IBM WebSphere Business Integration Server
Express and Express Plus

アクセス開発ガイド

バージョン 4.3.1

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、 111 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Server Express バージョン 4.3.1 および IBM WebSphere Business Integration Server Express Plus バージョン 4.3.1 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
Access Development Guide
Version 4.3.1

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

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

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

目次

本書について

製品 IBM(R)WebSphere Business Integration Server Express および IBM(R) 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。

本書では、IBM サーバー・アクセス API を使用して、コール・トリガー・フロー機能を使用可能にする方法について説明します。コール・トリガー・フローは、アクセス・クライアント・プロセスによって開始させるフローです。アクセス・クライアント・プロセスでは、ビジネス・オブジェクトを作成したり、コラボレーションを実行したりすることができます。

特に明記されていない限り、本書の情報は、いずれも、IBM WebSphere Business Integration Server Express と IBM WebSphere Business Integration Server Express Plus の両方に当てはまります。WebSphere Business Integration Server Express という用語と、これを言い換えた用語は、これらの 2 つの製品の両方を指します。

対象読者

本書は、コラボレーションの作成や変更を担当する IBM WebSphere のお客様、コンサルタント、および販売代理店を対象としています。実際の作業を開始する前に、「システム・インプリメンテーション・ガイド」で説明されている概念をお読みください。

サーバー・アクセス API を実装するには、Java (TM) プログラム言語をはじめ、標準プログラミングの概念および手法を知る必要があります。サーバー・アクセス・インターフェース API は Java プログラム言語をベースにしています。

本書の前提条件

本書では、仕様、フローチャート、または手書きの設計から開始することを前提としています。ビジネス・プロセスの分析、コラボレーションまたはコネクターの開発、またはビジネス・オブジェクトの設計に関する説明は含まれません。

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 システムの場合、円記号の代わりにスラッシュ (/) を使用します。すべてのファイルのパス名は、使用システムで IBM 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。

本書の使用方法

「*IBM WebSphere Business Integration Server Express and Express Plus アクセス開発ガイド*」は次のように編成されています。

第 1 部: 始めに

3 ページの『第 1 章 サーバー・アクセス機能の概要』	サーバー・アクセスの概説です。
13 ページの『第 2 章 アクセス・クライアント環境のセットアップ』	開発および実行時環境をインストールおよびセットアップする方法について説明します。
17 ページの『第 3 章 コール・トリガー・フロー用のコラボレーションの構成』	アクセス・クライアントと併用するためにコラボレーションを構成する方法について説明します。
23 ページの『第 4 章 アクセス・クライアントの実装』	アクセス・クライアントを実装してコラボレーションを実行する方法を概説します。

第 2 部: チュートリアル

31 ページの『第 5 章 HTML データ処理機能付きサーブレットのサンプル』	API を使用する Java で作成されたサーブレットについて説明します。

第 3 部: サーバー・アクセス API リファレンス

53 ページの『第 6 章 IAccessEngine インターフェース』	`IAccessEngine` インターフェースでのメソッドの使用方法を示す、構文とコードの断片を紹介します。
55 ページの『第 7 章 IInterchangeAccessSession インターフェース』	`IInterchangeAccessSession` インターフェースでのメソッドの使用方法を示す、構文とコードの断片を紹介します。
65 ページの『第 8 章 IBusinessObject インターフェース』	`IBusinessObject` インターフェースでのメソッドの使用方法を示す、構文とコードの断片を紹介します。
93 ページの『第 9 章 IBusinessObjectArray インターフェース』	`IBusinessObjectArray` インターフェースでのメソッドの使用方法を示す、構文とコードの断片を紹介します。
99 ページの『第 10 章 サーバー・アクセス例外』	サーバー・アクセス API の例外について説明します。

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。

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、リテラル値を示します。
太字	初出語を示します。
イタリック	変数名または相互参照を示します。InterChange Server 資料を PDF ファイルとして開くと、相互参照はイタリックの青字で表示されます。相互参照をクリックすることにより、目的の情報にジャンプできます。
イタリック Courier フォント	リテラル・テキスト内の変数名を示します。
枠付きの `courier`	コード・フラグメントをテキストの他の部分と区別します。
青い文字	オンラインで表示したときにのみ見られる青の部分は、相互参照用のハイパーリンクです。青い文字ストリングをクリックすることにより、参照先オブジェクトに飛ぶことができます。
{ }	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 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを意味します。

本リリースの新機能

本章では、IBM WebSphere Business Integration Server Express and Express Plus 開発環境用の「アクセス開発ガイド」で取り上げる新機能について説明します。

リリース 4.3.1

本書の最初のリリースです。

本リリースでは、以下のオペレーティング・システムのサポートが追加されました。

- IBM OS/400 V5R2、V5R3
- Red Hat enterprise Linux 3.0
- SuSE Linux Enterprise Server 8.1
- Microsoft Windows 2003 (実動モードでの InterChange Server Express およびアダプターのみ)

第 1 部 始めに

第 1 章 サーバー・アクセス機能の概要

InterChange Server Express の**サーバー・アクセス**機能は、IBM WebSphere Business Integration Server Express and Express Plus 内のコラボレーション実行の要求を外部プロセスで行えるようにする API です。**アクセス・クライアント**と呼ばれる、この外部プロセスでは、アクセス要求を送信してコール・トリガー・フローを開始します。

本章では、サーバー・アクセスについて、企業間コネクティビティーを実現する方法について、サーバー・アクセス API を使用したサイト固有のソリューションの開発を始める方法について概説します。

本章のセクションは、以下のとおりです。

コール・トリガー・フロー

サーバー・アクセスは、IBM WebSphere Business Integration Server Express and Express Plus 内のコラボレーション実行の要求を外部プロセスで行えるようにする API です。**コラボレーション**は、いくつかのアプリケーションを含めることができるビジネス・プロセスを表します。サーバー・アクセスを使用して、**アクセス・クライアント**と呼ばれるこの外部プロセスでは、IBM WebSphere Business Integration Server Express and Express Plus がコラボレーションの実行を介して処理するアプリケーションからデータを取得することができます。

サーバー・アクセス機能により、IBM WebSphere Business Integration Server Express and Express Plus は、コネクターからトリガー・イベントを受け取ることなしに、コラボレーション実行の要求を直接受け取ることができます。アクセス・クライアントが送信する要求は、**アクセス要求**と呼ばれます。アクセス要求を送信するとき、アクセス・クライアントは、実際にイベントを送信するのではなく、サーバー・アクセス内のメソッドに対して呼び出しを発行します。そのため、コネクターが開始するイベントにより起動されたフローに代わって、アクセス・クライアントが開始するフロー・トリガーは**コール・トリガー・フロー**と呼ばれます (図 1 を参照)。

コール・トリガー・フローは、イベントにより起動されたフローが効率よく容易に処理されます。運用上の主な違いは、コール・トリガー・フローは同期的に処理されるため、IBM WebSphere Business Integration Server Express and Express Plus システム内で永続的ではない ということです。対照的に、イベントにより起動されたフローは非同期的に処理されるため、永続的です。システムにおけるこれらのフローの処理方法の詳細については、「*システム・インプリメンテーション・ガイド*」を参照してください。

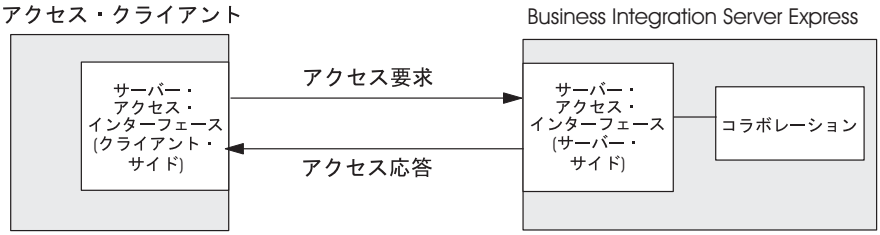

図 1. コール・トリガー・フロー

図 1 に示すように、アクセス・クライアントが開始するアクセス要求には、次の手順があります。

1. アクセス・クライアントが**トリガー・アクセス・データ**を作成します。アクセス要求時に、アクセス・クライアントはこのトリガー・アクセス・データを IBM WebSphere Business Integration Server Express and Express Plus に送信します。このデータは、指定されたコラボレーションを起動するデータです。すなわち、実行を開始するには、コラボレーションはこのデータを必要とします。
2. IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスに**トリガー・アクセス呼び出し**を送信するためにアクセス・クライアントがサーバー・アクセス API のメソッドを呼び出します。トリガー・アクセス呼び出しには、トリガー・アクセス・データ、および実行するコラボレーションの名前が含まれています。このメソッド呼び出しを介して、アクセス・クライアントはアクセス要求を実行します。これで、コール・トリガー・フローが開始されます。
3. IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスがトリガー・アクセス呼び出しを受け取り、システム・ビジネス・オブジェクトへのトリガー・アクセス・データの必要なすべての変換を実行します。このデータ変換の詳細については、5 ページの『IBM WebSphere Business Integration Server Express and Express Plus のデータ・ハンドラーの役割』を参照してください。
4. IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスがトリガー・アクセス・データを指定されたコラボレーションに送信し、コラボレーションの実行が起動されます。
5. コラボレーションが完了すると、コラボレーションは結果として生成されたビジネス・オブジェクトをサーバー・アクセスに送信します。
6. サーバー・アクセス・インターフェースは、結果として生成されたビジネス・オブジェクトからトリガー・アクセス・データの元のフォーマットへのすべての必要な変換を実行します。続いて、**アクセス応答**を実行して、アクセス応答データをアクセス・クライアントに送り戻します。このデータ変換の詳細については、

5 ページの『IBM WebSphere Business Integration Server Express and Express Plus のデータ・ハンドラーの役割』を参照してください。

本セクションでは、コール・トリガー・フローについて次の追加情報を提供します。

- IBM WebSphere Business Integration Server Express and Express Plus のデータ・ハンドラーの役割
- コール・トリガー・フローの例

IBM WebSphere Business Integration Server Express and Express Plus のデータ・ハンドラーの役割

IBM WebSphere Business Integration Server Express and Express Plus のデータ・ハンドラーは、直列化データと IBM WebSphere ビジネス・オブジェクトの間の変換を行います。これらのデータ・ハンドラーは、直列化データ用の各種のデータ・フォーマットをサポートします。サーバー・アクセス API を使用して、アクセス・クライアントはフォーマット済みのトリガー・イベントをいくつかの異なるフォーマットの 1 つで送信することができます。トリガー・アクセス・データが XML である場合、IBM WebSphere Business Integration Server Express and Express Plus のサーバー・アクセスは XML データ・ハンドラーを呼び出します。XML データ・ハンドラーはトリガー・アクセス・データを解析して、それを IBM WebSphere データ・フォーマットに変換します (ビジネス・オブジェクト)。オプションとして、アクセス・クライアントは、コラボレーション応答からの結果として生成されたビジネス・オブジェクトをサーバー・アクセスに渡すことができます。サーバー・アクセスは、元の着信フォーマット (この場合は、XML) に変換するための適切なデータ・ハンドラーを呼び出します。

データ・ハンドラーを起動するには、サーバー・アクセスはまず、データ・ハンドラー・インスタンスの作成に使用する、トップレベル・データ・ハンドラー・メタオブジェクトを位置指定する必要があります。 IBM WebSphere Business Integration Server Express and Express Plus のトップレベル・メタオブジェクトは `MO_Server_DataHandler` で、これは IBM WebSphere Business Integration Server Express and Express Plus と同じマシン上にあります。サーバー・アクセス開発ソフトウェアには、XML データ・ハンドラー、EDI データ・ハンドラー、NameValue データ・ハンドラー、FixedWidth データ・ハンドラー、および Delimiter データ・ハンドラーが組み込まれています。また、このソフトウェアは、カスタム・データ・ハンドラーの開発もサポートします。デフォルトでは、`MO_Server_DataHandler` メタオブジェクトは、サーバー・アクセス・インターフェースがアクセス・クライアントからの直列化データを受け取ったときに XML データ・ハンドラーを自動的に呼び出すように構成されています。アクセス・クライアントが XML 以外のフォーマットの直列化データを使用する場合には、該当するデータ・ハンドラーをサポートするように、この `MO_Server_DataHandler` メタオブジェクトを修正してください。詳しくは、「データ・ハンドラー・ガイド」を参照してください。

コール・トリガー・フローの例

サーバー・アクセスは、サプライヤー、ベンダー、またはネットワーク化された企業ユニットによる、バックエンド・アプリケーションへのセキュアな信頼できる外部アクセスを必要とする企業間トランザクションをサポートします。次に、架空の企業 2 社 (企業 A および企業 B) を使った企業間取引の例を示します。

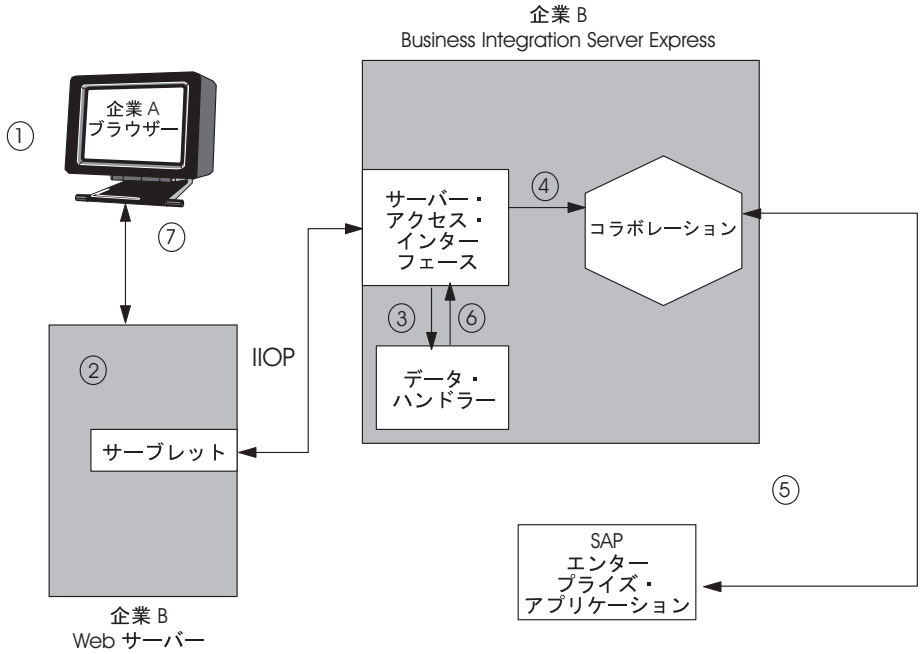

図 2. 企業間取引の例

この例で、企業 A は企業 B に 1,000 個の IC を注文します。企業 A をはじめとする与信済み企業に対して、企業 B は、自社の IBM WebSphere Business Integration Server Express and Express Plus 統合のバックエンドへのコール・トリガー・フローをサポートします。このプロセスは次のように展開されます。

1. 企業 A の社員が、アカウント ID およびパスワードを入力して、企業 B の Web サイトにログインします。続いて、社員は 1,000 個の IC を発注します。企業 B の Web サーバーは、このユーザーを与信済みベンダーとして認証します。
2. アクセス・クライアントは、企業 B の e-business サーバー (IBM WebSphere Business Integration Server Express and Express Plus) でコール・トリガー・フローを開始します。企業 B のサーバー・アクセスは、アクセス・クライアントから API 呼び出しを受け取って処理します。トリガー・アクセス呼び出しは、データが XML フォーマットであることを示します。
3. 企業 A のコール・トリガー・フローはデータを XML データ・ハンドラーに渡します。このデータ・ハンドラーは、直列化データを企業 B の汎用ビジネス・オブジェクト・フォーマットに変換します。ビジネス・オブジェクト定義は、XML データ・ストリーム中の DTD およびデータ・ハンドラー・メタオブジェクトから取り出されます。
4. 企業 A のアクセス・クライアントは、企業 B の IBM WebSphere Business Integration Server Express and Express Plus 内のコラボレーションを実行し、

Order_Generation プロセスを起動します。ビジネス・オブジェクトは、適切に構成された IBM WebSphere コラボレーション (アクセス・クライアント機能付きポートにバウンドされ、そのポートとの間でデータをやり取りするマップが含まれているコラボレーション) を使用します。

5. ビジネス・オブジェクトは、SAP 用のアダプターに経路指定されています。このアダプターは企業 B の SAP/R3 アプリケーションにアクセスして、発注します。(企業 B は、この注文を配送業務用のサプライヤーのサイトに経路指定します。) その結果である、注文の確認が生成されて、コネクターを介してアクセス・クライアントに渡されます。
6. 企業 A のアクセス・クライアントは、結果として生成されたビジネス・オブジェクトを XML データ・ハンドラーに送信します。XML データ・ハンドラーは、結果を解析して、XML データ・ストリームに変換します。
7. 結果は Web サーバー・サイトにストリームされます。これによって、別個のプロセスが起動して、オーダー番号をはじめとする、トランザクションの確認が企業 A に E メールが送信されます。

アクセス・クライアント開発手順の概要

アクセス・クライアントを開発するには、アクセス・クライアント・ソース・ファイルをコード化し、その他のタスクを完了します。アクセス・クライアントを作成するタスクには、次の一般的手順があります。

1. 開発環境をセットアップします。AccessInterfaces.idl ファイルをはじめとする、IBM WebSphere Business Integration Server Express and Express Plus ソフトウェアをインストールし、続いてユーティリティーを使用して、AccessInterfaces.idl ファイルから Java または C++ スタブを生成します。
2. コール・トリガー・フローによって、アクセスおよび実行用のコラボレーションのポートを構成します。この手順には、アクセス・クライアントの処理を可能にする、外部コラボレーション・ポートの構成が含まれます。
3. サーバー・アクセス API 呼び出しを実行するアクセス・クライアント (Web サーブレットなど) を実装およびデバッグします。IdlAccessInterfaces.* クラスをインポートし、次を実行する Java コードを実装します。
   - IBM WebSphere Business Integration Server Express and Express Plus へのアクセス・セッションを取得します。
   - 指定されたコラボレーション (データ・ハンドラー呼び出しなど) にトリガー・アクセス呼び出しを送信します。
   - コラボレーションを実行します。
4. 外部フォーマット (アクセス・クライアントから送信された) からのデータを IBM WebSphere ビジネス・オブジェクト・フォーマットに変換するために必要なデータ・ハンドラー・インスタンスを指すように、トップレベル・データ・ハンドラー・メタオブジェクト MO_Server_DataHandler を構成します。詳しくは、「*IBM WebSphere Business Integration Server Express and Express Plus データ・ハンドラー・ガイド*」を参照してください。

図 3 に、アクセス・クライアント開発プロセスの概要を示します。また、特定のトピックについての情報がどの章に記載されているかについても示します。アクセス・クライアント開発に対して何人かのグループをチームとして充てることができ

る場合には、アクセス・クライアント開発の主要なタスクを開発チームの別々のメンバーによって並行して行うことができます。

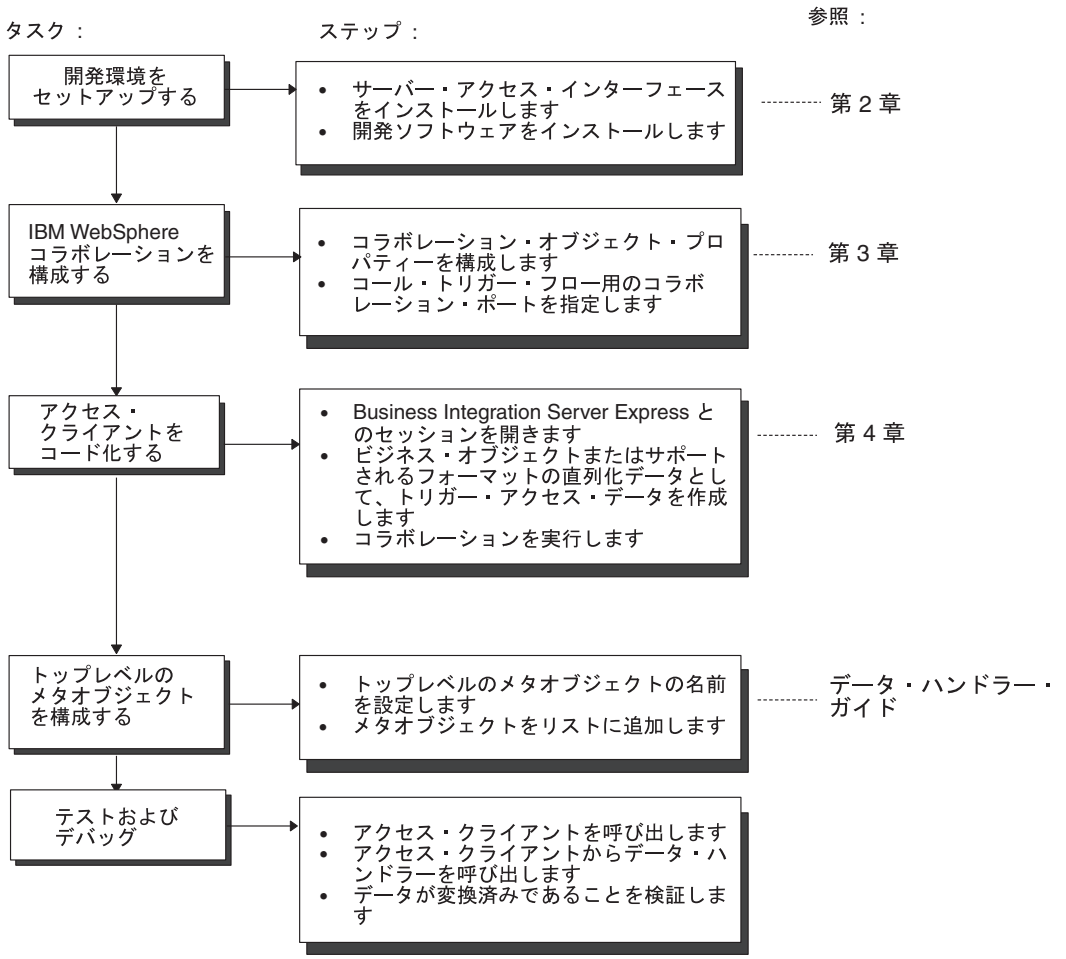

図 3. アクセス・クライアント開発タスクの概要

アクセス・クライアント開発用ツール

アクセス・クライアントは Java で書かれているため、あらゆるオペレーティング・システム上で開発することができます。IBM WebSphere のアクセス・クライアント開発用ツールを次の表にリストします。

IBM WebSphere ツール	説明
E-Business Development Kit (EDK)	次が組み込まれています。 • サンプル・データ・ハンドラー • `DataHandler` クラス拡張用のスタブ・ファイル

IBM WebSphere ツール	説明
サーバー・アクセス API	アクセス・クライアント内から IBM WebSphere Business Integration Server Express and Express Plus へアクセスするための Java クラスが含まれています。
データ・ハンドラー API	単一のクラス DataHandler が含まれています。拡張してカスタム・データ・ハンドラーを作成します。

E-Business Development Kit

WebSphere Business Integration Server Express and Express Plus E-Business Development Kit (EDK) は、次の表に示す、カスタム・ソフトウェアを開発するツールが開発者に提供されます。

カスタム・ソフトウェア	DevelopmentKits¥edk のサブディレクトリー
データ・ハンドラー	DataHandler
プロトコル・ハンドラー	ProtocolHandler
アクセス・クライアント	ServerAccessInterfaces
ユーティリティー。XMLBORGEN ユーティリティー (XML データ・ハンドラーによって使用される) を含む。	Utilities

上の表に示すように、アクセス・クライアント開発ツールは、*ProductDir* ディレクトリーの DevelopmentKits¥edk サブディレクトリーの下の ServerAccessInterfaces ディレクトリーにあります。

サンプル・アクセス・クライアント

アクセス・クライアントの開発を容易にするために、EDK では IBM WebSphere ディレクトリーに次のサンプル・アクセス・クライアントが組み込まれています。
DevelopmentKits¥edk¥ServerAccessInterfaces¥AccessSample

このディレクトリーには、次が格納されています。

- サンプル・アクセス・クライアント ATPServlet.java。HTML データをビジネス・オブジェクトに変換する機能を備えたサーブレットです。続いて、このビジネス・オブジェクトは、IBM WebSphere Business Integration Server Express and Express Plus でのコラボレーションに送信することができます。
- カスタム・データ・ハンドラー HtmlDataHandler.java。HTML データと IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトの間の変換を扱います。
- SampleRepos.jar ファイル。Access サンプルで使用されるコンポーネントのリポジトリー定義が含まれています。
- 次の表にリストするサブディレクトリーには、追加のサンプル・ファイルが格納されています。

名前	説明
collaborations	コール・トリガー・フロー用に構成されたコラボレーションが格納されています。
DLMs	必要なネイティブ・マップが格納されています。

注: このサンプルは、検証用として役に立ちますが、サーバー・アクセス API でサポートされているすべての機能性の実例を示すものではありません。

詳しくは、 31 ページの『第 5 章 HTML データ処理機能付きサーブレットのサンプル』を参照してください。

IBM WebSphere サーバー・アクセス API

IBM WebSphere サーバー・アクセス API には、次のインターフェースが用意されています。

サーバー・アクセス	説明	詳細情報の参照先
IAccessEngine	アクセス・クライアントを IBM WebSphere Business Integration Server Express and Express Plus にバインドするメソッドを提供します	53 ページの『第 6 章 IAccessEngine インターフェース』
IInterchangeAccessSession	IBM WebSphere Business Integration Server Express and Express Plus のアクセス・セッションへのアクセスを制御するメソッドを提供します	55 ページの『第 7 章 IInterchangeAccessSession インターフェース』
IBusinessObject	属性値の取得、設定、および比較などのビジネス・オブジェクト処理を実行するメソッドを提供します	65 ページの『第 8 章 IBusinessObject インターフェース』
IBusinessObjectArray	ビジネス・オブジェクト配列との対話およびその取り扱いをアクセス・クライアントに可能にするメソッドを提供します	93 ページの『第 9 章 IBusinessObjectArray インターフェース』

注: 上の表にリストされているインターフェースでのメソッドでは、 99 ページの『第 10 章 サーバー・アクセス例外』に説明されている例外がスローされます。

IBM WebSphere データ・ハンドラー API

IBM WebSphere データ・ハンドラー API には、DataHandler という名前の単一のクラスが用意されています。抽象基本クラス DataHandler は、カスタム・データ・ハンドラーの開発を容易にします。このクラスには、直列化入力データから取り出した値をビジネス・オブジェクトに移植するメソッド、および、ビジネス・オブジェクトをストリングまたはストリームに直列化するメソッドが格納されています。

また、このクラスには、カスタム・データ・ハンドラーが使用できるユーティリティー・メソッドも組み込まれています。この DataHandler クラスからカスタム・データ・ハンドラーを派生させることができます。DataHandler クラスに格納されているメソッドについて詳しくは、「データ・ハンドラー・ガイド」を参照してください。

注: アクセス・クライアントがその直列化データを、既存の IBM WebSphere Business Integration Server Express and Express Plus データ・ハンドラーがサポートするフォーマット以外のフォーマットでフォーマットする場合には、カスタム・データ・ハンドラーの開発のみを考慮する必要があります。これらのデータ・ハンドラーのリストについては、5 ページの『IBM WebSphere Business Integration Server Express and Express Plus のデータ・ハンドラーの役割』を参照してください。

第 2 章 アクセス・クライアント環境のセットアップ

本章では、アクセス・クライアントを開発および実行するために環境をセットアップする方法について説明します。本章のセクションは、以下のとおりです。

開発環境の設定

アクセス・クライアントの開発環境では、IBM WebSphere インストーラーによってインストールされるソフトウェアの一部である、サーバー・アクセス API スタブを使用できることが必要です。そのため、サーバー・アクセス API の呼び出しをアクセス・クライアントに組み込むために、次のソフトウェアを使用できなければなりません。

- IBM Java ORB 開発環境 (バージョン 4.5 以降。現行リリースについては、「*IBM WebSphere Business Integration Server Express and Express Plus システム・インストール・ガイド*」を参照)
- Java 開発環境および JDK 1.3.1
- IBM WebSphere ソフトウェアの現行リリース
- ブートされ、実行中の IBM WebSphere Business Integration Server Express and Express Plus
- コール・トリガー・フロー用に構成されたコラボレーション付きの IBM WebSphere リポジトリー (この構成を実行する方法について詳しくは、 17 ページの『第 3 章 コール・トリガー・フロー用のコラボレーションの構成』を参照)

上記にリストしたソフトウェアを使用できるようになったら、アクセス・クライアントに必要な開発環境をセットアップするために、次の手順を実行します。

- 『IBM WebSphere サーバー・アクセスのインストール』— 開発マシン上にサーバー・アクセスをインストールする。
- 14 ページの『アクセス・クライアントのコンパイル』— アクセス・クライアント用の実行可能ファイルを作成する。

IBM WebSphere サーバー・アクセスのインストール

アクセス・クライアントの開発を可能にするには、開発マシン上にサーバー・アクセスをインストールする必要があります。IBM WebSphere インストーラーは、IBM WebSphere サーバー・アクセスに関連付けられたファイルをインストールします。IBM WebSphere インストーラーは、 14 ページの表 1 に示すディレクトリーおよびファイルをインストールします。

表 1. IBM WebSphere サーバー・アクセスのインストール済みファイルの構造

ディレクトリー	説明
DevelopmentKits¥edk¥ ServerAccessInterfaces	アクセス・クライアント用の AccessInterfaces.idl ファイルが格納されています。
DevelopmentKits¥edk¥ ServerAccessInterfaces¥ AccessSample	サンプル・アクセス・クライアント用のソース・コードが格納されています。
repository¥edk	サーバー・アクセスがサポートするデータ・ハンドラーを定義する MO_Server_DataHandler メタオブジェクト用のファイルが格納されています。

IBM WebSphere インストーラーは、IBM WebSphere ソフトウェアのインストール時に、表 1 に示すファイルを自動的にインストールします。サーバー・アクセス API がインストールされていることを確認するには、IBM WebSphere インストーラーの「Select Components」画面上で「Server and Tools」コンポーネントを必ず選択してください。インストーラーがこのコンポーネントをインストールするとき、表 1 にリストしたディレクトリーおよびファイルが自動的にインストールされます。IBM WebSphere インストーラーについては、「*IBM WebSphere Business Integration Server Express and Express Plus システム・インストール・ガイド*」を参照してください。

注: また、IBM WebSphere インストーラーは、IBM WebSphere 提供のデータ・ハンドラーが必要とするファイルもインストールします。詳しくは、「*IBM WebSphere データ・ハンドラー・ガイド*」のインストールの章を参照してください。

アクセス・クライアントのコンパイル

アクセス・クライアントをコンパイルする準備ができたら、次のファイルへのパスがクラスパスにあることを確認します。

- IBM WebSphere crossworlds.jar ファイル
- IBM Java オブジェクト・リクエスト・ブローカー (ORB) の JAR ファイル

javac コンパイラー、または任意の統合開発環境 (IDE) を使用できます。

ランタイム環境の設定

実行時に、アクセス・クライアントは、IBM WebSphere Business Integration Server Express and Express Plus が格納されているマシン上にある必要はありません。また、開発環境と同じマシン上にある必要もありません。ただし、アクセス・クライアントが実行時に必要な IBM WebSphere Business Integration Server Express and Express Plus インスタンスを検索できるようにするには、IBM WebSphere Business Integration Server Express and Express Plus インスタンスなどの異なる CORBA オブジェクトを追跡するオブジェクト・リクエスト・ブローカーのサーバーを探し出して、その情報を ORB クライアント (アクセス・クライアントなど) に伝達する必要があります。ORB サーバーの場所を取得するために、アクセス・クライアントは IBM WebSphere Business Integration Server Express and Express Plus インスタンスが生成する相互運用オブジェクト参照ファイルを使用できます。 IBM WebSphere

Business Integration Server Express and Express Plus は、始動またはリブートするときに、.ior 拡張子を持つ相互運用オブジェクト参照ファイルを生成します。アクセス・クライアントはこのファイルを使用して ORB サーバーの場所を検索し、次に、IBM WebSphere Business Integration Server Express and Express Plus インスタンスと通信することができます。

つまり、アクセス・クライアントが IBM WebSphere Business Integration Server Express and Express Plus インスタンスを検索できるようにするために、以下の手順を実行する必要があります。

1. IBM WebSphere Business Integration Server Express and Express Plus が、永続的な .ior ファイルを生成することを要求します。
2. アクセス・クライアントがあるマシンが IBM WebSphere Business Integration Server Express and Express Plus インスタンスの .ior ファイルを検索できることを確認します。

この手順の各ステップは以下のセクションでより詳細に解説されています。

永続的な .ior ファイルの生成

IBM WebSphere Business Integration Server Express and Express Plus は、ブート時に新しい .ior ファイルを生成します。ただし、ORB サーバーのポート番号を動的に割り当てます。サーバーがブートするたびにポート番号が変化すると、アクセス・クライアントは .ior ファイルによって ORB サーバーを検索できなくなります。そこで、アクセス・クライアントにとっては、IBM WebSphere Business Integration Server Express and Express Plus が **永続的な .ior** ファイルを生成することが必要になります。

永続的な .ior ファイルを IBM WebSphere Business Integration Server Express and Express Plus に生成させるには、IBM WebSphere Business Integration Server Express and Express Plus 構成ファイル (InterchangeSystem.cfg) を XML エディターで編集して、CORBA サブセクションを追加します (まだ存在していない場合)。図 4 に、空の CORBA サブセクション (構成パラメーターが定義されていない サブセクション) を定義する XML コードを示します。

```
<tns:property>
   <tns:name>CORBA</tns:name>
   <tns:isEncrypted>false</tns:isEncrypted>
   <tns:updateMethod>system restart</tns:updateMethod>
   <tns:location>
      <tns:reposController>false</tns:reposController>
      <tns:reposAgent>false</tns:reposAgent>
      <tns:localConfig>true</tns:localConfig>
   </tns:location>
   XML definitions of CORBA properties go here
   </tns:property>
```

図 4. CORBA サブセクションの XML 定義

CORBA サブセクションでは、OAport 構成パラメーターに対して静的ポート番号を指定します。この構成パラメーターの構文は以下のとおりです。

```
OAport=portNumber
```

例えば、静的ポート番号が 15000 の場合、CORBA サブセクションの OAport パラメーターとして値 15000 を割り当てます。以下の XML フラグメントは、CORBA サブセクションの <tns:property> タグ内にあります。この場所は 15 ページの図 4 に「*XML definitions of CORBA properties go here*」という文字列で示されています。

```
<tns:property>
  <tns:name>OAport</tns:name>
  <tns:value xml:space="preserve">15000</tns:value>
  <tns:isEncrypted>false</tns:isEncrypted>
  <tns:updateMethod>system restart</tns:updateMethod>
  <tns:location>
     <tns:reposController>false</tns:reposController>
     <tns:reposAgent>false</tns:reposAgent>
     <tns:localConfig>true</tns:localConfig>
  </tns:location>
</tns:property>
```

要確認: IBM WebSphere Business Integration Server Express and Express Plus の構成ファイルは XML ファイルです。CORBA サブセクション、およびその構成パラメーターを追加するには、XML エディターを使用するか、または正しく適切な XML タグの形式を整える必要があります。

構成ファイルの CORBA サブセクションについての詳細は、「*IBM WebSphere Business Integration Server Express and Express Plus システム・インストール・ガイド*」を参照してください。

.ior ファイルの検索

アクセス・クライアントが実行時に ORB サーバーを検索できるようにするには、アクセス・クライアントが IBM WebSphere Business Integration Server Express and Express Plus インスタンスの .ior ファイルの場所を見付ける必要があります。このファイルを探し出すことは、アクセス・クライアントおよび IBM WebSphere Business Integration Server Express および Express Plus が同じマシン上にある場合には問題ありません。ただし、これらの 2 つのコンポーネントが同じマシン上に存在しない場合 は、以下のいずれか の操作を実行して、アクセス・クライアント・マシンが、.ior ファイルにアクセスできるようにします。

- アクセス・クライアントがあるマシンに IBM WebSphere Business Integration Server Express and Express Plus が生成した .ior ファイルをコピーします。
- IBM WebSphere Business Integration Server Express and Express Plus が常駐するマシン上に共用ディレクトリーを作成し、アクセス・クライアントのマシンがそのディレクトリーを指すようにします。

アクセス要求の順序付けのためのイベントの切り替え

アクセス・フレームワークを使用して同期要求がコラボレーションに送信されるときは、要求の順序付けは重要ではありません。特に、パフォーマンスのチューニングの際には問題にはなりません。イベントの順序付けは、デフォルトで同期アクセス要求についてコラボレーション・レベルでオンになります。同期アクセス要求についてイベントの順序付けをオフにするには、InterchangeSystem.cfg ファイルを編集して以下の行を追加します。

```
[ACCESS]
EVENT_SEQUENCING=FALSE
```

第 3 章 コール・トリガー・フロー用のコラボレーションの構成

本章では、コール・トリガー・フロー用のコラボレーションを構成する方法について説明します。コラボレーションの構成は、コラボレーションをアクセス・クライアントから実行する*前に* 行う必要があります。本章のトピックは、以下のとおりです。

要確認: コール・トリガー・フロー用のコラボレーションを構成するには、すべての IBM WebSphere ソフトウェアをインストールし、IBM WebSphere Business Integration Server Express and Express Plus を稼働させる必要があります。

コール・トリガー・フロー・オプションを実装する System Manager の使用

System Manager を使用して、コール・トリガー・フロー用のコラボレーションを構成します。コール・トリガー・フロー・オプションをコラボレーション用に実装するには、まず、リポジトリー内の既存のコラボレーション・テンプレートの 1 つから新規のコラボレーション・オブジェクトを作成する必要があります。新規コラボレーション・オブジェクトを作成するには、次の手順を実行します。

1. System Manager で、「コラボレーション・オブジェクト」フォルダーを右マウス・ボタンでクリックし、「新規コラボレーション・オブジェクト」を選択します。

   「新規コラボレーション・テンプレートの作成」ダイアログ・ボックスが開き、インストール済みテンプレートが「テンプレート名」列にリストされます。
2. コール・トリガー・フローをサポートするコラボレーション・オブジェクトを構成する元のコラボレーション・テンプレートの名前をクリックします。
3. コラボレーション・オブジェクトの名前を「コラボレーション」オブジェクト名フィールドに入力します。「次へ」をクリックします。

   「コラボレーション・ポートのバインド」ダイアログ・ボックスが開きます。

コラボレーション・オブジェクトが存在すると、 18 ページの表 2 にリストされている手順を実行して、それをコール・トリガー・フロー用に構成することができます。

表 2. コール・トリガー・フロー用のコラボレーション・ポートの構成

構成手順	詳細情報の参照先
コール・トリガー・フロー用のコラボレーション・ポートを指定して、そのポートをコラボレーションにバインドします。	『コール・トリガー・フロー用のコラボレーション・ポートの指定』
ビジネス・オブジェクト・フローをコラボレーションに関連付けるマップを選択します。	20 ページの『ビジネス・オブジェクトとマップの関連付け』
新規コラボレーション・オブジェクトのプロパティーを設定します。	21 ページの『コラボレーション・オブジェクト・プロパティーの構成』

注: コラボレーションでは、複数のコール・トリガー・フロー・ポートを構成することができます。

表 2 に、コール・トリガー・フロー用のコラボレーションの構成方法について要約します。コラボレーションの構成と System Manager について詳しくは、「システム・インプリメンテーション・ガイド」および「コラボレーション開発ガイド」を参照してください。

コール・トリガー・フロー用のコラボレーション・ポートの指定

コール・トリガー・フロー用に構成するそれぞれのコラボレーションごとに、コラボレーション・オブジェクト上でポートの構成を実行する必要があります。

コール・トリガー・フロー用のコラボレーション・ポートを構成するには、次の手順を実行します。

1. System Manager の「Server Monitor」領域にコラボレーション・オブジェクト用の「Collaboration Object View」ウィンドウが表示されていることを確認します。

   上記のウィンドウが現在表示されていない場合には、System Manager オブジェクト・ブラウザーの「コラボレーション・オブジェクト」フォルダーに移動して、構成するコラボレーション・オブジェクトをダブルクリックします。
2. コール・トリガー・フロー用に構成するポートを右マウス・ボタンでクリックして、「ポートをバインド」を選択します。

   「ポートを構成」ダイアログ・ボックスが開きます (図 5 を参照)。このダイアログでのポートのタイプのデフォルト設定は「内部」です。

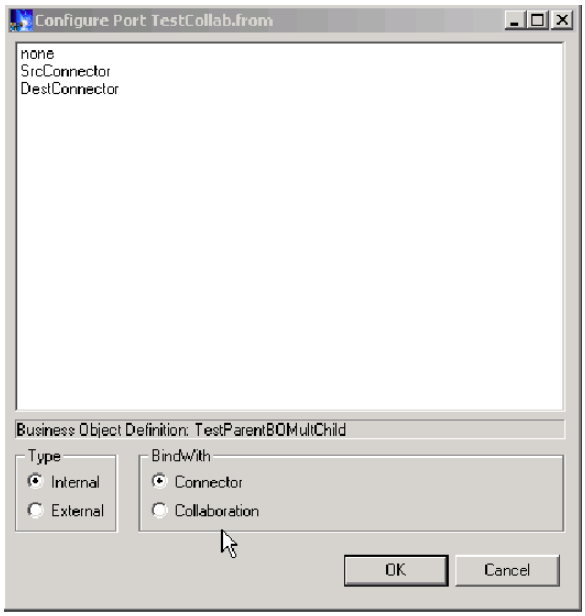

図 5. 「ポートを構成」(内部) ダイアログ・ボックス

3. 「タイプ」領域で「外部」をクリックします。

   図 6 に示すように、「ポートを構成」(外部) ダイアログ・ボックスが表示されます。ダイアログ・ボックスには「構成」領域に、構成するために選択したポートのタイプである「入力」または「出力」が表示されます。

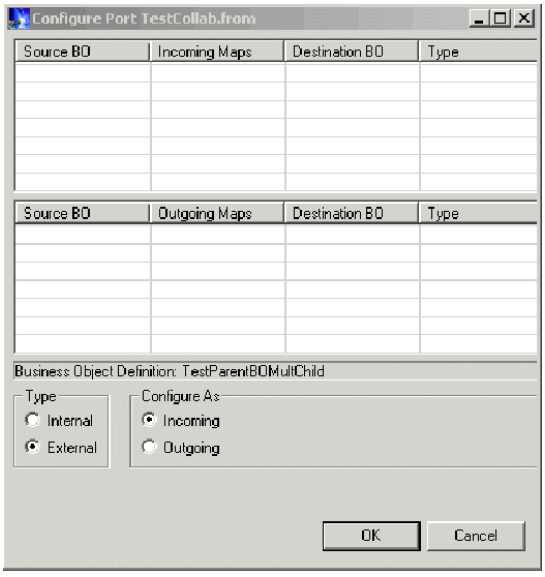

図 6. 「ポートを構成」(外部) ダイアログ・ボックス

ビジネス・オブジェクトとマップの関連付け

コール・トリガー・フロー用に構成したコラボレーションに対して、マップとビジネス・オブジェクトを関連付ける必要があります。この関連付けは、次の 3 つの方法のいずれかで行います。

- 「Object View」領域 (図 7) の左方にある「Designer Directory」内の「ビジネス・オブジェクト」フォルダーからビジネス・オブジェクトをクリックして、「ポートを構成」ダイアログ・ボックスの「宛先」ポケットにドラッグします。ビジネス・オブジェクトをこのポケットにドロップすると、ソースから宛先まで行くすべてのマップがポップアップ・ウィンドウに表示されます。適切なマップを選択します。

または

- コラボレーションに関連付けるマップがわかっている場合には、マップをクリックして、「ポートを構成」ダイアログ・ボックスの「Incoming Maps」または「Outgoing Maps」ポケットにドラッグすることができます。

または

- 外部エンティティーによって実行されたときに、このコラボレーションにマップを関連付けない場合には、入力または出力ビジネス・オブジェクト列を空の状態のままにします。

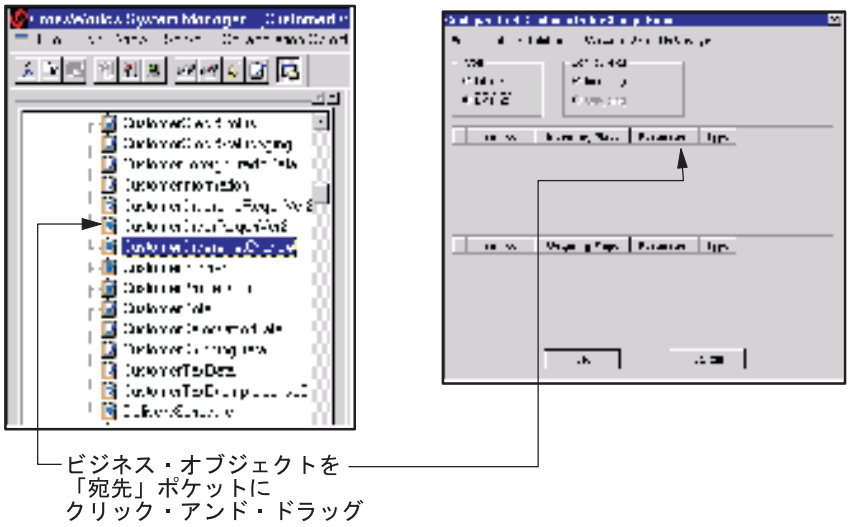

図 7. コール・トリガー・フロー・コラボレーションへのビジネス・オブジェクトのマッピング

注: コラボレーション、ビジネス・オブジェクト、およびマップについて詳しくは、「システム・インプリメンテーション・ガイド」および「コラボレーション開発ガイド」を参照してください。

フローの向き: コラボレーションへ

ビジネス・オブジェクトのドラッグ — ビジネス・オブジェクトはコラボレーションのための宛先タイプとして使用されます。ポップアップ・ウィンドウから適切なマップを選択します。宛先は常時、「ポートを構成」ウィンドウに表示されるビジネス・オブジェクト定義です。

マップのドラッグ — マップは、コラボレーションに対して呼び出しが作成されるときに使用されます。宛先ビジネス・オブジェクトをサポートするマップを選択します。

フローの向き: コラボレーションから

ビジネス・オブジェクトのドラッグ — ビジネス・オブジェクト・オプションは、コラボレーションが結果を戻すときに使用されます。

マップのドラッグ — マップは、コラボレーションが要求プロセスにデータまたは属性を戻すときに使用されます。

ビジネス・オブジェクトのドラッグ

ビジネス・オブジェクトをドラッグして、ビジネス・オブジェクト・タイプおよびマップをコラボレーションにバインドするには、次の手順を実行します。

1. 「Designer Directory」の「ビジネス・オブジェクト」フォルダーをダブルクリックします。

   これで、ビジネス・オブジェクトのリストが表示されます。
2. ビジネス・オブジェクトをクリックおよびドラッグして、「ポートを構成」ダイアログ・ボックスの「宛先」ポケットにドロップします。構成しているポートに応じて、「Incoming Maps」または「Outgoing Maps」を選択します。

   これで、「ポートを構成」ダイアログ・ボックスにマップおよびビジネス・オブジェクトが表示されます。
3. ビジネス・オブジェクトと一緒に表示されているマップからマップを 1 つ選択します (マップが 1 つのみ表示されている場合もあります)。「OK」をクリックします。

コラボレーション・オブジェクト・プロパティーの構成

コール・トリガー・フロー用に構成するそれぞれのコラボレーションごとに、その並行イベントの数をゼロ (0) に設定します。コール・トリガー・フロー用のコラボレーションのプロパティーを構成するには、次の手順を実行します。

1. System Manager の「Server Monitor」領域にコラボレーション・オブジェクト用の「Collaboration Object View」ウィンドウが表示されていることを確認します。

   上記のウィンドウが現在表示されていない場合には、System Manager オブジェクト・ブラウザーの「コラボレーション・オブジェクト」フォルダーに移動して、構成するコラボレーション・オブジェクトをダブルクリックします。

2. コラボレーションのアイコン (中央のアイコン) を右マウス・ボタンでクリックして、「プロパティー」を選択します。

   「Collaboration Properties」ダイアログ・ボックスが開きます。
3. 必要に応じて、コラボレーション・オブジェクトのプロパティーを構成します。

   要確認: プロパティーの「**並行イベントの最大数**」が `0` の値に設定されていることを確認してください。コール・トリガー・フローは、デフォルトでは、マルチスレッド化されています。そのため、このプロパティーを `0` に設定すると、マルチスレッド化機能を実装するために IBM WebSphere Business Integration Server Express and Express Plus が追加的に作成するスレッドが作成されなくなります。このプロパティーの詳細については、「*IBM WebSphere Business Integration Server Express and Express Plus システム管理ガイド*」を参照してください。
4. 「OK」をクリックすると、「Collaboration Properties」ダイアログ・ボックスがクローズします。

第 4 章 アクセス・クライアントの実装

本章では、アクセス・クライアントを実装する方法について概説します。アクセス・クライアントは、サーバー・アクセス API を介して IBM WebSphere Business Integration Server Express and Express Plus とのコラボレーションの実行を要求することができます。本章のトピックは、以下のとおりです。

アクセス・セッションの作成

アクセス・クライアントがアクセス要求を発行するには、最初に、IBM WebSphere Business Integration Server Express and Express Plus との**アクセス・セッション**を確立する必要があります。アクセス・クライアントが IBM WebSphere Business Integration Server Express and Express Plus に接続することを可能にするため、`IAccessEngine` インターフェースには IgetInterchangeAccessSession() メソッドが用意されています。このメソッドでは、IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスへのアクセスをアクセス・クライアントに提供する、アクセス・セッションが作成されます。引き数として、`IgetInterchangeAccessSession()` メソッドに有効な IBM WebSphere Business Integration Server Express and Express Plus ユーザー名およびパスワードを入力しなければなりません。

要確認: IBM WebSphere Business Integration Server Express and Express Plus のユーザー名は `admin` である必要があります。

`IAccessEngine` インターフェースについて詳しくは、53 ページの『第 6 章 IAccessEngine インターフェース』を参照してください。

アクセス要求の発行

アクセス・クライアントがアクセス・セッションを作成すると、IBM WebSphere Business Integration Server Express and Express Plus にアクセス要求を送信することができます。アクセス要求とは、IBM WebSphere Business Integration Server Express and Express Plus 内でコール・トリガー・フローを開始する要求です。アクセス・クライアントがトリガー・アクセス呼び出しを送信するには、最初にコラボレーションに送信するトリガー・アクセス・データを生成する必要があります。サーバー・アクセスでは、アクセス・クライアントがトリガー・アクセス・データのフォーマットに基づいてアクセス要求を発行するために、次の方法が用意されています。

- 『ビジネス・オブジェクトの送信』
- 25 ページの『直列化データの送信』

ビジネス・オブジェクトの送信

アクセス・クライアントは、IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクト内にカプセル化されたトリガー・アクセス・データを送信することができます。 `IInterchangeAccessSession` インターフェースには、ビジネス・オブジェクトを作成するメソッドおよびコラボレーションを実行する方法が用意されています。このインターフェースについて詳しくは、 55 ページの『第 7 章 IInterchangeAccessSession インターフェース』を参照してください。

ビジネス・オブジェクトをトリガー・アクセス・データとして送信するには、次の手順があります。

- 『ビジネス・オブジェクトの作成』
- 25 ページの『ビジネス・オブジェクトでの操作』
- 25 ページの『コラボレーションの実行の要求』

ビジネス・オブジェクトの作成

表 3 に、ビジネス・オブジェクトを作成するアクセス・クライアントに対してサーバー・アクセス API が `IInterchangeAccessSession` インターフェースで提供するメソッドを示します。

表 3. ビジネス・オブジェクトを作成する IInterchangeAccessSession メソッド

ビジネス・オブジェクトの作成	IInterchangeAccessSession メソッド
ビジネス・オブジェクトを作成します。	IcreateBusinessObject()
オブジェクト属性の操作を指定する動詞でビジネス・オブジェクトを作成します。	IcreateBusinessObjectWithVerb()
1 つ以上の属性が含まれるビジネス・オブジェクト配列を作成します (各属性にはその型として指定されたビジネス・オブジェクトがあります)。	IcreateBusinessObjectArray()
指定された MIME タイプ内にフォーマットされたデータからビジネス・オブジェクトを作成します。	IcreateBusinessObjectFrom()

ビジネス・オブジェクトでの操作

アクセス・クライアントは、ビジネス・オブジェクトを作成すると、表 4 に示したインターフェースを使用して、トリガー・アクセス・データをこのオブジェクトに格納するために必要な操作を実行することができます。

表 4. ビジネス・オブジェクトにアクセスするインターフェース

ビジネス・オブジェクトのタイプ	サーバー・アクセス API	詳細情報の参照先
ビジネス・オブジェクト (単一カーディナリティー)	IBusinessObject。このインターフェースを使用して、アクセス・クライアントは、属性値の取得、設定、および比較などのビジネス・オブジェクト処理を実行することができます。	65 ページの『第 8 章 IBusinessObject インターフェース』
ビジネス・オブジェクト配列	IBusinessObjectArray。このインターフェースを使用して、アクセス・クライアントは、ビジネス・オブジェクト配列と対話したり、ビジネス・オブジェクト配列を取り扱ったりすることができます。メソッドには、ビジネス・オブジェクト配列要素の設定または取得、配列のコピー、配列へのビジネス・オブジェクトの追加、またはビジネス・オブジェクト配列内の要素の数の取り出しが含まれています。	93 ページの『第 9 章 IBusinessObjectArray インターフェース』

コラボレーションの実行の要求

IInterchangeAccessSession インターフェースでは、トリガー・アクセス呼び出し内のトリガー・アクセス・データとしてビジネス・オブジェクトを送信するための IexecuteCollaboration() メソッドが提供されます。このメソッドは、IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスに対して、指定されたコラボレーションにビジネス・オブジェクトをトリガー・アクセス・データとして送信するように指示します。

注: コラボレーション、ポート、およびビジネス・オブジェクトは、直接呼び出しアクセスおよび操作に対して構成およびマップする必要があります。

直列化データの送信

アクセス・クライアントは、そのトリガー・アクセス・データを直列化データ・ハンドラーとして、指定された MIME タイプで送信することができます。 IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスは、直列化データから IBM WebSphere ビジネス・オブジェクトへの必要なデータ変換を行います。直列化データの送信には、サーバー・アクセス API の単一のメソッドの呼び出し IexecuteCollaborationExtFmt() が含まれます。このメソッドでは、アクセス・クライアントを対象とする、次のタスクが提供されます。

- データ・ハンドラーを指定して (直列化データの MIME タイプに基づいて)、直列化データをビジネス・オブジェクトに変換する。

- コラボレーションを起動するビジネス・オブジェクトを作成する。
- 動詞を指定値へ設定する。
- コラボレーションを実行する。

ロケールとエンコード

アクセス・セッションでは、デフォルトで IBM WebSphere Business Integration Server Express and Express Plus の Locale 値を使用します。ただし、Locale 値を変更して、アクセス・セッションを介して作成または実行する、ビジネス・オブジェクトまたはコラボレーションの Locale 値に一致させることができます。

サーバー・アクセスへ送信する入力データは、Unicode エンコードである必要があります。

ロケールの概説については、『付録. 国際化対応に対する考慮事項』を参照してください。

Locale 値の設定方法の詳細については、55 ページの『第 7 章 IInterchangeAccessSession インターフェース』の setLocale(String) を参照してください。

アクセス応答の取得

コラボレーションは、表 5 に示したメソッドの 1 つの戻り値を介して、アクセス応答をアクセス・クライアントに戻します。このアクセス要求のフォーマットは、アクセス・クライアントがアクセス要求の送信に使用したメソッドによって異なります。

表 5. アクセス応答を取得するためのメソッド

アクセス要求	サーバー・アクセスのメソッド	アクセス応答のフォーマット
トリガー・アクセス・データをビジネス・オブジェクトとして送信	IexecuteCollaboration()	ビジネス・オブジェクト
トリガー・アクセス・データを指定する MIME タイプの直列化データとして送信	IexecuteCollaborationExtFmt()	直列化データ (アクセス要求として同一の MIME フォーマットで)

注: アクセス応答が IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトの形式である場合には、このビジネス・オブジェクト上で作動する、25 ページの表 4 にリストされたインターフェースのメソッドを使用できます。

アクセス・セッションのクローズ

アクセス・クライアントは、そのアクセス要求を完了したとき、表 6 に示した手順を実行する必要があります。

表 6. アクセス・セッションのクローズ

タスク	サーバー・アクセスのメソッド
ビジネス・オブジェクトおよびビジネス・オブジェクト配列に対して IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスが使用しているリソースを解放	`IInterchangeAccessSession` メソッドは IreleaseBusinessObject()、IreleaseBusinessObjectArray()
アクセス・セッションをクローズ	`IAccessEngine` メソッドは IcloseSession()

注: `IcloseSession()` の呼び出しで、アクセス・セッションが使用しているリソースは解放されます。

コール・トリガー・フローの実装の例

図 8 に、コール・トリガー・フローの詳細を示します。ここでは、クライアント・ブラウザーであるアクセス・クライアントによって開始されています。

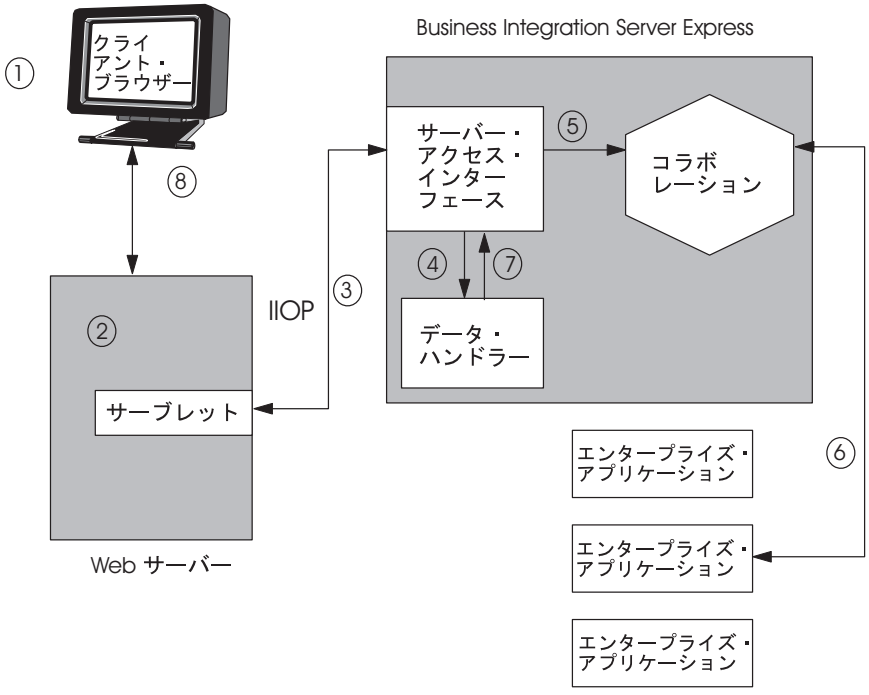

図 8. クライアント・ブラウザーによるコール・トリガー・フロー開始の例

図 8 は次のことを示しています。

1. クライアント・ブラウザーは、指定されたプロトコルおよびフォーマット (例えば、HTTP プロトコルと XML データ・フォーマット) で要求を発行します。
2. エンタープライズ Web サーバーは、要求を処理するサーブレットをロードします。このサーブレットはアクセス・クライアントです。CORBA 準拠の IBM

WebSphere Business Integration Server Express and Express Plus の名前を (CORBA レジストリーから) 検索するようにプログラムされています。

3. アクセス・クライアントは、サーバー・アクセス API の IAccessEngine インターフェースの IgetInterchangeAccessSession() メソッドとのアクセス・セッションを作成して、IBM WebSphere Business Integration Server Express and Express Plus に IIOP 接続を介してログインします。

   注: コラボレーションを実行するのに、IBM WebSphere Business Integration Server Express and Express Plus はそれ自体のスレッドを*作成しない* で、CORBA スレッドを使用します。スレッドを使用するコラボレーション方法の詳細については、「*コラボレーション開発ガイド*」を参照してください。

4. アクセス・クライアントは IInterchangeAccessSession インターフェース内の IcreateBusinessObjectFrom() メソッドを使用して、XML データを汎用 IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトに変換します。このメソッド呼び出しに応答して、IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスは、XML データ・ハンドラーを起動して、データ変換を行い、続いてビジネス・オブジェクトをアクセス・クライアントに戻します。

5. アクセス・クライアントは、IInterchangeAccessSession インターフェース内の IexecuteCollaboration() メソッドを使用して、トリガー・アクセス呼び出しを送信します。このトリガー・アクセス呼び出しには、ビジネス・オブジェクトがトリガー・アクセス・データとして含まれています。このプロセスでは、ビジネス・オブジェクトを操作する、コラボレーションの実行が要求されます。

   注: また、サーバー・アクセス API は IexecuteCollaborationExtFmt() メソッドも提供します。このメソッドでは、手順 4 と手順 5 が単一のメソッド呼び出しに結合されます。

6. コラボレーションは、コネクターをトラバースして、要求、分類を配置し、データを取り出し、必要に応じてエンタープライズ・アプリケーションを操作します。コラボレーションは、要求データ、または要求アクションの結果をビジネス・オブジェクト・フォーマットでアクセス・クライアントに戻します。

7. アクセス・クライアントが IexecuteCollaborationExtFmt() メソッドを使用してアクセス要求を発行した場合、アクセス・クライアントは手順 6 でのアクションを明示的に行う必要はありません。IexecuteCollaborationExtFmt() メソッドは、ビジネス・オブジェクトを元のフォーマットに自動的に変換して (この場合は、XML フォーマット)、この直列化データをアクセス・クライアントに戻します。

8. 結果はクライアント・ブラウザーに配送されます。

図 8 に示すように、呼び出しを処理する Web サーバーは、呼び出しを処理するサーブレットをロードします。これで IBM WebSphere Business Integration Server Express and Express Plus に接続されます。

第 2 部 例

第 5 章 HTML データ処理機能付きサーブレットのサンプル

本章では、サーバー・アクセス API を使用する典型的な e-commerce シナリオおよびサンプル・コードを示します。本章で扱うトピックは、以下のとおりです。

シナリオ

e-commerce 環境で発生する共通の問題は、品目の可用性と希望日までの確実な配送の見通しの問題です。このクラスの問題は一般に、ATP または納期回答 (販売可能在庫数量の確認) と呼ばれます。

サプライ・チェーン最適化システムまたはエンタープライズ・リソース・プランニング (ERP) システムを使用する企業は一般に、製品が希望配送日までに納入できるかどうかを判別するシステムに照会します。一部の企業 (特に複数のベンダーとオンライン取引関係がある企業) では、製品の注文を確定する前の製品納期情報が重要視されます。

ATP (納期回答) 機能とは、企業の ERP またはサプライ・チェーン最適化システムに効率的にアクセスできる機能です。次の例で、サーバー・アクセス API は、次のタスクを実行するために使用されます。

- **データ変換** - 着信引用オブジェクトを HTML フォーマットから IBM WebSphere ビジネス・オブジェクトに変換します。
- **コラボレーション実行** - 着信引用オブジェクトで検出されるそれぞれの品目ごとに ATP データを取得するコラボレーションを起動します。
- **結果の取得** - 結果を HTML 表として戻します。

図 9 に、単一の納期回答コラボレーションを示します。

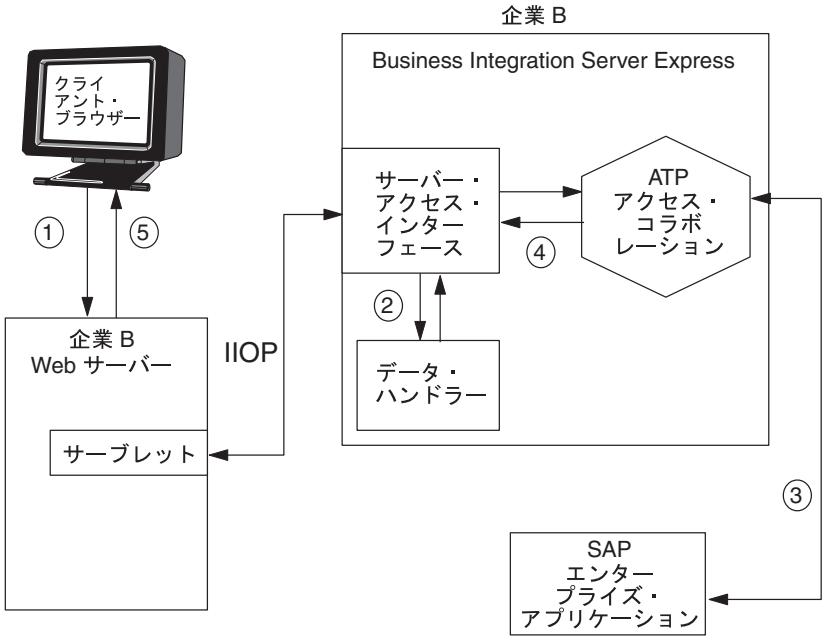

図 9. 納期回答 e-commerce シナリオ

1. ブラウザー・クライアントが、IncomingQuote オブジェクトに相当するデータが格納されている HTML フォームを送信します。IncomingQuote オブジェクトは、サード・パーティー・アプリケーションによって提供される HTML フォーマット済みデータです。
2. サーブレット (以下の実例コードを参照) は、サーバー・アクセス API を使用して HTML を汎用 SalesQuote オブジェクトに変換し、続いてそれをコラボレーションに送信します。
3. 続いて、ATP アクセス・コラボレーションは、SAP コネクターから納期回答日付を取得します。
4. コラボレーションは、この情報をサーブレットに戻します。
5. サーブレットは、それぞれの希望品目ごとに納期回答日付が格納された HTML 表を作成し、この表をクライアント・ブラウザー上に表示します。

Web サーバー上でのサンプルの実行

サンプルのサーバー・アクセスのコード・サンプルをロードおよび実行することができます。このセクションではその方法を示します。

1. Server Access Development ソフトウェアをインストールし、`DevelopmentKits¥edk¥ServerAccessInterfaces¥AccessSample` に移動して次を見つけます。
   - 2 つの Java コード・サンプル

     `HtmlDataHandler.java`

     `ATPServlet.java`
   - HTML 販売見積価格照会フォーム: `Example2.html`
   - サンプル・リポジトリー: `SampleRepos.jar`

- collaborations サブディレクトリーにはコラボレーションおよびクラスが格納されています。
- DLMs ディレクトリーには、ネイティブ・マップ・クラスが格納されています。

2. repos_copy ユーティリティーで SampleRepos.in をロードします。リポジトリーへのファイルのロードに関するヘルプ情報については、「*システム管理ガイド*」を参照してください。
3. サーブレット・ファイル ATPServlet.java をコンパイルします。
4. コンパイルしたサーブレットを Web サーバーに展開します。構成に合わせて、初期化パラメーター値を適切に設定します。サーブレットの展開および初期化の詳細については、Web サーバーの文書を参照してください。
5. Solaris オペレーティング・システムまたは HP-UX オペレーティング・システムをご使用の場合は、<ProductDirectory>jre¥lib¥ext (IBM Java ORB クラス・ファイル) にある ibmorb.jar をクライアントおよび Web サーバーのクラスパスに追加してください。必要に応じて、Web サーバーを再始動します。詳細については、Web サーバーの文書を参照してください。
6. Example2.html を Web サーバーで使用可能にします。
7. AccessSample¥collaborations ディレクトリーを *ProductDir*¥collaborations にコピーします。
8. AccessSample¥DLMs ディレクトリーを *ProductDir*¥DLMs にコピーします。
9. HtmlDataHandler.java をコンパイルします。
10. .jar ファイルを作成し、それを HtmlDataHandler.jar として保管して、出力ディレクトリー構造を保持します。
11. HtmlDataHandler.jar ファイルを *ProductDir*¥lib にコピーします。
12. start_server バッチ・ファイルを変更し、クラス・パスに *ProductDir*¥lib¥HtmlDataHandler.jar を追加します。
13. IBM WebSphere Business Integration Server Express and Express Plus を再始動します。
14. 相互運用オブジェクト参照 (.ior) ファイルを Web サーバーで使用できるようにします。

    詳しくは、14 ページの『ランタイム環境の設定』を参照してください。
15. ブラウザーを起動し、example2.html ページをオープンします (図 10 を参照)。
16. Test Connector を始動して、「SampleSapConnector」プロファイルをオープンおよび追加します。「接続」ボタンを押すと、コネクターが表示されます。
17. フィールドの 1 つ以上の行にデータを入力し (サンプル HTML ページの詳細については、34 ページの『サンプル HTML データ・ハンドラー』を参照)、サンプル検索操作を実行します。

次のセクションでは、上記のサンプルで使用した、データ・ハンドラーおよびサーブレットについて説明します。

- 34 ページの『サンプル HTML データ・ハンドラー』
- 42 ページの『サンプル Java コード — ATP サーブレット』

サンプル HTML データ・ハンドラー

このサンプルで、HTML データ・ハンドラーは着信 HTML 照会ストリングを IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトに変換します。IBM WebSphere データ・ハンドラー機能の詳細については、「*IBM WebSphere Business Integration Server Express and Express Plus データ・ハンドラー・ガイド*」を参照してください。次に、データ・ハンドラー・コンポーネントのうち、特に重要な機能を示します。

- **datahandler 基本クラス** - このサンプル HTML データ・ハンドラーでは、IBM WebSphere Business Integration Server Express and Express Plus 提供の `DataHandler` 基本クラスが拡張され、MIME タイプの「`text/html`」に対するアクセス要求が発生したとき、実行時に自動的にロードされます。
- **メタデータベース構成** - データ・ハンドラーを検出する場所およびデータ・ハンドラーを呼び出す方法について、メタデータがシステムに指示します。したがって、単一の IBM WebSphere Business Integration Server Express and Express Plus で複数のデータ・ハンドラーを並行して実行することができます。
- **汎用変換** - 本来、HTML データ・ハンドラーは汎用であるため、不特定型の HTML 照会ストリングを変換するのに、そのまま (変更しないで) 再使用することができます。

図 10 に、クライアント・ブラウザー上に表示された HTML ページを示します。HTML データ・ハンドラーは、このページのテキスト・ボックスに関連付けられたプロパティーに依存します。

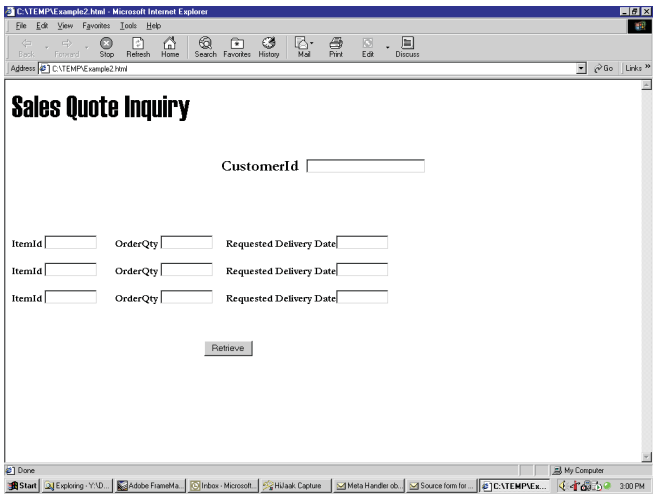

図 10. HTML 販売見積価格照会ページ

図 10 で、それぞれのテキスト・ボックスには、HTML プロパティーが関連付けられています。HTML テキスト・ボックス・プロパティーには、IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクト文法が格納されています。この文法に従って、HTML データ・ハンドラーは、プロパティーに関連付けられたデータをビジネス・オブジェクトに変換することができます。

例えば、最初の品目に関連付けられたプロパティーは、次のとおりです。

- **ItemId** - OrderItems[0].ItemID

- **OrderQty** - OrderItems[0].orderQty
- **Requested delivery date** - OrderItems[0].deliveryDate

図 11 に示すように、データ・ハンドラーは HTML ページ上のデータを、子 (orderQty、deliveryDate など) ビジネス・オブジェクト付きの階層型 SalesQuote ビジネス・オブジェクトに変換します。

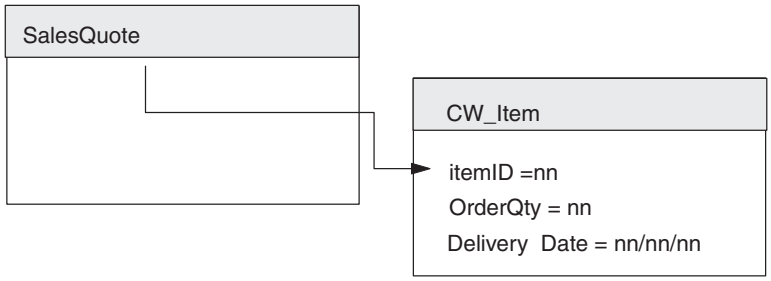

図 11. 階層型親子ビジネス・オブジェクト

データ・ハンドラー・メタオブジェクト

IBM WebSphere Business Integration Server Express and Express Plus ソフトウェアでは、2 つのトップレベル・データ・ハンドラー・メタオブジェクトが提供されます。1 つはサーバー用で、1 つはコネクター用です。さらに、それぞれのデータ・ハンドラーごとに子メタオブジェクトが用意されています。この子メタオブジェクトのいくつかは、IBM WebSphere Business Integration Server Express and Express Plus ソフトウェアに添付されています。環境を構成するとき、次を行うことができます。

- トップレベル・サーバー・メタオブジェクト属性名を変更する。

  サーバー・アクセスのコンテキストで呼び出されるデータ・ハンドラーで使用されるトップレベル・データ・ハンドラー・メタオブジェクトは、MO_Server_DataHandler です。
- 子メタオブジェクトのデフォルト値を変更して、作成する必要があるデータ・ハンドラー・インスタンスを反映させる。

トップレベル・メタオブジェクトの属性を、MIME タイプ、およびサポートを希望する任意のサブタイプ (BOPrefix) 用に定義します。この属性は子メタオブジェクトを表します。この子メタオブジェクトには、その作業を行うのにデータ・ハンドラーが必要とする、クラス名と構成プロパティーを提供する属性があります。

36 ページの図 12 に、2 つのメタオブジェクトのテキスト・フォーマットを示します。

- トップレベル・データ・ハンドラー・メタオブジェクト MO_Server_DataHandler。

  このメタオブジェクトには、HTML データ・ハンドラー (text.html) によってサポートされる MIME 用に名前付けされた属性が格納されています。この属性

は、HTML データ・ハンドラー MO_DataHandler_DefaultHtmlConfig 用の子データ・ハンドラー・メタオブジェクトを表します。

- HTML データ・ハンドラー MO_DataHandler_DefaultHtmlConfig 用の子データ・ハンドラー・メタオブジェクト。

  子メタオブジェクトは ClassName 属性を宣言します。この属性の DefaultValue 属性プロパティーには、HTML データ・ハンドラーの起動に使用するデータ・ハンドラー・クラス (com.crossworlds.DataHandlers.Html.HtmlDataHandler) の名前がリストされます。

```
[BusinessObjectDefinition]
Name = MO_Server_DataHandler
Version = 1.0.0

     [Attribute]
     Name = text.html
     Type = MO_DataHandler_DefaultHtmlConfig
     ContainedObjectVersion = 1.0.0
     Relationship = Containment
     Cardinality = 1
     MaxLength = 1
     IsKey = true
     IsForeignKey = false
     IsRequired = false
     IsRequiredServerBound = false
     [End]
     [Attribute]
     Name = ObjectEventId
     Type = String
     MaxLength = 255
     IsKey = false
     IsForeignKey = false
     IsRequired = false
     IsRequiredServerBound = false
     [End]

     [Verb]
     Name = Create
     [End]

     [Verb]
     Name = Delete
     [End]

     [Verb]
     Name = Retrieve
     [End]

     [Verb]
     Name = Update
     [End]
[End]
```

図 12. HTML メタオブジェクトのテキスト・フォーマット (1/2)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xsd:schema elementFormDefault="qualified"
   targetNamespace="http://www.ibm.com/websphere"
   xmlns:bx="http://www.ibm.com/websphere"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation><xsd:documentation>

Tue Mar 11 14:25:46 PST 2003

</xsd:documentation>

</xsd:annotation>

<xsd:element name="TestChildBO">d
<xsd:annotation>
<xsd:appinfo>
<bx:boDefinition version="3.0.0" />
</xsd:appinfo>
</xsd:annotation>
<xsd:complexType><xsd:sequence>
<xsd:element name="FirstName" minOccurs="0">
<xsd:annotation>
<xsd:appinfo>
<bx:boAttribute>
<bx:attributeInfo isForeignKey="false" isKey="false" />
</bx:boAttribute>
</xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="LastName" minOccurs="1">
<xsd:annotation>
<xsd:appinfo><bx:boAttribute>
<bx:attributeInfo isForeignKey="false" isKey="true" />
</bx:boAttribute>
|</xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="ObjectEventId" type="xsd:string" minOccurs="0" />
</xsd:sequence>
<xsd:attribute name="version" type="xsd:token" default="0.0.0" />
<xsd:attribute name="delta" type="xsd:boolean" default="false" />
<xsd:attribute name="verb" use="required"><xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="Create" />
<xsd:enumeration value="Delete" />
<xsd:enumeration value="Retrieve" />
<xsd:enumeration value="Update" />
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

図 12. HTML メタオブジェクトのテキスト・フォーマット (2/2)

HTML データ・ハンドラーのサンプル・コード

以下は HTML データ・ハンドラーの Java コード・サンプルです。

```
/**
 * @(#) HtmlDataHandler.java
 *
 * Copyright (c) 1997-2000 CrossWorlds Software, Inc.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of IBM, Inc.
 * You shall not disclose such Confidential information and shall
 * use it only in accordance with the terms of the license agreement you entered into
 * with CrossWorlds Software.
*/
import com.crossworlds.DataHandlers.*;
import com.crossworlds.DataHandlers.Exceptions.*;
import AppSide_Connector.JavaConnectorUtil;

import CxCommon.BusinessObjectInterface;

// java classes
import java.util.*;
import java.io.*;
/**
 ** This is a html data handler which converts a html query
 ** string to a Crossworlds Business object. This example is
 ** assumes the incoming html query is structured in a specific
 ** format as explained in the program below. See the comments
 ** associated with the method parse() in this class.
 */
public class HtmlDataHandler extends DataHandler
{

     /*
      **  A utility method to convert a HTML query string into a crossworlds BO.
      **  See comments associated with the parse() method for a detailed explanation
      **  @param String serializeddata
      **  @param Object the incoming mime type
      */
     public BusinessObjectInterface getBO(String serializedData,
       Object config)
          throws Exception
     {

          HashMap nameValuePairs = parse((String) serializedData);

          /*
           ** Get the BO to be created from the hidden tag BusObjName
           */
          String boName = (String) nameValuePairs.get("BusObjName");
          if (boName == null)
               throw new Exception("Unable to find business object name in "
                    + "serialized business object");

          BusinessObjectInterface bo = JavaConnectorUtil.createBusinessObject(boName);
          String verb = (String) nameValuePairs.get("Verb");
          if (verb == null)
               throw new Exception("Unable to find verb in serialized business object");

          bo.setVerb(verb);

          /*
           ** Get the elements from the HashMap and set it into the BO
           */
          setValues(bo, nameValuePairs);
          return bo;
```

```
}

/*
**  Parse an HTML query string looking for tokens of the form &name=value.
**  The format of the incoming query string must conform to the &name=value
**  format as well as the following semantics:
**        if name does not contain syntax of the form name[X].attribute it is
**        assumed to be the name of an attribute in the parent object otherwise
**        the expression will be used AS IS to set the value of a child object
**        and attribute.
**
**  For example, the following query string can be successfully parsed by
**  this method:
**
** CustomerID=&items[0].itemID=44&items[0].orderQty=25&items[0].
**          deliveryDate1=12/12/00
**  &items[1].itemID=67&items[1].orderQty=2&items[1].
** deliveryDate=12/12/00&Verb=Retrieve&
**  BusObjName=SalesQuote&SubObjName=CwItem
**
**
**  @param String query sent from the webserver to be parsed
**  @return HashMap a hash map containing the name value pairs
*/

private HashMap parse(String queryString)
{
     HashMap nameValuePairs = new HashMap();
     String content = queryString.replace('+', ' ');
     StringTokenizer st = new StringTokenizer(content,"&");

     while (st.hasMoreTokens())
     {
          String token = st.nextToken();
          int i = token.indexOf("=");
          String name = token.substring(0, i);
          String value = token.substring(i+1);

          /*
           **  HTTP will encode certain ASCII values as their hex equivalents.
           ** Convert any of these encodings back to ASCII for both the name
           **  and the value strings (i.e. right hand side of = and left hand
           **  side of =)
           */
          name = replaceHexEncodedWithAscii(name);
          value = replaceHexEncodedWithAscii(value);

          /*
           **  Store these value in the hashmap so that our caller can look
           **  them up.
           */
          nameValuePairs.put(name, value);
     }

     return(nameValuePairs);
}

/*
 *  Given a Hashmap of name/value pairs, enumerate through the business
 *  object and set each attribute in the BO with the corresponding
 *  value from the Hashtable
 *  @param IBusinessObject target of the set
 *  @param Hashmap contains the name/value pairs
 */
private void setValues(BusinessObjectInterface bo, HashMap nameValuePairs)
     throws Exception
{
```

```
        String SubObjName = null;
        Iterator aIterator = nameValuePairs.keySet().iterator();
        // Save the SubObject name so we need to save it
        while (aIterator.hasNext())
        {
             String name = (String) aIterator.next();
             /*
              **  Ignore any hidden keywords that we parsed out of the HTML and
              **  stored in the hash map
              */

             if (name.equalsIgnoreCase("BusObjName")  ||
                       name.equalsIgnoreCase("Verb") ||
                       name.equalsIgnoreCase("SubObjName") ||
                       name.equalsIgnoreCase("ContainerAttrName"))
             {
                  System.out.println("Skipping Item : " + name);
                  continue;
             }

             /*
              **  All subobjects have a grammar in the form of object[X].attribute
              **  where X is the index of the contained subobject.  Therefore, if
              **  the name does not have this embedded string, it's an attribute
              **  of the parent object
              */
             if (name.indexOf("[") == -1)
              bo.setAttrValue(name, (String) nameValuePairs.get(name));
             else
              bo.setAttributeWithCreate(name, (String) nameValuePairs.get(name));
        }
   }

   /*
    *  Replace any hex encoded bytes with the ASCII char equivalent and return
    *  the new string to the caller.
    *  @param name The string to convert.
    */
   private String replaceHexEncodedWithAscii(String name)
   {
        int nameLength = name.length();

        /*
         **  Replace any hex values (HTTP may send over a hex value int
         **  the form of %XX for certain characters) with their
         **  corresponding char equivalents
         */
        StringBuffer nameBuffer = new StringBuffer();
        for (int i = 0; i < nameLength; ++i)
        {
             char c = name.charAt(i);
             switch (c)
             {
                  case '%':
                       byte[] b = { Byte.parseByte(name.substring(i+1, i+3),
                            16) };
                       nameBuffer.append(new String(b));
                       i += 2;
                       break;
                  default:
                       nameBuffer.append(c);
             }
        }
        return(nameBuffer.toString());
   }

   /**
```

```
     ** Implementation of abstract methods in the Data Handler class
     **  @param BusinessObjectInterface the actual business object
     **  @param Object config
     **  @return String string representation of the BO
     */

    public String getStringFromBO(BusinessObjectInterface theObj, Object config)
         throws Exception
    {
         throw new Exception("Not implemented");
    }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param Reader actual data
     *  @param BusinessObjectInterface the actual business object
     *  @param Object config
     */
    public void getBO(Reader serializedData, BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param String actual data
     *  @param BusinessObjectInterface the actual business object
     *  @param Object config
     */
    public void getBO(String serializedData, BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
  }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param BusinessObjectInterface the actual business object
     *  @return InputStream a handle to the stream
     */
    public InputStream getStreamFromBO(BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param Reader actual data
     *  @param BusinessObjectInterface the actual business object
     *  @return BusinessObjectInterface the translated BO
     */
    public BusinessObjectInterface getBO(Reader serializedData, Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }
}
```

サンプル Java コード — ATP サーブレット

以下はサンプル ATP サーブレットです (31 ページの『シナリオ』を参照)。

```
/**
 * @(#) ATPServlet.java
 *
 * Copyright (c) 1997-2000 CrossWorlds Software, Inc.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of
 * IBM. You shall not disclose such Confidential information and shall
 * use it only in accordance with the terms of the license agreement you
 * entered into with IBM Software.
*/
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import java.util.*;
import java.text.*;
import IdlAccessInterfaces.*;
import CxCommon.BusinessObject;

/**
* Available To Promise Servlet example
*/
public class ATPServlet extends HttpServlet
{

     // Defines for some statics
     public static String DEFAULT_SERVER = "CrossWorlds";
     public static String DEFAULT_IOR = "CrossWorlds.ior";
     public static String DEFAULT_USER = "admin";
     public static String DEFAULT_PASSWD = "null";
     // User name to login into the IC Server
     private String userName = DEFAULT_USER;
     // Password
     private String passWord = DEFAULT_PASSWD;
     // ServerName
     private String serverName = DEFAULT_SERVER;
     // IOR File
     private String iorFile = DEFAULT_IOR;
     // AccessSession
     private IInterchangeAccessSession accessSession = null;
     // AccessEngine
     private IAccessEngine accessEngine = null;

     // Servlet Context for getting config information
     private ServletContext ctx;
     // A formatter to print the price with precision.
     private static DecimalFormat formatter;
     // MIME type
     private String mimeType = "text/html";

     /**
     * The init method. This method is used by the web server
     * when the Servlet is loaded for the first time.
     * @param ServletConfig Configuration information
     * associated with the servlet.
     * @exception ServletException is thrown when the
     * servlet cannot be initialized
     */
     public void init(ServletConfig aConfig)throws ServletException
     {
        super.init(aConfig);
        // Formatter for printing prices in the correct format
        formatter = new DecimalFormat();
```

```
      formatter.setDecimalSeparatorAlwaysShown(true);

      // Read up the initial parameters so we can connect to
      // the right ICS server
      String configuredServer = null;
      String configurediorFile = null;
      String configuredUser = null;
      String configuredpassWord = null;
      configuredServer = aConfig.getInitParameter("ICSNAME");
      if ( configuredServer != null)
             {
             this.serverName = configuredServer;
             }
      else
             {
             this.log(
                "No Business Integration Server Express and Express
                 Plus configured, using default of CROSSWORLDS");
             }
      configurediorFile = aConfig.getInitParameter("IORFILE");
      if (iorFile != null)
             {
             this.iorFile = configurediorFile;
             }
      else
             {
             this.log(
               "IOR file not defined, will use CrossWorlds.ior
                from home directory");
             }

      try
             {
             initAccessSession();
             }
      catch(Exception e)
             {
             this.log("Encountered Initialization error", e);
             throw new ServletException(e.toString());
             }
  }
  /**
  * Cleanup method called when the servlet is unloaded from the Web Server
  */
  public void destroy()
{
       // Release our session
       if ( ( accessEngine != null) && (accessSession != null))
             {
             accessEngine.IcloseSession(accessSession);
             accessEngine = null;
             accessSession = null;
             }
  }

  /*
  **  Utility method which creates an access session with Business
  **  Integration Server Express and Express Plus.
  **  If one has already been established then return that.
  **  @exception Exception when an error occurs while establishing
  **  the connection to Business Integration Server Express and Express Plus.
  */
  private synchronized void initAccessSession() throws Exception
  {
       try
             {
             /*
```

```
**  If the access session has already been established then
**  see if the session is still valid (i.e. InterChange
**  Server could have been rebooted since the last time
**   we used the session).
**  If it's not still valid, then open up a new one.
*/
if (accessSession != null)
     {
     try {
          accessSession.IcreateBusinessObject("");
     } catch (ICxAccessError e) {
          /*
          ** Cached session is still valid. We expect
              to get this exception
          */
               return;
     }
     // Catch Corba SystemException
     catch (org.omg.CORBA.SystemException se) {
          /*
           **  The session is invalid.
           **  Open a new one below
           */
          this.log("Re-establishing sessions to ICS");
     }
     }
/**
* Add the relevant Visigenic ORB properties to initialize the
* visigenic ORB.
*/
Properties orbProperties = new java.util.Properties();
orbProperties.setProperty("org.omg.CORBA.ORBClass",
     "com.ibm.CORBA.iiop.ORB");
orbProperties.setProperty("org.omg.CORBA.ORBSingletonClass",
     "com.ibm.rmi.corba.ORBSingleton");
org.omg.CORBA.ORB  orb =
     org.omg.CORBA.ORB.init((String[])null, orbProperties);
/*
**  Use the file that contains the Internet Inter-Orb
**  Reference.
**  This object reference will be a serialized CORBA object
**  reference to the running Business Integration Server
**  Express and Express Plus that
**  we wish to talk to.
*/
LineNumberReader input =
     new LineNumberReader(new FileReader(iorFile));
/*
**  Create a memory resident CORBA object reference from
**  the IOR
**  in the file
*/
org.omg.CORBA.Object object = orb.string_to_object
   (input.readLine());

/*
**  Now get create a real session with the running object
*/
accessEngine = IAccessEngineHelper.narrow(object);
if (accessEngine == null)
     throw new Exception("Unable to communicate with server
       " + serverName + " using IOR from " + iorFile);

/*
**  Now that we have an object reference to a running
**  server, we must authenticate ourselves before we
**  can get a session that is useful.
```

```
            */
            accessSession = accessEngine.IgetInterchangeAccessSession(
                 userName,
                 passWord);
            if (accessSession == null)
                 throw new Exception("Invalid user name and password");

            }
       catch (Exception e)
            {
            this.log("Encountered orb Initialization error" , e);
            if (e instanceof org.omg.CORBA.SystemException)
                 throw new Exception(e.toString());
            else
                 throw e;
            }
}

/**
* Get method called by the WebServer whenever a GET action
* is requested by an HTML page.
* @param HttpServletRequest handle to the http request
* object@param HttpServletResponse handle to the http response
* object @exception ServletException is thrown when the servlet
* encounters an error @exception is thrown when the
* Webserver cannot communicate to the calling
* html page
*/
public void doGet(HttpServletRequest req, HttpServletResponse res)
       throws ServletException, IOException
{
       // String serializedHTMLQuote = null;

       // A BusinessObject to hold our incoming BO from the
       // requesting HTML page
       IBusinessObject aBO = null;

       // A BusinessObject to hold our resultant BO from the
       // result of the Collaboration execution

       IBusinessObject returnedQuoteBusObj = null;

       /*
       **  Make sure we have a valid access session with the interchange
       **  server first
       */
       try
            {
             initAccessSession();
            }
       catch(Exception e)
            {
             throw new ServletException
                ("InitAccessSession Failed " + e.toString());
            }

       // Create a BO from the data provided by the HTML page
       try {
            aBO =
                 accessSession.IcreateBusinessObjectFrom
                   (req.getQueryString(),
                     mimeType);
       } catch (ICxAccessError e) {
             throw new ServletException
               (" Creating Business Object Failed : " +
                 e.IerrorMessage);
       }
```

```
if (aBO == null)
     {
      throw new ServletException("Attempting to use Null Bo ");
     }
/*
**  Execute the collaboration.  We'll get back a
**  CrossWorlds business object that contains an ATP
**  date for each item.
*/
try
     {
     returnedQuoteBusObj = accessSession.IexecuteCollaboration(
          "ATPExample","From", aBO);
     }
catch(IExecuteCollaborationError ae)
     {
     String error = "Collaboration Error :
          " + ae.IerrorMessage
                         + ae.status;
     this.log("Collaboration Error", ae);
     throw new ServletException(error);
     }
/*
**  Now create a table to send back that has:
**    ItemNumber    Quantity    Price
*/
res.setContentType(mimeType);
PrintWriter out = res.getWriter();
out.println("<body>");
out.println("<TABLE BORDER=¥"1¥">");
out.println("<caption align=¥"center¥" > " +
     "<font face=e=¥"Haettenschweiler¥" size=¥"7¥">" +
     "Sales Quote Response</caption>");

out.println("<TR> <TH>Item ID" +
     "<TH> Item Description"    +
     "<TH> Quantity " +
     "<TH> Item Price" +
     "<TH> Available Date " +
     "<TH> Total Price " +
     "</TH> </TR>");
IBusinessObjectArray itemContainer = null;
try {
     itemContainer =
          returnedQuoteBusObj.
           IgetBusinessObjectArrayAttribute
            ("OrderItems");
} catch (IInvalidAttributeTypeException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IInvalidAttributeNameException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IAttributeBlankException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IAttributeNotSetException e) {
     throw new ServletException(e.IerrorMessage);
}

// A subobject to hold each individual Item
IBusinessObject item = null;

int size = itemContainer.IgetSize();
// Loop thru the array and print each item
// separately
String attr = null;
int itemQuantity = 0;
double itemPrice = 0;
```

```
//Loop thru the array of returned items
for (int i = 0;i < size; i++)
     {
     try
          {
          // Get the item BusinessObject at the
              current indexitem =
              itemContainer.IgetBusinessObjectAtIndex(i);
          if (item != null)
               {
               // Build a html table row beginning with ITemID
               // attribute
               try {
                    attr = item.IgetStringAttribute("ItemID");
                    out.print("<TR> <TD> " +
                         attr +
                         "</TD>" + "<TD>");
                    // We have printed the value,
                    // set it to null again
                    attr = null;
               } catch (IAttributeNotSetException e) {
                    attr = "N/A";
                    out.print("<TR> <TD> ");
                    out.print(attr + "</TD>" + "<TD>");
               } catch (IInvalidAttributeNameException e) {
                         attr = "N/A";
                         out.print("<TR> <TD> ");
                         out.print(attr + "</TD>" + "<TD>");
               } catch (IInvalidAttributeTypeException e) {
                    attr = "N/A";
                    out.print("<TR> <TD> ");
                    out.print(attr + "</TD>" + "<TD>");
               }
               // Get the ItemType attribute
               try {
                    attr = item.IgetStringAttribute
                      ("itemType");
                    out.print(attr + "</TD>" + "<TD>");
                    // We have printed the value,
                    // set it to null again
                    attr = null;
               } catch (IAttributeNotSetException e) {
                    attr = "N/A";
                    out.print(attr + "</TD>" + "<TD>");
               } catch (IInvalidAttributeNameException e) {
                    attr = "N/A";
                    out.print(attr + "</TD>" + "<TD>");
               } catch (IInvalidAttributeTypeException e) {
                    attr = "N/A";
                    out.print(attr + "</TD>" + "<TD>");
               }
               // Get the orderQty Attribute
               try {
                    attr = item.IgetStringAttribute
                       ("orderQty");
                    try {
                         itemQuantity = Integer.parseInt(attr);
                    } catch (NumberFormatException e) {
                         itemQuantity = -1;
                    }
                    out.print(attr + "</TD>" + "<TD>");
                    // We have printed the value,
                    // set it to null again
                    attr = null;
               } catch (IAttributeNotSetException e) {
                    attr = "N/A";
                    itemQuantity = -1;
```

```
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
// Get the ItemPrice attribute
try {
     attr = item.IgetStringAttribute("itemPrice");
     int indexOfDollar = attr.indexOf("$");
     String priceToParse = null;
     // Locate if we have "$" in the value
     if (indexOfDollar == -1)
          priceToParse = attr;
     else
          priceToParse = attr.substring
              (indexOfDollar + 1);
     // Format the price so it looks like $NNNN.NN
     try {
          itemPrice = Double.parseDouble

          (priceToParse);
     } catch (NumberFormatException e) {
          itemPrice = -1;
     }
     out.print(attr + "</TD>" + "<TD>");
     // We have printed the value,
         set it to null again
     attr = null;
} catch (IAttributeNotSetException e) {
     attr = "N/A";
     itemPrice = -1;
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
// Get the ATPDate and print it
try {
     attr = item.IgetStringAttribute("ATPDate");
     out.print(attr + "</TD>" + "<TD>");
} catch (IAttributeNotSetException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
/*
**  Now print the total price for the item.
**  If we don't have sufficient information then
**  print N/A
*/
if ((itemPrice == -1) || (itemQuantity == -1))
     {
     out.println(attr + "</TD>" + "<TD>");
     // We have printed the value,
     // set it to null again
     attr = null;
     }
```

```
                    else
                         {
                         double totalPrice = itemQuantity
                           * itemPrice;
                         out.println("$" + formatter.format
                           (totalPrice).trim()
                              + "</TD>"
                              + "<TD>");
                         }
                    } // end if (Item != null)
               } // End try
               catch (IAttributeBlankException e2) {
                    continue;
               } catch (IInvalidIndexException e) {
                    throw new ServletException(e.getMessage());
               }

          }// End for loop
     // Close the HTML table
     out.println("</TABLE>");
     // Finish the page body
     out.println("</body></html>");
     } // end do get
}
```

第 3 部 サーバー・アクセス API リファレンス

第 6 章 IAccessEngine インターフェース

IAccessEngine インターフェースは、IBM WebSphere Business Integration Server Express and Express Plus とのセッションをオープンおよびクローズする方法を提供します。表 7 に、IAccessEngine インターフェースのメソッドについて要約します。

表 7. IAccessEngine インターフェースのメンバー・メソッド

メソッド	説明	ページ
IgetInterchangeAccessSession()	アクセス・クライアント用に、IBM WebSphere Business Integration Server Express and Express Plus へのアクセス・セッションを作成します。	53
IcloseSession()	IBM WebSphere Business Integration Server Express and Express Plus とのアクセス・セッションを閉じます。	54

IgetInterchangeAccessSession()

アクセス・クライアント用に、IBM WebSphere Business Integration Server Express and Express Plus へのアクセス・セッションを作成します。

構文

```
IInterchangeAccessSession IgetInterchangeAccessSession(
     string userName, string password);
```

パラメーター

userName IBM WebSphere Business Integration Server Express and Express Plus ユーザーの名前。

password IBM WebSphere Business Integration Server Express and Express Plus ユーザーのパスワード。

戻り値

アクセス・セッション用の IInterchangeAccessSession オブジェクト。

例外

ICxAccessError 無効なユーザー名またはパスワードが検出された場合にスローされます。

注記

IgetInterchangeAccessSession() メソッドは、*userName* と *password* が IBM WebSphere Business Integration Server Express and Express Plus インスタンスに有効であることを確認します。

要確認: このメソッドのユーザー名は admin である必要があります。

例

```
// Open the access session
String userName = "admin";
String password = "null";
IInterchangeAccessSession aSession =
     serverAccessEngine.IgetInterchangeAccessSession(
     userName,
     password);
```

IcloseSession()

IBM WebSphere Business Integration Server Express and Express Plus とのアクセス・セッションを閉じます。

構文

```
void IcloseSession(IInterchangeAccessSession session);
```

パラメーター

session クローズするアクセス・セッション・オブジェクト。

戻り値

なし。

例

```
// Close the access session
serverAccessEngine.IcloseSession(aSession);
```

第 7 章 IInterchangeAccessSession インターフェース

IInterchangeAccessSession インターフェースには、ビジネス・オブジェクトを作成するメソッドおよびコラボレーションを実行する方法が用意されています。表 8 に、IInterchangeAccessSession インターフェースのメソッドについて要約します。

表 8. IInterchangeAccessSession インターフェースのメンバー・メソッド

メソッド	説明	ページ
IcreateBusinessObject()	指定されたビジネス・オブジェクト定義からビジネス・オブジェクトを作成します。	55
IcreateBusinessObjectArray()	1 つ以上の要素が含まれるビジネス・オブジェクト配列を作成します (各要素にはそのタイプとして指定されたビジネス・オブジェクトがあります)。	56
IcreateBusinessObjectFrom()	指定された MIME フォーマットの直列化データを IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトに変換します。	57
IcreateBusinessObjectWithVerb()	指定された動詞付きのビジネス・オブジェクトを作成します。	58
IexecuteCollaboration()	コラボレーションを実行します。ビジネス・オブジェクトにアクセス要求内のトリガー・アクセス・データとして送信します。	59
IexecuteCollaborationExtFmt()	コラボレーションを実行します。直列化データにアクセス要求内のトリガー・アクセス・データとして送信します。	60
IreleaseBusinessObject()	ビジネス・オブジェクトのリソースを解放します。	62
IreleaseBusinessObjectArray()	ビジネス・オブジェクト配列のリソースを解放します。	62
setLocale(String)	ロケールを設定します。	63

IcreateBusinessObject()

指定されたビジネス・オブジェクト定義からビジネス・オブジェクトを作成します。

構文

```
IBusinessObject IcreateBusinessObject(string busObjName);
```

パラメーター

busObjName ビジネス・オブジェクト作成時に使用するビジネス・オブジェクト定義の名前。

戻り値

新規ビジネス・オブジェクトを保持する IBusinessObject オブジェクト。

例外

ICxAccessError 指定されたビジネス・オブジェクト定義が IBM WebSphere Business Integration Server Express and Express Plus リポジトリーに存在しない 場合にスローされます。

注記

サーバー・アクセスは、タイプ *busObjName* のビジネス・オブジェクトを作成し、それをアクセス・クライアントに送り戻します。

例

次のコード・フラグメントでは、ビジネス・オブジェクトが作成されます。

```
// This method creates a business object
// Declare our object
IBusinessObject exampleObj = null;
exampleObj = aSession.IcreateBusinessObject("PayablesNetChange");
```

IcreateBusinessObjectArray()

1 つ以上の要素が含まれるビジネス・オブジェクト配列を作成します (各要素にはそのタイプとして指定されたビジネス・オブジェクトがあります)。

構文

```
IBusinessObjectArray IcreateBusinessObjectArray(string busObjName);
```

パラメーター

busObjName ビジネス・オブジェクト配列内のビジネス・オブジェクト作成時に使用するビジネス・オブジェクト定義の名前。

戻り値

新規ビジネス・オブジェクト配列を保持する IBusinessObjectArray オブジェクト。

例外

ICxAccessError 指定されたビジネス・オブジェクト定義が IBM WebSphere Business Integration Server Express and Express Plus リポジトリーに存在しない 場合にスローされます。

注記

サーバー・アクセス方式は、ビジネス・オブジェクト配列を作成し、それをアクセス・クライアントに送り戻します。IcreateBusinessObjectArray() メソッドは IBusinessObjectArray オブジェクトを戻します。 IBusinessObjectArray インターフェース内の他のメソッドで、ビジネス・オブジェクト配列を取り扱うことができます。

例

次の例では、ビジネス・オブジェクト配列が作成されます。

```
// Declare the array
IBusinessObjectArray exampleObjArray = null;
// Create the business object array that holds "CustomerAcct"
// business objects
exampleObjArray =
     accessSession.IcreateBusinessObjectArray("CustomerAcct");
```

IcreateBusinessObjectFrom()

指定された MIME フォーマットの直列化データを IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトに変換します。

構文

```
IBusinessObject IcreateBusinessObjectFrom(string serializedData,
     string mimeType);
```

パラメーター

serializedData	着信直列化データ。
mimeType	*serializedData* データの MIME タイプ。

戻り値

serializedData データからデータ・ハンドラーが作成するビジネス・オブジェクトを保持する IBusinessObject オブジェクト。

例外

ICxAccessError	データをビジネス・オブジェクトに変換できないときに、またはデータ・ハンドラーにアクセスできない場合に、スローされます。

注記

IcreateBusinessObjectFrom() メソッドは、その指定された *mimeType* MIME タイプの *serializedData* データを IBM WebSphere Business Integration Server Express and Express Plus に送信します。IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスは、指定された MIME タイプを IBM WebSphere Business Integration Server Express and Express Plus 環境と互換性のある IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトに変換するために必要なデータ・ハンドラーを呼び出します。

serializedData データは、ビジネス・オブジェクト作成時に使用するビジネス・オブジェクト定義の名前を指定する必要があります。データ・ハンドラーは、データの解析およびビジネス・オブジェクトへの変換を行い、それを IBM WebSphere Business Integration Server Express and Express Plus 内のサーバー・アクセスに戻します。続いて、IBM WebSphere Business Integration Server Express and Express Plus はそれをアクセス・クライアントに戻します。直列化データの外部フォーマットは、データ・ハンドラー (IBM WebSphere Business Integration Server Express and Express Plus 提供の、またはユーザー作成のカスタム・データ・ハンドラー) がサポートするタイプである必要があります。詳しくは、「データ・ハンドラー・ガイド」を参照してください。

例

```
// Declare the object
String custData = "exampleXmlData";
String mimeType = "text/Xml";
IBusinessObject exampleObj = null;
// This method creates the business object from data in XML format
exampleObj =
     accessSession.IcreateBusinessObjectFrom(custData, mimeType);
```

IcreateBusinessObjectWithVerb()

指定された動詞付きのビジネス・オブジェクトを作成します。

構文

```
IBusinessObject IcreateBusinessObjectWithVerb(string busObjName,
     string verb);
```

パラメーター

busObjName ビジネス・オブジェクト作成時に使用するビジネス・オブジェクト定義の名前。

verb 新規ビジネス・オブジェクトの動詞。

戻り値

指定された *verb* 値付きの新規ビジネス・オブジェクトを保持する IBusinessObject オブジェクト。

例外

ICxAccessError	指定されたビジネス・オブジェクト定義が IBM WebSphere Business Integration Server Express and Express Plus リポジトリーに存在しない とき、または渡された *verb* がビジネス・オブジェクト定義に対して無効である場合にスローされます。

注記

サーバー・アクセスは、タイプ busObjName のビジネス・オブジェクトを作成し、それを *verb* 動詞で初期化します。続いて、このビジネス・オブジェクトをアクセス・クライアントに送り戻します。ビジネス・オブジェクト定義でサポートされた動詞のみが有効です。

例

```
// Create the business object
IBusinessObject exampleobj = null
exampleObj =
     accessSession.IcreateBusinessObjectWithVerb("AcctsRecCurrent",
     "Retrieve");
```

IexecuteCollaboration()

コラボレーションを実行します。ビジネス・オブジェクトにアクセス要求内のトリガー・アクセス・データとして送信します。

構文

```
IBusinessObject IexecuteCollaboration
     (string collabName, string portName, IBusinessObject busObj);
```

パラメーター

collabName	実行するコラボレーションの名前。
portName	アクセス・クライアントのバインド先の外部コラボレーション・ポートの名前。
busObj	コラボレーション用のトリガー・アクセス・データが格納されている汎用ビジネス・オブジェクト。

戻り値

コラボレーションが戻るビジネス・オブジェクトが格納されている IBusinessObject オブジェクト。

例外

IExecuteCollaborationError

コラボレーションがアクティブでない、またはマップが失敗した場合にスローされます。この例外には、例外発生時に呼び出しの詳細を示す、次の定数のいずれかに設定される状況値が含まれます。この状況へのアクセス方法の詳細については、100 ページの『IExecuteCollaborationError』を参照してください。

定数名	説明
UNKNOWNSTATUS	IexecuteCollaboration() メソッドの呼び出しの状況が不明です。
FAILEDTOREACHCOLLABORATION	アクセス要求は、コラボレーションに届きませんでした。

定数名	説明
FAILEDINEXECUTIONOFCOLLABORATION	アクセス要求は、コラボレーションの実行中に失敗しました。
FAILEDINRETURNTOCLIENT	コラボレーションが実行されましたが、アクセス・クライアントへの応答の配送時にエラーが発生しました。

注記

IexecuteCollaboration() メソッドは、*collabName* コラボレーションの実行を要求します。コラボレーションを開始するため、サーバー・アクセスは *busObj* ビジネス・オブジェクト内のトリガー・アクセス・データを *collabName* コラボレーションの *portName* ポートに送信します。このポートは、コール・トリガー・フローがサポートされるように、外部として構成する必要があります。

注: コラボレーション、ポート、およびビジネス・オブジェクトは、コール・トリガー・フローおよび操作に対して構成およびマップする必要があります。

例

```
String portName = "From";
IBusinessObject srcBO =
     accessSession.IcreateBusinessObject ("payableNetChange");

// set srcBO attributes, verb, or both
...
// Execute the collaboration
IBusinessObject resultantBO = null;
resultantBO = accessSession.IexecuteCollaboration(
     "getCustAcctPayable",
     portName,
     srcBO);
```

IexecuteCollaborationExtFmt()

コラボレーションを実行します。直列化データにアクセス要求内のトリガー・アクセス・データとして送信します。

構文

```
string IexecuteCollaborationExtFmt(string collabName, string portName,
          string serializedData, string mimeType, string verb);
```

パラメーター

collabName	実行するコラボレーションの名前。
portName	アクセス・クライアントのバインド先の外部コラボレーション・ポートの名前。
serializedData	トリガー・アクセス・データを表す直列化データ。
mimeType	直列化データの外部フォーマット (MIME タイプとして)。
verb	ビジネス・オブジェクトの動詞に関する値。

戻り値

コラボレーションが戻すビジネス・オブジェクトの直列化バージョンが格納されているストリング。このストリングは、*mimeType* 外部フォーマットです。

例外

IExecuteCollaborationError
: コラボレーションがアクティブでない、またはマップが失敗した場合にスローされます。この例外には、例外発生時に呼び出しの詳細を示す、次の定数のいずれかに設定される状況値が含まれます。この状況へのアクセス方法の詳細については、100 ページの『IExecuteCollaborationError』を参照してください。

定数名	説明
UNKNOWNSTATUS	IexecuteCollaborationExtFmt() メソッドの呼び出しの状況が不明です。
FAILEDTOREACHCOLLABORATION	アクセス要求は、コラボレーションに届きませんでした。
FAILEDINEXECUTIONOFCOLLABORATION	アクセス要求は、コラボレーションの実行中に失敗しました。
FAILEDINRETURNTOCLIENT	コラボレーションが実行されましたが、アクセス・クライアントへの応答の配送時にエラーが発生しました。

注記

IexecuteCollaborationExtFmt() メソッドは、IexecuteCollaboration() と同じ基本タスクを実行します。すなわち、*collabName* コラボレーションの実行を要求します。このメソッドでは単一の呼び出しで次のタスクを実行できることが主な違いです。

- データの *mimeType* MIME タイプに適切なデータ・ハンドラーを使用して、*serializedData* データをビジネス・オブジェクトに変換します。このビジネス・オブジェクトは、コラボレーション用のトリガー・アクセス・データを表します。
- ビジネス・オブジェクトの動詞を、指定された verb 値に設定します。
- ビジネス・オブジェクトをコラボレーションの *portName* ポートに送信して、コラボレーションの実行を開始します。このポートは、コール・トリガー・フローがサポートされるように、外部として構成する必要があります。

注: このメソッドとの間で引き渡される CORBA オブジェクトはありません。

コラボレーションおよびポートは、コール・トリガー・フローおよび操作に対して構成およびマップする必要があります。

mimeType パラメーターは、ビジネス・オブジェクト用の直列化データの外部フォーマットを指定します。サーバー・アクセスは、この MIME タイプを使用して、データの解析および IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトへの変換を行うために呼び出すデータ・ハンドラーを判別します。外部フォーマットは、データ・ハンドラー (IBM WebSphere Business

Integration Server Express and Express Plus 提供の、またはユーザー作成のカスタム・データ・ハンドラー) がサポートするタイプである必要があります。データ処理の詳細については、「データ・ハンドラー・ガイド」を参照してください。

例

```
String portName = "From";
// Execute the collaboration
IBusinessObject resultantBO = null;
resultantBO = accessSession.IexecuteCollaborationExtFmt(
     "getCustAcctPayable",
     portName,
     serializedXMLData,
     "text/xml",
     "Create");
```

IreleaseBusinessObject()

ビジネス・オブジェクトのリソースを解放します。

構文

```
void IreleaseBusinessObject(IBusinessObject releaseObject);
```

パラメーター

releaseObject リソースが解放されるビジネス・オブジェクト。

戻り値

なし。

注記

アクセス・クライアントがビジネス・オブジェクトの使用を終了したときには、IreleaseBusinessObject() メソッドを使用して、IBM WebSphere Business Integration Server Express and Express Plus メモリー内の IBusinessObject オブジェクトを解放する必要があります。

例

```
// Create the business object
IBusinessObj anObject = null;
accessSession.IcreateBusinessObjectWithVerb("AcctsRecCurrent",
     "Retrieve");
// Release the object
accessSession.IreleaseBusinessObject(anObject);
```

IreleaseBusinessObjectArray()

ビジネス・オブジェクト配列のリソースを解放します。

構文

```
void IreleaseBusinessObjectArray(IBusinessObjectArray releaseObject);
```

パラメーター

releaseObject リソースが解放されるビジネス・オブジェクト配列。

戻り値

なし。

注記

アクセス・クライアントがビジネス・オブジェクト配列の使用を終了したときには、IreleaseBusinessObjectArray() メソッドを使用して、IBM WebSphere Business Integration Server Express and Express Plus メモリー内の IBusinessObjectArray オブジェクトを解放する必要があります。

例

```
// Create the array
IBusinessObjectArray exampleObjArray = null;
exampleObjArray =
     accessSession.IcreateBusinessObjectArray("CustomerAcct");
// Release the object array
accessSession.IreleaseBusinessObjectArray(exampleObjArray);
```

---

setLocale(String)

アクセス・インターフェース・セッション・オブジェクトのロケールを設定します。

構文

```
public String setLocale(String);
```

パラメーター

次のフォーマットでロケールを指定するストリング。

ll_TT

ここで、*ll* は 2 文字言語コード (通常は小文字) で、*TT* はオプションの 2 文字国別および地域コード (通常は大文字) です。例えば、次のストリングは有効なロケールです。

```
en
de_DE
```

注記

setLocale() メソッドは、アクセス・インターフェース・セッション・オブジェクトに対してロケールを設定します。ロケールは、言語および国別 (または地域) に応じて国別情報を定義します。

デフォルトでは、セッション・オブジェクトの先頭で使用されるロケールは、IBM WebSphere Business Integration Server Express and Express Plus に使用されるロケー

ルと同じです。setLocale() メソッドで呼び出しを使用して新規のロケールに変更すると、セッション・オブジェクトでのそれ以降のすべてのメソッドでの呼び出しは、その新規ロケールを使用します。

第 8 章 IBusinessObject インターフェース

IBusinessObject インターフェースは、タイプ BusinesssObject のオブジェクトで作動するメソッドを提供します。これらは、IBM WebSphere リポジトリーで定義されている IBM WebSphere Business Integration Server Express and Express Plus システムのビジネス・オブジェクトを表します。表 9 に、IBusinessObject インターフェースのメソッドについて要約します。

表 9. IBusinessObject インターフェースのメンバー・メソッド

メソッド	説明	ページ
Iduplicate()	ビジネス・オブジェクトのクローンを作成します。	66
Iequals()	このビジネス・オブジェクトの属性値を入力ビジネス・オブジェクトの属性値と比較します。	67
IequalsKeys()	このビジネス・オブジェクトのキー属性値を入力ビジネス・オブジェクトのキー属性値と比較します。	68
IgetAppSpecificInfo()	属性に関するアプリケーション固有の情報を取得します。	69
IgetAttributeCount()	ビジネス・オブジェクト内にある属性の数を取得します。	69
IgetAttributeName()	ビジネス・オブジェクト定義内の指定された位置にある属性名を取得します。	70
IgetAttributeType()	属性のタイプを取得します。	70
IgetAttributeTypeAtIndex()	ビジネス・オブジェクト定義内の指定された位置にある属性のタイプを取得します。	71
IgetBooleanAttribute()	属性の boolean 値を取得します。	71
IgetBOAppSpecification()	ビジネス・オブジェクト配列 (複数カーディナリティー) である属性の値を取得します。	72
IgetBusinessObjectArrayAttribute()	ビジネス・オブジェクト配列 (複数カーディナリティー) であるビジネス・オブジェクト属性の値を取得します。	73
IgetBusinessObjectAttribute()	単一カーディナリティーの属性の値を取得します。	73
IgetDateAttribute()	日付属性の値を取得します。	74
IgetDefaultValue()	属性のデフォルト値を取得します。	75
IgetDoubleAttribute()	属性の double 値を取得します。	75
IgetFloatAttribute()	属性の float 値を取得します。	76
IgetIntAttribute()	属性の int 値を取得します。	77
IgetLongTextAttribute()	属性の longtext 値を取得します。	78
IgetName()	ビジネス・オブジェクト定義の名前を取得します。	78
IgetStringAttribute()	属性の string 値を取得します。	79

表 9. *IBusinessObject* インターフェースのメンバー・メソッド (続き)

メソッド	説明	ページ
IgetVerb()	ビジネス・オブジェクト用の動詞を取得します。	79
IisAttributeMultipleCardinality()	属性に複数カーディナリティーがあるかどうかを判別します。	80
IisBlankValue()	属性値がブランク値であるかどうかを判別します。	80
IisIgnoreValue()	属性値が「ignore」であるかどうかを判別します。	81
IisKey()	属性がキーであるかどうかを判別します。	82
IisRequired()	指定された属性が必要であるかどうかを判別します。	82
Iserialize()	読み取り可能 (直列化) フォーマットの属性データを戻します。	83
IsetAttributes()	指定された MIME タイプの直列化データからビジネス・オブジェクト内の属性を設定します。	83
IsetAttributeToBlank()	ビジネス・オブジェクト内の属性をブランク値に設定します。	84
IsetAttributeToIgnore()	ビジネス・オブジェクト内の属性を「ignore」に設定します。	84
IsetBooleanAttribute()	属性を `boolean` 値に設定します。	85
IsetBusinessObjectArrayAttribute()	ビジネス・オブジェクト配列 (複数カーディナリティー) である属性の値を設定します。	85
IsetBusinessObjectAttribute()	単一カーディナリティーの属性の値を設定します。	86
IsetDateAttribute()	属性を date 値に設定します。	87
IsetDoubleAttribute()	属性を `double` 値に設定します。	87
IsetFloatAttribute()	属性を `float` 値に設定します。	88
IsetIntAttribute()	属性を `int` 値に設定します。	88
IsetLongTextAttribute()	属性を `longtext` 値に設定します。	89
IsetStringAttribute()	属性を `string` 値に設定します。	90
IsetVerb()	ビジネス・オブジェクト用の動詞を設定します。	90
ItoExternalForm()	指定された MIME タイプの外部フォーマットにビジネス・オブジェクト・データを直列化します。	91
ItoString()	IBM WebSphere Business Integration Server Express and Express Plus フォーマットを使用してビジネス・オブジェクト・データを直列化します。	92

Iduplicate()

ビジネス・オブジェクトのクローンを作成します。

構文

```
IBusinessObject Iduplicate();
```

パラメーター

なし。

戻り値

重複するビジネス・オブジェクトが格納されている IBusinessObject オブジェクト。

例外

ICxAccessError	オブジェクトが検出できない場合にスローされます。

注記

Iduplicate() メソッドは、ビジネス・オブジェクトのクローンを作成して、それを戻します。このメソッドの戻り値を IBusinessObject タイプの宣言変数に明示的に割り当てる必要があります。

例

次の例では、sourceCustomer を複写して destCustomer を作成します。

```
IBusinessObject destCustomer = sourceCustomer.Iduplicate();
```

Iequals()

このビジネス・オブジェクトの属性値を入力ビジネス・オブジェクトの属性値と比較します。

構文

```
boolean Iequals(IBusinessObject obj2);
```

パラメーター

obj2　比較するビジネス・オブジェクト。

戻り値

すべての 属性の値と動詞が同じである場合は、true を戻します。同じでない場合は、false を戻します。

注記

Iequals() メソッドは、このビジネス・オブジェクトの属性値を入力ビジネス・オブジェクトの属性値と比較します。ビジネス・オブジェクトが階層型である場合、比較には、子ビジネス・オブジェクトのすべての 属性が含まれます。動詞と属性値は一致する必要があります。

比較で、null 値は、それが比較される対象の値に等価であるとみなされ、true の戻りを無効にしません。

例

次の例では、order2 の動詞と属性を order1 のすべての属性と比較します。

```
boolean isEqual = false;
IBusinessObject order1 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "create");
IBusinessObject order2 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "create");
isEqual = order1.Iequals(order2);
if(isEqual)
     System.out.println("order1 is the same as order2")
else
     System.out.println("order1 is not the same as order2");
```

IequalsKeys()

このビジネス・オブジェクトのキー属性値を入力ビジネス・オブジェクトのキー属性値と比較します。

構文

```
boolean IequalsKeys(IBusinessObject obj2);
```

パラメーター

obj2　比較用に評価するビジネス・オブジェクト。

戻り値

すべての キー属性の値が同じである場合は、true を戻します。同じでない場合は、false を戻します。

注記

IequalsKeys() メソッドは浅い比較を実行します。すなわち、子ビジネス・オブジェクト内のキーは比較しません。

例

次の例では、order2 のキー属性を order1 のキー属性と比較します。ただし、子ビジネス・オブジェクトの属性がある場合は、これを除きます。

```
boolean keyEqual = false;
IBusinessObject order1 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "retrieve");
IBusinessObject order2 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "retrieve");
keyEqual = order1.IequalsKeys(order2);
if(keyEqual)
     System.out.println("order1 is the same as order2")
else
     System.out.println("order1 is not the same as order2");
```

IgetAppSpecificInfo()

属性に関するアプリケーション固有の情報を取得します。

構文

```
string IgetAppSpecificInfo(string attributeName)
```

パラメーター

attributeName 属性の名前。

戻り値

指定された属性に関連付けられたアプリケーション固有の情報が格納されている string。

例外

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IValueNotSetException
: 属性にアプリケーション固有の情報がない場合にスローされます。

注記

IgetAppSpecificInfo() メソッドは null を戻すことができます。

例

```
// This method determines the app-specific info of an attribute
String appSpecificInfo;
appSpecificInfo = aBusObj.IgetAppSpecificInfor();
```

IgetAttributeCount()

ビジネス・オブジェクト内にある属性の数を取得します。

構文

```
long IgetAttributeCount();
```

パラメーター

なし。

戻り値

現行のビジネス・オブジェクトにある属性の数を示す整数値。

例

```
long attributeCount = 0;
attributeCount = aBusObj.IgetAttributeCount();
```

IgetAttributeName()

ビジネス・オブジェクト定義内の指定された位置にある属性名を取得します。

構文

```
string IgetAttributeName(long position);
```

パラメーター

position ビジネス・オブジェクト定義内での属性の位置。

戻り値

ビジネス・オブジェクト定義内の指定された位置にある属性の名前が格納されている string。

例外

IInvalidIndexException 位置指標が無効な場合にスローされます。

例

```
int position = 1;
String attribute name;
attributeName = aBusObj.IgetAttributeName(position);
```

IgetAttributeType()

属性のタイプを取得します。

構文

```
long IgetAttributeType(string attributeName);
```

パラメーター

attributeName タイプを戻す属性の名前。

戻り値

ビジネス・オブジェクト内の指定された属性のデータ型を示す整数。次を参照してください。

0 Object

1 boolean

2 int

3 float

4 double

5 string

6 date

7 longtext

例外

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
String attributeName = "Name";
long attributeType = 0;
attributeType = aBusObj.IgetAttributeType(attributeName);
```

IgetAttributeTypeAtIndex()

ビジネス・オブジェクト定義内の指定された位置にある属性のタイプを取得します。

構文

```
long IgetAttributeTypeAtIndex(long position);
```

パラメーター

position ビジネス・オブジェクト定義内での属性の位置。

戻り値

ビジネス・オブジェクト内の指定された位置にある属性のデータ型を示す整数。次を参照してください。

0 Object

1 boolean

2 int

3 float

4 double

5 string

6 date

7 longtext

例外

IInvalidIndexException 位置指標が無効な場合にスローされます。

例

```
int indexPosition = 1;
long attributeType = 0;
attributeType = aBusObj.IgetAttributeTypeAtIndex(indexPosition);
```

IgetBooleanAttribute()

属性の boolean 値を取得します。

構文

```
boolean IgetBooleanAttribute(string attributeName);
```

パラメーター

attributeName　値を取得する boolean 属性の名前。

戻り値

属性の boolean 値。

例外

IAttributeNotSetException
: 属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
: 属性が boolean 日付型でない場合にスローされます。

IAttributeBlankException
: 属性にブランク値がある場合にスローされます。

例

```
// Call the boolean method
String booleanAttribute = "MyBooleanAttribute";
boolean value = exampleBusObj.IgetBooleanAttribute(booleanAttribute);
```

IgetBOAppSpecification()

アプリケーション固有の情報を取得します。

構文

```
public String IgetBOAppSpecificInfo();
```

パラメーター

このメソッドには入力パラメーターがありません。

戻り値

ビジネス・アプリケーションのアプリケーション固有の情報を含む IgetBOAppSpecificInfo() オブジェクト。

例外

IValueNotSetException
: 属性値が無効な場合にスローされます。

IgetBusinessObjectArrayAttribute()

ビジネス・オブジェクト配列 (複数カーディナリティー) である属性の値を取得します。

構文

```
IBusinessObjectArray IgetBusinessObjectArrayAttribute(
     string attributeName);
```

パラメーター

attributeName
値を取得する複数カーディナリティー属性の名前。

戻り値

複数カーディナリティー属性の値が格納されている IBusinessObjectArray オブジェクト。

例外

IAttributeNotSetException
属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
属性が単一カーディナリティー属性でない場合 (他の日付型の場合) にスローされます。

IAttributeBlankException
属性にブランク値がある場合にスローされます。

例

```
// Call the BusinessObjectArray method and get the attribute
String arrayAttribute = "Account";
IBusinessObjectArray aBusObj =
     exampleBusObj.IgetBusinessObjectArrayAttribute(arrayAttribute);
```

IgetBusinessObjectAttribute()

単一カーディナリティーの属性の値を取得します。

構文

```
IBusinessObject IgetBusinessObjectAttribute(string attributeName);
```

パラメーター

attributeName
値を取得する単一カーディナリティー属性の名前。

戻り値

単一カーディナリティー属性の値が格納されている IBusinessObject オブジェクト。

例外

IAttributeNotSetException
: 属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
: 属性が単一カーディナリティー属性でない場合 (他の日付型の場合) にスローされます。

IAttributeBlankException
: 属性にブランク値がある場合にスローされます。

例

```
// Call the get business object method and get the attribute
String busObjAttribute = "Customer";
IBusinessObject aBusObj =
     exampleBusObj.IgetBusinessObjectAttribute(busObjAttribute);
```

IgetDateAttribute()

日付属性の値を取得します。

構文

```
string IgetDateAttribute(string attributeName);
```

パラメーター

attributeName　値を取得する日付カーディナリティー属性の名前。

戻り値

日付属性の値が格納されている string。

例外

IAttributeNotSetException
: 属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
: 属性が日付型でない場合にスローされます。

IAttributeBlankException
: 属性にブランク値がある場合にスローされます。

例

```
//call the Date method and get the attribute
String dateAttributeName = "DateOfBirth";
String aDate;
aDate = exampleBusObj.IgetDateAttribute(dateAttributeName);
```

IgetDefaultValue()

属性のデフォルト値を取得します。

構文

```
string IgetDefaultValue(string attributeName);
```

パラメーター

attributeName　デフォルト値を取得する属性の名前。

戻り値

属性のデフォルト値が格納されている string。

例外

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IValueNotSetException
: 属性にデフォルト値がある場合にスローされます。

例

```
// Call the default value method
String attributeName = "Name";
String defaultAttributeValue;
defaultAttributeValue =
     exampleBusObj.IgetDefaultValue (attributeName);
```

IgetDoubleAttribute()

属性の double 値を取得します。

構文

```
double IgetDoubleAttribute(string attributeName);
```

パラメーター

attributeName　double 値を取得する属性の名前。

戻り値

属性の double 値。

例外

IAttributeNotSetException
: 属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
: 属性が double 型でない場合にスローされます。

IAttributeBlankException
: 属性にブランク値がある場合にスローされます。

例

```
// Call the double method and get the attribute
double doubleValue = 0;
String doubleAttributeName = "Average";
doubleValue = exampleBusObj.IgetDoubleAttribute(doubleAttributeName);
```

IgetFloatAttribute()

属性の float 値を取得します。

構文

```
float IgetFloatAttribute(string attributeName);
```

パラメーター

attributeName　float 値を取得する属性の名前。

戻り値

属性の float 値。

例外

IAttributeNotSetException
: 属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
: 属性が float 型でない場合にスローされます。

IAttributeBlankException
: 属性にブランク値がある場合にスローされます。

例

```
// Call the Float method and get the attribute
float floatValue = 0.0;
String floatAttributeName = "Height";
floatValue = exampleBusObj.IgetFloatAttribute(floatAttributeName);
```

IgetICSVersion()

IBM WebSphere Business Integration Server Express and Express Plus フレームワークのバージョン番号を取得します。

構文

```
public String IgetICSVersion();
```

パラメーター

入力パラメーターなし

戻り値

IBM WebSphere Business Integration Server Express and Express Plus フレームワークのバージョン番号を戻します。

例外

このメソッドは例外をスローしません。

IgetIntAttribute()

属性の `int` 値を取得します。

構文

```
long IgetIntAttribute(string attributeName);
```

パラメーター

attributeName 整数値を取得する属性の名前。

戻り値

属性の整数値を保持する `long` 値。

例外

`IAttributeNotSetException`
属性値が設定されていない場合にスローされます。

`IInvalidAttributeNameException`
属性名が無効な場合にスローされます。

`IInvalidAttributeTypeException`
属性が整数型でない場合にスローされます。

`IAttributeBlankException`
属性にブランク値がある場合にスローされます。

例

```
// Call the int method and get the attribute
int intValue = 1;
String intAttributeName = "priority";
intValue = exampleBusObj.IgetIntAttribute(intAttributeName);
```

IgetLongTextAttribute()

属性の longtext 値を取得します。

構文

```
string IgetLongTextAttribute(string attributeName);
```

パラメーター

attributeName　longtext 値を取得する属性の名前。

戻り値

ストリングとしての属性の longtext 値。

例外

IAttributeNotSetException
: 属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
: 属性が longtext 型でない場合にスローされます。

IAttributeBlankException
: 属性にブランク値がある場合にスローされます。

例

```
// Call the LongText method and get the attribute
long longValue = "net30";
String longAttributeName = "Customer";
longValue = exampleBusObj.IgetLongTextAttribute(longAttributeName);
```

IgetName()

ビジネス・オブジェクト定義の名前を取得します。

構文

```
string IgetName();
```

パラメーター

なし。

戻り値

ビジネス・オブジェクト定義の名前が格納されている string。

例

```
// Get the name of the business object definition
String busObjName;
busObjName = exampleBusObj.IgetName();
```

IgetStringAttribute()

属性の string 値を取得します。

構文

```
string IgetStringAttribute(string attributeName);
```

パラメーター

attributeName　string 値を取得する属性の名前。

戻り値

属性の値が格納されている string。

例外

IAttributeNotSetException
: 属性値が設定されていない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

IInvalidAttributeTypeException
: 属性が string 型でない場合にスローされます。

IAttributeBlankException
: 属性にブランク値がある場合にスローされます。

例

```
// Call the String method and get the attribute
String stringValue = "declined";
String stringAttributeName = "SalesOrder";
stringValue = exampleBusObj.IgetStringAttribute(stringAttributeName);
```

IgetVerb()

ビジネス・オブジェクト用の動詞を取得します。

構文

```
string IgetVerb();
```

パラメーター

なし。

戻り値

ビジネス・オブジェクトの動詞が格納されている string (null も可)。

例外

IVerbNotSetException
動詞が設定されていない場合にスローされます。

例

```
// Get the verb of the business object.
String busObjName;
busObjName = exampleBusObj.IgetVerb();
```

IisAttributeMultipleCardinality()

属性に複数カーディナリティーがあるかどうかを判別します。

構文

```
boolean IisAttributeMultipleCardinality(string attributeName);
```

パラメーター

attributeName　カーディナリティーが判別される属性の名前。

戻り値

属性に複数カーディナリティーがある場合には true を戻します。それ以外の場合には、false を戻します。

例外

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
// Call the multiple cardinality method.
boolean multCard = false;
String busAttribute = "AttributeName";
multCard =
     exampleBusObj.IisAttributeMultipleCardinality(busAttribute);
if (multCard)
     System.out.println ("attribute is multiple cardinality");
else
     System.out.println ("attribute is not multiple cardinality");
```

IisBlankValue()

属性値がブランク値であるかどうかを判別します。

構文

```
boolean IisBlankValue(string attributeName);
```

パラメーター

attributeName　属性値がブランク値であるかどうか検査される属性の名前。

戻り値

属性値がブランク値である場合には `true` を戻します。それ以外の場合には、`false` を戻します。

例外

`IInvalidAttributeNameException`
属性名が無効な場合にスローされます。

例

```
// See if attribute is blank
boolean isBlank = false;
String busAttribute = "AttributeName";
isBlank = exampleBusObj.IisBlankValue(busAttribute);
if (isBlank)
     ...
```

IisIgnoreValue()

属性値が「ignore」であるかどうかを判別します。

構文

```
boolean IisIgnoreValue(string attributeName);
```

パラメーター

attributeName　値が「ignore」であるかどうか検査される属性の名前。

戻り値

属性値が「ignore」である場合には `true` を戻します。それ以外の場合には、`false` を戻します。

例外

`IInvalidAttributeNameException`
属性名が無効な場合にスローされます。

`IValueNotSetException`
属性にデフォルト値がある場合にスローされます。

例

```
// Call the attribute ignore method
boolean isIgnore = false;
String busAttribute = "AttributeName";
isIgnore = exampleBusObj.IisIgnoreValue(busAttribute);
if (isIgnore)
     ...
```

IisKey()

属性がキーであるかどうかを判別します。

構文

```
boolean IisKey(string attributeName);
```

パラメーター

attributeName　キーであるかどうか検査される属性の名前。

戻り値

メソッドは、属性がキーである場合には `true` を戻します。それ以外の場合には、`false` を戻します。

例外

`IInvalidAttributeNameException`
属性名が無効な場合にスローされます。

例

```
// See if attribute is key
boolean isKey = false;
String busAttribute = "AttributeName";
isKey = exampleBusObj.IisKey(busAttribute);
if (isKey)
     ...
```

IisRequired()

指定された属性が必要であるかどうかを判別します。

構文

```
boolean IisRequired(string attributeName);
```

パラメーター

attributeName　必要かどうか検査される属性の名前。

戻り値

属性が必要である場合には `true` を戻します。それ以外の場合には、`false` を戻します。

例外

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
// Call the isRequired method
boolean isReq = false;
String busAttribute = "AttributeName";
isReq = exampleBusObj.IisRequired (busAttribute);
if (isReq)
...
```

Iserialize()

IBM WebSphere Business Integration Server Express and Express Plus の直列化フォーマットを使用してビジネス・オブジェクト・データを直列化します。

構文

```
string Iserialize();
```

パラメーター

なし。

戻り値

ビジネス・オブジェクトの直列化データが格納されている string。

例

```
// Call the serialize data method
IBusinessObject srcBO =
     accessSession.IcreateBusinessObject("Customer");
...
String serializedCustomer = scrBO.Iserialize();
```

IsetAttributes()

指定された MIME タイプの直列化データからビジネス・オブジェクト内の属性を設定します。

構文

```
void IsetAttributes(string serializedData, string mimeType);
```

パラメーター

serializedData MIME タイプ・フォーマットの直列化データ。

mimeType 直列化データの外部フォーマットを識別する MIME タイプ。

戻り値

なし。

例外

IMalFormedDataException
データが適切にフォーマットされていない場合にスローされます。

例

```
// Establish data format type
String externalData = "incomingData"
String mimeType = "text/xml";
exampleBusObj.IsetAttributes (externalData, mimeType);
```

IsetAttributeToBlank()

ビジネス・オブジェクト内の属性をブランク値に設定します。

構文

```
void IsetAttributeToBlank(string attributeName);
```

パラメーター

attributeName　値がブランクに設定されている属性の名前。

戻り値

なし。

例外

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
// Call the set-attribute-to-blank method
String attributeName = "checkType";
exampleBusObj.IsetAttributeToBlank(attributeName);
```

IsetAttributeToIgnore()

ビジネス・オブジェクト内の属性を「ignore」に設定します。

構文

```
void IsetAttributeToIgnore(string attributeName);
```

パラメーター

attributeName　値が「ignore」に設定されている属性の名前。

戻り値

なし。

例外

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
// Set the Default Attribute to a CxIgnore value
String attributeName = "Ignore";
exampleBusObj.IsetAttributeToIgnore(attributeName);
```

IsetBooleanAttribute()

属性を boolean 値に設定します。

構文

```
void IsetBooleanAttribute(string attributeName, boolean value);
```

パラメーター

attributeName 値が設定されている属性の名前。

value 属性の boolean 値。

戻り値

なし。

例外

IInvalidAttributeTypeException
属性が boolean 型でない場合にスローされます。

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
// Call the Boolean method
String attributeName = "custID";
boolean value = false;
exampleBusObj.IsetBooleanAttribute(attributeName, false);
```

IsetBusinessObjectArrayAttribute()

ビジネス・オブジェクト配列 (複数カーディナリティー) である属性の値を設定します。

構文

```
void IsetBusinessObjectArrayAttribute(string attributeName,
     IBusinessObjectArray value);
```

パラメーター

attributeName 値が設定されている複数カーディナリティー属性の名前。

value 属性の値であるビジネス・オブジェクト配列。

戻り値

なし。

例外

IInvalidAttributeTypeException
属性がビジネス・オブジェクト配列でない場合にスローされます。

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
// Call the BusinessObjectArray attribute method
String arrayAttribute = "CustomerAddress";
IBusinessObject CustomerAddress =
     accessSession.IcreateBusinessObjectArray ("Address");
IBusinessObject exampleBO =
     accessSession.IcreateBusinessObject ("Customer");
exampleBO.IsetBusinessObjectArrayAttribute(arrayAttribute,
     CustomerAddress);
```

IsetBusinessObjectAttribute()

単一カーディナリティーの属性の値を設定します。

構文

```
void IsetBusinessObjectAttribute(string attributeName,
     IBusinessObject value);
```

パラメーター

attributeName 値が設定されている単一カーディナリティー属性の名前。

value 属性の値であるビジネス・オブジェクト。

戻り値

なし。

例外

IInvalidAttributeTypeException
属性がビジネス・オブジェクトでない場合にスローされます。

IInvalidAttributeNameException
属性名が無効な場合にスローされます。

例

```
// Call the BusinessObject attribute method
String attributeName = "AccountStatus";
String value = "delqnt";
exampleBusObj.IsetBusinessObjectAttribute(attributeName, value);
```

IsetDateAttribute()

属性を date 値に設定します。

構文

```
void IsetDateAttribute(string attributeName, string value);
```

パラメーター

attributeName 値が設定されている属性の名前。

value 属性の date 値 (ストリング・フォーマット)。

戻り値

なし。

例外

IInvalidAttributeTypeException
: 属性が日付でない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

例

```
// Call the set Date attribute method
String dateAttribute = "DateofBirth";
String dateValue = "11/18/1966";
exampleBusObj.IsetDateAttribute(dateAttribute, dateValue);
```

IsetDoubleAttribute()

属性を double 値に設定します。

構文

```
void IsetDoubleAttribute(string attributeName, double value);
```

パラメーター

attributeName 値が設定されている属性の名前。

value 属性の double 値。

戻り値

なし。

例外

IInvalidAttributeTypeException
: 属性が double 型でない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

例

```
// Call the double method
String doubleAttributeName = "Average";
double value = 5.75;
exampleBusObj.IsetDoubleAttribute(doubleAttributeName, value);
```

IsetFloatAttribute()

属性を float 値に設定します。

構文

```
void IsetFloatAttribute(string attributeName, float value);
```

パラメーター

attributeName
: 値が設定されている属性の名前。

value
: 属性の float 値。

戻り値

なし。

例外

IInvalidAttributeTypeException
: 属性が float 型でない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

例

```
// Call the Float method
String floatAttributeName "FloatAttributeName";
float value = 0.999;
exampleBusObj.IsetFloatAttribute(floatAttributeName, value);
```

IsetIntAttribute()

属性を int 値に設定します。

構文

```
void IsetIntAttribute(string attributeName, long value);
```

パラメーター

attributeName 値が設定されている属性の名前。

value 整数属性の `long` 値。

戻り値

なし。

例外

`IInvalidAttributeTypeException`
属性が整数型でない場合にスローされます。

`IInvalidAttributeNameException`
属性名が無効な場合にスローされます。

例

```
// Call the int method
String intAttribute = "CustomerNumber";
int value = 5002;
exampleBusObj.IsetIntAttribute(intAttribute, value);
```

IsetLongTextAttribute()

属性を `longtext` 値に設定します。

構文

```
void IsetLongTextAttribute(string attributeName, string value);
```

パラメーター

attributeName 値が設定されている属性の名前。

value 属性の値 (ストリング・フォーマット)。

戻り値

なし。

例外

`IInvalidAttributeTypeException`
属性が `longtext` 型でない場合にスローされます。

`IInvalidAttributeNameException`
属性名が無効な場合にスローされます。

例

```
// Call the LongText method
String longTextAttributeName = "Description";
String value = "A very long text"
exampleBusObj.IsetLongTextAttribute(longTextAttributeName, value);
```

IsetStringAttribute()

属性を string 値に設定します。

構文

```
void IsetStringAttribute(string attributeName, string value);
```

パラメーター

attributeName 値が設定されている属性の名前。

value 属性の string 値。

戻り値

なし。

例外

IInvalidAttributeTypeException
: 属性が string 型でない場合にスローされます。

IInvalidAttributeNameException
: 属性名が無効な場合にスローされます。

例

```
// Call the String method
String stringAttribute = "CustomerName";
String value = "Greatest Customer";
exampleBusObj.IsetStringAttribute(stringAttribute, value);
```

IsetVerb()

ビジネス・オブジェクト用の動詞を設定します。

構文

```
void IsetVerb(string verb);
```

パラメーター

verb ビジネス・オブジェクトの動詞。

戻り値

なし。

例外

IInvalidVerbException
: 動詞がビジネス・オブジェクトによってサポートされていない場合にスローされます。

例

```
// Set the verb
String verb = "Create";
exampleBusObj.IsetVerb(verb);
```

ItoExternalForm()

指定された MIME タイプの外部フォーマットにビジネス・オブジェクト・データを直列化します。

構文

```
string ItoExternalForm(string mimeType);
```

パラメーター

mimeType
: ビジネス・オブジェクトを変換する対象の (アクセス・クライアントの) MIME タイプ。

戻り値

ビジネス・オブジェクトの直列化バージョンが格納されている string (指定された MIME タイプ)。

例外

IMalFormedDataException
: 変換がエラーになる場合にスローされます。

注記

ItoExternalForm() メソッドは、データ・ハンドラーを起動して、直列化データの MIME タイプを渡します。データ・ハンドラーは、IBM WebSphere Business Integration Server Express and Express Plus ビジネス・オブジェクトを解析して、要求された MIME タイプの直列化データに変換して、この直列化データをアクセス・クライアントに戻します。直列化データの外部フォーマットは、IBM WebSphere Business Integration Server Express and Express Plus ソフトウェアまたはユーザー作成のカスタム・データ・ハンドラーがサポートするタイプである必要があります。詳しくは、「*データ・ハンドラー・ガイド*」を参照してください。

例

```
// Serialize data into html
String mimeType = "text/html");
String htmldata = exampleBusObj.ItoExternalForm(mimeType);
```

ItoString()

IBM WebSphere Business Integration Server Express and Express Plus ブローカーの直列化フォーマットを使用してビジネス・オブジェクトのダンプを戻します。

構文

```
string ItoString();
```

パラメーター

なし。

戻り値

IBM WebSphere Business Integration Server Express and Express Plus 互換フォーマットの直列化データが格納されている `string`。

例

```
// Convert to IBM format
String stringBusObj;
stringBusObj = exampleBusObj.ItoString();
```

第 9 章 IBusinessObjectArray インターフェース

IBusinessObjectArray インターフェースでは、ビジネス・オブジェクト、配列、配列属性を戻したり、または配列内の属性またはオブジェクトを設定したりするメソッドが提供されます。表 10 に、IBusinessObjectArray インターフェースのメソッドについて要約します。

表 10. IBusinessObjectArray インターフェースのメンバー・メソッド

メソッド	説明	ページ
Iduplicate()	ビジネス・オブジェクト配列のクローンを戻します。	93
IdeleteBusinessObjectAtIndex()	ビジネス・オブジェクト配列の、指定された指標にあるビジネス・オブジェクトを削除します。	94
IgetBusinessObjectAtIndex()	ビジネス・オブジェクト配列の任意の指標にあるビジネス・オブジェクトを取得します。	94
IgetSize()	ビジネス・オブジェクト配列のサイズを戻します。	95
IremoveAllElements()	ビジネス・オブジェクト配列内のすべての要素 (ビジネス・オブジェクト) を削除します。	95
IsetBusinessObject()	ビジネス・オブジェクト配列の末尾にあるビジネス・オブジェクトを設定します。	96
IsetBusinessObjectAtIndex()	ビジネス・オブジェクト配列の、指定された指標にあるビジネス・オブジェクトを設定します。	96

Iduplicate()

ビジネス・オブジェクト配列のクローンを戻します。

構文

```
IBusinessObjectArray Iduplicate();
```

パラメーター

なし。

戻り値

重複するビジネス・オブジェクト配列が格納されている IBusinessObjectArray オブジェクト。

例外

ICxAccessError ビジネス・オブジェクト配列にアクセスできない場合にスローされます。

例

次の例では、destCustomer を作成するために sourceCustomer を複写します。

```
IBusinessObjectArray srcBOArray =
     accessSession.IcreateBusinessObjectArray ("Customer");
IBusinessObjectArray destBOArray = scrBOArray.Iduplicate();
```

IdeleteBusinessObjectAtIndex()

ビジネス・オブジェクト配列の、指定された指標にあるビジネス・オブジェクトを削除します。

構文

```
void IdeleteBusinessObjectAtIndex(long index);
```

パラメーター

index 削除するビジネス・オブジェクトのビジネス・オブジェクト配列内の指標。

戻り値

なし。

例外

IInvalidIndexException
指標が無効な場合にスローされます。

例

```
//Delete the business object
long index = 5;
exampleBusObjArray.IdeleteBusinessObjectAtIndex(index);
```

IgetBusinessObjectAtIndex()

ビジネス・オブジェクト配列の任意の指標にあるビジネス・オブジェクトを取得します。

構文

```
IBusinessObject IgetBusinessObjectAtIndex(long index);
```

パラメーター

index 取得するビジネス・オブジェクトのビジネス・オブジェクト配列内の指標。

戻り値

ビジネス・オブジェクト配列の、指定された指標にあるビジネス・オブジェクトが格納されている IBusinessObject オブジェクト。

例外

IInvalidIndexException
: 指標が無効な場合にスローされます。

例

```
// call the get business object at index method
IBusinessObject aBusinessObject = null;
long index = 1;
aBusinessObject = exampleBusObjArray.IgetBusinessObjectAtIndex(index);
```

IgetSize()

ビジネス・オブジェクト配列のサイズを戻します。

構文

```
long IgetSize();
```

パラメーター

なし。

戻り値

ビジネス・オブジェクト配列内の要素 (ビジネス・オブジェクト) の数を示す整数。

例

```
// get the array size
long = arraySize = 0;
arraySize = exampleBusObjArray.IgetSize();
```

IremoveAllElements()

ビジネス・オブジェクト配列内のすべての要素 (ビジネス・オブジェクト) を削除します。

構文

```
void IremoveAllElements()
```

パラメーター

なし。

戻り値

なし。

例

```
// remove array elements
exampleBusObjArray.IremoveAllElements();
```

IsetBusinessObject()

ビジネス・オブジェクト配列の末尾にあるビジネス・オブジェクトを設定します。

構文

```
void IsetBusinessObject(IBusinessObject value);
```

パラメーター

value　配列の末尾で設定するビジネス・オブジェクト。

戻り値

なし。

例外

IInvalidBusinessObjectTypeException
　　ビジネス・オブジェクトがサポートされていない場合にスローされます。

例

```
// Set the business object at the end of the array
IBusinessObject srcBO = accessSession.IcreateBusinessObject(
     "PayableNetChange");
exampleBusObjArray.IsetBusinessObject(srcBO);
```

IsetBusinessObjectAtIndex()

ビジネス・オブジェクト配列の、指定された指標にあるビジネス・オブジェクトを設定します。

構文

```
void IsetBusinessObjectAtIndex(long index, IBusinessObject inObj);
```

パラメーター

index　ビジネス・オブジェクト配列内の指標。

inObj　配列内に格納されるビジネス・オブジェクト。

例外

IInvalidIndexException
　　指標が無効な場合にスローされます。

IInvalidBusinessObjectTypeException
: ビジネス・オブジェクト・タイプがビジネス・ビジネス配列によってサポートされていない場合にスローされます。

例

```
// Set the business object at the index
long index = 1;
IBusinessObject aBusObj = accessSession.IcreateBusinessObject(
     "PayableNetChange");
exampleBusObjArray.IsetBusinessObjectAtIndex(index, aBusObj);
```

第 10 章 サーバー・アクセス例外

本章では、サーバー・アクセス例外について説明します。サーバー・アクセスのメソッドによってスローされる例外は、次の例外クラスのサブクラスです。

```
org.omg.CORBA.UserException
```

注: この `UserException` クラスは外部クラスです。IBM Crosswords 例外クラスではありません。`UserException` のメンバーおよびメソッドについては、IBM Java ORB 資料を参照してください。

すべてのサーバー・アクセス例外には、`IerrorMessage` と呼ばれる `string` エラー・メッセージ・メンバーが格納されています。

表 11 に、サーバー・アクセスの例外について要約します。

表 11. 例外の要約

例外	ページ
IAttributeBlankException	99
IAttributeNotSetException	99
ICxAccessError	100
IExecuteCollaborationError	100
IInvalidAttributeNameException	100
IInvalidAttributeTypeException	101
IInvalidBusinessObjectTypeException	101
IInvalidIndexException	101
IInvalidVerbException	101
IMalFormedDataException	101
IValueNotSetException	101
IVerbNotSetException	101

IAttributeBlankException

この例外は、属性にブランク値が格納されている場合にスローされます。

メンバー

```
string IerrorMessage;
```

IAttributeNotSetException

この例外は、属性に値が格納されていない場合にスローされます。

メンバー

```
string IerrorMessage;
```

ICxAccessError

この例外は、オブジェクトがアクセスできない場合にスローされます。

メンバー

```
string IerrorMessage;
```

IExecuteCollaborationError

この例外は、コラボレーションの実行が失敗する場合にスローされます。

メンバー

```
string IerrorMessage;
long status;
```

注記

コラボレーションの実行を要求する、次の 2 つのメソッドでは、IExecuteCollaborationError 例外がスローされる場合があります。

- IexecuteCollaboration()
- IexecuteCollaborationExtFmt()

この例外には、例外発生時の詳細を示す、status という名前のパブリック int 変数が格納されています。サーバー・アクセスでは、この status 変数の可能な値を表す実行状況定数が提供されます。この例外の実行状況定数を表 12 にリストします。

表 12. IExecuteCollaborationError Status の値

定数名	説明
UNKNOWNSTATUS	IexecuteCollaboration() または IexecuteCollaborationExtFmt() メソッドの呼び出しの状況。
FAILEDTOREACHCOLLABORATION	アクセス要求は、コラボレーションに届きませんでした。
FAILEDINEXECUTIONOFCOLLABORATION	アクセス要求は、コラボレーションの実行中に失敗しました。
FAILEDINRETURNTOCLIENT	コラボレーションが実行されましたが、アクセス・クライアントへの応答の配送時にエラーが発生しました。

この値を取得するには、次に示すように、例外変数を間接参照します。

this_exception_name_caught.status

IInvalidAttributeNameException

この例外は、属性名が無効な場合にスローされます。

メンバー

```
string IerrorMessage;
```

IInvalidAttributeTypeException

この例外は、属性タイプが無効な場合にスローされます。

メンバー

```
string IerrorMessage;
```

IInvalidBusinessObjectTypeException

この例外は、ビジネス・オブジェクト・タイプがコンテナーに一致しない場合にスローされます。

メンバー

```
string IerrorMessage;
```

IInvalidIndexException

この例外は、指標が無効な場合にスローされます。

メンバー

```
string IerrorMessage;
```

IInvalidVerbException

この例外は、動詞が無効な場合にスローされます。

メンバー

```
string IerrorMessage;
```

IMalFormedDataException

この例外は、データのフォーマットが誤っている場合にスローされます。

メンバー

```
string IerrorMessage;
```

IValueNotSetException

この例外は、属性にデフォルト値がない場合にスローされます。

メンバー

```
string IerrorMessage;
```

IVerbNotSetException

この例外は、動詞が設定されていない場合にスローされます。

メンバー

```
string IerrorMessage;
```

第 4 部 付録

付録. 国際化対応に関する考慮事項

国際化対応アクセス・クライアントとは、特定のロケールに対してカスタマイズできるように作成されたアクセス・クライアントのことです。エンド・ユーザーの特定の国、言語、または地域に固有のデータの処理方法に関する情報を集める、ユーザー環境の一部です。

本セクションでは、国際化対応アクセス・クライアントについて、次の情報が提供されます。

- 『ロケールとは』
- 『国際化対応アクセス・クライアントの設計』

ロケールとは

ロケールとは、エンド・ユーザーの特定の国、言語、または地域に固有のデータの処理方法に関する情報を集める、ユーザー環境の一部です。ロケールは通常、オペレーティング・システムの一部としてインストールされます。

ロケールは、ユーザー環境に関して、次の情報を提供します。

- 言語および国別 (または地域) に応じた国別情報
  - データ・フォーマットは次のとおりです。

    日付: 日付の構造 (日付分離文字を含む) と共に、省略しない形と省略形の曜日および月の名前を定義します。

    数値: 3 桁ごとの区切りおよび小数点に使用する記号を、数値内にそれらの記号を挿入する位置と共に定義します。

    時刻: 12 時間制に使用する指標 (AM および PM 指標など) を、時刻の構造と共に定義します。

    通貨値: 数値および通貨記号を、通貨値内にそれらの記号を挿入する位置と共に定義します。
  - 照合順序では、特定の文字コード・セットおよび言語のデータの分類方法を示します。
  - ストリング処理には、文字の「大文字小文字」 (大文字と小文字) 比較、サブストリング、および連結などのタスクが含まれます。

国際化対応アクセス・クライアントの設計

国際化対応コンテキストでアクセス・クライアントを使用するには、ロケールおよびストリング・エンコード方式に関して考慮します。

ロケールに関する考慮事項

国際化対応のため、アクセス・クライアントはロケール依存にする必要があります。すなわち、アクセス・クライアントの振る舞いで、ロケール設定を考慮し、そのロケールに適合したタスクを実行する必要があります。

一般に、アクセス・クライアントは次のロケール依存設計方針に従う必要があります。

- エラー、状況、およびトレース・メッセージのテキストは、メッセージ・ファイルでアプリケーション固有のコンポーネントから分離させ、ロケールの言語に変換する。
- データの分類または照合では、ロケールの言語および国別に適合した照合を使用する。
- ストリング処理 (比較、サブストリング、および文字の大文字小文字) は、ロケールの言語の文字に適合させる。
- 日付、数値、時刻のフォーマットはロケールに適合させる。

文字エンコード方式

サーバー・アクセスは、Unicode 形式の UCS-2 を使用します。アクセス・クライアントがサーバー・アクセスに変換する日付は、Unicode 文字エンコード方式である必要があります。

索引

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

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[ラ行]

A

D

E

F

I

M

S

U

特記事項

特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものでは ありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

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

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

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。

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。

注: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

以下は、IBM Corporation の商標です。

IBM
IBM ロゴ
AIX
CrossWorlds
DB2
DB2 Universal Database
Lotus
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

MMX 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。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

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。

この製品には、Eclipse Project (http://www.eclipse.org) により開発されたソフトウェアが含まれています。

IBM WebSphere Business Integration Server Express V4.3.1、IBM WebSphere Business Integration Server Express Plus V4.3.1