実行した場合と、オンラインで実行した場合とで異なる動作を指示しているためであることがあります。その結果、トランザクションを段階ごとにたどっていく際に失敗したオブジェクト・イベントを再処理すると、オンライン・テストでは成功することがありますが、コネクターで同じオブジェクトを処理すると常に失敗します。このような事態が発生したら、バックグラウンドで処理するように BDC を変更してください。BDC を変更すると、BDC はバックグラウンドでは処理できるが、オンラインで処理すると失敗する、という事態が発生する場合があります。

動的トランザクション表で定義した BDC は静的です。そのため、なんらかの入力データが別の画面をポップアップさせたり、実行時に他のフィールドが必須になった場合、トランザクション中には対応できません。一貫性のある動作を予期できるためには、トランザクションの構成を正しく調査することが重要です。トランザクションを複数回試行して、同じ動作が繰り返されるかどうかをガイドラインとすることができます。

画面のフローを決定したら、以下のステップを実行し、収集した情報をスプレッドシートに文書化しておきます。

1. オブジェクトをサポートするトランザクションに移動し、トランザクション・コードを確認します。
2. 必要な画面および入力フィールドの BDC 要素を確認します。
3. 次の画面に処理を継続するために必要なメニュー・コマンドを確認します。
4. 必要な各画面について、ステップ 2 および 3 を繰り返します。
5. トランザクションを保管するコマンドを記録して終了します。

表 48 に、動的トランザクション・テーブル /CWLD/WIZ_IN の列名を示します。

表 48. Dynamic Retrieve の /CWLD/WIZ_IN テーブル・エントリー

フィールド名	説明	使用する場合	テクニカル名
Object name	IBM WebSphere ビジネス・オブジェクト名	常時	OBJ_NAME
Verb	動詞 (Create、Update、Delete、または Retrieve)	常時	OBJ_VERB
Counter	カウンター	常時	POSNR
Program	画面に関連付けられているプログラム	BDC 画面識別	PROG_NAME
Screen number	画面に関連付けられている画面番号	BDC 画面識別	DYNPRO
Start	新しい画面を指定	BDC 画面識別	DYNBEGIN
Screen description	画面、フィールド、またはコマンドの自由なテキスト記述	ユーザーの任意	SCR_DESCR
BDC field name	BDC 入力フィールド名	BDC 入力フィールド	FNAM
Field name in business object	入力値を提供する IBM WebSphere ビジネス・オブジェクト内の属性	BDC 入力フィールド	SOURCEFLD
Default value	IBM WebSphere ビジネス・オブジェクトでエントリーが提供されない場合、またはコマンド値である BDC_OKCODE を使用している場合に使用する静的なデフォルト値	値が渡されない可能性があるが、トランザクションには必須である場合	DEFLT_VAL
SY Field name	デフォルト値として使用される動的システム・フィールド (例えば DATUM)	値が渡されないか、または SAP システム・フィールドによって決定される必要がある場合	SYFIELD
Return	トランザクション完了時にキー値を戻すシステム・メッセージ・フィールドを識別する 1 から 4 の数値 (SY-MSGV#)	キー値を受け取るビジネス・オブジェクト・キー属性	RETURNFLD
Length	入力に使われる属性値のゼロ位置からの文字長	複合値を含んだ属性を使用する場合のみ	LENGTH

ビジネス・オブジェクトのメタデータを定義または変更する (情報を /CWLD/WIZ_IN に転送する) には、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。

   重要: IBM WebSphere BI Station を使用して、ビジネス・オブジェクト定義または ABAP Handler を生成するためには、SAP システムに英語でログオンする必要があります。WebSphere BI Station ログは英語でのみ、使用可能です。

2. 「Development」タブをクリックします。

3. 「Transaction based - Inbound」セクションで「Modify BO Metadata」ボタンをクリックします。

ビジネス・オブジェクトのメタデータは簡単に定義できます。各画面で、最初のエントリーは画面を識別し、後続のエントリーは入力フィールドを識別し、最後のエントリーはコマンドです。このグループ化が各画面について繰り返されます。

SAP4_CustomerMaster の例について、「Counter」列を行番号として使いながら、段階ごとに説明します。

100 画面番号 100 のプログラム SAPMF02D から始まります。これは新しい、先頭の画面なので、「Start」列にフラグが立てられています。

110 画面上の 110 では、このビジネス・オブジェクトの Customer_account_group 属性の値を使用し、「BDC field name」列 (値は RF02D-KT0KD) に追加します。デフォルト値を 0001 と指定します。Customer_account_group 属性が CxIgnore の場合、「BDC field name」列はデフォルト値 0001 を受けとります。

120 Customer_Account_Number 属性はキー値なので、呼び出しトランザクション中には設定されません。SAP では、キー値は内部的に割り当てられ、トランザクションが正常に通知された後にのみ使用可能になります。このため、「BDC field name」列はブランクにしておきますが、Customer_Account_Number 属性は呼び出しトランザクションの終了時に戻される際にこのキー値を使用して設定される必要があるため、表にエントリーを設けておきます。また、CustomerNumber の「Program」列に RETURN と入力します。

トランザクションによって、SAP は SY-MSGV1、SY-MSGV2、SY-MSGV3、または SY-MSGV4 の 4 つのフィールドのうちの、いずれか 1 つにキー値を戻します。特定の属性に戻り値を設定するには、「Return」列に 1 から 4 の数値を入力します。この番号はキー値を持つ SY-MSGV 番号フィールドに対応しています。

130 最初の画面に必要な値の入力を終えたら、「Default Value」列にコマンド /00 を入力します。これは、Enter キーを押す操作をシミュレートします。これにより、次のトランザクション画面に移動します。コマンドは、トランザクション・コードを入力する画面入力フィールド BDC_OKCODE に入力します。

140 ここから、次のトランザクション画面の設定に移ります。アドレス情報を入力します。これは新しい画面なので、「Start」列にフラグを立てます。この例では、2 番目の画面は最初の画面と同じプログラムに関連付けられているため、画面番号が 100 から 110 に変わるだけです。これは場合によって異なります。

150 - 210 ビジネス・オブジェクト内の Name_1、 Sort_field、 City、 P_0_Box_postal_code、 Country_key、 Language_keys、および Post_office_box の各属性の値を使用して、「BDC field name」列に対応する値を追加します。

220 行 130 と同様です。この画面の処理は完了しました。ただし、単に

Enter キーをシミュレートするより、コマンド値 UPDA を入力して、トランザクションを保管してください。これにより、次のトランザクション画面に移動します。

230 ここから、3 番目のトランザクション画面の設定に移るため、「Start」列にフラグを立てます。この例のビジネス・オブジェクトでは、この画面のデータは必要ないので、次の行でこの画面の処理を完了します。

240 行 130 と同様です。この画面の処理は完了しました。コマンド値 /00 を入力して、Enter キーを押す操作をシミュレートします。これにより、最終のトランザクション画面に移動します。

250 ここから、最終のトランザクション画面の設定に移ります。「Start」列にフラグを立てます。

260 行 150 から 210 と同様です。ビジネス・オブジェクト属性 *Transport_zone_to_which_or_from_which_the_goods_are_delivered* の値を使用して、それに対応する値 (KNA1-LZONE) を「BDC field name」列に追加します。

270 行 220 と同様です。この画面の処理が完了し、トランザクションが完了したため、保管するためのコマンド値 UPDA を入力します。これは、呼び出しトランザクション API が受け取る最後のアクションです。

280 どのビジネス・オブジェクトでも、最後のエントリーは常にトランザクション・コードの指定です。キーワード TCODE は「Program」列に入力され、トランザクション・コードは「BDC field name」列に入力されます。

これにより、SAP4_CustomerMaster ビジネス・オブジェクトの BDC セッションの定義が完了します。

呼び出しトランザクションが失敗してエラー・メッセージを戻した場合、以下に説明する一般的なエラーのいずれかが発生している可能性があります。

- SAP アプリケーションが、BDC が予期しなかった画面を呼び出したために No input available for program XX and screen YY というメッセージが戻された。この事態が発生した場合は、動的トランザクション表に、プログラム XX および画面 YY の入力画面を処理するための適切なエントリーを追加します。
- SAP アプリケーションが、存在しないフィールドを設定するように BDC から指示される場合があります。その場合、SAP アプリケーションは明示的に設定されていない独自の指示を実行した可能性があります。その結果、意図とは異なる画面に移動します。この事態が発生した場合は、正しい画面に移動するための設定だけを追加して、もう一度指示を行います。

動的トランザクションに対する Inbound Wizard の使用

Inbound Wizard は、最初のフィールドをクリックするか、または画面を変更した際に、トランザクションでのナビゲーション、アクション、およびフィールド入力を記録します。レコーダーは発生したすべてのアクションを記録しますが、表示されたものすべてを記録するわけではありません。例えば、最初に初期画面が表示された際、レコーダーはトランザクションへの最初の呼び出しを取り込みますが、その

画面に表示された入力フィールドのすべてを取り込むわけではありません。入力フィールドを使用できるようにするには、そのフィールドに何らかのデータを入力する必要があります。また、入力フィールドにデフォルト・データが格納されている場合でも、そのデータを手動で入力しない限り記録されません。

新しい WebSphere ビジネス・オブジェクト定義を作成するには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。

   重要: IBM WebSphere BI Station を使用して、ビジネス・オブジェクト定義または ABAP Handler を生成するためには、SAP システムに英語でログオンする必要があります。WebSphere BI Station ログは英語でのみ、使用可能です。

2. 「Development」タブで、「Inbound Wizard」ボタンをクリックします。
3. 以下の情報を入力します。
   - ビジネス・オブジェクト名 — ビジネス・オブジェクト・タイプの名前と、そのオブジェクトのすべてのインスタンスの名前。新しいビジネス・オブジェクトを作成する場合は、新しい名前を入力します。ビジネス・オブジェクトを定義する単純な名前を使用することをお勧めします。既存のビジネス・オブジェクトを使用する場合は、ドロップダウン・リストから選択します。
   - 動詞 — そのビジネス・オブジェクトでサポートされる動詞。
   - トランザクション・コード — ビジネス・オブジェクトで実行される必要な機能をサポートする画面のトランザクション・コード。画面のトランザクション・コードを取得するには、「System」メニューで「Status」をクリックします。コードは、SAP データの下の「Transaction」フィールドにリストされます。
4. 「Record」をクリックします。
5. ビジネス・オブジェクトの機能をサポートするトランザクションを、段階ごとにたどっていきます。必要なすべてのフィールドおよび画面を使用してください。終了したら、トランザクションを保管します。
6. ビジネス・オブジェクトにメタデータとして組み込むコンポーネントを選択します。カーソルをコンポーネントの上に置き、「Select/Deselect sub-tree」ボタンをクリックします (F9)。デフォルトでは、すべてのコンポーネントが選択されます。
7. 新しい動的オブジェクトまたはソース・コードを生成します。
   - 動的トランザクション表のメタデータを生成および挿入するには、「Generate Meta data」ボタンをクリックします (F6)。このデータから、WebSphere ビジネス・オブジェクト定義を生成できます。
   - BDC データおよびフィールド記述を収めたテキスト・ファイルを生成するには、「Generate Code in Text File」ボタンをクリックします (F5)。このデータからは、WebSphere ビジネス・オブジェクト定義を生成できません。

IDoc を使用したビジネス・オブジェクトの開発

ABAP Extension Module 用の WebSphere ビジネス・オブジェクトは、SAP 内で IDoc として定義できます。IDoc は、ALE (Application Link Enabling) と呼ばれる SAP の EDI ソリューションの一部です。IDoc の定義は SAP の BOR (ビジネス・オブジェクト・リポジトリー) に保管され、SAP システム内でグローバルにアクセスできます。ここでは、SAP ネイティブ API と組み合わせて使用するための準備として、SAP アプリケーション内の WebSphere ビジネス・オブジェクトを解釈および解析できるように、ALE の定義部を活用しています。アダプターは、IDoc を使用して開発されたビジネス・オブジェクトをサポートする IDoc Handler を提供しています。

IDoc Handler は、2 つの機能モジュールから構成されています。Dynamic Retrieve や Dynamic Transaction などの、その他の ABAP Handler は、単一の機能モジュールのみで構成されています。

/CWLD/RFC_DO_VERB_NEXTGEN はビジネス・オブジェクト・データを IDoc Handler /CWLD/IDOC_HANDLER に受け渡します。この IDoc Handler は、すべてのオブジェクト・タイプに汎用で、指定された IDoc のタイプを取得するため、およびビジネス・オブジェクト・データを IDoc の構造に再フォーマットするためにビジネス・オブジェクトのアプリケーション固有情報を使用します。データを再フォーマットした後、汎用 IDoc Handler はビジネス・オブジェクト・データをオブジェクト固有 IDoc Handler に (そのビジネス・オブジェクトのタイプと動詞の組み合わせに基づいて) 渡します。このオブジェクト固有 IDoc Handler は、SAP ネイティブ API との統合を処理します。オブジェクト固有 IDoc Handler は、ビジネス・オブジェクト・データの処理を終えると、IDoc 形式のビジネス・オブジェクト・データを /CWLD/IDOC_HANDLER に戻します。この汎用 IDoc Handler は、ビジネス・オブジェクト・データを変換して元のフォーマットに戻し、/CWLD/RFC_DO_VERB_NEXTGEN に戻します。

図 75 に、IDoc Handler の基本アーキテクチャーを示します。

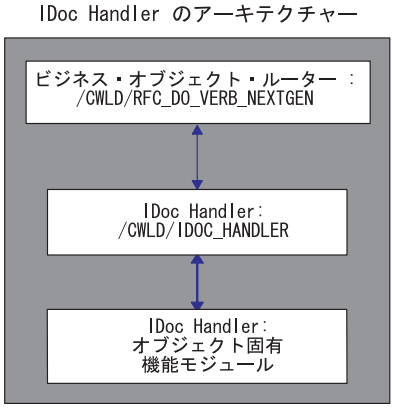


図 75. IDoc Handler のアーキテクチャー

アダプター提供の IDoc Handler を使用するには、IDoc を SAP アプリケーション内で定義する必要があります。SAP 提供、またはカスタマー作成の IDoc を使用できます。IDoc 定義は SAP 用の WebSphere ビジネス・オブジェクトの定義をミラ

ーする必要があるため、アダプターは IDoc を基にした WebSphere ビジネス・オブジェクト定義を生成する SAPODA を提供しています。

SAPODA を使用したビジネス・オブジェクト定義の生成

SAPODA を使用して、以下の IDoc に基づいた ABAP Extension Module 用のビジネス・オブジェクト定義を生成できます。

- ファイルとして抽出された IDoc
- SAP システム内で定義された IDoc

重要: SAPODA を使用するために、SAP システムに英語でログオンする必要があります。

SAPODA を使用してビジネス・オブジェクト定義を生成する場合は、Business Object Designer Express を使用して定義を表示、および変更できます。SAPODA の使用方法の詳細については、35 ページの『第 4 章 SAPODA を使用したビジネス・オブジェクト定義の生成』を参照してください。

IDoc を定義したら、そのビジネス・オブジェクトがサポートする必要のある各動詞について、機能モジュールを作成します。各機能は、/CWLD/IDOC_HANDLER から呼び出すことができるように、次に示すインターフェースを備える必要があります。

```
*" IMPORTING
*"        VALUE(OBJECT_KEY_IN) LIKE /CWLD/LOG_HEADER-OBJ_KEY OPTIONAL
*"        VALUE(INPUT_METHOD) LIKE  BDWFAP_PAR- NPUTMETHD OPTIONAL
*"        VALUE(LOG_NUMBER) LIKE /CWLD/LOG_HEADER-LOG_NR OPTIONAL
*" EXPORTING
*"        VALUE(OBJECT_KEY_OUT) LIKE /CWLD/LOG_HEADER-OBJ_KEY
*"        VALUE(RETURN_CODE) LIKE /CWLD/RFCRC_STRU-RFCRC
*"        VALUE(RETURN_TEXT) LIKE /CWLD/LOG_HEADER-OBJ_KEY
*" TABLES
*"        IDOC_DATA STRUCTURE  EDID4
```

IDoc Handler と create、update、および delete 動詞

Create、Update、および Delete 操作をサポートする IDoc Handler は、IDoc としてフォーマットされたビジネス・オブジェクト・データを受け取ります。これらの操作の役割は、ビジネス・オブジェクト・データを SAP の呼び出しトランザクション API と統合し、オブジェクト・キーを生成することです。オブジェクト・キーのみが /CWLD/IDOC_HANDLER を通じてコネクターに戻され、ビジネス・オブジェクト・データは戻されません。/CWLD/IDOC_HANDLER は、ビジネス・オブジェクト・データをメモリーに格納し、オブジェクト・キーを親ビジネス・オブジェクトで IsKey としてマークされた最初の属性に挿入します。それから /CWLD/IDOC_HANDLER は、コネクターにビジネス・オブジェクト・データを戻します。

注: 統合ブローカーが InterChange Server Express であるため、ビジネス・オブジェクト・データを維持することが重要です。マッピング・インフラストラクチャーが、動的相互参照のために ObjectEventId の保持を必要とするためです。

次に示すサンプル・コードは、以下のフローを表しています。

1. グローバル・データを初期化します。
2. IDoc をデコンストラクションして作業表に入れます。

- すべてのオブジェクトが SAP アプリケーションに送信されるわけではないため、ターゲットの構造を '/' (CxIgnore) を使用して初期化します。/CWLD/INBIDOC_FRMS0 内の形式を使用します。
- 複数のオブジェクト間で振る舞いに一貫性を持たせるため、/CWLD/INBIDOC_FRMS0 内の形式を使用して、データを IDoc から内部表に転送します。

3. BDC を作成します。複数のオブジェクト間で振る舞いに一貫性を持たせるため、/CWLD/INBIDOC_FRMS0 内の形式を使用して、データを内部表から BDC 表に転送します。
4. 呼び出しトランザクションを作成します。
5. オブジェクト・キーを取り込みます。

次に示すサンプル・コードでは、SAP Sales Quote Create がサポートされます。

```
*- Initialize working variables and internal tables
  PERFORM INITIALIZE_IN.

*- I01(MF): Begin IDoc interpretation
  PERFORM LOG_UPDATE(/CWLD/SAPLLOG) USING C_INFORMATION_LOG TEXT-I01
                                     SPACE SPACE SPACE.

*- Interpret IDoc data structure
  IF NOT IDOC_DATA[] IS INITIAL.

*- Move IDoc to internal tables
    PERFORM INTERPRET_IDOC.

*- Check some of the input fields
    PERFORM CHECK_INPUT.

*- If key values were missing, exit function
    IF RETURN_CODE NE 0.
      EXIT.
    ENDIF.

*- E01(MF): No Idoc data lines sent for processing.
  ELSE.

    RETURN_CODE = 2.
    RETURN_TEXT = TEXT-E01.
    EXIT.

  ENDIF.

*- Build the BDC session for transaction VA21.
  PERFORM BUILD_BDC_VA21.

*- Call Transaction
  PERFORM LOG_UPDATE(/CWLD/SAPLLOG) USING C_INFORMATION_LOG TEXT-I02
                                     'VA21' C_BLANK C_BLANK.

  CALL TRANSACTION 'VA21' USING BDCDATA
                           MODE INPUT_METHOD
                         UPDATE 'S'
                       MESSAGES INTO BDC_MESSAGES.

*- Capture return code and object key from transaction
  PERFORM PREPARE_RETURNED_MESSAGE.

ENDFUNCTION.
```

Create ロジックの主要な機能は次の 2 つです。

- IDoc データを管理可能なデータ構造に変換します。
- 呼び出しトランザクションを実行します。

IDOC 構造の変換

Create ロジックの最初の部分では、IDoc 構造のデータを作業データ構造に変換する操作を行います。そのためには、以下のようなコードを作成する必要があります。

```
loop at idoc_data.

    case idoc_data-segnam.
      when 'ZSQVBAK'.                  " Header Data
        move idoc_data-sdata to zsqvbak.

      when 'ZSQVBUK'.                  " Status Segment
        move idoc_data-sdata to zsqvbuk.

      when 'ZSQVBP0'.                  " Partner Header Level
        move idoc_data-sdata to zsqvbp0.

      when 'ZSQVBAP'.                  " Item Detail
        move idoc_data-sdata to zsqvbap.

      when 'ZSQVBA2'.                  " Item Detail Part 2
        move idoc_data-sdata to zsqvba2.

      when 'ZSQVBUP'.                  " Item Status
        move idoc_data-sdata to zsqvbup.

      when 'ZSQVBKD'.                  " Commercial data
        move idoc_data-sdata to zsqvbkd.

      when 'ZSQKONV'.                  " Condition
        move idoc_data-sdata to zsqkonv.

      when 'ZSQVBPA'.                  " Partner Item Level
        move idoc_data-sdata to zsqvbpa.

    endcase.

  endloop.
```

インバウンド呼び出しトランザクション・ロジックの作成

作成ロジックの第 2 の部分では、データを SAP アプリケーション・データベースに追加する操作を行います。BAPI や SAP 標準機能などの提供されている機能を使用したり、独自に開発した呼び出しトランザクション機能を使用できます。提供されている機能を使用する場合は、その機能が将来のリリースで変更される場合があることに注意してください。データベースに書き込む代わりに、呼び出しトランザクションを使用することをお勧めします。呼び出しトランザクションを使用すると、SAP データベースの変更に影響を受けず、必要な範囲と機能に焦点を絞ったカスタム機能を開発できます。

ビジネス・オブジェクト・データを SAP に渡すには、ABAP コードの一部を、IBM WebSphere BI Station の Inbound Wizard (トランザクション /n/CWLD/HOME) または SAP BDC レコーダーを使用するか、手動で開発することによって生成します。

Inbound Wizard は、作成トランザクションのアクティビティーを記録し、BDC ロジックを記述したテキスト・ファイルを作成します。Sales Quote の例では、トランザクション VA21 が記録されています。

Inbound Wizard を使用してトランザクション VA21 を記録するには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション `/n/CWLD/HOME`)。

   重要: IBM WebSphere BI Station を使用して、ビジネス・オブジェクト定義または ABAP Handler を生成するためには、SAP システムに英語でログオンする必要があります。WebSphere BI Station ログは英語でのみ、使用可能です。

2. 「Development」タブで、「Inbound Wizard」ボタンをクリックします。
3. 以下の情報を入力します。
   - ビジネス・オブジェクト名 — ビジネス・オブジェクト・タイプの名前と、そのオブジェクトのすべてのインスタンスの名前。新しいビジネス・オブジェクトを作成する場合は、新しい名前を入力します。ビジネス・オブジェクトを定義する単純な名前を使用することをお勧めします。既存のビジネス・オブジェクトを使用する場合は、ドロップダウン・リストから選択します。
   - 動詞 — そのビジネス・オブジェクトでサポートされる動詞。
   - トランザクション・コード — ビジネス・オブジェクトで実行される必要な機能をサポートする画面のトランザクション・コード。画面のトランザクション・コードを取得するには、「System」メニューで「Status」をクリックします。コードは、SAP データの下の「Transaction」フィールドにリストされます。
4. 「Record」をクリックします。
5. ビジネス・オブジェクトの機能をサポートするトランザクションを、段階ごとにたどっていきます。必要なすべてのフィールドおよび画面を使用してください。終了したら、トランザクションを保管します。
6. ビジネス・オブジェクトにメタデータとして組み込むコンポーネントを選択します。カーソルをコンポーネントの上に置き、「Select/Deselect sub-tree」ボタンをクリックします (F9)。デフォルトでは、すべてのコンポーネントが選択されます。
7. 新しい動的オブジェクトまたはソース・コードを生成します。
   - 動的トランザクション表のメタデータを生成および挿入するには、「Generate Meta data」ボタンをクリックします (F6)。このデータから、WebSphere ビジネス・オブジェクト定義を生成できます。
   - BDC データおよびフィールド記述を収めたテキスト・ファイルを生成するには、「Generate Code in Text File」ボタンをクリックします (F5)。このデータからは、WebSphere ビジネス・オブジェクト定義を生成できません。

次に示すサンプル・コードは、生成された BDC セッションの最初の数行から抜粋したものです。

```
* Sales doc. Initial screen Create
perform dynpro_new using 'SAPMV45A' '0101' .

* Sales document type
```

```
perform dynpro_set using 'VBAK-AUART' 'QT' .

* Distribution channel
perform dynpro_set using 'VBAK-VTWEG' 'sourcefield' .

* Division
perform dynpro_set using 'VBAK-SPART' 'sourcefield' .

* Function Command
perform dynpro_set using 'BDC_OKCODE' '=ENT2' .

* 4.0: Screen Container for Overview Screens (normal header)
perform dynpro_new using 'SAPMV45A' '4001' .

* Sold-to party
perform dynpro_set using 'KUAGV-KUNNR' '238' .

* Ship-to party
perform dynpro_set using 'KUWEV-KUNNR' '238' .

* Function Command
perform dynpro_set using 'BDC_OKCODE' '=KKAU' .

* 4.0: Screen container for document header screens
perform dynpro_new using 'SAPMV45A' '4002' .

* Date until which bid/quotation is binding (valid-to date)
perform dynpro_set using 'VBAK-BNDDT' '20000630' .
```

SAP の BDC レコーダーを使用することもできます (トランザクション SHDB)。次に示すサンプル・コードは、BDC レコーダーを使用して生成されたものです。

```
start-of-selection.

read dataset dataset into record.
 if sy-subrc <> 0. exit. endif.

 perform bdc_dynpro      using 'SAPMV45A' '0101'.
 perform bdc_field       using 'BDC_CURSOR'
                               'VBAK-AUART'.
 perform bdc_field       using 'BDC_OKCODE'
                               '=ENT2'.
 perform bdc_dynpro      using 'SAPMV45A' '4001'.
 perform bdc_field       using 'BDC_OKCODE'
                               '=KKAU'.
 perform bdc_field       using 'BDC_CURSOR'
                               'KUWEV-KUNNR'.
 perform bdc_field       using 'KUAGV-KUNNR'
                               record-KUNNR_001.
 perform bdc_field       using 'KUWEV-KUNNR'
```

この方法からの出力には、最初の方法にあったビジネス・オブジェクト・コメントが含まれないため、あまりお勧めできません。SAP の BDC レコーダーを使用することの利点は、BDC の記録を検証するための独立した方法をもたらすことです。

もう 1 つの方法は、BDC を手動で生成することです。これは、機能全体を作成する場合には推奨できるアプローチではありませんが、上に挙げた方法を補助するためには利用できます。この方法は、入力データに対応して SAP トランザクション中に表示されることのある追加画面やポップアップ・ボックスのためにロジックを追加する必要がある場合に有用です。

IDoc Handler および retrieve 動詞

Retrieve 動詞をサポートするオブジェクト固有 IDoc Handler は、/CWLD/IDOC_HANDLER からビジネス・オブジェクト・データを受け取りません。その代わりに、/CWLD/IDOC_HANDLER はオブジェクト固有 IDoc Handler 関数の OBJECT_KEY_IN パラメーターを使用して、IsKey とマークされた最初の属性の値のみを渡します。オブジェクト固有 IDoc Handler は、この属性の値を使用して、ABAP SQL を使用するビジネス・オブジェクトのインスタンスに関するすべての情報を検索したり、適切な IDoc 構造に含まれるそのデータをフォーマットしたりします。

注: キーが複数のフィールドから構成されている場合、イベント検出機構 (または統合ブローカーが InterChange Server Express の場合はマップ) は、これらのフィールドの値を連結して、トップレベル・ビジネス・オブジェクトの最初のキー属性にします。/CWLD/IDOC_HANDLER は、この連結されたキーをその OBJECT_KEY_IN パラメーターにロードします。オブジェクト固有 IDoc Handler は、OBJECT_KEY_IN パラメーターの値を解析して、複数キー・フィールドにする必要があります。この機能を維持するためには、/CWLD/IDOC_HANDLER を使用する場合に、キーに対して名前と値のペアを指定しないことが重要です。

次に示すコード・フラグメントは、Sales Quote を検索するためのオブジェクト固有 IDoc Handler を示したものです。Sales Quote ビジネス・オブジェクトは、VBAK、VBUK、VBPO、VBAP、VBUP、VBKD、KNOV、および VBPA の表からデータを検索します。これらの表は、IDoc タイプ ZSLSQUOT の階層およびカーディナリティーに従います。このコードでは、以下の処理が実行されます。

1. グローバル・データを初期化します。
2. SAP アプリケーション・データベースからビジネス・オブジェクト・データを戻します。
3. 戻されたデータから IDoc を作成し、そのデータを /CWLD/IDOC_HANDLER に戻します。

IDoc タイプ ZSLSQUOT に対するオブジェクト固有 IDoc Handler のコード・フラグメントは次のとおりです。

```
*- Clear the interface structures.
  clear: g_text, object_key_out, return_code, return_text, idoc_data.
  refresh: idoc_data.

*  If no key value is specified, log it as an error and exit.
  if object_key_in is initial or
     object_key_in = c_cxignore_const.
    perform log_update(/cwld/sapllog) using c_error_log text-e02
                                       space space space.
    return_code = 1.
    return_text = text-e02.
    exit.
  endif.

  perform initialize_global_structures.

  perform fill_internal_tables.
  if not return_code is initial.
    exit.
  endif.

* Build Idoc segments from internal tables
```

```
  perform fill_idoc_inttab.

  return_code = 0.
  return_text = text-s01.

  perform log_update(/cwld/sapllog) using c_information_log text-s01
                                    space space space.
endfunction.
```

最も重要なパラメーター 2 つは、インバウンド・キーのための OBJECT_KEY_IN、およびアウトバウンド・データのための IDOC_DATA です。OBJECT_KEY_IN は、複数キーを表す連結されたストリングである場合があることに注意してください (定義した規則に依存)。オブジェクト固有 IDoc Handler は、連結された値を解析して、その各部分を適切なキー・フィールドにロードします。この機能を維持するためには、/CWLD/IDOC_HANDLER を使用する場合に、キーに対して名前と値のペアを指定しないことが重要です。

VBAK 表は、子テーブルに対する選択基準を操作します。そのため、各表は作業表にロードされます。VBAK 表を使用すると、追加のキーを使用して子テーブルを検索できます。そのため、Sales Quote の例では、コードは次のようになります。

```
form fill_internal_tables.

  * Get information from VBAK, VBUK, VBAP, VBKD, KONV, VBPA

  select single * from vbak
         where vbeln = object_key_in.

  if sy-subrc <> 0.
    perform log_update(/cwld/sapllog) using c_error_log text-e01
                                object_key_out c_blank c_blank.
    return_code = '1'.
    g_text = text-e01.
    replace '&' with order_number into g_text.
    return_text = g_text.

    exit.
  endif.

  select single * from vbuk
         where vbeln = vbak-vbeln.

  select * from vbap into table t_vbap
         where vbeln = vbak-vbeln.

  * Continue for other tables
```

以下に示すコードは、要求されたデータをアプリケーション・データベースから内部表および作業変数にコピーするために使用されます。次に、このコードは WebSphere ビジネス・オブジェクト定義に直接に対応するセグメントを作成し、それらを SAP セグメント構造に挿入します。

IDoc タイプと作業構造の間でフィールドがほとんど一致している場合は、ABAP で移動に相当するコマンドを実行できる場合があります。そうでない場合、構造内のフィールドの総数に比べて移動するフィールドの数は少ないため、フィールドを作業表から IDoc タイプ表に手動で移動するようにお勧めします。これは、単純にデータを作業データ構造から IDoc 構造に転送し、次にフラットなデータ・フィールドに転送するために使用されます。

コードは次のとおりです。

```
form fill_idoc_inttab.

perform fill_zsqvbak.                   " Fill the Sales Quote Header
perform fill_zsqvbuk.                   " Fill the Sales Quote Status
perform fill_zsqvbap.                   " Fill Sales Quote Lines

endform.                                " FILL_IDOC_INTTAB

*-- fill the Sales Quote Header
form fill_zsqvbak.

  clear idoc_data.
  clear zsqvbak.
  idoc_data-segnam = 'ZSQVBAK'.

  move-corresponding vbak to zsqvbak.
  move zsqvbak to idoc_data-sdata.
  append idoc_data.

endform.                                " FILL_ZSQVBAK

*-- fill the Sales Quote Header Status
form fill_zsqvbuk.

  clear idoc_data.
  clear zsqvbuk.
  idoc_data-segnam = 'ZSQVBUK'.

  move-corresponding vbuk to zsqvbuk.
  move zsqvbuk to idoc_data-sdata.
  append idoc_data.

endform.                                " FILL_ZSQVBAK

*-- fill the Sales Quote Line and the Line Child segments
form fill_zsqvbap.

  loop at t_vbap.
    clear idoc_data.
    clear zsqvbap.
    idoc_data-segnam = 'ZSQVBAP'.

    move-corresponding t_vbap to zsqvbap.
    move zsqvbap to idoc_data-sdata.
    append idoc_data.

    perform fill_zsqvba2.
    perform fill_zsqvbup.
    perform fill_zsqvbkd.
    perform fill_zsqkonv.
    perform fill_zsqvbpa.


  endloop.

endform.

*-- fill second part of vbap
form fill_zsqvba2.
" etc.
```

ABAP Extension Module および ABAP Handler の呼び出し

コネクターは、ビジネス・オブジェクトの動詞アプリケーション固有情報の値を使用して、ABAP Extension Module の適切な ABAP Handler を呼び出します。ABAP Extension Module の適切な ABAP Handler を呼び出すためには、ABAP Extension Module のクラス名を指定し、また、そのビジネス・オブジェクトで使用される ABAP Handler 機能モジュールを指定する必要があります。例えば、SAP バージョン 4.6 をサポートする動的トランザクション ABAP Handler の動詞アプリケーション固有情報は、次のようになります。

```
AppSpecificInfo = sap.sapextensionmodule.VSapBoHandler,:/CWLD/DYNAMIC_TRANSACTION
```

注: コネクター・モジュール (クラス名) と ABAP Handler の間には、コンマ区切り文字を入れる必要があります。

ABAP Extension Module でのビジネス・オブジェクト処理の詳細については、210 ページの『ビジネス・オブジェクトの処理』を参照してください。

第 22 章 ABAP Extension Module のイベント検出の開発

イベント検出は、ABAP Extension Module の ABAP コンポーネントにおけるイベント・トリガー・プロセスの一部です。すべてのイベント検出機構は、イベント・トリガーを呼び出す必要があります。これは、検出されたイベントを取得して、イベント表に追加します。イベントのトリガーの詳細については、217 ページの『イベント・トリガー』を参照してください。

この章の内容は以下のとおりです。

イベント検出機構の設計

SAP アプリケーションでイベントを検出するには、さまざまな機構を利用できます。イベント検出機構は、機能モジュールを呼び出すことができる必要があります。コネクターは、次の 4 種類のイベント検出機構を実装しています。

- コード拡張 — SAP トランザクションの適切なポイントにイベント検出コードを挿入することにより、ビジネス・プロセス (通常は単一の SAP トランザクション) のために実装されています。
- バッチ・プログラム — イベントを検出するための基準を組み込んだ ABAP プログラムの開発を必要とします。
- ビジネス・ワークフロー — SAP 独自のオブジェクト指向イベント検出機能を使用します。
- 変更ポインター — ビジネス・ワークフローの一種である変更ポインター機構を実装し、変更文書の概念を使用してさまざまなビジネス・プロセスの変更を検出します。

これらのイベント検出機構の一部は特定のビジネス・プロセスに対しては利用できないため、開発する各ビジネス・オブジェクトについて、実装する適切なイベント検出機構を決定する必要があります。イベント検出を実装する必要のある各トランザクションについて、特定のビジネス・プロセスに関する技術的および機能的知識が必要です。

ビジネス・プロセスに実装するイベント検出機構を決定する際には、以下に示す実装上の考慮事項を検討してください。

可用性
そのビジネス・プロセスには、どのイベント検出機構が利用できるでしょうか? これは、最初に考慮する必要のある問題の 1 つです。コード拡張およびバッチ・プログラムは可用性が高いのに対して、ビジネス・ワークフローおよび変更ポインターの可用性は高くありません。

リアルタイム統合
イベントは同期的に検出する必要があるでしょうか? 多数のイベントを一度に検出する必要がありま

すか? バッチ・プログラム以外のすべての機構は、リアルタイム統合に適しています。

信頼性

イベントが生成された場合、そのビジネス・プロセスについてのデータ変更はすべて検出されますか? コード拡張、バッチ・プログラム、および変更ポインターは、オブジェクトのすべてのイベントの取り込みを最もよく制御できます。ビジネス・ワークフローの信頼性には制限があります。例えば、ビジネス・ワークフローは、ベンダー・トランザクション更新中のアドレス変更を検出しません。

柔軟性

イベントが起動される前に、特定の基準を評価する必要がありますか? トランザクションの特定のポイントでイベントを検出する必要がありますか? コード拡張は、イベント・データがコミットされる前に特定のポイントにコードを挿入できるため、最も柔軟性に優れています。変更ポインターとバッチ・プログラムは中程度の柔軟性を備えていますが、ビジネス・ワークフローの実装には、柔軟性はほとんどありません。

アップグレード依存性

SAP アプリケーションにアップグレードした場合、そのビジネス・プロセスについてイベントが検出される方法が変わりますか? 一般的には、これは一概には言えませんが、ビジネス・ワークフローおよび変更ポインターは、SAP によって管理されているため、アプリケーション変更の影響を最も強く受けます。

障害

時間や困難さのレベルは問題になりますか? 実装の困難さのレベルは、それぞれの機構によって異なります。一般に、バッチ・プログラムが最も簡単です。コード拡張およびビジネス・ワークフローはそれよりも若干困難ですが、変更ポインターは、SAP および評価対象のビジネス・プロセスについて、より詳細な知識が必要になるため、最も困難です。

将来のイベント

イベントをリアルタイムで取り込んでから、その検索を指定した日付まで遅らせることができる必要がありますか? 例えば、従業員レコードの住所を今日更新して、その変更を 3 週間後に有効化する必要がある場合があります。この場合、イベントの取り込みは更新時に行い、検索は発効日まで遅らせる必要があります。

この時点で、考慮する必要のあるイベント検出機構について理解しておく必要があります。 271 ページの表 49 でサポートする必要のある各ビジネス・プロセスに対して使用できる機構を判断するための一般的なガイドラインを示します。

表 49. イベント検出機構デシジョン・テーブル

	コード拡張	バッチ・プログラム	ビジネス・ワークフロー	変更ポインター
可用性	高	高	低	低
リアルタイム統合	可	なし	可	可
信頼性	高	高	低	中
柔軟性	高	中	低	中
アップグレード依存性	低	低	中	中
障害	中	低	中	高
将来のイベント	可	可	なし	なし

注意する必要のある最後の考慮事項は、サイトの開発の方法論です。多くの場合、ビジネス・ワークフローのみを使用したイベント検出が推奨される方法であり、コード拡張はまったく使用されません。

コード拡張の使用は、信頼でき、柔軟性に優れ、同期可能で、可用性が高いため、イベント検出には推奨できるアプローチです。これに対し、ビジネス・ワークフローおよび変更ポインター機構は、すべてのビジネス・プロセスに対して一般的に使用できるわけではありません。バッチ・プログラムは一般に、リアルタイム統合が必要ない場合に使用されます。

それぞれのイベント検出機構には、ビジネス・プロセス内のイベントを検出するための利点と欠点があります。以下のセクションでは、各イベント検出機構の詳細と、それぞれの主な利点と欠点について説明します。

これらのイベント検出機構は、すべてイベントのリアルタイム・トリガーおよび検索をサポートしています。ただし、コード拡張およびバッチ・プログラムだけは、遅延検索という追加機能を提供します。後の時点で検索されるように指定されたイベントは、将来のイベントと呼ばれます。

コード拡張

コード拡張は、SAP トランザクションのコード内の特定のポイントに実装します。ユーザー出口を利用すると、トランザクションの最もロジカルなポイントに、イベント検出コードを挿入できます。イベント検出コードは、イベントが生成されたかどうかの判断基準を評価するために使用します。

この機構の一般的戦略は、トランザクションのデータがデータベースにコミットされる直前に、イベント検出コードを挿入することです。

利点

- イベント検出プロセスに使用する SAP トランザクション情報にアクセスできます。
- トランザクションの適切なポイントにイベント検出コードを追加できます。
- 同期イベント検出を可能にします。
- SAP 機能への依存が限定されるため、保守や拡張が比較的容易です。
- 将来のイベントをサポートします。

欠点

- ユーザー出口が常にトランザクションの適切な位置にあるとは限りません。
- SAP 変更機能が必要になる場合があります。

バッチ・プログラム

バッチ・プログラムは、同じタイプの多数のイベント (例えば顧客注文)を起動する必要がある場合、あるいはビジネス・プロセスが長い処理時間を必要とする場合に役立ちます。この機構では、SAP 提供コードを変更する必要はありませんが、イベント検出の基準を評価する ABAP プログラムを使用する (作成する) 必要があります。

利点

- ほとんどのビジネス・プロセスに対して実装できます。
- イベントを正確に検出します。
- 実装が容易です。
- 実行時のリソースが問題である場合は、特定の時刻に実行するようにスケジュールできます。
- 将来のイベントをサポートします。

欠点

- 同期イベント検出はできません。
- SAP トランザクション情報は利用できません。
- 状態 (Create、Update または Delete) または状況の変更は、検出できない場合、あるいは検出が困難な場合があります。
- バッチ・プログラムを自動化するバックグラウンド・ジョブが作成されている場合、追加のタスクは保守および監視される必要があります。

ビジネス・ワークフロー

ビジネス・ワークフローは、複数のアプリケーション間でビジネス・タスクを統合できる、SAP アプリケーション内のクロス・アプリケーション・ツールです。このツールは、SAP アプリケーションの既存のビジネス機能を補足します。ビジネス・ワークフローを使用すると、必要なビジネス機能の特定の要件に合わせて、SAP の標準機能を改造できます。ビジネス・ワークフローは、アプリケーション内の各 SAP オブジェクトの定義を保管する、ビジネス・オブジェクト・リポジトリー (BOR) を使用します。

利点

- 同期イベント検出を可能にします。
- SAP のオブジェクト指向ビジネス・オブジェクト機能を利用して、イベントの検出を ABAP 機能モジュールにリンクします。
- 実装が容易です。

欠点

- すべてのビジネス・プロセスの SAP BOR に SAP オブジェクトが存在するとは限りません。

- SAP オブジェクトに対して SAP イベント (例えば Created や Deleted など) が存在しない場合があります。
- ビジネス・プロセス内のすべての変更を検出できるとは限りません。
- 適切な時刻にイベントを検出する柔軟性を提供できない場合があります。
- SAP 提供の機能に依存します。この機能は、SAP のバージョンによって異なる場合があります。

変更ポインター

変更ポインターは、変更文書を使用してイベントを検出する、ビジネス・ワークフローの関連機能です。変更文書は、一部のビジネス・プロセスに対して作成され、そのビジネス・プロセスに関するすべての変更が取り込まれるようにします。

利点

- 同期イベント検出を可能にします。
- 1 つのアダプター機能モジュールについて、1 つの SAP 変更だけで、すべてのビジネス・プロセスを処理できます。
- Logistics モジュールに対して一般的に利用できます。
- イベント検出プロセスに使用する SAP 変更ポインター情報にアクセスできます。
- ビジネス・プロセスに対して変更文書が既に使用されている場合は、最小限の作業でイベントを検出できます。

欠点

- ある程度の柔軟性は備えていますが、イベント検出の配置は SAP によって行われるため変更できません。
- 変更文書およびビジネス・ワークフロー環境について、詳しい知識が必要になります。
- SAP データ・エレメントの変更文書フラグをオンにするには、SAP 変更を行う必要があります。
- SAP の変更ポインター情報が、イベント検出プロセスには不十分である場合があります。

イベント検出機構の実装

サポートするビジネス・プロセス (例えば見積や販売注文) と、使用するイベント検出機構を決定したら、ビジネス・プロセスに対して機構を実装します。

注: イベント検出機構を実装する場合は、1 つのビジネス・プロセスのためのすべての機能を 1 つの機構でサポートすることをお勧めします。これにより、SAP アプリケーションでの影響を小さくし、イベント検出の管理を容易にすることができます。

以下のセクションでは、コネクターで実装される 4 種類のイベント検出機構の実装プロセスについて説明します。可能な場合には、サンプル・コードとともに例が提供されています。

コード拡張

コード拡張では、ABAP コードの一部をカスタム機能モジュールにカプセル化する必要があります。イベント検出コードは、処理をトランザクションから分離しておくため、機能モジュールとして記述します。トランザクションから使用されるすべての表や変数は、参照によってではなく、値で機能モジュールに渡される必要があります。

イベントを検索する際にビジネス・オブジェクトがロックされる影響を最小限にするために、通常、機能モジュールは更新タスク・モードで実行します。更新タスク・モードのプロセスで既に機能モジュールが呼び出されている場合は、不整合を避けるために、更新タスクは使用しないでください。

トランザクションでの影響を最小限にするため、機能モジュールは別の組み込みプログラム内に配置してください。組み込みプログラムを使用することで、SAP コードではなくカスタム・コードを変更できます。

イベント検出コードには、イベントのオブジェクトを識別するロジックが含まれています。例えば、販売注文トランザクションでは、さまざまなタイプの注文を処理しますが、必要な注文タイプは 1 つだけです。このロジックは、イベント検出コードに記述します。このイベント検出コードを配置するための一般的戦略は、データがデータベースにコミットされる直前に挿入することです。イベント検出コードを含んだ機能モジュールは、通常はビジネス・オブジェクトに対する機能グループの一部として作成します。

イベント検出のためにコード拡張を実装するには、以下の手順を実行します。

- Create、Update、または Delete の中から、サポートする動詞を決定します。これは、検出対象とするトランザクションを定義するために役立ちます。
- トランザクションのビジネス・オブジェクト・キーを決定します。このキーは、コネクターがデータベースからビジネス・オブジェクトを検索できるように、固有である必要があります。複合キーが必要な場合は、各キー属性と、それに対応する値を、名前と値のペアとしてトリガー時に指定できます。ビジネス・オブジェクトをポーリング時に作成すると、コネクターは自動的に属性に値を取り込みます。詳細については、276 ページの『名前と値のペアとしての複合キーのコーディング』を参照してください。
- トランザクション内の SAP 提供のユーザー出口が、イベントの検出に必要なすべての情報を持っているかどうか検査します。例えば、ユーザー出口は、それが挿入されているポイントよりも前にビジネス・オブジェクトがデータベースから削除されてしまうため、Delete 動詞を実装できない場合があります。
- ユーザー出口を使用できない場合は、イベント検出コードを配置する適切な場所を決定し、SAP 変更を使用してイベント検出コードを追加します。決定する際に使用したビジネス・オブジェクト・キーおよびその他の変数にアクセスできる場所を選択してください。

  将来のイベントに対するイベント検出コードを追加するほかに、将来のイベントに関する機能を実装する場合には、BASIS 管理者に連絡して、アダプター提供のバッチ・プログラム `/CWLD/SUBMIT_IN_FUTURE` を毎日 1 回実行するようにスケジュールしてください。

- ビジネス・プロセスのトランザクションによって実行されるコードで “commit work statement” を探して、ビジネス・プロセスを調べます。そのポイントで異なる属性の値を調査するには、ABAP デバッガーを使用します。
- イベントを検出するための基準を決定します。
- イベント検出コードを含んだ機能モジュールを作成します。
- 組み込みプログラムを作成し、それをトランザクションのコードに追加します。イベントを検出するために設計したすべてのシナリオをテストします。

次に示すステップでは、コード拡張イベント検出機構を使用して、サンプルの SAP Sales Quote を作成するプロセスを説明します。それに続くコードは、このプロセスの結果です。

1. SAP Sales Quote トランザクションを調べると、必要な Sales Quote 作成のビジネス処理をサポートするトランザクション VA21 が見つかります。
2. Sales Quote 番号を、固有キーにするように決定します。Sales Quote 番号は、表/フィールド VBAK-VBELN に保管されます。

   注: このイベントは単一の固有キーを使用するため、このコード例では OBJKEY パラメーターを使用してキーの値を渡しています。複合キーを使用するイベントをコーディングする例については、276 ページの『名前と値のペアとしての複合キーのコーディング』を参照してください。
3. トランザクション VA21 のトランザクション・フローには、文書保管プロセスの一部として、ユーザー出口が含まれています (Form Userexit_save_document)。トランザクションのこのポイントでは、ユーザー出口が実行されると、Quote 番号が入手できます。
4. ユーザー出口は他のビジネス・プロセスに属しているため、Sales Quote を他の文書のカテゴリーから区別するには、追加のコーディングが必要です。文書カテゴリーを判別するために、VBAK-VBTYP が使用可能です。Sales Quote は、SAP データベースに文書カテゴリー B として保管されます。
5. ユーザー出口に、組み込みプログラムを指す include ステートメントを追加します。
6. この時点で、組み込みプログラムと機能モジュールを作成する必要があります。

/CWLD/ADD_TO_QUEUE: 単一キー値の例

次のコード・フラグメントに、/CWLD/ADD_TO_QUEUE イベント・トリガーへの機能呼び出しを示します (単一キー値を使用)。

```
If VBAK-VBTYP = 'B'.
      C_OBJ_ORDER = 'SAP4_SalesQuote'.
         TMP_OBJKEY = XVBAK-VBELN.
    TMP_EVENT = 'Create'.

      CALL FUNCTION '/CWLD/ADD_TO_QUEUE'
            EXPORTING
                  OBJ_NAME                = C_OBJ_ORDER
                  OBJKEY                  = TMP_OBJKEY
                  EVENT                   = TMP_EVENT
                  GENERIC_RECTYPE = ''
            IMPORTING
                  RECTYPE                 = TMP_RECTYPE
            TABLES
                  EVENT_CONTAINER = TMP_EVENT_CONTAINER
```

```
            EXCEPTIONS
                  OTHERS                  = 1.

Endif.
```

/CWLD/ADD_TO_QUEUE_IN_FUTURE: 単一キー値の例

次のコード・フラグメントに、/CWLD/ADD_TO_QUEUE イベント・トリガーへの機能呼び出しを示します (単一キー値)。

```
DATA: DATE_IN_FUTURE LIKE SY_DATUM.

DATE_IN_FUTURE = VBAK-VDATU.

If VBAK-VBTYP = 'B'.
      C_OBJ_ORDER = 'SAP4_SalesQuote'.
         TMP_OBJKEY = XVBAK-VBELN.
    TMP_EVENT = 'Create'.

      CALL FUNCTION '/CWLD/ADD_TO_QUEUE_IN_FUTURE'
            EXPORTING
                  OBJ_NAME               = C_OBJ_ORDER
                  OBJKEY                 = TMP_OBJKEY
                  EVENT                  = TMP_EVENT
                  VALID_DATE             = DATE_IN_FUTURE
            IMPORTING
                  RECTYPE                = TMP_RECTYPE
            TABLES
                  EVENT_CONTAINER        = TMP_EVENT_CONTAINER
            EXCEPTIONS
                  OTHERS                 = 1.

Endif.
```

名前と値のペアとしての複合キーのコーディング

イベントのキーが単一のキー・フィールドでなく、複数のフィールドから構成されている場合には、各キー属性の名前と、それに対応する値を指定できます。属性の名前を指定するので、属性には、コネクターが値を取り込んだり、検索に使用するために、IsKey としてマーク付けする必要はありません。

名前と値のペアを複数指定すると、コネクターは、アプリケーションから完全なオブジェクトを検索するために、それが作成したビジネス・オブジェクトに複数の属性の値を設定します。名前と値のペアを 1 つだけ指定すると、コネクターは、IsKey としてマークされた最初の属性ではなく、指定した属性の値を設定します。

IDoc Handler は名前と値のペアを使用しないため、/CWLD/IDOC_HANDLER を使用する場合は、名前と値のペアを指定しないことが重要です。詳細については、265 ページの『IDoc Handler および retrieve 動詞』を参照してください。

次に示すステップでは、複合キーで 3 つのフィールドを使用する、サンプルの SAP Sales Quote を作成するプロセスを説明します。それに続くコードは、このプロセスの結果です。

1. アダプターに付属の構造体 (/CWLD/NAME_VALUE_PAIRS) に基づいて、ローカルの name_value_pairs 内部表を作成します。この構造体には列が 2 つあります。ATTR_NAME および ATTR_VALUE です。
2. 機能モジュール /CWLD/ADD_TO_QUEUE または /CWLD/ADD_TO_QUEUE_IN_FUTURE を呼び出す前に、キー属性の名前とそれらの値を内部表に追加するコードを記述します。

3. 機能モジュール /CWLD/ADD_TO_QUEUE または /CWLD/ADD_TO_QUEUE_IN_FUTURE を変更します。
   - キーの値を渡すために OBJKEY パラメーターは使用しないので、このパラメーターの行をコメント化します。
   - 複合キーの値を渡すために NAME_VALUE_PAIRS 表を使用するので、この表の行を追加します。
4. 各イベント・キーは、トリガー機能によって自動的にフォーマットされます。フォーマットには、次の構文が使用されます。

   attribute1=value1|Cx|*attribute2=value2*|Cx|[*attributeN=valueN*|Cx|]

   は、次のように説明されます。

属性	キー属性の名前 (大文字小文字が区別されません)
値	キー属性の値 (大文字小文字が区別されます)
|Cx|	それぞれの名前と値のペアの終了文字 (名前と値のペアを 1 つしか指定しない場合でも使用します)

コード内で名前と値のペアを指定する順序は、ビジネス・オブジェクト内での属性の順序と一致している必要はありません。ただし、ビジネス・オブジェクト内に存在しない属性を指定すると、イベントは失敗します。

次に示すコード・フラグメントでは、トリガー時に、顧客番号、販売組織、および流通経路を、名前と値のペアとして表 KNVV に指定しています。機能モジュール /CWLD/ADD_TO_QUEUE のコードで、次の 2 行が強調表示されています。

- 値を OBJKEY パラメーターに渡す行 (コメント化されています)
- NAME_VALUE_PAIRS 表を指定する行

```
DATA:  name_value_pairs LIKE /cwld/name_value_pairs OCCURS 5 with header line.

MOVE   'CustomerId' TO name_value_pairs-attr_name.
MOVE   knvv-kunnr TO name_value_pairs-attr_value.
APPEND name_value_pairs.

MOVE   'SalesOrg' TO name_value_pairs-attr_name.
MOVE   knvv-vkorg TO name_value_pairs-attr_value.
APPEND name_value_pairs.

MOVE   'DistributionChannel' TO name_value_pairs-attr_name.
MOVE   knvv-vtweg TO name_value_pairs-attr_value.
APPEND name_value_pairs.
If VBAK-VBTYP = 'B'.
      C_OBJ_ORDER = 'SAP4_SalesQuote'.
         TMP_OBJKEY = XVBAK-VBELN.
    TMP_EVENT = 'Create'.

      CALL FUNCTION '/CWLD/ADD_TO_QUEUE'
       EXPORTING
                 OBJ_NAME               = C_OBJ_ORDER
*                OBJKEY                 = TMP_OBJKEY
                 EVENT                  = TMP_EVENT
            GENERIC_RECTYPE = ''
            IMPORTING
                 RECTYPE                 = TMP_RECTYPE
            TABLES
                 NAME_VALUE_PAIRS = name_value_pairs
                 EVENT_CONTAINER = TMP_EVENT_CONTAINER
```

```
            EXCEPTIONS
                  OTHERS                  = 1.

Endif.
```

バッチ・プログラム

イベント検出機構としてバッチ・プログラムを実装する場合は、データベース情報を評価する ABAP プログラムを作成する必要があります。ABAP プログラムの実行時に、そのプログラムでの基準が満たされると、イベントが起動されます。

イベント検出のためにバッチ・プログラムを実装するには、以下の手順を実行します。

- Create、Update、または Delete の中から、サポートする動詞を決定します。
- トランザクションのビジネス・オブジェクト・キーを決定します。ビジネス・オブジェクト・キーは、そのビジネス・オブジェクトをデータベースから検索できるように、固有である必要があります。複合キーが必要になる場合があります。例えば、複数の工場における材料の在庫レベル用バッチ・プログラムを実装するには、キー `Material_key + Plant_key` が必要になります。
- イベントを検出するための基準を決定します。ビジネス・オブジェクトに関連付けられているデータベース表について知識を持っている必要があります。
- イベントを生成するための基準を組み込んだ ABAP プログラムを作成します。
- 将来のイベントに対するイベント検出コードを追加するほかに、将来のイベントに関する機能を実装する場合には、BASIS 管理者に連絡して、アダプター提供のバッチ・プログラム `/CWLD/SUBMIT_IN_FUTURE` を毎日 1 回実行するようにスケジュールしてください。

  将来のイベントに関する機能を実装するサンプル・コードについては、 276 ページの『/CWLD/ADD_TO_QUEUE_IN_FUTURE: 単一キー値の例』を参照してください。
- バッチ・プログラムを自動化するためにバックグラウンド・ジョブが必要かどうかを判断します。バックグラウンド・ジョブは、バッチ・プログラムをオフピーク時に実行する必要を生じるような、システム・リソースへの影響がある場合に役立ちます。

次に示すステップでは、今日の日付で作成されたすべての Sales Quote について、イベントを検出するバッチ・プログラムを作成するプロセスを説明します。それに続くコードは、このプロセスの結果です。

1. Create 動詞をサポートすることに決定します。
2. Quote 番号を、イベントを検索するための固有キーにするように決定します。
3. 作成日 (`VBAK-ERDAT`) および文書カテゴリー (`VBAK-VBTYP`) をチェックする必要があります。

次に示すサンプル・コードでは、SAP sales quote がバッチ・プログラムとしてサポートされます。

```
REPORT ZSALESORDERBATCH.

tables: vbak.

parameter: d_date like sy-datum default sy-datum.
```

```
data: tmp_key like /CWLD/LOG_HEADER-OBJ_KEY,
      tmp_event_container like swcont occurs 0.

" retrieve all sales quotes for today's date
" sales quotes have vbtyp = B
select * from vbak where erdat = d_date
                     and vbtyp = 'B'.

  tmp_key = vbak-vbeln.

  CALL FUNCTION '/CWLD/ADD_TO_QUEUE'
       EXPORTING
            OBJ_NAME        = 'SAP4_SalesQuote'
            OBJKEY          =  tmp_key
            EVENT           = 'Create'
            GENERIC_RECTYPE = ''
       IMPORTING
            RECTYPE         = r_rectype
       TABLES
            EVENT_CONTAINER = tmp_event_container.

  write: / vbak-vbeln.
endselect.
```

ビジネス・ワークフロー

ビジネス・ワークフローは、論理的に関連するビジネス・オペレーションのセットまたはシーケンスです。ワークフロー内の処理ロジックがイベントを検出します。ビジネス・ワークフロー・イベント検出機構は、オブジェクトおよびそれらに関連付けられた属性、メソッド、およびイベントのディレクトリーを格納する SAP ビジネス・オブジェクト・リポジトリー (BOR) に依拠しています。

イベント検出のためにビジネス・ワークフローを実装するには、以下の手順を実行します。

- 必要な機能をどの SAP ビジネス・オブジェクトが表しているかを判断します。イベントがワークフローを起動、開始、または終了するかどうかを検査します。適切なビジネス・オブジェクトを検索するには、ビジネス・オブジェクト・ビルダー (トランザクション SWO1) を使用します。
- この SAP ビジネス・オブジェクトのサブタイプを作成します。サブタイプは、このスーパータイプのプロパティーを継承し、またカスタマイズして使用することができます。
- サブタイプをカスタマイズして、ビジネス・オブジェクトのイベント (例えば CREATED、CHANGED、および DELETED) をアクティブにします。

次に示す SAP Sales Quote の例は、ビジネス・ワークフローを使用してイベント・トリガーを実装するために使用できます。

1. BOR を検索して、適切な Sales Quote ビジネス・オブジェクトを探します。検索は、簡略説明フィールドと、文字列 '*quot*' を使用して実行します。BUS2031 (Customer Quotes) は、戻されるビジネス・オブジェクトの 1 つです。
2. BUS2031 をさらに調べることで、キー・フィールドが CustomerQuotation.SalesDocument (VBAK-VBELN) であることがわかります。
3. BUS2031 のサブタイプは、以下のエントリーを使用して作成します。

オブジェクト・タイプ — ZMYQUOTE

イベント — SAP4_SalesQuote

名前 — SAP4 Sales Quote

説明 — SAP 4 Sales Quote Subtype の例

プログラム — ZMYSALESQUOTE

アプリケーション — V

4. イベント検出機構は、Event Linkage 表にエントリーを追加することでアクティブ化されます (トランザクション SWE3)。作成イベントは、以下のエントリーを使用してアクティブ化します。

   オブジェクト・タイプ — ZMYQUOTE

   イベント — SAP4_SalesQuote

   受信側機能モジュール — /CWLD/ADD_TO_QUEUE_DUMMY

   受信側タイプ機能モジュール — /CWLD/ADD_TO_QUEUE_WF

注: 受信側および受信側タイプ機能モジュール (Function Module、FM) は、/CWLD/ADD_TO_QUEUE を指します。DUMMY 機能モジュールは、SAP アプリケーションが両方のフィールドに値を取り込むことを要求する場合があるために使用されています。WF 機能モジュールは、SAP 標準インターフェースを /CWLD/ADD_TO_QUEUE で使用されるものに変換します。

ビジネス・ワークフロー・イベント検出機構が作成され、アクティブ化されました。作成されたすべての SAP Customer Quotes を検出するようにセットアップされています。

変更ポインター

変更ポインターは、変更文書を使用する、より実装の難しいイベント検出機構の 1 つです。アプリケーション・リンク・イネーブリング (ALE) テクノロジーとともに、SAP のビジネス・オブジェクト・リポジトリー (BOR) が使用されます。変更文書は、少なくとも 1 つのデータベース表が割り当てられているビジネス・ドキュメント・オブジェクトを常に参照します。表内のデータ・エレメントが変更文書を必要としているとしてマーク付けされており、表がビジネス・ドキュメント・オブジェクトに割り当てられている場合、データ・エレメントで定義されているフィールドの値が変更されると、変更文書が生成されます。変更は、表 CDHDR および CDPOS にキャプチャーされ、イベント検出のために使用されます。

イベント検出のために変更ポインターを実装するには、以下の手順を実行します。

- トランザクション BD61 で、グローバルな Change pointer フラグをアクティブ化します。
- SAP 機能モジュール CHANGE_POINTERS_CREATE を変更して、機能モジュール /CWLD/EVENT_FROM_CHANGE_POINTR の呼び出しを組み込みます。
- Create、Update、または Delete の中から、サポートする動詞を決定します。

- SAP ビジネス・プロセス (トランザクション) が変更文書を利用しているかどうかを、次のように検査します。
  - トランザクションの「Environment」メニューに「Change function」が存在するかどうか。「移動」をクリックし、次に「統計」をクリックした場合はどうか。
  - トランザクションでデータを変更した場合、変更を反映した新しいエントリーが表 CDHDR に存在するかどうか?
  - トランザクションに関連付けられているデータベース表で、Change Document フラグが設定されているデータ・エレメントが存在するかどうか?

  これらの質問で答えが 1 つでも Yes の場合は、トランザクションで変更文書が使用されています。
- Change Document フラグを設定したデータ・エレメントで、イベントを検出するために必要なすべての情報を取り込むかどうかを決定します。Change Document フラグを変更すると、SAP 提供のオブジェクトが変更されるため、この変更はお勧めできません。
- トランザクションのビジネス・オブジェクト・キーを決定します。ビジネス・オブジェクト・キーは、そのビジネス・オブジェクトをデータベースから検索できるように、固有である必要があります。複合キーが必要になる場合があります。これは通常、表/フィールド CDHDR-OBJECTID になります。
- イベントを検出するための基準を決定します。区別する基準として、主に表/フィールド CDHDR-OBJECTCLAS を使用します。イベントを検出するために、CDPOS-TABNAME を使用することもできます。
- 機能モジュール /CWLD/EVENT_FROM_CHANGE_POINTR を更新して、イベントを検出するロジックを加えます。

次に示す SAP Sales Quote の例は、変更ポインターを使用してイベント・トリガーを実装するために使用できます。

1. Update 動詞をサポートすることに決定します。Sales Quote 作成トランザクションを調査すると、Create 動詞はこの機構全体を通じて検出されないことが明らかになります。
2. Sales Quote に関してビジネスの検査を実行すると、次の結果が得られます。
   - トランザクション VA22 の「Environment」メニューで、「Change function」が利用できます。
   - Sales Quote を変更すると、表 CDHDR に新しいエントリーが作成されます。
   - 表 VBAP を見ると、フィールド ZMENG に Change Document フラグが立っています。
3. この例では、データ・エレメントの評価は行われません。
4. Sales Quote 番号を、CDHDR-OBJECTID で固有キーにするように決定します。
5. CDHDR-OBJECTCLAS には、区別する主な基準である値 VERKBELEG が設定されています。Sales Quote のみが選出される必要があります。コードはヘッダー表の TCODE フィールドを検査しますが、VBAK 表で適切な検索が実行される必要があります。

以下のサンプル・コードは、/CWLD/EVENT_FROM_CHANGE_POINTR に追加されます。

```
when 'VERKBELEG'.
   data: skey    like /cwld/log_header-obj_key,
      s_event like swetypecou-event,
      r_genrectype like swetypecou-rectype,
      r_rectype like swetypecou-rectype,
      t_event_container like  swcont occurs 1 with header line.

   " Quick check. Should check document category (VBTYP) in VBAK.
   check header-tcode = 'VA22'.

   " Event detection has started
   perform log_create using c_log_normal c_blank
                            c_event_from_change_pointer c_blank.

   " Set the primary key
   skey = header-objectid.

   " Set the verb
   s_event = c_update_event.

   " Log adding the event to the queue
   perform log_update using c_information_log text-i44
                           'SAP4_SalesQuote' s_event skey.

      " Event detection has finished.
      perform log_update using c_finished_log c_blank
                               c_blank c_blank c_blank.

   call function '/CWLD/ADD_TO_QUEUE'
      exporting
            obj_name                 = 'SAP4_SalesQuote'
            objkey                   = skey
            event                    = s_event
            generic_rectype          = r_genrectype
      importing
            rectype                  = r_rectype
      tables
            event_container          = t_event_container
      exceptions
            others                    = 1.
```

第 23 章 ABAP Extension Module の管理

IBM WebSphere BI Station ツール (トランザクション /n/CWLD/HOME) により、Adapter for mySAP.com のイベント処理用の保守が可能になります。このツールを使用して、SAP アプリケーションへの接続を保守することもできます。コネクターのログ・ファイルや、SAP Gateway Service 接続を表示できます。また、アーカイブされたオブジェクトをコネクター・ログから再処理したり、処理待ちのイベントを表示したり、特定のイベントを後で処理するようにスケジュールしたり、イベントをアーカイブ表から再サブミットしたり、削除したりすることができます。

この章の内容は以下のとおりです。

コネクター・ログ・ファイルの管理

SAP アプリケーションのコネクター・ログには、Create または Update 操作などの、コネクターに関連するすべてのイベントおよびエラーや、あるいはイベント・キューに到着したイベントが、発生日時の新しいものから順に表示されます。ログ・ファイルの各ログ・エントリーには、日付、時刻、およびイベントがリストされます。ログ・ファイルは、問題のトラブルシューティングをはじめるための、良い手掛かりになります。

ログ・オプションの設定

グローバル設定およびユーザー設定は、表示するエントリーの数およびデータのタイプや、コネクター・ログ・ファイルに記録する詳細レベルに応じて設定できます。IBM WebSphere BI Station を使用してコネクターのロギング・レベルを設定するには、「Configuration」タブをクリックし、「Logging Level」から 0 から 3 のレベルを選択します。

ロギングには、次の 4 つのレベルがあります。

- 0 — オフ
- 1 — 警告およびエラーのみを記録
- 2 — すべてのイベントを最少の情報で記録
- 3 — 各イベントを、すべてのビジネス・オブジェクトのすべての属性も含めて詳細に記録

注: ロギング・レベル 0 はお勧めできません。実動システムでは、ロギング・レベル 1 を推奨します。開発システムまたはデバッグ・システムでは、ロギング・レベル 3 を推奨します。

ログの表示

最近に処理したオブジェクトと、それに関連する詳細を表示するには、コネクター・ログを表示します。SAP アプリケーションでコネクター・ログを表示するには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。
2. 「Management」タブをクリックし、次に「Log」ボタンをクリックします。

ログ・エントリーには、日付、時刻、およびイベントが表示されます。各エントリーは、次のように色分けされています。

緑 — 正常なイベントを示します。

黄 — 警告メッセージを示します。

赤 — エラーを示します。

白 — アーカイブされたオブジェクトを示します。

マジェンタ (SAP アプリケーション GUI バージョン 4.6 以前) またはオレンジ (SAP アプリケーション GUI バージョン 4.6 以降) で表示されたエントリーでは、イベントの開始および終了に関する情報が提供されます。任意の矢印をクリックすると、そのビジネス・オブジェクトに関する SAP の表示トランザクションにリンクします。

ログ詳細のフィルター操作

各イベントについて表示される詳細の量を変更できます。表示レベルを変更するには、必要な詳細のレベルに応じて、「More Details」または「Fewer Details」ボタンをクリックします。

表示されるデータの量が現在必要としている量よりも多い場合は、表示される情報を制限します。例えば、ビジネス・オブジェクトをユーザー、名前、日付、またはログ・エントリー番号で表示できます。

1. 「Filter Data」ボタンをクリックします。
2. ログ・ファイルにフィルターを掛けるための適切なフィールドに入力します。
3. 「Filter」をクリックします。

「Configuration」タブで、一度に表示されるログ・エントリーの数、およびデフォルトのロギング表示レベルについて、ユーザー設定を設定できます。

アーカイブされたオブジェクトの再処理

失敗した、あるいはアーカイブされたオブジェクトを、コネクター・ログ・ファイルから再処理できます。失敗したオブジェクトとは、正常に処理できなかった SAP 内のオブジェクトのことです。アーカイブされたオブジェクトとは、処理されずにアーカイブされるように構成したオブジェクトのことです。どちらの場合にも、コードの特定の位置にブレークポイントを設定することで、オブジェクトを手動で段階ごとにたどっていくことができます。Dynamic Transaction および IDoc オブジェクトについては、トランザクションの画面を段階ごとにたどることができます。

ブレークポイントは、以下の位置の直前に設定できます。

- 機能モジュール /CWLD/RFC_DO_VERB_NEXTGEN が呼び出される位置
- 最初の機能モジュールが実行される位置
- メインの処理ステップが実行される位置

  ブレークポイントの配置は、オブジェクトのタイプによって異なります。
  - Dynamic Retrieve — Select ステートメントの直前
  - Dynamic Transaction — Call Transaction ステートメントの直前
  - IDoc — IDoc 機能モジュールが呼び出される位置の直前
  - BAPI — BAPI-Wrapper 機能モジュールが呼び出される位置の直前

Dynamic Transaction および IDoc オブジェクトでは呼び出しトランザクションが使用されるため、これらのオブジェクトに対する画面処理を表示できます。表示には、以下のオプションがあります。

- すべての画面
- エラーのある画面のみ
- 画面を表示しない

Dynamic Retrieve および BAPI オブジェクトでは、画面処理は使用されません。

アーカイブ対象オブジェクトの構成

デフォルトでは、いずれのアーカイブ・オプション (A、X、または N) も動詞のアプリケーション固有情報内に指定されていない ABAP Extension Module ビジネス・オブジェクトは、障害時にアーカイブされます。つまり、処理によって 0 または 21 以外の戻りコードが発生した場合、ビジネス・オブジェクトは、/cwld/obj_arc_h 表および /cwld/obj_arc_i 表にアーカイブされます。

重要: これらのアーカイブ表は次第にサイズが大きくなるため、データベースの全体的なパフォーマンスに影響を与えないように、内容を定期的に削除またはアーカイブする必要があります。

アーカイブ動作を変更する場合は、ビジネス・オブジェクトの動詞レベルで行います。つまり、各ビジネス・オブジェクトについて、アーカイブ活動は動詞によって変化します。オブジェクトがアーカイブされる方法を指定するには、動詞のアプリケーション固有情報内で、以下の構文を使用します。

```
AppSpecificInfo = connectormodule.class, ArchiveParameter: ABAPhandler
```

ArchiveParameter の説明を以下に示します。

A オブジェクトが最初に SAP アプリケーションに入ったときにアーカイブします。

N オブジェクトのアーカイブを抑止します。失敗した場合でも、オブジェクトはアーカイブされません。

X 即時にオブジェクトをアーカイブします。ログは、処理が終了したことを知らせる警告メッセージで更新されます。成功コードがコネクターに戻されるので、要求側統合ブローカーは正常に処理を行います。

一度に複数のパラメーターを指定できます。A および X アーカイブ・パラメーターを指定すると、IBM WebSphere BI Station 内の再処理ツールへのリンクの付いたエ

ントリーがログ表に追加されます。アーカイブされたオブジェクトの状況は、アーカイブされたビジネス・オブジェクトのエントリーの下の行に入力されます。

次に示す例では、Dynamic Transaction オブジェクトをアーカイブし、ログ表にエントリーを追加します。

```
AppSpecificInfo = sap.sapextensionmodule.VSapBOHandler,
A:/CWLD/DYNAMIC_RETRIEVE
```

次に示す例では、IDoc オブジェクトの SAP4_Order Create を SAP アプリケーションに入った時点でアーカイブし、次にそのオブジェクトの処理を停止します。

```
AppSpecificInfo = sap.sapextensionmodule.VSapBOHandler,
X:/CWLD/ORDER:ORDER_C1
```

注: 実稼働環境では、ビジネス・オブジェクトとそれらのすべての動詞に対して、N パラメーターだけを使用してください。統合ブローカーが InterChange Server Express であるため、System Manager を使用するのは、ビジネス・オブジェクトの再処理、および再サブミットのときだけにしてください。ご使用の SAP アプリケーション内で、IBM WebSphere BI Station の再処理ツールを*使用しない*でください。

reprocessing tool の使用

Reprocessing Tool では、ABAP Debugger を使用して、SAP の WebSphere ビジネス・オブジェクトを再処理することができます。

重要: このツールを使用するのは、開発環境のみにする必要があります。

- 開発およびテスト中には、特定のビジネス・オブジェクトを SAP アプリケーションへの到着時にアーカイブするように指定し、次にこれらのビジネス・オブジェクトを ABAP Debugger で処理できます。
- 同じビジネス・オブジェクトを、必要に応じて何度でも処理することができます。ビジネス・オブジェクトは、削除されるまではいつでも再処理することができます。

アーカイブされたオブジェクトを再処理するには、以下の手順を行います。

1. SAP アプリケーションで、コネクターのログに移動します。
2. アーカイブされたオブジェクトのエントリーをダブルクリックします。

   「CW reprocess objects from archive tables」ウィンドウが表示されます。その「Archived Object Number」フィールドには、オブジェクト番号が取り込まれます。
3. 設定するブレークポイントについて、「Set Breakpoint」チェック・ボックスをクリックします。必要な場合には、複数のブレークポイントを設定できます。
4. Call Transaction を使用するオブジェクトについては、画面処理オプションを選択できます。
5. 「Execute」をクリックします (F8)。

   ABAP Debugger が、アーカイブされたオブジェクトとともに呼び出されます。
6. ABAP Debugger を使用して、オブジェクトを段階ごとにたどっていきます。

IBM WebSphere BI Station の Reprocessing Tool に手動でアクセスするには、「ツール」タブで「Reprocess Object」をクリックします。表示されたフィールドに、適切な値を入力します。

アーカイブされたオブジェクトの削除

アーカイブされたオブジェクトを、SAP アプリケーションからアダプター提供の Delete Archive Objects ツールを使用して削除することができます。このツールでは、アーカイブされたオブジェクトを手動で削除できます。アーカイブされたオブジェクトを削除すると、コネクター・ログにあるそのオブジェクトのエントリーは、新しい状況で更新されます。オブジェクトは物理的に削除され、オブジェクトの状況だけが参照のために保持されます。

IBM WebSphere BI Station (トランザクション `/n/CWLD/HOME`) を使用して、アーカイブされたオブジェクトを削除するには、以下の操作を実行します。

1. 「Maintenance」タブで、「Del Object Archive」ボタンをクリックします。
2. 削除するオブジェクトを指定します。オブジェクトは、以下の基準に基づいて削除できます。
   - アーカイブ番号
   - オブジェクト名
   - ユーザー (コネクター名)
   - 作成日
   - 状況
3. 「Execute」をクリックします (F8)。

オブジェクトを自動的に削除するようにアーカイブ・オブジェクト・プログラムをスケジュールするには、BASIS 管理者に連絡し、レポート `/CWLD/DELETE_OBJECT_ARCHIVE` をスケジュールします。このレポートは、バックグラウンド・プロセスとして実行するようにスケジュールすることができます。

イベント・ログの切り捨てのセットアップ

SAP はコネクターのアクティビティーのイベント・ログを保持しています。このログは、時間の経過とともに、多くのディスク・スペースを占有するようになる可能性があります。ディスク・スペースを節約するために、このログを自動的に切り捨てるように設定できます。自動切り捨てを設定すると、デフォルトでは、SAP は切り捨てられたエントリーを、このジョブをセットアップしたユーザーのデフォルト・プリンターに出力するため、印刷オプションを制御する必要が生じる場合があります。

ログを手動で切り捨てるには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション `/n/CWLD/HOME`)。
2. 「Maintenance」タブをクリックします。
3. 「Online」セクションで、「Delete Log」をクリックします。
4. 該当するフィールドに入力します。
5. 「Execute」ボタンをクリックします (F8)。

イベント・ログの自動切り捨てをスケジュールするには、切り捨てオプションをセットアップし、BASIS 管理者に連絡してレポート /CWLD/DELETE_LOG をスケジュールします。

重要: このレポートは、定期的に実行することをお勧めします。

SAP gateway service 接続のモニター

コネクターと SAP アプリケーションとの間の SAP gateway service 接続をモニターすることができます。各エントリーには、コネクター・ホスト名、ユーザー名、および接続状況などの情報が表示されます。

SAP Gateway Service 接続をモニターするには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。
2. 「Management」タブをクリックし、次に「Gateway」をクリックします。
3. 詳細を表示するには、サーバー名をクリックします。

コネクターのシャットダウン

コネクターを停止する方法は、コネクターが始動された方法によって異なります。

- **Windows**:
  - コネクター用の別個の「コンソール」ウィンドウを作成する始動スクリプトを起動できます。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。
  - 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。
- **i5/OS**:
  - コンソールを使用して、または QSHELL で「submit_adapter.sh」スクリプトを使用してコネクターを始動した場合は、次の 2 つの方法のうちの 1 つを使用してコネクターを停止できます。
  - WebSphere Business Integration Server Express Console がインストールされている Windows システムから、**「IBM WebSphere Business Integration Express」>「Toolset Express」>「管理」>「コンソール」**を選択します。次に、OS/400 または i5/OS システムの名前または IP アドレス、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。リストから E メール・アダプターを選択して、「停止」ボタンを選択します。CL コマンド WRKACTJOB SBS (QWIBSVR44) を使用して Server Express 製品に対するジョブを表示します。リストをスクロールして、コネクターのジョブ記述に一致するジョブ名を持つジョブを探し出します。例えば、E メール・コネクターの場合のジョブ名は QWBIEMAILC です。このジョブに対してオプション 4 を選択し、F4 を押して ENDJOB コマンドのプロンプトを取得します。次に、オプション・パラメーターとして *IMMED を指定し、Enter を押します。

    注: QWBISVR44 サブシステムが終了すると、コネクターは終了します。

– QSHELL から start_connName.sh スクリプトを使用してアダプターを始動した場合は、F3 を押してコネクターを終了しま
す。/QIBM/ProdData/WBIServer44/bin ディレクトリーにあるスクリプト *stop_adapter.sh* を使用して、エージェントを停止することもできます。

- **Linux**:

  コネクターはバックグラウンドで実行されるので、個別のウィンドウはありません。代わりに、以下のコマンドを実行してコネクターを停止します。

  ```
  connector_manager -stop connName
  ```

  ここで、*connName* はコネクターの名前です。

イベント・キューの保守

発信側の現在のイベント・キューで、コネクターによって処理されていないイベントの有無を検査できます。

1. IBM WebSphere BI Station に移動します (トランザクション /n/CWLD/HOME)。
2. 「Management」タブをクリックし、次に「Current Events」をクリックします。
3. 「Execute」ボタンをクリックして (F8)、現在のイベント・キューを表示します。

表示されるイベント・エントリーの数を制限するには、「Current Event Selection」セクションの該当するフィールドに入力します。例えば、特定のビジネス・オブジェクトについて表示されるエントリーを制限するには、「Object Name」フィールドにビジネス・オブジェクト名を入力します。ビジネス・オブジェクト名の正確な構文が不明な場合は、「Object Name」フィールドをクリックし、矢印ボタンをクリック (F4) してから、適切なビジネス・オブジェクト名を選択します。

イベントの詳細について表示するには、イベント・フィールドをダブルクリックします。通常の条件下では、イベントは数秒ごとに選出されます。イベントが表示されたら、そのイベントはコネクターによって処理されていません。これは、コネクターが実行されていないことを示している場合があります。

イベント・キューの、可能なイベント状況値のリストを次に示します。

P — プレキュー	イベントが起動されると、ビジネス・オブジェクトがロックされるかどうかはまだ決定されていないため、状況は最初はプレキュー (P) に設定されます。
L — ロック	SAP でユーザーがビジネス・オブジェクトを作成または更新すると、そのビジネス・オブジェクトに対してロックが発行されます。ビジネス・オブジェクトがデータベースにコミットされると、SAP はロックを解除します。ビジネス・オブジェクトがロックされている間にイベントが起動されると、ロックが解除されるまでは、そのイベントはロック (L) の状況でイベント・キューに残されます。
Q — キュー	ビジネス・オブジェクトのロックが解除されると、状況はキュー (Q) に切り替わり、イベントはコネクターによって選出される準備が整います。イベントは、検索の確認が受信されるまで、この状況にとどまります。

R — 検索	ビジネス・オブジェクトが検索されると、イベント・キューの中で R のマークが付けられます。イベントは、イベントの処理が終了するまでキューに残っています。

アーカイブ表の保守

IBM WebSphere BI Station ツールを使用すると、アーカイブ表を表示して、アーカイブされたイベントの状況を判別できます。この表では、統合ブローカーがサブスクライブされた際に、ポーリングの再実行を依頼する必要のあるイベントを識別できます。

アーカイブ表を表示するには、以下の手順を行います。

1. IBM WebSphere BI Station に移動します (トランザクション `/n/CWLD/HOME`)。
2. 「Management」タブをクリックし、次に「Archived Events」をクリックします。
3. 「Execute」ボタンをクリックして (F8)、アーカイブ・キューを表示します。

表示されるアーカイブ・エントリーの数を制限するには、「Archived Event Selection」セクションの該当するフィールドに入力します。例えば、特定のビジネス・オブジェクトについて表示されるエントリーを制限するには、「Object Name」フィールドにビジネス・オブジェクト名を入力します。ビジネス・オブジェクト名の正確な構文が不明な場合は、「Object Name」フィールドをクリックし、矢印ボタンをクリックしてから、適切なビジネス・オブジェクト名を選択します。

イベントの詳細について表示するには、イベント・フィールドをダブルクリックします。アーカイブ表の、可能なイベント状況値のリストを次に示します。

0 — 成功	コネクターはイベントを正常に処理し、ビジネス・オブジェクトを統合ブローカーに送信しました。
1 — SAP 内でエラー検出	コネクターは、このイベントに関して SAP 内でビジネス・オブジェクトを検索している間にエラーを検出しました。
2 — サブスクライブなし	このイベント用のビジネス・オブジェクトと動詞の組み合わせには、いずれの統合ブローカーもサブスクライブされませんでした。
3 — Java でエラー検出	コネクターは、次のいずれかを実行している間にエラーを検出しました。 • SAP からのビジネス・オブジェクトの受信 • SAP ビジネス・オブジェクトの SAP 用 WebSphere ビジネス・オブジェクトへの変換 • ビジネス・オブジェクトのメッセージ・キューへの挿入
4 — 再キューイングの最大回数	イベントが、再キューイング定数 `c_maximum_requeue` で指定された最大回数 (通常は 100) を超えて再キューイングされました。イベントは、そのビジネス・オブジェクトがロックされている場合に再キューイングされます。
5 — 複数イベント	一部のビジネス・オブジェクトでは、イベント表内の単一イベントによって検索時に複数のイベントが作成されます。元の単一イベントはビジネス・オブジェクトを作成しないため、このイベント状況を使用してアーカイブされます。
6 — イベント削除	ユーザーがイベントを手動でイベント表から削除しました。

アーカイブ表からのイベントの再サブミット

イベントは、再処理するためにアーカイブ表からイベント・キューに再サブミットすることができます。アーカイブ表でのイベントの処理方法に応じて、単一イベントを再サブミットするか、複数のイベントを再サブミットするかを選択できます。イベントを再サブミットすると、イベントはアーカイブ表からイベント表に移動されるのみで、イベント分配、イベント制限、あるいはイベント優先順位による処理は行われないことに注意する必要があります。「Archived Events」ウィンドウで、以下のステップを実行します。

1. 「Execute」ボタンをクリックして (F8)、アーカイブ・キューを表示します。
2. 再サブミットするイベントを選択します。
3. 「Resubmit」ボタンをクリックするか、または「Archive Entry」メニューで「Resubmit」をクリックします (F8)。

状況メッセージが表示されます。イベントとその新しい状況を調べるには、コネクター・ログを表示します。

アーカイブ表からのイベントの削除

アーカイブ・イベントは、手動で削除したり、自動的に削除されるようにスケジュールすることができます。

手動でアーカイブ・イベントを削除するには以下の操作を実行します。

1. IBM WebSphere BI Station に移動します (トランザクション `/n/CWLD/HOME`)。
2. 「Maintenance」タブをクリックします。
3. 「Online」セクションで、「Delete Event Archive」をクリックします。
4. 該当するフィールドに入力します。
5. 「Execute」ボタンをクリックします (F8)。

アーカイブ・イベントの自動削除をスケジュールするには、BASIS 管理者に連絡し、レポート `/CWLD/TRUN_EVENT_ARCHIVE_TAB` をスケジュールします。

第 24 章 ABAP Extension Module のアップグレード

この章では、ABAP Extension Module のアップグレード・プロセスについて説明します。説明に際しては、明示的に指示されている場合を除き、コネクターまたはすべてのオブジェクトについて、リポジトリー定義を変更していないことが想定されています。この章では、コネクターの ABAP コンポーネントを中心に説明します。

この章の内容は以下のとおりです。

- 『SAP R/3 の新しいバージョンでのアップグレード』
- 294 ページの『ABAP Handler のアップグレード』
- 297 ページの『アップグレード考慮事項』

アップグレードする際には、使用する SAP のバージョンに対応する最新の ABAP Extension Module コンポーネントを用意しておく必要があります。アップグレード・プロセスの目標は、ABAP Handler 開発を最新の ABAP Extension Module コンポーネントに対応させることです。

ABAP Extension Module のアップグレードには、次の 2 種類のシナリオがあります。

- アダプター提供の ABAP Handler を含む SAP システムをアップグレードする場合

  例えば、SAP バージョン 4.0 システムが稼働しており、これを SAP バージョン 4.6 にアップグレードする場合です。SAP システムをアップグレードした後、アダプター環境をアップグレードする必要があります。SAP の新しいバージョンでのアダプター環境のアップグレードの詳細については、『SAP R/3 の新しいバージョンでのアップグレード』 を参照してください。
- SAP の旧バージョンをサポートするオブジェクト用にアダプター提供の ABAP Handler を実装する場合

  例えば、SAP バージョン 4.6 アプリケーションをサポートするコネクターを使用している環境で、SAP バージョン 4.0 または 4.5 をサポートする Material オブジェクトを使用する必要が生じた場合です。この Material オブジェクトを使用するには、Handler を SAP バージョン 4.6 システム用にアップグレードする必要があります。オブジェクトをより新しい SAP のバージョン用にアップグレードする方法の詳細については、 294 ページの『ABAP Handler のアップグレード』を参照してください。

SAP R/3 の新しいバージョンでのアップグレード

SAP R/3 アプリケーションのアップグレード・プロセスは、アダプターの ABAP 開発を変更することはありませんが、SAP R/3 アプリケーションを変更する場合があり、その結果、一部のアダプターの ABAP 開発が正しく動作しなくなることがあります。

このセクションでは、アップグレード済みの SAP R/3 アプリケーションでアダプターの ABAP 開発をアップグレードする方法について説明します。アダプターをアップグレードする前に、SAP R/3 アプリケーションのアップグレードを済ませておく必要があります。

アダプターの ABAP 開発をアップグレードするには、以下の手順を行います。

1. SAP R/3 アプリケーションの正しいバージョンに対応した、最新の ABAP Extension Module トランスポート・ファイルをインストールします。

   正しいバージョン固有トランスポート・ファイルをインストールする必要があります。これらのトランスポート・ファイルのインストールの詳細については、223 ページの『コネクター・トランスポート・ファイルのインストール』を参照してください。
2. すべてのプログラムをコンパイルし、ABAP 開発に関連する構文エラーを解決します。

   構文エラーを見つけ出す最も簡単な方法は、各オブジェクトに関連付けられた各機能グループを生成し、一度に 1 つずつエラーを修正していくことです。すべての機能グループが正常にコンパイルされるまで、このプロセスを繰り返します。トリガー・プログラムなどのような、機能グループに関連付けられていないその他のプログラムも必ず生成してください。アダプターをアップグレードするために必要な新規のトランスポートが適用された後、ABAP トリガー・プログラムに小さな更新が必要になる場合があります。

   SAP R/3 バージョン 4.x にアップグレードする場合は、4.x の ABAP Handler が製品のネーム・スペース `/CWLD/` を使用することに注意してください。SAP R/3 バージョン 4.x をサポートするコネクターにアップグレードする場合の特別な考慮事項については、297 ページの『Connector for SAP R/3』を参照してください。
3. 新しい環境をテストし、必要に応じて変更を加えます。

   アップグレードに関するすべての問題を解決するには、完全なシステム・テストを行う必要があります。適切なトランザクションやプログラムを実行し、ビジネス・オブジェクトを SAP システムに送信して、イベント検出機構をテストします。その他の問題を識別するには、SAP システム内のコネクターのログが利用できます。

ABAP Handler のアップグレード

ABAP Handler のアップグレードは、2 段階のステップで行います。

1. ABAP Handler を ABAP Extension Module のバージョンが異なる環境に導入することで発生した、すべてのコンパイル・エラーを解決します。
2. 新しい SAP R/3 バージョンでビジネス・オブジェクトが提供する機能を評価します。例えば、ビジネス・オブジェクトが正しく動作するにもかかわらず正しい情報を戻さない場合や、SAP で Call Transaction の画面が変更されたため、ビジネス・オブジェクトが機能しない場合があります。

このセクションでは、最初のステップでの処理の詳細について説明します。例えば、ビジネス・オブジェクトの ABAP Handler のパッケージ化や、コンパイルで発

生する可能性のある競合ポイントについてのガイドラインを示します。第 2 のステップについては、このセクションでは取り上げません。オブジェクトの機能の拡張方法の詳細については、247 ページの『第 21 章 ABAP Extension Module のビジネス・オブジェクトの開発』を参照してください。

重要: ビジネス・オブジェクトをアップグレードすると、そのオブジェクトが元は IBM で開発されたものであっても、カスタム作業と見なされます。

ABAP Handler のアップグレードは、次のような場合に行います。

- 以前にインプリメントした IBM 提供の SAP R/3 ビジネス・オブジェクトを、それ以降の SAP R/3 のバージョンで使用する場合。例えば、SAP R/3 バージョン 3.x システムで既にインプリメントした Customer ビジネス・オブジェクトが 4.6 システムには存在しない場合です。
- 必要なバージョン以外の SAP R/3 バージョンをサポートするアダプター提供の SAP R/3 ビジネス・オブジェクトを使用する場合。例えば、アダプターが提供した SAP R/3 バージョン 3.x 用の Material ビジネス・オブジェクトを、SAP R/3 バージョン 4.6 システムで使用する場合です。

アップグレード手順は、基本的には同じです。唯一の違いは、以前に実装したビジネス・オブジェクトをアップグレードする場合には、次の手順として、そのビジネス・オブジェクトをトランスポート・ファイルにパッケージする必要がある点です。

注: SAP R/3 バージョン 4.6 に IBM 製品のネーム・スペースを使用しないビジネス・オブジェクトが存在する場合は、これらのビジネス・オブジェクトをネーム・スペース対応にアップグレードする必要があります。

アダプター提供の ABAP Handler を、SAP R/3 のあるバージョンから別のバージョンにアップグレードするには、以下の手順を行います。

1. SAP R/3 の使用するバージョンに対応した、ABAP Extension Module トランスポート・ファイルの最新バージョンがインストールされていることを確認します。
2. 既存のビジネス・オブジェクトをトランスポート・ファイルにパッケージします。実装のために変更されていないビジネス・オブジェクトをアップグレードする場合は、ロードされた元のトランスポートを使用できるため、ステップ 3 に進んでください。

   各ビジネス・オブジェクトについて何を組み込む必要があるかについては、アダプター提供のトランスポート・ファイルをテンプレートとして使用してください。組み込む必要があるものとしては、機能グループ、IDoc 定義、Dynamic Retrieve および Dynamic Transaction データなどがあります。

   - 追加のプログラムおよびカスタム作業を組み込みます。

     コネクターの ABAP コンポーネントで行ったカスタム作業は、コネクターの新しい SAP R/3 ABAP コンポーネントに、手動で適用する必要があります。例えば、IDoc Handler や Dynamic Transaction などのアダプター提供の ABAP Handler に対して、すべての変更を手動で適用する必要があります。

- プログラムの /CWLD/TRIGGERING_RESTRICTIONS に対して変更が行われているかどうかを確認します。このプログラムは、カスタマーが変更を行うように作成されています。

  変更が行われている場合は、カスタム作業をリファレンスとして使用するために、トランスポート・ファイルとしてではなく、テキスト・ファイルとしてダウンロードすることで、競合を回避できます。
- トランスポートをリリースし、トランスポート番号をメモします。この情報は、BASIS 管理者がオブジェクトを新しい SAP R/3 システムにロードするために必要です。

3. SAP R/3 バージョン 3.x システムの IDoc に限り、IDoc の構造体およびセグメント定義を取り込み、それらを新しいシステムで手動で再作成します。

   SAP R/3 バージョン 3.x 環境や IDoc を使用していない場合は、このステップをスキップしてください。
4. ビジネス・オブジェクト・トランスポート・ファイルをインストールします。ステップ 1 でパッケージしたビジネス・オブジェクトのトランスポートのインストールは、ローカルの BASIS 管理者に依頼する必要があります。

   BASIS 管理者は、トランスポートのために利用できるすべてのオーバーライド・コードを使用する必要があります。これにより、ビジネス・オブジェクトはコンパイル・エラーが発生した場合でも環境内にインポートされます。ビジネス・オブジェクトをインポートする前に、BASIS 管理者はインポート処理の間にアップグレード担当者が不整合に遭遇する可能性があることに注意する必要があります。
   - ステップ 2 で既存のビジネス・オブジェクトをパッケージした場合は、これらのトランスポート・ファイルをインストールします。
   - 実装されていないビジネス・オブジェクトを使用している場合は、使用するビジネス・オブジェクトの最新のトランスポート・ファイルをインストールします。正しいバージョン固有トランスポート・ファイルをインストールする必要があります。
5. すべてのプログラムをコンパイルし、ABAP 開発に関連する構文エラーを解決します。

   構文エラーを見つけ出す最も簡単な方法は、各ビジネス・オブジェクトに関連付けられた各機能グループを生成し、一度に 1 つずつエラーを修正していくことです。すべての機能グループが正常にコンパイルされるまで、このプロセスを繰り返します。トリガー・プログラムなどのような、機能グループに関連付けられていないその他のプログラムも必ず生成してください。アダプターをアップグレードするために必要な新規のトランスポートが適用された後、ABAP トリガー・プログラムに小さな更新が必要になる場合があります。

   SAP R/3 バージョン 4.x にアップグレードする場合は、4.x の ABAP Handler が IBM 製品のネーム・スペース /CWLD/ を使用することに注意してください。SAP R/3 バージョン 4.x をサポートするコネクターにアップグレードする場合の特別な考慮事項については、297 ページの『Connector for SAP R/3』を参照してください。
6. イベント検出機構を適用します。

ユーザー出口の正確な位置が以前とは異なっている場合があります。コードのキー SAP 行を検索することで、最も近い値を推定することができます。

7. 新しい環境をテストし、必要に応じて変更を加えます。

   アップグレードに関するすべての問題を解決するには、完全なシステム・テストを行う必要があります。適切なトランザクションやプログラムを実行し、ビジネス・オブジェクトを SAP システムに送信して、イベント検出機構をテストします。その他の問題を識別するには、SAP システム内のコネクターのログが利用できます。

アップグレード考慮事項

以下のセクションでは、アップグレードのシナリオに関する参照情報を提供します。この参照情報は、Connector for SAP R/3 バージョン 4.6 および IDocs のアップグレード・プロセスに役立つように提供されます。

Connector for SAP R/3

Connector for SAP R/3 バージョン 4.x では、IBM 製品のネーム・スペース /CWLD/ を使用するため、以下に示すガイドラインは、このリネームされた環境で ABAP Handler を動作させるための作業を容易にします。ビジネス・オブジェクトが処理される方法の詳細、およびオブジェクト開発の背景情報については、233 ページの『第 20 章 ABAP Extension Module でのビジネス・オブジェクトの処理』を参照してください。

Dynamic Retrieve または Dynamic Transaction を使用するビジネス・オブジェクト

トランザクション・ベースの (Dynamic Retrieve および Dynamic Transaction) タイプのビジネス・オブジェクトの変換機能は、IBM WebSphere BI Station. を通じて提供されます。古いシステムの IBM WebSphere BI Station からビジネス・オブジェクトをテキスト・ファイルにダウンロードした後、新しいシステムの IBM WebSphere BI Station を使用してこのテキスト・ファイルを新しいテーブルにアップロードできます。これを行うには、「ツール」タブの「Object MetaData」オプションを使用します。

以下の点に注意する必要があります。

- トランザクション・ベースの Dynamic Retrieve の Long Text Declarations は、新しいテーブルに手動で入力する必要があります。
- トランザクション・ベースの Dynamic Retrieve の Table Declarations は、古い組み込みプログラムから新しいテーブル宣言組み込みプログラムに手動で移植する必要があります。

IDoc または BAPI Handler、およびカスタム作業を使用するビジネス・オブジェクト

Y* で始まる SAP R/3 バージョン 4.x ビジネス・オブジェクトは、IBM 製品の /CWLD/ ネーム・スペースにリダイレクトする必要があります。名前のみが変更されています。SAP の「where used list」機能は、変更する必要のあるすべての参照を

検索するために、非常に役立ちます。変更する必要のある最も一般的な参照のリストを以下に示します。検索を完全に実行できたかどうかについてテストします。

表 50 に、/CWLD/ ネーム・スペースの命名規則の変更を示します。パラメーター・リストを変更する必要はありません。

表 50. ネーム・スペース・オブジェクト名の変更

旧名	新規名
機能モジュールのインターフェース・パラメーター	
YXR_EVENT-OBJ_KEY	/CWLD/LOG_HEADER-OBJ_KEY (3 箇所)
YXR_LOG_H-LOG_NR	/CWLD/LOG_HEADER-LOG_NR
YXR_RFCRC-YXR_RFCRC	/CWLD/RFCRC_STRU-RFCRC
通常はビジネス・オブジェクト機能グループの TOP インクルード内にある変更点	
YXR_CNST	/CWLD/CONSTANTS
YXRIFRM0	/CWLD/INBIDOC_FRMS0
データ・エレメント	
YXR_VERB	/CWLD/OBJ_VERB
テーブル構造体	
YXR_CONFIG	/CWLD/CONF_VAL
YXR_EVENTS	/CWLD/EVT_CUR
YXR_LOG_I	/CWLD/LOG_ITEM
YXR_RFC_S	/CWLD/OBJ_STRU
LOG_UPDATE 実行ステートメントで参照されるプログラム	
SAPLYXR1	/CWLD/SAPLLOG
トリガー機能モジュール (パラメーター・リストを変更する必要はありません)	
Y_XR_COMMIT_IDOC_RAISE_DELETE	/CWLD/ COMMIT_IDOC_RAISE_DELETE
Y_XR_/ADD_TO_QUEUE	/CWLD/ADD_TO_QUEUE

追加の IBM WebSphere ABAP コンポーネント

カスタム・オブジェクトおよびカスタム作業をアップグレードするほかに、以下の作業を行う必要があります。

- すべての ABAP コードを、古いイベント制限プログラムから新しいイベント制限プログラムにアップグレードします。
- すべての構成オブジェクトおよび構成値を、古いテーブルから新しいテーブルに手動でアップグレードします。
- すべてのイベント分配エントリーを、古いテーブルから新しいテーブルにアップグレードします。
- ログ・オブジェクト・リンクを、古いテーブルから新しいテーブルにアップグレードします。

既存の SAP R/3 バージョン 4.x イベント表に既にイベントを持っている実動場所については、特別な考慮が必要です。これらのイベントを既存のイベント表から新しいイベント表に転送する場合は、必ず IBM ソフトウェア・サポートと連携して行ってください。

IDoc のパッケージ化と再作成

このセクションの説明は、IBM WebSphere SAP R/3 バージョン 3.x ビジネス・オブジェクトにのみ適用されます。

IDoc オブジェクトは、SAP R/3 バージョン 3.x から移送することができないため、新しい SAP R/3 システムで手動で再作成する必要があります。それには、以下の作業を行う必要があります。

- IDoc の構造体およびセグメント定義の取り込み
- 手動による IDoc の再作成

IDoc の構造体およびセグメント定義の取り込み

IDoc の最も有用な表現を取り込むには、すべてのセグメントを識別する構造体の全体を取り込んだ後、各セグメントのビジネス・オブジェクト定義を取り込みます。IDoc の明確な表現が得られたら、その表現を使用し、新しいシステムで IDoc を手動で再作成します。

新旧両方のシステムにアクセスできる場合は、古いシステムでビジネス・オブジェクトをコピーし、新しいシステムに貼り付けます。しかし、両方のシステムを同時には利用できない場合は、SAP システムの外部で IDoc およびセグメント定義を参照用として記録する必要があります。これはオプションですが、定義を記録することを強くお勧めします。

IDoc およびセグメント定義の最も有用な表現をダウンロードするには、最初に IDoc の構造体全体をダウンロードし、次に IDoc セグメント定義をダウンロードします。

IDoc 構造体全体のダウンロード: IDoc 構造体全体をダウンロードするには、以下の手順を行います。

1. 「Develop IDocs Type」画面に移動します (トランザクション WE30)。
2. IDoc オブジェクト名を入力し、次に「Display」をクリックします (F7)。
3. すべてのセグメントが表示されるように、IDoc 構造体を展開します。
   a. 構造体のテキスト・バージョンをダウンロードします。
   b. 「System」メニューで、「List」をクリックし、「Save」をクリックしてから、「Local File」をクリックします。
   c. デフォルト・オプションをそのまま受け入れ、「Enter」をクリックします。

      ファイルはテキスト・ファイルとしてダウンロードされるので、任意のテキスト・エディターで表示できます。
   d. ファイルをダウンロードする場所を指定し、「Transfer」をクリックします。

セグメント定義のダウンロード: 一度に 1 つのセグメント定義をダウンロードできます。各セグメントについて、以下のステップを繰り返します。セグメント定義をダウンロードするには、以下の手順を行います。

1. トランザクション SE11 に移動し、セグメント名を入力します。
2. 「Dictionary Object」メニューで、「Print」をクリックします。

   「Table Structure」ボックスに、必ずチェックマークを付けてください。
3. 「Print immediately」チェック・ボックスを選択解除し、「new spool request」チェック・ボックスにチェックマークを付け、「Continue」をクリックします。
4. 「Spool Request Selection」画面に移動し (トランザクション SP01)、印刷要求を表示します。

5. 「Execute」をクリックし、要求の横にあるチェック・ボックスを選択し、次に「Display comments」をクリックします。
6. データをダウンロード可能な形式に変換します。
   a. 「Goto」メニューで、「List Display」をクリックします。
   b. セグメントのテキスト・バージョンをダウンロードします。「System」メニューで、「List」をポイントし、「Save」をポイントしてから、「Local File」をクリックします。
   c. デフォルト・オプションをそのまま受け入れ、「Enter」をクリックします。

      ファイルはテキスト・ファイルとしてダウンロードされるので、任意のテキスト・エディターで表示できます。
   d. ファイルをダウンロードする場所を指定し、「Transfer」をクリックします。

テキスト・ファイルを使用してオブジェクトを表現したら、オブジェクト階層をセットアップするために、それらのテキスト・ファイルをスプレッドシート・アプリケーションにインポートします。このようにすると、フィールドを切り取り、SAP アプリケーションのセグメント・エディターに直接貼り付けることができるので、IDoc セグメントの作成が容易になります。

手動による IDoc の再作成

IDoc の表現が得られたら、それを新しいシステムで手動で再作成する必要があります。SAP R/3 バージョン 4.x 環境では、IDoc タイプおよびセグメント定義を格納するために、SAP R/3 バージョン 3.x とは異なるテーブルを使用します。そのため、IDoc 定義を再定義して適切なテーブルを更新するには、SAP のツールを使用する必要があります。このプロセスには、次の 2 つのステップがあります。

- 「Develop Segments」画面 (トランザクション WE31) を使用して、セグメント定義を再作成します。
- IDoc タイプを再作成し、それにすべてのセグメントを割り当てます。

SAP R/3 バージョン 3.x データ・エレメントを新しいセグメント・フィールドに割り当てることでセグメントを再作成している際に表示される共通エラー・メッセージに、`Invalid data element` があります。SAP では、SAP R/3 バージョン 3.x データ・エレメントの多くを、SAP R/3 バージョン 3.x 名の末尾にアンダースコアーと文字 D (_D) の付いたデータ・エレメントに置き換えました。例えば、SAP R/3 バージョン 3.x の CHARG はデータ・エレメントの Batch Number で、SAP R/3 バージョン 4.x では CHARG_D に置き換えられています。

新しい形式のデータ・エレメントが存在しない場合は、SAP R/3 バージョン 4.x システムで新しい形式を探してください。このデータ・エレメントは、SAP R/3 バージョン 3.x システムでのオリジナルと、タイプおよび長さが同じである必要があります。記述は、処理には影響を与えず、また、ログでのみ表示することができます。

重要: IDoc 定義と IBM WebSphere ビジネス・オブジェクト・リポジトリーには直接的な関係があるため、IDoc、セグメント、あるいはセグメント・フィールドの名前は変更しないでください。また、ABAP の機能でも、これらの名前に依拠して IDoc を処理していました。

付録 A. クイック・ステップ

この付録は、「*Adapter for mySAP.com ユーザーズ・ガイド* 」に記載されている情報を補足するものです。ユーザーズ・ガイドの情報を差し替える意図はありません。

注: ここに示すクイック・ステップは、WebSphere メッセージ・ブローカーまたは InterChange Server Express のいずれかを統合ブローカーとして、スタンドアロンで稼働するアダプターを対象としています。

このステップを始める前に、以下の作業を行う必要があります。

- WebSphere メッセージ・ブローカーまたは InterChange Server Express をブローカーとしてインストールします。
- SAP JCo API をインストールします。SAP JCo は、SAP の Web サイト、`http://service.sap.com/connectors` からダウンロードできます。SAP JCo にアクセスするには、SAPNet アカウントが必要です (アカウントがない場合は、ローカルの SAP BASIS 管理者にご連絡ください)。ダウンロードしたファイルは *ProductDir*¥ODA¥SAP ディレクトリーおよび *ProductDir*¥connectors¥SAP ディレクトリー (*ProductDir* はコネクターがインストールされているディレクトリー) に追加してください。
- JDK をインストールします。
- WebSphere Business Integration Server Express Adapter のランタイム環境 (アダプター・フレームワーク) をインストールします。
- 標準の MQ キューを構成します。

このクイック・ステップを実行しながら、独自のビジネス・オブジェクトを作成することも、提供されているサンプルのビジネス・オブジェクトを使用することもできます。サンプルは、*ProductDir*¥connectors¥SAP¥samples ディレクトリー (*ProductDir* はコネクターがインストールされているディレクトリー) にあります。

一般的な構成プロパティー

次の表に、WebSphere メッセージ・ブローカー用に維持する必要のある構成プロパティーを示します。`CN_SAP.txt` を使用して、SAP 構成ファイルを作成してください。このファイルは *ProductDir*¥repository¥SAP に格納されています。このファイルは、Connector Configurator Express を使用して開きます。

表 51. 標準構成プロパティー

プロパティー名	デフォルト値	必要な値
ApplicationName	なし	SAPConnector
BrokerType	InterChange Server Express	ICS
AdminInQueue	/ADMININQUEUE	ADMININQUEUE
AdminOutQueue	/ADMINOUTQUEUE	ADMINOUTQUEUE
DeliveryQueue	/DELIVERYQUEUE	DELIVERYQUEUE

表 51. 標準構成プロパティー (続き)

プロパティー名	デフォルト値	必要な値
FaultQueue	/FAULTQUEUE	FAULTQUEUE
RequestQueue	/REQUESTQUEUE	REQUESTQUEUE
ResponseQueue	/RESPONSEQUEUE	RESPONSEQUEUE
SynchronousRequestQueue	/SYNCHRONOUSREQUESTQUEUE	SYNCHRONOUSREQUESTQUEUE
SynchronousResponseQueue	/SYNCHRONOUSRESPONSEQUEUE	SYNCHRONOUSRESPONSEQUEUE
MessageFileName	SAPConnector.txt	SAPCONNECTOR.TXT
RepositoryDirectory	*ProductDir*¥repository	<ビジネス・オブジェクト指定の位置>
Jms.MessageBrokerName	crossworlds.queue.manager	<キュー・マネージャー名>
AgentTraceLevel	0	5

表 52. コネクター固有のプロパティー

プロパティー名	デフォルト値	必要な値
ApplicationPassword	SOFTWARE	<*SAP* アプリケーションのパスワード>
ApplicationUserName	CROSSWORLDS	<*SAP* アプリケーションのユーザー名>
Client	なし	<クライアント番号>
Hostname	なし	<*SAP* アプリケーション・サーバー名>

BAPI Module のクイック・ステップ

BAPI Module を構成する前に、次のコネクター固有のプロパティーを構成する必要があります。

プロパティー名	デフォルト値	必要な値
Modules	なし	BAPI

BAPI Module でのビジネス・オブジェクトの生成

BAPI Module 用のビジネス・オブジェクトを生成するには、以下の手順を実行します。

1. SAPODA を始動します。
2. Business Object Designer Express を始動します。
3. Business Object Designer Express で、「ファイル」>「新規」を選択します。ウィザードが開始します。

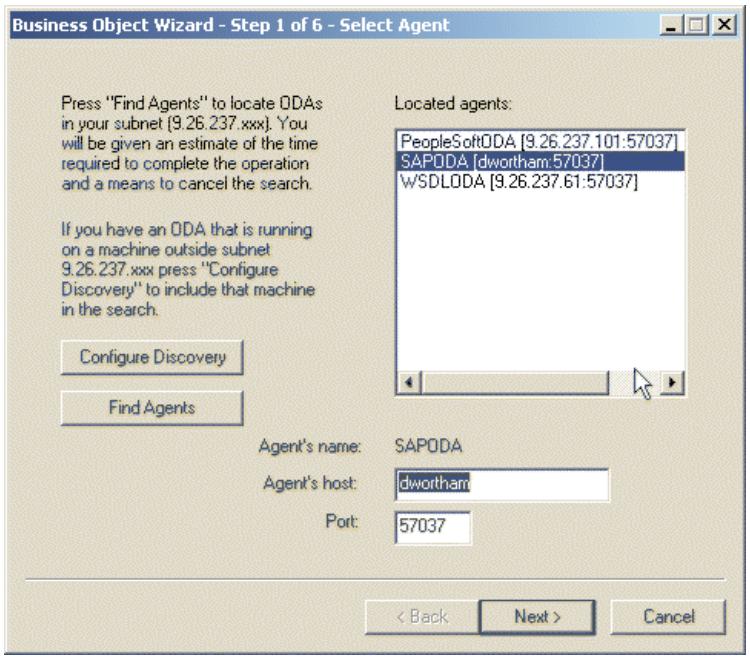


図 76. 「ビジネス・オブジェクト・ウィザード」—「エージェントの選択」

4. 「検索の構成」を選択します。
   a. 検索を実行するマシンのホスト・アドレスを入力します。
   b. 「ホストの追加 (Add Host)」を選択します。
   c. 「OK」を選択します。
5. 「エージェントの検索」を選択します。
   a. エージェントを強調表示します。「次へ」を選択します。
   b. UserName、Password、Client、SystemNumber、ASHostName、および FileDestination の値を入力します。プロファイルを保管します。
6. ウィザードのステップ 3 で、RFC ノードを展開します。
   a. 「名前で検索 (Search By Name)」を右クリックします。
   b. `bapi_customer_getdetail` と入力します。
   c. `bapi_customer_getdetail` を強調表示します。
   d. 「次へ」を選択します。

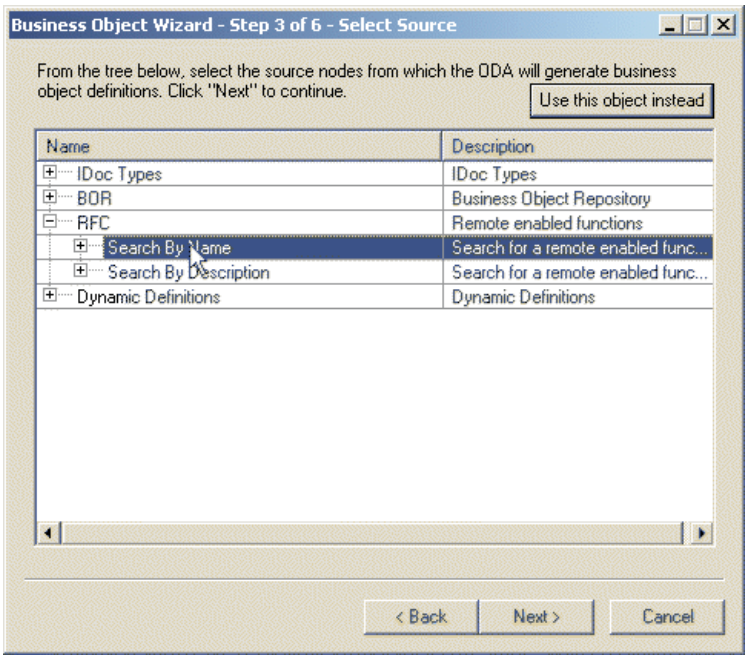


図 77. 「ビジネス・オブジェクト・ウィザード」—「ソースの選択」

7. 「次へ」を選択します。
8. 「動詞」を「**Retrieve**」 に設定し、「サーバー・サポート (Server Support)」を「**なし**」に設定します。「OK」を選択します。
9. 「エージェントの SAPODA の通知 (Agent SAPODA Notification)」で、「いいえ」を選択します。
10. 別のウィンドウでビジネス・オブジェクトを開きます。生成されたビジネス・オブジェクト指定を、リポジトリー・ディレクトリーの標準のプロパティー値で指定した位置に保管します。

BAPI Module の構成

ビジネス・オブジェクトを生成したら、構成ファイルの「サポートされているビジネス・オブジェクト」セクションに親オブジェクトの名前を追加することにより、BAPI Module の構成を続行します。

BAPI Module のテストのための準備

BAPI Module をテスト用にセットアップするには、ポート・コネクターを使用します。

1. SAP 構成ファイルをコピーします。コピーしたファイルの名前を `portconnector.cfg` に変更します。
2. Connector Configurator Express で portconnector.cfg を開きます。
3. 「標準」タブで次のプロパティーを変更します。
   - `ApplicationName` を `PortConnector` に
   - `DELIVERYQUEUE` を `REQUESTQUEUE` に
   - `REQUESTQUEUE` を `RESPONSEQUEUE` に
4. 変更内容を保管します。`portconnector.cfg` を閉じます。

5. sapconnector.cfg を開きます。
6. 変更内容を保管します。mySAP.com を始動します。

BAPI Module のテスト

BAPI Module をテストするには、以下の手順を実行します。

1. Test Connector を開きます。

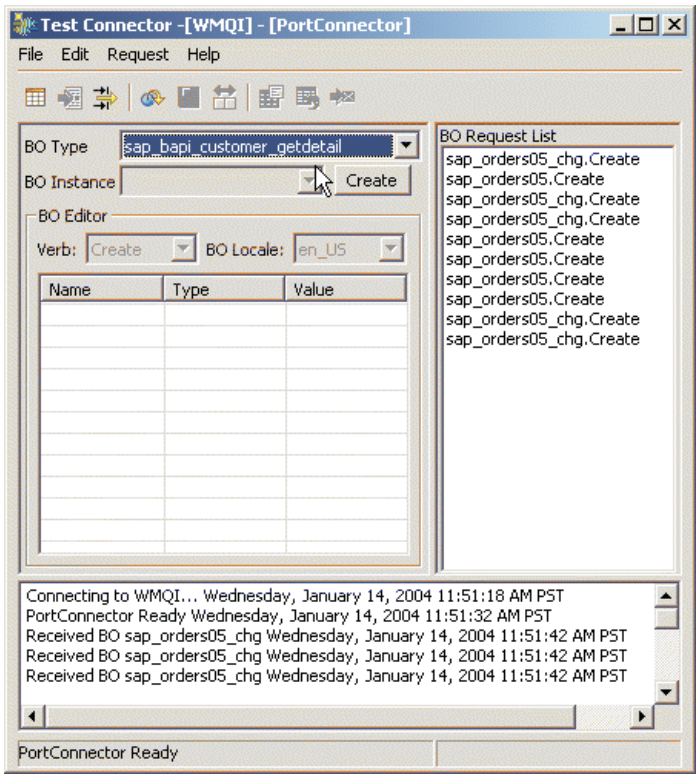


図 78. 「Test Connector」

2. 「ファイル」>「プロファイルの作成/選択」を選択します。
3. 「ファイル」>「新規プロファイル」を選択します。

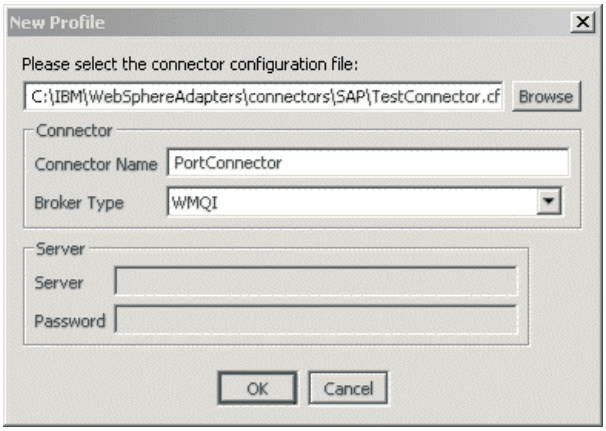


図 79. 「Test Connector」—「新規プロファイル」

4. 「参照」を選択します。
   a. portconnector.cfg を探します。「開く」を選択します。
   b. 「コネクター名」に、PortConnector と入力します。
   c. 「ブローカー・タイプ」に、ICS と入力します。
   d. 「OK」を選択します。
5. PortConnector を強調表示します。「OK」を選択します。
6. 「ファイル」>「接続」を選択します。
7. ビジネス・オブジェクトのインスタンスを、以下の手順で作成します。
   a. 「ビジネス・オブジェクト・タイプ」で、SAP_BAPI_customer_getdetail を選択します。
   b. 「作成」を選択します。
   c. 新規オブジェクトを入力します。「OK」を選択します。
8. 「動詞」を Retrieve に変更します。Customer_to_be_required に既存のカスタマーを取り込みます。
9. 「要求」>「送信」を選択します。
10. ログ・ファイルに成功のメッセージがあるかどうか確認します。

RFC Server Module のクイック・ステップ

RFC Module を構成する前に、次のコネクター固有のプロパティーを構成する必要があります。

プロパティー名	デフォルト値	必要な値
Modules	なし	Rfcserver
RfcProgramId	CWLDSERVER	*<SAP トランザクション sm59 に登録されているプログラム ID >*

RFC Server Module でのビジネス・オブジェクトの生成

RFC Module 用のビジネス・オブジェクトを生成するには、以下の手順を実行します。

1. SAPODA を始動します。
2. Business Object Designer Express を始動します。
3. Business Object Designer Express で、「ファイル」>「新規」を選択します。ウィザードが開始します。
4. 「検索の構成」を選択します。
   a. 検索を実行するマシンのホスト・アドレスを入力します。
   b. 「ホストの追加 (Add Host)」を選択します。
   c. 「OK」を選択します。
5. ウィザードのステップ 3 で、RFC ノードを展開します。
   a. 「名前で検索 (Search By Name)」を右クリックします。
   b. bapi_customer_getdetail と入力します。

c. bapi_customer_getdetail を強調表示します。

d. 「次へ」を選択します。

6. 「次へ」を選択します。
7. 「動詞」を「Retrieve」に設定し、「サーバー・サポート (Server Support)」を「なし」に設定します。「OK」を選択します。
8. 「エージェントの SAPODA の通知 (Agent SAPODA Notification)」で、「いいえ」を選択します。
9. 別のウィンドウでビジネス・オブジェクトを開きます。「一般」> 「コラボレーションの設定 = ″RFCCollab″ (Set Collab = ″RFCCollab″)」を選択します。
10. 生成されたビジネス・オブジェクト指定を、リポジトリー・ディレクトリーの標準のプロパティー値で指定した位置に保管します。

RFC Server Module の構成

ビジネス・オブジェクトを生成したら、以下の手順で RFC Server Module の構成を続行します。

1. 構成ファイルの「サポートされているビジネス・オブジェクト」セクションに親オブジェクトの名前を追加します。
2. 生成された BOHandler .class ファイルを、ODA 構成プロパティーに指定された定義から %CROSSWORLD%¥connectors¥SAP¥rfc¥client にコピーします。

SAP サーバーのプロファイルの作成

SAP サーバー用のプロファイルを作成するには、以下の手順を実行します。

1. 「SAP ログオン (SAP Logon)」を開きます。
2. 「新規 (New)」を選択します。
3. 以下のフィールドに入力し、「OK」を選択します。

説明	サーバーのホスト名
アプリケーション・サーバー	サーバーのホスト名
システム番号	00
説明	ホスト名は標準です。必要な説明を入力します。

4. 作成したプロファイルをダブルクリックして開きます。
5. ユーザー名とパスワードを入力します。「トランザクション (Transaction)」> 「タイプ /nse37 (Type /nse37)」を選択します。Function Builder が開きます。
6. 機能モジュールの場合、bapi_customer_getdetail と入力します。「機能モジュール (Funtion Module)」>「テスト (Test)」>「単一のテスト (Single Test)」を選択します。
7. RFC ターゲット・システムの場合は、コネクター固有のプロパティーに設定した Rfcprogramid の値を使用します。また、以下のフィールドに入力します。

フィールド	例
Customer Number	0000000001
PI_SALESORG	0001

PI_DISTR_CHAN	01
PI_DIVISION	01

RFC Server Module のテスト

BAPI Module をテスト用にセットアップするには、ポート・コネクターを使用します。

1. SAP 構成ファイルをコピーします。コピーしたファイルの名前を portconnector.cfg に変更します。
2. Connector Configurator Express で portconnector.cfg を開きます。
3. 「標準」タブで次のプロパティーを変更します。
   - ApplicationName を PortConnector に
   - REQUESTQUEUE を SYNCHRONOUSREQUESTQUEUE に

   変更内容を保管して、ウィンドウを閉じます。
4. sapconnector.cfg を開きます。
5. REQUESTQUEUE を SYNCHRONOUSREQUESTQUEUE に変更します。変更内容を保管します。
6. コネクターを始動します。「機能モジュール (Function Module)」>「実行」を選択します。
7. Test Connector で、「BO 要求リスト」内のオブジェクトを探します。このオブジェクトを強調表示して、「要求」>「応答」>「成功」を選択します。
8. ログに成功のメッセージがあるかどうか調べます。

ALE Module のクイック・ステップ

ALE Module を構成する前に、以下の永続的 WebSphere MQ キューを作成します。

- SAPtid_Queue
- SAPtid_QueueManager
- SAPALE_Event_Queue
- SAPALE_Wip_Queue
- SAPALE_Archive_Queue
- SAPALE_UnSubscribed_Queue
- SAPALE_Error_Queue

MQ キューの作成方法については、WebSphere MQ の資料を参照してください。

次に、以下のコネクター固有のプロパティーを構成します。

プロパティー名	デフォルト値	必要な値
Modules	なし	Ale
AleEventDir	なし	*ProductDir*¥connectors¥SAP¥ale
SAPtid_QueueManager	なし	<キュー・マネージャー名>
SAPtid_Queue	なし	<キュー名>

プロパティー名	デフォルト値	必要な値
SAPALE_Event_Queue	なし	<イベント・キュー名>
SAPALE_Wip_Queue	なし	<*WIP* キュー名>
SAPALE_Archive_Queue	なし	<アーカイブ・キュー名>
SAPALE_UnSubscribed_Queue	なし	<アンサブスクライブ・キュー名>
SAPALE_Error_Queue	なし	<エラー・キュー名>
RfcProgramId	なし	<*SAP* トランザクション *sm59* に定義されたプログラム *ID* 名>
NumberOfListeners	1	1 (単一スレッドの場合)

リモートの WebSphere キューの場合は、以下のプロパティーも構成してください。

プロパティー名	デフォルト値	必要な値
SAPtid_QueueManagerLogin	なし	<キュー・マネージャーのログイン>
SAPtid_QueueManagerPassword	なし	<キュー・マネージャーのパスワード>
SAPtid_QueueManagerHost	なし	<キュー・マネージャーのホスト>
SAPtid_MQPort	なし	<*MQ* ポート>
SAPtid_MQChannel	なし	<*MQ* チャネル>

ALE Module でのビジネス・オブジェクトの生成

ALE Module でビジネス・オブジェクトを生成するには、以下の手順を実行します。

1. SAPODA を始動します。
2. Business Object Designer Express を始動します。
3. Business Object Designer Express で、「ファイル」>「新規」を選択します。ウィザードが開始します。
4. 「検索の構成」を選択します。
   a. 検索を実行するマシンのホスト・アドレスを入力します。
   b. 「ホストの追加 (Add Host)」を選択します。
   c. 「OK」を選択します。
5. ウィザードのステップ 3 で、「IDoc タイプ (IDoc Types)」を展開します。
   a. 「システムから生成 (Generate From System)」を展開します。
   b. 「基本 IDoc タイプ (Basic IDoc Types)」を展開します。
   c. 「名前で選択... (Select by Name...)」を右クリックします。
   d. 「項目を検索...」を選択します。
   e. orders03 と入力します。「OK」を選択します。
6. ORDERS03 を強調表示します。「次へ」を選択します。
7. 「次へ」を選択します。
8. 「OK」を選択します。ビジネス・オブジェクトを生成します。

9. 「ビジネス・オブジェクト定義のコピーを個別のファイルに保管する」を選択してから、「新規ビジネス・オブジェクト定義を個別のウィンドウで開く (Open new business object definition to a separate window)」を選択します。「完了」を選択します。

ビジネス・オブジェクトの編集

ビジネス・オブジェクトを編集するには、以下の手順を実行します。

1. 「一般」タブを選択します。
2. アプリケーション固有の情報作成のメッセージ・タイプを `MsgType = ORDERS` に変更します。
3. *ProductDir*¥repository¥SAP¥BO_SAPIDocControl.txt を開いて、これをリポジトリー・ディレクトリーに保管します。
4. 構成ファイルの「サポートされているビジネス・オブジェクト」セクションに親オブジェクトの名前を追加します。
5. SAP トランザクション SM59 を使用して、RFC Server Module をSAP Gateway に登録します。
6. 以下のことを確認します。
   - 論理システムが定義され、SAP システムおよび外部システム用に割り当てられているかどうか (SALE)。
   - 分散モデルが維持されていること、および必要なメッセージ・タイプがモデルに追加されているかどうか (トランザクション・コード BD64)。
   - 論理システムまたは分散モデルのパートナー・プロファイルが存在するかどうか (トランザクション・コード WE20)。
   - 論理システムまたは分散モデル用にポートが定義されているかどうか (トランザクション・コード WE21)。

ALE Module のテストのための準備

ALE Module をテスト用にセットアップするには、ポート・コネクターを使用します。

1. SAP 構成ファイルをコピーします。コピーしたファイルの名前を `portconnector.cfg` に変更します。
2. Connector Configurator Express で portconnector.cfg を開きます。
3. 「標準」タブで次のプロパティーを変更します。
   - `ApplicationName` を `PortConnector` に
   - `DELIVERYQUEUE` を `REQUESTQUEUE` に
   - `REQUESTQUEUE` を `RESPONSEQUEUE` に
4. 変更内容を保管します。`portconnector.cfg` を閉じます。
5. `sapconnector.cfg` を開きます。
6. 変更内容を保管します。mySAP.com を始動します。

ALE Module の要求処理のテスト

ALE Module をテストするには、以下の手順を実行します。

1. Test Connector を開きます。

2. 「ファイル」>「プロファイルの作成/選択」を選択します。
3. 「ファイル」>「新規プロファイル」を選択します。
4. 「参照」を選択します。
   a. 「開く」を選択します。
   b. 「コネクター名」に、PortConnector と入力します。
   c. 「ブローカー・タイプ」に、ICS と入力します。
   d. 「OK」を選択します。
5. PortConnector を強調表示します。「OK」を選択します。
6. 「ファイル」>「接続」を選択します。
7. ビジネス・オブジェクトのインスタンスを、以下の手順で作成します。
   a. 「ビジネス・オブジェクト・タイプ」で、sap_order03 を選択します。
   b. 「作成」を選択します。
   c. 「名前の入力」に、新規オブジェクトを入力します。「OK」を選択します。
8. 動詞を Create に変更します。
9. 「制御レコード (Control Record)」を右クリックします。「インスタンスを追加」を選択します。
10. 「制御レコード (Control Record)」を展開します。次のフィールドに入力します。
    - IDoc_number
    - Sender_port
    - Partner_number_of_sender
    - Receiver_port
    - Partner_number_of_recipient
    - Client
    - SAP_Release
11. コネクターを始動します。
12. Test Connector で、「要求」>「送信」を選択します。ログに成功のメッセージがあるかどうか調べます。

ALE Module のイベント処理のテスト

ALE Module のイベント処理をテストするには、以下の手順を実行します。

1. トランザクション we19 の「IDoc 処理用テスト・ツール (Test Tool for IDoc processing)」に移動します。
2. フィールドに既存の IDoc を取り込みます。「IDoc」>「作成 (Create)」を選択します。
3. 「StandardOutboundProcessing」を選択して、IDoc をTest Connector に送信します。
4. ポップアップ・ウィンドウで、チェック・マークをクリックします。

5. IDoc が SAP から送信されたことを確認するため、mySAP.com コネクターのログ・ファイルに成功のメッセージがあるかどうか調べます。トランザクション sm58 にイベントが存在する場合は、このイベントは正しく送信されていません。
6. SAPALE_Archive_Queue に送信されたメッセージを表示して、ProcessingStatus が成功であるかどうか確認します。成功のメッセージが表示されない場合は、SAPALE_Error_Queue を確認して障害が発生していないかどうか調べます。

HDR Module のクイック・ステップ

HDR Module を構成する前に、次のコネクター固有のプロパティーを構成する必要があります。

プロパティー名	デフォルト値	必要な値
Modules	なし	BAPI

HDR Module でのビジネス・オブジェクトの生成

HDR Module でビジネス・オブジェクトを生成するには、以下の手順を実行します。

1. SAPODA を始動します。
2. Business Object Designer Express を始動します。
3. Business Object Designer Express で、「ファイル」>「新規」を選択します。ウィザードが開始します。
4. 「検索の構成」を選択します。
   a. 検索を実行するマシンのホスト・アドレスを入力します。
   b. 「ホストの追加 (Add Host)」を選択します。
   c. 「OK」を選択します。
5. ウィザードのステップ 3 で、「動的定義 (Dynamic Definitions)」を展開します。
   a. 「HDR」を展開します。
   b. 「名前で検索.... (Search by Name....)」を右クリックします。「項目を検索」を選択します。
   c. kna1 と入力します。「OK」を選択します。
   d. kna1 を強調表示します。「次へ」を選択します。
   e. 「次へ」を選択します。
   f. 「OK」を選択します。
6. 「通知 (Notification)」で、「いいえ」を選択します。
7. 「新規ビジネス・オブジェクト定義を個別のウィンドウで開く (Open new business object definition to a separate window)」を選択します。「完了」を選択します。
8. 新しいビジネス・オブジェクトをリポジトリー・ディレクトリーに保管します。

HDR Module のテストのための準備

HDR Module をテスト用にセットアップするには、ポート・コネクターを使用します。

1. SAP 構成ファイルをコピーします。コピーしたファイルの名前を portconnector.cfg に変更します。
2. Connector Configurator Express で portconnector.cfg を開きます。
3. 「標準」タブで次のプロパティーを変更します。
   - ApplicationName を PortConnector に
   - DELIVERYQUEUE を REQUESTQUEUE に
   - REQUESTQUEUE を RESPONSEQUEUE に
4. 変更内容を保管します。
5. sapconnector.cfg を開きます。
6. REQUESTQUEUE を SYNCHRONOUSREQUESTQUEUE に変更します。
7. 変更内容を保管します。

HDR Module のテスト

HDR Module をテストするには、以下の手順を実行します。

1. Test Connector を開きます。
2. 「ビジネス・オブジェクト・タイプ」で、SAP_kna1 を選択します。「作成」を選択します。
3. 「名前の入力」に、**新規オブジェクト**を入力します。「OK」を選択します。
4. 「動詞」を Retrieve に変更します。
5. customer_number_KUNNR に既存の SAP カスタマー番号を取り込みます。この番号は必ず 10 桁にします (例: 0000000001)。
6. 「要求」>「送信」を選択します。
7. ログ・ファイルに成功のメッセージがあるかどうか確認します。

付録 B. 標準構成プロパティー

この付録では、WebSphere Business Integration Server Express アダプターのコネクター・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。説明は、InterChange Server Express が対象となります。

このコネクターに固有のプロパティーについては、本書の該当するセクションを参照してください。

新規プロパティー

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。

- AdapterHelpName
- BiDi.Application
- BiDi.Broker
- BiDi.Metadata
- BiDi.Transformation
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- TivoliTransactionMonitorPerformance

標準コネクター・プロパティーの概要

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。フレームワークが使用します。
- アプリケーション固有またはコネクター固有の構成プロパティー。エージェントが使用します。

これらのプロパティーは、アダプターのフレームワークおよびエージェントの実行時の振る舞いを決定します。

このセクションでは、Connector Configurator Express の始動方法について説明し、すべてのプロパティーに共通する特性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 Express の始動

コネクター・プロパティーの構成は Connector Configurator Express から行います。Connector Configurator Express には、System Manager からアクセスします。Connector Configurator Express の使用法の詳細については、本書の Connector Configurator Express に関するセクションを参照してください。

Connector Configurator Express と System Manager は、Windows システム上でのみ動作します。コネクターを Linux システム上で稼働している場合でも、これらのツールがインストールされた Windows マシンが必要です。

Linux 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、Linux の統合ブローカーに接続してから、コネクター用の Connector Configurator Express を開く必要があります。

構成プロパティー値の概要

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。

1. デフォルト
2. InterChange Server Express 統合ブローカー用のリポジトリー
3. ローカル構成ファイル
4. コマンド行

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。STRING プロパティー・タイプの長さに制限はありません。INTEGER タイプの長さは、アダプターを実行しているサーバーによって決まります。

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。

プロパティーの更新特性 (すなわちコネクター・プロパティーへの変更を有効にする方法とタイミング) は、プロパティーの性質によって異なります。

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、新しい値が即時に有効になります。ただし、コネクターがスタンドアロン・モードの場合 (System Manager に依存しない) です。
- **エージェント再始動 (InterChange Server Express のみ)**
  コネクター・エージェントを停止して再始動しなければ、新規の値が有効になりません。
- **コンポーネント再始動**
  System Manager でコネクターを停止してから再始動しなければ、新しい値が有効になりません。エージェントまたはサーバー・プロセスを停止して再始動する必要はありません。
- **システム再始動**
  コネクター・エージェントおよびサーバーを停止して再始動しなければ、新規の値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 Express」ウィンドウ内の「**更新メソッド**」列を参照するか、317 ページの表 53 の「更新メソッド」列を参照してください。

標準プロパティーが存在できる場所が 3 箇所あります。一部のプロパティーは複数の場所にあってもかまいません。

- **ReposController**
  このプロパティーはコネクター・コントローラー内にあり、その場所でのみ有効です。エージェント・サイドで値を変更した場合、コントローラーには影響しません。
- **ReposAgent**
  このプロパティーはエージェント内にあり、その場所でのみ有効です。プロパティーによっては、ローカル構成によってこの値をオーバーライドされることがあります。
- **LocalConfig**
  このプロパティーは、コネクターの構成ファイル内にあり、構成ファイルを通じてのみ機能することができます。コントローラーはこのプロパティーの値を変更することができず、システムが再配置されてコントローラーが明示的に更新されなければ、構成ファイルに加えられた変更を認識しません。

標準プロパティーの早見表

表 53 は、標準コネクター構成プロパティーの早見表です。すべてのコネクターでこれらのプロパティーすべてを必要とするわけではなく、プロパティー設定は異なる場合があります。

各プロパティーの説明については、表の次のセクションを参照してください。

注: 表 53 の注の欄で、「RepositoryDirectory が <REMOTE> に設定され」という句は、ブローカーが InterChange Server Express であることを示します。

表 53. 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
AdapterHelpName	有効な <Regional Setting> ディレクトリーを含む <ProductDir>¥bin¥Data¥App¥Help 内の有効なサブディレクトリーのいずれか	テンプレート名 (有効な場合) またはブランク・フィールド	コンポーネント再始動	サポートされる地域設定。chs_chn、cht_twn、deu_deu、esn_esp、fra_fra、ita_ita、jpn_jpn、kor_kor、ptb_bra、および enu_usa (デフォルト) を含む。
AdminInQueue	有効な JMS キュー名	*<CONNECTORNAME>* /ADMININQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
AdminOutQueue	有効な JMS キュー名	*<CONNECTORNAME>* /ADMINOUTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
AgentConnections	1 から 4	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ または IDL であり、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。

表 53. 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
AgentTraceLevel	0 から 5	0	ICS では動的、その他の場合はコンポーネント再始動	
ApplicationName	アプリケーション名		コンポーネント再始動	
BiDi.Application	以下の双方向属性の任意の有効な組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント再始動	このプロパティーは、BiDi.Transformation の値が true の場合のみ有効です。
BiDi.Broker	以下の双方向属性の任意の有効な組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント再始動	このプロパティーは、BiDi.Transformation の値が true の場合のみ有効です。 BrokerType の値が ICS の場合、プロパティーは読み取り専用です。
BiDi.Metadata	以下の双方向属性の任意の有効な組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント再始動	このプロパティーは、BiDi.Transformation の値が true の場合のみ有効です。
BiDi.Transformation	true または false	false	コンポーネント再始動	このプロパティーは、BrokerType の値が WAS でない場合のみ有効です。
BrokerType	ICS	ICS	コンポーネント再始動	
CharacterEncoding	サポートされる任意のコード。 次のリストはその一部です。 ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437	ascii7	コンポーネント再始動	このプロパティーは、C++ コネクターでのみ有効です。
CommonEventInfrastructure	true または false	false	コンポーネント再始動	

表 53. 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
CommonEventInfrastructureURL	URL ストリング。例えば、corbaloc:iiop:host:2809。	デフォルト値はありません。	コンポーネント再始動	このプロパティーは、CommonEvent Infrastructure の値が true の場合のみ有効です。
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定されて、BrokerType の値が ICS の場合にのみ有効です。
ContainerManagedEvents	ブランクまたは JMS	ブランク	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
ControllerEventSequencing	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ControllerStoreAndForwardMode	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定されて、 BrokerType の値が ICS の場合にのみ有効です。
DeliveryQueue	任意の有効な JMS キュー名	*<CONNECTORNAME>*/DELIVERYQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
DeliveryTransport	IDL または JMS	RepositoryDirectory の値が <REMOTE> の場合は IDL。 それ以外の場合は JMS。	コンポーネント再始動	RepositoryDirectory の値が <REMOTE> ではない場合、このプロパティーの有効な値は JMS のみです。
DuplicateEventElimination	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	このプロパティーは、BrokerType の値が ICS である場合のみ有効です。

表 53. 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
FaultQueue	任意の有効なキュー名	*<CONNECTORNAME>* /FAULTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.FactoryClassName	CxCommon.Messaging.jms .IBMMQSeriesFactory、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	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.ListenerConcurrency	1 から 32767	1	コンポーネント再始動	このプロパティーは、jms.TransportOptimized の値が true の場合にのみ有効です。
jms.MessageBrokerName	jms.FactoryClassName の値が IBM の場合は、crossworlds.queue.manager を使用します。	crossworlds.queue. manager	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.NumConcurrent 要求	正整数	10	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.Password	任意の有効なパスワード		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.TransportOptimized	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS で、 BrokerType の値が ICS である場合にのみ有効です。
jms.UserName	任意の有効な名前		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。

表 53. 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
ListenerConcurrency	1 から 100	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ の場合のみ有効です。
Locale	これは、サポートされるロケールの一部です。 en_US、ja_JP、 ko_KR、zh_CN、 zh_TW、fr_FR、 de_DE、it_IT、 es_ES、pt_BR	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	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
MaxEventCapacity	1 から 2147483647	2147483647	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
MessageFileName	有効なファイル名	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	このプロパティーは、DuplicateEventElimination の値が true で、ContainerManagedEvents に値がない場合にのみ有効です。
OADAutoRestartAgent	true または false	false	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
OADMaxNumRetry	正整数	1000	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
OADRetryTimeInterval	正整数 (単位: 分)	10	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
PollEndTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	

表 53. 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
PollFrequency	正整数 (単位: ミリ秒)	10000	ブローカーが ICS の場合は動的。 そうでない場合は、コンポーネント再始動。	
PollQuantity	1 から 500	1	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS の場合のみ有効です。
PollStartTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
RepositoryDirectory	ブローカーが ICS の場合は <REMOTE>。 それ以外の場合は任意の有効なローカル・ディレクトリー。	ICS の場合、値は <REMOTE> に設定されます。	エージェント再始動	
RequestQueue	有効な JMS キュー名	*<CONNECTORNAME>*/REQUESTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
ResponseQueue	有効な JMS キュー名	*<CONNECTORNAME>*/RESPONSEQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
RestartRetryCount	0 から 99	3	ICS の場合は動的、その他の場合はコンポーネント再始動	
RestartRetryInterval	1 から 2147483647 までの値 (分単位)。	1	ICS の場合は動的、その他の場合はコンポーネント再始動	
RHF2MessageDomain	mrm または xml	mrm	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS であり、WireFormat の値が CwXML である場合のみ有効です。
SourceQueue	任意の有効な WebSphere MQ キュー名	*<CONNECTORNAME>*/SOURCEQUEUE	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS の場合のみ有効です。
SynchronousRequest Queue	任意の有効なキュー名	*<CONNECTORNAME>*/SYNCHRONOUSREQUEST QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。

表 53. 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
SynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
SynchronousResponse Queue	任意の有効なキュー名	<*CONNECTORNAME*> /SYNCHRONOUSRESPONSE QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
TivoliMonitorTransaction Performance	true または false	false	コンポーネント再始動	
WireFormat	CwXML または CwBO	CwXML	エージェント再始動	RepositoryDirectory の値が <REMOTE> に設定されていない場合、このプロパティーの値は、CwXML でなければなりません。RepositoryDirectory の値が <REMOTE> に設定されている場合、値は CwBO でなければなりません。
WsifSynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	BrokerType の値が ICS の場合、このプロパティーは無効です。
XMLNameSpaceFormat	short または long	short	エージェント再始動	BrokerType の値が ICS の場合、このプロパティーは無効です。

標準プロパティー

このセクションでは、標準コネクター構成プロパティーについて説明します。

AdapterHelpName

AdapterHelpName プロパティーは、コネクター固有の全般ヘルプ・ファイルがあるディレクトリーの名前です。ディレクトリーは、<*ProductDir*>¥bin¥Data¥App¥Help 内に配置される必要があり、少なくとも言語ディレクトリー enu_usa が含まれていなければなりません。ロケールに応じて、その他のディレクトリーが含まれることがあります。

デフォルト値は、テンプレート名が有効であればテンプレート名、有効でなければブランクです。

AdminInQueue

AdminInQueue プロパティーは、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を指定します。

デフォルト値は <*CONNECTORNAME*>/ADMININQUEUE です。

AdminOutQueue

AdminOutQueue プロパティーは、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を指定します。

デフォルト値は *<CONNECTORNAME>*/ADMINOUTQUEUE です。

AgentConnections

AgentConnections プロパティーは、ORB (オブジェクト・リクエスト・ブローカー) が初期化するときに開かれる ORB 接続の数を制御します。

このプロパティーのデフォルト値は 1 です。

AgentTraceLevel

AgentTraceLevel プロパティーは、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を設定します。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。

デフォルト値は 0 です。

ApplicationName

ApplicationName プロパティーは、コネクター・アプリケーションの名前を一意的に識別します。この名前は、システム管理者が統合環境をモニターするために使用します。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。

デフォルトはコネクターの名前です。

BiDi.Application

BiDi.Application プロパティーは、このアダプターがサポートする任意のビジネス・オブジェクトの形式で、外部アプリケーションからアダプターに入ってくるデータの双方向フォーマットを指定します。このプロパティーは、アプリケーション・データの双方向属性を定義します。これらの属性は以下のとおりです。

- テキストのタイプ: 暗黙または可視 (I または V)
- テキストの方向: 左から右または右から左 (L または R)
- 対称スワッピング: オンまたはオフ (Y または N)
- 成形 (アラビア語): オンまたはオフ (S または N)
- 数字成形 (アラビア語): ヒンディ語、コンテキスト、または名詞 (H、C、または N)

このプロパティーは、BiDi.Transformation プロパティーの値が true に設定されている場合のみ有効です。

デフォルト値は ILYNN (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Broker

BiDi.Broker プロパティーは、サポートされる任意のビジネス・オブジェクトの形式で、アダプターから統合ブローカーに送信されるデータの双方向フォーマットを指定します。データの双方向属性を定義します。属性は、前述の BiDi.Application の下にリストされています。

このプロパティーは、BiDi.Transformation プロパティーの値が true に設定されている場合のみ有効です。BrokerType プロパティーが ICS の場合、プロパティー値は読み取り専用です。

デフォルト値は ILYNN (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Metadata

BiDi.Metadata プロパティーは、メタデータの双方向フォーマットまたは属性を定義します。メタデータは、外部アプリケーションへのリンクを確立および保守するために、コネクターが使用します。属性の設定は、双方向機能を使用する各アダプターに固有です。アダプターが双方向処理をサポートする場合、詳細についてはアダプター固有のプロパティーに関するセクションを参照してください。

このプロパティーは、BiDi.Transformation プロパティーの値が true に設定されている場合のみ有効です。

デフォルト値は ILYNN (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Transformation

BiDi.Transformation プロパティーは、システムが実行時に双方向変換を実行するかどうかを定義します。

プロパティー値が true に設定されている場合、BiDi.Application、BiDi.Broker、および BiDi.Metadata プロパティーが使用可能です。プロパティー値が false に設定されている場合は、それらは非表示になります。

デフォルト値は false です。

BrokerType

BrokerType プロパティーは、使用している統合ブローカーのタイプを識別します。値は、ICS (InterChange Server Express) です。

CharacterEncoding

CharacterEncoding プロパティーは、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、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、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。サポートされる他の値をリストに追加するには、製品ディレクトリー (<*ProductDir*>) 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、本書の付録『Connector Configurator Express』を参照してください。

ConcurrentEventTriggeredFlows

ConcurrentEventTriggeredFlows プロパティーは、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されるビジネス・オブジェクトの数に設定します。例えば、このプロパティーの値を 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。

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、以下のプロパティーを構成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、複数のスレッドを使用できるようにコラボレーションを構成する必要があります。
- 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を、複数の要求を並行して処理できるように構成する必要があります。

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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 1 です。

ContainerManagedEvents

ContainerManagedEvents 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、1 つの JMS トランザクションとして宛先キューに配置されます。

このプロパティーを 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設定する必要があります。

- PollQuantity = 1 から 500
- SourceQueue = /SOURCEQUEUE

また、MimeType および DHClass (データ・ハンドラー・クラス) 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。DataHandlerConfigMOName

(オプションのメタオブジェクト名) を追加することもできます。これらのプロパティーの値を設定するには、Connector Configurator Express の「**データ・ハンドラー**」タブを使用します。

これらのプロパティーはアダプター固有ですが、以下に値の例をいくつか示します。

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

「**データ・ハンドラー**」 タブのこれらの値のフィールドは、ContainerManagedEvents プロパティーを JMS という値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。

ContainerManagedEvents プロパティーは、DeliveryTransport プロパティーの値が JMS に設定されている場合のみ有効です。

デフォルト値はありません。

ControllerEventSequencing

ControllerEventSequencing プロパティーは、コネクター・コントローラーでイベント順序付けを使用可能にします。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType は ICS) のみ有効です。

デフォルト値は true です。

ControllerStoreAndForwardMode

ControllerStoreAndForwardMode プロパティーは、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 InterChange Server Express (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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType プロパティーの値が ICS) のみ有効です。

デフォルト値は true です。

ControllerTraceLevel

ControllerTraceLevel プロパティーは、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を設定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 0 です。

DeliveryQueue

DeliveryQueue プロパティーは、コネクターが統合ブローカーへビジネス・オブジェクトを送信するときに使用するキューを定義します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS に設定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/DELIVERYQUEUE です。

DeliveryTransport

DeliveryTransport プロパティーは、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。Java Messaging Service の場合、値は JMS です。

- RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合、DeliveryTransport プロパティーの値には IDL または JMS を指定することができ、デフォルトは IDL です。
- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合、値に使用できるのは JMS のみです。

RepositoryDirectory プロパティーの値が 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。

デフォルト値は JMS です。

JMS

JMS トランスポート機構は、Java Messaging Service (JMS) を使用した、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 Express 内にリストされま

す。jms.MessageBrokerName プロパティーおよび jms.FactoryClassName プロパティーは、このトランスポートの必須プロパティーです。

InterChange Server Espress (ICS) が統合ブローカーである場合、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。

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・サイドの) コネクター・コントローラーと (クライアント・サイドの) 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 768MB 未満である場合には、次の変数およびプロパティーを設定してください。

- 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 (<*ProductDir*>) の下の ¥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、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」、または「*WebSphere Business Integration Server Express インストール・ガイド (i5/OS 版)*」を参照してください。

DuplicateEventElimination

このプロパティーの値が true の場合、JMS 対応コネクターでは重複イベントがデリバリー・キューへデリバリーされないようにすることができます。この機能を使用するには、コネクター開発時に、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。

注: このプロパティーの値が true の場合、保証付きイベント・デリバリーを提供するには、MonitorQueue プロパティーを使用可能にする必要があります。

デフォルト値は false です。

EnableOidForFlowMonitoring

このプロパティーの値が true の場合、アダプター・ランタイムは、着信 ObjectEventID にフロー・モニターの外部キーのマークを付けます。

このプロパティーは、BrokerType プロパティーが ICS に設定されている場合のみ有効です。

デフォルト値は false です。

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 (および状況標識と問題説明) を FaultQueue プロパティーで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。

jms.FactoryClassName

jms.FactoryClassName プロパティーは、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。DeliveryTransport プロパティーの値が JMS に設定されている場合、このプロパティーを設定する必要があります。

デフォルト値は CxCommon.Messaging.jms.IBMMQSeriesFactory です。

jms.ListenerConcurrency

jms.ListenerConcurrency プロパティーは、JMS コントローラーの並行リスナーの数を指定します。コントローラー内部で、並行してメッセージを取り出して処理するスレッドの数を指定します。

このプロパティーは、jms.OptimizedTransport プロパティーの値が true の場合のみ有効です。

デフォルト値は 1 です。

jms.MessageBrokerName

jms.MessageBrokerName は、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として (DeliveryTransport プロパティーで) 指定する場合、このコネクター・プロパティーを設定する必要があります。

リモート・メッセージ・ブローカーに接続した場合、このプロパティーでは以下の値を指定する必要があります。
QueueMgrName:*Channel*:*HostName*:*PortNumber*
ここで、以下のように説明されます。
QueueMgrName は、キュー・マネージャー名です。
Channel は、クライアントが使用するチャネルです。
HostName は、キュー・マネージャーの配置先のマシン名です。
PortNumber は、キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のようにします。

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

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

jms.NumConcurrentRequests

jms.NumConcurrentRequests プロパティーは、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はブロックされ、処理を続行するには他のいずれかの要求が完了するのを待機する必要があります。

デフォルト値は 10 です。

jms.Password

jms.Password プロパティーは、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。

デフォルト値はありません。

jms.TransportOptimized

jms.TransportOptimized プロパティーは、WIP (処理中の作業) が最適化されるかどうかを決定します。WIP を最適化するには、WebSphere MQ プロバイダーが必要です。最適化された WIP が作動するためには、メッセージング・プロバイダーが以下の操作を実行できなければなりません。

1. メッセージをキューから削除せずに読み取る。
2. メッセージ全体を受信側のメモリー空間に転送することなく、固有の ID を使用してメッセージを削除する。
3. 固有の ID を使用してメッセージを読み取る (リカバリーのために必要)。
4. 読み取られなかったイベントが現れるポイントを追跡する。

JMS API は、上記の条件 2 および 4 を満たさないため、最適化された WIP には使用できませんが、MQ Java API は 4 つの条件をすべて満たすため、最適化された WIP には必要です。

このプロパティーは、DeliveryTransport の値が JMS で、BrokerType の値が ICS である場合にのみ有効です。

デフォルト値は false です。

jms.UserName

jms.UserName プロパティーは、JMS プロバイダー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。

デフォルト値はありません。

JvmMaxHeapSize

JvmMaxHeapSize プロパティーは、エージェントの最大ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128M です。

JvmMaxNativeStackSize

JvmMaxNativeStackSize プロパティーは、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) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128K です。

JvmMinHeapSize

JvmMinHeapSize プロパティーは、エージェントの最小ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 1M です。

ListenerConcurrency

ListenerConcurrency 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 WebSphere 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。

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のみで有効です。DeliveryTransport プロパティーの値は `MQ` でなければなりません。

デフォルト値は 1 です。

Locale

Locale プロパティーは、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 は、2 文字の言語コード (小文字を使用) です。
TT は、2 文字の国または地域コード (大文字を使用) です。
codeset は、関連文字コード・セットの名前です (オプションの場合があります)。

デフォルトでは、サポートされるロケールの一部のみがリストされます。サポートされる他の値をリストに追加するには、*<ProductDir>*¥bin ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを変更します。詳細については、本書の付録『Connector Configurator Express』を参照してください。

コネクターが国際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、そのアダプターのユーザーズ・ガイドを参照してください。

デフォルト値は en_US です。

LogAtInterchangeEnd

LogAtInterchangeEnd プロパティーは、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。

ログ宛先にログを記録すると、E 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 ファイルで MESSAGE_RECIPIENT の値として指定された宛先に対する E メール・メッセージが生成されます。例えば、LogAtInterChangeEnd の値を 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、E メール・メッセージが送信されます。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は false です。

MaxEventCapacity

MaxEventCapacity プロパティーは、コントローラー・バッファー内のイベントの最大数を指定します。このプロパティーは、フロー制御機能によって使用されます。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

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

MessageFileName プロパティーは、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、製品ディレクトリーの ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクター・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: 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・ガイドを参照してください。

デフォルト値は InterchangeSystem.txt です。

MonitorQueue

MonitorQueue プロパティーは、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を指定します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS であり、DuplicateEventElimination の値が true である場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/MONITORQUEUE です。

OADAutoRestartAgent

OADAutoRestartAgent プロパティーは、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、WebSphere 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。WebSphere MQ によりトリガーされる OAD 機能の構成方法については、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」、または「*WebSphere Business Integration Server Express インストール・ガイド (i5/OS 版)*」を参照してください。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は false です。

OADMaxNumRetry

OADMaxNumRetry プロパティーは、異常シャットダウンの後で WebSphere MQ によりトリガーされる Object Activation Daemon (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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は 1000 です。

OADRetryTimeInterval

OADRetryTimeInterval プロパティーは、WebSphere MQ によりトリガーされる Object Activation Daemon (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。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は 10 です。

PollEndTime

PollEndTime プロパティーは、イベント・キューのポーリングを停止する時刻を指定します。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、値を含まない HH:MM であるため、この値は必ず変更する必要があります。

アダプター・ランタイムが以下のことを検出した場合、

- PollStartTime が設定されて、PollEndTime が設定されていない、または
- PollEndTime が設定されて、PollStartTime が設定されていない

PollFrequency プロパティーに構成された値を使用してポーリングします。

PollFrequency

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

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

このプロパティーでは、以下の値が有効です。

- ポーリング・アクション間のミリ秒数 (正整数)。
- 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。
- 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。

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。このようなコネクターが存在する場合には、アダプターのインストールと構成に関する章で制約事項が説明されています。

PollQuantity

PollQuantity プロパティーは、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。

このプロパティーは、DeliveryTransport プロパティーの値が JMS であり、ContainerManagedEvents プロパティーに値がある場合のみ有効です。

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

- 一度ポーリングされると、コネクターはメッセージの本文を検出し、それを添付ファイルとして読み取ります。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターはメッセージを無視します。
- コネクターは最初の BO 添付ファイルを処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- 二度目にポーリングされると、コネクターは 2 番目の BO 添付ファイルを処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- それが受け入れられると、3 番目の BO 添付ファイルが送信されます。

PollStartTime

PollStartTime プロパティーは、イベント・キューのポーリングを開始する時刻を指定します。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、値を含まない HH:MM であるため、この値は必ず変更する必要があります。

アダプター・ランタイムが以下のことを検出した場合、

- PollStartTime が設定されて、PollEndTime が設定されていない、または
- PollEndTime が設定されて、PollStartTime が設定されていない

PollFrequency プロパティーに構成された値を使用してポーリングします。

RepositoryDirectory

RepositoryDirectory プロパティーは、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保管されています。

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 Express 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、この値はデフォルトで <*ProductDir*>¥repository に設定されます。ただし、これには任意の有効なディレクトリー名を設定することができます。

RequestQueue

RequestQueue プロパティーは、統合ブローカーがコネクターへビジネス・オブジェクトを送信するときに使用するキューを指定します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS の場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/REQUESTQUEUE です。

ResponseQueue

ResponseQueue プロパティーは 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 InterChange Server Express (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。

このプロパティーは、DeliveryTransport プロパティーの値が JMS の場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/RESPONSEQUEUE です。

RestartRetryCount

RestartRetryCount プロパティーは、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に接続された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クライアント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

RestartRetryInterval プロパティーは、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にリンクされた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クライアント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。

プロパティーに使用可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

RHF2MessageDomain 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 WebSphere Message Broker 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。構成可能ドメイン名によって、WebSphere Message Broker がメッセージ・データを処理する方法を追跡できます。

ヘッダーの例を示します。

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

BrokerType の値が ICS の場合、このプロパティーは無効です。また、このプロパティーは、DeliveryTransport プロパティーの値が JMS であり、WireFormat プロパティーの値が CwXML である場合のみ有効です。

可能な値は、mrm および xml です。デフォルト値は mrm です。

SourceQueue

SourceQueue プロパティーは、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、326 ページの『ContainerManagedEvents』を参照してください。

このプロパティーは、DeliveryTransport の値が JMS であり、ContainerManagedEvents の値が指定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/SOURCEQUEUE です。

SynchronousRequestQueue

SynchronousRequestQueue プロパティーは、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、同期要求キューにメッセージを送信し、同期応答キューでブローカーからの応答を待機します。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。

このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。

デフォルト値は *<CONNECTORNAME>*/SYNCHRONOUSREQUESTQUEUE です。

SynchronousRequestTimeout

SynchronousRequestTimeout プロパティーは、コネクターが同期要求への応答を待機する時間をミリ秒単位で指定します。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 (およびエラー・メッセージ) を障害キューに移動します。

このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。

デフォルト値は 0 です。

SynchronousResponseQueue

SynchronousResponseQueue プロパティーは、同期要求に対する応答メッセージを、ブローカーからコネクター・フレームワークにデリバリーします。このキューは、コネクターが同期実行を使用する場合にのみ必要です。

このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。

デフォルトは *<CONNECTORNAME>*/SYNCHRONOUSRESPONSEQUEUE です。

TivoliMonitorTransactionPerformance

TivoliMonitorTransactionPerformance プロパティーは、IBM Tivoli Monitoring for Transaction Performance (ITMTP) を実行時に起動するかどうかを指定します。

デフォルト値は false です。

WireFormat

WireFormat プロパティーは、トランスポートでのメッセージ・フォーマットを指定します。

- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合、値は CwXML です。
- RepositoryDirectory プロパティーの値がリモート・ディレクトリーの場合、値は CwBO です。

付録 C. コネクター固有の構成プロパティー

コネクターには、コネクター固有の構成プロパティーと標準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。この付録では、Connector for mySAP.com に固有のプロパティーについて説明します。Connector Configurator Express の使用法については、13 ページの『第 2 章 コネクターの構成』を参照してください。

標準の構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。これらのプロパティーの詳細については、315 ページの『付録 B. 標準構成プロパティー』を参照してください。いくつかの標準の構成プロパティーには、表 54 に示すような Connector for SAP に固有の問題があります

表 54. このコネクターに固有のプロパティー情報

プロパティー	注
CharacterEncoding	コネクターでは、このプロパティーは使用されません。
Locale	このコネクターは国際化されているため、このプロパティーの値は変更できます。現在サポートされているロケールを知るには、アダプターのリリース情報を参照してください。
PollFrequency	イベントの処理のために RFC Server Module または ALE Module を使用している場合、このプロパティーの値を key または no に設定しないでください。この値を key または no に設定することにより、コネクターが始動時にこれらモジュールのインスタンスを作成することを防止することができます。

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

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。また、コネクター固有のプロパティーを使用すると、コネクターのコード変更や再ビルドを行わなくても、コネクター・フレームワークおよびコネクターのアプリケーション固有コンポーネント内の静的情報またはロジックを変更できます。

表 55 は、コネクター固有構成プロパティーの早見表です。「モジュール」列には、関連付けられているプロパティーを使用するコネクター・モジュールのリストが記載されています。

表 55. コネクター固有構成プロパティーの早見表

名前	指定可能な値	デフォルト値	モジュール
ABAPDebug	true または false	false	ABAP Extension BAPI HDR
AleEventDir	パス		ALE
AleUpdateStatus	true または false	false	ALE
AleSelectiveUpdate	*IDocType:MessageType*		ALE

表 55. コネクター固有構成プロパティーの早見表 (続き)

名前	指定可能な値	デフォルト値	モジュール
AleStatusMsgCode	*MessageCode*		ALE
AleSuccessCode	52 または 53	52	ALE
AleFailureCode	68 または 58	68	ALE
AleSuccessText	*SuccessText*		ALE
AleFailureText	*FailureText*		ALE
ApplicationPassword		SOFTWARE	すべて
ApplicationUserName		CROSSWORLDS	すべて
ArchiveDays			ALE
Client			すべて
Group	アプリケーション・サーバーのグループを表すログオン・グループの有効な名前		すべて
gwService	ゲートウェイ・サーバー *ID*	sapgw00	RFC Server ALE
Hostname	IP-address または *server-name*		すべて
InDoubtEvents	Reprocess、FailOnStartUp、 LogError、または Ignore	Ignore	ABAP Extension
Language		E	すべて
MaxNumberOfConnections		2	ABAP Extension、ALE (要求処理のみ)、BAPI HDR
Modules	*ModuleName*		すべて
Namespace	true または false	true	ABAP Extension
NumberOfListeners	*任意の正整数*	1	RFC Server、ALE
PollQuantity	*任意の正整数*	20	ABAP Extension、ALE
RefreshLogonCycle	true	true	すべて
RfcProgramId	プログラム *ID*	CWLDSERVER	RFC Server、ALE
RfcTraceOn	true または false	false	すべて
SAPALE_Archive_Queue	*任意の有効な WebSphere MQ キュー名*		ALE
SAPALE_Event_Queue	*任意の有効な WebSphere MQ キュー名*		ALE
SAPALE_Wip_Queue	*任意の有効な WebSphere MQ キュー名*		ALE
SAPALE_Error_Queue			
SAPALE_Unsubscribed_Queue			
SAPSystemID	*SAP システムの論理名*		すべて
SAPtid_MQChannel	*任意の有効な MQ チャネル*		ALE
SAPtid_MQPort	*任意の有効な MQ ポート*		ALE
SAPtid_Queue	*任意の有効な MQ キュー名*		ALE (要求処理のみ)
SAPtid_QueueManager	*任意の有効な MQ キュー・マネージャー名*		ALE
SAPtid_QueueManagerHost	*任意の有効な MQ キュー・マネージャー・ホスト名*		ALE
SAPtid_QueueManagerLogin	*任意の有効な MQ キュー・マネージャー・ログイン名*		ALE

表 55. コネクター固有構成プロパティーの早見表 (続き)

名前	指定可能な値	デフォルト値	モジュール
SAPtid_QueueManagerPassword	*任意の有効な MQ キュー・マネージャー・パスワード*		ALE
Sysnr	*システム番号*	00	すべて
DateTimeFormat	*なしまたはレガシー*		すべて
TransIdCollabName			**現在サポートはありません**
UpdateIDocStatus	true または false	True	ALE
IDocSuccessCode	12		ALE
IDocFailureCode	11		ALE
IDocSuccessText	Dispatched Okay		ALE
IDocFailureText	Dispatch failed		ALE
UseDefaults	true または false	false	ABAP Extension ALE BAPI

ABAPDebug

コネクターでビジネス・オブジェクトの処理を開始する際に、適切な機能モジュール用の ABAP Debugger を呼び出すかどうかを指定します。このプロパティーを true に設定すると、コネクターは次のコネクター・モジュール用の ABAP Debugger をオープンします。

- ABAP Extension — SAP からのイベントと、SAP へのサービス呼び出し要求を処理する場合。
- BAPI — SAP へのサービス呼び出し要求のみを処理する場合。
- 階層による動的検索 — SAP へのサービス呼び出し要求を処理する場合。

コネクターで ABAP Debugger が呼び出されるのは、次の場合のみです。

- **346 ページの『ApplicationUserName』**構成プロパティーのデフォルト値を、CROSSWORLDS から適切なユーザー権限を持つ Dialog ユーザーに変更した場合。
- ABAPDebug プロパティーを true に設定した場合。

注: ブレークポイントの追加を行えるのは、デバッガーが開いた後のみです。

重要: このプロパティーは、実稼働環境では常に false に設定する必要があります。

デフォルト値は false です。

AleEventDir

ALE Module がイベントの記録およびリカバリーのために使用する event ディレクトリーに対応するルート・ディレクトリー (¥ale) の場所を指定します。コネクターが最初に始動したときに、始動元のディレクトリー内にこのルート・ディレクトリーが存在しない場合、コネクターはルート・ディレクトリーと event サブディレクトリーを作成します。

- このプロパティーでパスが指定されている場合は、そのパスを使用してディレクトリーを作成する。

- パスの指定がない場合には、コネクターの始動元のディレクトリー内にルート・ディレクトリーを作成する。

例えば、コネクターの場所が ¥connectors¥SapConnector1 (製品ディレクトリー内) の場合、コネクターは次のディレクトリーを作成します。

```
¥connectors¥SapConnector1¥ale
```

Linux

コネクターを最初に始動する際に、コネクターが配置されているディレクトリー以外のディレクトリーから始動した場合、コネクターはこのプロパティーの値に関係なく、コネクターを始動したディレクトリーにルート・ディレクトリーを作成します。

OS/400 および i5/OS

コネクターを最初に始動する際に、コネクターが配置されているディレクトリー以外のディレクトリーから始動した場合、コネクターはこのプロパティーの値に関係なく、コネクターを始動したディレクトリーにルート・ディレクトリーを作成します。

詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値は次のとおりです。

Linux

$<ProductNameDir>/connectors/SAP/ale

OS/400 および i5/OS

$<ProductNameDir>/connectors/SAP/ale

Windows

%ProductNameDir%¥connectors¥SAP¥ale

AleUpdateStatus

すべてのメッセージ・タイプに対して監査証跡を必要とするかどうかを指定します。ALE Module がイベント処理のために IDoc オブジェクトを検索した後で、コネクターで標準の SAP 状況コードが更新されるようにするためには、このプロパティーを true に設定する必要があります。

詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値は false です。

AleSelectiveUpdate

コネクターで標準の SAP 状況コードが更新されるように構成した場合、どの IDocType と MessageType の組み合わせを更新するかを指定します。このプロパティーの値は、AleUpdateStatus が true に設定されている場合にのみ定義できます。

このプロパティーの構文は次のとおりです。

IDocType:MessageType[,*IDocType:MessageType* [,...]]

ここで、コロン (:) 区切り文字はそれぞれの IDocType と MessageType を分離し、コンマ (,) 区切り文字はエントリーをセットごとに分離します。下の例では、セットが 2 つある場合が示されています。この例で、MATMAS03 と DEBMAS03 は IDoc、MATMAS と DEBMAS はメッセージ・タイプです。

```
MATMAS03:MATMAS,DEBMAS03:DEBMAS
```

詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

AleStatusMsgCode

必要があれば、コネクターが ALEAUD メッセージ IDoc (ALEAUD01) を通知する際に使用するメッセージ・コードを指定します。このメッセージ・コードは、受信側のパートナー・プロファイルで構成してください。このプロパティーの値は、AleUpdateStatus が true に設定されている場合にのみ定義できます。

詳細については、 127 ページの『IDoc の状況を更新するための SAP の構成』を参照してください。

AleSuccessCode

Application Document Posted の正常状況コードを指定します。ALE Module がイベント処理のために IDoc オブジェクトを検索した後で、コネクターで SAP 正常状況コードが更新されるようにするには、このプロパティーに値を指定する必要があります (52 または 53)。SAP では、この値を状況 41 (Application Document Created in Receiving System) に変換します。

詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

AleFailureCode

ディスパッチ失敗に対する状況コードを指定します。ALE Module がイベント処理のために IDoc オブジェクトを検索した後で、コネクターで SAP 失敗状況コードが更新されるようにするには、このプロパティーに値を指定する必要があります (68 または 58)。SAP はこの値を 40 に変換します。

詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

AleSuccessText

正常な Application Document Posted に対する記述テキストを指定します。このプロパティー値の設定は、AleUpdateStatus を true に設定した場合でも省略可能です。

詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

AleFailureText

ディスパッチ失敗に対する記述テキストを指定します。このプロパティー値の設定は、AleUpdateStatus を true に設定した場合でも省略可能です。

詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

ApplicationPassword

SAP アプリケーションでのコネクターのユーザー・アカウントのパスワードです。デフォルトは SOFTWARE です。

ApplicationUserName

SAP アプリケーションでのコネクターのユーザー・アカウントの名前です。デフォルトは CROSSWORLDS です。

ArchiveDays

TIDManagement ファイルを要求ディレクトリーから削除する必要が生じたあとの日数は、ArchiveDays コネクター構成のプロパティーによって決まります。内部的に保守されるデフォルト値は、7 日です。半端な日数値 (例: 1.234) を指定することもできます。

Client

コネクターがログインする際のクライアント番号で、多くの場合 100 です。

Group

ロード・バランシング用のコネクターを構成しているときには、アプリケーション・サーバーのグループを表すログオン・グループの名前を指定します。詳細については、 33 ページの『ロード・バランシングの活用』を参照してください。

gwService

ゲートウェイ・サーバーの ID。多くの場合、sapgw00 です。00 は、SAP Gateway を実行しているサーバー (通常はアプリケーション・サーバー) のシステム番号で、複数存在する場合は 00 を指定することはできません。デフォルトは sapgw00 です。

Hostname

ロード・バランシング用のコネクターを構成しているときには、メッセージ・サーバーの名前を指定します。ロード・バランシングなしに実行するコネクターを構成しているときには、コネクターのログイン先となるアプリケーション・サーバーの IP アドレスまたは名前を指定します。いずれの場合も、コネクターは、ゲートウェイ・ホストの名前がこのプロパティーに対して指定された値と同じと仮定します。

InDoubtEvents

InDoubtEvents は、イベント表の進行中のイベントの処理法を記述します。`Reprocess` は、イベント表の進行中のイベントを再処理します。`FailOnStartup` は、コネクターをシャットダウンし、進行中のイベントがあった場合は致命的エラーをログに記録します。`LogError` は、進行中のイベントがイベント表にあることを通知するエラーをログに記録します。`Ignore` は、進行中のイベントを無視します。

Language

コネクターがログインする際の言語です。デフォルトは、`E` で、英語に設定されます。

MaxNumberOfConnections

コネクターと SAP アプリケーションとの間の、同時実行可能な相互作用の最大数。これらの相互作用には、イベントのポーリングや、サービス呼び出し要求の処理が含まれます。このプロパティーは、ABAP Extension Module、BAPI Module、および ALE Module でのみ使用されます。ALE Module は、サービス呼び出し要求のためにのみこのプロパティーを使用します。

各相互作用は SAP アプリケーション・サーバー上でダイアログ・プロセスを使用するため、接続の数は、使用可能なダイアログ・プロセスの数を超えることはできません。詳細については、xxiii ページの『同時実行可能な複数の相互作用の処理』を参照してください。

このプロパティーに値を設定しない場合、コネクターではデフォルト値の 2 が使用されます。

Modules

コネクターが `init()`、`pollForEvents()`、および `Terminate()` の要求を実行するために使用するモジュールを指定します。具体的には、Vision コネクター・フレームワークで使用されるコネクター・モジュールを指定します。複数のコネクター・モジュールを指定する場合は、それぞれの値をコンマで区切ります。スペースは入れないでください。

サポートされるコネクター・モジュールと、それらを指定するための構文は次のとおりです。

ABAP Extension Module—`Extension`

ALE Module—`ALE`

BAPI Module—`Bapi`

Namespace

コネクターが、コネクター・ネーム・スペース /CWLD/ で定義された ABAP コンポーネントを使用するかどうかを指定します。コネクターがネーム・スペースで定義された ABAP コンポーネントを使用するようにするには、値を true に設定する必要があります。デフォルトは true です。

NumberOfListeners

コネクターが初期化される際に作成されるリスナー・スレッドの数を指定します。1 つのリスナー・スレッドは、一度に 1 つの要求を処理できます。それぞれのリスナー・スレッドは一度に 1 つのイベントを処理します。そのため、複数のリスナー・スレッドが作成されていると、コネクターは複数のイベントを並行処理できます。デフォルトは、1 です。

リスナー・スレッドの数は、SAP で使用可能な作業プロセスの数を超えないことを推奨します。

PollQuantity

単一のポーリングで選出されるイベントの最大数を定義します。デフォルトは 20 です。

RefreshLogonCycle

SAP クライアントの接続について、すべてのリソースを解放するかどうかを指定します。デフォルトは false です。

RfcProgramId

リスナー・スレッドが RFC 対応機能からのイベントを処理できるように、コネクターが SAP Gateway に登録する ID です。この値は、SAP アプリケーションに登録されたプログラム ID (トランザクション SM59) と一致している必要があります。デフォルトは CWLDSERVER です。

SAP アプリケーション内のプログラム ID の構成の詳細については、165 ページの『RFC Server Module の SAP gateway への登録』を参照してください。

RfcTraceOn

各リスナー・スレッドに関する RFC アクティビティーの詳細を記述したテキスト・ファイルを生成するかどうかを指定します。指定できる値は true または false です。true を指定するとトレースが活動化し、トレースがテキスト・ファイルに生成されます。これらのテキスト・ファイルは急速に大きくなるため、これらのファイルを開発環境のみで使用することをお勧めします。デフォルトは false です。

SAPALE_Archive_Queue

ALE Module がイベントの処理を終えた後、TID と IDoc データをアーカイブする WebSphere MQ キューを指定します。詳細については、117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPALE_Event_Queue

ALE Module によるイベント処理の間、TID と IDoc データを保管する WebSphere MQ キューを指定します。詳細については、117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPALE_Wip_Queue

ALE Module がイベント・キューあての MQ メッセージを作成しているとき、TID と IDoc データを保持する WebSphere MQ 処理中 (wip) キューを指定します。コネクターは、1 つのイベントのデータをすべて受信した後、このキューのデータを SAPALE_Event_Queue に移します。詳細については、117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPALE_Error_Queue

WIP キューとイベント・キューの間で失敗した MQ メッセージを処理するためのキューを定義します。詳細については、117 ページの『第 9 章 ALE Module の概要』を参照してください。

SAPALE_Unsubscribed_Queue

アンサブスクライブされた IDoc オブジェクトを収集するためのキューを定義します。以前は、アンサブスクライブされた IDoc オブジェクトはアーカイブ・キューに置かれていました。これらのメッセージは、イベント管理ユーティリティーを使用して再サブミットできます。コネクターは、SAP からコネクターへのデータを処理するときにサブスクリプションを検査するようになったため、トランザクションはコラボレーションが開始されるまで SAP に残ります。詳細については、117 ページの『第 9 章 ALE Module の概要』を参照してください。

SAPSystemID

ロード・バランシング用のコネクターを構成しているときには、SAP システムの論理名 (R3name とも呼びます) を指定します。詳細については、33 ページの『ロード・バランシングの活用』を参照してください。

SAPtid_MQChannel

WebSphere MQ キュー・マネージャー用のクライアント・チャネルを指定します。詳細については、117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPtid_MQPort

ALE Module 用のキューを処理する WebSphere MQ キュー・マネージャーとの通信に使用するポートを指定します。詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPtid_Queue

TID と TID 状況が設定されているメッセージが存在する WebSphere MQ キューを指定します。このプロパティーは、ALE Module が要求を処理するときにのみ使用します。詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPtid_QueueManager

TID および IDoc データを格納するキューを管理する WebSphere MQ キュー・マネージャーの名前です。このプロパティーは、ALE Module がイベントおよび要求を処理するときに使用します。詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPtid_QueueManagerHost

WebSphere MQ キュー・マネージャーが存在するホストの名前です。このプロパティーは、ALE Module がイベントおよび要求を処理するときに使用します。詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPtid_QueueManagerLogin

WebSphere MQ キュー・マネージャーにログインするためのユーザー名です。このプロパティーは、ALE Module がイベントおよび要求を処理するときに使用します。詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

SAPtid_QueueManagerPassword

WebSphere MQ キュー・マネージャーにユーザーがログインするためのパスワードです。このプロパティーは、ALE Module がイベントおよび要求を処理するときに使用します。詳細については、 117 ページの『第 9 章 ALE Module の概要』を参照してください。

デフォルト値はありません。

Sysnr

アプリケーション・サーバーのシステム番号です。この値は 2 桁の数値で、多くの場合 00 です。デフォルトは 00 です。

DateTimeFormat

DATE および TIME の各フィールド値で指定された区切り文字を保存します。Legacy に設定すると、コネクターは DATE および TIME フィールドの区切り文字を保存します。それ以外の場合は、区切り文字が除去され、値の長さが属性で定義された長さに合わせられます。

TransIdCollabName

重要: コネクターによるこのプロパティーのサポートはなくなりました。

UpdateIDocStatus

監査証跡がすべてのメッセージ・タイプに必要とするかどうかを指定します。

IDocSuccessCode

正常なディスパッチの場合の標準の IDoc 状況コード。

IDocFailureCode

ディスパッチの失敗の場合の標準の IDoc 状況コード。

IDocSuccessText

正常なディスパッチの場合の、IDocSuccessCode に関連付けられた IDoc 状況メッセージのテキスト。

IDocFailureText

ディスパッチの失敗の場合の、IDocFailureCode に関連付けられた IDoc 状況メッセージのテキスト。

UseDefaults

Create または Update 操作の場合、UseDefaults が true に設定されていれば、統合ブローカー用のアダプター・フレームワークは、必要に応じてマーク付けされた各ビジネス・オブジェクト属性に有効な値またはデフォルト値が指定されているか検査します。値が指定されている場合には、Create または Update は正常に実行されます。パラメーターが false に設定されていると、コネクターは有効な値のみチェックし、値が設定されていなければ、Create 処理または Update 処理は失敗します。デフォルトは false です。

付録 D. IBM WebSphere BI Station のサポート・レベル

この付録では、すべての IBM WebSphere BI Station ツールのリストと、それらがサポートされる環境を示します。これらのツールには、開発環境のみでサポートされるものと、実稼働環境と開発環境の両方でサポートされるものがあります。

重要: これらのツールの出力は、ユーザーの責任です。IBM では、実稼働環境での開発ツールの使用はサポートされていません。

この章の内容は以下のとおりです。

IBM WebSphere BI Station にアクセスするには、SAP アプリケーションでトランザクション `/n/CWLD/HOME` を使用します。これは、各種ツールにアクセスするためのボタンのある、複数のタブから構成されています。この付録の表に、各ツールのサポート・レベルを示します。

「Development」タブ

「Development」タブには開発用のツールのみが含まれています。表 56 に使用可能なツールを示します。

表 56. 「Development」タブ

セクション	ボタン	開発専用	開発および実動両用
Transaction Based -Inbound			
	Inbound Wizard	X	
	Modify BO Meta Data	X	
Transaction Based - Outbound — Flat			
	Modify BO Meta Data	X	
	Modify Long Text	X	

「Tools」タブ

「Tools」タブには開発用のツールのみが含まれています。表 57 に使用可能なツールを示します。

表 57. 「Tools」タブ

セクション	ボタン	開発専用	開発および実動両用
Tools			
	Config objects	X	
	Configuration values	X	
	Transport layer	X	

表 57. 「Tools」タブ (続き)

セクション	ボタン	開発専用	開発および実動両用
	Log object links	X	
	Delete object	X	
	Object metadata	X	
	Reprocess object	X	

「Management」タブ

「Management」タブには、開発および実動両用のツールが含まれています。表 58 に使用可能なツールを示します。

表 58. 「Management」タブ

セクション	ボタン	開発専用	開発および実動両用
Activity			
	Log		X
	Gateway		X
Event queues			
	Current events		X
	Future events		X
	Archived events		X
Online			
	Delete Event queue		X
	Delete Event archive		X
	Delete log		X
	Del Obj archive		X

「Configuration」タブ

「Configuration」タブには開発および実動両用のツールと共に開発専用のツールが含まれています。表 59 に使用可能なツールを示します。

表 59. 「Configuration」タブ

セクション	ボタン	開発専用	開発および実動両用
Global Settings			
	Configuration values	X	
	Event Distribution	X	
	Event restriction	X	
	Logging level		X
User Settings			X

「Troubleshooting」タブ

「Troubleshooting」タブには開発および実動両用のツールが含まれています。表 60 に使用可能なツールを示します。

表 60. 「Troubleshooting」タブ

セクション	ボタン	開発専用	開発および実動両用
Local Tools			

表 60. 「Troubleshooting」タブ (続き)

セクション	ボタン	開発専用	開発および実動両用
	Short dump		X
Customer Support			
	Log		X
	Short dump		X
	Event restriction		X
	CrossWorlds config		X
	Object metadata		X

索引

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

[ア行]

[カ行]

［サ行］

［タ行］

［ハ行］

［マ行］

［ヤ行］

［ラ行］

A

B

C

D

G

H

I

J

O

R

S

T

U

V

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
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。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。本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。著作権使用許諾: 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

以下は、IBM Corporation の商標です。
IBM
IBM ロゴ
AIX
CICS
CrossWorlds
DB2
DB2 Universal Database
i5/OS
IMS
Informix
iSeries
Lotus
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS

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 および Pentium 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。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

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。

WebSphere Business Integration Server Express and Express Plus には、Eclipse Project (http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

WebSphere Business Integration Server Express バージョン 4.4、および WebSphere Business Integration Server Express Plus バージョン 4.4

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12