IBM WebSphere Business Integration Adapters

Adapter for i2 ユーザーズ・ガイド

バージョン 1.3.x

お願い

本書および本書で紹介する製品をご使用になる前に、89 ページの『付録 C. 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 Adapter for i2 (5724-G91) バージョン 1.3.x に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter for i2 User Guide
Version 1.3.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.7

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

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

目次

本書について

IBM(R) WebSphere(R) Business Integration Adapter ポートフォリオは、主要な e-business テクノロジー、エンタープライズ・アプリケーション、レガシー、およびメインフレーム・システムに統合コネクティビティーを提供します。製品セットには、ビジネス・プロセスの統合に向けてコンポーネントをカスタマイズ、作成、および管理するためのツールとテンプレートが含まれています。

本書では、IBM WebSphere Business Integration Adapter for i2 のインストール、構成、ビジネス・オブジェクト開発、およびトラブルシューティングについて説明します。

対象読者

本書は、お客様のサイトでコネクターを使用するコンサルタント、開発者、およびシステム管理者を対象にしています。

関連文書

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

以下のサイトから、関連資料をインストールすることができます。

- アダプターの一般情報が必要な場合、アダプターを WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) とともに使用する場合、およびアダプターを WebSphere Application Server とともに使用する場合は、以下のサイトを参照してください。

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

- WebSphere InterChange Server とともにアダプターを統合ブローカーとして使用する場合、以下のサイトを参照してください。

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

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

- Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker) の詳細については、以下のサイトを参照してください。

  http://www.ibm.com/software/integration/mqfamily/library/manualsa/

- WebSphere Application Server の詳細については、以下のサイトを参照してください。

  http://www.ibm.com/software/webservers/appserv/library.html

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

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

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

表記上の規則

本書では、以下の規則を使用します。

ProductDir	製品のインストール先ディレクトリーを表します。
`Courier フォント`	コマンド名、ファイル名、ユーザーが入力する情報、システムが画面上に出力する情報などのリテラル値を示します。
イタリック、*イタリック*	初出語、変数名、または相互参照を示します。
青のアウトライン	オンラインで表示したときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
{ }	構文行では、中括弧によって囲まれた複数のオプションから、1 つのオプションだけを選択する必要があります。
|	構文行では、パイプ文字によって区切られた複数のオプションから、1 つのオプションだけを選択する必要があります。
[ ]	構文行では、大括弧によってオプショナル・パラメーターが囲まれます。
...	構文行では、省略符号は直前のパラメーターの繰り返しを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って入力できることを示します。
< >	1 つの名前の個々の要素を互いに区別するために、不等号括弧によって個々の要素が囲まれます。例えば、`<server_name><`*`connector_name`*`>tmp.log` のように使用します。
/、¥	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。UNIX のインストールでは、円記号の代わりにスラッシュ (/) を使用します。すべての製品パス名は、システム上の、i2 用のコネクターがインストールされているディレクトリーを基準とした相対パス名です。
%text% および *$text*	% 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、UNIX 環境変数 *text* の値を示します。

本リリースの新機能

このセクションでは、本書で取り上げる IBM WebSphere Business Integration Adapter for i2 の新機能および変更された機能について説明します。

リリース 1.3.x の新機能

- Solaris 7.0 はもはやサポートされていません。
- コネクター定義ファイルのフォーマットは、.txt から .xsd に変更されました。
- イベント通知で共用される接続の登録抹消の処理が改善されました。
- 同じ接続 ID を使用して複数のサービス要求が行われる場合の接続パフォーマンスが改善されました。

リリース 1.2.x の新機能

- アダプターのインストール情報は、本書から移動しました。この情報の新たな入手先については、第 2 章を参照してください。
- i2 アダプターはグローバル化され、DBCS をサポートするようになりました。
- HP-UX がサポートされました。
- 複数のコネクター・インスタンスの作成がサポートされました。
- i2 ODA の新規構成プロパティーとして、MetadataService、PortTypesOperation、および SchemasOperation の 3 つが追加されました。
- 異なるサブネット上の CIS に対する i2 アダプターの実行がサポートされました。

リリース 1.1.x の新機能

- CrossWorlds という名前は、現在では、システム全体を表したり、コンポーネント名やツール名を修飾するためには使用されなくなりました。例えば、「CrossWorlds System Manager」は現在は「System Manager」で、「CrossWorlds Interchange Server」は現在は「WebSphere Interchange Server」です。これに合わせて、本書でも名前が変更されています。
- 統合ブローカーとしての WebSphere Application Server (WAS) の使用がサポートされました。
- コネクター構成プロパティーの Application Username および Password を使用したグローバル・レベルでの認証と、MO_instance オブジェクト属性 username および password を使用したビジネス・オブジェクト・レベルでの認証の両方がサポートされました。
- ステートフル操作がサポートされました。これは、ステートフル操作要求ビジネス・オブジェクトを使用するもので、これにより CIS エージェントは、セッション状態を維持することができます。
- 保証されたメッセージングとデリバリー、および永続クライアント接続がサポートされました。

- 指定の実行時間内に実行できなかった操作に関する対話の解決がサポートされました。
- サンプル・ポーリング・ビジネス・オブジェクトが拡張されました。
- i2 ODA の機能拡張によって以下の内容が更新されました。
  - 起動情報からの VisiBroker OAD の除去
  - 「タイプ」と「フォーマット」の区別
  - 「any」タグがあっても、ビジネス・オブジェクトの生成が可能
  - ODA ウィザードの画面ショット

第 1 章 コネクターの概要

この章では、IBM WebSphere Business Integration Adapter for i2 のコネクター・コンポーネント、および関連する Business Integration システム体系について説明します。

i2 コネクターは、i2 の Common Integration Services (CIS) API を介して、i2 アプリケーション・モジュールと統合します。i2 の CIS API には、JCA Common Client Interface が実装されています。i2 は、CIS をサポートする 1 組のアプリケーション・モジュールを持ちます。i2 コネクターはメタデータ主導型で、Object Discovery 機能があり、バージョン 6.0.1 SDK CIS に対応する任意の i2 アプリケーションへの統合が可能です。バージョン 5.2 以上の i2 モジュールの多くが、バージョン 6.0.1 CIS SDK をサポートしています。このコネクターは、Windows、Solaris、AIX および HP-UX で使用可能です。

コネクターは、*アプリケーション固有のコンポーネント* とコネクター・フレームワーク という 2 つの部分からなっています。アプリケーション固有のコンポーネントには、特定のアプリケーションまたはテクノロジー (この場合は i2) に合わせて調整されたコードが含まれています。コネクター・フレームワークは、統合ブローカー (WebSphere InterChange Server (ICS)、WebSphere MQ Integrator Broker (WMQI)、または WebSphere Application Server (WAS)) とアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはすべてのコネクターに共通です。コネクター・フレームワークによって、統合ブローカーとアプリケーション固有のコンポーネント間に以下のサービスが提供されています。

- ビジネス・オブジェクトの送受信
- 始動メッセージおよび管理メッセージのやり取りの管理

注: 本書には、コネクター・フレームワークとアプリケーション固有のコンポーネントの両方に関する情報が記載されています。本書ではこれらを両方ともコネクターと呼んでいます。

統合ブローカーとコネクターの関係の詳細については、「*システム管理ガイド*」(ICS の場合)、「*WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」、または「*アダプター実装ガイド (WebSphere Application Server)*」を参照してください。

この章には、以下のセクションが含まれています。

コネクター・アーキテクチャー

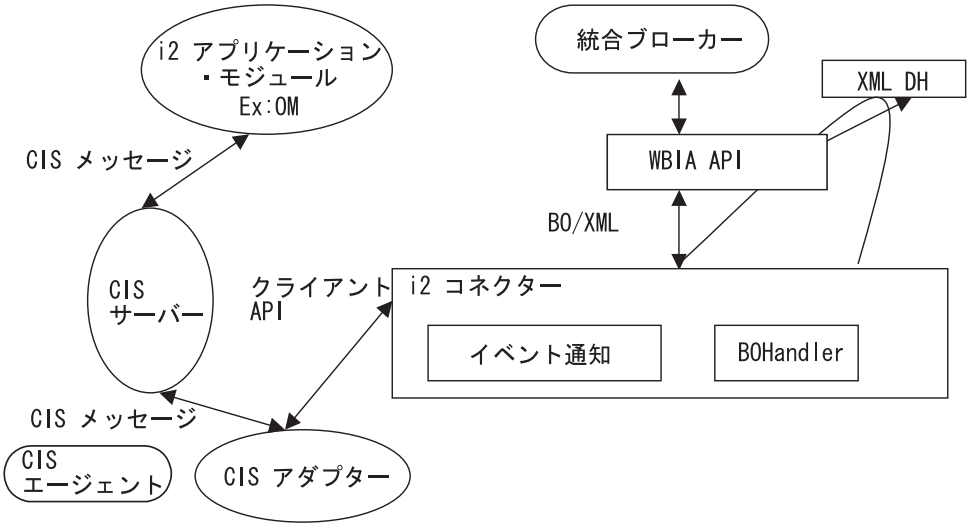
i2 の Common Integration Service (CIS) は、外部アプリケーションと i2 アプリケーション・モジュールの間の接続を可能にします。

CIS には、主に 3 つのコンポーネントがあります。

- CIS フロント・バス: アプリケーションはこのバスを使用して、使用可能な機能および予期される入出力データの詳細を定めた XML メタデータ・フォーマット・インターフェースを指定します。CIS スクリプトは、必要な入出力データ表記を XML スキーマおよび Java Beans に生成します。製品チームがこれらの機能を実装するには、標準 CIS インターフェースを実装して必要なロジックを実行するハンドラーを Java で記述します。このハンドラーは、データを XML または Java Beans として処理します。CIS インフラストラクチャーは、クライアントがこの機能をさまざまなリソースから呼び出せるように、これらのインターフェースを配置します。
- CIS バック・バス: アプリケーションはこのバスを使用して、データ転送用のバルク・インポート/エクスポート・インターフェースを作成します。
- CIS シングル・サインオン: Web アプリケーションが使用する Java インターフェースの標準セットで、中央認証ストアに対してユーザーを認証します。

i2 コネクターは、i2 が CIS アダプターとともに提供する CIS クライアント API を使用して、CIS フロント・バスと対話します。CIS クライアント API には、JCA Common Client Interface が実装されています。CIS アダプターは、さまざまなバインドを使用して、CIS メタデータ情報に基づいて動作します。

次の図は、i2 コネクターおよび i2 フレームワークのコンポーネントを示します。



次の表では、i2 コネクターおよび i2 フレームワークのコンポーネントに使用する用語について説明します。

コンポーネント	説明
CIS	i2 が提供する Common Integration Services。外部アプリケーションと i2 アプリケーション・モジュールの間の接続を可能にします。
OM	Order Management。i2 アプリケーション・モジュールの一例です。

コンポーネント	説明
CIS エージェント	中央サーバーで稼働する CIS サーバー・アプリケーション。クライアント・アプリケーションに関する接続情報を保守し、ソリューションに含まれるすべてのクライアント・アプリケーションを管理、モニターします。CIS アダプターおよび i2 アプリケーション・モジュールは CIS エージェントに登録されます。
JCA CCI	Java Connector Architecture の Common Client Interface。
CIS アダプター	i2 の CIS クライアント API。JCA-CCI が実装されています。
WBIA API	i2 コネクターが、指定された統合ブローカーとの通信に使用する API。
統合ブローカー	ビジネス・オブジェクト処理ロジックの実行を管理するプログラム。サポートされるブローカーは、InterChange Server (ICS)、WebSphere MQ Integrator Broker (WMQI)、および WebSphere Application Server (WAS) です。
XML DH	XML メッセージから IBM ビジネス・オブジェクトへの変換、またはその逆の変換に使用される IBM のデータ・ハンドラー (DH)。i2 コネクターともに使用するには、XML DH を構成する必要があります。
CIS サーバー	操作の呼び出しを処理する統合コンテナー。統合コンテナーと CIS サーバーは、本書では同じ意味で使用されます。

コネクターの動作方法

i2 コネクターは、CIS (Common Integration Services) クライアントになります。コネクターは、非管理環境で CIS クライアント API に接続します。つまり、アプリケーションのユーザー名とパスワードを使用してアプリケーション・サーバーに認証されるのではなく、コネクターが直接 CIS アダプターに接続します。柔軟性を持たせるため、ビジネス・オブジェクトに含まれる信任状情報を使用して、要求サービス処理のときに認証を行うことも可能です。詳細については、19 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』を参照してください。

現在、CIS クライアント API は、どの接続プール機構もサポートしてません。接続の数に制限はありません。i2 CIS アダプターは、i2 CIS エージェントの構成の設定次第で、保証されたメッセージングとデリバリー、および永続クライアント接続を実現できます。2 バイト文字が含まれるデータを、i2 コネクターで処理しても、整合性は保たれます。コネクターは、グループ 1 言語をサポートしています。

i2 コネクターは、双方向に機能します。ブローカーから i2 アプリケーションに送信される要求だけでなく、アプリケーションで発生するイベントも処理できます。

イベントのサブスクリプションの場合、i2 コネクターは i2 メタオブジェクト内の情報を使用します。これらのメタオブジェクトに指定されたタイプに関する操作を、CIS アダプターを介して i2 の CIS エージェントに登録します。CIS エージェントは、登録済みの操作を listen して、登録済み操作のイベント・メッセージのみを検出します。i2 コネクターは、ポーリング呼び出しを使って、これらのメッセージを取得します。

要求処理の場合、i2 コネクターは、統合ブローカーから受信した要求を処理するときに、着信ビジネス・オブジェクトを CIS レコードに変換し、適切な CIS クライアント API 呼び出しを使用して、i2 アプリケーション・モジュールの操作を実行します。i2 コネクターに対する要求処理はマルチスレッド化されており、多重要求を処理することができます。

i2 コネクターは、「*IBM Connector Developer's Guidelines*」で概説されているメタデータ設計方針に準拠しています。したがって、新規 IBM ビジネス・オブジェクトを定義するときは、i2 コネクターのコード・レベルで追加コーディングまたはカスタマイズをする必要はありません。詳細については、19 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』を参照してください。

サブスクリプションの処理

以降のセクションでは、コネクターがアプリケーション・イベントを処理する方法について説明します。

イベント検出および通知

本書の目的とするイベントは、i2 アプリケーション・モジュールからパブリッシュされる CIS メッセージです。i2 アプリケーションは、CIS エージェントに登録済みの操作に対応するモジュールで発生したすべてのイベントをコネクターに通知します。

通知を受ける操作の登録は、i2 コネクターが行います。

例: Bidding タイプ操作 addBid を i2 コネクターに通知することを希望する場合は、i2 コネクターが一度 addBid 操作を登録すれば、新規 Bidding が i2 アプリケーション・モジュールに追加されるたびにそのイベントが CIS サーバーのキューに入れられます。

i2 コネクターは、i2 メタオブジェクト内の情報を使用します。ポーリング呼び出しに対して、いくつかの操作を listen する意向を登録します。これによって CIS エージェントは、i2 コネクターが登録済み操作の出力を検査するつもりであることを認識します。

状況の更新

i2 アプリケーションに対して状況の更新は行われません。一般に、イベント状況 (例: SUCCESS、FAIL、UNSUBSCRIBED) はアプリケーションのイベント・ストアに書き込まれます。i2 ではイベント・ストアは保守されないので、状況更新ストラテジーは i2 コネクターには関係しません。エラー・メッセージがある場合、そのメッセージは i2 アダプター・ログ・ファイルに記録されます。詳細については、43 ページの『第 5 章 トラブルシューティングとエラー処理』を参照してください。

イベント取得

i2 コネクターでは、ポーリングは単一スレッド化されています。コネクターは、i2 メタオブジェクトを使用して、通知を受ける操作を CIS エージェントがポーリングするように登録します。これらのメタオブジェクトは、名前に i2MO プレフィックスが付いていて、操作に関する情報と、その指定された操作およびタイプに対応する IBM ラッパー・ビジネス・オブジェクト名を保管します。メタオブジェクトの属性は、静的デフォルト値として指定されます。デフォルト値は属性プロパティー

であり、ビジネス・オブジェクト設計時に設定できます。ラッパー・ビジネス・オブジェクト構造および属性プロパティーの詳細については、19 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』および 31 ページの『第 4 章 i2 ODA によるビジネス・オブジェクトの生成』を参照してください。

サブスクリプション・メッセージの取得に伴うステップは、以下のとおりです。

1. i2 コネクターは、コネクター・プロパティー Application Username および Password か、あるいはポーリング・メタオブジェクトにある Username および Password を使って認証された接続を使用します。
2. i2 コネクターは、i2MO メタオブジェクトの情報を読み取ってから、CIS エージェントにその操作を登録します。メタオブジェクト内の情報は、最初のポーリング呼び出しの際に i2 コネクターによってキャッシュされます。コネクターは、メタオブジェクト内に指定された Username、Password、InstanceId、および ConnectionId を使用して、このメタオブジェクトの接続を割り振ります。
3. 各ポーリング呼び出しは、コネクター・プロパティー PollFrequency に基づいて、統合ブローカーから発行されます。最初のポーリング呼び出しでなんらかの登録エラーが発生した場合、i2 コネクターはその後のポーリング呼び出しで同じ操作を登録しようとします。

   i2 コネクターが持続接続を使用して通知を受けるデータを登録すると、i2 コネクターが稼働していないときに送信されたデータでも確実に取得できるようになります。EventSubscriptionConnectionID プロパティー値、またはポーリング・メタオブジェクトの ConnectionId 属性を使用することにより、コネクターを再始動してもサブスクリプションとの関連を保持できます。

   永続ストレージを使用した CIS エージェントの構成方法については、i2 の CIS インフラストラクチャー・ガイドを参照してください。
   EventSubscriptionConnectionID プロパティーの構成方法については、9 ページの『第 2 章 コネクターのインストールと構成』を参照してください。
4. すべてのポーリング呼び出しで、i2 コネクターは、CIS エージェントに登録した操作の出力を検査します。これらの操作のいずれかから出力がある場合、コネクターはその出力を CIS レコードの形式で取得します。i2 コネクターは、各登録済み操作のポーリング呼び出しのたびに、PollQuantity (コネクター・プロパティー) の数のメッセージを取得します。

   例: PollQuantity が 5 に設定されていて、登録済みの操作が 5 つある場合、ポーリング呼び出しのたびに出力を 25 回検査します。PollQuantity が設定されていない場合は、各操作のポーリング呼び出しのたびにデフォルトとして 1 つのメッセージが取得されます。
5. 取得された XML メッセージは、ビジネス・オブジェクトに変換されます。ビジネス・オブジェクトは、操作のラッパー・ビジネス・オブジェクトの子属性として設定されます。この出力が取得された InstanceId は、ラッパーのメタオブジェクト属性の InstanceId として設定されます。詳細については、19 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』を参照してください。
6. コネクターは、後続の処理のためにラッパー・ビジネス・オブジェクトを統合ブローカーに送信します。

動詞 (操作) の処理

操作は動詞に相当する i2 用語で、ポートごとに i2 が提供する XML 構造で定義されます。詳細については、19 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』、および 31 ページの『第 4 章 i2 ODA によるビジネス・オブジェクトの生成』を参照してください。

サービス呼び出し要求の処理

i2 コネクターがアプリケーションで操作を実行するためのサービス呼び出し要求を統合ブローカーから受信するとき、その要求はラッパー・ビジネス・オブジェクトの形式をとります。ラッパー・ビジネス・オブジェクトにはメタオブジェクト、操作または動詞、その操作の入出力タイプのいずれかまたは両方、または操作のフォーマットが含まれます。メタオブジェクト (MO_Instance) および入出力ビジネス・オブジェクトは、ラッパー・ビジネス・オブジェクトの子ビジネス・オブジェクトです。MO_Instance オブジェクトに含まれているのは、InstanceId、ConnectionId、Username、および Password の属性値です。Username および Password は、要求処理時の認証に使用されます。ラッパー・ビジネス・オブジェクトの動詞は、指定されたインスタンスに対して有効な操作にする必要があります。

子ビジネス・オブジェクトが入力タイプか出力タイプかという情報は、ラッパー・ビジネス・オブジェクトの属性のアプリケーション固有の情報 (ASI) から取得されます。

例: `ASI Type=input` は、子ビジネス・オブジェクトが入力タイプであることを示します。

入力子ビジネス・オブジェクトは、最初に XML データ・ハンドラーによって XML メッセージに変換されます。次に、このビジネス・オブジェクトは CIS ユーティリティーによって CIS レコードに変換されます。そして、CIS クライアント API を使用して、操作が実行されます。

この操作によってなんらかの出力 XML メッセージが送信される場合、そのメッセージは出力子ビジネス・オブジェクトに変換されます。そして、ラッパー・ビジネス・オブジェクト内の出力子ビジネス・オブジェクトには、該当する値が取り込まれます。

i2 CIS エージェントがパーシスタンスを使用可能な状態で稼働している場合、CIS エージェントはステートフル操作を使用して、セッション状態を維持することができます。コネクターは、指定された実行時間内に実行できなかった操作の結果を取得し、それらの操作を解決しようと試みます。そして、その結果を統合ブローカーに戻します。

対話の解決が試行される回数を制御するには、InteractionResolutionAttempts プロパティーの値を設定します。InteractionResolutionAttempts の値をゼロ (0) に設定すると、コネクターは対話の解決を試行しません。対話の解決は、CIS アダプター状態が INDOUBT の対話についてのみ試行されます。

ステートフル操作の詳細については、23 ページの『ステートフル操作のメタオブジェクトの使用』を参照してください。InteractionResolutionAttempts および

ExecutionTimeout プロパティーの構成方法については、9 ページの『第 2 章 コネクターのインストールと構成』を参照してください。

状況の更新

処理中に発生するエラー状態は、詳細なエラー・メッセージとしてアダプター・ログに記録されます。

ReturnStatusDescriptor: コネクターは、サービス呼び出し要求の処理中に発生した最新のエラーのメッセージや状況を、ReturnStatusDescriptor という構造体に取り込みます。ReturnStatusDescriptor にアクセスすれば、サービス呼び出し要求中に発生したエラーの原因がわかります。アダプター・フレームワークは、必要に応じて、この構造体を伝搬します。

CIS アダプターとの持続接続を使用する場合の状況値には CREATED、STARTED、UNDELIVERABLE、DELIVERED、SUCCESS、INDOUBT、FAILED があり、これらは要求処理がエラーになったときの対話の状態を示します。

第 2 章 コネクターのインストールと構成

この章では、IBM WebSphere Business Integration Adapter for i2 のコネクター・コンポーネントをインストールし構成する方法、およびコネクターと連動するようにアプリケーションを構成する方法について説明します。本章の内容は、次のとおりです。

アダプター環境

アダプターをインストール、構成、使用する前に、環境要件を理解しておく必要があります。

ブローカーの互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for i2 バージョン 1.3.x は、以下のアダプター・フレームワークと統合ブローカーでサポートされます。

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

例外については、『*リリース情報*』を参照してください。

統合ブローカーのインストール手順およびその前提条件については、次の資料を参照してください。

- WebSphere InterChange Server (ICS) については、「*システム・インストール・ガイド (UNIX 版)*」または「*システム・インストール・ガイド (Windows 版)*」を参照してください。
- Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、および WebSphere Business Integration Message Broker) の場合は、「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」およびそれぞれの Message Brokers のインストールに関する資料を参照してください。一部の資料は次の Web サイトにあります。

http://www.ibm.com/software/integration/mqfamily/library/manualsa/

- WebSphere Application Server については、「*アダプター実装ガイド (WebSphere Application Server)*」および次の資料を参照してください。

  http://www.ibm.com/software/webservers/appserv/library.html

必要なハードウェアおよびソフトウェア

アダプターには以下のハードウェアおよびソフトウェアが必要です。

- アプリケーション:
  - CIS SDK 6.0.1
  - J2EE.jar
  - 適切な CIS アダプター (i2 ODA には MetadataService アダプターが必要です)
- 以下のアプリケーション・プラットフォームのいずれか:
  - Windows: 2000

    注: バージョン 1.2.x から、Adapter for i2 は Microsoft Windows NT 上ではサポートされなくなりました。
  - UNIX: Solaris、AIX、または HP
- 以下のアダプター・プラットフォームのいずれか:
  - Windows: 2000
  - UNIX: Solaris 8.0、AIX 5.1 または 5.2、または HP-UX 11i

ロケール依存データの処理

コネクターは国際化されています。2 バイト文字セット (DBCS) を、同じく 2 バイト文字セットをサポートするインターフェースに送信することができ、また指定された言語のメッセージ・テキストの送信をサポートします。コネクターは、1 文字のコードを使用する地域から異なるコード・セットを使用する地域にデータを転送する場合、文字変換を実行し、データの意味を維持します。

Java 仮想マシン (JVM) 内の Java ランタイム環境は、Unicode 文字コード・セットでデータを表現します。Unicode は周知の文字コード・セット (単一バイトとマルチバイトの両方) で文字をエンコードします。WebSphere Business Integration システム内のほとんどのコンポーネントは、Java で書かれています。そのため、統合コンポーネント間でデータを転送する際に、ほとんどの場合文字変換は必要ありません。

アダプターと関連ファイルのインストール

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

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

インストール済みファイルの構造

次の表は、コネクターが使用するファイル構造、およびインストールされるファイルを示します。

注:

1. 本書では、ディレクトリー・パスとして円記号 (¥) を使用しています。UNIX のインストールでは、円記号の代わりにスラッシュ (/) を使用します。
2. すべての製品パス名は、システム上の、製品のインストール先ディレクトリーを基準とした相対パス名です。
3. Windows の場合、インストーラーは「IBM WebSphere Business Integration Adapters」メニューにコネクター・ファイルのアイコンを追加します。コネクターをすばやく始動するには、このファイルへのショートカットをデスクトップに作成してください。

ProductDir のサブディレクトリー	説明
connectors¥i2	コネクター・コードを保有するコネクター CWi2.jar ファイル、start_i2.bat ファイル (WIN) または start_i2.sh ファイル (UNIX) が含まれています。
connectors¥messages	エラー・メッセージ用の i2Adapter.txt ファイルが含まれています。
repository¥i2	CN_i2.txt ファイルが含まれています。
connectors¥i2¥Sample	ビジネス・オブジェクトを作成するためのサンプル・ファイルが含まれています。

コネクター・コンポーネントのインストールの詳細については、ご使用の統合ブローカーに応じて、以下のいずれかのガイドを参照してください。

- ご使用のプラットフォーム用の「システム・インストール・ガイド」(InterChange Server を統合ブローカーとして使用している場合)
- 「*WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」(WebSphere MQ Integrator を統合ブローカーとして使用している場合)
- 「*アダプター実装ガイド (WebSphere Application Server)*」(WebSphere Application Server を統合ブローカーとして使用している場合)

コネクターの構成

コネクターの構成プロパティーには、標準構成プロパティーとコネクター固有の構成プロパティーの 2 種類があります。コネクターを実行する前に、必ずこれらのプロパティーの値を設定してください。構成値を入力すると、その値はリポジトリーに保管されます。

コネクター・プロパティーを構成するには、次のいずれかのツールを使用します。

- Connector Designer: InterChange Server が統合ブローカーの場合

  ヒント: このツールには System Manager からアクセスします。
- Connector Configurator: WebSphere MQ Integrator が統合ブローカーの場合

  ヒント: このツールには「IBM WebSphere Business Integration Adapter」プログラ

ム・フォルダーからアクセスします。
Connector Configurator の詳細については、71 ページの『付録 B. Connector Configurator』を参照してください。

コネクターは、始動時に構成値を取得します。実行時セッション中に、1 つ以上のコネクター・プロパティーの値の変更が必要になることがあります。一部のコネクター構成プロパティー (AgentTraceLevel など) への変更は動的に反映されるので、即時に有効になります。その他のコネクター・プロパティーへの変更は静的変更のため、変更後にコンポーネントまたはシステムを再始動する必要があります。プロパティーが動的か静的かを判断するには、Connector Designer の「更新メソッド」列を参照してください。

標準コネクター・プロパティー

標準構成プロパティーには、コネクターが使用する情報が用意されています。これらのプロパティーの詳細については、51 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

注: i2 用コネクターは統合ブローカーとして InterChange Server、WebSphere MQ Integrator、および WebSphere Application Server をサポートしているため、このコネクターには、3 つのブローカーすべての構成プロパティーが関係します。

さらに、i2 の場合は、標準コネクター・プロパティーに関する以下の補足情報があります。

LogAtInterchangeEnd

InterChange Server (ICS) のログにエラーを記録するかどうかを示します。

デフォルト値は `false` です。

MessageFileName

エラー・メッセージ・ファイルが標準メッセージ・ロケーション *ProductDir*¥connectors¥messages にない場合の、エラー・メッセージ・ファイルのパスを示します。メッセージ・ファイル名が完全修飾パスでない場合、メッセージ・ファイルは、HOME 環境変数、または起動パラメーター `user.home` で指定されたディレクトリー内にあるとみなされます。コネクター・メッセージ・ファイルが存在しない場合は、WBIA API メッセージ・ファイルが使用されます。このファイルが存在しない場合は、`InterchangeSystem.txt` ファイルをメッセージ・ファイルとして使用します。

デフォルト値は `i2Adapter.txt` です。

コネクター固有のプロパティー

コネクター固有の構成プロパティーには、コネクターが実行時に必要とする情報が用意されています。これらのプロパティーは、コネクターを再コーディングおよび再ビルドせずに、コネクター内の静的情報またはロジックを変更する手段にもなっています。

次の表に、コネクターのコネクター固有構成プロパティーとその説明、および指定可能な値を示します。

プロパティー	説明	指定可能な値
ApplicationName	*コネクターごとに指定される固有の名前。*	i2 adapter
ApplicationUserName	*認証に使用される i2 接続のユーザー名。*	i2User
ApplicationPassword	*認証に指定される i2 接続のパスワード。*	i2Password
CISAgentHostName	*CIS エージェントがリモート・マシンで稼働するときに使用する名前。この名前が設定されていない場合は、現在のローカル・ホストで CIS エージェントが稼働しているとみなされます。名前が設定されている場合、i2 コネクターはこのリモート・ホストとの接続を確立します。*	ホスト名のストリング **例:** 任意のマシン名。California など。
ExecutionTimeout	i2 アプリケーションへの呼び出しを強制終了するまでの時間 (ミリ秒)。	デフォルトは 30000 です。
EventSubscriptionConnectionId	*要求処理に使用するデフォルトの connectionId。*	EventConn1
InteractionResolutionAttempts	*対話の解決が試行される回数。*	デフォルトは 2 です。
PollQuantity	*ポーリング時に、各登録済み操作についてクライアント・キューから検索されるメッセージの数。pollQuantity に登録済み操作数を掛けた値が、全体のメッセージ数になります。*	デフォルトは 1 です。
UseDefaults	*要求処理時に属性のデフォルト値を識別するためにコネクターが検査する値。このプロパティーは、i2 コネクターでは使用しません。*	このコネクターでは不要です。

start_i2.bat (Windows) または start_i2.sh (UNIX) の構成

CIS-SDK および j2ee.jar の始動ファイルへの適切なパスを追加する必要があります。

例: 以下のパス情報を start_i2.bat ファイルに追加する必要があります。これらは単なる例です。ローカル・インストールに応じてパス情報を変更してください。

```
set I2_CIS_HOME_DIR=C:¥i2¥CIS¥6.0.1¥cis-sdk
set J2EE_PATH=C:¥J2EE_JAR
set i2PROPERTIES="%i2_CIS_HOME_DIR%"¥properties;
```

(最後の行は、ご使用の i2 CIS の「properties」ディレクトリーの内容を参照します。)

注: 保証されたメッセージングを行うように CIS エージェントが構成され、アダプターが MQ バインドを使用する場合は、i2 CIS インストールと同じ相対パスに `.bindings` ファイルを置く必要があります。これは、MQ バインド構成要素の `jndiProviderURL` 属性です。

データ・ハンドラーの構成

データ・ハンドラーも構成する必要があります。MO_DataHandler_Default で、子ビジネス・オブジェクト text/xml に関する以下の値を設定してください。

Validation	false
ClassName	com.crossworlds.DataHandlers.text.xml
UseNewLine	false
InitialBufferSize	任意の適切な値。2097152 など。
DummyKey	1

注: 残りのフィールドはブランクのままにしてください。

データ・ハンドラー構成の詳細については、「*データ・ハンドラー・ガイド*」を参照してください。

複数のコネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

1. 初期コネクターに関連付けられているビジネス・オブジェクト定義を変更する必要がある場合は、適切なファイルをコピーし、Business Object Designer を使用

してそれらのファイルをインポートします。初期コネクターの任意のファイルをコピーできます。変更を加えた場合は、名前を変更してください。

2. 初期コネクターのファイルは、次のディレクトリーに入っていなければなりません。

   ProductDir¥repository¥initialConnectorInstance

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

2. この始動スクリプトを、14 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 1 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 1. コネクターの始動スクリプト

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

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

- Windows システムで**「スタート」**メニューから。

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

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

    connector_manager_*connName* -start

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

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

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

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

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

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

コネクターの停止

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

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

    ```
    connector_manager_connName -stop
    ```

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

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

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

第 3 章 コネクターのビジネス・オブジェクトについて

この章では、i2 ビジネス・オブジェクトの構造、コネクターによるビジネス・オブジェクトの処理方法、およびコネクターにおけるビジネス・オブジェクトの前提事項について説明します。ここで示す情報は、既存の i2 ビジネス・オブジェクトを変更するためのガイドとして使用することも、新規ビジネス・オブジェクト実装の提案として使用することもできます。

本章の内容は、次のとおりです。

IBM WebSphere Business Integration Adapter for i2 のビジネス・オブジェクトの作成を自動化する Object Discovery Agent (ODA) ユーティリティーについては、31 ページの『第 4 章 i2 ODA によるビジネス・オブジェクトの生成』を参照してください。

コネクター・メタデータの定義

i2 コネクターはメタデータ主導型です。WebSphere Business Integration システムで、メタデータとはビジネス・オブジェクトに保管されるアプリケーション固有の情報であり、コネクターがアプリケーションと対話する際に役立ちます。メタデータ主導型のコネクターによって処理される各ビジネス・オブジェクトは、コネクター内にハードコーディングされた命令ベースではなく、ビジネス・オブジェクト定義内にエンコードされたメタデータをベースにしてサポートされています。ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、属性プロパティーの設定、およびアプリケーション固有の情報コンテンツが含まれます。コネクターは、メタデータ主導型であるため、コネクターのコードを修正せずに、新規または変更済みのビジネス・オブジェクトを処理できます。

コネクターには、サポートするビジネス・オブジェクトの構造、親のビジネス・オブジェクトと子のビジネス・オブジェクトの間の関係、アプリケーション固有情報のフォーマットについての前提事項があります。

そのため、ビジネス・オブジェクトを作成または変更する場合は、その変更がコネクターの従うべきルールに適合している必要があります。そうでないと、コネクターは新規または変更されたビジネス・オブジェクトを正しく処理することができません。

ビジネス・オブジェクト構造の概要

WebSphere Business Integration システムでは、ビジネス・オブジェクト定義はタイプ名、サポートされる動詞、および属性からなっています。アプリケーション・ビジネス・オブジェクトは、ビジネス・オブジェクト定義のインスタンスです。これは、個々のアプリケーションのデータ構造および属性プロパティーを反映します。

一部の属性では、子ビジネス・オブジェクトまたは子ビジネス・オブジェクト配列を指すデータが格納されるのではなく、これらのオブジェクトに対応するデータが格納されます。親レコードと子レコード間のデータはキーにより関連付けられます。

WebSphere Business Integration Adapter にはフラットなビジネス・オブジェクトと階層のあるビジネス・オブジェクトがあります。フラット・ビジネス・オブジェクトには、単純属性、つまり、単一の値 (ストリングなど) を表す属性だけを格納できます。これは、子ビジネス・オブジェクトを指しません。階層ビジネス・オブジェクトには、単純属性と、子ビジネス・オブジェクトまたは値が含まれる子ビジネス・オブジェクト配列の両方が格納されます。

カーディナリティー 1 コンテナー・オブジェクト、または単一カーディナリティー関係は、親ビジネス・オブジェクトの属性が単一の子ビジネス・オブジェクトを含む場合に生じます。この場合、子ビジネス・オブジェクトは単一のレコードのみを含むことができるコレクションを表します。属性のタイプは、子ビジネス・オブジェクトのタイプと同じです。

カーディナリティー n コンテナー・オブジェクト、または複数カーディナリティー関係が生じるのは、親ビジネス・オブジェクトの属性に子ビジネス・オブジェクトの配列が含まれている場合です。この場合、子ビジネス・オブジェクトは複数のレコードを含むことができるコレクションを表します。属性のタイプは、子ビジネス・オブジェクトの配列のタイプと同じです。

階層ビジネス・オブジェクトは、単純属性を持つことができるだけでなく、単一カーディナリティーの子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す属性を持つこともできます。これらの各ビジネス・オブジェクトもまた、単一カーディナリティーの子ビジネス・オブジェクトおよびビジネス・オブジェクトの配列を含むことができます。それ以降の階層も同様です。

カーディナリティーの各タイプにおいて、親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は、子オブジェクトのキー属性のアプリケーション固有テキストで記述されます。

i2 ビジネス・オブジェクトの構造

i2 IBM ビジネス・オブジェクトは、i2 メッセージを IBM 固有の表現にしたものです。メッセージの各タイプには、対応する IBM ビジネス・オブジェクトがあります。

ビジネス・オブジェクトは、WebSphere Business Integration Adapter ユーティリティー XML ODA を使用して生成されます。このユーティリティーは、これらのタイプの XML スキーマ・ファイルを読み取って、対応する IBM ビジネス・オブジ

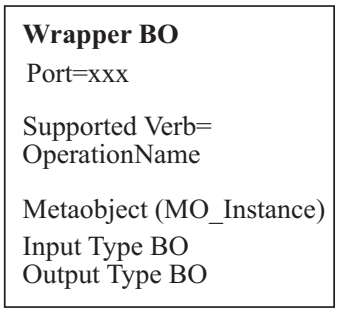
ェクトを生成します。(31 ページの『第 4 章 i2 ODA によるビジネス・オブジェクトの生成』、および「*IBM WebSphere Business Integration Adapters データ・ハンドラー・ガイド*」の第 3 章『XML データ・ハンドラー』を参照してください。)

i2 ビジネス・オブジェクトは、メタオブジェクト、操作、およびその操作の入出力データのいずれかまたは両方 (一方または両方があるか、あるいはどちらもない) のタイプまたはフォーマットをカプセル化したラッパー・ビジネス・オブジェクトです。トップレベルのオブジェクト・ラッパーには、メタオブジェクト (MO_Instance) の中に InstanceId、ConnectionId、Username、および Password の属性値があります。Username および Password は、要求処理時に別のユーザーで認証する場合に提供されます。

注: このビジネス・オブジェクト信任状情報は、グローバル・レベルでの認証で提供される Application Username および Password コネクター構成プロパティーを補うものです。メタオブジェクト (MO_Instance) レベルの値が空白のままである場合、アダプターは Connector Configurator レベルで検出されたグローバル信任状を使用します。

各操作に 1 つのラッパー・ビジネス・オブジェクトがあります。詳細については、31 ページの『第 4 章 i2 ODA によるビジネス・オブジェクトの生成』を参照してください。

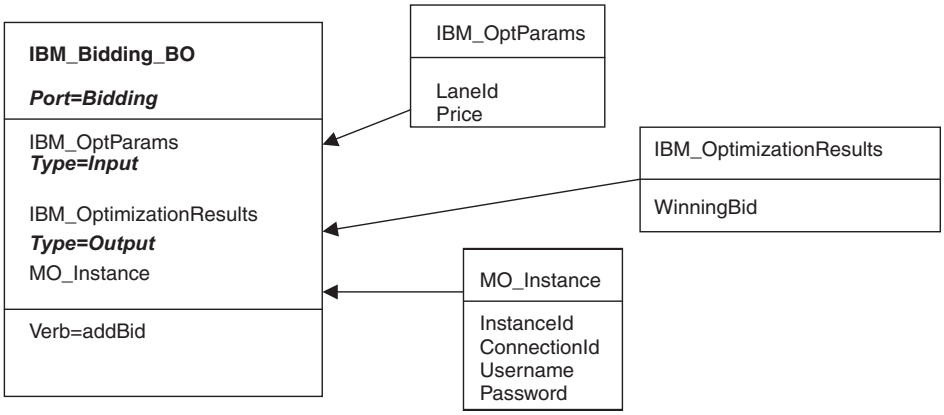
次の図は、ラッパー・ビジネス・オブジェクトのパーツを示します。図の後に、各パーツについての説明があります。



- ラッパー内に組み込まれた *metaobject* は、MO_Instance の構成に使用されます。MO_Instance には、属性 *InstanceId*、*ConnectionId*、*Username*、および *Password* の値が含まれます。詳細については、22 ページの『ポーリング用メタオブジェクトの構成』を参照してください。
- *operation* は、動詞としてラッパー・ビジネス・オブジェクトに設定され、ポートに関連付けられます。i2 には標準の動詞はありません。操作には入力と出力があります。複数の操作は、入出力タイプが同じでもサポートされるポートが異なる場合には、別々のポート用として 2 つの異なるラッパー・ビジネス・オブジェクトになります。
- *port* は、操作の i2 ポート・タイプの名前です。
- *type*、または *format* は、操作のデータ・タイプまたはフォーマットを表すビジネス・オブジェクト属性です。

次の図は、サンプル・ビジネス・オブジェクト IBM_Bidding_BO を示したものです。このオブジェクトには、3 つの子ビジネス・オブジェクトがあります。図の内容を以下に示します。

- IBM_OptParams および IBM_OptimizationResults は、XML ODA によって生成されたトップレベルのビジネス・オブジェクトを表します。
- ビジネス・オブジェクトのアプリケーション固有情報は、*Port* 属性 (Bidding) および *Type* 属性 (Input: IBM_OptParams および Output: IBM_OptimizationResults) にあります。
- 操作は addBid です。
- 子ビジネス・オブジェクトは、以下のとおりです。
  - IBM_OptParams。属性として、LaneId および Price の 2 つがあります。
  - IBM_OptimizationResults。属性として WinningBid があります。
  - MO_Instance。属性として、InstanceId、ConnectionId、Username、および Password の 4 つがあります。



ポーリング用メタオブジェクトの構成

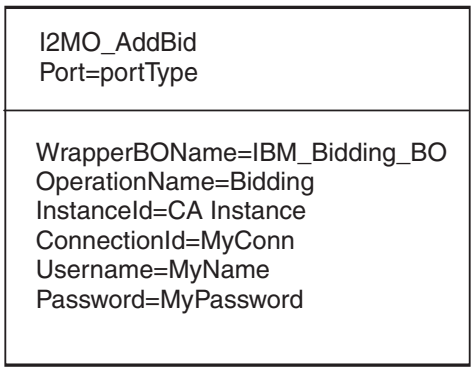
コネクターは、i2 メタオブジェクトを使用して、特定の操作の通知を受けるように CIS エージェントに登録します。これにより、ポーリングが実行されます。通知を受ける操作ごとに 1 つのメタオブジェクトを構成する必要があります。

メタオブジェクト名は常に i2MO で始まります。各メタオブジェクトは、操作をサポートするインスタンス、および操作のラッパー・ビジネス・オブジェクト名に関する情報を保持します。すべてのメタオブジェクトには、ダミー動詞を追加する必要があります。

メタオブジェクト内の属性 (InstanceId、ConnectionId、Username、Password、ラッパー・ビジネス・オブジェクト名、および操作名) の値は、静的デフォルト値です。別のインスタンスにある同じ操作を登録する場合は、デフォルト値を変更して i2 インスタンスを再始動するか、あるいは新規インスタンス用に別のメタオブジェクトを構成する必要があります。

次の図では、i2MO_AddBid というメタオブジェクトを使用して InstanceId CA_Instance を構成し、これを Bidding 操作 addBid として、IBM_Bidding_BO と

いうラッパー・ビジネス・オブジェクトに設定しています。図中の属性 (ConnectionId、Username、および Password) の値は、デフォルト値です。



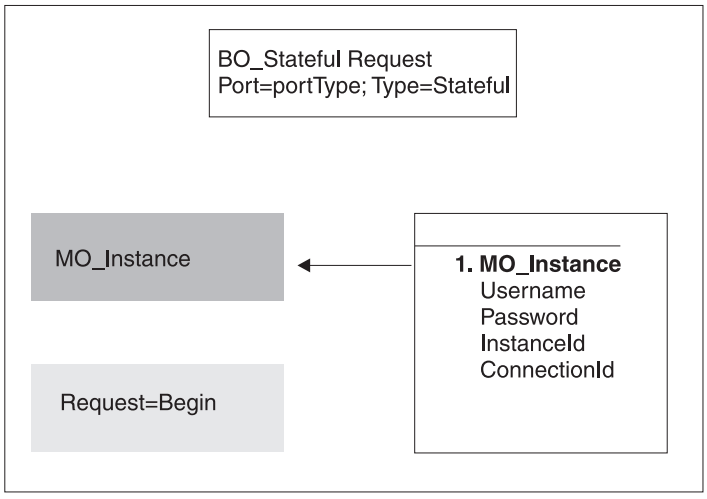
ステートフル操作のメタオブジェクトの使用

ステートフル操作 を使用すれば、i2 コネクターは一連の操作の間、セッション状態を維持することができます。このようにグループ化された操作は、i2 コネクターのパースペクティブからの複数のサービス呼び出しまたは要求とみなされます。

i2 コネクターでステートフル操作を使用する場合、統合ブローカーは、一連のステートフル操作の開始と終了を示すメタオブジェクト BO_StatefulRequest を使用して、ステートフル接続を要求します。この接続を ConnectionId に関連付けることにより、ステートフル操作用の接続が維持されます。この接続は、前述の ConnectionId を使って送信されるすべての要求について管理されます。

注: 一連のステートフル操作の処理中に i2 コネクターが破損した場合、これらの操作の状態は維持されません。

次の図は、メタオブジェクト BO_StatefulRequest を示します。



Request 属性の有効な値は「Begin」および「End」です。これにより、MO_Instance で提供される信任状情報を使って一連のステートフル操作を開始することを i2 コネクターにシグナル通知します。ConnectionId は、i2 CIS エージェントが提供または生成します。この要求から戻される ConnectionId を使用すれば、以降に i2 コネク

ターへの呼び出しを行うビジネス・プロセスは、ステートフル操作呼び出しを実行できます。i2 コネクターは、このビジネス・オブジェクトが、その内部にある ConnectionId に関連付けられた一連のステートフル操作の開始を要求しているのか終了を要求しているのかを、アプリケーション固有の情報「Type=Stateful」によって判断できます。

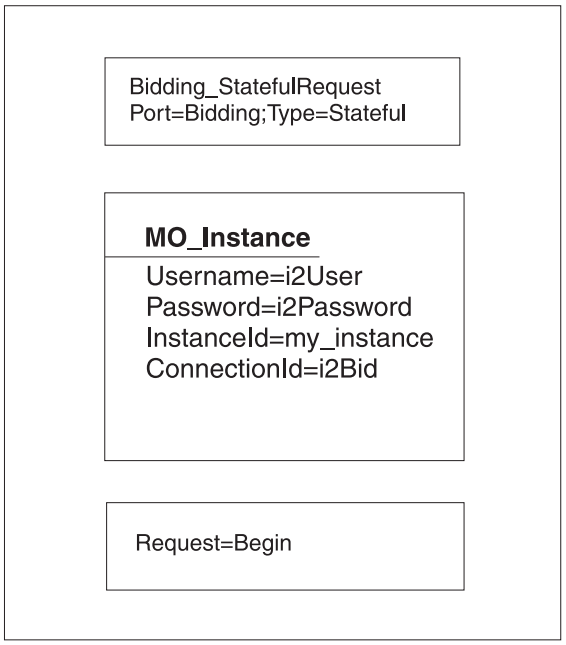
以降では、ステートフル操作のメタオブジェクトの具体例をあげて説明します。

ステートフル操作の例 1

オークション・アプリケーションには、i2 CIS アダプターを使用して、いくつかのオークション・サービスを実行する入札システムがあります。この i2 CIS アダプターには、さまざまなユーザーによる各入札を追跡し、入札処理中はいつでも落札価格を戻すステートフル操作があります。

i2 コネクターを使用してステートフル呼び出しを実行して、落札価格を戻すには、一連のステートフル操作 (入札) の開始と終了をシグナル通知するビジネス・オブジェクトを統合ブローカーに用意する必要があります。さらには、入札をサブミットするビジネス・オブジェクト、および落札価格を戻すビジネス・オブジェクトも必要です。

ブローカー内の Integrator は、メタオブジェクト「BO_StatefulRequest」を基にして、Bidding_StatefulRequest という名前のビジネス・オブジェクトを作成します。次の図は、このメタオブジェクトを示します。



ブローカーのビジネス・プロセス・フローを使用してこれらの要求を実行する例を以下に示します。

入札は、「LaneId」と「BiddingAmount」を保有し、ビジネス・オブジェクト BO_Bid で表されます。現在の落札価格を要求する呼び出しは、「BO_optimizeBids」ビジネス・オブジェクトを使って実行されます。このオブジェクトには、入力ビジネス・オブジェクトとして「LaneId」、出力または戻りビジネ

ス・オブジェクトとして落札価格があります。戻りビジネス・オブジェクトは、「LaneId」と「WinningAmount」を保有します。

注: --> 要求処理: で始まる行は、要求の実行方法を示します。

1. 入札開始: LaneId に基づいて、一連のステートフル操作を開始します。

   -->要求処理: Bidding_StatefulRequest を使用し、要求は「Begin」、ConnectionId は i2Bid とします。
2. Bid1 送信: この LaneId の入札を BiddingAmount (8) で送信します。

   -->要求処理: BO_Bid の BiddingAmount を使用します。
3. Bid2 送信: この LaneId の入札を BiddingAmount (35) で送信します。

   -->要求処理: BO_Bid の BiddingAmount を使用します。
4. Bid3 送信: この LaneId の入札を BiddingAmount (20) で送信します。

   -->要求処理: BO_Bid の BiddingAmount を使用します。
5. OptimizeBids 送信: 現在の落札価格の取得要求を送信します。

   -->要求処理: BO_OptimizeBids を使用します。

   結果: 要求は処理され、落札価格ビジネス・オブジェクトは (35) として戻されます。
6. Bid4 送信: この LaneId の入札を BiddingAmount (40) で送信します。

   -->要求処理: BO_Bid の BiddingAmount を使用します。
7. OptimizeBids 送信: 現在の落札価格の取得要求を送信します。

   -->要求処理: BO_OptimizeBids を使用します。

   結果: 要求は処理され、落札価格ビジネス・オブジェクトは (40) として戻されます。
8. 入札終了: ConnectionId を「i2Bid」として Bidding_StatefulRequest を送信して、この一連のステートフル操作の終了を要求します。

   -->要求処理: Bidding_StatefulRequest で「Request=End」とします。

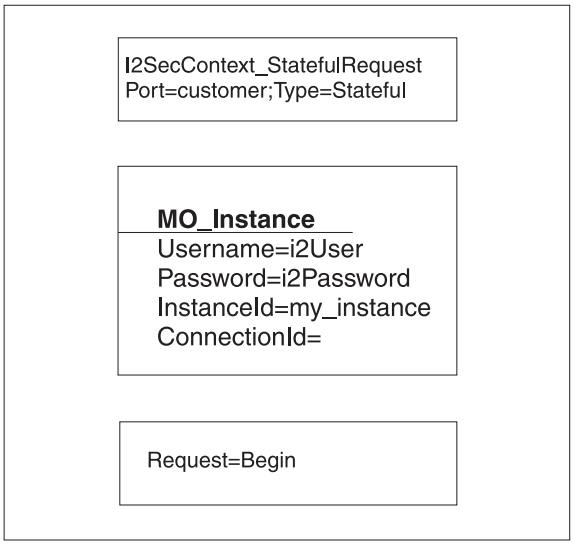
注: ConnectionId は、一連のステートフル操作と相互に関連しているので、一連の入札ごとに異なる ConnectionId を提供する必要があります。ブローカーは、i2 CIS エージェントに ConnectionId を生成させることもできます。その場合 ConnectionId は、i2 コネクターの「Request=Begin」要求呼び出しで戻されます。戻された ConnectionId を使用して、ブローカーは一連のステートフル操作を要求できます。

ステートフル操作の例 2

あるアプリケーションには、カスタマー構成プロセスの呼び出し側クライアントが使用する、特別なセキュリティー・コンテキストが必要です。i2 CIS アダプターは、以降の呼び出しで使用するためのセキュリティー・コンテキストを作成し保管するステートフル操作を提供します。

Integrator は、i2 コネクターを使用することにより、i2 CIS アダプターによって作成され提供されたセキュリティー・コンテキストを使ってこれらのステートフル操作を処理することができます。

メタオブジェクト「BO_StatefulRequest」を基にして、「i2SecContext_StatefulRequest」というビジネス・オブジェクトが作成されます。次の図は、i2SecContext_StatefulRequest メタオブジェクトを示します。



ブローカーのビジネス・プロセス・フローを使用してこれらの要求を実行する例を以下に示します。

注: --> 要求処理: で始まる行は、要求の実行方法を示します。

1. カスタマー追加の開始: i2SecContext_StatefulRequest を使用して、一連のステートフル操作を開始します。

   -->要求処理: 「Begin」を指定して、i2SecContext_StatefulRequest を使用します。

   結果: このビジネス・オブジェクトには ConnectionId が設定されていないため、要求の呼び出し後、戻されたオブジェクトに、これらのステートフル操作と関連付ける必要のある ConnectionId が含まれています。
2. セキュリティー・コンテキスト送信: セキュリティー・コンテキスト信任状情報を持つビジネス・オブジェクトを送信します。その際、ステートフル操作から戻されたものと同じ ConnectionId を使用します。この ID は、このビジネス・オブジェクトの MO_Instance に設定されます。

   -->要求処理: Security_Context ビジネス・オブジェクトを使用します。
3. カスタマー追加の送信: 同じ ConnectionId を使用して、カスタマーを追加するビジネス・オブジェクトを送信します。

   -->要求処理: Customer_Add ビジネス・オブジェクトを使用します。その際、ステートフル操作要求によって提供されたものと同じ ConnectionId を使用します。

4. カスタマー追加の終了: この一連のステートフル操作を終了するため、「Request=End」、および同じ Connectionid を指定して i2SecContext_StatefulRequest ビジネス・オブジェクトをサブミットします。

ビジネス・オブジェクトの属性プロパティーの指定

i2 コネクターには、ビジネス・オブジェクト属性に設定可能な、さまざまなプロパティーがあります。このセクションでは、コネクターがこれらのプロパティーのいくつかを解釈する方法と、ビジネス・オブジェクトを変更する際にそのプロパティーをセットする方法を解説します。

次の表に、単純属性のプロパティーを示します。

属性	説明
Name	属性の固有の名前。
Type	すべての単純属性は、String 型とする必要があります。
MaxLength	使用されません。
IsKey	各ビジネス・オブジェクトには、少なくとも 1 つのキー属性が必要です。これは、属性のキー・プロパティーを true に設定することによって指定します。i2 コネクターは、このプロパティーを検査しません。
IsForeighKey	使用されません。
Is Required	この属性が発信 XML メッセージで値を持つ必要がある場合は、true に設定します。
AppSpecInfo	使用されません。
DefaultValue	インバウンド・ビジネス・オブジェクトの単純属性が必須属性でありながら設定されていない場合に、コネクターがその属性に使用するデフォルト値を指定します。 **規則:** ポーリング・メタオブジェクトおよび MO_Instance メタオブジェクトの属性には、必ずデフォルト値を設定し、それを使用してください。InstanceId にデフォルト値が設定されていて、着信ビジネス・オブジェクトに値が設定されていない場合、コネクターはデフォルト値を取得して、このインスタンスと接続しようとします。

次の表に、子オブジェクト属性のプロパティーを示します。

属性	説明
Name	子オブジェクトの名前。
Type	子のビジネス・オブジェクト・タイプ。
Contained ObjectVersion	子ビジネス・オブジェクトを表すすべての属性に関して、このプロパティーは子のビジネス・オブジェクト・バージョン番号を示します。
Relationship	子がコンテナー属性の場合、これは `Containment` に設定されます。
IsKey	使用されません。
IsForeighKey	使用されません。

属性	説明
Is Required	XML 要素と必要性の関係の詳細については、「データ・ハンドラー・ガイド」の第 3 章『XML データ・ハンドラー』を参照してください。
AppSpecInfo	このプロパティーに関する詳細については、28 ページの『ビジネス・オブジェクトのアプリケーション固有情報の識別』を参照してください。
Cardinality	XML 要素とカーディナリティーの関係の詳細については、「データ・ハンドラー・ガイド」の第 3 章『XML データ・ハンドラー』を参照してください。

特殊属性値

ビジネス・オブジェクトの単純属性は特殊値 CxIgnore を持つことがあります。コネクターは統合ブローカーからビジネス・オブジェクトを受け取ると、CxIgnore という値を持つすべての属性を無視します。まるでコネクターにはそれらの属性が見えないかのように処理されます。これらの属性については、XML は生成されません。

i2 コネクターは、ビジネス・オブジェクトの作成に少なくとも 1 つの基本キー属性を必要とするため、コネクターに渡されるビジネス・オブジェクトに、CxIgnore が設定されていない基本キーが少なくとも 1 つはあることを確認する必要があります。

さらに i2 コネクターは、ビジネス・オブジェクト・タイプ の属性は CxBlank 値をとらないことを前提としています。CxBlank 値を持つ単純 (ストリング) 属性は、XML 文書に組み込まれます。XML 文書では、PCDATA で CxBlank に相当するものとして、空の二重引用符 ("") が使用されます。

ビジネス・オブジェクトのアプリケーション固有情報の識別

アプリケーション固有情報は、ビジネス・オブジェクトの処理方法に関するアプリケーション固有の手順をコネクターに提供します。アプリケーション固有のビジネス・オブジェクトを拡張、または変更する場合は、ビジネス・オブジェクト定義のアプリケーション固有情報が、コネクターの予期する構文に必ず合致するよう確認してください。

ビジネス・オブジェクト・レベルのアプリケーション固有情報

次の表は、i2 コネクターでサポートされるラッパー・ビジネス・オブジェクトのビジネス・オブジェクト・レベルでのアプリケーション固有情報を示します。

パラメーター	説明
Port=	操作の i2 ポート・タイプの名前。
Type=Stateful	一連のステートフル要求の開始または終了をシグナル通知するメタオブジェクト。

属性レベルのアプリケーション固有情報

次の表は、i2 コネクターでサポートされるラッパー・ビジネス・オブジェクトの属性レベルのアプリケーション固有情報を示します。

パラメーター	説明
Type=	ラッパー・ビジネス・オブジェクトの属性レベルで、属性が表しているタイプ。タイプには、操作の入出力を表すものとして、入力タイプと出力タイプがあります。

第 4 章 i2 ODA によるビジネス・オブジェクトの生成

この章では、i2 ODA (Object Discovery Agent) について説明します。ODA は、XML スキーマ ODA と連動することにより、IBM WebSphere Business Integration Adapter for i2 のビジネス・オブジェクトを生成します。

この章には、以下のセクションが含まれています。

i2 ODA の概要

i2 Object Discovery Agent (ODA) は、i2 の CIS レジストリーにあるメタデータ情報から i2 ビジネス・オブジェクトの仕様を取得するために使用するユーティリティーです。このプロセスは、ビジネス・オブジェクト開発ウィザードによって自動化されます。ビジネス・オブジェクトをサーバーに保管する前に、表示または変更することができます。

i2 ビジネス・オブジェクトの生成プロセスは、3 つの段階に分かれています。

1. i2 ODA によるスキーマ・ファイル生成の対象となるポート、操作、およびタイプを識別します。次に、そのタイプの XML スキーマを生成します。さらに、そのタイプを属性として持つ操作を表すラッパー・ビジネス・オブジェクトを生成します。
2. i2 によって生成された XML スキーマ・ファイルを処理して、XML スキーマをそのタイプの実際のビジネス・オブジェクトに変換します。
3. MO_Instance ビジネス・オブジェクトと、XML ODA を使用してそのタイプ用に生成されたビジネス・オブジェクトをリポジトリーに保管してから、ラッパー・ビジネス・オブジェクトを保管します。

詳細については、34 ページの『i2 ODA の使用手順』を参照してください。

i2 ODA のインストール

このセクションでは、i2 ODA のインストールと起動、i2 ODA の複数インスタンスの実行、エラーおよびトレースのメッセージ・ファイルの処理の方法について説明します。

i2 ODA のインストール手順

始める前に: この章は、i2 コネクターや、コネクターを使用する上で必要となるソフトウェアがすでにインストール済みであることを前提としています (9 ページの『第 2 章 コネクターのインストールと構成』を参照)。使用する i2 アプリケーションがバージョン 6.0.1、i2 ODA が 1.2.x であることを確認してください。

i2 ODA をインストールするには、IBM WebSphere Business Integration Adapters のインストーラーを使用します。「システム・インストール・ガイド *(UNIX 版)*」または「システム・インストール・ガイド *(Windows 版)*」(ICS の場合)、「*WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド*」、あるいは「*アダプター実装ガイド (WebSphere Application Server)*」の説明に従ってください。インストールが完了すると、製品がインストールされているシステムのディレクトリーに、以下のファイルがインストールされます。

- ODA¥i2¥i2ODA.jar
- ODA¥messages¥i2ODAAgent.txt
- ODA¥i2¥start_i2ODA.bat (Windows のみ)
- ODA/i2/start_i2ODA.sh (UNIX のみ)

注:

1. 特に断りのない限り、円記号 (¥) をディレクトリー・パスの規則として使用します。UNIX のインストールでは、円記号の代わりにスラッシュ (/) を使用します。
2. すべての製品パス名は、システム上の、製品のインストール先ディレクトリーを基準とした相対パス名です。

その他のインストール要件

- i2 には、レジストリーからメタデータ情報を取得する MetadataService アダプターがあります。このアダプターを i2 アプリケーションのインスタンスにインストールする必要があります。アダプターを始動してから、MetadataService を使用するようにしてください。
- ポート MetadataService のバインディング・ファイル (例: `TDMMetadata.xml`) には、ポート、操作、およびタイプ情報が含まれています。このファイルは、i2 構成ディレクトリーに置く必要があります。

i2 ODA の起動

始める前に: i2 ODA および XML スキーマ ODA がシステムにインストール済みであることを確認してください。

i2 ODA を起動するには、次のファイルを実行します。

- **UNIX:** start_i2ODA.sh
- **Windows:** start_i2ODA.bat

CIS-SDK および j2ee.jar の始動ファイルへの適切なパスを追加する必要があります。i2 lib ディレクトリーにあるコネクター jar のロケーションを正確に指定してください。

例: 以下のパス情報を start_i2.bat ファイルに追加する必要があります。

```
set I2_CIS_HOME_DIR=C:¥i2¥CIS¥6.0¥cis-sdk
set J2EE_PATH=C:¥J2EE_JAR
set CONNECTORJAR=
```

注: これらは単なる例です。ローカル・インストールに応じてパス情報を変更してください。

Business Object Designer を使用して、i2 ODA を構成し実行してください。Business Object Designer は、各スクリプトまたはバッチ・ファイルの *AGENTNAME* 変数に指定された名前で、各 ODA を位置指定します。このコネクターのデフォルトの ODA の名前は、*i2ODA* です。

エラーおよびトレースのメッセージ・ファイルの処理

エラーおよびトレースのメッセージ・ファイル (デフォルトは `i2ODAAgent.txt`) は、製品ディレクトリーの下の `¥ODA¥messages¥` にあります。これらのファイルには、以下の命名規則が使用されます。

AgentName`Agent.txt`

例: *AGENTNAME* 変数に *i2ODA1* が指定されている場合、ツールは、関連するメッセージ・ファイルの名前は `i2ODA1Agent.txt` であるとみなします。

ODA インスタンスごとにメッセージ・ファイルを 1 つずつ用意することも、名前の異なる ODA が同じメッセージ・ファイルを使用するようにすることもできます。メッセージ・ファイルの名前は、Business Object Designer で ODA 構成の一部として指定されます。

注: メッセージ・ファイルの名前が正しく指定されていない場合は、ODA を構成したときに、ODA がメッセージを使用せずに実行されてしまう原因になります。メッセージ・ファイル名の指定の詳細については、35 ページの『エージェント・プロパティーを構成する』を参照してください。

構成処理中に、以下を指定してください。

- i2 ODA がエラーおよびトレース情報を書き込むファイル
- 0 から 5 を範囲とするトレース・レベル

次の表に、トレース・レベルを示します。

トレース・レベル	説明
0	• i2 ODA アプリケーションからのエラーおよび致命的エラーをログに記録します。 • システム管理者による対応が必要となる警告をログに記録します。
1	メソッドの開始メッセージおよび既存のメッセージをすべてトレースします。
2	ODA のプロパティーおよびプロパティー値をトレースします。
3	すべてのビジネス・オブジェクトの名前をトレースします。
4	ビジネス・オブジェクト・プロパティーおよび受信した値をトレースします。
5	• ODA のすべてのプロパティーに対する初期値を示します。 • ビジネス・オブジェクト定義のダンプをトレースします。

これらの値をどこで構成するかについては、35 ページの『エージェント・プロパティーを構成する』を参照してください。

Business Object Designer での i2 ODA の使用

このセクションでは、Business Object Designer で i2 ODA を使用してビジネス・オブジェクトを生成する方法について説明します。Business Object Designer の起動については、「*IBM WebSphere Business Integration Adapters ビジネス・オブジェクト開発ガイド*」を参照してください。

ODA を起動させた後、Business Object Designer を起動し、構成して実行する必要があります。Business Object Designer には、ODA を使用してビジネス・オブジェクト定義を生成する 6 つのステップを順を追って実行するためのウィザードが用意されています。この 6 つのステップは以下のとおりです。

1. エージェントを選択します。
2. エージェント・プロパティーを構成します。
3. ノードを展開し、ポート・タイプ、操作、および入出力タイプを選択します。
4. 選択内容を確認し、ラッパー・ビジネス・オブジェクトを生成、保管します。
5. ビジネス・オブジェクトを完成させて、そのタイプのビジネス・オブジェクトを生成します。
6. ビジネス・オブジェクト・ファイルを保管します。

以降では、各ステップの詳細を説明します。

i2 ODA の使用手順

始める前に: i2 Business Object Designer ウィザードを始動する必要があります。

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

   結果: Business Object Designer のウィザードに、「エージェントの選択」という名前の最初のウィンドウが表示されます。

以下のステップを実行します。

エージェントを選択する

ODA を選択するには、以下のようにします。

1. 「エージェントの検索」をクリックすることにより、登録済みまたは現在実行中の ODA のすべてを「検索されたエージェント」フィールドに表示します。
2. 表示されたリストから、希望する ODA を選択します。

   注: i2 用 ODA のデフォルトの名前は *i2ODA* です。エージェント名は、`start_i2ODA.bat` または `start_i2ODA.sh` ファイルの i2 変数の値によって異なります。

   推奨: ODA ユーティリティーの複数インスタンスを実行する場合は、デフォルト名を変更する必要があります。この変更を行うには、インスタンスごとに別のバッチ・ファイルを作成するか、あるいは各バッチ・ファイルの *AGENTNAME* 変数に固有の名前を指定します。

   複数の ODA インスタンスが別のマシンで稼働している場合、各インスタンスはそれぞれの i2 ODA 値で Business Object Designer 画面に表示されます。2 つの

ODA が同じ i2 値を持つ場合は、どちらか一方の ODA を使用できますが、それが目的の ODA ではない可能性もあります。このような ODA に固有の名前を割り当てるには、i2 名の前にホスト・マシン名を付けるか、あるいは ORB ファインダー (osfind など) を使ってネットワーク上の既存 CORBA オブジェクト名を探し出します。

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

エージェント・プロパティーを構成する

Business Object Designer で最初に i2 ODA とやり取りするときに、一連の初期設定プロパティーの入力プロンプトが出されます。これらのプロパティーを指定したファイル名でプロファイルに保管しておくと、i2 ODA を使用するたびに同じプロパティーを再入力する必要はありません。 ODA プロファイルの指定については、「*ビジネス・オブジェクト開発ガイド*」を参照してください。

このような一回限りのプロパティーを構成するほかに、CIS エージェントへの接続プロパティーや、ツリー・ノードの定義プロパティーを構成する必要があります。

次の表に、構成すべきプロパティーを示します。

行番号	プロパティー名	プロパティー・タイプ	説明
1	DefaultBOPrefix	String	ビジネス・オブジェクト名が固有となるように、名前の先頭に付加されるテキスト。 **例:** i2_BO
2	SchemaFileLocation	String	生成された .xsd ファイルが保管されるパス。これは必須です。スキーマ・ファイルを保管するパスを必ず指定してください。
3	MessageFile	String	エラーおよびメッセージのファイルのパス。このファイルが指定されない場合、ODA からのエラー・メッセージは表示されません。i2 ODA には、命名規則に従ったファイル名が表示されます。**例:** エージェントの名前が i2ODA の場合、メッセージ・ファイル・プロパティーの値は i2ODAAgent.txt と表示されます。
4	CISAgentHostName	String	CIS エージェントのホスト名。CIS エージェントがリモート・マシンで稼働する場合に指定します。
5	MetadataService	String	MetadataService がある i2 CIS ポートの名前。デフォルト値は MetadataService です。
6	PortTypesOperation	String	メタデータ・イントロスペクションを使ってポート・タイプを取得する操作の名前。デフォルト値は getPorttypes です。
7	SchemasOperation	String	選択された操作のスキーマを取得する操作の名前。デフォルト値は getSchemasForOperation です。

エージェント始動時には、エージェントに関するいくつかのオプショナル・パラメーターも構成できます。

- TraceFileName: i2 ODA がトレース情報を書き込むファイル。このパラメーターのコマンド行オプションは -t です。i2ODA は、命名規則に従ってファイルに名

前を付けます。**例:** エージェントの名前が i2ODA の場合は、i2ODAtrace.txt という名前のトレース・ファイルが生成されます。

- TraceLevel: i2 ODA で使用可能なトレースのレベル。詳細については、33 ページの『エラーおよびトレースのメッセージ・ファイルの処理』を参照してください。

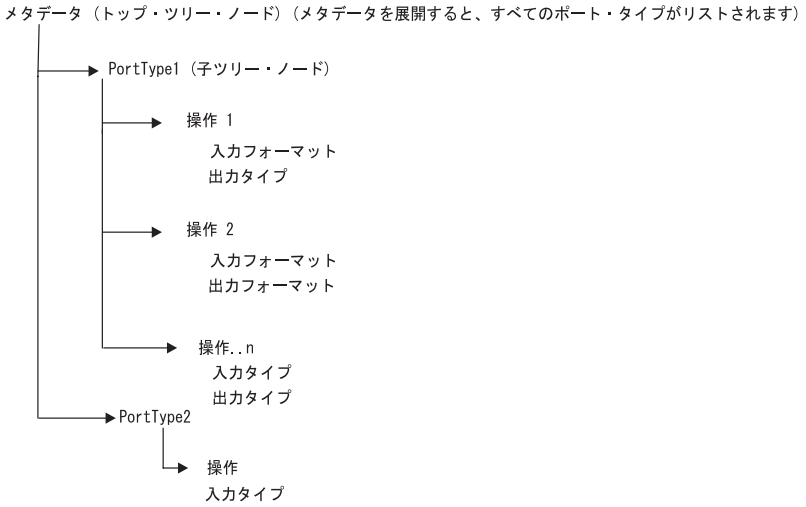
ノードを展開し、ポート・タイプ、操作、および入出力のタイプやフォーマットを選択する

i2 ODA は、前のステップで構成したプロパティーを使用して、指定された i2 アプリケーションに接続します。Business Object Designer では、登録済みポート、各ポートの操作、および各操作の入出力タイプに関するメタデータ情報がツリー構造で表示されます。ポートや操作のツリー・ノードを展開するには、ノードを右マウス・ボタンでクリックします。タイプ ・ノードは、ツリーのリーフ・ノードなので、展開できません。

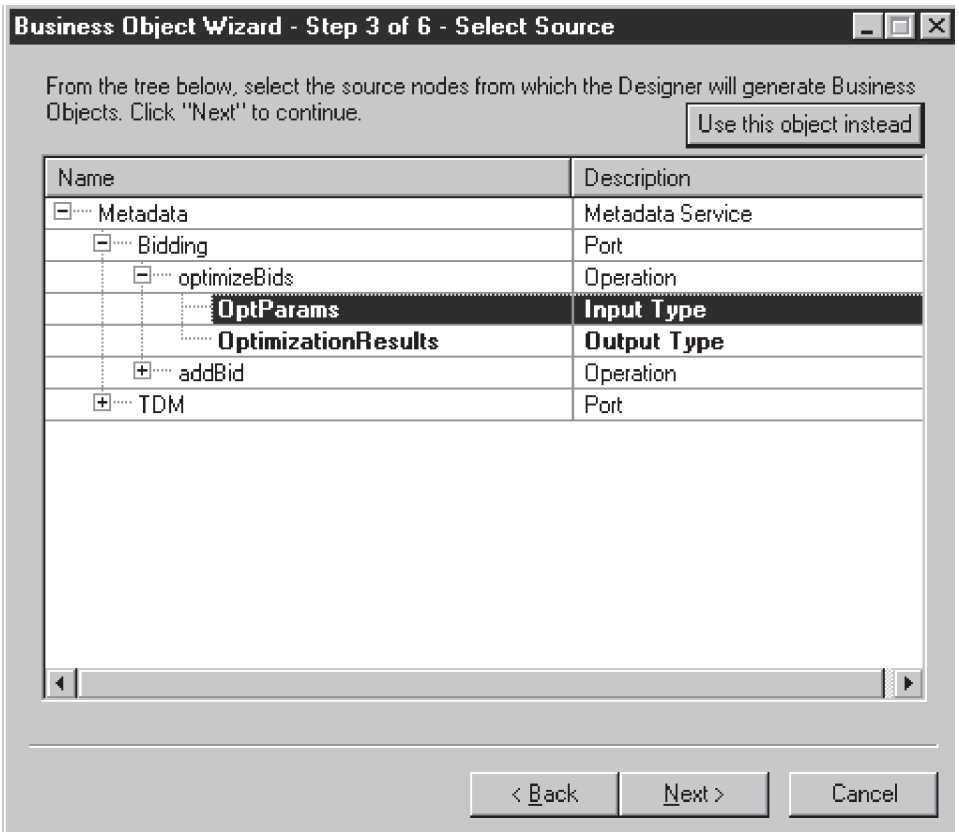
「タイプ」と「フォーマット」は、操作の入力値および出力値を処理するときに区別されます。操作の入力および出力ビジネス・オブジェクトを表示したときに、その表示が「タイプ」ではなく「フォーマット」となっている場合は、「出力タイプ」ではなく「出力フォーマット」を表します。

注: 本書では「タイプ」は、「タイプ」または「フォーマット」を意味します。

次の図は、ノードを展開した状態のツリー表示を示します。



- ビジネス・オブジェクトを生成するポート、操作、およびタイプを選択します。各操作には、ポートの入力タイプおよび出力タイプがあり、これらの各タイプに、対応するビジネス・オブジェクトが必要です。



結果: i2 ODA は、選択されたタイプのスキーマ・ファイルを作成します。このスキーマ・ファイルが XML スキーマ ODA の入力となって、対応するビジネス・オブジェクトが生成されます。XML スキーマ・ファイルの名前は、命名規則に従って、`input/output_operation_type` となります。

例: persistOrder 操作は、入力タイプも出力タイプも Order ですが、この 2 つのフォーマットは異なります。i2 ODA はこれらを検証し、入力タイプ用と出力タイプ用に異なるスキーマ・ファイルを生成します。

- i2BO_in_persistorder_Order.xsd (入力)
- i2BO_out_persistorder_Order.xsd (出力)

スキーマ・ファイルは、ODA の SchemaFileLocation プロパティーに指定されたディレクトリーに保管されます。i2 ODA は、スキーマを新規に作成する前に、スキーマ・ロケーションにスキーマがあるかどうかを検査します。同じタイプの `xsd` がすでに存在する場合、i2 ODA はスキーマを重複して作成することはありません。

選択内容を確認し、ラッパー・ビジネス・オブジェクトを生成、保管する

1. i2 ODA のラッパー・ビジネス・オブジェクト生成に関して選択した操作を確認します。

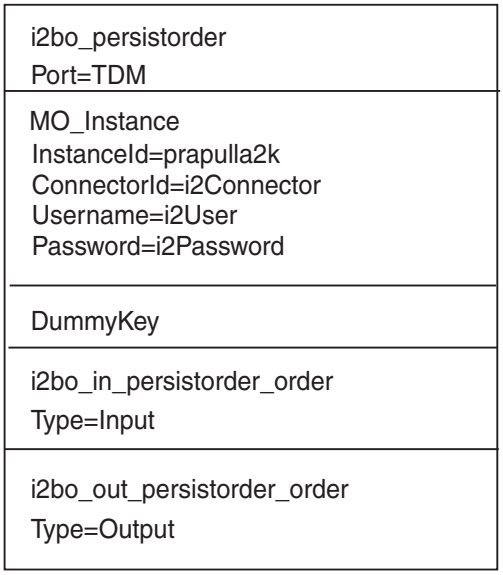
   結果: i2 ODA は、選択されたポート・タイプの操作を表すラッパー・ビジネス・オブジェクトを作成します。各操作に対して、ラッパー・ビジネス・オブジ

ェクトは 1 つです。操作の入力タイプおよび出力タイプは、このラッパー・ビジネス・オブジェクトの属性になり、操作は動詞になります。ポート・タイプは、このビジネス・オブジェクトのアプリケーション固有情報になります。

ラッパー・ビジネス・オブジェクトの名前は DefaultBOPrefix_operation です。ラッパー・ビジネス・オブジェクトの内容を以下に示します。

- ポート情報
- MO_instance。以下の要素が含まれます。
  - InstanceId
  - ConnectionId
  - Username
  - Password
- ダミー・キー (キーがないと、ビジネス・オブジェクトの作成は失敗するため)
- 操作の入力タイプおよび出力タイプを表す、2 つの単一カーディナリティー属性。属性には、BOPrefix_in_operation_type および BOPrefix_out_operation_type という名前が付けられます。

例: 操作 persistOrder のラッパー BO は i2BO_persistOrder です。

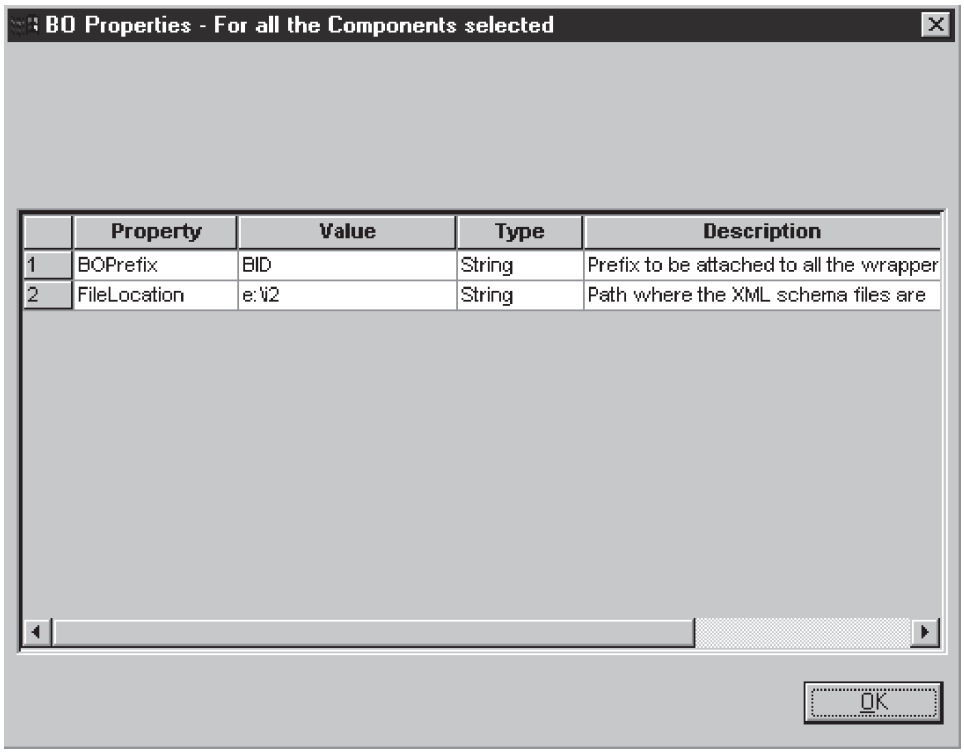


2. ラッパー・ビジネス・オブジェクトをファイルに保管してください。InterChange Server に保管しようとすると、対応する従属属性ビジネス・オブジェクトがまだ XML ODA スキーマによって作成されていないため、エラーになります。

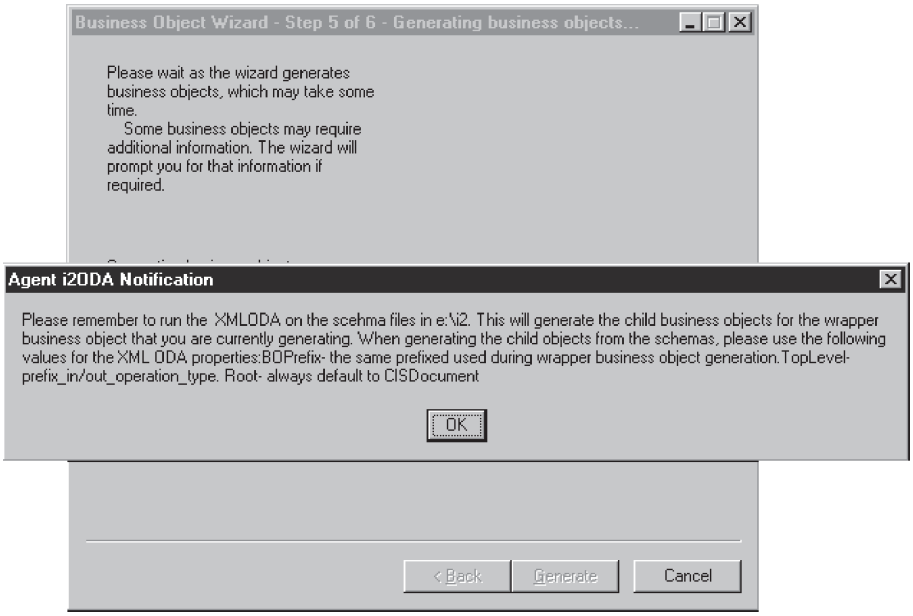
   ガイドライン: このステップでビジネス・オブジェクトをファイルに保管するとき、現在の Business Object Designer ウィザードの参照先は、エージェントが稼働中のマシンになっています。別のポップアップ・ウィンドウが表示されて、生成されたラッパー・ビジネス・オブジェクトの保管場所を指定するよう促されます。

ビジネス・オブジェクトを完成させて、そのタイプのビジネス・オブジェクトを生成する

1. Business Object Designer の「BO プロパティー」ウィンドウに表示されるプロパティー値を確認します。「エージェントの構成」ウィンドウで入力した値 (例: DefaultBOPrefix) のままでよい場合は、ここで値を変更する必要はありません。



2. XML スキーマ ODA エージェントが稼働中であることを確認してください。
   a. XML スキーマ ODA は、事前に SchemaFileLocation に保管されたスキーマ・ファイルを読み取って、その入力タイプと出力タイプのビジネス・オブジェクトを生成します。



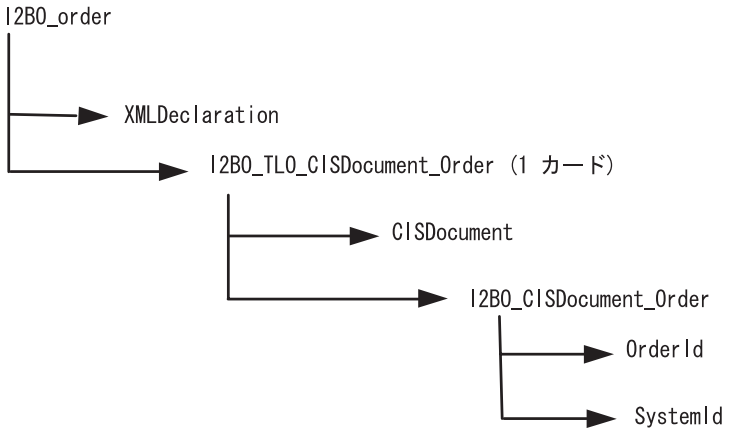
b. これらのビジネス・オブジェクトを、ラッパー・ビジネス・オブジェクトと同じディレクトリーに保管してください。

ガイドライン: ODA を実行するときに指定したディレクトリーにあるスキーマ・ファイルに対して XML ODA を実行するようにしてください。これにより、生成したラッパー・ビジネス・オブジェクトの子ビジネス・オブジェクトが生成されます。スキーマから子オブジェクトを生成するときは、XML ODA プロパティーに以下の値を使用してください。

- BOPrefix: ラッパー・ビジネス・オブジェクト生成で使用したプレフィックスと同じにします。
- TopLevel: prefix_in/out_operation_type とします。
- Root: 常にデフォルトで CISDocument になります。

XML スキーマ ODA の BOPrefix は、i2 ODA の BOPrefix と同じにしてください。

次の図は、XML スキーマ ODA が i2_order.xsd について生成するビジネス・オブジェクトを示します。XML データ・ハンドラーは、CISDocument の次の要素と BOPrefix の組み合わせにより、ビジネス・オブジェクト名を判別します。



注: <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##any" processContents=skip"/> タグが含まれていても、ビジネス・オブジェクト生成がエラーにならなくなりました。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。

ビジネス・オブジェクト・ファイルを保管する

必要なすべてのビジネス・オブジェクトが生成されたら、それらを、コラボレーションが使用できるように InterChange Server に保管する必要があります。Business Object Designer ユーティリティーを使用して、ビジネス・オブジェクトをサーバーにコピーしてください。これらのファイルを 1 つのファイルにまとめて、サーバーにコピーすることができます。

ガイドライン: XML スキーマ ODA を実行してから、ラッパー・ビジネス・オブジェクトをサーバーに保管するようにしてください。

例:

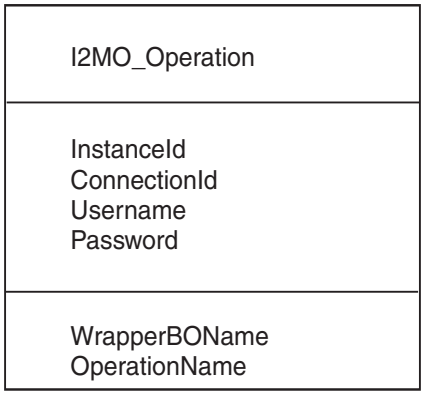
プロパティー	値
FileName	D:¥i2¥odaschema¥i2persist_in_persistOrder_Order_Order.xsd
Root	CISDocument
TopLevel	in_persistOrder_Order
BOSelection	false
BOPrefix	i2persist
Doctypeor SchemaLocation	true
TraceFileName	XMLODAtrace.txt
TraceLevel	5
MessageFile	XMLODAAgent.txt

この段階で、異なるインスタンス上で別の操作を実行するかどうかを決定できます。MO_Instance の複製を作成して、それらの InstanceId にデフォルト値を設定できます。デフォルトの MO_Instance を、それぞれの操作ごとのラッパー・ビジネス・オブジェクト内で新しく作成する MO_Instance に置き換える必要があります。

ポーリング用メタオブジェクトの作成

ビジネス・オブジェクトを作成したら、次に System Manager を使って、ポーリング用のメタオブジェクトを作成する必要があります。これらのオブジェクトの名前は i2MO プレフィックスで始まり、操作名がそれに続きます。属性にはデフォルト値が設定されている必要があります。この情報は、ポーリングで特定の操作を登録し、i2 アプリケーションから登録済み操作についての出力があるかを検査する際に使用されます。

次の図は、i2 MO_Operation の i2 メタオブジェクトの構造を示します。図中には、メタオブジェクトの属性 (InstanceId、ConnectionId、Username、および Password)、操作名、およびビジネス・オブジェクト・ラッパー名があります。



メタオブジェクトの構成については、 19 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』を参照してください。

第 5 章 トラブルシューティングとエラー処理

この章では、i2 コネクターのエラー処理方法について説明します。このコネクターは、メッセージのロギングおよびトレースを生成します。本章の内容は、次のとおりです。

エラー・メッセージのロギング

コネクターは、処理中に異常な状態が発生した場合は、トレース・レベルにかかわらず、エラー・メッセージをログに記録します。このようなエラーが発生すると、コネクターは、受信した情報に基づいて、失敗したビジネス・オブジェクトのテキスト表現も出力します。このテキストは i2 アダプターのログ・ファイルに書き込まれます。このログ・ファイル名は、コネクター・プロパティー LogFileName で設定します。

メッセージには、状態や結果が詳細に記述されます。また、デバッグに役立つ追加情報として、ビジネス・オブジェクト・ダンプやスタック・トレース (例外用) などが含まれる場合もあります。

エラー・タイプ: エラー・メッセージには、次の 2 種類があります。

- エラー: コネクターがリカバリー可能な状態。通常は、現在の処理が中止されます。
- 致命的エラー: リカバリー不能なエラー状態。ポーリング関連のエラーおよび要求処理エラーについては、以降のセクションを参照してください。

エラー・メッセージの構造

コネクターが生成するすべてのエラー・メッセージは、`i2Adapter.txt` というメッセージ・ファイルに保管されます。一般に、各エラー・メッセージにはメッセージ ID、エラー・メッセージ、および詳細記述や問題修正のヒントに関する説明セクションがあります。

メッセージ ID

メッセージ

[EXPL]

i2 コネクターのメッセージ ID の範囲は 90000 から 92000 です。このうち、90000 から 91000 はポーリング関連のエラー・メッセージ用で、91000 から 92000 は要求処理エラー・メッセージ用です。

例: メッセージ構造の例を以下に示します。ここで *nnnnn* はメッセージ ID を表します。

```
nnnnn
Not able to get a connection for this instance {1}.
[EXPL]
Please ensure that the instance specified is up.
{1}--Parameter to the error message, in this case the instance id.
```

ポーリング関連のエラー・メッセージ

次の表に、ポーリング関連のエラー・メッセージを示します。これらは、i2 アダプターのログ・ファイルに記録されます。

注:

1. 場合によっては、コネクターは、E メール通知を起動させるために、致命的エラー (ログ・メッセージ・タイプ XRD_FATAL) をログに記録します。このエラーを統合ブローカーのログに記録する場合は、コネクター・プロパティー LogAtInterchangeEnd を true に設定する必要があります。
2. E メール通知は、E メール・コネクターが構成済みの場合のみ送信されます。

エラーの説明	エラー・タイプ	i2 コネクターによる処理
Connection lost while polling	致命的エラー	コネクターは、ポーリング呼び出しのときに接続エラーを検出しました。致命的エラー (ログ・メッセージ・タイプ XRD_FATAL) がログに記録され、XML メッセージがダンプされます。致命的エラーのため、E メール通知が起動します。このエラーを統合ブローカーのログに記録する場合は、コネクター・プロパティー LogAtInterchange を true に設定して、アクティブなインスタンスで実行中のほかの操作についてポーリングを続行する必要があります。

エラーの説明	エラー・タイプ	i2 コネクターによる処理
Not able to register an operation with the CIS Agent	致命的エラー	致命的エラー (ログ・メッセージ・タイプ XRD_FATAL) がログに記録され、XML メッセージがダンプされます。致命的エラーのため、E メール通知が起動します。このエラーを統合ブローカーのログに記録する場合は、コネクター・プロパティー LogAtInterchange を true に設定する必要があります。 その後のポーリング呼び出しで、i2 コネクターは、前のポーリング呼び出しで登録できなかった操作を再登録しようとします。すべての操作の登録に失敗した場合は、CIS エージェントがダウンしているとみなされ、コネクターは APPRESPONSETIMEOUT を戻します。これにより、i2 コネクターはシャットダウンされます。
No metaobjects configured for polling	エラー	i2 コネクターは、メタオブジェクトがポーリング用に構成されていないことをログに記録してから、FAIL を戻します。
Default value for the metadata object attributes not set	エラー	デフォルト値属性プロパティーの詳細については、19 ページの『第 3 章 コネクターのビジネス・オブジェクトについて』を参照してください。エラー「指定のメタオブジェクトにはポーリングに必要な情報がありません (required information for polling was not found for the specified metaobject)」がログに記録され、ほかのメッセージの処理が続行されます。
Not able to fetch the message from CIS Agent	エラー/SUCCESS	i2 アプリケーションへのポーリング呼び出しでヌルが戻された場合、または操作のイベントがない場合、コネクターはほかの操作のポーリングを続行します。 CIS クライアント API から例外をキャッチした場合は、ERROR_PROCESSING_EVENT という語を含むエラー・メッセージがログに記録されます。コネクターは、ほかの操作のポーリングを続行します。

エラーの説明	エラー・タイプ	i2 コネクターによる処理
Fail to convert XML message to IBM business object	エラー	このメッセージについては、XML メッセージに構文エラーがあるという内容のエラーがログに記録され、XML メッセージがログ・ファイルにダンプされます。ほかのメッセージの処理は続行されます。
Any error when posting event to the broker	エラー/致命的	戻りコードが CONNECTOR_NOT_ACTIVE の場合、コネクターは致命的エラーをログに記録します。さらに、コネクターをシャットダウンする APPRESPONSETIMEOUT を戻します。このエラーは、ERROR_POSTING_EVENT 状況としてログに記録されます。 戻りコードが NO_SUBSCRIPTION_FOUND の場合、このエラーは UNSUBSCRIBED 状況としてログに記録され、ポーリングは続行されます。戻りコードが FAIL の場合、このエラーは、ビジネス・オブジェクトの ERROR_POSTING_EVENT 状況として、ログに記録されます。ほかのメッセージのポーリングは続行されます。
Event not subscribed to	エラー	このエラーは、ビジネス・オブジェクトの UNSUBSCRIBED 状況としてログに記録され、XML メッセージがログ・ファイルにダンプされます。ほかのメッセージのポーリングは続行されます。

サービス呼び出し要求処理のエラー・メッセージ

次の表に、サービス呼び出し要求処理のエラー・メッセージを示します。これらは、i2 アダプターのログ・ファイルに記録されます。

エラーの説明	エラー・タイプ	i2 コネクターによる処理
Not able to get a connection for the specified port type and instance.	致命的エラー	コネクターは、サービス要求処理呼び出しのときに接続エラーを検出しました。致命的エラーがログに記録され、例外の状況は FAIL となります。また、例外オブジェクトには、例外の原因に関する詳細な例外メッセージが設定されます。

エラーの説明	エラー・タイプ	i2 コネクターによる処理
No instanceId found in the metadata object within the incoming business object	エラー	コネクターは、ラッパー・ビジネス・オブジェクト内のメタオブジェクトの InstanceId 属性用にデフォルト値が設定されているかどうかを検査します。設定されている場合、コネクターはこのインスタンスに接続して、操作を実行しようとします。デフォルト値がない場合は、このエラー・メッセージがログに記録され、その状況は FAIL となります。また、例外オブジェクトには、例外の原因に関する詳細な例外メッセージが設定されます。
Not able to convert the incoming child business object attributes to XMl.	エラー	i2 コネクターは、このメッセージをアダプター・ログに記録し、例外の状況を FAIL に設定します。
Failure executing the operation on the i2 side.	エラー	i2 コネクターは、Resource Exception からのメッセージをアダプター・ログに記録し、例外の状況を FAIL に設定します。
Not able to convert the returned CIS Record if the operation was SUCCESSFUL to XML.	エラー	i2 コネクターは、このメッセージをアダプター・ログに記録し、例外の状況を FAIL に設定します。
Not able to convert the XML message to the business object.	エラー	i2 コネクターは、このメッセージをアダプター・ログに記録し、例外の状況を FAIL に設定します。さらに、エラー・メッセージと一緒に XML メッセージもログ・ファイルにダンプされます。
Execute method returns null output	エラー/SUCCESS	操作に出力タイプがない場合は、この実行メソッドの実行は SUCCESS とみなされます。出力が期待される場合、実行メソッドは例外を発行し、この例外はコネクターによってキャッチされます。

トレース・メッセージ

トレースは、コネクターの振る舞いを詳細にトレースするためにオンにできるオプションのデバッグ機能です。トレース・メッセージは構成可能であり、動的変更が可能です。要求の詳細に応じてさまざまなレベルを設定します。トレース・メッセージは、デフォルトでは「STDOUT」(画面) に書き込まれます。また、トレースをファイルに書き込むように構成することもできます。

推奨 : トレースは、パフォーマンスを向上させ、ファイル・サイズを小さくするために、実動システム上ではオフにするか、できるだけ低レベルに設定しておいてください。

次の表では、i2 コネクターが各トレース・レベルで出力するトレース・メッセージのタイプについて説明します。すべてのトレース・メッセージがコネクター・プロパティー TraceFileName によって指定されたファイルに表示されます。これらのメッセージは、IBM WebSphere Business Integration Adapter アーキテクチャーによって出力されるトレース・メッセージに追加されます。

トレース・レベル	トレース・メッセージ

レベル 0	コネクター・バージョンを識別するメッセージ。このレベルではその他のトレースは実行されません。このメッセージは、常時表示されます。
	
	例:
	
	`'2002/07/10 15:01:46.812:`
	`This is version 1.0 of the i2 Adapter'.`

レベル 1	`pollForEvents` メソッドが実行される度に配信されるメッセージ。
レベル 2	• gotApplEvent から統合ブローカーにビジネス・オブジェクトが通知される度にログに記録されるメッセージ。 • ビジネス・オブジェクト要求を受信する度に表示されるメッセージ。
レベル 3	適用されない
レベル 4	• アプリケーション固有の情報メッセージ。例えば、ビジネス・オブジェクトのアプリケーション固有の情報フィールドを解析する関数によって戻された値を示すメッセージです。 • コネクターが、コネクターのプロセス・フローのトレースに役立つ関数を開始または終了したかどうかを識別するメッセージ。
レベル 5	• コネクターの初期設定を示すメッセージ。例えば、統合ブローカーから検索された構成プロパティーの値を示すメッセージです。 • ビジネス・オブジェクト・ダンプを示すメッセージ。このトレース・レベルでは、コネクターは、オブジェクト (コネクターがコラボレーションから受信するオブジェクトを示す) の処理を開始する前、およびオブジェクト (コネクターがコラボレーションに戻すオブジェクトを示す) の処理を終了した後に、処理対象のビジネス・オブジェクトのテキスト表現を出力します。

異なるサブネット上の CIS に対するアダプターの実行

CIS エージェントおよび CIS アダプターが i2 用アダプターとは異なるサブネットで稼働している場合は、RMI サーバー Hostname に使用されている完全修飾名または IP アドレスを、CIS エージェントと CIS アダプターに知らせる必要があります。

CIS エージェントやインストール済みアダプターを始動するときに、JVM 引き数を設定する必要があります。

インスタンス ID および RMI サーバー Hostname を設定するには、以下のステップを実行します。

注: 以降の説明で、Hostname とは、CIS エージェントおよび CIS アダプターが稼働しているマシン名またはマシンの IP アドレスを指します。IP アドレスが変更されることがなければ、IP アドレスを使用できます。

1. アダプターがインストール済みの場合は、uninstallAdapter.py を実行して、アダプターをアンインストールします。
2. CIS アダプターのメタデータ・ファイル Bindings.xml で、Bindings 要素に属性として <bindings instanceID=Hostname> を挿入して変更します。
3. 次の JVM 引き数を付けて、アダプターを再インストールします。

   ```
   installAdapter.py -j-Djava.rmi.server.hostname=<hostname> -m
    <metadatabinding file>
   ```

4. 同じ JVM 引き数を付けて、CIS エージェントを始動します。

   ```
   -j-Djava.rmi.server.hostname=<hostname>
   ```

5. CIS エージェントを再始動します。その際に、RMI サーバー Hostname を JVM 引き数プロパティーとして指定します。
6. 新規に構成されたアダプターを始動します。その際に、RMI サーバー Hostname を JVM 引き数プロパティーとして指定します。
7. CIS エージェントを使用可能にして、Adapter for i2 が稼働しているマシンにイベント通知を送信するには、Adapter for i2 が稼働しているマシンの完全修飾ホスト名を提供する必要があります。このためには、次の行を追加します。

   ```
   MACHINE_NAME=mymachine.company.com
   ```

   追加先のファイルは
   [i2_CIS_install]/{cis-sdk}/properties/cisclient.properties です。

   ここで、mymachine.company.com は Adapter for i2 が稼働しているマシンの完全修飾名です。これにより、CIS クライアント (Adapter for i2) に対して、リモートで稼働する場合は cisclient.properties ファイルに指定されたマシン名を該当のマシン名として使用するよう指示します。
8. 場合によっては、Adapter for i2 が稼働しているリモート・マシンから CIS エージェントのマシン名が解決できないときに、CIS エージェントのマシン名を入力する必要がある場合があります。これは以下のように行います。
   a. CIS エージェント・マシンの IP アドレスを、ご使用の CIS エージェントの settings.xml ファイルの <MACHINE_NAME> エレメントの値として設定します。
   b. configure.py を実行して変更を加えます。
   c. エージェントおよびすべてのアダプターを再始動します。

異なるサブネットでの CIS の実行については、ご使用の CIS ガイドを参照してください。

トラブルシューティングのヒント

問題をトラブルシューティングするときは、以下のヒントを参考にしてください。

- CIS エージェントがリモートで稼働しているときは、リモート・マシンに ping し、リモート・マシンからこのマシンにも ping してみてください。
- CIS エージェント・サービスを検査します。
- アダプターが稼働であることを確認します。
- ビジネス・オブジェクトの構造がオペレーションと矛盾していないことを確認してください。

- InstanceId のデフォルト値がメタオブジェクトに設定されているかを確認します。
- コネクターを要求処理モードでのみ実行する場合は、**-fno** オプション・セットを付けてコネクターを始動するようにします。

付録 A. コネクターの標準構成プロパティー

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

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

コネクターによっては、一部の標準プロパティーが使用されないことがあります。Connector Configurator から統合ブローカーを選択すると、その統合ブローカーと連携するアダプターのために構成する必要のある標準プロパティーのリストが表示されます。

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

アダプター・コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有のプロパティー

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

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

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

特定のプロパティーの更新方法を確認するには、「Connector Configurator」ウィンドウ内の**「更新メソッド」**列を参照するか、次に示す 53 ページの表 2 の「更新メソッド」列を参照してください。

標準プロパティーの要約

表 2 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

注: 表 2 の「注」列にある「Repository Directory は REMOTE」という句は、ブローカーが InterChange Server であることを示します。ブローカーが WMQI または WAS の場合には、リポジトリー・ディレクトリーは LOCAL に設定されます。

表 2. 標準構成プロパティーの要約

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

表 2. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
ControllerStoreAndForwardMode	true または false	true	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ
DuplicateEventElimination	true または false	false	コンポーネント再始動	JMS トランスポートのみ、Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonicの場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ

表 2. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE> (ブローカーは ICS)
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
MessageFileName	パスまたはファイル名	CONNECTORNAMEConnector.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は true でなければならない

表 2. 標準構成プロパティーの要約 (続き)

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

表 2. 標準構成プロパティーの要約 (続き)

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

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

AgentConnections プロパティーは、orb.init[] により開かれる ORB (オブジェクト・リクエスト・ブローカー) 接続の数を制御します。

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

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

ConcurrentEventTriggeredFlows

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

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

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

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

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

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

ContainerManagedEvents

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

デフォルト値はありません。

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

- PollQuantity = 1 から 500
- SourceQueue = /SOURCEQUEUE

また、MimeType、DHClass (データ・ハンドラー・クラス)、および DataHandlerConfigMOName (オプションのメタオブジェクト名) プロパティーを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティーの値を設定するには、Connector Configurator の**「データ・ハンドラー」**タブを使用します。

これらのプロパティーはアダプター固有ですが、**例**の値は次のようになります。

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

「データ・ハンドラー」タブのこれらの値のフィールドは、ContainerManagedEvents を JMS に設定した場合にのみ表示されます。

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

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

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

JMS

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

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

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

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

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

- CWSharedEnv.sh スクリプト内で LDR_CNTRL 環境変数を設定する。

  このスクリプトは、製品ディレクトリー配下の ¥bin ディレクトリーにあります。テキスト・エディターを使用して、CWSharedEnv.sh スクリプトの最初の行として次の行を追加します。

  ```
  export LDR_CNTRL=MAXDATA=0x30000000
  ```

  この行は、ヒープ・メモリーの使用量を最大 768 MB (3 セグメント * 256 MB) に制限します。プロセス・メモリーがこの制限値を超えると、ページ・スワッピングが発生し、システムのパフォーマンスに悪影響を与える場合があります。
- IPCCBaseAddress プロパティーの値を 11 または 12 に設定する。このプロパティーの詳細については、「*システム・インストール・ガイド (UNIX 版)*」を参照してください。

DuplicateEventElimination

このプロパティーを true に設定すると、JMS 対応コネクターによるデリバリー・キューへの重複イベントのデリバリーが防止されます。この機能を使用するには、コネクターに対し、アプリケーション固有のコード内でビジネス・オブジェクトの **ObjectEventId** 属性として一意のイベント ID が設定されている必要があります。これはコネクター開発時に設定されます。

このプロパティーは、false に設定することもできます。

注: DuplicateEventElimination を true に設定する際は、MonitorQueue プロパティーを構成して保証付きイベント・デリバリーを使用可能にする必要があります。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、そのメッセージを状況表示および問題説明とともにこのプロパティーに指定されているキューに移動します。

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128M です。

JvmMaxNativeStackSize

エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 128K です。

JvmMinHeapSize

エージェントの最小ヒープ・サイズ (メガバイト単位)。このプロパティーは、RepositoryDirectory の値が <REMOTE> の場合にのみ適用されます。

デフォルト値は 1M です。

jms.FactoryClassName

JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクター・プロパティーを必ず 設定してください。

デフォルト値は CxCommon.Messaging.jms.IBMMQSeriesFactory です。

jms.MessageBrokerName

JMS プロバイダーのために使用するブローカー名を指定します。JMS をデリバリー・トランスポート機構 (DeliveryTransport) として選択する際は、このコネクター・プロパティーを必ず 設定してください。

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

リモート・メッセージ・ブローカーに接続すると、このプロパティーは次の (必須) 値をとります。
QueueMgrName:<Channel>:<HostName>:<PortNumber>
各変数の意味は以下のとおりです。
QueueMgrName: キュー・マネージャー名です。
Channel: クライアントが使用するチャネルです。
HostName: キュー・マネージャーの配置先のマシン名です。
PortNumber: キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のように指定します。

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

jms.NumConcurrentRequests

コネクターに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出し要求はブロックされ、既存のいずれかの要求が完了した後で処理されます。

デフォルト値は 10 です。

jms.Password

JMS プロバイダーのためのパスワードを指定します。このプロパティーの値はオプションです。

デフォルトはありません。

jms.UserName

JMS プロバイダーのためのユーザー名を指定します。このプロパティーの値はオプションです。

デフォルトはありません。

ListenerConcurrency

このプロパティーは、統合ブローカーとして ICS を使用する場合の MQ Listener でのマルチスレッド化をサポートしています。このプロパティーにより、データベースへの複数イベントの書き込み操作をバッチ処理できるので、システム・パフォーマンスが向上します。デフォルト値は 1 です。

このプロパティーは、MQ トランスポートを使用するコネクターにのみ適用されます。`DeliveryTransport` プロパティーには `MQ` を設定してください。

Locale

言語コード、国または地域、および、希望する場合には、関連した文字コード・セットを指定します。このプロパティーの値は、データの照合やソート順、日付と時刻の形式、通貨記号などの国/地域別情報を決定します。

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、

ll	2 文字の言語コード (普通は小文字)
TT	2 文字の国または地域コード (普通は大文字)
codeset	関連文字コード・セットの名前。名前のこの部分は、通常、オプションです。

デフォルトでは、ドロップダウン・リストには、サポートされるロケールの一部のみが表示されます。ドロップダウン・リストに、サポートされる他の値を追加する

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

デフォルト値は en_US です。コネクターがグローバル化に対応していない場合、このプロパティーの有効な値は en_US のみです。特定のコネクターがグローバル化に対応しているかどうかを判別するには、以下の Web サイトにあるコネクターのバージョン・リストを参照してください。

http://www.ibm.com/software/websphere/wbiadapters/infocenter、または
http://www.ibm.com/websphere/integration/wicserver/infocenter

LogAtInterchangeEnd

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

統合ブローカーのログ宛先にエラーを記録するかどうかを指定します。ブローカーのログ宛先にログを記録すると、電子メール通知もオンになります。これにより、エラーまたは致命的エラーが発生すると、InterchangeSystem.cfg ファイルに指定された MESSAGE_RECIPIENT に対する電子メール・メッセージが生成されます。

例えば、LogAtInterChangeEnd を true に設定した場合にコネクターからアプリケーションへの接続が失われると、指定されたメッセージ宛先に、電子メール・メッセージが送信されます。デフォルト値は false です。

MaxEventCapacity

コントローラー・バッファー内のイベントの最大数。このプロパティーはフロー制御が使用し、RepositoryDirectory プロパティーの値が <REMOTE> の場合にのみ適用されます。

値は 1 から 2147483647 の間の正整数です。デフォルト値は 2147483647 です。

MessageFileName

コネクター・メッセージ・ファイルの名前です。メッセージ・ファイルの標準位置は、製品ディレクトリーの ¥connectors¥messages です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクター・メッセージ・ファイルが存在しない場合は、コネクターは InterchangeSystem.txt をメッセージ・ファイルとして使用します。このファイルは、製品ディレクトリーに格納されています。

注: 特定のコネクターについて、コネクター独自のメッセージ・ファイルがあるかどうかを判別するには、該当するアダプターのユーザーズ・ガイドを参照してください。

MonitorQueue

コネクターが重複イベントをモニターするために使用する論理キューです。このプロパティーは、DeliveryTransport プロパティー値が JMS であり、かつ DuplicateEventElimination が TRUE に設定されている場合にのみ使用されます。

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

RepositoryDirectory が <REMOTE> の場合のみ有効です。

コネクターが自動再始動およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用して、異常シャットダウン後にコネクターを再始動したり、System Monitor からリモート・コネクターを始動したりします。

自動再始動機能およびリモート再始動機能を使用可能にするには、このプロパティーを true に設定する必要があります。MQ により起動される OAD 機能の構成方法については、「システム・インストール・ガイド *(Windows 版)*」または「システム・インストール・ガイド *(UNIX 版)*」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

RepositoryDirectory が <REMOTE> の場合のみ有効です。

異常シャットダウンの後で MQ により起動される OAD がコネクターの再始動を自動的に試行する回数の最大数を指定します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

RepositoryDirectory が <REMOTE> の場合のみ有効です。

MQ により起動される OAD の再試行時間間隔の分数を指定します。コネクター・エージェントがこの再試行時間間隔内に再始動しない場合は、コネクター・コントローラーはコネクター・エージェントを再び再始動するように OAD に要求します。OAD はこの再試行プロセスを OADMaxNumRetry プロパティーで指定された回数だけ繰り返します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを true に設定する必要があります。

デフォルト値は 10 です。

PollEndTime

イベント・キューのポーリングを停止する時刻です。形式は HH:MM です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

PollFrequency

これは、前回のポーリングの終了から次のポーリングの開始までの間の間隔です。PollFrequency は、あるポーリング・アクションの終了から次のポーリング・アク

ションの開始までの時間をミリ秒単位で指定します。これはポーリング・アクション間の間隔ではありません。この論理を次に説明します。

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

PollFrequency は以下の値のいずれかに設定します。

- ポーリング・アクション間のミリ秒数 (整数)。
- ワード key。コネクターは、コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力されたときにのみポーリングを実行します。このワードは小文字で入力します。
- ワード no。コネクターはポーリングを実行しません。このワードは小文字で入力します。

デフォルト値は 10000 です。

重要: 一部のコネクターでは、このプロパティーの使用が制限されています。このようなコネクターが存在する場合には、アダプターのインストールと構成に関する章で制約事項が説明されています。

PollQuantity

コネクターがアプリケーションからポーリングする項目の数を指定します。アダプターにコネクター固有のポーリング数設定プロパティーがある場合、標準プロパティーの値は、このコネクター固有のプロパティーの設定値によりオーバーライドされます。

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

コネクターは、1 回目のポーリングを受けると、メッセージの本文を選出します。これは、本文が添付とも見なされるからです。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターは本文を無視します。
コネクターは PO の最初の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
2 回目のポーリングを受けると、コネクターは PO の 2 番目の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
これが受け入れられると、PO の 3 番目の添付が届きます。

PollStartTime

イベント・キューのポーリングを開始する時刻です。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

このプロパティーには必ず有効な値を指定してください。デフォルト値は HH:MM ですが、この値は必ず変更する必要があります。

RequestQueue

統合ブローカーが、ビジネス・オブジェクトをコネクターに送信するときに使用されるキューです。

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

コネクターが XML スキーマ文書を読み取るリポジトリーの場所です。この XML スキーマ文書には、ビジネス・オブジェクト定義のメタデータが含まれています。

統合ブローカーが ICS の場合はこの値を <REMOTE> に設定する必要があります。これは、コネクターが InterChange Server リポジトリーからこの情報を取得するためです。

統合ブローカーが WebSphere Message Broker または WAS の場合は、この値を <*local directory*> に設定する必要があります。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用可能で、RepositoryDirectory が <REMOTE> の場合のみ必須です。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・フレームワークから統合ブローカーへデリバリーします。統合ブローカーが ICS の場合、サーバーは要求を送信し、JMS 応答キューの応答メッセージを待ちます。

RestartRetryCount

コネクターによるコネクター自体の再始動の試行回数を指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する回数が指定されます。

デフォルト値は 3 です。

RestartRetryInterval

コネクターによるコネクター自体の再始動の試行間隔を分単位で指定します。このプロパティーを並列コネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがスレーブ側のアプリケーション固有のコンポーネントの再始動を試行する間隔が指定されます。指定可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

WebSphere Message Brokers および WAS でのみ使用されます。

このプロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WMQI に送信するときに、アダプター・フレームワークにより JMS ヘッダー情報、ドメイン名、および固定値 mrm が書き込まれます。この構成可能なドメイン名により、ユーザーは WMQI ブローカーによるメッセージ・データの処理方法を追跡できます。

サンプル・ヘッダーを以下に示します。

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

デフォルト値は mrm ですが、このプロパティーには xml も設定できます。このプロパティーは、DeliveryTransport が JMS に設定されており、かつ WireFormat が CwXML に設定されている場合にのみ表示されます。

SourceQueue

DeliveryTransport が JMS で、ContainerManagedEvents が指定されている場合のみ適用されます。

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、59 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期応答を要求する要求メッセージを、コネクター・フレームワークからブローカーに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。同期実行の場合、コネクター・フレームワークは、SynchronousRequestQueue にメッセージを送信し、SynchronousResponseQueue でブローカーから戻される応答を待機します。コネクターに送信される応答メッセージには、元のメッセージの ID を指定する相関 ID が含まれています。

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

同期要求に対する応答として送信される応答メッセージを、ブローカーからコネクター・フレームワークに配信します。このキューは、コネクターが同期実行を使用する場合にのみ必要です。

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

- RepositoryDirectory がローカル・ディレクトリーの場合は、設定は CwXML になります。
- RepositoryDirectory の値が <REMOTE> の場合には、設定値は CwBO です。

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

コネクターが同期要求への応答を待機する時間を分単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージをエラー・メッセージとともに障害キューに移動します。

デフォルト値は 0 です。

XMLNameSpaceFormat

WebSphere Message Brokers および WAS 統合ブローカーでのみ使用されます。

ビジネス・オブジェクト定義の XML 形式でネーム・スペースを short と long のどちらにするかをユーザーが指定できるようにするための、強力なプロパティーです。

デフォルト値は short です。

付録 B. Connector Configurator

この付録では、Connector Configurator を使用してアダプターの構成プロパティー値を設定する方法について説明します。

Connector Configurator を使用して次の作業を行います。

- コネクターを構成するためのコネクター固有のプロパティー・テンプレートを作成する
- 構成ファイルを作成する
- 構成ファイル内のプロパティーを設定する

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

Connector Configurator では、次の統合ブローカーで使用するアダプターのコネクター・コンポーネントを構成できます。

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

Connector Configurator を使用して次の作業を行います。

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成します。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定します。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、ICS の場合はコラボレーションとともに使用するマップを

指定し、必要に応じてメッセージング、ロギング、トレース、およびデータ・ハンドラー・パラメーターを指定する必要があります。

Connector Configurator の実行モードと使用する構成ファイルのタイプは、実行する統合ブローカーによって異なります。例えば、使用している統合ブローカーが WMQI の場合、Connector Configurator を System Manager から実行するのではなく、直接実行します (『スタンドアロン・モードでの Configurator の実行』を参照)。

コネクター構成プロパティーには、標準の構成プロパティー (すべてのコネクターがもつプロパティー) と、コネクター固有のプロパティー (特定のアプリケーションまたはテクノロジーのためにコネクターで必要なプロパティー) とが含まれます。

標準プロパティーはすべてのコネクターにより使用されるので、標準プロパティーを新規に定義する必要はありません。ファイルを作成すると、Connector Configurator により標準プロパティーがこの構成ファイルに挿入されます。ただし、Connector Configurator で各標準プロパティーの値を設定する必要があります。

標準プロパティーの範囲は、ブローカーと構成によって異なる可能性があります。特定のプロパティーに特定の値が設定されている場合にのみ使用できるプロパティーがあります。Connector Configurator の「標準のプロパティー」ウィンドウには、特定の構成で設定可能なプロパティーが表示されます。

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、74 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

注: Connector Configurator は、Windows 環境内でのみ実行されます。UNIX 環境でコネクターを実行する場合には、Windows で Connector Configurator を使用して構成ファイルを変更し、このファイルを UNIX 環境へコピーします。

Connector Configurator の始動

以下の 2 種類のモードで Connector Configurator を開始および実行できます。

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

スタンドアロン・モードでの Configurator の実行

どのブローカーを実行している場合にも、Connector Configurator を個別に実行し、コネクター構成ファイルを編集できます。

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Tools」>「Connector Configurator」**をクリックします。
- **「ファイル」>「新規」>「コネクター構成」**を選択します。

- **「システム接続: Integration Broker」**の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (79 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

Connector Configurator を実行するには、以下のステップを実行します。

1. System Manager を開きます。
2. 「System Manager」ウィンドウで、**「統合コンポーネント・ライブラリー」**アイコンを展開し、**「コネクター」**を強調表示します。
3. System Manager メニュー・バーから、**「ツール」>「Connector Configurator」**をクリックします。「Connector Configurator」ウィンドウが開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。
4. **「システム接続: Integration Broker」**の隣のプルダウン・メニューをクリックします。使用しているブローカーに応じて、ICS、WebSphere Message Brokers、または WAS を選択します。

既存の構成ファイルを編集するには、以下のステップを実行します。

- 「System Manager」ウィンドウの「コネクター」フォルダーでいずれかの構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
- Connector Configurator で**「ファイル」>「開く」**を選択します。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクター構成ファイルを選択します。
- 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

コネクター固有プロパティーのテンプレートを新規に作成するか、または既存のコネクター定義をテンプレートとして使用します。

- テンプレートの新規作成については、74 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。既存のテンプレートは ¥WebSphereAdapters¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティーを作成し、プロパティーの一般特性および値を定義し、プロパティー間の依存関係を指定する方法について説明します。次にそのテンプレートを保管し、新規コネクター構成ファイルを作成するためのベースとして使用します。

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

1. **「ファイル」>「新規」>「コネクター固有プロパティー・テンプレート」**をクリックします。
2. **「コネクター固有プロパティー・テンプレート」** ダイアログ・ボックスが表示されます。
   - **「新規テンプレート名を入力してください」**の下の**「名前」**フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、**「テンプレート名」**表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが**「テンプレートのプレビュー」**表示に表示されます。
3. テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。ご使用のコネクターで使用するコネクター固有のプロパティーが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。
   - 既存のテンプレートを変更する場合には、**「変更する既存のテンプレートを選択してください: 検索テンプレート」**の下の**「テンプレート名」**テーブルのリストから、テンプレート名を選択します。
   - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

「次へ」をクリックしてテンプレートを選択すると、**「プロパティー: コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスには、定義済みプロパティーの「一般」特性のタブと「値」の制限のタブがあります。「一般」表示には以下のフィールドがあります。

- **一般:**
  プロパティー・タイプ
  更新されたメソッド
  説明
- **フラグ**
  標準フラグ
- **カスタム・フラグ**
  フラグ

プロパティーの一般特性の選択を終えたら、**「値」**タブをクリックします。

値の指定

「値」タブを使用すると、プロパティーの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。編集可能な値も許可されます。これを行うには、以下のステップを実行します。

1. **「値」**タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドに値を入力します。

新規プロパティー値を作成するには、以下のステップを実行します。

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

「値」パネルには、3 つの列からなるテーブルが表示されます。

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

「デフォルト値」の列では、値のいずれかをデフォルトとして指定することができます。

「値の範囲」の列には、**「プロパティー値」**ダイアログ・ボックスで入力した範囲が表示されます。

値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。

テーブルにある既存の値の変更を行うには、その行の行番号をクリックして行全体を選択します。次に**「値」**フィールドを右マウス・ボタンでクリックし、**「値の編集 (Edit Value)」**をクリックします。

依存関係の設定

「一般」タブと**「値」**タブで変更を行ったら、**「次へ」**をクリックします。**「依存関係: コネクター固有プロパティー・テンプレート」**ダイアログ・ボックスが表示されます。

依存プロパティーは、別のプロパティーの値が特定の条件に合致する場合にのみ、テンプレートに組み込まれて、構成ファイルで使用されるプロパティーです。例えば、テンプレートに `PollQuantity` が表示されるのは、トランスポート機構が JMS であり、`DuplicateEventElimination` が `True` に設定されている場合のみです。プロパティーを依存プロパティーとして指定し、依存する条件を設定するには、以下のステップを実行します。

1. **「使用可能なプロパティー」**表示で、依存プロパティーとして指定するプロパティーを選択します。

2. **「プロパティーを選択」**フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティーを選択します。
3. **「条件演算子」**フィールドで以下のいずれかを選択します。

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

   <= (より小か等しい)
4. **「条件値」**フィールドで、依存プロパティーをテンプレートに組み込むために必要な値を入力します。
5. **「使用可能なプロパティー」**表示で依存プロパティーを強調表示させて矢印をクリックし、**「依存プロパティー」**表示に移動させます。
6. **「完了」**をクリックします。Connector Configurator により、XML 文書として入力した情報が、Connector Configurator がインストールされている `¥bin` ディレクトリーの `¥data¥app` の下に保管されます。

新規構成ファイルの作成

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

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

また、統合ブローカーも選択する必要があります。選択したブローカーによって、構成ファイルに記述されるプロパティーが決まります。ブローカーを選択するには、以下のステップを実行します。

- **「Integration Broker」**フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って**「新規コネクター」**ウィンドウの残りのフィールドに入力します。

コネクター固有のテンプレートからの構成ファイルの作成

コネクター固有のテンプレートを作成すると、テンプレートを使用して構成ファイルを作成できます。

1. **「ファイル」>「新規」>「コネクター構成」**をクリックします。
2. 以下のフィールドを含む**「新規コネクター」**ダイアログ・ボックス表示されます。

- **名前**

  コネクターの名前を入力します。名前では大文字と小文字が区別されます。入力する名前は、システムにインストールされているコネクターのファイル名に対応した一意の名前でなければなりません。

  重要: Connector Configurator では、入力された名前のスペルはチェックされません。名前が正しいことを確認してください。

- **システム接続**

  ICS 接続、WebSphere Message Brokers 接続、WAS のいずれかをクリックします。

- **「コネクター固有プロパティー・テンプレート」を選択します。**

  ご使用のコネクター用に設計したテンプレートの名前を入力します。**「テンプレート名」**表示に、使用可能なテンプレートが表示されます。「テンプレート名」表示で名前を選択すると、**「プロパティー・テンプレートのプレビュー」**表示に、そのテンプレートで定義されているコネクター固有のプロパティーが表示されます。

  使用するテンプレートを選択し、**「OK」**をクリックします。

3. 構成しているコネクターの構成画面が表示されます。タイトル・バーに統合ブローカーとコネクターの名前が表示されます。ここですべてのフィールドに値を入力して定義を完了するか、ファイルを保管して後でフィールドに値を入力するかを選択できます。
4. ファイルを保管するには、**「ファイル」>「保管」>「ファイルに」**をクリックするか、**「ファイル」>「保管」>「プロジェクトに」**をクリックします。プロジェクトに保管するには、System Manager が実行中でなければなりません。
   ファイルとして保管する場合は、**「ファイル・コネクターを保管」**ダイアログ・ボックスが表示されます。`*.cfg` をファイル・タイプとして選択し、「ファイル名」フィールド内に名前が正しいスペル (大文字と小文字の区別を含む) で表示されていることを確認してから、ファイルを保管するディレクトリーにナビゲートし、**「保管」**をクリックします。Connector Configurator のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

   重要: ここで設定するディレクトリー・パスおよび名前は、コネクターの始動ファイルで指定するコネクター構成ファイルのパスおよび名前に一致している必要があります。
5. この章で後述する手順に従って、「Connector Configurator」ウィンドウの各タブにあるフィールドに値を入力し、コネクター定義を完了します。

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

- コネクター定義ファイル。
  コネクター定義ファイルは、特定のコネクターのプロパティーと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクターの配布パッケー

ジの ¥repository ディレクトリー内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は .txt です。例えば、XML コネクターの場合は CN_XML.txt です)。

- ICS リポジトリー・ファイル。
  コネクターの以前の ICS インプリメンテーションで使用した定義は、そのコネクターの構成で使用されたリポジトリー・ファイルで使用可能になります。そのようなファイルの拡張子は、通常 .in または .out です。
- コネクターの以前の構成ファイル。
  これらのファイルの拡張子は、通常 *.cfg です。

これらのいずれのファイル・ソースにも、コネクターのコネクター固有プロパティーのほとんど、あるいはすべてが含まれますが、この章内の後で説明するように、コネクター構成ファイルは、ファイルを開いて、プロパティーを設定しない限り完成しません。

既存ファイルを使用してコネクターを構成するには、Connector Configurator でそのファイルを開き、構成を修正し、そのファイルを再度保管する必要があります。

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

1. Connector Configurator 内で、**「ファイル」>「開く」>「ファイルから」**をクリックします。
2. **「ファイル・コネクターを開く」**ダイアログ・ボックス内で、以下のいずれかのファイル・タイプを選択して、使用可能なファイルを調べます。
   - 構成 (*.cfg)
   - ICS リポジトリー (*.in、*.out)

     ICS 環境でのコネクターの構成にリポジトリー・ファイルが使用された場合には、このオプションを選択します。リポジトリー・ファイルに複数のコネクター定義が含まれている場合は、ファイルを開くとすべての定義が表示されます。
   - すべてのファイル (*.*)

     コネクターのアダプター・パッケージに *.txt ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。
3. ディレクトリー表示内で、適切なコネクター定義ファイルへ移動し、ファイルを選択し、**「開く」**をクリックします。

System Manager プロジェクトからコネクター構成を開くには、以下のステップを実行します。

1. System Manager を始動します。System Manager が開始されている場合にのみ、構成を System Manager から開いたり、System Manager に保管したりできます。
2. Connector Configurator を始動します。
3. **「ファイル」>「開く」>「プロジェクトから」**をクリックします。

構成ファイルの完成

構成ファイルを開くか、プロジェクトからコネクターを開くと、「Connector Configurator」ウィンドウに構成画面が表示されます。この画面には、現在の属性と値が表示されます。

構成画面のタイトルには、ファイル内で指定された統合ブローカーとコネクターの名前が表示されます。正しいブローカーが設定されていることを確認してください。正しいブローカーが設定されていない場合、コネクターを構成する前にブローカー値を変更してください。これを行うには、以下のステップを実行します。

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 82 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

   ファイルに保管する場合は、*.cfg を拡張子として選択し、ファイルの正しい格納場所を選択して、**「保管」**をクリックします。

   複数のコネクター構成を開いている場合、構成をすべてファイルに保管するには**「すべてファイルに保管」**を選択し、コネクター構成をすべて System Manager プロジェクトに保管するには**「すべてプロジェクトに保管」**をクリックします。

   Connector Configurator では、ファイルを保管する前に、必須の標準プロパティーすべてに値が設定されているかどうかが確認されます。必須の標準プロパティーに値が設定されていない場合、Connector Configurator は、検証が失敗したというメッセージを表示します。構成ファイルを保管するには、そのプロパティーの値を指定する必要があります。

構成ファイル・プロパティーの設定

新規のコネクター構成ファイルを作成して名前を付けるとき、または既存のコネクター構成ファイルを開くときには、Connector Configurator によって構成画面が表示されます。構成画面には、必要な構成値のカテゴリーに対応する複数のタブがあります。

Connector Configurator では、すべてのブローカーで実行されているコネクターで、以下のカテゴリーのプロパティーに値が設定されている必要があります。

- 標準のプロパティー
- コネクター固有のプロパティー
- サポートされるビジネス・オブジェクト
- トレース/ログ・ファイルの値
- データ・ハンドラー (保証付きイベント・デリバリーで JMS メッセージングを使用するコネクターの場合に該当する)

注: JMS メッセージングを使用するコネクターの場合は、データをビジネス・オブジェクトに変換するデータ・ハンドラーの構成に関して追加のカテゴリーが表示される場合があります。

ICS で実行されているコネクターの場合、以下のプロパティーの値も設定されている必要があります。

- 関連付けられたマップ
- リソース
- メッセージング (該当する場合)

重要: Connector Configurator では、英語文字セットまたは英語以外の文字セットのいずれのプロパティー値も設定可能です。ただし、標準のプロパティーおよびコネクター固有プロパティー、およびサポートされるビジネス・オブジェクトの名前では、英語文字セットのみを使用する必要があります。

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

- コネクターの標準プロパティーは、コネクターのアプリケーション固有のコンポーネントとブローカー・コンポーネントの両方によって共用されます。すべてのコネクターが同じ標準プロパティーのセットを使用します。これらのプロパティーの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。
- アプリケーション固有のプロパティーは、コネクターのアプリケーション固有コンポーネント (アプリケーションと直接対話するコンポーネント) のみに適用されます。各コネクターには、そのコネクターのアプリケーションだけで使用されるアプリケーション固有のプロパティーがあります。これらのプロパティーには、デフォルト値が用意されているものもあれば、そうでないものもあります。また、一部のデフォルト値は変更することができます。各アダプター・ガイドのインストールおよび構成の章に、アプリケーション固有のプロパティーおよび推奨値が記述されています。

「標準プロパティー」と**「コネクター固有プロパティー」**のフィールドは、どのフィールドが構成可能であるかを示すために色分けされています。

- 背景がグレーのフィールドは、標準のプロパティーを表します。値を変更することはできますが、名前の変更およびプロパティーの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティーを表します。これらのプロパティーは、アプリケーションまたはコネクターの特定のニーズによって異なります。値の変更も、これらのプロパティーの除去も可能です。
- 「値」フィールドは構成できます。
- プロパティーごとに**「更新メソッド」**フィールドが表示されます。これは、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。この設定を構成することはできません。

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。

2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。
3. 標準のプロパティーの値をすべて入力後、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator を終了するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「いいえ」**をクリックします。
   - Connector Configurator 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。**「標準のプロパティー」**(またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
   - 修正した値を保管するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「はい」**をクリックします。「ファイル」メニューまたはツールバーから**「保管」>「ファイルに」**をクリックする方法もあります。

アプリケーション固有の構成プロパティーの設定

アプリケーション固有の構成プロパティーの場合、プロパティー名の追加または変更、値の構成、プロパティーの削除、およびプロパティーの暗号化が可能です。プロパティーのデフォルトの長さは 255 文字です。

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。
4. 80 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

各プロパティーごとに表示される「更新メソッド」は、変更された値をアクティブにするためにコンポーネントまたはエージェントの再始動が必要かどうかを示します。

重要: 事前設定のアプリケーション固有のコネクター・プロパティー名を変更すると、コネクターに障害が発生する可能性があります。コネクターをアプリケーションに接続したり正常に実行したりするために、特定のプロパティー名が必要である場合があります。

コネクター・プロパティーの暗号化

「コネクター固有プロパティー」ウィンドウの**「暗号化」**チェック・ボックスにチェックマークを付けると、アプリケーション固有のプロパティーを暗号化することができます。値の暗号化を解除するには、**「暗号化」**チェック・ボックスをクリックしてチェックマークを外し、**「検証」**ダイアログ・ボックスに正しい値を入力し、**「OK」**をクリックします。入力された値が正しい場合は、暗号化解除された値が表示されます。

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

プロパティーに複数の値がある場合には、プロパティーの最初の値に**「暗号化」**チェック・ボックスが表示されます。**「暗号化」**を選択すると、そのプロパティーのすべての値が暗号化されます。プロパティーの複数の値を暗号化解除するには、そのプロパティーの最初の値の**「暗号化」**チェック・ボックスをクリックしてチェックマークを外してから、**「検証」**ダイアログ・ボックスで新規の値を入力します。入力値が一致すれば、すべての複数値が暗号化解除されます。

更新メソッド

付録 A『*コネクターの標準構成プロパティー*』の 52 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

コネクターで使用するビジネス・オブジェクトを指定するには、Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブを使用します。汎用ビジネス・オブジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

注: コネクターによっては、アプリケーションでイベント通知や (メタオブジェクトを使用した) 追加の構成を実行するために、特定のビジネス・オブジェクトをサポートされているものとして指定することが必要な場合もあります。詳細は、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

ご使用のブローカーが ICS の場合

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

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

1. **「ビジネス・オブジェクト名」**リストで空のフィールドをクリックします。System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの**「エージェント・サポート」**(以下で説明) を設定します。
4. 「Connector Configurator」ウィンドウの「ファイル」メニューで、**「プロジェクトに保管」**をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

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

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

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

エージェント・サポート: ビジネス・オブジェクトがエージェント・サポートを備えている場合、システムは、コネクター・エージェントを介してアプリケーションにデータを配布する際にそのビジネス・オブジェクトの使用を試みます。

一般に、コネクターのアプリケーション固有ビジネス・オブジェクトは、そのコネクターのエージェントによってサポートされますが、汎用ビジネス・オブジェクトはサポートされません。

ビジネス・オブジェクトがコネクター・エージェントによってサポートされるよう指定するには、**「エージェント・サポート」**ボックスにチェックマークを付けます。「Connector Configurator」ウィンドウでは「エージェント・サポート」の選択の妥当性は検査されません。

最大トランザクション・レベル: コネクターの最大トランザクション・レベルは、そのコネクターがサポートする最大のトランザクション・レベルです。

ほとんどのコネクターの場合、選択可能な項目は「**最大限の努力**」のみです。

トランザクション・レベルの変更を有効にするには、サーバーを再始動する必要があります。

ご使用のブローカーが WebSphere Message Broker の場合

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス・オブジェクト名を入力する必要があります。

System Manager を実行している場合、**「サポートされているビジネス・オブジェクト」**タブの**「ビジネス・オブジェクト名」**列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。リストから必要なビジネス・オブジェクトを選択します。

「メッセージ・セット ID」は、WebSphere Business Integration Message Broker 5.0 のオプションのフィールドです。この ID が提供される場合、一意である必要はありません。ただし、WebSphere MQ Integrator および Integrator Broker 2.1 の場合は、一意の **ID** を提供する必要があります。

ご使用のブローカーが WAS の場合

使用するブローカー・タイプとして WebSphere Application Server を選択した場合、Connector Configurator にメッセージ・セット ID は必要ありません。**「サポートされているビジネス・オブジェクト」**タブには、サポートされるビジネス・オブジェクトの**「ビジネス・オブジェクト名」**列のみが表示されます。

スタンドアロン・モードで作業している (System Manager に接続していない) 場合、手動でビジネス・オブジェクト名を入力する必要があります。

System Manager を実行している場合、「サポートされているビジネス・オブジェクト」タブの「ビジネス・オブジェクト名」列の下にある空のボックスを選択できます。コンボ・ボックスが表示され、コネクターが属する統合コンポーネント・ライブラリー・プロジェクトから選択可能なビジネス・オブジェクトのリストが示されます。このリストから必要なビジネス・オブジェクトを選択します。

関連付けられているマップ (ICS のみ)

各コネクターは、現在 WebSphere InterChange Server でアクティブなビジネス・オブジェクト定義、およびそれらの関連付けられたマップのリストをサポートします。このリストは、**「関連付けられたマップ」**タブを選択すると表示されます。

ビジネス・オブジェクトのリストには、エージェントでサポートされるアプリケーション固有のビジネス・オブジェクトと、コントローラーがサブスクライブ・コラボレーションに送信する、対応する汎用オブジェクトが含まれます。マップの関連によって、アプリケーション固有のビジネス・オブジェクトを汎用ビジネス・オブジェクトに変換したり、汎用ビジネス・オブジェクトをアプリケーション固有のビジネス・オブジェクトに変換したりするときに、どのマップを使用するかが決定されます。

特定のソースおよび宛先ビジネス・オブジェクトについて一意的に定義されたマップを使用する場合、表示を開くと、マップは常にそれらの該当するビジネス・オブジェクトに関連付けられます。ユーザーがそれらを変更する必要はありません (変更できません)。

サポートされるビジネス・オブジェクトで使用可能なマップが複数ある場合は、そのビジネス・オブジェクトを、使用する必要のあるマップに明示的にバインドすることが必要になります。

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

  これらは、**「サポートされているビジネス・オブジェクト」**タブで指定した、このコネクターでサポートされるビジネス・オブジェクトです。「サポートされているビジネス・オブジェクト」タブでビジネス・オブジェクトを追加指定した場合、その内容は、「Connector Configurator」ウィンドウの**「ファイル」**メニューから**「プロジェクトに保管」**を選択して、変更を保管した後に、このリストに反映されます。

- **関連付けられたマップ**

  この表示には、コネクターの、サポートされるビジネス・オブジェクトでの使用のためにシステムにインストールされたすべてのマップが示されます。各マップのソース・ビジネス・オブジェクトは、**「ビジネス・オブジェクト名」**表示でマップ名の左側に表示されます。

- **明示的**

  場合によっては、関連付けられたマップを明示的にバインドすることが必要になります。

  明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・オブジェクトにマップを自動的にバインドしようとします。複数のマップでその入力データとして同一のビジネス・オブジェクトが使用されている場合、サーバーは、他のマップのスーパーセットである 1 つのマップを見つけて、バインドしようとします。

  他のマップのスーパーセットであるマップがないと、サーバーは、ビジネス・オブジェクトを単一のマップにバインドすることができないため、バインディングを明示的に設定することが必要になります。

  以下の手順を実行して、マップを明示的にバインドします。

  1. **「明示的 (Explicit)」**列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
  2. ビジネス・オブジェクトに関連付けるマップを選択します。
  3. 「Connector Configurator」ウィンドウの**「ファイル」**メニューで、**「プロジェクトに保管」**をクリックします。
  4. プロジェクトを ICS に配置します。
  5. 変更を有効にするため、サーバーをリブートします。

リソース (ICS)

「リソース」タブでは、コネクター・エージェントが、コネクター・エージェント並列処理を使用して同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定できます。

すべてのコネクターがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

メッセージング・プロパティーは、`DeliveryTransport` 標準プロパティーの値として MQ を設定し、ブローカー・タイプとして ICS を設定した場合にのみ、使用可能です。これらのプロパティーは、コネクターによるキューの使用方法に影響します。

トレース/ログ・ファイル値の設定

コネクター構成ファイルまたはコネクター定義ファイルを開くと、Connector Configurator は、そのファイルのログおよびトレースの値をデフォルト値として使用します。Connector Configurator 内でこれらの値を変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

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

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

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

データ・ハンドラー

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

これらのプロパティーに使用する値については、『付録 A. コネクターの標準構成プロパティー』の ContainerManagedEvents の下の説明を参照してください。その他の詳細は、「*コネクター開発ガイド (C++ 用)*」または「*コネクター開発ガイド (Java 用)*」を参照してください。

構成ファイルの保管

コネクターの構成が完了したら、コネクター構成ファイルを保管します。Connector Configurator では、構成中に選択したブローカー・モードでファイルを保管します。Connector Configurator のタイトル・バーには現在のブローカー・モード (ICS、WMQI、または WAS) が常に表示されます。

ファイルは XML 文書として保管されます。XML 文書は次の 3 通りの方法で保管できます。

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- 指定したディレクトリーに保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。デフォルトでは、このファイルは ¥WebSphereAdapters¥bin¥Data¥App に保管されます。
- WebSphere Application Server プロジェクトをセットアップしている場合には、このファイルを WebSphere Application Server プロジェクトに保管することもできます。

System Manager でのプロジェクトの使用法、および配置の詳細については、以下のインプリメンテーション・ガイドを参照してください。

- ICS:「*WebSphere InterChange Server システム・インプリメンテーション・ガイド*」
- WebSphere Message Brokers: 「*WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド*」
- WAS:「*アダプター実装ガイド (WebSphere Application Server)*」

構成ファイルの変更

既存の構成ファイルの統合ブローカー設定を変更できます。これにより、他のブローカーで使用する構成ファイルを新規に作成するときに、このファイルをテンプレートとして使用できます。

注: 統合ブローカーを切り替える場合には、ブローカー・モード・プロパティーと同様に他の構成プロパティーも変更する必要があります。

既存の構成ファイルでのブローカーの選択を変更するには、以下の手順を実行します (オプション)。

- Connector Configurator で既存の構成ファイルを開きます。
- **「標準のプロパティー」**タブを選択します。
- 「標準のプロパティー」タブの**「BrokerType」**フィールドで、ご使用のブローカーに合った値を選択します。
  現行値を変更すると、プロパティー画面の利用可能なタブおよびフィールド選択がただちに変更され、選択した新規ブローカーに適したタブとフィールドのみが表示されます。

構成の完了

コネクターの構成ファイルを作成し、そのファイルを変更した後で、コネクターの始動時にコネクターが構成ファイルの位置を特定できるかどうかを確認してください。

これを行うには、コネクターが使用する始動ファイルを開き、コネクター構成ファイルに使用されている格納場所とファイル名が、ファイルに対して指定した名前およびファイルを格納したディレクトリーまたはパスと正確に一致しているかどうかを検証します。

グローバル化環境における Connector Configurator の使用

Connector Configurator はグローバル化され、構成ファイルと統合ブローカー間の文字変換を処理できます。 Connector Configurator では、ネイティブなエンコード方式を使用しています。構成ファイルに書き込む場合は UTF-8 エンコード方式を使用します。

Connector Configurator は、以下の場所で英語以外の文字をサポートします。

- すべての値のフィールド
- ログ・ファイルおよびトレース・ファイル・パス (**「トレース/ログ・ファイル」**タブで指定)

CharacterEncoding および Locale 標準構成プロパティーのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーの ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。

例えば、Locale プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太文字で示した行を追加してください。

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
                <ValidType>String</ValidType>
            <ValidValues>
                                <Value>ja_JP</Value>
                                <Value>ko_KR</Value>
                                <Value>zh_CN</Value>
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
                                <Value>it_IT</Value>
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>

                    <DefaultValue>en_US</DefaultValue>
            </ValidValues>
    </Property>
```

付録 C. 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。

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

商標

以下は、IBM Corporation の商標です。

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

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

MMX、Pentium および ProShare は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Adapter Framework V2.4