IBM WebSphere Business Integration Server Express and Express Plus

Adapter for JText ユーザーズ・ガイド

アダプター・バージョン 5.6.x

IBM WebSphere Business Integration Server Express and Express Plus

Adapter for JText ユーザーズ・ガイド

アダプター・バージョン 5.6.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、 135 ページの『特記事項』に記載されている情報をお読みください。

本書は、WebSphere Business Integration Server Express Adapter for JText (5724-H03) バージョン 5.6.x、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： WebSphere Business Integration
Server Express and Express Plus
Adapter for JText User Guide
Adapter Version 5.6.x

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2005.8

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

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

目次

本書について v
本書の内容 v
本書の対象範囲外 v
対象読者 v
関連資料 vi
表記上の規則 vi

本リリースの新機能 vii
バージョン 5.6.x vii

第 1 章 JText アダプターの概要 1
アダプター・コンポーネント 2
JText コネクターが使用するビジネス・オブジェクト 3
コネクターの動作 7
コネクターの機能 15
ロケール依存データの処理 16

第 2 章 JText アダプターのインストール 17
インストール・タスクの概要 17
アダプター環境 17
JText アダプターのインストール 19
インストールの検証 19

第 3 章 JText アダプターの構成 23
Connector Configurator Express の概要 23
Connector Configurator Express の始動 24
System Manager からのコンフィギュレーターの実行 25
コネクター固有のプロパティー・テンプレートの作成 25
新規構成ファイルの作成 29
既存ファイルの使用 30
構成ファイルの完成 31
構成ファイル・プロパティーの設定 32
構成ファイルの保管 43
構成ファイルの変更 44
構成の完了 44
グローバル化環境における Connector Configurator Express の使用 44
コネクターの始動 45
コネクターの停止 47
複数のコネクター・インスタンスの作成 47
サポートされるビジネス・オブジェクトの追加 50

第 4 章 JText コネクター・メタオブジェクトの使用 51
JText メタオブジェクトの命名規則 52
JText メタオブジェクトの構造 52
共通の構成タスク 66

第 5 章 JText コネクターのトラブルシューティング 93
エラー・メッセージのロギング 93
メタオブジェクトの名前に関する問題 93
イベントの起動に関する問題 94

JText での障害の処理 94

第 6 章 JText コネクターへのマイグレーションまたはアップグレード 101
アップグレード・シナリオ 101
バージョン 5.3.x から 5.6.x へのアップグレード 101
バージョン 3.2.0 から 4.0.x にアップグレードする理由 102
バージョン 4.0.x へのアップグレード 103
Text コネクターをアップグレードする理由 104
JText コネクターへのアップグレード 105

付録. コネクターの標準構成プロパティー 107
新規プロパティー 107
標準コネクター・プロパティーの概要 107
標準プロパティーの早見表 109
標準プロパティー 115

索引 133

特記事項. 135
プログラミング・インターフェース情報 136
商標 137

本書について

製品 IBM(R) WebSphere(R) Business Integration Server Express および IBM WebSphere Business Integration Server Express Plus は、InterChange Server Express、関連する Toolset Express、CollaborationFoundation、およびソフトウェア統合アダプターのセットで構成されています。Toolset Express に含まれるツールは、ビジネス・オブジェクトの作成、変更、および管理に役立ちます。プリパッケージされている各種アダプターは、お客様の複数アプリケーションにまたがるビジネス・プロセスに応じて、いずれかを選べるようになっています。標準的なプロセス・テンプレート (CollaborationFoundation) により、カスタマイズ・プロセスをすばやく作成できます。

本書では、WebSphere Business Integration Server Express JText Adapter(TM) 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。

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

本書の内容

本書では、WebSphere Business Integration Server Express JText Adapter のインストール、コネクター・プロパティー構成、ビジネス・オブジェクト開発、およびトラブルシューティングについて説明します。

本書の対象範囲外

本書では、サーバーのロード・バランシング、アダプター処理スレッドの数、最大および最小スループット、および許容しきい値など、配置のメトリックやキャパシティー・プランニング問題については説明しません。

そうした問題は、お客様の配置方法に固有のものであり、アダプターが配置される実際の環境またはそれに近い環境で測定する必要があります。お客様の配置サイトの構成についての質問や、この種のメトリックの計画と評価について詳しくは、お客様固有の構成を明らかにした上で、IBM サービス技術員にお問い合わせください。

対象読者

本書は、WebSphere のコンサルタントとカスタマーを対象として書かれています。使用している統合ブローカーおよびビジネス・オブジェクトの開発に関する基礎知識、さらにはデータ・ハンドラーの開発についての知識も必要です。

関連資料

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 Server Express 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照資料も含まれています。

WebSphere Business Integration Server Express の資料は、Web サイト http://www.ibm.com/websphere/wbiserverexpress/infocenter からダウンロードし、インストールして表示できます。

表記上の規則

本書は、以下のような規則を使用しています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、リテラル値を示します。
太字	新規用語が最初に登場したときに使用します。
イタリック、*イタリック*	変数名または相互参照を表します。
青のアウトライン	青のアウトラインは、マニュアルをオンラインで表示するときにのみ見られるもので、相互参照用のハイパーリンクを示します。 アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
{ }	構文行で、中括弧はオプションのセットを囲みます。オプションは必須で、1 つのみ選択することができます。
[ ]	構文行で、大括弧はオプショナル・パラメーターを囲みます。
...	構文行で、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。
< >	命名規則では、1 つの名前の各要素を不等号括弧で囲んで、それぞれ区別できるようにします。例えば、`<server_name><connector_name>tmp.log` のように使用します。
/、¥	本書では、Windows のディレクトリー・パスの表記規則として円記号 (¥) を使用します。Linux および i5/OS では、ディレクトリー・パスにスラッシュ (/) を使用します。すべての WebSphere Business Integration システム製品のパス名は、使用システムで製品がインストールされたディレクトリーを基準とした相対パス名です。
`%text%` および `$text`	パーセント (%) 記号で囲まれたテキストは、Windows の `text` システム変数またはユーザー変数の値を示します。Linux 環境においてこれに相当する表記は `$text` で、*text*Linux 環境変数の値を示します。
ProductDir	製品がインストールされているディレクトリーを表しています。各プラットフォームのデフォルトは、以下のとおりです。 • Windows: IBM¥WebSphereServer • i5/OS: /WQIBM/ProdData/WBIServer44/product • Linux: /home/${username}/IBM/WebSphereServer

本リリースの新機能

バージョン 5.6.x

本書の JText コネクター・バージョン 5.6.x に対応したリリースの新機能は、以下のとおりです。

- Windows で実行されるアダプターにより、JText メタデータをサポートする双方向スクリプトに対応
- GB18030 中国語コードのサポートをこのリリースに追加
- セキュア FTP のサポートを追加
- 新しいコネクター固有 boolean プロパティー「NoPoll」を導入。オプションでポーリングをオフにできます。デフォルト値は false です。true に設定されている場合、アダプターは要求のみを処理し、ポーリングは処理しません。

5.6.x バージョンのアダプターは、以下のプラットフォームでサポートされています。

- WIN 2003
- Linux:
  RedHat 2.1 Enterprise Linux WS/AS/ES 3.0 Update 2、Intel (IA32)
  SuSE Linux ES 8.1 および Intel (IA32)
  SuSE Linux ES 9.0、Intel (IA32)
- IBM i5/OS V5R3、OS/400 V5R2

  注: 明示されない限り、i5/OS は i5/OS および OS/400 を意味します。

第 1 章 JText アダプターの概要

この章では、WebSphere Business Integration Server Express Adapter for JText について説明します。アダプターは、統合ブローカーがテキスト・ファイルまたはバイナリー・ファイルの交換によりアプリケーションと通信できるようにしますこのコネクターは API を持たないアプリケーションとのデータの統合を容易にします。

アダプターは、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 2 つのパーツで構成されています。コネクター・フレームワークは統合ブローカーとアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはどの コネクターにも共通です。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用に調整されたコードが組み込まれています。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントの間で、以下のサービスを提供します。

- ビジネス・オブジェクトの送受信
- 始動メッセージおよび管理メッセージの交換の管理

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に関する情報が含まれています。本書では、これらのコンポーネントの両方をアダプターとして参照します。

この章には以下のトピックが含まれます。

- 2 ページの『アダプター・コンポーネント』
- 3 ページの『JText コネクターが使用するビジネス・オブジェクト』
- 7 ページの『コネクターの動作』
- 15 ページの『コネクターの機能』
- 16 ページの『ロケール依存データの処理』

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*」を参照してください。

JText アダプターは、次の場合に使用します

- アプリケーションが、統合ブローカーの通信可能な場所を介して C、C++、または Java 標準 API を持たない場合。
- 特注のアプリケーションのイベント表を持つことができない場合。
- データを交換するメソッドとしてストリング・ファイルまたはバイナリー・ファイルが最も適切な場合。

上記の事例でアプリケーションを大容量のシステムに統合する場合には、JText コネクターを介してストリング・ファイルまたはバイナリー・ファイルを交換するのが最も簡単な方法です。

アダプター・コンポーネント

JText アダプターには以下のコンポーネントが組み込まれています。

- 『アプリケーション固有のコンポーネント』
- 『データ・ハンドラー』
- 3 ページの『メタオブジェクト』

図 1 は、WebSphere InterChange Server Express を統合ブローカーとして使用する場合の JText コネクターのアーキテクチャーを示しています。

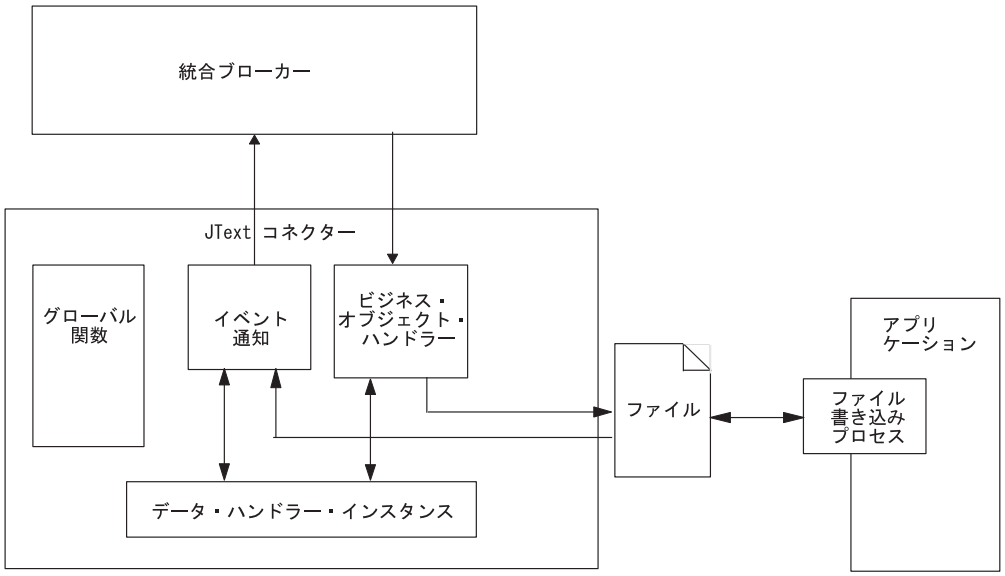

図 1. JText コネクター・アーキテクチャー

アプリケーション固有のコンポーネント

JText アダプターのアプリケーション固有のコンポーネントは、各種ファイルを操作します。また、ビジネス・オブジェクトとストリング (またはバイト配列) の間でデータ変換を行うため、指定されたデータ・ハンドラーを呼び出します。このコンポーネントは、統合ブローカーとの通信も処理します。

データ・ハンドラー

JText コネクターは、あらゆる既存のファイル・フォーマットとビジネス・オブジェクトの間の変換を提供するためのものです。コネクターは、これを提供するために、コネクターのメタオブジェクト構成で指定されているデータ・ハンドラーを使用します。

データ・ハンドラーは、ファイル・システムとの対話 (ファイルからの読み取り、またはファイルへの書き込み) を一切行わずに変換を実行します。テキスト・ファイルとのすべての対話は、他のコネクター・コンポーネントによって処理されます。

データ変換を行うには、WebSphere Business Integration Server Express Adapter Framework が提供するデータ・ハンドラーか、または特定のテキストをフォーマットするためにユーザーが作成するデータ・ハンドラーを使用します。この製品では、以下のデータ・ハンドラーが提供されます。

- NameValue — テキスト・データを名前付きフィールドに基づいて解析します。この場合は、テキスト・ファイルにビジネス・オブジェクト・タイプ (`BusinessObject=BOname`)、動詞 (`Verb=verbName`)、および属性の数 (`AttributeCount=numericValue`) を識別するフィールドが含まれます。
- Delimited — 機械読み取りの効率が最も重要視される場所で主に使用されます。ビジネス・オブジェクト・データの個別のフィールドを区切るために指定された区切り文字に基づいて、テキスト・データを解析します。
- FixedWidth — テキスト・データを固定長フィールドを使用して解析します。フィールド長は、各ビジネス・オブジェクト属性の `MaxLength` プロパティーで指定されます。このプロパティーの値は、ビジネス・オブジェクト定義に保管されます。

詳細については、 14 ページの『データ・ハンドラー処理の仕組み』を参照してください。本製品で提供されている各データ・ハンドラーに関する詳細は、「データ・ハンドラー・ガイド」を参照してください。

メタオブジェクト

JText コネクターには、Connector Configurator Express で設定した標準のアプリケーション固有のコネクターの構成プロパティーの他に、異なるビジネス・オブジェクトに異なる処理を行うようにコネクターを構成できるようにする構成プロパティーのセットが含まれています。JText メタオブジェクトを使用して各プロパティーを設定します。メタオブジェクトは、構成情報を含む特別な種類のビジネス・オブジェクトです。

コネクターは、ファイルから読み取ったストリングまたはバイト配列をビジネス・オブジェクトに変換したり、ビジネス・オブジェクトから読み取ったストリングまたはバイト配列をフォーマット設定してファイルにしたりする際に使用するクラスを判別するために、メタオブジェクトの情報を使用します。JText メタオブジェクトは、イベント処理、および要求処理中に使用するディレクトリー、ファイル拡張子、ファイル名、ビジネス・オブジェクトの区切り文字、およびデータ・ハンドラーを指定します。

JText アダプターはメタオブジェクトを内部で使用します。それらを統合ブローカーを介して送信することはありません。メタオブジェクトを使用したコネクターの構成に関する詳細は、51 ページの『第 4 章 JText コネクター・メタオブジェクトの使用』を参照してください。

JText コネクターが使用するビジネス・オブジェクト

JText コネクターのビジネス・オブジェクトは、変換用に指定したデータ・ハンドラーが必要とするフォーマットでデータを引き渡す必要があります。ただし、JText コネクターは、アプリケーション・コネクターのアプリケーション固有ビジネス・オブジェクトと互換性のある特別に設計されたビジネス・オブジェクトのセットを必要としないこともあります。

例えば、NameValue データ・ハンドラーは、各データが CustomerName=Kumar、Region=NE、および Department=HR などのデータを識別するためのストリングを持つことを要求します。JText コネクターが汎用ビジネス・オブジェクトを使用できるのは、すべての汎用ビジネス・オブジェクト定義に属性が含まれていて、その名前でそれぞれのデータを識別できるためです。

ただ、汎用ビジネス・オブジェクトは多数の異なるアプリケーションで必要な情報のスーパーセットであり、各汎用ビジネス・オブジェクトには、通常 1 つのアプリケーションで必要な情報よりはるかに多くの情報が含まれています。

このため、データをそれぞれのアプリケーションが処理しやすいサイズに変換するには、処理するデータのタイプによってビジネス・オブジェクトを作成することをお勧めします。この種のビジネス・プロジェクトは、任意のアプリケーションで必要なデータとデータ・ハンドラーで必要な情報のみを提供します。

例えば、FixedWidth データ・ハンドラーの場合は、すべてのビジネス・オブジェクト属性に対して MaxLength 属性プロパティー用に指定された値を設定していなければなりません。XML 用の WebSphere Business Integration Server Express データ・ハンドラーの場合他の指定情報が必要です。その一方で、NameValue および Delimited データ・ハンドラーの場合は、すでに汎用ビジネス・オブジェクトに含まれている情報をビジネス・オブジェクトに含める必要はありません。各データ・ハンドラー固有の情報についての詳細は、「データ・ハンドラー・ガイド」を参照してください。

ビジネス・オブジェクトには、データを引き渡す以外にも、コネクターがビジネス・オブジェクトのイベント・ファイル名を動的に取得したり、出力ファイル名を統合ブローカーに戻すための情報を含めることができます。この動的処理を行えるようにコネクターを構成するには、ビジネス・オブジェクト・レベルのアプリケーション固有の情報に、以下の名前と値のペアを含める必要があります。

- cw_mo_JTextConfig = *DynChildMOAttrName*

ビジネス・オブジェクトにデータ・ハンドラーが使用する追加のアプリケーション固有の情報が含まれる場合は、ビジネス・オブジェクトで名前と値のペアを最初に置き、セミコロン (;) で追加のアプリケーション固有の情報と区切る必要があります。コネクターは、動的処理を使用するかどうかを判別するためにセミコロンより前の名前と値のペアを読み取り、さらにセミコロンより後に情報があればそれをデータ・ハンドラーに受け渡します。

動的子メタオブジェクトの使用

動的子メタオブジェクトを使用すると、InterChange Server Express とのファイル名の交換が可能になります。このセクションの内容は、以下のとおりです。

- 『動的子メタオブジェクトの使用目的』
- 5 ページの『動的子メタオブジェクトの使用方法』
- 5 ページの『動的子メタオブジェクトの属性』

動的子メタオブジェクトの使用目的

動的子メタオブジェクトを作成して使用すると、コネクターは以下を行うことができます。

サービス呼び出し要求

- ビジネス・オブジェクトの種類ごとに (統合ブローカーが子の OutFileName 属性に挿入する値に基づく)、または個々のビジネス・オブジェクトごとに (統合ブローカーが順序付けを指定している場合) 出力ファイル名を動的に作成します。

  注: コネクターは子の FileWriteMode 属性を使用して、子の OutFileName 属性で指定されているファイルに上書きするか、または追加するかどうかを判別します。
- コネクターが生成した各出力ファイル名の名前を戻します (子の OutFileName 属性に値が含まれていない場合)。この場合、コネクターは以下を行います。
  - 親ビジネス・オブジェクトの名前から新しい名前を作成します。
  - そのファイルにオブジェクトを書き込みます。
  - 作成した名前を OutFileName メタオブジェクト属性に取り込みます。
  - 作成した名前を統合ブローカーに受け渡します。統合ブローカーは事前に指定していなくても動的に作成された出力ファイル名を取得します。

イベント処理

コネクターは、ビジネス・オブジェクトの読み取り先であるファイルの名前を子の InFileName 属性に取り込みます。

動的子メタオブジェクトの使用方法

コネクターがファイル名を動的に処理できるようにするには、以下を行う必要があります。

1. 特定の属性を持つ動的子メタオブジェクトを作成します。
2. データ・ビジネス・オブジェクトで、動的子メタオブジェクトを表す属性を追加します。
3. データ・ビジネス・オブジェクトは、以下のビジネス・オブジェクト・レベルのアプリケーション固有の情報を指定します。

   ```
   cw_mo_JTextConfig = DynChildMOAttrName
   ```

   ここで、*DynChildMOAttrName* は、動的子ビジネス・オブジェクトを表すデータ・ビジネス・オブジェクト内の属性の名前です。その例については、図 2 を参照してください。

   重要: データ・ハンドラーを使用するときは、cw_mo_ 接頭部が必要です。接頭部を付けない場合は、コネクターは、指定された出力ファイルをデータ・ビジネス・オブジェクトと見なしてこれに動的子メタオブジェクトを書き込みます。
4. 動的子メタオブジェクトは、動的子メタオブジェクト内の属性の値を指定します。

動的子メタオブジェクトの属性

動的子メタオブジェクトには、以下の属性を含める必要があります。

- FileWriteMode — コネクターが既存の出力ファイルに追加するか、または上書きするかを指定する値を持つストリング属性。この属性の値は、「a」が追加、「o」が上書きを示します。コネクターは最初の文字のみを検査し、大文字小文字を区別しません。
- InFileName — イベント・ファイル名 (ビジネス・オブジェクトの取得先であるファイルと絶対パス) が取り込まれるストリング属性。
- OutFileName — コネクターが出力ファイルへの書き込みで使用するファイル名、絶対パスおよびファイル名、または FTP URL を含むことができる値を持つストリング属性。
  - この属性にファイル名のみが含まれている場合は、コネクターは指定されたファイルを、コネクターが始動したディレクトリーに書き込みます。
  - この属性に絶対パスとファイル名が含まれている場合は、コネクターは指定されたファイルを指定されたディレクトリーに書き込みます。
  - この属性に FTP URL のみが含まれている場合は、コネクターはトップレベル JText メタオブジェクトの EventDir 属性からログイン、パスワード、およびポートの値を取得します。
  - この属性に、ログイン、パスワード、ポートの値が組み込まれた FTP URL が含まれている場合は、コネクターはこの属性で指定されている値を使用して、トップレベル JText メタオブジェクトの EventDir 属性で指定されている値をオーバーライドします。

    詳細については、 78 ページの『リモート FTP ファイル・システムの指定』を参照してください。

図 2 は、動的子メタオブジェクトを含むカスタマー・ビジネス・オブジェクトの例を示しています。

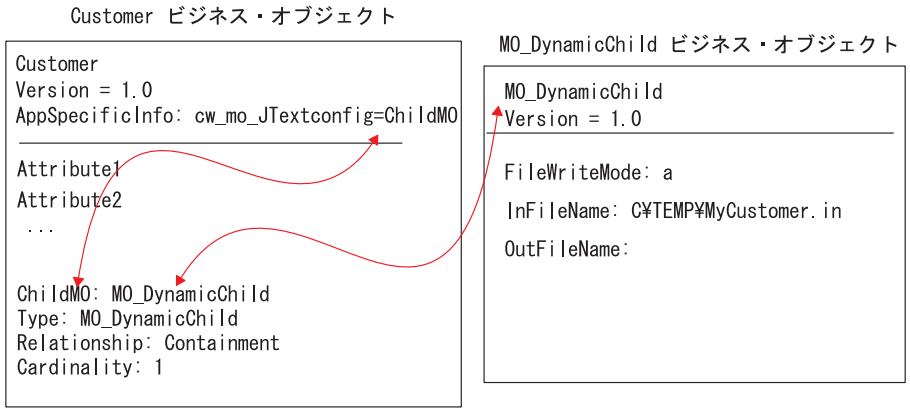

図 2. 動的子メタオブジェクトの例

コネクターの動作

JText コネクターは、テキスト・ファイルまたはバイナリー・ファイルの交換を通じてアプリケーションと通信します。このコネクターは、ビジネス・オブジェクトを処理する際に、以下の基本タスクを実行します。

- イベント通知
- 要求処理

このセクションでは、以下のタスクについて説明します。また、データ・ハンドラーの処理がどのように行われるか、および JText コネクターが動詞をどのように処理するかについても説明します。

イベント通知

JText コネクターは、他のコネクターとは異なる方法でイベントを処理します。JText コネクターは、サード・パーティーのアプリケーションに依存するコネクターと異なり、イベント表を持ちません。その代わりに、コネクターはイベント・ディレクトリーをイベント表として使用します。

以下に、JText コネクターがイベントを処理するときに行う操作を示します。

1. コネクターは、指定されたディレクトリー内の指定された拡張子が付いたファイルを検査することによりイベントをポーリングします。指定されたディレクトリー内に存在する指定された拡張子が付いたファイルは、イベントと同等と見なされます。コネクターは、イベント・ディレクトリーからイベント・ファイルを解釈せずに直接読み取ります。その後、いずれかの解析手法を使用して、どのサブセクションがどのビジネス・オブジェクトを表しているかを判別します。詳細については、74 ページの『特定のビジネス・オブジェクトのポーリング』を参照してください。
2. コネクターはデータ・ハンドラーのインスタンスを作成します (これはデータ・ビジネス・オブジェクトの JText メタオブジェクトで指定された値に基づきます)。
3. コネクターは、データ・ハンドラー・インスタンスの `getBO()` または `getBOByteArray()` を呼び出し、この呼び出したメソッドにビジネス・オブジェクトを表すストリングまたはバイト配列を送信します。コネクターはビジネス・オブジェクトを表すそれぞれの要素をデータ・ハンドラーに受け渡します。1 つのファイルが複数のビジネス・オブジェクトを表している場合、コネクターはファイル全体ではなく、1 つの要素 (単一のビジネス・オブジェクトを表すストリングまたはバイト配列) のみを送信します。
4. データ・ハンドラーは、ストリングまたはバイト配列をビジネス・オブジェクトに変換し、そのビジネス・オブジェクトをコネクターに戻します。データ・ハンドラーはエラーの報告やトレースの提供も行います。
5. データ・ハンドラーはデフォルトの動詞処理を実行します。データ・ハンドラーの開発者は、動詞を設定するためにロジックを指定する必要がありますが、これは、コネクターがこのロジックを提供しないためです。データ・ハンドラーが動詞を設定する方法についての例は、「*データ・ハンドラー・ガイド*」を参照してください。
6. ビジネス・オブジェクトの作成を妨げるエラーがデータ・ハンドラーで発生すると、コネクターは、ストリングまたはバイト配列を、拡張子 `.fail` を付けてア

ーカイブします。データ・ハンドラーの処理が成功すると、コネクターはビジネス・オブジェクトのサブスクリプションをチェックします。

- コネクターがビジネス・オブジェクトをサブスクライブしない場合は、コネクターは .unsub 拡張子が付いたストリングをアーカイブ・ファイルに書き込みます。
- コネクターがビジネス・オブジェクトをサブスクライブする場合は、コネクターはビジネス・オブジェクトを統合ブローカーに送信します。

7. コネクターがビジネス・オブジェクトを統合ブローカーに正常に送信する場合は、イベント・ファイル内の任意のビジネス・オブジェクトの処理が失敗したかどうかによって、.success または .partial 拡張子が付いたファイルをアーカイブします。コネクターがビジネス・オブジェクトの送信に失敗した場合は、コネクターは .fail 拡張子が付いたファイルをアーカイブします。

JText コネクターは、ファイル名に基づき、アルファベット順にイベント・ファイルを処理します。JText コネクターは、構成によって、イベント・ディレクトリー内のすべてのファイルを選択したり、指定された拡張子が付いたファイルのみを選択できます。詳細については、74 ページの『複数のイベント・ファイルまたは複数のイベント・ディレクトリーの指定』を参照してください。SortFilesOnTimestamp プロパティーにより、JText コネクターは、イベント・ファイルの最も古いものから最も新しいものまでを、ロケーションに関係なく、タイム・スタンプの順に処理することができます。つまり JText コネクターは、別々のディレクトリーに格納されているファイルを、それぞれのタイム・スタンプによって日時順に処理します。詳細については、37 ページの『SortFilesOnTimestamp』を参照してください。

PollQuantity プロパティーは、コネクターが任意のポーリング中に統合ブローカーに通知できるビジネス・オブジェクトの最大数を指定します。例えば、PollQuantity の値を 5 に設定し、コネクターがポーリングを行っているディレクトリー内に 2 つのファイルが存在すると想定します。最初のファイルにはビジネス・オブジェクトが 4 つあり、2 つ目のファイルには 12 のビジネス・オブジェクトがあるとします。コネクターは 1 度目のポーリング呼び出しのときに以下の手順を実行します。

1. 最初のファイルから 4 つのビジネス・オブジェクトのすべてを送信し、処理が完了するごとに各ビジネス・オブジェクトをアーカイブします。
2. 2 つ目のファイルの最初のビジネス・オブジェクトを送信します。

コネクターは 2 度目のポーリング呼び出しのときに、2 つ目のファイルの 2 から 6 番目のビジネス・オブジェクトを送信します。コネクターは 3 度目のポーリング呼び出しのときに、2 つ目のファイルの 7 から 11 番目のビジネス・オブジェクトを送信します。コネクターは 4 度目のポーリング呼び出しのときに、最後のビジネス・オブジェクトを送信します。コネクターは処理が完了するごとに各ビジネス・オブジェクトをアーカイブします。ファイル内のビジネス・オブジェクトのいずれかの処理が失敗した場合は、コネクターは .orig 拡張子を付けてファイル全体をアーカイブします。

詳細については、以下を参照してください。

- PollQuantity プロパティーを使用する際にパフォーマンスを調整する場合は、89 ページの『JText コネクターのパフォーマンス調整』を参照してください。

- イベント・ディレクトリーおよび拡張子の指定については、 66 ページの『イベント・ディレクトリーおよび拡張子の指定』を参照してください。
- イベント処理の指定については、 66 ページの『イベント通知の指定』を参照してください。

図 3 はイベント通知操作を表します (図形の数字は、上記で説明した手順とは対応していません)。

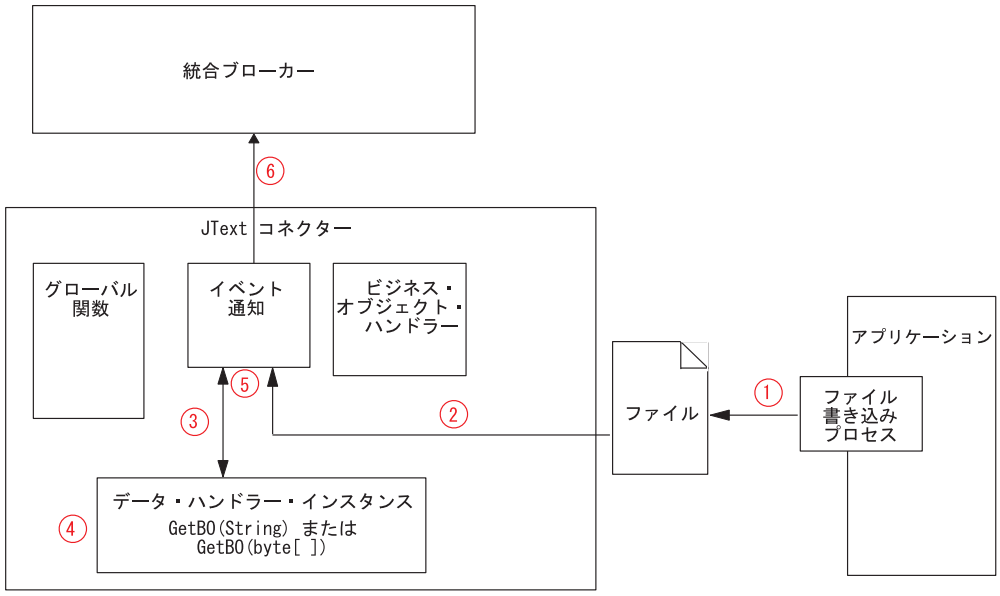

図 3. イベント通知操作

イベント・アーカイブ

JText コネクターは、アーカイブを行うように構成されている場合には、イベント処理の完了後、1 つのビジネス・オブジェクトを表しているストリングまたはバイト配列を、ローカル・アーカイブ・ディレクトリー内のファイルに書き込みます。コネクターは下線 (_)、タイム・スタンプ、およびイベント状況に応じたファイル拡張子を持つ名前をそのファイルに付けます。引き渡されるデフォルトの拡張子は、`success`、`partial`、`unsub`、`orig`、および `fail` です。下線とタイム・スタンプは、ファイル名とファイル拡張子の間に追加されます。

タイム・スタンプでは、システム時刻の年、月、日、時、分、秒、ミリ秒がそれぞれ下線で区切られています。アーカイブされたファイル名は常に固有なので、コネクターが既存のファイルを同じ名前のファイルで上書きすることはありません。アーカイブされたファイル名の形式を以下に示します。

BOName`_YYYY_MM_DD_HH_MM_SS_sss.[`*extension*`]`

例えば、コネクターは、`Customer.in` という名前のファイルを正常に処理したときに、`Customer_2003_11_15_18_24_59_999.success` というファイル名に変更します。

JText コネクターは、フォーマット・エラーが発生したり、統合ブローカーへのビジネス・オブジェクトの送信に失敗した場合は、そのビジネス・オブジェクトを

.fail ファイルにアーカイブします。また、コネクターがサブスクライブしない場合は、JText コネクターは、拡張子が .unsub のファイルにビジネス・オブジェクトをアーカイブします。ユーザーが上記のアーカイブ・ファイルをチェックして、フォーマット・エラーがあればそれを訂正したり、ビジネス・オブジェクトをサブスクライブする処理を開始したら、次にこれらのアーカイブ・ファイルの中のビジネス・オブジェクトを再サブミットしてください。

アーカイブに関する詳細は、 67 ページの『イベント・アーカイブの指定』を参照してください。

イベントおよびアーカイブ・ファイル用のデフォルトのファイル拡張子

JText コネクターは、イベント表やアーカイブ表を使用しないためイベント状況を更新する場合はファイル拡張子を変更します。表 1 は、JText アダプターが、イベント・ファイルとアーカイブ・ファイル用に引き渡すデフォルトのファイル拡張子の値を示しています。

表 1. デフォルトのファイル拡張子

ファイル・タイプ	イベント状況/説明	デフォルトのファイル拡張子	引き渡されるデフォルト・ディレクトリー
イベント	new	in	**Linux:** /tmp/JTextConn/Default/Event **Windows:** C:¥temp¥JTextConn¥Default¥Event **i5/OS:** デフォルトなし
アーカイブ	success (イベント・ファイル内のすべてのビジネス・オブジェクトが正常に処理された場合は、このファイルにはすべてのビジネス・オブジェクトが含まれます)	success	**Linux:** /tmp/JTextConn/Default/Archive **Windows:** C:¥temp¥JTextConn¥Default¥Archive i5/OS: デフォルトなし
アーカイブ	success (イベント・ファイル内の一部のビジネス・オブジェクトの処理に失敗した場合は、このファイルには正常に処理されたビジネス・オブジェクトのみが含まれます)	partial	**Linux:** /tmp/JTextConn/Default/Archive **Windows:** C:¥temp¥JTextConn¥Default¥Archive **i5/OS:** デフォルトなし

表 1. デフォルトのファイル拡張子 (続き)

ファイル・タイプ	イベント状況/説明	デフォルトのファイル拡張子	引き渡されるデフォルト・ディレクトリー
アーカイブ	unsubscribed	unsub	**Linux:** /tmp/JTextConn/Default/Archive **Windows:** C:¥temp¥JTextConn¥Default¥Archive **i5/OS:** デフォルトなし
アーカイブ	entire original event file (このファイルは処理に失敗したビジネス・オブジェクトがある、またはアンサブスクライブされたビジネス・オブジェクトがある場合にのみ作成される。ただし、このイベント・ファイルにビジネス・オブジェクトが 1 つしか含まれていない場合も同様。)	orig	**Linux:** /tmp/JTextConn/Default/Archive **Windows:** C:¥temp¥JTextConn¥Default¥Archive **i5/OS:** デフォルトなし
アーカイブ	fail	fail	**Linux:** /tmp/JTextConn/Default/Archive **Windows:** C:¥temp¥JTextConn¥Default¥Archive **i5/OS:** デフォルトなし
出力	out	out	**Linux:** /tmp/JTextConn/Default/Out **Windows:** C:¥temp¥JTextConn¥Default¥Out **i5/OS:** デフォルトなし

重要: 同じファイルに同時にアクセスしてこのファイルを処理しているアプリケーションが複数存在する場合に、アプリケーションのアクセス・シーケンスは重要です。ファイル・ロックや不完全なデータによる問題を避けるために、任意のファイル上で実行されるすべての操作を解析してください。

注: コネクターは、イベント・ディレクトリー内のすべてのファイルを入力ファイルとして扱う場合にも拡張子を付けます。コネクターがアーカイブされたファイルをイベントとして扱わないようにするために、入力ファイルの拡張子をアーカイブ・ファイルの拡張子と同じにしないか、あるいは入力ファイルとアーカイブ・ファイルをそれぞれ別のディレクトリーに保管するようにしてください。

新しいファイル拡張子、イベント、ディレクトリー、および出力ディレクトリーの指定に関する詳細については、 55 ページの表 8 を参照してください。

要求処理

コネクターは、サービス呼び出し要求を処理するとき、ビジネス・オブジェクトを出力ストリングまたは出力バイト配列に変換し、ファイルに書き込みます。

ただし、コネクターはビジネス・オブジェクトを変換する前に、そのビジネス・オブジェクトが動的なファイルの命名のために構成されているかどうかについて判別します。これは、ビジネス・オブジェクトに動的子メタオブジェクトが含まれているかどうかで判別します。このケースでは、コネクターは動的に出力ファイルの名前を付けたり、コネクターが生成する出力ファイルの名前を戻したりします。

このセクションでは、以下の場合のサービス呼び出し要求処理について説明します。

- 『データ・ビジネス・オブジェクトが動的ファイル命名を指定しない場合』
- 13 ページの『データ・ビジネス・オブジェクトに動的子メタオブジェクトが含まれる場合』

データ・ビジネス・オブジェクトが動的ファイル命名を指定しない場合

データ・ビジネス・オブジェクトが動的ファイルの命名を指定しない場合は、コネクターはサービス呼び出し要求を処理するために以下の操作を実行します。

1. コネクターはビジネス・オブジェクト要求を受け取ります。
2. コネクターは、ビジネス・オブジェクト・レベルの AppSpecificInfo プロパティーに以下の行が含まれているかどうかを判別します。

   `cw_mo_JTextConfig = DynChildMOAttrName`
3. コネクターはトップレベルの JText メタオブジェクトの構成をチェックして、呼び出すデータ・ハンドラーを判別します。デフォルトではこのメタオブジェクトが NameValue データ・ハンドラーを表す `MO_DataHandler_DefaultNameValueConfig` データ・ハンドラーのメタオブジェクトを指定します。
4. コネクターは適切なデータ・ハンドラーのインスタンスを作成してそこにビジネス・オブジェクトを送信します。
5. データ・ハンドラーは、ビジネス・オブジェクトをストリングまたはバイト配列に変換します (これは構成に戻されます)。データ・ハンドラーはエラーの報告やトレースの提供も行います。
6. コネクターは、戻されたストリングまたはバイト配列をファイルに書き込みます。

要求を処理するようにコネクターを構成するための情報は、 69 ページの『要求処理の指定』を参照してください。

データ・ビジネス・オブジェクトに動的子メタオブジェクトが含まれる場合

データ・ビジネス・オブジェクトに動的子メタオブジェクトが含まれる場合は、コネクターはサービス呼び出し要求を処理するために以下の操作を実行します。

1. コネクターはビジネス・オブジェクト要求を受け取ります。
2. コネクターはビジネス・オブジェクト・レベルの AppSpecificInfo プロパティーに以下のテキストが含まれているかどうかを判別します。

   ```
   cw_mo_JTextConfig = DynChildMOAttrName
   ```

   注: ビジネス・オブジェクトのアプリケーション固有の情報が動的子メタオブジェクトを指定しておらず、このような子を含まない場合は、コネクターは 12 ページの『データ・ビジネス・オブジェクトが動的ファイル命名を指定しない場合』に従ってビジネス・オブジェクトを処理します。
3. コネクターは動的子メタオブジェクトの OutFileName 属性から出力ファイルの名前を取得します。
   - この属性に値が含まれている場合は、コネクターはその名前でファイルが存在するかどうかをチェックします。ファイルが存在しない場合は、コネクターは新しい出力ファイルを作成し、その属性の値を使用してファイルに名前を付けます。ファイルがすでに存在する場合は、コネクターは、子メタオブジェクトの FileWriteMode の値に基づいて既存のファイルに追加するか、または上書きします。

     重要: FileWriteMode 属性の値が「o」以外の値で始まる場合は、コネクターはデフォルトで追加モードになります。
   - この属性に値 (つまり OutFileName=CxIgnore) が含まれていない場合は、コネクターは、この子メタオブジェクトを含む親ビジネス・オブジェクトの名前をもとにファイル名を作成し、トップレベルの JText メタオブジェクトの構成を使用して出力ファイルの場所を判別します。コネクターはビジネス・オブジェクトをそのファイルに書き込んだ後、ファイルの名前とパスをこの属性内に戻します。
4. コネクターはトップレベルの JText メタオブジェクトの構成をチェックして、呼び出すデータ・ハンドラーを判別します。デフォルトではこのメタオブジェクトが NameValue データ・ハンドラーを表す MO_DataHandler_DefaultNameValueConfig データ・ハンドラーのメタオブジェクトを指定します。
5. コネクターは適切なデータ・ハンドラーのインスタンスを作成してそこにビジネス・オブジェクトを送信します。
6. データ・ハンドラーは、ビジネス・オブジェクトをストリングまたはバイト配列に変換します (これは構成に戻されます)。データ・ハンドラーはエラーの報告やトレースの提供も行います。
7. コネクターは、戻されたストリングまたはバイト配列を、上のステップ 3 で取得した名前を持つファイルに書き込みます。

図 4 に、JText コネクターが、統合ブローカー InterChange Server Express から宛先アプリケーションへの要求を処理するときの、JText コネクターのコンポーネントを示します。

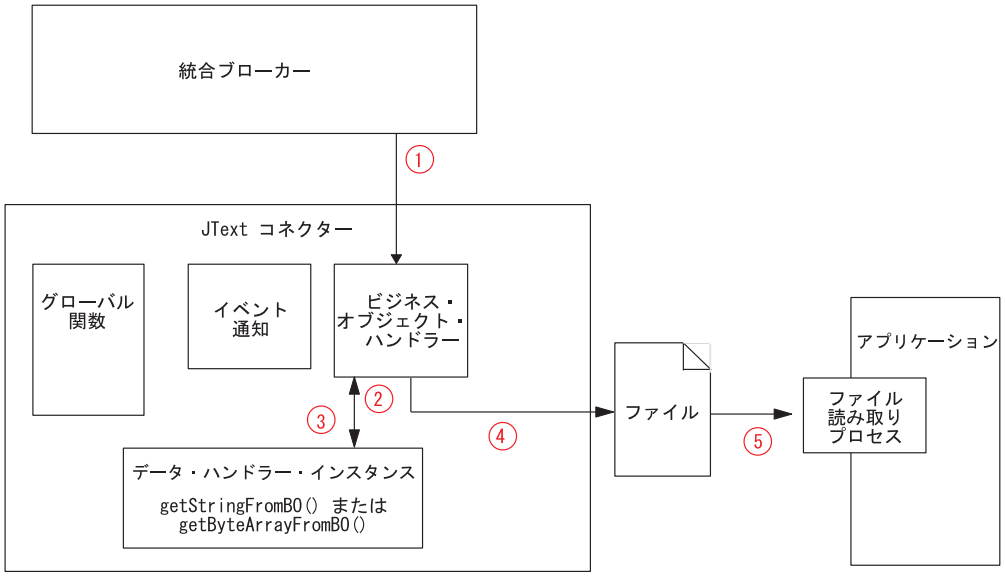

図 4. ビジネス・オブジェクト要求操作

データ・ハンドラー処理の仕組み

コネクターは、データ・ハンドラー・インスタンスを使用して、ビジネス・オブジェクトと、イベント・ファイルから読み取ったストリングまたはバイト配列との間で、変換を行います。データ・ハンドラーはエラーの報告やトレースの提供も行います。

コネクターはトップレベルの JText メタオブジェクト内の `EventDataHandler` 属性と `OutputDataHandler` 属性の値に基づいてデータ・ハンドラーのインスタンスを作成します。これらの属性は、コネクターがデータ・ハンドラーのインスタンスを作成するために使用するデータ・ハンドラーのメタオブジェクトを識別します。データ・ハンドラーのメタオブジェクトは引き渡されたデータ・ハンドラー、またはカスタム・データ・ハンドラーを表せます。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。

コネクターは、`DataProcessingMode` メタオブジェクトの設定を基に、インターフェース (ストリングまたはバイト配列) を判別します。このメタオブジェクトの詳細については、55 ページの表 8 を参照してください。

コネクターは構成情報を受け取ると、以下の手順を実行します。

1. データ・ハンドラーをインスタンス化します。
2. データ・ハンドラーの `TracingSubSystem` 属性をコネクターの名前に対して設定するために、データ・ハンドラーの `setOption()` メソッドを呼び出します。データ・ハンドラーは、書き込むトレース・メッセージにコネクターの名前を含めるためにこの値を使用します。

データ・ハンドラーの作成と構成が完了すると、コネクターはビジネス・オブジェクトに、またはビジネス・オブジェクトからデータの変換を行うためにデータ・ハンドラー内の適切なメソッドを呼び出します。

- イベント通知の場合は、コネクターはデータ・ハンドラー上の getBO(String) または getBO(byte[]) メソッドを呼び出します。コネクターは、ビジネス・オブジェクトに変換されるストリングをファイルからデータ・ハンドラーに受け渡します。データ・ハンドラーはビジネス・オブジェクトを戻します。
- コネクターは要求処理のためにデータ・ハンドラーの getStringFromBO() または getByteArrayFromBO() メソッドを呼び出します。コネクターは、ストリングまたはバイト配列に変換する必要があるビジネス・オブジェクトを、データ・ハンドラーに渡します。データ・ハンドラーは、ビジネス・オブジェクトを直列化したものを、ストリングまたはバイト配列の形で戻します。

getBO(String) (または getBO(byte[])) メソッドと getStringFromBO() (または getByte ArrayFromBO()) メソッドは、それぞれ、トップレベルの親のビジネス・オブジェクトとすべての子ビジネス・オブジェクトからなるビジネス・オブジェクト階層全体を送信するか、これらを受信するかのどちらかです。

いずれの場合も、データ・ハンドラーはすべてのメタオブジェクト・データをフィルターに掛けて、ビジネス・オブジェクト固有のデータのみを受け渡します。本製品で提供されている各データ・ハンドラーは、この機能を備えています。カスタム・データ・ハンドラーを使用する場合は、この機能を備えていなければなりません。

要求に対するビジネス・オブジェクト動詞の処理

JText コネクターは要求を処理する際に、1 つの動詞を他の動詞と分けて処理しません。コネクターは動詞がビジネス・オブジェクトと関連付けられているかどうかにかかわらず、更新、検索、削除を行わずにファイルに書き込みます。

JText コネクターは要求を処理する際に、以下の条件が true の場合に CxIgnore の値を持つすべての属性をそれらのデフォルト値に設定します。

- 動詞が Create である。
- コネクターの UseDefaults プロパティーが true に設定されている。
- 属性が Required に設定されている。
- ビジネス・オブジェクトの指定で属性にデフォルト値が設定されている。

コネクターの機能

JText コネクターは、イベント通知とビジネス・オブジェクトの要求処理の他に以下の機能を提供します。

- 異なるビジネス・オブジェクトごとに異なる構成。例えば、異なるディレクトリーおよびファイル拡張子、あるいは異なるデータ・フォーマットを使用するために異なるビジネス・オブジェクトを構成できます。
- ファイル拡張子、アーカイブ・ファイル・ストレージ用のディレクトリー・ロケーション、フォーマット・タイプ、およびファイルの順序付けのための構成機能。

- ビジネス・オブジェクトごとの出力ファイル名を動的に判別する、または生成済み出力ファイルの絶対パス名を戻すための構成機能。
- 障害リカバリー。
- カスタム・データ・ハンドラー機能。ユーザーはコネクター・コードを再コンパイルせずに新たなデータ・ハンドラーを作成できます。構成プロパティーを変更するだけで、作成済みの新規クラスを使用できるようになります。
- リモート FTP ロケーションおよびローカル・ファイル・システムのディレクトリーとのデータ交換機能。

詳細については、17 ページの『第 2 章 JText アダプターのインストール』、 51 ページの『第 4 章 JText コネクター・メタオブジェクトの使用』、および「データ・ハンドラー・ガイド」を参照してください。

JText アダプターの他のアダプターとの違い

JText コネクターは、他のアダプターと同様にソース・アプリケーションから宛先アプリケーションへのデータの転送を可能にする他、以下の面で他と異なります。

- すべてのビジネス・オブジェクトを同じ方法で処理します。つまり、JText コネクターはビジネス・オブジェクトを常時ファイルに書き込むため、どの動詞が入ってきても `Create` 操作のみを実行します。
- 処理するビジネス・オブジェクトの内容を解釈しません。つまり、どのビジネス・オブジェクトも、ストリングまたはバイト配列に変換されるものとして読み取られます。このとき、キー値が他のデータ以上の重要性を持つことはありません。
- 構成の大部分にメタオブジェクト値を使用します。詳細については、51 ページの『第 4 章 JText コネクター・メタオブジェクトの使用』を参照してください。
- イベント表を持ちません。その代わりに、コネクターは構成済みのイベント・ディレクトリーをイベント表として使用します。

ロケール依存データの処理

コネクターは、2 バイト文字セットをサポートし、指定された言語でメッセージ・テキストを送達できるように国際化されています。コネクターは、1 つの文字コードを使用する場所から別のコードを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系を含む) の文字に対応できるエンコード方式が組み込まれています。WebSphere Business Integration Server Express システムのほとんどのコンポーネントは Java で記述されています。そのため、WebSphere Business Integration Server Express システム・コンポーネント間でデータを転送するときは、ほとんどの場合文字変換は必要ありません。エラー・メッセージと通知メッセージを適切な言語で、適切な地域用に記録するには、該当する環境の Locale 標準構成プロパティーを設定します。これらのプロパティーに関する詳細は、107 ページの『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。

第 2 章 JText アダプターのインストール

本章では、JText コネクターのインストールと構成の方法について説明します。

- 『インストール・タスクの概要』
- 『アダプター環境』
- 19 ページの『JText アダプターのインストール』
- 19 ページの『インストールの検証』

インストール・タスクの概要

Adapter for JText をインストールするには、以下の作業を実行する必要があります。

- 統合ブローカーのインストール: この作業では、WebSphere Business Integration システムと i5/OS 統合ブローカーのインストールを行います。作業の詳細については、使用するブローカーおよびオペレーティング・システムのインストール資料に説明があります。
- アダプターおよび関連ファイルのインストール: この作業では、アダプター用のファイルをソフトウェア・パッケージから使用システムにインストールします。19 ページの『JText アダプターのインストール』を参照してください。

アダプターをインストールする前に、アダプターの環境について理解しておく必要があります。詳細については、『アダプター環境』を参照してください。

この章で説明する作業

この章で説明する作業は、以下のとおりです。

表 2. アダプターのインストール: 作業ロードマップ

作業	関連手順 (参照先)	詳細情報 (参照先)
アダプターのインストール	19 ページの『JText アダプターのインストール』	Windows 版、Linux 版、または i5/OS 版の「WebSphere Business Integration Server Express インストール・ガイド」
インストールの検証	19 ページの『インストールの検証』	

アダプター環境

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

- 『インストール・タスクの概要』
- 18 ページの『アダプター・プラットフォーム』
- 18 ページの『アダプターの前提条件』

- 『ロケール依存データ』

アダプター・プラットフォーム

アダプターは以下のプラットフォームでサポートされています。

- すべてのオペレーティング・システム環境で、アダプターのコンパイルのために Java コンパイラー (Windows 2003 対応 IBM JDK 1.4.2) が必要になります。
- IBM i5/OS V5R2 および OS/400 V5R3
- Windows 2003 (Standard Edition および Enterprise Edition)
- Windows XP Service Pack 1A 適用済み、WebSphere Business Integration Adapter Framework 対応 (管理ツールのみ)
- Linux:
  RedHat 2.1 Enterprise Linux WS/AS/ES 3.0 Update 2、Intel (IA32)
  SuSE Linux ES 8.1、Intel (IA32)
  SuSE Linux ES 9.0、Intel (IA32)

アダプターの前提条件

JText コネクターを稼働する前に、コネクターが読み取り/書き込みを行うテキスト・ファイルを格納するイベント・ディレクトリー、出力ディレクトリー、アーカイブ・ディレクトリーについての読み取り/書き込み許可を作成しておいてください。この作業は、ローカル・サーバーとリモート・サーバーの両方で行う必要があります。

ロケール依存データ

コネクターは、2 バイト文字セットをサポートし、指定された言語でメッセージ・テキストを送達できるように国際化されています。コネクターは、1 つの文字コードを使用する場所から別のコード・セットを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、最も広く知られている文字コード・セット (単一バイトおよびマルチバイトの両方) の文字のエンコードが含まれています。WebSphere Business Integration システムのほとんどのコンポーネントは、Java で記述されています。したがって、ほとんどの統合コンポーネントの間でデータが転送されても、文字変換の必要はありません。エラー・メッセージと通知メッセージを適切な言語で、適切な地域用に記録するには、該当する環境の Locale 標準構成プロパティーを設定します。構成プロパティーの詳細については、107 ページの『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。

このアダプターは、アラビア語、ヘブライ語、ウルドゥー語、ペルシア語、およびイディッシュ語などの言語における双方向スクリプト・データの処理をサポートします。双方向能力を使用するには、双方向標準プロパティーを構成する必要があります。詳細については、107 ページの『コネクターの標準構成プロパティー』にあるコネクターの標準構成プロパティーを参照してください。

JText アダプターのインストール

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

インストールの検証

以下のセクションでは、インストール後の製品のパスとファイル名、およびアダプター・インストールの検証方法について説明します。

Windows システムでのインストールの検証

始める前に、アダプターをインストールします。インストーラーにより、アダプターに関連付けられている標準ファイルがシステムにコピーされます。ユーティリティーは、コネクターを *ProductDir*¥connectors¥JText ディレクトリーにインストールし、そのコネクターへのショートカットを「スタート」メニューに追加します。

Windows システムでアダプターのインストールを検証するには、アダプターをインストールした場所 ProductDir¥ にディレクトリーを変更し、その内容を表 3 のリストにある内容と比較します。

表 3 で、アダプターにより使用される Windows ファイル構造について説明し、インストーラーによるアダプターのインストールを選択したときに自動的にインストールされるファイルを示します。

表 3. コネクターの Windows ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
¥connectors¥JText¥CWJText.jar	JText コネクターのみに使用されるクラスを含みます。
¥connectors¥JText¥start_JText.bat	汎用コネクターの始動スクリプト
¥connectors¥messages¥JTextConnector.txt	コネクターのメッセージ・ファイル
¥repository¥JText¥CN_JText.txt	コネクターのリポジトリー定義。デフォルトの名前は CN_JText.txt です。
¥connectors¥JText¥start_JText_service.bat	コネクター・サービスの始動スクリプト
¥connectors¥JText¥dependencies¥commons-net-1.1.0.jar、dependencies¥jakarta-oro-2.0.8.jar	コネクターとパッケージされるサード・パーティーの jar

注: すべての製品のパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。

Linux システムでのインストールの検証

始める前に、アダプターをインストールします。インストーラーにより、アダプターに関連付けられている標準ファイルがシステムにコピーされます。このユーティリティーは、コネクター・エージェントを *ProductDir*/connectors/JText ディレクトリーにインストールします。

Linux システムでアダプターのインストールを検証するには、アダプターをインストールした場所 ProductDir/ にディレクトリーを変更し、その内容を表 9 のリストにある内容と比較します。

表 4 で、アダプターにより使用される Linux ファイル構造について説明し、インストーラーによるアダプターのインストールを選択したときに自動的にインストールされるファイルを示します。

表 4. コネクターの Linux ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
/connectors/JText/CWJText.jar	JText コネクターのみに使用されるクラスを含みます。
/connectors/JText/start_JText.sh	汎用コネクターの始動スクリプト
/connectors/messages/JTextConnector.txt	コネクターのメッセージ・ファイル
/repository/JText/CN_JText.txt	コネクターのリポジトリー定義。デフォルトの名前は CN_JText.txt です。
/connectors/JText/dependencies/commons-net-1.1.0.jar, dependencies/jakarta-oro-2.0.8.jar	コネクターとパッケージされるサード・パーティーの jar

注: すべての製品のパス名は、ご使用のシステムで製品がインストールされているディレクトリーを基準とした相対パス名です。

i5/OS システムでのインストールの検証

デフォルトでは、アダプターは /QIBM/ProdData/WBIServer44/product ディレクトリー (ProductDir という) にインストールされています。表 4 に、コネクターが使用する i5/OS ファイル構造と、標準インストール時に自動的にインストールされるファイルを示します。i5/OS システムでアダプターのインストールを検証するには、アダプターをインストールした場所にディレクトリーを変更し、その内容を表 5 のリストにある内容と比較します。

表 5. コネクターの i5/OS ファイル構造

ディレクトリー	**説明**
ProductDir/connectors/JText	コネクターの CWJText.jar ファイルと start_JText.sh 始動ファイルが格納されています。
ProductDir/repository/JText	BIA_CN_JText.txt リポジトリー定義が格納されています。

表 5. コネクターの i5/OS ファイル構造 (続き)

ディレクトリー	説明
ProductDir/connectors/JText/Dependencies	使用許諾契約書、REAME ファイル、およびサード・パーティーの commons-net-1.10.jar ファイルと jakarta-oro-2.0.0.jar ファイルが格納されています。
ProductDir/connectors/messages	JTextConnector.txt ファイルと JTextConnector_*ll*_TT.Ttxt ファイル (言語に固有のメッセージ・ファイル (*ll*) と国/地域に固有のメッセージ・ファイル (*TT*)) が格納されています。
ProductDir/lib	WBIA.jar ファイルが格納されています。
ProductDir/bin	CWConnEnv.sh ファイルが格納されています。

注: Console 機能を使用して、コネクターをすばやく始動することができます。詳しくは、Console に付属のオンライン・ヘルプを参照してください。

第 3 章 JText アダプターの構成

本章では、JText コネクターのインストールと構成の方法について説明します。

- 『Connector Configurator Express の概要』
- 24 ページの『Connector Configurator Express の始動』
- 25 ページの『System Manager からのコンフィギュレーターの実行』
- 25 ページの『コネクター固有のプロパティー・テンプレートの作成』
- 29 ページの『新規構成ファイルの作成』
- 30 ページの『既存ファイルの使用』
- 31 ページの『構成ファイルの完成』
- 32 ページの『構成ファイル・プロパティーの設定』
- 43 ページの『構成ファイルの保管』
- 44 ページの『構成ファイルの変更』
- 44 ページの『構成の完了』
- 44 ページの『グローバル化環境における Connector Configurator Express の使用』
- 45 ページの『コネクターの始動』
- 45 ページの『コネクターの始動』
- 47 ページの『コネクターの停止』
- 47 ページの『複数のコネクター・インスタンスの作成』
- 50 ページの『サポートされるビジネス・オブジェクトの追加』

Connector Configurator Express の概要

Connector Configurator Express では、InterChange Server Express で使用するアダプターのコネクター・コンポーネントを構成できます。

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成する。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定する。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義を指定し、InterChange Server Express によりコラボレーションで使用するマップを指定し、同時にロギング、メッセージング、およびトレースのパラメーターを指定し、さらに必要に応じてデータ・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 Express 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。

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

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

Connector Configurator Express の「標準のプロパティー」ウィンドウには、現在ご使用の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、26 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。

Linux でのコネクターの実行

Connector Configurator Express は、Windows 環境でのみ実行できます。Linux 環境でコネクターを実行する場合には、Windows でConnector Configurator Express を使用して構成ファイルを変更し、このファイルを Linux 環境へコピーします。

Connector Configurator Express のプロパティーの中には、ディレクトリー・パスを使用するものがあります。このパスは、Windows のディレクトリー・パスの規則がデフォルトになっています。Linux 環境で構成ファイルを使用する場合は、これらのパスの Linux の規則に一致するようにディレクトリー・パスを修正します。ツールバーのドロップ・リストでターゲットのオペレーティング・システムを選択し、拡張検証で正しいオペレーティング・システム規則が使用されるようにします。

Connector Configurator Express の始動

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

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

スタンドアロン・モードでのコンフィギュレーターの実行

System Manager を実行せずに Connector Configurator Express を実行し、コネクターの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**」>「**すべてのプログラム**」から、「**IBM WebSphere Business Integration Express**」>「**Toolset Express**」>「**開発**」>「**Connector Configurator Express**」をクリックします。
- 「**ファイル**」>「**新規**」>「**コネクター構成**」を選択します。
- 「**システム接続: Integration Broker**」の隣のプルダウン・メニューをクリックします。InterChange Server Express を選択します。

Connector Configurator Express を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存する方法が便利です (31 ページの『構成ファイルの完成』を参照)。

System Manager からのコンフィギュレーターの実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

- 「System Manager」ウィンドウの「コネクター」フォルダーで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 Express が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
- Connector Configurator Express で「**ファイル**」>「**開く**」を選択します。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクター構成ファイルを選択します。
- 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれ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、26 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 ¥ProductDir¥bin¥Data¥App ディレクトリーにあります。

新規テンプレートの作成

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

Connector Configurator Express で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
  プロパティー・サブタイプ
  更新されたメソッド
  説明
- **フラグ**
  標準フラグ

- **カスタム・フラグ**
  フラグ

「**プロパティー・サブタイプ**」は、「**プロパティー・タイプ**」が String の場合に選択することができます。これは、構成ファイルの保管時に構文検査を実行するオプションの値です。デフォルトはブランク・スペースで、これはプロパティーのサブタイプが指定されていないことを意味します。

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

値の指定

「**値**」タブを使用すると、プロパティーの最大長、最大複数値、デフォルト値、または値の範囲を設定できます。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. 「値」列見出しの左側にある正方形を右マウス・ボタンでクリックします。
2. ポップアップ・メニューから「**追加**」を選択して、「プロパティー値」ダイアログ・ボックスを表示します。このダイアログ・ボックスではプロパティーのタイプに応じて、値だけを入力できる場合と、値と範囲の両方を入力できる場合があります。
3. 新規プロパティー値を入力し、「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。入力した情報が、Connector Configurator Express によって、Connector Configurator Express がインストールされている `¥bin` ディレクトリーの `¥data¥app` の下に XML 文書として保管されます。

パス名の設定

パス名を設定する場合の一般的な規則のいくつかを次に示します。

- Windows、Linux、および i5/OS のファイル名の最大長は 255 文字です。
- Windows では、絶対パス名は [ドライブ:][ディレクトリー]¥filename: の形式に従う必要があります。例えば、
  C:¥WebSphereAdapters¥bin¥Data¥Std¥StdConnProps.xml のようにします。
  Linux では、最初の文字を「/」にします。
- キュー名には、先行スペースまたは埋め込みスペースを使用できません。

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、統合ブローカーとして InterChange Server Express を選択する必要があります。

また、ファイルで拡張検証するオペレーティング・システムを選択します。ツールバーには、「**ターゲット・システム**」というドロップ・リストがあり、ここから、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択できます。選択可能なオプションは、「Windows」、「Linux」、「i5/OS」、または「その他」(Windows でも Linux でも i5/OS でもない場合)、および「なし (拡張検証なし)」(拡張検証をオフに切り替え) です。始動時のデフォルトは Windows です。

Connector Configurator Express を始動するには、以下のステップを実行します。

- 「System Manager」ウィンドウで、「**ツール**」メニューから「**Connector Configurator Express**」を選択します。Connector Configurator Express が開きます。
- スタンドアロン・モードで Connector Configurator Express を起動します。

構成ファイルの拡張検証用のオペレーティング・システムを設定するには、以下のステップを実行します。

- メニュー・バーの「**ターゲット・システム:**」ドロップ・リストをプルダウンします。
- 稼働中のオペレーティング・システムを選択します。

次に、「**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。「**Integration Broker**」フィールドでブローカーを選択するには、「InterChange Server Express」を選択します。この章で後述する説明に従って、「**新規コネクター**」ウィンドウの残りのフィールドの入力を完了します。

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. メニュー・バーの「**ターゲット・システム:**」ドロップ・リストを使用して、構成ファイルの拡張検証用のオペレーティング・システムを設定します (前述の『新規構成ファイルの作成』を参照)。
2. 「**ファイル**」>「**新規**」>「**コネクター構成**」をクリックします。
3. 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 Express では、入力された名前のスペルはチェックされません。名前が正しいことを確認してください。

- **システム接続**

  「InterChange Server Express」をクリックします。

- **「コネクター固有プロパティー・テンプレート」を選択**

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

4. 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。
5. 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 Express のメッセージ・パネルの状況表示に、構成ファイルが正常に作成されたことが示されます。

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

- コネクター定義ファイル。
  コネクター定義ファイルは、特定のコネクターのプロパティーと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクターの配布パッケージの `¥repository` ディレクトリー内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は `.txt` です。例えば、JTEXT コネクターの場合は CN_JTEXT.txt です)。
- ICS リポジトリー・ファイル。
  以前にコネクターの InterChange Server Express インプリメンテーションの際に使用された定義が、そのコネクターの構成に使用されたリポジトリー・ファイルに残されていることがあります。そのようなファイルの拡張子は、通常 `.in` または `.out` です。

- コネクターの以前の構成ファイル。
  こ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 Express でそのファイルを開き、構成を修正してから、再度保管する必要があります。

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 Express で、「**ファイル**」>「**開く**」>「**ファイルから**」をクリックします。
2. 「**ファイル・コネクターを開く**」ダイアログ・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)

     リポジトリー・ファイルを使用して InterChange Server Express 環境のコネクターを構成した場合は、このオプションを選択します。リポジトリー・ファイルに複数のコネクター定義が含まれている場合は、ファイルを開くとすべての定義が表示されます。
   - すべてのファイル (*.*)

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

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

1. System Manager を始動します。System Manager が開始されている場合にのみ、構成を System Manager から開いたり、System Manager に保管したりできます。
2. Connector Configurator Express を始動します。
3. 「**ファイル**」>「**開く**」>「**プロジェクトから**」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 Express」ウィンドウに構成画面が表示されます。この画面には、現在の属性と値が表示されます。

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 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コネクターのプロパティーが「標準のプロパティー」タブに表示されます。表には、「**プロパティー名**」、「**値**」、「**タイプ**」、「**サブタイプ**」(タイプが String の場合)、「**説明**」、および「**更新メソッド**」が表示されます。
3. ここでファイルを保管するか、または 38 ページの『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。
4. 構成が完了したら、「**ファイル**」>「**保管**」>「**プロジェクトに**」を選択するか、または「**ファイル**」>「**保管**」>「**ファイルに**」を選択します。

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

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

   構成ファイルの作成前は、「**ターゲット・システム**」ドロップ・リストを使用して、プロパティーの拡張検証用のターゲット・オペレーティング・システムを選択できました。

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

   「**ターゲット・システム**」ドロップ・リストから、「Windows」、「Linux」、「i5/OS」、または「その他」の値を選択することによって拡張検証機能を使用するように選択した場合、システムはタイプだけでなくプロパティー・サブタイプも検証し、検証に失敗すると警告メッセージを表示します。

構成ファイル・プロパティーの設定

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

Connector Configurator Express では、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。

InterChange Server Express 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。

- 関連マップ
- セキュリティー

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

- コネクターの標準プロパティーは、コネクターのアプリケーション固有のコンポーネントとブローカー・コンポーネントの両方によって共用されます。すべてのコネクターが同じ標準プロパティーのセットを使用します。これらのプロパティーについては、標準のプロパティーの付録で説明します。変更できるのはこれらの値のすべてではなく一部のみです。
- 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。

   注: プロパティーのタイプが String の場合は、「サブタイプ」列にサブタイプ値が含まれている場合があります。このサブタイプは、プロパティーの拡張検証で使用します。

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 Express を終了するには、「**ファイル**」>「**終了**」をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら「**いいえ**」をクリックします。
   - Connector Configurator Express 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。「**標準のプロパティー**」(またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されますウィンドウを閉じると、すべてのカテゴリーで入力した値を一括して保管するかまたは破棄するかを確認するプロンプトが出されます。
   - 修正した値を保管するには、「**ファイル**」>「**終了**」をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら「**はい**」をクリックします。「ファイル」メニューまたはツールバーから「**保管**」>「**ファイルに**」をクリックする方法もあります。

特定の標準プロパティーの詳細を参照するには、「標準のプロパティー」タブ付きシートで該当するプロパティーの「説明」列にある項目を左マウス・ボタンでクリックします。全般ヘルプがインストールされている場合は、右側に矢印ボタンが表示されます。ボタンをクリックすると、「ヘルプ」ウィンドウが開き、標準プロパティーの詳細が表示されます。

注: ホット・ボタンが表示されない場合は、そのプロパティーの全般ヘルプが検出されなかったことを示します。

インストールされている場合、全般ヘルプ・ファイルは
<*ProductDir*>¥bin¥Data¥Std¥Help¥<*RegionalSetting*>¥ にあります。

コネクター固有の構成プロパティーの設定

コネクター固有の構成プロパティーの場合、プロパティー名の追加または変更、値の構成、プロパティーの削除、およびプロパティーの暗号化が可能です。プロパティーのデフォルトの長さは 255 文字です。

1. グリッドの左上端の部分で右マウス・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。

   注: プロパティーのタイプが String の場合は、「サブタイプ」ドロップ・リストからサブタイプを選択できます。このサブタイプは、プロパティーの拡張検証で使用します。

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. 特定のプロパティーの詳細を参照するには、該当するプロパティーの「説明」列にある項目を左マウス・ボタンでクリックします。全般ヘルプがインストールされている場合は、ホット・ボタンが表示されます。ホット・ボタンをクリックすると、「ヘルプ」ウィンドウが開き、標準プロパティーの詳細が表示されます。

   注: ホット・ボタンが表示されない場合は、そのプロパティーの全般ヘルプが検出されなかったことを示します。

5. 33 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。

全般ヘルプ・ファイルがインストールされており、AdapterHelpName プロパティーがブランクの場合、Connector Configurator Express は <*ProductDir*>¥bin¥Data¥App¥Help¥<*RegionalSetting*>¥ に配置されているアダプター固有の全般ヘルプ・ファイルを指します。それ以外の場合、Connector Configurator Express は、<*ProductDir*>¥bin¥Data¥App¥Help¥<*AdapterHelpName*>¥<*RegionalSetting*>¥ に配置されているアダプター固有の全般ヘルプ・ファイルを指します。標準プロパティーの付録にある AdapterHelpName プロパティーの説明を参照してください。

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。また、このプロパティーには、コネクター内の静的情報やロジックを、コネクターの再コーディングや再ビルドなしに変更する方法が指定されます。

表 6 に、コネクターのコネクター固有の構成プロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

表 6. コネクター固有の構成プロパティー

名前	指定可能な値	デフォルト値	必須
ArchivingEnabled	`true` または `false`	`true`	`Yes`
EventLog	*ファイル名とファイルの場所*	`event.log`	`No`
EventRecovery	`abort` または `retry`	`retry`	`Yes`
FTPPollFrequency	*ポーリング・サイクル数*		`No`
GenerateTemplate	*BOName*		`No`
OutputLog	*要求の処理中に、着信ビジネス・オブジェクトごとの次のシーケンス番号を登録するファイル*	`Output.Log`	`No`

表 6. コネクター固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
PollQuantity	*各ポーリング時に処理されるイベントの数*	25	No
SortFilesOnTimestamp	true または false	False	No
NoPoll	true または false	False	No

ArchivingEnabled: アーカイブをオンにします。このプロパティーを true に設定した場合、イベント・ファイルがアーカイブ・ディレクトリーに指定の拡張子でアーカイブされます。このプロパティーを false に設定した場合は、イベント・ファイルはアーカイブされません。この場合、コネクターがすべてのイベントを統合ブローカーに送信した後、このファイルを削除します。詳細については、67 ページの『イベント・アーカイブの指定』を参照してください。

デフォルト値は true です。

EventLog: コネクターによって生成されたイベントのファイルの保管場所を示します。このファイルは、この製品がインストールされている connectors ディレクトリーの JText サブディレクトリーの中にあります。

デフォルト値は event.log です。

EventRecovery: リカバリー動作を指定します。このプロパティーを retry に設定した場合、コネクターは event.log ファイルを使用して失敗したイベントをリカバリーします。このプロパティーを abort に設定した場合、コネクターは失敗したイベントを検出すると終了します。詳細については、95 ページの『イベント・ログ・ファイル』を参照してください。

デフォルト値は retry です。

FTPPollFrequency: 標準のポーリング・サイクル数を基準として、コネクターによる FTP サーバーのポーリング回数を決定します。例えば PollFrequency 標準構成プロパティーが 10000 に設定され、FTPPollFrequency が 6 に設定されている場合、コネクターはローカル・イベント・ディレクトリーに 10 秒ごとにポーリングし、リモート・ディレクトリーに 60 秒ごとにポーリングします。コネクターが FTP ポーリングを実行するのは、このプロパティーに値を指定した場合のみです。FTPPollFrequency に 0 またはブランクが設定された場合、コネクターは FTP ポーリングを実行しません。デフォルトでは、ポーリングを実行しません。

このプロパティーのデフォルト値はありません。

GenerateTemplate: コネクターの始動後に、コネクターがサポートされる各ビジネス・オブジェクト用のテンプレートを生成できるようにします。このプロパティーの構文は *BOName*;*BOName* になります。*BOName* には、特定のビジネス・オブジェクトの名前が入ります。例えば、Customer ビジネス・オブジェクトと Item ビジネス・オブジェクトの 2 つのテンプレートを生成する場合、Customer;Item と指定します。詳細については、91 ページの『テスト用サンプル・ビジネス・オブジェクトの生成』を参照してください。

このプロパティーのデフォルト値はありません。

OutputLog: 要求の処理中に、コネクターがビジネス・オブジェクトのタイプごとの固有の出力ファイルの作成時に使用するシーケンス番号を格納するファイルの名前を指定します。ファイルのフォーマットは以下のようになります。

BusinessObjectName = NextSequenceNumber

ここで、*BusinessObjectName* は要求ビジネス・オブジェクトの名前で、*NextSequenceNumber* は最近受け取ったビジネス・オブジェクトのシーケンス番号に 1 を加えた値を表します。例えば、コネクターが Customer と Item というビジネス・オブジェクトを処理している場合に、出力ログ・ファイルに以下の項目が入っているとします。

```
Customer = 12
Item = 2
```

このファイルは、コネクターがすでに 11 の Customer と 1 つの Item を処理済みであることを示しています。次の Customer と Item のビジネス・オブジェクトはそれぞれ、Customer_12.out ファイルと Item_2.out ファイルに書き込まれます。コネクターが、要求 Order ビジネス・オブジェクトを受け取ると、出力ログ・ファイルに新しい行を追加して、このビジネス・オブジェクトを Order_1.out ファイルに書き込みます。

FileSeqEnabled メタオブジェクトが true に設定されている場合、コネクターはこのシーケンス番号を使用して、各ビジネス・オブジェクト用に作成する出力ファイルに一意の名前を付けます。コネクターが各出力ファイルに名前を付けるとき、ビジネス・オブジェクトの名前、または OutputFileName メタオブジェクト属性に名前が指定されているファイルに下線 (_) とシーケンス番号を付加します。出力ログはユーザーが読み取り可能なフォーマットで格納されるため、標準のテキスト・エディターを使用してこのファイルを読んだり、値をリセットしたりすることができます。

OutputFileName 属性の詳細については、 64 ページの『出力ファイル名の指定』を参照してください。出力ログの詳細については、 69 ページの『要求処理の指定』を参照してください。生成されたファイルの名前を元に戻す方法については、 71 ページの『ファイル名の戻り』を参照してください。

デフォルトは Output.Log です。

PollQuantity: ポーリングごとに処理するイベントの数を指定します。コネクターのポーリング・メソッドにより、指定の数のイベント・レコードを検索し、検索されたイベント・レコードを 1 回のポーリングで処理します。1 回のポーリングで複数のイベントを処理することで、アプリケーションが多数のイベントを生成するときのパフォーマンスを向上させることができます。ただし、ポーリング・メソッドによるイベントの処理中には統合ブローカーの要求がブロックされるため、イベント数にあまり大きい数を設定しないでください。各ポーリング呼び出しに時間がかかると、統合ブローカーの要求処理が遅れます。詳細については、 89 ページの『JText コネクターのパフォーマンス調整』を参照してください。

デフォルト値は 25 です。

SortFilesOnTimestamp: アダプターは、タイム・スタンプに基づいてイベント・ファイルを選択できます。このプロパティーは、ポーリング時にアダプターによって

少数の巨大なイベント・ファイルだけが選択された場合に、true に設定するのが最善です。小さなイベント・ファイルが数多く存在する場合は、各ポーリングでファイルをソートするためにかかる時間が超過しないようにするため、この値を false に設定します。

デフォルト値は `False` です。

注: タイム・スタンプに基づくソートは、Windowsi5/OS および Linux のプラットフォームでのみサポートされています。タイム・スタンプに基づくソートは、MVS プラットフォームではサポートされていません。また、JText アダプターは、FTP サーバーが `mdtm` (変更日時) コマンドをサポートする場合のみ、FTP サイトのタイム・スタンプに基づいてリモート・ファイルをポーリングすることができます。

NoPoll: このプロパティーが true に設定されている場合、アダプターはイベント処理を実行しません。アダプターは、要求処理の場合にのみ使用できます。

デフォルト値は `False` です。

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

標準のプロパティーの付録にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

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。

ブローカーとしての InterChange Server Express

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・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの「**エージェント・サポート**」(以下で説明) を設定します。
4. 「Connector Configurator Express」ウィンドウの「ファイル」メニューで、「**プロジェクトに保管**」をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

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

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator Express」ウィンドウの「**編集**」メニューから、「**行を削除**」をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. 「**ファイル**」メニューから、「**プロジェクトの保管**」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 Express」ウィンドウでは、「エージェント・サポート」を選択しても問題ないかどうかの検証は行われません。

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

ほとんどのコネクターの場合、選択可能な項目は「最大限の努力」のみです。

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

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

各コネクターは、ビジネス・オブジェクト定義とそれらに関連付けられたマップのうち現在 InterChange Server Express でアクティブであるものを示すリストをサポートします。このリストは、「**関連付けられたマップ**」タブを選択すると表示されます。

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

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

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

「**関連付けられたマップ**」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的バインディング**

  場合によっては、関連マップを明示的にバインドすることが必要になります。

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

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

以下の手順を実行して、マップを明示的にバインドします。

1. 「**明示的 (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 Express」ウィンドウの「**ファイル**」メニューで、「**プロジェクトに保管**」をクリックします。
4. プロジェクトを InterChange Server Express に配置します。
5. 変更を有効にするため、サーバーをリブートします。

InterChange Server Express のセキュリティー

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

デフォルトでは、プライバシーはオフになっています。使用可能にするには、「**プライバシー**」ボックスにチェック・マークを付けます。

「**鍵ストア・ターゲット・システムの絶対パス名**」は次のとおりです。

- Windows の場合:
  `<ProductDir>¥connectors¥security¥<connectorname>.jks`
- Linux の場合:
  `/ProductDir/connectors/security/<connectorname>.jks`
- i5/OS の場合:
  /ProductDir/Connectors/security/<connectoryname>.jks

このパスとファイルは、コネクターを開始する予定のシステム、つまりターゲット・システム上に存在していなければなりません。

右側の「参照」ボタンは、ターゲット・システムが現在稼働中のシステムである場合にのみ使用できます。「**プライバシー**」が使用可能で、メニュー・バーの「**ターゲット・システム**」が「Windows」に設定されていなければ、これはグレー表示になります。

「**メッセージのプライバシー・レベル**」は、3 種類のメッセージ・カテゴリー (「全メッセージ」、「全管理メッセージ」、および「全ビジネス・オブジェクト・メッセージ」) ごとに次のように設定されます。

- “” はデフォルトです。メッセージ・カテゴリーに対してプライバシー・レベルが何も設定されていない場合に使用します。
- none
  これは、デフォルトと同じではありません。これは、メッセージ・カテゴリーに対してプライバシー・レベルを故意に none に設定する場合に使用します。
- integrity
- privacy
- integrity_plus_privacy

「**鍵の保守**」機能により、サーバーおよびアダプターの公開鍵を生成、インポート、およびエクスポートできます。

- 「**鍵の生成**」を選択すると、「鍵の生成」ダイアログ・ボックスが表示され、鍵を生成する keytool のデフォルトが示されます。
- 鍵ストア値のデフォルトは、「セキュリティー」タブの「**鍵ストア・ターゲット・システムの絶対パス名**」に入力した値に設定されます。
- 「OK」を選択すると、項目の検証が行われ、鍵証明書が生成されて、出力が Connector Configurator Express ログ・ウィンドウに送信されます。

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

- エクスポート証明書は、デフォルトでは、ファイルの拡張子が <filename>.cer である点を除いて、鍵ストアと同じ値になります。

「**サーバー公開鍵のインポート**」を選択すると、「サーバー公開鍵のインポート」ダイアログ・ボックスが表示されます。

- インポート証明書は、デフォルトでは、(システムにファイルが存在していれば) <*ProductDir*>¥bin¥ics.cer になります。
- インポート証明書関連はサーバー名でなければなりません。サーバーが登録されている場合は、ドロップ・リストからそれを選択できます。

アダプター・アクセス制御機能は、DeliveryTransport の値が IDL になっている場合にのみ使用可能になります。デフォルトでは、アダプターはゲスト ID を使用してログインします。「**ゲスト ID の使用**」ボックスにチェック・マークが付いていない場合は、「**アダプター ID**」フィールドと「**アダプター・パスワード**」フィールドが使用可能になります。

トレース/ログ・ファイル値の設定

コネクター構成ファイルまたはコネクター定義ファイルを開くと、Connector Configurator Express は、そのファイルに含まれるロギングとトレースに関する値をデフォルト値として使用します。これらの値は、Connector Configurator Express 内で変更できます。

ログとトレースの値を変更するには、以下の手順を実行します。

1. 「**トレース/ログ・ファイル**」タブをクリックします。

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

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

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

データ・ハンドラー

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

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

構成ファイルの保管

コネクターの構成が完了したら、コネクター構成ファイルを保管します。Connector Configurator Express では、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 Express のタイトル・バーには、現在使用されているブローカー・モード (InterChange Server Express) が常に表示されます。

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

- System Manager から、統合コンポーネント・ライブラリーに `*.con` 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに `*.cfg` 拡張子付きファイルとして保管します。デフォルトでは、ファイルが `¥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、「システム・インプリメンテーション・ガイド」を参照してください。

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、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。

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 Express 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・ウィンドウの利用可能なタブとフィールドの選択項目がただちに変更され、選択した新規ブローカーに関係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 Express の使用

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

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

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

`CharacterEncoding` および `Locale` 標準構成プロパティーのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サポート

される他の値を追加するには、製品ディレクトリーの ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。

例えば、Locale プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太字で示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>
```

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。Windows システムでは、始動スクリプトはコネクターのランタイム・ディレクトリー *ProductDir*¥connectors¥*connName* に存在していなければなりません。ここで、*connName* はコネクターを示します。

Linux システムでは、始動スクリプトは *ProductDir*/bin ディレクトリーに存在していなければなりません。

i5/OS システムでは、始動スクリプトは、コネクターの実行に使用する /QIBM/UserData/WBIServer44/<instance>/connectors/<ConnInstance/ に存在しなければなりません。

始動スクリプトの名前は、表 7 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 7. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
Linux ベース・システム	connector_manager
i5/OS	start_connName.sh
Windows	start_*connName*.bat

始動スクリプトが実行されると、デフォルトで構成ファイルは *Productdir* にあることが要求されます (下記のコマンドを参照)。ここに構成ファイルを格納します。

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

Windows システムでのコネクターの開始

- 「**スタート**」メニューから、「**プログラム**」>「**IBM WebSphere Business Integration Server Express**」>「**アダプター**」>「**コネクター**」を選択します。デフォルトでは、プログラム名は「IBM WebSphere Business Integration Server Express」となっています。ただし、これはカスタマイズすることができます。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。
- Windows コマンド行から、次を入力します: start_connName connName brokerName {-cconfigFile}
- Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、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。

Linux ベース・システムでのコネクターの開始

- コマンド行から、以下を入力します。

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

  ここで、*connName* はコネクターの名前であり、*brokerName* はご使用の統合ブローカーを表します。
- InterChange Server Express の場合は、*brokerName* に InterChange Server Express インスタンスの名前を指定します。

i5/OS システムでのコネクターの開始

- WebSphere Business Integrations Server Express Console がインストールされている Windows システムから、「**IBM WebSphere Business Integration Server Express**」>「**Toolset Express**」>「**管理**」>「**コンソール**」を選択します。次に、OS/400 または i5/OS システム名または IP アドレスと、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。コネクターのリストからコネクターを選択して、「開始」をクリックします。
- コンソールを使用してアダプターを自動的に開始するには、submit_adapter.sh スクリプトを使用します。これが、サーバーの自動開始ジョブ・エントリー内のサブシステムを使用してアダプターが開始する唯一の方法です。
- バッチ・モードでは、i5/OS コマンド行から CL コマンド QSH を実行し、QSHELL 環境から /QIBM/ProdData/WBIServer44/bin/submit_adapter.sh *connName WebSphereICSName pathToConnNameStartScript jobDescriptionName* を実行する必要があります。ここで、*connName* はコネクター名、*WebSphereICSName* は Interchange Server Express サーバー名 (デフォルトは QWBIDFT44)、*pathToConnNameStartScript* はコネクター始動スクリプトの絶対パス、*jobDescriptionName* は QWBISVR44 ライブラリーで使用するジョブ記述の名前です。
- 対話モードでは、CL コマンド QSH を実行し、QSHELL 環境から */QIBM/UserData/WBIServer44/WebSphereICSName/connectors/connName/start_connName.sh connNameWebsphereICSName* [-cConfigFile] を実行する必要があります。ここで、*connName* はコネクターの名前であり、*WebSphereICSName* は InterChange Server Express インスタンスの名前です。

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、「システム管理ガイド」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、コネクターが始動された方法によって異なります。

- **Windows:**
  - コネクター用の別個の「コンソール」ウィンドウを作成する始動スクリプトを起動できます。このウィンドウで、「q」と入力して Enter キーを押すと、コネクターが停止します。
  - コネクターを Windows のサービスとして始動するように構成できます。この場合、Windows システムのシャットダウン時に、コネクターは停止します。
- **i5/OS:**
  - コンソールを使用して、または QSHELL で「submit_adapter.sh」スクリプトを使用してコネクターを始動した場合は、次の 2 つの方法のうちの 1 つを使用してコネクターを停止できます。
  - WebSphere Business Integration Server Express Console がインストールされている Windows システムから、「**IBM WebSphere Business Integration Express**」>「**Toolset Express**」>「**管理**」>「**コンソール**」を選択します。次に、OS/400 または i5/OS システム名または IP アドレスと、*JOBCTL 特殊権限を持つユーザー・プロファイルおよびパスワードを指定します。リストから JText アダプターを選択して、「停止」ボタンを選択します。 CL コマンド WRKACTJOB SBS (QWBISVR44) を使用して Server Express 製品に対するジョブを表示します。リストをスクロールして、コネクターのジョブ記述に一致するジョブ名を持つジョブを探します。例えば、JText コネクターの場合のジョブ名は QWBIJTEXTC です。「コントロール パネル」このジョブに対してオプション 4 を選択し、F4 を押して ENDJOB コマンドのプロンプトを取得します。次に、オプション・パラメーターとして *IMMED を指定し、Enter を押します。

    注: QWBISVR44 サブシステムが終了すると、コネクターは終了します。
  - QSHELL から start_connName.sh スクリプトを使用してアダプターを始動した場合は、F3 を押してコネクターを終了します。/QIBM/ProdData/WBIServer44/bin ディレクトリーにあるスクリプト *stop_adapter.sh* を使用して、エージェントを停止することもできます。
- **Linux:**

  コネクターはバックグラウンドで実行されるため、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 -stop connName
  ```

  ここで、*connName* はコネクターの名前です。

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

- **Windows プラットフォームの場合:**

  ProductDir¥connectors¥connectorInstance

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

  ProductDir¥repository¥connectorInstance

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

  InterChange Server Express サーバー名は、例えば start_JText.bat connName serverName のように、startup.bat のパラメーターとして指定できます。
- **i5/OS プラットフォームの場合:**

  `/QIBM/UserData/WBIServer44/WebShereICSName/connectors/connectorInstance`

  ここで、connectorInstance はコネクター・インスタンスを一意的に示し、WebSphereICSName はコネクターの実行に使用する Interchange Server Express インスタンスの名前です。

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

  /QIBM/UserData/WBIServer44/WebSphereICSName/repository/connectorInstance。ここで WebSphereICSName はコネクターの実行に使用する Interchange Server Express インスタンスの名前です。
- **Linux プラットフォームの場合**:
  ProductDir/connectors/connectorInstance
  ここで connectorInstance は、コネクター・インスタンスを一意的に示します。コネクターに、コネクター固有のメタオブジェクトがある場合、コネクター・インスタンス用のメタオブジェクトを作成する必要があります。メタオブジェクトをファイルとして保管する場合は、ディレクトリー
  ProductDir/repository/connectorInstance を作成し、ファイルをここに格納します。
  InterChange Server Express のサーバー名を connector_manager のパラメーターとして指定することができます。
  例: connector_manager -start connName WebSphereICSName [-cConfigFile]。

ビジネス・オブジェクト定義の作成

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 Express を使用してそれらのファイルをインポートします。初期コネクターの任意のファイルをコピーできます。変更を加えた場合は、名前を変更してください。
2. 初期コネクターのファイルは、次のディレクトリーに入っていなければなりません。

   ```
   ProductDir¥repository¥initialConnectorInstance
   ```

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

コネクター定義の作成

Connector Configurator Express 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、 48 ページの『ビジネス・オブジェクト定義の作成』で作成したコネクター・ディレクトリーに格納します。
3. (Windows の場合のみ) 始動スクリプトのショートカットを作成します。
4. (Windows の場合のみ) 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。
5. (i5/OS の場合のみ) 次の情報を使用して、コネクターのジョブ記述を作成します。
   CRTDUPOBJ(QWBIJTEXT) FROMLIB(QWBISVR44)OBJTYPE(*JOBD) TOLIB (QWBISVR44) NEWOBJ (newjtextname)。ここで、newjtextname は新規コネクターのジョブ記述で使用する 10 文字の名前です。
6. (i5/OS の場合のみ) 新規コネクターを WebSphere Business Integration Server Express Console に追加します。WebSphere Business Integration Server Express Console の詳細については、コンソールに付属のオンライン・ヘルプを参照してください。

サポートされるビジネス・オブジェクトの追加

デフォルトでは、JText コネクターは MO_JTextConnector_Default および MO_DataHandler_Default メタオブジェクトをサポートします。コネクターを完全に構成するためには、Connector Configurator Express を使用して、サポートされるビジネス・オブジェクトのリストに、必要なその他のビジネス・オブジェクトを追加してください。コネクターの使用法によっては、以下のすべてまたは多数のビジネス・オブジェクトを追加する必要があります。

- データ・ハンドラーのメタオブジェクト (MO_JTextConnector_Default メタオブジェクトの EventDataHandler 属性および OutputDataHandler 属性に指定されているもの)。デフォルトでは、これらの属性は NameValue データ・ハンドラーを表す MO_DataHandler_DefaultNameValueConfig データ・ハンドラー・メタオブジェクトを指定します。詳細については、86 ページの『データ・ハンドラーの指定』を参照してください。
- MO_JTextConnector_BOName*BusObjName* — 特定のビジネス・オブジェクト用のメタオブジェクトを作成する場合。詳細については、87 ページの『特定のビジネス・オブジェクトの JText メタオブジェクトの作成』を参照してください。
- ファイルから読み取られる、またはファイルに書き込まれるビジネス・オブジェクト。詳細については、3 ページの『JText コネクターが使用するビジネス・オブジェクト』を参照してください。

第 4 章 JText コネクター・メタオブジェクトの使用

メタオブジェクトは WebSphere Business Integration Adapters ビジネス・オブジェクトで、コネクターまたはデータ・ハンドラーによって使用される構成情報が含まれています。 JText コネクターでは、サポートされる各ビジネス・オブジェクトが、そのビジネス・オブジェクト・タイプの関連する JText メタオブジェクトを持つ必要があります。このトップレベル・メタオブジェクトには、1 つ以上の子メタオブジェクトが含まれます。

- コネクターは、トップレベル JText メタオブジェクトを使用して構成情報を入手します。構成情報には、データ変換に使用するデータ・ハンドラー、ビジネス・オブジェクトのイベント、アーカイブおよび出力ディレクトリーのパス、ビジネス・オブジェクトのイベント、アーカイブ、および出力ファイルのファイル拡張子、コネクターが FTP システムでファイルを処理する場合に必要な情報、コネクターが出力ファイルに固有のファイル ID を生成するかどうか、などがあります。
- コネクターは、データ・ハンドラーがビジネス・オブジェクトとストリング (またはバイト配列) の間でデータ変換を行う際に使用する構成値を、子メタオブジェクトによって指定します。デフォルトでは、トップレベル・メタオブジェクトは NameValue データ・ハンドラーを指定して、データを変換します。

コネクターがサポートするビジネス・オブジェクトにそれぞれ異なる構成情報を提供するには、個々にカスタム・トップレベル JText メタオブジェクトを作成します。各トップレベル・メタオブジェクトが自身のデータ・ハンドラー・メタオブジェクトを指定するので、コネクターはさまざまな形式のビジネス・オブジェクト・タイプを処理することができます。データ・ハンドラー・メタオブジェクトによって、新しいデータ・フォーマットを導入したり既存のフォーマットを変更するときに、ビジネス・オブジェクト定義を編集したりコネクター自体を変更する必要がなくなります。

メタオブジェクトは始動時にメモリーにロードされ、コネクターが構成情報を使用できるようになります。メタオブジェクトは、処理のために統合ブローカーへは送信されません。メタオブジェクトが影響を及ぼすのは、コネクターの振る舞いのみです。

この章では、JText メタオブジェクトを使用して JText コネクターを構成する方法について説明します。データ・ハンドラー・メタオブジェクトの使用についての詳細は、「データ・ハンドラー・ガイド」を参照してください。この章には以下のトピックが含まれます。

- 52 ページの『JText メタオブジェクトの命名規則』
- 52 ページの『JText メタオブジェクトの構造』
- 66 ページの『共通の構成タスク』

JText メタオブジェクトの命名規則

トップレベル JText メタオブジェクトの名前には、3 つのコンポーネントがあります。例えば、デフォルトのトップレベル・メタオブジェクト名 MO_JTextConnector_Default のようになります。トップレベル JText メタオブジェクト名のコンポーネントは、以下のとおりです。

- MO_ は、メタオブジェクトを示すプレフィックスです。
- ConnectorInstanceName_ は、コネクター・インスタンス (JText など) の名前を示します。この名前は、複数のコネクター・インスタンスをサポートするために構成可能です。例えば、JText2 という名前のコネクターには、メタオブジェクト名 MO_JText2Connector_Default が付けられます。
- Default は、関連するビジネス・オブジェクトを示します。特定のビジネス・オブジェクトのメタオブジェクトを作成するには、ストリング *Default* をビジネス・オブジェクトの名前に変更します。例えば、*Customer* というビジネス・オブジェクトの場合、MO_JTextConnector_Customer にします。メタオブジェクト名には、追加コンポーネントと下線を組み込むことができます。対応するビジネス・オブジェクト固有のメタオブジェクトが存在しない場合、コネクターはデフォルトのメタオブジェクトを使用します。

特定のビジネス・オブジェクト用のメタオブジェクトの作成についての詳細は、87 ページの『特定のビジネス・オブジェクトの JText メタオブジェクトの作成』を参照してください。

JText メタオブジェクトの構造

JText メタオブジェクトは階層構造になっています。デフォルトのトップレベル・メタオブジェクトは、MO_JTextConnector_Default です。トップレベル・メタオブジェクトの 2 つの属性 EventDataHandler および OutputDataHandler は、コネクターが使用するデータ・ハンドラーの構成情報を提供する子メタオブジェクトを表します。コネクターは、データ・ハンドラーを使用して、ビジネス・オブジェクトとストリング (またはバイト配列) の間でデータ変換を行います。

デフォルトでは、これらの 2 つの属性は同じデータ・ハンドラー・メタオブジェクト (MO_DataHandler_DefaultNameValueConfig) を指定します。このデータ・ハンドラー・メタオブジェクトが NameValue データ・ハンドラーを呼び出して、実際にデータを変換します。つまり、出荷時のデフォルト構成は、イベントおよび出力ファイル変換が同一のデータ・ハンドラーを使用することを指定します。データ・ハンドラーのインスタンス作成についての詳細は、「データ・ハンドラー・ガイド」を参照してください。

注: フォーマッターはデータ・ハンドラーに使用すべきでないので、以前にフォーマッターを表していた EventFormat 属性および OutputFormat 属性は、MO_JTextConnector_Default メタオブジェクトから除去されました。フォーマッターを使用するには、以下を実行する必要があります。

- トップレベル・メタオブジェクトに、EventFormat 属性および OutputFormat 属性を追加します。
- これらの属性のタイプとして適切なビジネス・オブジェクトを指定します。

- EventDataHandler 属性および OutputDataHandler 属性のタイプを String に変更します。

フォーマッターの使用についての詳細は、JText コネクターの 3.0.0 または 2.3.0 リリースの資料を参照してください。

図 5 は、デフォルト JText メタオブジェクトの階層構造および各属性の名前とタイプを示しています。

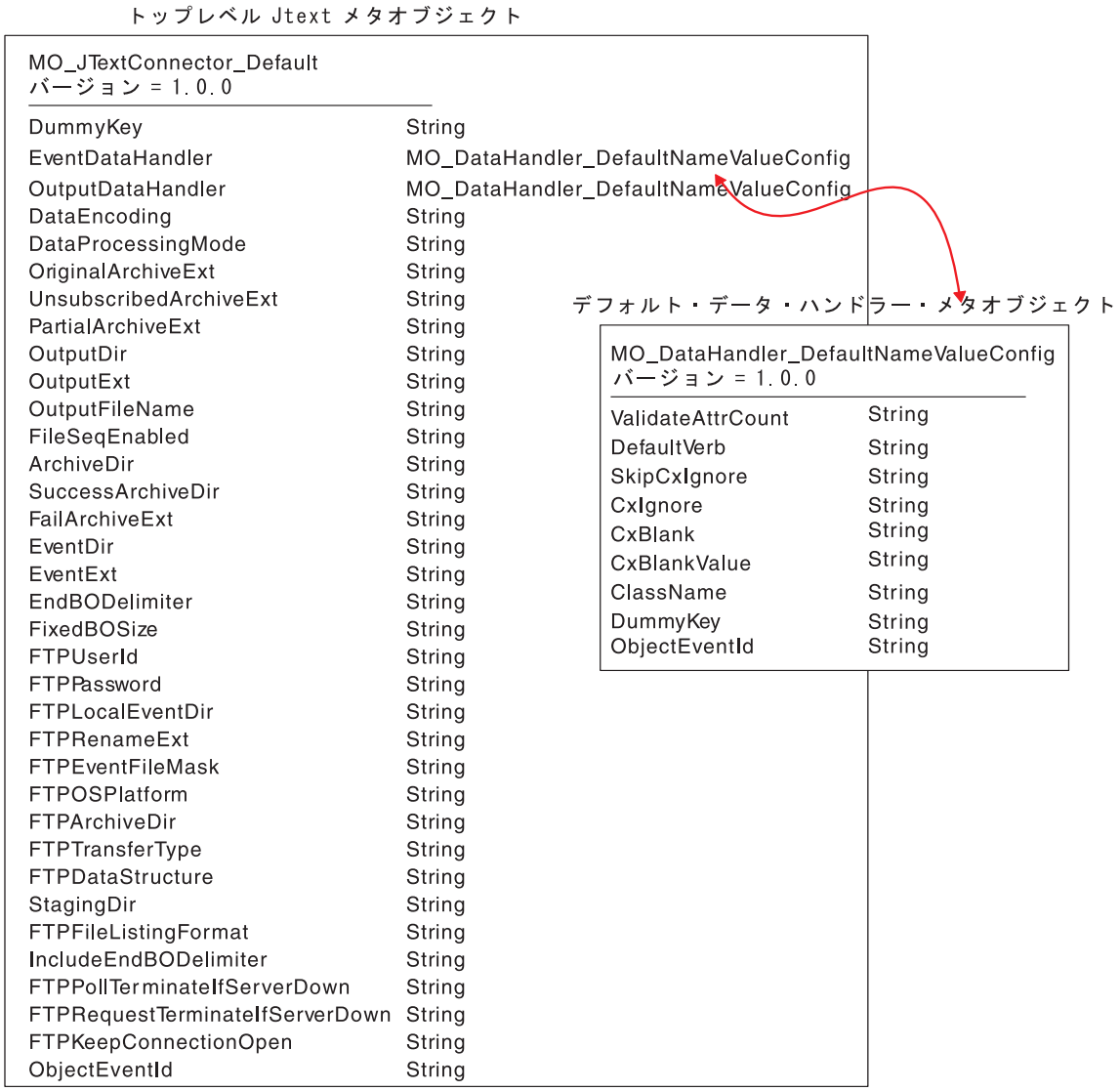

図 5. JText メタオブジェクトの階層構造

カスタム・メタオブジェクトの作成

トップレベル JText メタオブジェクト MO_JTextConnector_Default には、コネクターの構成情報と子メタオブジェクトが含まれています。コネクターが処理するビジネス・オブジェクトのタイプごとに、個別のトップレベル・メタオブジェクトを作

成できます。こうしたカスタム・メタオブジェクトに同一または異なる子メタオブジェクトを持たせて、データ・ハンドラーのタイプを構成することができます。例えば、Customer ビジネス・オブジェクトと Item ビジネス・オブジェクトを処理が異なるように構成するには、MO_JTextConnector_Customer メタオブジェクトと MO_JTextConnector_Item メタオブジェクトを作成し、異なるデータ・ハンドラー・メタオブジェクトを持つようにトップレベル・メタオブジェクトを設計します。

初期設定時に、コネクターはサポートされているメタオブジェクトとビジネス・オブジェクトのリストを統合ブローカーで検索します。これらのオブジェクト名から、コネクターは、どのビジネス・オブジェクトが自身に関連するトップレベル・メタオブジェクトを持っているかを判別します。実行時に、コネクターは要求ビジネス・オブジェクト名とサポートされているメタオブジェクトのいずれかとを突き合わせて、適切な構成情報を検索します。

例えば、コネクターが以下のメタオブジェクトをサポートしているとします。

- MO_JTextConnector_Default
- MO_JTextConnector_Customer
- MO_JTextConnector_Item

ビジネス・オブジェクトは以下のとおりです。

- Customer
- Item
- Order

統合ブローカーが要求 Customer ビジネス・オブジェクトを送信すると、コネクターは MO_JTextConnector_Customer メタオブジェクトに指定された構成情報を使用します。統合ブローカーが要求 Order ビジネス・オブジェクトを送信すると、コネクターは MO_JTextConnector_Default メタオブジェクトに指定された構成情報を使用します。

MO_JTextConnector_Default 属性

このセクションでは、MO_JTextConnector_Default メタオブジェクトの属性について説明します。

注: 属性の DefaultValue プロパティーの値には、大文字小文字の区別があります。ディレクトリー情報はディレクトリーの絶対パスを指定する必要があります。

55 ページの表 8 および以下のセクションでは、MO_JTextConnector_Default メタオブジェクトの各属性の機能について説明します。その他の情報では、この表には各単純属性の DefaultValue プロパティーの値が含まれています。製品提供時の値は、ユーザー独自の値に置き換えることができます。

表 8. *MO_JTextConnector_Default* メタオブジェクト定義の属性

属性名	説明
ArchiveDir	アーカイブ・ディレクトリーの絶対パスを指定します。ディレクトリーは既存のものである必要があります。出荷時のデフォルト値は以下のとおりです。 **Linux:** /tmp/JTextConn/Default/archive **Windows:** C:¥temp¥JTextConn¥Default¥Archive **i5/OS:** /QIBM/UserData/WBIServer44/<instance>/connectors/JText/archive Windows 対応の BIDI です。BIDI が使用可能なすべてのプロパティーは、Windows BIDI 形式から、JText コネクター標準プロパティーである BiDi.Metadata 形式に変換されます。
DataEncoding	DataEncoding は、ビジネス・オブジェクト・ストリングの読み取りおよび書き込みに使用するエンコードです。静的メタオブジェクトでこのプロパティーが指定されていない場合は、コネクターは、特定のエンコードを使用せずにビジネス・オブジェクト・ストリングの読み取りまたは書き込みを行います。この属性には、Java でサポートされている任意のエンコード・セットを指定できます。
DataProcessingMode	この属性は、バイナリー・ファイルの読み取りと書き込みを可能にするために追加されました。この MO プロパティーが Binary に設定されている場合、JText では、適切なデータ・ハンドラー・インターフェースを呼び出して BO からバイト配列への変換またはその逆方向の変換を実行し、ファイル・システムのバイナリー・ファイルを読み取ることや書き込むことができます。通常の設定は Text です。Text モードでは、BO からストリングへの変換、またはその逆方向の変換のためのデータ・ハンドラー・インターフェースが使用されます。このプロパティーは、設定されていない場合、デフォルト値の Text になります。Binary モードを使用するのは、getBO(byte[]) および getByteArrayFromBO() メソッドが適切に実装されているデータ・ハンドラーとともに使用する場合に限る必要があります。
DummyKey	この属性は、各ビジネス・オブジェクト定義の 1 つの属性で Key プロパティーが使用可能になっている必要があるために存在します。
EndBODelimiter	入力ファイル内のビジネス・オブジェクトを分離する区切り文字を指定します。EndBODelimiter 属性の詳細については、74 ページの『特定のビジネス・オブジェクトのポーリング』を参照してください。 構成時にデフォルト値を指定していない場合、このプロパティーのデフォルト値は、DataProcessingMode が Text に設定されていれば <EndBO:*BOName*> になります。DataProcessingMode が Binary のとき、このプロパティーのデフォルト値は FF01 になります。 **注:** NameValue データ・ハンドラーはデフォルトのデータ・ハンドラーに設定されているので、EndBODelimiter 値は <EndBO:*BOName*> に設定されています。Delimited データ・ハンドラーなどの別のデータ・ハンドラーを使用する場合は、対応する値を指定する必要があります。Delimited データ・ハンドラーの場合は、EOL ストリングが有効な EndBODelimiter の値です。
EndOfFileDelimiter	DataProcessingMode が Binary であり、かつ FTPDataStructure が Record である場合には、EndBODelimiter と EndOfFileDelimiter の両方が使用されます。このプロパティーは、レコード・ファイルの EOF (ファイル終わり) マーク文字として使用されているバイトを 16 進数で表したものに設定します。設定されていない場合には、デフォルト値 FF02 が使用されます。

表 8. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
EventDataHandler	データ・ハンドラーでイベント処理 (ビジネス・オブジェクト・ストリングがビジネス・オブジェクトに変換される) に使用される構成値を提供する属性を持つ、子メタオブジェクトを表します。出荷時のデフォルト値は MO_DataHandler_DefaultNameValueConfig です。
EventDir	イベント・ディレクトリーの絶対パスを指定します。ディレクトリーは既存のものである必要があります。異なるビジネス・オブジェクトに個別にメタオブジェクトを作成し、両方に同一の EventDir パスを指定する場合、それぞれのメタオブジェクトの EventExt 属性に固有な値を指定する必要があります。詳細については、66 ページの『イベント・ディレクトリーおよび拡張子の指定』を参照してください。コネクターを構成して、イベント処理用にリモート FTP ファイル・システムを使用するには、この属性に FTP URL を指定します。オプションで、この属性を使用して URL 内に以下の追加情報を指定することができます。 • FTP サーバーに接続して FTP 操作を実行するための特権を持つ、ユーザーの ID および パスワード。EventDir で指定しない場合は、FTPUserId および FTPPassword で指定する必要があります。 • FTP ポート。EventDir で指定しない場合、コネクターはデフォルトの FTP ポートを使用します。 • リモート・イベント・ディレクトリー。EventDir で指定しない場合、コネクターは FTP サーバーへの接続が確立しているディレクトリーからイベント・ファイルをポーリングします。 EventDir 属性で FTP 情報を指定するための構文は、以下のとおりです。 ftp://[*UserId*:*password*@]*FTPserver*[:*port*][*RemoteEventDirectory*] 詳細については、78 ページの『リモートのイベント処理』を参照してください。EventDir 属性でローカル・ファイル情報を指定するには、ファイルの絶対パスを使用します。あるいは、以下の形式の FILE URL を使用します。 [file://]*FullPathname* 出荷時のデフォルト値は以下のとおりです。 **Linux:** /tmp/JTextConn/Default/event **Windows:** C:¥temp¥JTextConn¥Default¥Event **i5/OS:** /QIBM/UserData/WBIServer44/<instance>/connectors/JText/event Windows 対応の BIDI です。BIDI が使用可能なすべてのプロパティーは、Windows BIDI 形式から、E メール・コネクター標準プロパティーである BiDi.Metadata 形式に変換されます。 **注:** 新しく導入されたコネクター固有 boolean プロパティー「NoPoll」により、オプションでポーリングをオフにできます。デフォルト値は false です。true に設定されている場合、アダプターは要求のみを処理し、ポーリングは処理しません。

表 8. MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)

属性名	説明
EventExt	イベント通知に使用されるファイルの拡張子を指定します。値が指定されない場合、JText コネクターはファイル拡張子なしでファイルをポーリングします。詳細については、74 ページの『複数のイベント・ファイルまたは複数のイベント・ディレクトリーの指定』を参照してください。 **注:** この属性にアスタリスク (*) を使用すると、拡張子がサポートされていなくても、単一イベント・ディレクトリー内のすべてのファイルをポーリングできます。出荷時のデフォルト値は in です。
FailArchiveExt	正常に処理されなかったビジネス・オブジェクトのアーカイブに使用するファイル拡張子を指定します。詳細については、67 ページの『イベント・アーカイブの指定』を参照してください。出荷時のデフォルト値は fail です。
FileSeqEnabled	各ビジネス・オブジェクトを個別のファイルに出力する、ファイル名の順序付けを指定します。ファイル名には固有のシーケンス番号が含まれます。詳細については、69 ページの『要求処理の指定』を参照してください。出荷時のデフォルト値は true です。
FixedBOSize	このメタオブジェクト・プロパティーが有効な値とともに指定されている場合、このプロパティーが EndBODelimiter プロパティーに優先し、ユーザーは従来の区切り文字ベースの BO 解析に代わる解析を行うことができます。

表 8. MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)

属性名	説明
FTPArchiveDir	FTP サーバーにあるアーカイブ・ディレクトリーの相対パスを指定します。ディレクトリーは既存のものである必要があります。この属性を使用したアーカイブの指定には、いくつかのオプションがあります。 • この属性に値を指定し、FTPRenameExt 属性に値を指定しない場合、コネクターはイベント・ファイル名にタイム・スタンプを追加し、それをこの属性で指定した FTP サーバーのアーカイブ・ディレクトリーに移動します。 • この属性と FTPRenameExt 属性の両方に値を指定する場合、コネクターはタイム・スタンプと FTPRenameExt で指定した値を持つ処理済みのイベント・ファイル名を変更し、それをこの属性で指定した FTP サーバーのアーカイブ・ディレクトリーに移動します。 • この属性にも FTPRenameExt 属性にも値を指定しない場合、コネクターは処理済みのイベント・ファイルをアーカイブせずに削除します。 • この属性に値を指定しないで FTPRenameExt 属性に値を指定した場合、コネクターは、処理したイベント・ファイルの名前を変更し (タイム・スタンプと FTPRenameExt に指定されている値を追加します)、さらにそのファイルを EventExt 属性に指定されているディレクトリーに移動します。 • この属性に / (スラッシュ) を指定し、FTPRenameExt 属性に値を指定しない場合、コネクターは処理済みのイベント・ファイルを FTP サーバーのルート・ディレクトリーに移動します。 • この属性に / (スラッシュ) を指定し、FTPRenameExt 属性に値を指定する場合、コネクターは FTPRenameExt で指定した拡張子を持つ処理済みのイベント・ファイル名を変更し、それを FTP サーバーのルート・ディレクトリーに移動します。 詳細については、67 ページの『イベント・アーカイブの指定』を参照してください。この属性には、出荷時のデフォルト値はありません。 Windows 対応の BIDI です。BIDI が使用可能なすべてのプロパティーは、Windows BIDI 形式から、E メール・コネクター標準プロパティーである BiDi.Metadata 形式に変換されます。
FTPDataStructure	この属性は String 型です。この属性を使用すると、リモート・サイトに対するファイルの読み取りと書き込みを、FTP データ構造 (File または Record) を指定して実行できます。何も指定されていない場合、JText は File をデフォルト値として使用します。

表 8. MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)

属性名	説明
FTPEventFileMask	組み込みワイルドカード文字を使用して、イベント処理用のリモート FTP ファイルのマスクまたはプレフィックスを指定します。この属性に値を指定するのは、Windows、Linux、または i5/OS システムに適用されるのと同じ命名標準に準拠しないメインフレーム上で、ファイル・マスクを識別する場合だけです。ファイル名にワイルドカード文字を使用すると、イベント処理に複数のファイルを指定できるようになります。例えば、ACT.Z1UC.INPT* というフォーマットを使用すると複数のイベント・ファイルを指定できます。詳細については、82 ページの『メインフレーム上のファイルの識別: オプション構成』を参照してください。出荷時のデフォルト値はありません。 ポーリングする場合は、かなり具体的なマスクを指定してください。例えば、USER.JTEXT.TEST001.EVENT、USER.JTEXT.TEST002.EVENT、USER.JTEXT.TEST003.EVENT、USER.JTEXT.TEST004.EVENT のすべてのイベント・ファイルをポーリングし、FTPArchiveDir が / と設定されるかブランクのまま場合、および FTPRenameExt が RENAME と設定されている場合は、これらのファイルがUSER.JTEXT.TEST001.RENAME、USER.JTEXT.TEST002.RENAME、 USER.JTEXT.TEST003.RENAME のようにアーカイブされます。したがって、FTPEventFileMask=USR.JTEXT.TEST*.* と指定すると、すべてのイベントが最初のポーリングで選択されます。アーカイブ・ファイルも同じファイル・マスクに適合しているので、次のポーリングですべてのアーカイブ・ファイルが選択されます。この状態を避けるには、かなり具体的なマスクを指定する必要があります。例えば、USR.JTEXT.TEST*.EVENT のように入力し、ポーリング時に USR.JTEXT.TEST*.RENAME が選択されないようにします。 **注:** イベント・ファイルとアーカイブ・ファイルの両方に適用可能なマスクを指定しないでください。
FTPFileListingFormat	Windows 対応の BIDI です。BIDI が使用可能なすべてのプロパティーは、Windows BIDI 形式から、E メール・コネクター標準プロパティーである BiDi.Metadata 形式に変換されます。 JText コネクターがファイルを読み取るときに想定するファイル情報のフォーマットを指定します。これにより、コネクターは、異なるロケールの (例えば日時情報がファイル・フォーマット情報と異なる順序で格納されている) ファイルを読み取ることができます。所定のロケールのフォーマットを使用するようにコネクターを構成するには、ファイル属性が出現する順序を表す文字をセミコロンで区切って列挙します。以下に、指定可能な文字とファイル属性の対応関係を示します。 P アクセス権 L リンク U ユーザー G グループ S サイズ D 日付 M 月 T 時刻 N 名前 例えば、この属性の値は P:L:U:G:S:D:M:T:N のようになります。
FTPGetQuantity	各リモート・ポーリングにより、リモート FTP URL から検索されるファイル数を決定します。

表 8. MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)

属性名	説明
FTPKeepConnectionOpen	この属性の Default Value プロパティーを値 true に設定すると、JText コネクターは FTP サイトとの接続を保持します。この属性を値 true に設定すると、コネクターは、コネクターが終了する場合または (構成されたタイムアウトなどを理由として) FTP サーバー自体が接続をクローズした場合にのみ接続をクローズします。コネクターは、FTP サーバーがタイムアウトを理由として接続をクローズする場合に対処するため、リモート操作を実行するたびに接続がアクティブになっていることを検査します。接続がクローズされている場合は、コネクターは接続を再確立します。この属性の Default Value プロパティーを値 false に設定すると、JText コネクターは、操作を実行するたびに FTP サーバーとの接続をオープンし、操作が完了すると接続をクローズします。コネクターが接続を保持するように構成すると、FTP サイトでの要求処理時のパフォーマンスが向上します。
FTPLocalEventDir	コネクターが FTP サイトからダウンロードしたイベント・ファイルを格納する、ローカル・システム・ディレクトリーを指定します。コネクターが FTP を使用してイベントを処理できるようにするために、この属性に値を指定する必要があります。詳細については、79 ページの『ローカル・ディレクトリーの指定』を参照してください。出荷時のデフォルト値はありません。
	Windows 対応の BIDI です。BIDI が使用可能なすべてのプロパティーは、Windows BIDI 形式から、E メール・コネクター標準プロパティーである BiDi.Metadata 形式に変換されます。
FTPOSPlatform	この属性を使用するのは、コネクターを構成して、リモート FTP サーバーが MVS プラットフォームであるリモート FTP ファイル・システムを使用できるようにする場合のみです。この場合、属性の値に MVS を指定します。大文字小文字は区別されません。詳細については、78 ページの『リモート FTP ファイル・システムの指定』を参照してください。出荷時のデフォルト値はありません。
FTPPassword	FTP サーバーに接続して FTP 操作を実行する特権を持つユーザーのパスワードを指定します。EventDir 属性または OutputDir 属性で指定した URL にパスワードが含まれている場合、この属性に値を指定する必要はありません。詳細については、79 ページの『FTP URL およびログイン情報の指定』を参照してください。この属性には、出荷時のデフォルト値はありません。
FTPPollTerminateIfServerDown	イベントがあるかどうか FTP サイトをポーリングするように構成した場合に FTP サイトが使用不能なときのコネクターの振る舞いを指定します。 FTPPollTerminateIfServerDown 属性の Default Value プロパティーを値 true に設定した場合、コネクターが呼び出しをポーリングしたときに FTP サイトが使用不能な場合は、コネクターは終了します。 FTPPollTerminateIfServerDown 属性の Default Value プロパティーを値 false に設定した場合、コネクターが呼び出しをポーリングしたときに FTP サイトが使用不能な場合でもコネクターは終了しません。 出荷時のデフォルト値はありません。

表 8. *MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)*

属性名	説明
FTPRenameExt	コネクターがリモート FTP ファイルのポーリング後にそのファイル名を変更するときに使用する、ファイル拡張子またはサフィックスを指定します。ファイル名を変更することによって、次のポーリング・サイクルに同じファイルをポーリングすることがなくなります。あるいは、コネクターを構成して処理済みのイベント・ファイル名を変更し、それをアーカイブ・ディレクトリーに移動することもできます。詳細については、FailArchiveExt 属性を参照してください。詳細については、82 ページの『メインフレーム上のファイルの識別: オプション構成』を参照してください。出荷時のデフォルト値はありません。
FTPRequestTerminateIfServerDown	FTP サイトで要求処理および通信を行うようにコネクターを構成した場合に、FTP サイトが使用不能なときのコネクターの振る舞いを指定します。 FTPRequestTerminateIfServerDown 属性の Default Value プロパティーを値 true に設定した場合、コネクターが要求を処理するときに FTP サイトが使用不能な場合は、コネクターは終了します。 FTPRequestTerminateIfServerDown 属性の Default Value プロパティーを値 false に設定した場合、コネクターが要求を処理するときに FTP サイトが使用不能な場合でもコネクターは終了しません。 出荷時のデフォルト値はありません。
FTPTransferType	この JText メタオブジェクト・プロパティーは、イベント処理時と要求処理時の両方で使用されます。このプロパティーに指定可能な値は、Binary と ASCII です。このプロパティーは、JText がリモートで FTP サーバーに対してファイルの配置や取得を実行するときに使用する転送タイプを示します。このプロパティーが存在しない場合、アダプターの動作のタイプは Binary になります。
FTPUserId	FTP サーバーに接続して FTP 操作を実行する特権を持つユーザー名を指定します。EventDir 属性または OutputDir 属性で指定した URL にユーザー ID が含まれている場合、この属性に値を指定する必要はありません。 EventDir 属性 (イベント処理時) または OutputDir 属性 (要求処理時) に FTP URL が見つからない場合、コネクターはこの属性を無視します。詳細については、79 ページの『FTP URL およびログイン情報の指定』を参照してください。この属性には、出荷時のデフォルト値はありません。
IncludeEndBODelimiter	EndBODelimiter メタオブジェクト属性に指定された値を、JText コネクターがファイルに書き込むストリングに含めるかどうかを指定します。この属性の Default Value プロパティーを true に設定した場合は、コネクターは、ファイルへの書き込み時に EndBODelimiter 属性に指定された値を含めます。この属性の Default Value プロパティーを false に設定した場合は、コネクターは、ファイルへの書き込み時に EndBODelimiter 属性で指定された値を含めません。

表 8. MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)

属性名	説明
LargeObject	JText アダプター (true に設定されている場合) の大規模オブジェクト最適化機能をオンにするために使用されるフラグです。このフラグを true に設定すると、アーカイバーの振る舞いが次のように変更されます。 1. 1 つのイベント・ファイルに複数のビジネス・オブジェクトがある場合は、そのイベント・ファイルのすべてのビジネス・オブジェクトが処理された後にのみ、アーカイブが実行されます。 2. そのイベント・ファイルのすべてのビジネス・オブジェクトの処理が失敗またはアンサブスクライブされた状況になった場合は、元のファイルはアーカイブされません。 また、アーカイブ状況を追跡するため、余分のログ・ファイルが内部で作成されます。
MVSSiteCommand	MVS FTP サイト・コマンドを発行するために使用されます。このサイト・コマンドは、SITE または QUOTE キーワードを使用せずに指定してください。サイト・コマンド値には、例えば LRECL=<value> BLKSIZE=<value> などがあります。ここで、<value> は渡されるサイト・コマンドの引数を表します。
ObjectEventID	メタオブジェクト内のコネクターは使用しないが、統合ブローカーで必要なプレースホルダー。この属性は、メタオブジェクト内の最後の属性である必要があります。出荷時のデフォルト値はありません。
OriginalArchiveExt	オリジナルのイベント・ファイルのアーカイブに使用するファイル拡張子を指定します。これにより、ビジネス・オブジェクトが処理に失敗したりアンサブスクライブされた場合の参照用に、イベント・ファイルが保存されます。詳細については、67 ページの『イベント・アーカイブの指定』を参照してください。出荷時のデフォルト値は orig です。
OutputDataHandler	データ・ハンドラーでサービス呼び出し要求 (ビジネス・オブジェクトがビジネス・オブジェクト・ストリングに変換される) に使用される構成値を提供する属性を持つ、子メタオブジェクトを表します。出荷時のデフォルト値は MO_DataHandler_DefaultNameValueConfig です。

表 8. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
OutputDir	出力ディレクトリーの絶対パスを指定します。ディレクトリーは既存のものである必要があります。コネクターを構成して、要求処理用にリモート FTP ファイル・システムを使用するには、この属性に FTP URL を指定します。オプションで、この属性を使用して URL 内に以下の追加情報を指定することができます。 • FTP サーバーに接続して FTP 操作を実行するための特権を持つ、ユーザーの ユーザー ID およびパスワード。EventDir で指定しない場合は、FTPUserId および FTPPassword で指定する必要があります。 • FTP ポート。OutputDir で指定しない場合、コネクターはデフォルトの FTP ポートを使用します。 • リモート出力ディレクトリー。OutputDir で指定しない場合、コネクターは要求ファイルをデフォルトの接続ディレクトリー (接続が確立している FTP サーバーのディレクトリー) にロードします。 OutputDir 属性で FTP 情報を指定する構文は、ftp://[*UserId*:*password*@]*FTPserver*[:*port*] です。詳細については、83 ページの『リモートの要求処理』を参照してください。OutputDir 属性でローカル・ファイル情報を指定するには、ファイルの絶対パスを使用します。あるいは、以下の形式の FILE URL を使用します。 [file://]*FullPathname* 出荷時のデフォルト値は以下のとおりです。 **Linux:** /tmp/JTextConn/Default/out **Windows:**c:¥temp¥JTextConn¥Default¥Out **i5/OS:** /QIBM/UserData/WBIServer44/<instance>/connectors/JText/out Windows 対応の BIDI です。BIDI が使用可能なすべてのプロパティーは、Windows BIDI 形式から、E メール・コネクター標準プロパティーである BiDi.Metadata 形式に変換されます。
OutputExt	要求処理に使用されるファイルの拡張子を指定します。出荷時のデフォルト値は out です。 **注:** OutputFileName 属性に拡張子が含まれず、OutputExt 属性に拡張子が含まれている場合、生成される出力ファイルにはファイル名と拡張子が付きます。どちらの属性にも拡張子が含まれない場合、生成される出力ファイルには拡張子が付きません。
OutputFileName	要求処理時にコネクターが着信ビジネス・オブジェクトを書き込む、出力ファイルの名前とパスを指定します。OutputDir 属性に有効な出力ディレクトリーが含まれている場合、出力ファイルは指定されたディレクトリーに生成されます。詳細については、64 ページの『出力ファイル名の指定』を参照してください。 **注:** OutputFileName 属性および OutputExt 属性に拡張子が含まれない場合、生成される出力ファイルには拡張子が付きません。 出荷時のデフォルト値は Native です。 対応する BIDI です。ターゲット・プラットフォームで使用される BIDI 形式が Windows 2003 形式と異なる場合、メタデータ属性の値が変換されます。

表 8. *MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)*

属性名	説明
PartialArchiveExt	正常に処理されたビジネス・オブジェクトのアーカイブに使用するファイル拡張子を指定します (イベント・ファイルに複数のビジネス・オブジェクトが含まれている場合、一部は正常に処理されません)。詳細については、67 ページの『イベント・アーカイブの指定』を参照してください。出荷時のデフォルト値は partial です。
StagingDir	コネクターが、OutputDir 属性で指定されたディレクトリーにファイルを移動する前にファイルを書き込むディレクトリーを指定します。これは、JText コネクターがファイルを出力するディレクトリーを他のソフトウェア・プロセスがモニターおよび操作する環境 (コネクターによって作成されたファイルを FTP プロセスが検出し、別の場所に移動するなど) を扱うために設計されています。このような場合は、ファイルの書き込みが完了する前に外部プロセスがファイルを移動してしまう危険性があります。StagingDir 属性でステージング・ディレクトリーを指定し、コネクターがファイルを完全にステージング・ディレクトリーに書き込み、その書き込みが完了してからファイルを出力ディレクトリーに移動することにより、外部プロセスが不完全なファイルを抽出してしまう危険性を回避できます。 ステージング・ディレクトリーと出力ディレクトリーは同じファイル・システムまたはドライブに置き、オペレーティング・システムごとのファイル移動方式の差異を吸収してください。StagingDir は、リモート・ディレクトリーにすることができます。出荷時のデフォルト値はありません。 Windows 対応の BIDI です。BIDI が使用可能なすべてのプロパティーは、Windows BIDI 形式から、E メール・コネクター標準プロパティーである BiDi.Metadata 形式に変換されます。
SuccessArchiveExt	すべて正常に処理されたビジネス・オブジェクトのアーカイブに使用するファイル拡張子を指定します。詳細については、67 ページの『イベント・アーカイブの指定』を参照してください。出荷時のデフォルト値は success です。
UnsubscribedArchiveExt	すべてアンサブスクライブされたビジネス・オブジェクトのアーカイブに使用するファイル拡張子を指定します。詳細については、67 ページの『イベント・アーカイブの指定』を参照してください。出荷時のデフォルト値は unsub です。

注: 属性 FTPTransferType、FTPDataStructure、DataProcessingMode、EndOfFileDelimiter、および FixedBOSize は、出荷時の JText メタオブジェクトには含まれていません。これらの属性を使用するには、明示的にメタオブジェクトに追加する必要があります。また、これらの属性のデフォルト値は、必ず設定しなければなりません。

出力ファイル名の指定

出力ファイル名を指定する方法は 3 つあります。

- OutputFileName 属性を使用する

  この属性を使用するのは、コネクターを構成して、同一タイプのビジネス・オブジェクトを固有のシーケンス番号付きの別々のファイルに書き込むか、複数のビジネス・オブジェクトを指定された名前の単一ファイルに追加する場合です。
- 動的子メタオブジェクトを使用する

動的子メタオブジェクトを使用するのは、ビジネス・オブジェクトのタイプごとに出力ファイル名を動的に生成するか、コネクターが生成した出力ファイル名を戻す場合です。詳細については、4 ページの『動的子メタオブジェクトの使用』を参照してください。

OutputFileName 属性を使用して出力ファイル名を使用する方法はいくつかあります。

- OutputFileName 属性がストリング Native に設定され、FileSeqEnabled 属性が true に設定された場合、コネクターはビジネス・オブジェクト・ストリングを固有ファイルに送信します。このとき、ファイル名は着信ビジネス・オブジェクト名から派生し、拡張子は OutputExt 属性から派生し、パスは OutputDir 属性から派生します。この場合、デフォルトでは、コネクターは同一タイプの各ビジネス・オブジェクトを固有のシーケンス番号付きの別々のファイルに書き込みます。コネクターが同一タイプのビジネス・オブジェクトを受信したら常に出力ファイルを上書きするようにするには、FileSeqEnabled 属性を false に設定します。
- OutputFileName 属性がストリング Native 以外に設定され、FileSeqEnabled 属性が true に設定された場合、コネクターは出力ファイルの値を以下のいずれかの方法で処理します。
  - OutputFileName 属性に絶対パスが含まれる場合 (出力ファイルのファイル名と拡張子が含まれる。例えば、OutputFileName= C:¥temp¥Out¥test.out)、コネクターはこの属性のみを使用して出力ファイルを生成します。この場合、デフォルトでは、コネクターは同一タイプの各ビジネス・オブジェクトを、指定された名前と固有のシーケンス番号付きの別々のファイルに書き込みます。
  - OutputFileName 属性に絶対パスとファイル名が含まれるが拡張子が含まれず、OutputExt 属性に値が含まれる場合 (例えば、OutputFileName= C:¥temp¥Out¥test および OutputExt=out)、コネクターは両方の属性の値を使用して出力ファイルを生成します。この場合、コネクターは C:¥temp¥Out¥test_1.out という名前のファイルを生成します。
  - OutputFileName 属性に絶対パスとファイル名が含まれるが拡張子が含まれず、OutputExt 属性に値が含まれない場合、コネクターは拡張子のない出力ファイルを生成します。この場合、コネクターは C:¥temp¥Out¥test_1 という名前のファイルを生成します。
  - OutputFileName 属性にパスと拡張子が含まれずファイル名のみが含まれ、OutputDir 属性に値が含まれる場合、コネクターは OutputDir に指定したディレクトリーに出力ファイルを生成します。OutputExt に値が含まれる場合、コネクターはその値も使用します。値が含まれない場合、コネクターは拡張子なしでファイル名を作成します。

注: コネクターが複数のタイプのビジネス・オブジェクトを処理していて、OutputFileName にストリング Native 以外が設定されている場合、各ビジネス・オブジェクトに自身のトップレベル・メタオブジェクトを持たせ、固有の出力ファイル名を指定する必要があります。例えば、Customer ビジネス・オブジェクトが使用するメタオブジェクトは MO_JTextConnector_Customer、Item ビジネス・オブジェクトが使用するメタオブジェクトは MO_JTextConnector_Item になります。これらの各メタオブジェクトの OutputFileName 属性には、固有値を設定します。

- コネクターが複数のビジネス・オブジェクトを指定された名前の付いた単一ファイルに追加するようにするには、OutputFileName 属性に値を指定し、FileSeqEnabled 属性を false に設定します。
- コネクターが同一タイプのビジネス・オブジェクトを受信したら常に出力ファイルを上書きするようにするには、動的子メタオブジェクトを使用します。InFileName 属性に動的子メタオブジェクトの絶対パスとファイル名を指定し、FileWriteMode 属性を「o」に設定します。動的子メタオブジェクトについての詳細は、4 ページの『動的子メタオブジェクトの使用』を参照してください。

Native は予約語です。

詳細については、69 ページの『要求処理の指定』を参照してください。

共通の構成タスク

このセクションでは、共通の構成タスクについて説明します。

- 66 ページの『イベント通知の指定』
- 67 ページの『イベント・アーカイブの指定』
- 69 ページの『要求処理の指定』
- 74 ページの『複数のイベント・ファイルまたは複数のイベント・ディレクトリーの指定』
- 74 ページの『特定のビジネス・オブジェクトのポーリング』
- 78 ページの『リモート FTP ファイル・システムの指定』
- 85 ページの『セキュア FTP の構成』
- 86 ページの『データ・ハンドラーの指定』
- 87 ページの『特定のビジネス・オブジェクトの JText メタオブジェクトの作成』
- 87 ページの『同一ファイルから異なるタイプの複数のビジネス・オブジェクトを読み取る』
- 88 ページの『ObjectEventID 属性値の指定』
- 88 ページの『JText コネクターの 2 番目のインスタンスのセットアップ』
- 89 ページの『JText コネクターのパフォーマンス調整』
- 90 ページの『テスト用サンプル・ファイルの生成』
- 91 ページの『テスト用サンプル・ビジネス・オブジェクトの生成』

イベント通知の指定

このセクションの内容は、以下のとおりです。

- 『イベント・ディレクトリーおよび拡張子の指定』
- 67 ページの『ポーリングの振る舞いの構成』

イベント・ディレクトリーおよび拡張子の指定

複数のタイプのビジネス・オブジェクトをコネクターに送信して処理するときに、各ビジネス・オブジェクト・タイプにトップレベル・メタオブジェクトが含まれる

場合、EventDir および EventExt 属性に指定する値の組み合わせは、各ビジネス・オブジェクトのディレクトリー/拡張子のペアで固有でなければなりません。

つまり、2 つのビジネス・オブジェクト・タイプに同じイベント・ディレクトリーを指定する場合、これらのビジネス・オブジェクトに異なるイベント拡張子を指定する必要があります。また、2 つのビジネス・オブジェクト・タイプに同じ拡張子を指定する場合は、これらのビジネス・オブジェクトに異なるイベント・ディレクトリーを指定する必要があります。

例えば、MO_JTextConnector_Customer および MO_JTextConnector_Item メタオブジェクトを作成して、それぞれ Customer および Item ビジネス・オブジェクトに構成値を提供するとします。コネクターが同一ディレクトリー内で両方のビジネス・オブジェクトの入力ファイルを検索するように構成する場合 (EventDir 属性に同一パスを指定します)、EventExt 属性に異なる値を指定することによって入力ファイルを一意的に識別する必要があります。

したがって、EventDir 属性が Customers と Items 両方の C:¥temp¥event を評価する場合、2 つのビジネス・オブジェクトの EventExt 属性値が異なる必要があります (例えば、Customer 入力ファイルが in、Items 入力ファイルが inp)。

注: 新しく導入されたコネクター固有 boolean プロパティー「NoPoll」により、オプションでポーリングをオフにできます。デフォルト値は false です。true に設定されている場合、アダプターは要求のみを処理し、ポーリングは処理しません。

ポーリングの振る舞いの構成

ポーリングの振る舞いを構成するには、以下のステップを実行します。

1. 以下の MO_JTextConnector_Default メタオブジェクトの属性を構成します。
   - EventDir— イベント通知を起動するファイルが格納されている既存のディレクトリーの絶対パスを指定します。
   - EventExt— コネクターは、出荷時のデフォルト拡張子が付いたファイルを検索します。この属性を使用して異なる拡張子を指定する場合、コネクターは指定された拡張子を検索します。この属性を空にしておくと、コネクターは拡張子のないファイルをポーリングします。
   - EventDataHandler— イベント通知時にデータ変換に使用するデータ・ハンドラーを指定します。
2. Connector Configurator Express を使用して次のコネクター・プロパティーを構成します。
   - PollFrequency— 間隔頻度を指定します。
   - PollQuantity— ポーリング間隔ごとのイベント数を指定します。
   - PollEndTime— イベントのポーリングを完了する時間を指定します。
   - PollStartTime— イベントのポーリングを開始する時間を指定します。
3. イベント・ディレクトリーに読み取りアクセス権を設定します。

イベント・アーカイブの指定

イベント・ファイル内で正常に処理されたビジネス・オブジェクトがそのすべてであるか一部かによって、JText コネクターは、正常に処理されたビジネス・オブジェ

クトのアーカイブ・ファイルを作成するときに異なる拡張子を使用します。また、コネクターは、処理に失敗したりアンサブスクライブされたビジネス・オブジェクトをさまざまな名前のアーカイブ・ファイルに書き込みます。

このセクションの内容は、以下のとおりです。

- 『ローカル・アーカイブ・ファイル名』
- 69 ページの『ローカル・アーカイブの構成』

ローカル・アーカイブ・ファイル名

アーカイブ拡張子属性の出荷時のデフォルト値を保存する場合、コネクターは以下に示す名前のアーカイブ・ファイルを作成します。

- イベント・ファイルが単一のビジネス・オブジェクトを持つ

  JText コネクターが単一のビジネス・オブジェクトを含むイベント・ファイルを処理すると、アーカイブ・ディレクトリーに以下のいずれかのファイルを作成します。

  - *filename_timestamp*.success。正常に処理されたビジネス・オブジェクトをアーカイブします。
  - *filename_timestamp*.fail。正常に処理されなかったビジネス・オブジェクトをアーカイブします。
  - filename_timestamp.unsub。サブスクライブされていないビジネス・オブジェクトをアーカイブします。

  ビジネス・オブジェクトが処理に失敗したりアンサブスクライブされたりした場合、コネクターは *filename_timestamp*.orig というファイルも作成します。ここには、コネクターが最初にイベント・ファイルを受け取ったときにそれを保存します。
- イベント・ファイルが複数のビジネス・オブジェクトを持ち、そのすべてが正常に処理されている

  JText コネクターが複数のビジネス・オブジェクトを含むイベント・ファイルを正常に処理すると、アーカイブ・ディレクトリーに *filename_timestamp*.success を作成します。
- イベント・ファイルが複数のビジネス・オブジェクトを持ち、その一部がアンサブスクライブされているか処理に失敗している

  JText コネクターが複数のビジネス・オブジェクトを含むイベント・ファイルを処理すると、アーカイブ・ディレクトリーに以下のすべてのファイルを作成する場合があります。

  - filename_timestamp.partial。正常に処理されたすべてのビジネス・オブジェクトをアーカイブします。
  - filename_timestamp.fail。正常に処理されなかったすべてのビジネス・オブジェクトをアーカイブします。
  - filename_timestamp.unsub。コネクターからサブスクライブされないすべてのビジネス・オブジェクトをアーカイブします。
  - *filename_timestamp*.orig。コネクターが最初にイベント・ファイルを受け取ったときにそれを保存します。

例えば、LegacyApp.in ファイルに以下の 4 つのビジネス・オブジェクトが含まれるとします。

– Contract。正常に処理されています。
– Customer。フォーマット設定に失敗しています。
– Order。正常に処理されています。
– Item。コネクターからサブスクライブされていません。

この場合、コネクターはアーカイブ・ディレクトリーに以下のファイルを作成します。

– LegacyApp_*timestamp*.partial。Contract および Order を含みます。
– LegacyApp_*timestamp*.fail。Customer を含みます。
– LegacyApp_*timestamp*.unsub。Item を含みます。
– LegacyApp_*timestamp*.orig。Contract、Customer、Order、および Item を含みます。

ローカル・アーカイブの構成

コネクターをアーカイブのために構成するには、以下のステップを実行します。

1. 以下の MO_JTextConnector_Default メタオブジェクトの属性を構成します。
   - ArchiveDir—コネクターが処理後のイベントを (処理状況を示すファイル拡張子とともに) 置く、既存のローカル・ディレクトリーまたは FTP サーバー・ディレクトリーの絶対パスを指定します。
   - SuccessArchiveExt— (すべてのビジネス・オブジェクトが正常に処理された場合に) 正常に処理されたビジネス・オブジェクトが含まれるファイルの拡張子を指定します。
   - PartialArchiveExt— (イベント・ファイル内の一部のビジネス・オブジェクトが正常に処理されない場合に) 正常に処理されたすべてのビジネス・オブジェクトが含まれるファイルの拡張子を指定します。
   - UnsubscribedArchiveExt— コネクターからサブスクライブされないビジネス・オブジェクトが含まれるファイルの拡張子を指定します。
   - OriginalArchiveExt— イベント・ファイル内のすべてのビジネス・オブジェクトが保存されるファイルの拡張子を指定します。
   - FailArchiveExt— 処理に失敗したビジネス・オブジェクトが含まれるファイルの拡張子を指定します。
2. Connector Configurator Express を使用して ArchivingEnabled コネクター・プロパティーを構成します。
3. アーカイブ・ディレクトリーに書き込みアクセス権を設定します。

要求処理の指定

JText コネクターが、(各ビジネス・オブジェクト・インスタンス内で) 動的に名前を付けられるファイル、または (メタオブジェクトによって) 静的に名前を付けられるファイルにビジネス・オブジェクトを書き込むように設定することができます。また、コネクターが静的に生成するファイル名を戻すように設定することもできます。この機能は、固有のシーケンス番号付きのファイル名を取得するのに役立ちます。このセクションに含まれるサブセクションは、以下のとおりです。

- 70 ページの『動的なファイルの命名』
- 70 ページの『静的なファイルの命名』
- 71 ページの『ファイル名の戻り』
- 72 ページの『ローカル処理とリモート処理の違い』
- 72 ページの『出力ファイルの構成』

動的なファイルの命名

コネクターが各ビジネス・オブジェクト・タイプに動的に出力ファイル名を生成するように設定するには、動的子メタオブジェクトを作成します。子メタオブジェクトを使用して、以下のことを実行します。

- 出力ファイル名を指定するか、生成されたファイル名を受け取ります。
- 出力ファイルに追加するか、または上書きするかを指定します。

重要: 動的子メタオブジェクトを作成してコネクターが出力ファイル名を生成または戻すように設定するほかに、InterChange Server Express を統合ブローカーとして使用している場合は、マップまたはコラボレーション・ロジックを変更して、動的子メタオブジェクトの `InFileName` 属性に各ビジネス・オブジェクトのパスとファイル名、そして必要な場合は固有のシーケンス番号を挿入する必要があります。

詳細については、 4 ページの『動的子メタオブジェクトの使用』を参照してください。

コネクターがメタオブジェクトを処理する方法の詳細については、 12 ページの『要求処理』を参照してください。

コネクターを構成して動的に生成された出力ファイル名を使用する方法の詳細については、72 ページの『出力ファイルの構成』を参照してください。

静的なファイルの命名

メタオブジェクトを使用して出力ファイル名を指定する場合、変更内容を有効にするにはコネクターを再始動する必要があります。コネクターが任意のタイプのすべてのビジネス・オブジェクトを単一ファイルに追加するか、またはビジネス・オブジェクトごとに別々のファイルを作成するように指定できます。

出荷時のデフォルト構成を使用する場合、コネクターは処理するビジネス・オブジェクトごとに出力ファイルを作成します。コネクターは受信したビジネス・オブジェクトの出力ファイルに名前を付け、シーケンス番号を追加して名前を固有のものにします。また、`.out` という拡張子を付けます。例えば、Customer および Item というビジネス・オブジェクトを受信した場合、コネクターはこれらのデータをそれぞれ `Customer_1.out` および `Item_1.out` という出力ファイルに書き込みます。出力ファイル名の取得の詳細については、71 ページの『ファイル名の戻り』を参照してください。シーケンス番号を保管するファイルの詳細については、 37 ページの『OutputLog』を参照してください。

メタオブジェクトを使用して出力ファイル名を構成するには、以下の手順を実行します。

1. 以下の `MO_JTextConnector_Default` メタオブジェクトの属性を構成します。

- OutputDir— コネクターが要求の処理時にファイルを書き込む、既存のディレクトリーの絶対パスを指定します。詳細については、72 ページの『出力ファイルの構成』を参照してください。
- OutputExt— 出荷時のデフォルト構成を変更する場合、この属性を使用して拡張子を指定します。これにより、コネクターは out 拡張子の付いたファイルを作成するようになります。
- FileSeqEnabled— 常に true に設定すると、コネクターはファイルごとに固有のシーケンス番号が付いた 1 つのビジネス・オブジェクトを出力します。false に設定すると、コネクターは任意のタイプのすべてのビジネス・オブジェクトを単一ファイルに出力 (上書きまたは追加) します。上書きまたは追加動作の構成の詳細については、 72 ページの表 9 を参照してください。
- OutputFileName—コネクターが、ビジネス・オブジェクトをファイル内のデータを上書きせずに単一の出力ファイルに追加するか、ビジネス・オブジェクトごとに固有ファイルを生成するように設定するには、出力ファイルの絶対パスおよびファイル名を指定します。

  コネクターが同一タイプのビジネス・オブジェクトを受信したら常に出力ファイルを上書きするように設定するには、OutputFileName に値を指定しないでください。

  どちらの場合も、FileSeqEnabled を false に設定します。

  上書きまたは追加動作の構成の詳細については、 72 ページの表 9 を参照してください。

2. 出力ディレクトリーに書き込みアクセス権を設定します。

注: コネクターが、さまざまなビジネス・オブジェクトにさまざまなデータ・フォーマットやファイル命名規則を使用する場合、特定のビジネス・オブジェクトにメタオブジェクトを作成する必要があります。

ファイル名の戻り

コネクターが生成したファイル名を戻すように設定するには、以下のことを実行します。

- メタオブジェクトを使用してパスとファイル名を指定して、コネクターが出力ファイルごとに固有のシーケンス番号を生成するように設定します。詳細については、70 ページの『静的なファイルの命名』を参照してください。
- 動的子メタオブジェクトを使用して、コネクターが生成したファイル名を戻すように設定します。 4 ページの『動的子メタオブジェクトの使用』にあるステップを実行しますが、InFileName 属性の値は指定しないでください。コネクターが受け取ったビジネス・オブジェクトの動的子メタオブジェクトが OutFileName=CxIgnore を指定している場合、コネクターはトップレベル・メタオブジェクトの構成に基づいてファイル名を作成し、InFileName 属性の値として絶対パスとファイル名を戻します。

注: コネクターは、FTP サーバーを介してファイルを処理する場合でも、InFileName 属性にローカル・パスのみを設定します。

重要: 動的子メタオブジェクトを作成してコネクターが出力ファイル名を生成または戻すように設定するほかに、InterChange Server Express を統合ブローカー

として使用している場合は、マップまたはコラボレーション・ロジックを変更して、動的子メタオブジェクトの InFileName 属性に各ビジネス・オブジェクトのパスとファイル名、そして必要な場合は固有のシーケンス番号を挿入する必要があります。

ローカル処理とリモート処理の違い

コネクターは、リモートでファイルを処理する方法は、ローカルでの処理とほぼ同じです。ただし、いくつかの違いがあります。

- イベントを処理して動的にファイル名を生成する場合、コネクターは動的子メタオブジェクトの InFileName 属性にローカル・パス名のみを設定し、リモート・パスを設定しません。
- 要求処理時に、コネクターが動的ファイル命名のために構成されておらず、FileSeqEnabled が false に設定され、出力ファイルがすでに存在する場合:
  - ローカルに処理する場合、コネクターは既存ファイルを上書きします。
  - リモートに処理する場合、コネクターは例外をスローします。
- ローカル・イベント処理に標準アーカイブ拡張子属性を構成するほかに、コネクターを使用して FTP サーバーを介してリモートにファイルを処理する場合、FTPArchiveDir および FTPRenameExt 属性も構成することができます。この属性によって、処理が正常に行われるごとに、リモートにアーカイブされたファイルの名前変更および移動ができるようになります。

詳細については、 80 ページの『リモート・アーカイブの指定』を参照してください。

出力ファイルの構成

表 9 は、出力ファイルで使用できる構成オプションを示しています。

表 9. 出力ファイルの指定

要求される出力条件	属性/プロパティーに必要な構成	属性/プロパティーの値
任意のタイプのビジネス・オブジェクトがファイルに追加され、その絶対パスとファイル名が実行時にビジネス・オブジェクトの属性から指定される。	動的子メタオブジェクトを使用する AppSpecificInfo (ビジネス・オブジェクト・レベル) 動的子メタオブジェクトの場合: OutFileName FileWriteMode	 cw_mo_JTextConfig = *DynChildMOName* *ユーザー指定のパス名およびファイル名* a または append
任意のタイプのビジネス・オブジェクトが出力ファイルを上書きし、その絶対パスとファイル名が実行時にビジネス・オブジェクトの属性から指定される。	動的子メタオブジェクトを使用する AppSpecificInfo (ビジネス・オブジェクト・レベル) 動的子メタオブジェクトの場合: OutFileName FileWriteMode	 cw_mo_JTextConfig = *DynChildMOName* *ユーザー指定のパス名およびファイル名* o または overwrite
任意のタイプのビジネス・オブジェクトが自身の固有ファイルに書き込まれ、そのファイル名がビジネス・オブジェクトの名前と生成された固有のシーケンス番号から指定される。	OutputDir FileSeqEnabled OutputFileName	*ユーザー指定のパス名* true Native

表 9. 出力ファイルの指定 (続き)

要求される出力条件	属性/プロパティーに必要な構成	属性/プロパティーの値
コネクターが生成したファイル名をすべて戻す。任意のタイプのビジネス・オブジェクトが自身の固有ファイルに書き込まれ、そのファイル名がビジネス・オブジェクトの名前と生成された固有のシーケンス番号から指定される。	動的子メタオブジェクトを使用する	
	AppSpecificInfo (ビジネス・オブジェクト・レベル)	cw_mo_JTextConfig = *DynChildMOName*
	InFileName (動的子メタオブジェクトの場合)	CxIgnore
	FileWriteMode (動的子メタオブジェクトの場合)	該当なし
	次のメタオブジェクト構成を使用する。	
	MO_JTextConnector_businessobjectname:	
	OutputDir	*ユーザー指定のパス名*
	FileSeqEnabled	true
	OutputFileName	Native
任意のタイプのすべてのビジネス・オブジェクトが、ユーザー指定の名前が付いた単一ファイルに追加される。	FileSeqEnabled	false
	OutputFileName	*ユーザー指定のパス名およびファイル名*
任意のタイプのビジネス・オブジェクトが自身の固有ファイルに書き込まれ、そのファイル名がユーザー指定の名前と固有のシーケンス番号から構成される。	FileSeqEnabled	true
コネクターが複数のタイプのビジネス・オブジェクトを処理していて、OutputFileName にストリング Native 以外が設定されている場合、各ビジネス・オブジェクトに自身のトップレベル・メタオブジェクトを持たせる必要があります。詳細については、64 ページの『出力ファイル名の指定』を参照してください。	OutputFileName	*ユーザー指定のパス名およびファイル名*
任意のタイプのビジネス・オブジェクトが出力ファイルを上書きし、そのファイル名がビジネス・オブジェクト名から指定される。	OutputDir	*ユーザー指定のパス名*
	FileSeqEnabled	false
	OutputFileName	Native

表 9. 出力ファイルの指定 (続き)

要求される出力条件	属性/プロパティーに必要な構成	属性/プロパティーの値
コネクターが生成したファイル名をすべて戻す。任意のタイプのビジネス・オブジェクトが自身の固有ファイルに書き込まれ、そのファイル名がユーザー指定の名前と固有のシーケンス番号から構成される。	動的子メタオブジェクトを使用する	
	AppSpecificInfo (ビジネス・オブジェクト・レベル)	cw_mo_JTextConfig = *DynChildMOName*
	InFileName (動的子メタオブジェクトの場合)	CxIgnore
	FileWriteMode (動的子メタオブジェクトの場合)	該当なし
	次のメタオブジェクト構成をしようする。	
	MO_JTextConnector_BOName:	
	FileSeqEnabled	true
	OutputFileName	*ユーザー指定のパス名およびファイル名*

複数のイベント・ファイルまたは複数のイベント・ディレクトリーの指定

コネクターを構成して、指定された拡張子を持つファイルのみを取り出すことができます。また、コネクターを構成して、複数のディレクトリーからファイルを取り出すこともできます。

重要: EventExt 属性にアスタリスク (*) を使用すると、拡張子がサポートされていなくても、単一イベント・ディレクトリー内のすべてのファイルをポーリングできます。

ビジネス・オブジェクト・タイプごとに別々のイベント・ディレクトリーを指定するには、以下のステップを実行します。

1. サポートされるビジネス・オブジェクトごとに個別のメタオブジェクトを作成します。例えば、MO_JTextConnector_Customer や MO_JTextConnector_Itemを作成します。詳細については、87 ページの『特定のビジネス・オブジェクトの JText メタオブジェクトの作成』を参照してください。
2. 各メタオブジェクトの EventDir 属性に適切なディレクトリーを指定します。

注: JText コネクターは、イベント・ファイルをタイム・スタンプの古いものから順に、ロケーションに関係なく処理します。つまり JText コネクターは、タイム・スタンプの日時順に別々のディレクトリーにあるファイルを処理します。

特定のビジネス・オブジェクトのポーリング

JText コネクターの構成は、さまざまな要素によって異なります。例えば、すべてのイベント・ファイルが単一ディレクトリーにあるか、すべて同じ拡張子が付いているか、含まれるビジネス・オブジェクトが単一か複数か、含まれるビジネス・オブジェクトのタイプが単一か複数か、各ビジネス・オブジェクトを単一行で表しているか複数行で表しているか、などです。

このセクションの内容は、以下のとおりです。

- 75 ページの『EndBODelimiter ベースの解析手法の使用』
  - 76 ページの『EndBODelimiter の値としての印刷不可能文字の使用』

- 78 ページの『FixedBOSize ベースの解析手法の使用』

EndBODelimiter ベースの解析手法の使用

EndBODelimiter メタオブジェクト属性に値が指定されない場合、コネクターは以下のことを実行します。

- イベント・ファイルでビジネス・オブジェクト・ストリングが <EndBO:BOName> で区切られるよう設定します。
- ビジネス・オブジェクト・ストリングを出力ファイルに書き込むときに、区切り文字として <EndBO:*BOName*> を指定します。

イベント・ファイルに 1 つのビジネス・オブジェクトしか含まれない場合、この属性に EOF (ファイル終わり) を指定できます。

EndBODelimiter 属性の値に空でないストリングを設定すると、ストリングはすべてのファイルのビジネス・オブジェクト区切り文字と見なされます。値が設定されていないかクリアされている場合、コネクターは <EndBO:*BOName*> を区切り文字と見なします。

重要: DataProcessingMode が Binary に設定されていて、EndBODelimiter の値が指定されていない場合、JText はデフォルトの EndBODelimiter を FF01 (2 バイト) に設定し、デフォルトの EndOfFileDelimiter を FF02 (2 バイト) に設定します。

表 10 は区切り文字のオプションを示しています。

表 10. EndBODelimiter 属性の使用

条件	区切り文字	注
ファイルに 1 つ以上のビジネス・オブジェクト・タイプのビジネス・オブジェクト・ストリングが 1 つ以上含まれているか、同一タイプのビジネス・オブジェクトのビジネス・オブジェクト・ストリングが複数含まれている。各ストリングの行が複数になっている。	<EndBO:*BOName*>または EOL またはユーザー指定値	• ビジネス・オブジェクト・ストリングの間に改行があるときは、その数だけセミコロンで区切られた EOL を指定します。 • EOL とともにカスタム区切り文字を指定します。カスタム区切り文字を EOL とともに使用するときは、常に最初の要素にする必要があります。次の例は有効です。 customEndBO;EOL;EOL次の例は有効では**ありません**。 EOL;customEndBO;EOL

表 10. EndBODelimiter 属性の使用 (続き)

条件	区切り文字	注
各ファイルにビジネス・オブジェクト・ストリングが 1 つのみ含まれる。	EOL FまたはユーザーEOL 指定値	• ビジネス・オブジェクト・ストリングの間に改行があるときは、その数だけセミコロンで区切られた EOL を指定します。 • 入力ストリングに必要な場合は、EOL および EOF とともにユーザー指定の区切り文字を指定します。カスタム区切り文字を EOL とともに使用するときは、常に最初の要素にする必要があります。次の例は有効です。customEndBO;EOL;EOL次の例は有効では**ありません**。EOL;customEndBO;EOL
ファイルに 1 行につき 1 つずつ複数のビジネス・オブジェクト・ストリングが含まれる。	EOL	
ファイルに同一タイプのビジネス・オブジェクトのビジネス・オブジェクト・ストリングが複数含まれている。各ストリングは複数行に渡り、ビジネス・オブジェクト・ストリング間に区切り文字がない。	なし	出荷時のデフォルト・メタオブジェクトまたはカスタム・メタオブジェクトを使用できます。 **注:** このオプションを使用できるのはサービス呼び出し要求時のみで、イベント通知には使用できません。この区切り文字を他の区切り文字とともに使用しないでください。

注: ソース・ファイルに空の行が含まれる場合、コネクターはそれらを無視します。

EndBODelimiter の値としての印刷不可能文字の使用: 複数のディレクトリーでファイルをポーリングするには、サポートされるビジネス・オブジェクトごとにメタオブジェクトを作成する必要があります。各メタオブジェクトの EndBODelimiter 属性に指定する値は、ソース・ファイルに含まれるビジネス・オブジェクトが単一か複数かによって異なります。

- ファイルに単一のビジネス・オブジェクトが含まれる。

  データ・ファイル全体に含まれるビジネス・オブジェクト・ストリングが 1 つのみの場合、EOF を EndBODelimiter として指定できます。
- ファイルに複数のビジネス・オブジェクトが含まれる。

  入力ファイルに複数のビジネス・オブジェクトが含まれ、ビジネス・オブジェクト区切り文字として改行しか使用していない場合、EndBODelimiter 属性にストリング EOL を指定します。この場合、ソース・ファイルに含まれるストリングは、同一タイプの複数のビジネス・オブジェクトを表します。

  重要: 複数のビジネス・オブジェクト・タイプを含むファイルからポーリングするには、MO_JTextConnector_Default メタオブジェクトを使用し、その EventExt および EventDir 属性に、このイベント・ファイルが置かれているディレクトリーを正しく指定する必要があります。また、別々のイベン

ト・ファイルに表されているビジネス・オブジェクト・タイプ、または別々のディレクトリーにイベント・ファイルを格納するビジネス・オブジェクト・タイプをポーリングするには、それぞれのタイプごとにトップレベル・メタオブジェクトを作成する必要があります。EventExt および EventDir 属性を使用して、適切なディレクトリーを指定します。

異なるタイプの複数のビジネス・オブジェクトを含むファイルをポーリングするとき、カスタム・データ・ハンドラーを使用するには、87 ページの『同一ファイルから異なるタイプの複数のビジネス・オブジェクトを読み取る』を参照してください。

名前/値のフォーマットを使用する場合、イベント・ファイルで複数行に渡るビジネス・オブジェクト・データが分割されていると、EOL ビジネス・オブジェクト区切り文字を指定できません。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。

以下の例は、さまざまなイベント・ファイルのフォーマットに使用する区切り文字を示しています。

- ファイルに 4 つのビジネス・オブジェクト・ストリングが含まれ、ビジネス・オブジェクト区切り文字の終了として印刷不可能文字 EOL が使用されている。

```
Sample_BO~Create~1~TableGenKey5~strange~TextConnector_924055528_0
Sample_BO~Create~2~TableGenKey5~strange~TextConnector_924055528_0
Sample_BO~Create~3~TableGenKey5~strange~TextConnector_924055528_0
Sample_BO~Create~4~TableGenKey5~strange~TextConnector_924055528_0
```

- ファイルに 4 つのビジネス・オブジェクト・ストリングが含まれ、ユーザー指定値が使用されている。ビジネス・オブジェクト区切り文字の終了として印刷不可能文字 EOL が使用されている。つまり、CustomEndBO;EOL となる。

```
Sample_BO~Create~1~TableGenKey5~strange~TextConnector_924055528_0CustomEndBO
Sample_BO~Create~2~TableGenKey5~strange~TextConnector_924055528_0CustomEndBO
Sample_BO~Create~3~TableGenKey5~strange~TextConnector_924055528_0CustomEndBO
Sample_BO~Create~4~TableGenKey5~strange~TextConnector_924055528_0CustomEndBO
```

- ファイルに 4 つのビジネス・オブジェクト・ストリングが含まれ、ビジネス・オブジェクト区切り文字の終了として印刷不可能文字 EOL;EOL が使用されている。

```
Sample_BO~Create~1~TableGenKey5~strange~TextConnector_924055528_0

Sample_BO~Create~2~TableGenKey5~strange~TextConnector_924055528_0

Sample_BO~Create~3~TableGenKey5~strange~TextConnector_924055528_0

Sample_BO~Create~4~TableGenKey5~strange~TextConnector_924055528_0
```

- ファイルに 4 つのビジネス・オブジェクト・ストリングが含まれ、ビジネス・オブジェクト区切り文字の終了として None が使用されている。

```
Sample_BO~Create~1~TableGenKey5~strange~TextConnector_924055528_0Sample_BO
~Create~2~TableGenKey5~strange~TextConnector_924055528_0Sample_BO~Create~3
~TableGenKey5~strange~TextConnector_924055528_0Sample_BO~Create~4
~TableGenKey5~strange~TextConnector_924055528_0
```

注: コネクターは、指定したストリングの大/小文字を区別します (EOL および EOF 区切り文字を除く)。

メタオブジェクトの作成の詳細については、87 ページの『特定のビジネス・オブジェクトの JText メタオブジェクトの作成』を参照してください。

FixedBOSize ベースの解析手法の使用

このメタオブジェクト・プロパティーが有効になるのは、以下の両方に該当する場合に限られます。

1. イベント処理を実行中である。
2. DataProcessingMode が Binary に設定されている。

このメタオブジェクト・プロパティーが有効な値とともに指定されている場合、このプロパティーが EndBODelimiter プロパティーに優先し、ユーザーは従来の区切り文字ベースのビジネス・オブジェクト解析に代わる解析を行うことができます。このプロパティーを使用すると、コネクターは一定数のバイトを 1 つのビジネス・オブジェクトに対応付けます。例えば、あるファイルが 300 バイトで構成されており、FixedBOSize プロパティーが 100 に設定されている場合、JText アダプターは、バイナリー対応のデータ・ハンドラーを使用してこのファイルを 3 つの 100 バイト長のパケットに変換し、InterChange Server Express に送信します。

FixedBOSize と EndBODelimiter の両方に値が設定されている場合、Jtext は FixedBOSize に基づいてファイル解析を行い、EndBODelimiter を無視します。

リモート FTP ファイル・システムの指定

このセクションでは、JText アダプターを構成して、イベント処理および要求処理にリモート FTP ファイル・システムを使用する方法について説明します。

重要: コネクターがリモート FTP ファイル・システムを使用するためには、EventDir 属性 (イベント処理用) または OutputDir 属性 (要求処理用) に FTP URL を指定する必要があります。また、コネクターを使用して FTP 操作を実行する前に、すべてのファイアウォール問題を解決する必要があります。

このセクションの内容は、以下のとおりです。

- 『リモートのイベント処理』
- 83 ページの『リモートの要求処理』
- 84 ページの『FTP 転送用にコネクターを構成する際の注意事項』

リモートのイベント処理

コネクターを構成してイベント処理用にリモート FTP ファイル・システムを使用するには、FTP URL、FTP ログイン情報、コネクターがリモート・ディレクトリーからダウンロードしたイベント・ファイルを格納するローカル・ディレクトリー、アーカイブ情報、および FTP サーバーが使用不能な場合のコネクターの振る舞いに関する情報を指定する必要があります。このセクションでは、これらの構成と追加のオプション構成について説明します。

- 79 ページの『FTP URL およびログイン情報の指定』
- 79 ページの『ローカル・ディレクトリーの指定』
- 80 ページの『リモート・アーカイブの指定』
- 81 ページの『リモート・ポーリングの指定』
- 81 ページの『コネクターがリモート・サイトからイベントを処理する方法』
- 82 ページの『メインフレーム上のファイルの識別: オプション構成』

- 82 ページの『イベント処理の構成操作の要約』

FTP URL およびログイン情報の指定: コネクターは、EventDir メタオブジェクト属性で指定したディレクトリーからイベントをポーリングします。コネクターを構成して、イベント処理用にリモート FTP ファイル・システムを使用するには、この属性の値として FTP URL を指定します。FTP URL は、IETF 標準に準拠している必要があります。

URL に FTP サーバーを指定するほか、オプションで以下の情報を EventDir メタオブジェクト属性に指定することができます。

- FTP サーバーに接続して FTP 操作を実行するための特権を持つユーザー名— EventDir でユーザー名を指定しない場合、FTPUserId メタオブジェクト属性で指定します。
- FTP サーバーに接続して FTP 操作を実行するための特権を持つユーザーのパスワード— EventDir でパスワードを指定しない場合、FTPPassword メタオブジェクト属性で指定します。
- ポート番号— EventDir でポート番号を指定しない場合、コネクターはデフォルトのポートを使用します。
- リモート・イベント・ディレクトリー— EventDir でリモート・イベント・ディレクトリーを指定しない場合、コネクターは FTP サーバーへの接続が確立しているディレクトリーからイベント・ファイルをポーリングします。

重要: FTP の値は、静的トップレベル・メタオブジェクトまたは動的子メタオブジェクトのどちらにも指定できます。ユーザー名とパスワードがいずれのメタオブジェクト属性にも指定されていない場合、コネクターは FTP サーバーへの接続の試行を終了します。詳細については、4 ページの『動的子メタオブジェクトの使用』を参照してください。

以下の例は、EventDir 属性値の 3 つの異なるフォーマットを示しています。

URL および必須値のみ:

```
ftp://ftp.companyA.com
```

URL およびオプションのユーザー名とポート番号:

```
ftp://companyA:admin@ftp.companyA.com:1433
```

URL およびオプションのユーザー名、ポート番号、リモート・イベント・ディレクトリー:

```
ftp://companyA:admin@ftp.companyA.com:1433/temp/JTextConn/Default/Event
```

Linux/MVS 関連の FTP セットアップの URL

```
ftp://ftpuser:ftppwd@ftpserver.in.ibm.com:21/home/ftpuser/JText/event
```

ローカル・ディレクトリーの指定: FTP URL および関連するログイン情報を指定するほかに、コネクターがリモート・ディレクトリーからダウンロードしたイベント・ファイルを格納するローカル・ディレクトリーを指定する必要があります。ローカル・ディレクトリーを指定するには、FTPLocalEventDir メタオブジェクト属性を使用します。

重要: コネクターが EventDir に適切な FTP URL を見つけても、FTPLocalEventDir メタオブジェクト属性がなかったり、この属性に無効な値やブランクが指定されている場合、コネクターは始動しません。コネクターがローカルに実行するよう構成されている場合、FTPLocalEventDir 属性は評価されません。

リモート・アーカイブの指定: コネクターによるリモート・アーカイブの処理方法の指定には、いくつかのオプションがあります。リモート・アーカイブ・ディレクトリーを指定するには、FTPArchiveDir メタオブジェクト属性を使用します。この属性により、FTP サーバーにあるアーカイブ・ディレクトリーの相対パスを指定します。ディレクトリーは既存のものである必要があります。この属性を使用したアーカイブの指定には、いくつかのオプションがあります。

- FTPArchiveDir 属性には値を指定し、FTPRenameExt 属性には値を指定しない場合、コネクターは、イベント・ファイル名にタイム・スタンプを追加し、さらにそのファイルを FTPArchiveDir 属性に指定されているリモート FTP サーバーのアーカイブ・ディレクトリーに移動します。
- FTPArchiveDir 属性と FTPRenameExt 属性の両方に値を指定した場合、コネクターは、処理したイベント・ファイルの名前を変更し (タイム・スタンプのみ追加し、FTPRenameExt は無視します)、さらにそのファイルを FTPArchiveDir 属性に指定されている FTP サーバーのアーカイブ・ディレクトリーに移動します。
- FTPArchiveDir 属性にも FTPRenameExt 属性にも値を指定しない場合、コネクターは処理済みのイベント・ファイルをアーカイブせずに削除します。
- FTPArchiveDir 属性に値を指定しないで FTPRenameExt 属性に値を指定した場合、コネクターは、処理したイベント・ファイルの名前を変更し (FTPRenameExt 属性に指定されている値を追加します)、さらにそのファイルを EventDir 属性に指定されているディレクトリーに移動します。

ファイル名へのタイム・スタンプの追加 (リモート FTP サーバー対応): 順次データ・セットを使用するホスト・ファイル・システム (MVS) のサポートが拡張され、ファイル名の重複を回避するためにタイム・スタンプを追加できるようになりました。MVS の場合、データ・セット名やレコード・セット名に特殊文字 (「_」など) を使用できません。Windows、Linux、または i5/OS プラットフォームでは、ファイルのアーカイブ時に、オリジナルのファイル名の中にタイム・スタンプが使用されます。これにより、アーカイブ・フォルダー内でのファイル名の重複が回避され、既存ファイルの上書きが防止されています。

MVS システムでは、上記の制限に対処するため、次のフォーマットが使用されます。
イベント・ファイル: Test.in
アーカイブ・ファイル: Test.TSyyyyMM.TSDDHHMM.TSSsSss
各部分の意味:
yyyy -- 年
MM -- 月
DD -- 日
HH -- 時間
MM -- 分
Ss -- 秒
Sss -- ミリ秒
MVS プラットフォームでは、データ・セット名とレコード・セット名に使用される

区切り文字は「.」 (ドット) であり、1 つのデータ・セット名またはレコード・セット名に使用できる「.」(ドット) の数は最大 6 個です。データ・セット名またはレコード・セット名では、「.」 (ドット) によって区切られている各部分の文字数が 8 文字を超えてはならず、全文字数が 44 文字を超えてはなりません。このフォーマットのファイル名の例を、以下に示します。

```
FTPRenameExt -- ARCHIVE
アーカイブ・ファイル -- (SAMPLE).ARCHIVE.TS200304.TS290535.TS42234
```

注: PDS のメンバーは、アーカイブ中にタイム・スタンプによって名前変更することはできません。そのため、PDS アーカイブ用に代替手段が用意されています。FTPEventFileMask に入る PDS の各メンバーは、ファイル名を FTPRenameExt として指定して親 PDS の下位にアーカイブされます。アーカイブ・ファイルは、最新の処理済みファイルごとに再書き込みされます。

リモート・ポーリングの指定: 36 ページの『FTPPollFrequency』 構成プロパティーを使用して、コネクターが FTP サーバーにポーリングする頻度を標準ポーリング・サイクル数で設定できます。この設定は、コネクターが次のポーリング・サイクルを開始するときに、まだローカル・イベント・ディレクトリーからファイルを読み取っている場合に便利です。

例えば、 127 ページの『PollFrequency』 が 10000 に設定され、FTPPollFrequency が 6 に設定されている場合、コネクターはローカル・イベント・ディレクトリーに 10 秒ごとにポーリングし、リモート・ディレクトリーに 60 秒ごとにポーリングします。コネクターが FTP ポーリングを実行するのは、このプロパティーに値を指定した場合のみです。FTPPollFrequency に 0 またはブランクが設定された場合、コネクターは FTP ポーリングを実行しません。

詳細については、 89 ページの『JText コネクターのパフォーマンス調整』を参照してください。

コネクターがリモート・サイトからイベントを処理する方法: リモート・サイトからイベントをポーリングする場合、コネクターは以下のステップを実行します。

1. メタオブジェクト属性またはデフォルト値から、サーバー名、ポート番号、ユーザー名、パスワード、およびリモート・イベント・ディレクトリーを取得します。
2. リモート FTP サイトへの接続を確立して、リモート・ディレクトリーからイベント・ファイルを取得します。
3. リモート・ディレクトリーから、FTPLocalEventDir メタオブジェクト属性で指定したローカル・ディレクトリーにイベント・ファイルをダウンロードします。

   注: コネクターが FTP を使用してイベントを処理するには、この属性に値が設定されていなければなりません。
4. ローカル・ディレクトリーにポーリングします。

図 6 は、ローカルおよびリモートのイベント処理を示しています。

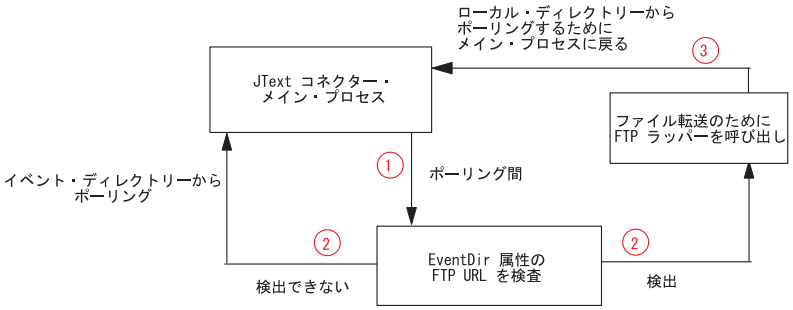

図 6. ローカルおよびリモートのイベント通知操作

メインフレーム上のファイルの識別: オプション構成: FTPEventFileMask 属性を使用して、Windows、Linux、または i5/OS システムに適用されるのと同じ命名標準に準拠しないメインフレーム上でファイル拡張子を識別します。この属性に値が設定されないと、コネクターは EventExt 属性で指定された値を使用します。

FTPEventFileMask に値を設定するときには、ワイルドカード文字を組み込むことができます。次の例は、この属性に使用できるいくつかのフォーマットです。

```
ACT.Z1UC.*
ACT.*.INPT
*.Z1UC.INPT
```

コネクターは、リモート・サイトで FTPEventFileMask に指定された基準に一致するファイルを複数見つけると、以下のことを実行します。

1. 指定されたすべてのリモート・イベント・ファイルを、FTPLocalEventDir 属性で指定されたディレクトリーにダウンロードします。
2. FTPRenameExt メタオブジェクト属性に指定された値を持つリモート・ファイルの拡張子を変更します。ファイル名を変更することによって、次のポーリング・サイクルに同じファイルをポーリングすることがなくなります。
3. FTP サーバーから接続を切断します。
4. FTPEventFileMask メタオブジェクト属性に指定されたディレクトリー内で、ローカルにファイルを処理します。

イベント処理の構成操作の要約: コネクターを構成して、イベント処理用にリモート FTP ファイル・システムを使用するには、以下の構成値を指定します。

- EventDir メタオブジェクト属性に FTP URL を指定します。オプションで、FTP サーバーに接続して FTP 操作を実行する特権を持つユーザー名とパスワードを指定します。
- EventDir メタオブジェクト属性にログイン名とパスワードを指定しない場合、FTPUserId および FTPPassword メタオブジェクト属性にそれらを指定します。
- EventDir メタオブジェクト属性にポートを指定しない場合、コネクターはデフォルトの FTP ポートを使用します。
- FTPLocalEventDir メタオブジェクト属性を使用して、コネクターが FTP サイトからダウンロードしたイベント・ファイルを格納するローカル・システム・ディレクトリーを指定します。

- Windows、Linux、または i5/OS システムに適用されるのと同じ命名標準に準拠しないメインフレーム上で、FTPEventFileMask メタオブジェクト属性を使用してポーリングするファイルを指定します。
- リモート・システムが MVS のとき、MVS FTP サーバーと連動するようにコネクターを構成するには、FTPOSPlatform 属性に MVS 属性を指定します。

リモートの要求処理

コネクターを構成してイベント処理用にリモート FTP ファイル・システムを使用するには、FTP URL、FTP ログイン情報、コネクターがローカル・ディレクトリーからアップロードした要求ファイルを格納するリモート・ディレクトリーを指定する必要があります。このセクションでは、これらの構成と追加のオプション構成について説明します。

- 『FTP URL およびログイン情報の指定』
- 84 ページの『コネクターがリモート・サイトへのサービス呼び出し要求を処理する方法』
- 84 ページの『要求処理の構成操作の要約』

FTP URL およびログイン情報の指定: コネクターは、OutputDir メタオブジェクト属性で指定したディレクトリーにサービス呼び出し要求ファイルをアップロードします。コネクターを構成して要求処理用にリモート FTP ファイル・システムを使用するには、この属性の値として FTP URL を指定します。FTP URL は、IETF 標準に準拠している必要があります。

FTP URL のほかに、オプションで以下の情報を OutputDir メタオブジェクト属性に指定することができます。

- FTP サーバーに接続して FTP 操作を実行するための特権を持つユーザー名— OutputDir でユーザー名を指定しない場合、FTPUserId メタオブジェクト属性で指定します。
- FTP サーバーに接続して FTP 操作を実行するための特権を持つユーザーのパスワード— OutputDir でパスワードを指定しない場合、FTPPassword メタオブジェクト属性で指定します。
- ポート番号— EventDir でポート番号を指定しない場合、コネクターはデフォルトのポートを使用します。
- リモート出力ディレクトリー— OutputDir でリモート出力ディレクトリーを指定しない場合、コネクターは要求ファイルをデフォルトの接続ディレクトリー (接続が確立している FTP サーバーのディレクトリー) にロードします。

重要: FTP の値は、静的トップレベル・メタオブジェクトまたは動的子メタオブジェクトのどちらにも指定できます。ユーザー名とパスワードがいずれのメタオブジェクト属性にも指定されていない場合、コネクターは例外のスローによって終了します。詳細については、4 ページの『動的子メタオブジェクトの使用』を参照してください。

以下の例は、OutputDir 属性値の 3 つの異なるフォーマットを示しています。

URL および必須値のみ:

```
ftp://ftp.companyA.com
```

URL およびオプションのユーザー名とポート番号:

```
ftp://companyA:admin@ftp.companyA.com:1433
```

URL およびオプションのユーザー名、ポート番号、リモート出力ディレクトリー:

```
ftp://companyA:admin@ftp.companyA.com:1433/temp/JTextConn/Default/Out
```

コネクターがリモート・サイトへのサービス呼び出し要求を処理する方法: コネクターが FTP 処理用に構成されている場合、サービス呼び出し要求を受け取ると以下のステップを実行します。

1. メタオブジェクト属性またはデフォルト値から、サーバー名、ポート番号、ユーザー名、およびパスワードを取得します。
2. リモート FTP サイトへの接続を確立して、ローカル・ディレクトリーからサービス呼び出し要求ファイルを置きます。
3. ローカル・ディレクトリーからリモート・ディレクトリーに要求ファイルをアップロードします。
4. リモート・サーバーから接続を切断します。

図 7 は、ローカルおよびリモートの要求処理を示しています。

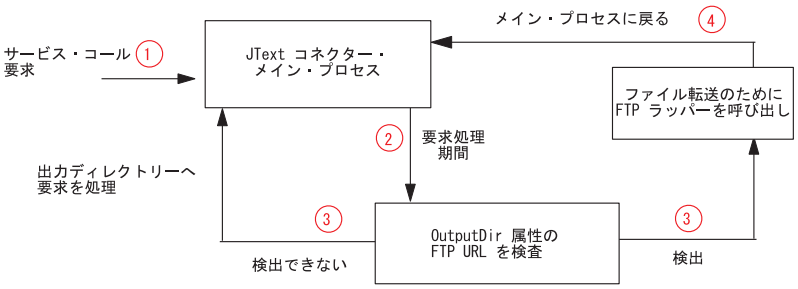

図 7. ローカルおよびリモートの要求操作

要求処理の構成操作の要約: コネクターを構成して、要求処理用にリモート FTP ファイル・システムを使用するには、以下の構成値を指定します。

- OutputDir メタオブジェクト属性に FTP URL を指定します。オプションで、FTP サーバーに接続して FTP 操作を実行する特権を持つユーザー名とパスワードを指定します。
- OutputDir メタオブジェクト属性にログイン名とパスワードを指定しない場合、FTPUserId および FTPPassword メタオブジェクト属性にそれらを指定します。
- OutputDir メタオブジェクト属性にポートを指定しない場合、コネクターはデフォルトのポートを使用します。
- リモート・システムが MVS のとき、MVS FTP サーバーと連動するようにコネクターを構成するには、FTPOSPlatform 属性に MVS 属性を指定します。

FTP 転送用にコネクターを構成する際の注意事項

以下の機能は、データの FTP 転送に適用されます。

- FTP 操作時、コネクターはバイナリー・モードのデータ転送を行います。
- EventDir または OutputDir メタオブジェクト属性の値が ftp:// で始まらない場合、コネクターはデータの FTP 転送を行いません。

- イベント処理時に、イベント・ビジネス・オブジェクトに InFileName 属性を持つ動的子メタオブジェクトが含まれない場合、コネクターはこの属性を FTPLocalEventDir で指定されたファイルの絶対パスとともに取り込みますが、リモート・システムのパスは取り込みません。
- EventExt および FTPRenameExt メタオブジェクト属性に入力する値は、同じにはできません。これらが同じ場合、コネクターは以前に取り出したファイルを取り出し続けてしまいます。
- コネクターは、FTP でサポートされないファイル・サイズをサポートしません。
- FTP サイトのプラットフォームに従って、ファイル名、拡張子、および他のコンポーネントには大文字小文字を区別する必要があります。
- リモート FTP サイトからファイルを転送すると、コネクターのパフォーマンスに影響を及ぼす場合があります。
- データがリモート FTP サイトとの間で交換されるとき、ネットワーク接続の損失や同様の問題によって、データが破壊または破損する可能性があります。
- 統合ブローカーは、いかなるタイプの接続キャッシュやプールも保持しません。接続は、ポーリング・サイクルおよび要求処理のたびに開閉します。コネクターの制御が及ばないネットワーク待ち時間や他の構成が、そのパフォーマンスに影響を及ぼす場合があります。
- FTPLocalEventDir メタオブジェクト属性の値には、FTP 値を指定していないメタオブジェクトの EventDir メタオブジェクト属性と同じ値を指定できません。この制約事項によって、同一ディレクトリーにあるがまったく異なるタイプの処理が必要な異なるタイプのビジネス・オブジェクトに指定された値を、コネクターが使用しないようになります。
- FTP URL の終わりにリモート・イベント・ディレクトリーまたは出力ディレクトリーが存在しない場合、コネクターは FTP サイトと対話するとシャットダウンします。コネクター始動時にはシャットダウンしません。
- FTP サーバーを介するファイルの処理のためにコネクターを構成するときには、FTP サーバーを構成して、Apache Commons Net API に必要な Linux 設定を使用するようにします。

セキュア FTP の構成

JText アダプターは、SSL (Secure Socket Layer) プロトコルを使用して、アダプターと FTP サーバーの間で機密保護機能のある通信チャネルを確立します。この機能では、FTP サーバーをこのプロトコルをサポートするセキュア FTP サーバーにする必要があります。JText アダプターは、SSL プロトコルの実装である IBM JSSE パッケージを利用します。アダプターは受動 FTP モードと暗黙的 SSL モードで動作します。このセクションでは、JText アダプターのセキュア FTP 機能を使用するために必要な、追加の構成について説明します。

セキュア FTP を構成するには、次のステップを実行します。

1. SSL プロトコルをサポートするセキュア FTP サーバーをインストールして構成します。JText アダプターは、SSL プロトコルを使用してクライアント・アプリケーションと FTP サーバーの間でデータを転送します。したがって、セキュア FTP サーバーは、SSL プロトコルに対応するようにインストールされ、SSL 通信用に適切に構成される必要があります。サーバーには、秘密鍵と証明書が必要です。

2. 適用可能であれば、ファイアウォールの設定を構成します。JText アダプターは、セキュア FTP サーバーによりデータ転送の受動 FTP モードを使用します。したがって、クライアントとサーバーの間にファイアウォールがある場合は、ファイアウォールの設定を構成してこのモードを使用可能にする必要があります。
3. クライアントのトラスト・ストアを設定します。SSL での通信時に、サーバーは、検証のため自身の証明書をクライアントに送信します。クライアントはその証明書を検証し、意図するサーバーとの通信であることを確認します。この検証プロセスを使用可能にするため、サーバーの証明書をクライアントのトラスト・ストアに格納してください。サーバーの証明書は、keytool ユーティリティーを使用してクライアントのトラスト・ストアにインポートできます。例えば、次のようにします。

   ```
   keytool -import -v -alias serverCert -file server.cert
   -keystore clientTrustStore
   ```

   ここで、server.cert はサーバーの証明書、clientTrustStore はクライアントのトラスト・ストアです。
4. アダプターの始動スクリプトの trustStore システム・プロパティーを設定します。アダプターの始動スクリプトには、次のシステム・プロパティーを組み込んでください。

   ```
   -Djavax.net.ssl.trustStore=C:¥MyKeyStore¥clientTrustStore
   ```

   ここで、clientTrustStore はステップ 3 で指定したクライアントのトラスト・ストアです。
5. FTP URL で ftps を使用していることを確認します。機密保護機能のある通信の場合は、FTP URL のプロトコルを ftps にしてください。例えば、`ftps://host:port/ftpdir` のようにします。

データ・ハンドラーの指定

JText コネクターで使用されるデータ・ハンドラーを指定するには、以下のステップを実行します。

1. JText コネクターが通信するアプリケーションで使用されるフォーマットを判別します。どのフォーマット・タイプでも、登録できるデータ・ハンドラー・クラスは 1 つだけです。
2. 以下のトップレベル JText メタオブジェクトの子オブジェクトを構成します。
   - `EventDataHandler`: イベント処理 (ビジネス・オブジェクトを表すストリングまたはバイト配列をビジネス・オブジェクトに変換する処理) に使用するデータ・ハンドラー・メタオブジェクトを指定します。
   - `OutputDataHandler`: 要求処理 (ビジネス・オブジェクトを、ビジネス・オブジェクトを表すストリングまたはバイト配列に変換する処理) に使用するデータ・ハンドラー・メタオブジェクトを指定します。

指定されたデータ・ハンドラーの変更

出荷時のデフォルト・データ・ハンドラーを (出荷時の別のデータ・ハンドラーまたはカスタム・データ・ハンドラーに) 変更するには、以下のことを実行します。

- EventDataHandler および OutputDataHandler 属性でデフォルト値として指定されているビジネス・オブジェクトを、コネクターがサポートするかどうかを検証します。
- コネクターの始動時に、データ・ハンドラーを含むクラスまたは JAR ファイルがクラス・パスに組み込まれているかどうかを検証します。(「データ・ハンドラー・ガイド」で推奨されているように) 出荷時のデータ・ハンドラーを使用するか、CustDataHandler.jar ファイルにカスタム・データ・ハンドラーを追加する場合、ファイルは出荷時の開始スクリプト (start_JText.bat または connector_manager_JText.sh) に組み込まれています。
- 必ず、使用しているデータ・ハンドラーに適切な EndBODelimiter 値を指定します。

データ・ハンドラーの作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。

特定のビジネス・オブジェクトの JText メタオブジェクトの作成

特定のビジネス・オブジェクトの JText メタオブジェクトを作成するときは、メタオブジェクト名を変更して特定のビジネス・オブジェクトを示すようにします。例えば、Customer および Item ビジネス・オブジェクトのメタオブジェクトを作成するには、メタオブジェクトにそれぞれ MO_JTextConnector_Customer および MO_JTextConnector_Item と名前を付けます。

ヒント: ファイルに書き込まれるすべてのビジネス・オブジェクトがまったく同じ構成の場合、デフォルト・メタオブジェクトを使用します。つまり、同じイベント・ディレクトリーにあり、同じ出力ディレクトリーに書き込まれるすべてのテキスト・ファイルは、同じデータ・ハンドラーを使用し、同じファイル拡張子を持ちます (または同じファイルに格納されます)。要求時に、コネクターが異なるビジネス・オブジェクトに必ず異なる処理を行うようにする場合、またはポーリングに特定の処理命令が必要な場合、独自のメタオブジェクトを作成します。特定のビジネス・オブジェクトに個別のメタオブジェクトを作成すると、コネクターは統合ブローカー要求とサブスクリプション送達操作の両方にそのメタオブジェクトを使用します。

メタオブジェクトを作成していないすべてのビジネス・オブジェクトは、デフォルトの MO_JTextConnector_Default メタオブジェクトの値で構成されます。このデフォルト・メタオブジェクトのビジネス・オブジェクト定義については、¥repository¥JText ディレクトリーを参照してください。

同一ファイルから異なるタイプの複数のビジネス・オブジェクトを読み取る

テキスト・ファイルに異なるタイプの複数のビジネス・オブジェクトが含まれている場合、MO_JTextConnector_Default メタオブジェクトを使用し、その EventExt および EventDir 属性がこのイベント・ファイルの格納先ディレクトリーを正しく示している必要があります。ファイル内の各ビジネス・オブジェクトは、同じ区切り文字で区切られていなければなりません。

出荷時のデータ・ハンドラーは、各ビジネス・オブジェクトの名前を入力ストリングから判別できます。つまり、デフォルトのトップレベル JText メタオブジェクトと出荷時のデータ・ハンドラーを使用すれば、<EndBO:*BOName*> 区切り文字を使用して、複数のビジネス・オブジェクト・タイプを含むファイル内で各タイプを識別する必要はありません。

カスタム・データ・ハンドラーを作成して、ビジネス・オブジェクト・ストリングをビジネス・オブジェクトに変換する場合、データ・ハンドラーがビジネス・オブジェクトのタイプを入力ストリングから解釈できるようにします。

ObjectEventID 属性値の指定

ビジネス・オブジェクト・ストリングに ObjectEventId 属性を追加する必要はありません。イベント通知ビジネス・オブジェクトの場合、コネクターによって ID が取り込まれないと、コネクター・フレームワークによってこれらのビジネス・オブジェクト属性が取り込まれます。

サービス呼び出し要求ビジネス・オブジェクトでは、ObjectEventId 属性は無視されるかファイルに書き込まれるストリングに組み込まれます。ObjectEventId 属性が出力ファイルに組み込まれるかどうかは、使用するデータ・ハンドラーによって異なります。

JText コネクターの 2 番目のインスタンスのセットアップ

JText コネクターに 2 番目のインスタンスをセットアップするには、以下のステップを実行します。

1. JText コネクター・ディレクトリーおよびリポジトリー・ディレクトリーのコピーを作成し、名前を変更します。例えば、2 番目のコネクター定義に JText2 という名前を付けます。2 番目のディレクトリーを作成すると、ディレクトリー構造は以下のようになります。

   ¥connectors¥JText
   ¥connectors¥JText2
   ¥repository¥JText
   ¥repository¥JText2

2. JText コネクターのすべてのメタオブジェクト (少なくとも 2 つあるはずです) をコピーし、ビジネス・オブジェクト名を変更します。例えば JText2 コネクターの場合、名前を MO_JText2Connector*BOName* から MO_JText2Connector*BOName* に変更します。

   メタオブジェクトをコピーする方法は 2 つあります。

   - テキスト・ファイルを作成し、そこに MO_JText2Connector_*BOName* メタオブジェクトおよびその子メタオブジェクトを持たせます。テキスト・エディターの検索と置換オプションを使用して、MO_JTextConnector_ を MO_JText2Connector_BOName に置換します。
   - Business Object Designer Express を使用して、メタオブジェクトを 1 つずつコピーします。

重要: Business Object Designer Express でビジネス・オブジェクト定義を操作する前に、¥repository¥ReposVersion.txt ファイルの先頭から各定義ファイルの先頭にテキストをコピーする必要があります。

3. Connector Configurator Express でコネクターの定義をコピーし、この名前を JText2Connector に変更します。サポートされるメタオブジェクトおよびビジネス・オブジェクトを変更します。
4. 新規定義ファイルをリポジトリーにコピーします。Business Object Designer Express を使用してビジネス・オブジェクト定義をリポジトリーにコピーするには、「ファイル」メニューから「サーバーに保管」サブメニューを選択します。または、InterChange Server Express で以下のステップを実行して、ビジネス・オブジェクト定義をオペレーティング・システムからリポジトリーにコピーします。
   a. ¥repository¥ReposVersion.txt ファイルの先頭から各定義ファイルの先頭にテキストをコピーします。
   b. 次の repos_copy コマンドを使用して、新規メタオブジェクトとビジネス・オブジェクトにコピーします。

      ```
      repos_copy -sServerName -iFileName
      ```
5. 統合ブローカーの管理ユーティリティーをリフレッシュして、新規ビジネス・オブジェクトを確認します。
6. Linux の場合、JText コネクターの既存のコネクター・マネージャー・スクリプトをコピーし、パラメーターを JText2 を参照するように変更します。Windows の場合、JText コネクターの既存のショートカットをコピーし、パラメーターを JText2 を参照するように変更し、JText ディレクトリーではなく、JText2 ディレクトリーを参照するように変更します。
7. i5/OS システムの場合は、コンソールに新規コネクター・インスタンスを追加します。
8. 統合ブローカーを再始動します。
9. Linux の場合、コネクター・マネージャー・スクリプトを実行します。Windows の場合、新規ショートカットをクリックします。i5/OS の場合、コンソールを使用してコネクターを始動します。

JText コネクターのパフォーマンス調整

JText コネクターのポーリング・パフォーマンスを調整するには、以下のようにコネクター構成プロパティーを設定します。

- PollQuantity - このプロパティーは、コネクターがイベントのポーリングのために、1 回の呼び出しで統合ブローカーに送達できるビジネス・オブジェクトの最大数を設定します。PollQuantity に高い値を設定すると、コネクターは 1 回のポーリングでより多くのビジネス・オブジェクトをサブミットします。これによってパフォーマンスが向上し、内部キューおよびメモリー使用量の消去に役立ちます。

  ただし、コネクターが大量のビジネス・オブジェクトを統合ブローカーに POST できるようにすると、他のビジネス・インテグレーション・コンポーネントに影響を及ぼす場合があります。例えば、メッセージ・キューイング・システムにデフォルト値がセットアップされている場合、JText コネクターが大量のビジネス・オブジェクトをシステムに送信すると、キューがすぐに一杯になることがあ

ります。このため、パフォーマンスを調整するときは、PollQuantity に最適なパフォーマンス設定をするように留意してください。

- PollFrequency - このコネクター構成プロパティーは、ポーリング・アクションの間の時間を指定します。このプロパティーに長い時間を設定すると、イベント処理時にコネクターがスローダウンします。短い時間を設定すると、イベントの選出、ビジネス・オブジェクトへの変換、およびデリバリーが迅速になります。

  つまり、コネクターはポーリング呼び出し時に新規ファイルを選出します。コネクターのポーリング頻度が少ないと、イベント・ディレクトリーで増加するファイルの引き渡しに時間がかかります。コネクターのポーリング頻度が多いと、ファイルの選出および引き渡しがより頻繁に行われます。

  ただし、コネクターがイベントをポーリングする頻度が多いほど、要求を処理する時間が短くなります。コネクターを主に要求処理に使用する場合、PollFrequency には、コネクターを主にイベント処理に使用する場合よりも小さい値を設定します。

  前述した PollQuantity 構成プロパティーと同様に、PollFrequency に極端な値 (長すぎるまたは短すぎる時間) を設定すると、他のビジネス・インテグレーション・コンポーネントのパフォーマンスに影響を及ぼします。

- FTPPollFrequency - このコネクター構成プロパティーは、コネクターが FTP サーバーにポーリングする頻度を標準ポーリング・サイクル数で指定します。例えば、PollFrequency が 10000 に設定され、FTPPollFrequency が 6 に設定されている場合、コネクターはローカル・イベント・ディレクトリーに 10 秒ごとにポーリングし、リモート・ディレクトリーに 60 秒ごとにポーリングします。コネクターが FTP ポーリングを実行するのは、このプロパティーに値を指定した場合のみです。FTPPollFrequency に 0 またはブランクが設定された場合、コネクターは FTP ポーリングを実行しません。

要約すると、ポーリングのパフォーマンスを向上させる最善の方法は、PollQuantity、PollFrequency、および FTPPollFrequency が相互に補完するように値を設定することです。

テスト用サンプル・ファイルの生成

JText コネクターで使用する入力ファイルに似たファイルを生成することができます。このファイルは、ソース・アプリケーションで出力フォーマットをセットアップするのに役立ちます。サンプル・ファイルはテストにも使用できます。

InterChange Server Express で、入力ファイルに似たファイルを生成する最も簡単な方法は以下のとおりです。

1. ファイルに書き込まれるビジネス・オブジェクトを入力として扱い、宛先に送信するパススルー・コラボレーションを作成します。
2. ビジネス・オブジェクトをサポートし、Test Connector でエミュレートできるコネクターにソース・ポートをバインドします。
3. 宛先ポートを JText コネクターにバインドします。
4. Test Connector にビジネス・オブジェクトのサンプル値を入力し、そのビジネス・オブジェクトを JText コネクターに送信します。JText コネクターは、構成されたフォーマットで出力ファイルに値を書き込みます。

このプロセスによって、単一ファイルに書き込まれた複数のビジネス・オブジェクトを確認でき、それをテストで入力として使用できます。

テスト用サンプル・ビジネス・オブジェクトの生成

JText コネクターで使用するビジネス・オブジェクトに似たビジネス・オブジェクトを生成することができます。値を持つビジネス・オブジェクトを生成して、テストに使用できます。

コネクターが自動的にビジネス・オブジェクトを生成するようにするには、GenerateTemplate 構成プロパティーを使用します。コネクターがサポートするビジネス・オブジェクトごとに、定義を生成できます。

コネクターは GenerateTemplate プロパティーの値を使用して、コネクターの始動時に、直列化したビジネス・オブジェクトのインスタンスを作成します。**直列化**したビジネス・オブジェクトとは、データ・ハンドラーが作成するビジネス・オブジェクトのストリング表記です。Connector Configurator を使用して、このプロパティーのビジネス・オブジェクト名を指定します。

このプロパティーの構文は *BOName*;*BOName* です。ここで、*BOName* は特定のビジネス・オブジェクト名です。大文字小文字は区別されます。複数のビジネス・オブジェクトを指定するには、Customer;Item のように名前をセミコロンで区切ります。終了句読点は必要ありません。次にコネクターを始動するときに、これらのビジネス・オブジェクトのテンプレートが作成されます。

生成されるテンプレートには、出荷時のデフォルト値が含まれます。これらの値は、ビジネス・オブジェクト定義内のビジネス・オブジェクト属性用に設定されています。属性用に出荷時のデフォルト値がない場合、無視される (CxIgnore を使用) かブランクのままにされます (CxBlank を使用)。各単一カーディナリティーの子ビジネス・オブジェクトに 1 つの子ビジネス・オブジェクトが作成され、複数カーディナリティーのビジネス・オブジェクトに 2 つの等しい子ビジネス・オブジェクトのインスタンスが作成されます。

特定のビジネス・オブジェクトのテンプレートを生成するには、コネクターを始動します。コネクターは、出力ファイルと同じファイルにテンプレートを書き込みます。この機能を使用しない場合は、GenerateTemplate プロパティーを空のままにします。

第 5 章 JText コネクターのトラブルシューティング

この章では、JText コネクターで発生した問題を診断するために役立つ情報を提供します。

- 『エラー・メッセージのロギング』
- 『メタオブジェクトの名前に関する問題』
- 94 ページの『イベントの起動に関する問題』
- 94 ページの『JText での障害の処理』
- 95 ページの『イベント・ログ・ファイル』
- 96 ページの『障害リカバリー』
- 98 ページの『ビジネス・オブジェクトの区切り文字エラーからのリカバリー』
- 98 ページの『サブスクリプション・エラーからのリカバリー』
- 98 ページの『フォーマット・エラーからのリカバリー』
- 99 ページの『送信エラーからのリカバリー』
- 99 ページの『データ・ハンドラーとサポートされているビジネス・オブジェクト』

エラー・メッセージのロギング

エラー・メッセージは標準コネクター・ログ・ファイル STDOUT か、または LogFileName 標準コネクター・プロパティーが指定するファイルに記録されます。

エラーはイベント・ログ・ファイルにも記録されます。イベント・ログ・ファイルの詳細については、95 ページの『イベント・ログ・ファイル』を参照してください。

メタオブジェクトの名前に関する問題

コネクターの始動時に以下のエラー・メッセージが出されるときは、メタオブジェクト名がコネクター・インスタンス名に対応していません。

```
誤ったサブスクリプション: JText_Customer にサポートする MO がありません。
このビジネス・オブジェクトはアンサブスクライブされます。(Wrong subscription:
JText_Customer doesn't have supporting MO: this BO is unsubscribed.")
```

メタオブジェクト名がコネクター・インスタンスの名前と一致しない場合は、メタオブジェクトはコネクターがサポートするビジネス・オブジェクトを認識しません。これを防ぐためには、メタオブジェクトにコネクター・インスタンスに対応する名前を付けてください。例えば、MO_JTextConnector2_Default という名前のメタオブジェクトは JText2 コネクターがサポートするビジネス・オブジェクトを認識します。

イベントの起動に関する問題

コネクターは区切り文字により以下の問題が生じるイベント・ファイルを無視します。

- トップレベルのメタオブジェクトの EndBODelimiter 属性が符号 (+) やパイプ・シンボル (|) など有効な値に設定されているが、イベント・ファイルには、指定された区切り文字がそれぞれのビジネス・オブジェクトの末尾に含まれていない。
- コネクターは EndBO:BOName というビジネス・オブジェクトの区切り文字を検索するように構成されているが、イベント・ファイルにこの区切り文字が含まれていない。コネクターは以下に示す警告メッセージをログに記録します。

ファイル *filename* から作業単位を作成できません。
EndBODelimiter がファイルにあるか確認してください。
(Unable to create Workunits from file *filename*.
Check EndBODelimiter in the file.)

上記のどちらの場合でも、イベント・ファイルは変更されずにそのままイベント・ディレクトリーにとどまります。

コネクターは、ファイルにアクセスしている、ファイルを開いている、または閉じているときにデバイスで障害が発生した場合にも、イベント・ファイルをイベント・ディレクトリー内にそのまま保持します。例えば、システムがファイルにアクセスしようとしてメモリー不足となった場合は、コネクターはそのファイルを無視します。

JText での障害の処理

JText コネクターでは、以下のタイプのエラーが発生することがあります。

表 11. JText エラー・タイプ

エラーのタイプ	説明
ビジネス・オブジェクトの区切り文字に関する障害	ビジネス・オブジェクトの区切り文字に関する障害は、トップレベルのメタオブジェクトの EndBODelimiter 属性が有効な値に設定されていて、イベント・ファイルにはそれぞれのビジネス・オブジェクトの末尾に指定された区切り文字が含まれているが、データ自体が区切り文字をテキスト内で使用しているときに発生します。コネクターはテキスト内の区切り文字の値を検出すると、ビジネス・オブジェクト・ストリングの一部を処理に失敗したフォーマッターに送信します。この場合は、コネクターはそのイベントを *filename_timestamp*.fail ファイルに書き込みます。このファイルには、区切り文字の障害が検出されたすべてのビジネス・オブジェクトのレコードが含まれます。
サブスクリプション・エラー	コネクターがビジネス・オブジェクトの区切り文字を検出しそのビジネス・オブジェクト名を取得できるが、そのビジネス・オブジェクトがサブスクライブされていないときに発生することがあります。この場合は、イベントが *filename_timestamp*.unsub ファイルに送信されます。このファイルには、アンサブスクライブされたビジネス・オブジェクトのレコードがすべて含まれます。

表 11. *JText* エラー・タイプ *(続き)*

エラーのタイプ	説明
フォーマット・エラー	コネクターが入力ビジネス・オブジェクト名と一致しないビジネス・オブジェクト名の区切り文字を検出する場合や、ビジネス・オブジェクト・ファイルのフォーマットがメタオブジェクトのフォーマットと一致しない場合に発生することがあります。イベントは *filename_timestamp*.fail ファイルに送信されます。このファイルには、フォーマットに失敗したすべてのビジネス・オブジェクトのレコードが含まれます。
送信エラー	統合ブローカーがダウンしているときにコネクターがビジネス・オブジェクトを送信しようとすると発生することがあります。Send 操作に失敗すると、イベントは *filename_timestamp*.fail ファイルに送信されます。このファイルには、正常に送信されなかったすべてのビジネス・オブジェクトのレコードが含まれます。

イベント・ログ・ファイル

コネクターは正常に処理されたビジネス・オブジェクトに関する情報を event.log ファイルに記録します。コネクターが、イベント・ファイル内のすべてのビジネス・オブジェクトを処理する前にダウンした場合は、それぞれのビジネス・オブジェクトを統合ブローカーに一度だけ送信するように、リカバリーのときにこのログ・ファイルを使用します。

ログ・ファイルの形式を以下に示します。

EventFileName::1,2,n

ここで、EventFileName は現行のイベント・ファイルの名前を表し、それぞれの番号はそのファイルの中の正常に処理されたビジネス・オブジェクトのシーケンス番号を表します。

例えば、コネクターが Customer.in ファイルの中の最初の 4 つのビジネス・オブジェクトのうち 3 つを正常に処理し、2 つ目のビジネス・オブジェクトの処理に失敗したとします。さらに、コネクターが Customer.in の処理をまだ完了していないとします。この場合に event.log ファイルは、次のように表示されます。

Linux:

$***ProductDir***/JText/Event/Customer.in:: 1,3,4

Windows:

C:¥JText¥Event¥Customer.in:: 1,3,4

i5/OS:

/QIBM/UserData/WebBIICS/JText/Event/Customer.in:: 1,3,4

Customer.in ファイル全体の処理の途中でコネクターがダウンした場合は、コネクターは始動するときに、イベント・ファイルの処理を処理が停止したポイントから再開するためにログ・ファイルの情報を使用します。コネクターは、ログ・ファイルを参照して、リカバリーさせるイベント・ファイルの名前や最後に処理したビジネス・オブジェクトのシーケンス番号を取得します。次にコネクターは、ログ・ファイルに記録されている最後の番号より大きいシーケンス番号が付いたイベント・ファイル内のすべてのビジネス・オブジェクトを統合ブローカーに送信します。例

えば、上記のファイルの場合は、コネクターは Customer.in ファイルの 5 番目のビジネス・オブジェクトから処理を開始します。

コネクターはパフォーマンスを改善するためにログ・ファイルの内容をメモリーに保持します。そして、ログ・ファイルに新規エントリーがあり、更新するときにのみディスク上にあるこのファイルにアクセスします。コネクターはリカバリー時間にのみこのログ・ファイルを参照します。

コネクターがリカバリー処理でどのように event.log ファイルを使用するかについての情報は、『障害リカバリー』を参照してください。

障害リカバリー

注: 以下に示すリカバリーのための手順は、ディスクで障害が発生した場合やディスクがいっぱいの場合には適用しません。

イベント通知時に発生した障害からリカバリーするには、コネクターは以下の手順を行います。

1. コネクターはイベント・ファイルからビジネス・オブジェクト・ストリングを処理します。コネクターはエントリーを正常に処理したときにこのエントリーを event.log ファイルに記録します。また、このエントリーを (ArchiveDir メタオブジェクト属性で指定された) アーカイブ・ディレクトリー内のファイルにも書き込みます。
   - イベント・ファイル内のどのビジネス・オブジェクトの処理も失敗しなかった場合は、コネクターは正常に処理したビジネス・オブジェクトを、SuccessArchiveExt 属性で指定された拡張子が付いたアーカイブ・ファイルにアーカイブします。
   - イベント・ファイル内のビジネス・オブジェクトのいずれかの処理に失敗した場合は、コネクターは正常に処理したビジネス・オブジェクトを、PartialArchiveExt 属性で指定された拡張子が付いたアーカイブ・ファイルにアーカイブします。
   - コネクターは SuccessArchiveExt 属性で指定されたファイルにビジネス・オブジェクトを書き込んでから、処理に失敗したビジネス・オブジェクトがある場合はこのファイルの拡張子を、PartialArchiveExt で指定された拡張子に変更します。

   出荷時のこれらの拡張子のデフォルト値は .success および .partial です。
2. エラーが発生すると、コネクターは以下を行います。
   - サブスクリプション・エラー— コネクターはアーカイブ・ディレクトリー内に UnsubscribedArchiveExt メタオブジェクト属性で指定された拡張子が付いたアーカイブ・ファイルを作成します。引き渡されるこの拡張子のデフォルト値は .unsub です。
   - フォーマット・エラーまたは送信エラー— コネクターはアーカイブ・ディレクトリー内に、FailArchiveExt メタオブジェクト属性で指定された拡張子が付いたアーカイブ・ファイルを作成します。出荷時のこの拡張子のデフォルト値は .fail です。
   - ビジネス・オブジェクト区切り文字エラー— コネクターはアーカイブ・ディレクトリー内に、FailArchiveExt 属性で指定された拡張子が付いたアーカイ

ブ・ファイルを作成します。さらにコネクターは、イベント・ファイルをバックアップするために、アーカイブ・ディレクトリーに移動させて、拡張子を OriginalArchiveExt で指定されたものに変更します。

コネクターは処理に失敗したビジネス・オブジェクトを event.log に記録しません。

3. コネクターはすべてのビジネス・オブジェクトをイベント・ファイルに処理してから event.log ファイルを消去し、次のイベント・ファイルからそのファイルへの書き込みを開始します。
4. コネクターが、イベント・ファイル内のすべてのビジネス・オブジェクトを処理する前にダウンした場合は、リカバリー処理中に処理を開始する場所を判別するために event.log 内の情報を使用します。バックアップ時、コネクターはログ・ファイル内にエントリーがあるかどうかをチェックします。
   - エントリーがない場合は、コネクターはイベント・ファイル内のすべてのビジネス・オブジェクトを統合ブローカーに送信します。
   - エントリーがある場合は、コネクターはこの情報を、処理が停止したポイントからイベント・ファイルの処理を再開するために使用します。コネクターは、ログ・ファイルを参照して、リカバリーさせるイベント・ファイルの名前や最後に処理したビジネス・オブジェクトのシーケンス番号を取得します。次にコネクターは、ログ・ファイルに記録されている最後の番号より大きいシーケンス番号が付いたイベント・ファイル内のすべてのビジネス・オブジェクトを統合ブローカーに送信します。例えば、イベント・ファイル内に 15 のビジネス・オブジェクトがあり、ログ・ファイルに記録されている最後のシーケンス番号が 8 の場合に、コネクターは残りの 7 つのビジネス・オブジェクトを統合ブローカーに送信します。

     ログ・ファイルを使用することでコネクターが同じイベントを何度も統合ブローカーに送信することがなくなります。コネクターはパフォーマンスを改善するためにログ・ファイルの内容をメモリーに保持します。そして、ログ・ファイルに新規エントリーがあり、更新するときにのみディスク上にあるこのファイルにアクセスし、リカバリー時にのみログ・ファイルを読み取ります。

   ユーザーが 36 ページの『EventRecovery』構成プロパティーを retry に設定している場合は、コネクターは始動時に、前に処理したファイルから未解決のイベントを自動的にリカバリーします。ただし、このプロパティーを abort に設定している場合は、コネクターは、リカバリーさせるイベントがある場合に始動時に終了します。
5. イベント通知プロセスの間に発生したエラーからリカバリーするには、コネクターを再始動する必要があります。ただし、コネクターを再始動する前に以下を行ってください。
   - 処理に失敗した、およびアンサブスクライブされたビジネス・オブジェクト用にコネクターが作成したファイルを検査します。適切な修正を加えて、コネクターが始動した時にビジネス・オブジェクト・ストリングを正しく処理できるようにします。
   - 適切なファイルをアーカイブ・ディレクトリーからイベント・ディレクトリーにコピーして、すべての .fail または .unsub 拡張子を EventExt 属性で指定された拡張子 (デフォルトでは .in) に変更します。レコードの保持を容易

にするために、これらのファイルの名前をわかりやすく変更します。例えば、Customer.unsub を Customer_unsub_resubmit.in に変更します。

- 発生した障害のタイプに応じて、リカバリーに必要な追加の手順を手動で実行しなければならないことがあります。

以下のガイドラインは、ユーザーが、発生したエラーのタイプに応じて実行するリカバリーの手順を判別する際に役立ちます。

ビジネス・オブジェクトの区切り文字エラーからのリカバリー

コネクターがビジネス・オブジェクトをアーカイブ・ディレクトリーに書き込むときに、FailArchiveExt メタオブジェクト属性で指定された拡張子を付けます。このような障害のリカバリーを行うには、以下を行います。

1. イベント・ファイルにビジネス・オブジェクトの区切り文字が含まれ、この区切り文字に間違いがなく、それにデータ自体の区切り文字の値がテキストとして含まれないようにしてください。この区切り文字の使用法が正しくない場合は訂正してください。
2. 処理に失敗した他の理由を特定するには、コネクターのログ・ファイル (LogFileName 構成の属性で指定されている) を参照してください。
3. そのファイルをアーカイブ・ディレクトリーからイベント・ディレクトリーにコピーして、.fail 拡張子を EventExt 属性で指定された拡張子 (デフォルトでは .in) に変更します。レコードの保持を容易にするために、このファイルの名前をわかりやすく変更します。例えば、Customer.fail を Customer_delimiter_error.in に変更します。

サブスクリプション・エラーからのリカバリー

コネクターがビジネス・オブジェクトをアーカイブ・ディレクトリーに格納されているファイルに書き込むときに、UnsubscribedArchiveExt メタオブジェクト属性で指定された拡張子を付けます。このような障害のリカバリーを行うには、以下を行います。

1. アーカイブされたファイルを開き、そのビジネス・オブジェクト・ストリングを検出して、ビジネス・オブジェクト名と動詞がサブスクライブされていることを確認してください。必要があれば適切な修正を加えます。
2. 統合ブローカーが実行中であることを確認します。
3. そのファイルをアーカイブ・ディレクトリーからイベント・ディレクトリーにコピーして、.unsub 拡張子を EventExt 属性で指定された拡張子 (デフォルトでは .in) に変更します。レコードの保持を容易にするために、このファイルの名前をわかりやすく変更します。例えば、Customer.unsub を Customer_unsub_resubmit.in に変更します。

フォーマット・エラーからのリカバリー

コネクターがビジネス・オブジェクトをアーカイブ・ディレクトリーに格納されているファイルに書き込むときに、FailArchiveExt メタオブジェクト属性で指定された拡張子を付けます。このような障害のリカバリーを行うには、以下を行います。

1. アーカイブされたファイルを開き、以下を確認してください。

- ビジネス・オブジェクト・ストリングのフォーマットが、メタオブジェクトの予期されたフォーマットと一致すること。一致しない場合は、メタオブジェクトかビジネス・オブジェクト・ストリングのどちらか一方のフォーマット・タイプを変更してください。
- ビジネス・オブジェクト・ストリングのフォーマット構文が正しいこと。正しくない場合は修正してください。

2. そのファイルをアーカイブ・ディレクトリーからイベント・ディレクトリーにコピーして、.fail 拡張子を EventExt 属性で指定された拡張子 (デフォルトでは .in) に変更します。レコードの保持を容易にするために、このファイルの名前をわかりやすく変更します。例えば、Customer.fail を Customer_fail_formatting.in に変更します。

送信エラーからのリカバリー

コネクターがビジネス・オブジェクトをアーカイブ・ディレクトリーに格納されているファイルに書き込むときに、FailArchiveExt メタオブジェクト属性で指定された拡張子を付けます。このような障害のリカバリーを行うには、以下を行います。

1. ビジネス・インテグレーション・システムのすべてのコンポーネントが実行中であることを確認します。
2. そのファイルをアーカイブ・ディレクトリーからイベント・ディレクトリーにコピーして、.fail 拡張子を EventExt 属性で指定された拡張子 (デフォルトでは .in) に変更します。レコードの保持を容易にするために、このファイルの名前をわかりやすく変更します。例えば、Customer.fail を Customer_fail_sending.in に変更します。
3. コネクターを再始動します。

データ・ハンドラーとサポートされているビジネス・オブジェクト

データ・ハンドラーが構成されていないことを示すエラーをコネクターが戻すときは、データ・ハンドラーのメタオブジェクトがサポートされているビジネス・オブジェクトのリストに含まれていることを確認してください。コネクターが戻す最も一般的なエラーは、BOPrefix が設定されていないことを示すものです。

DHFormatter のサポートされているビジネス・オブジェクトのリストには、以下が含まれています。

- MO_JTextConnector_Default
- MO_JTextConnector_*BusObjName* (特定のビジネス・オブジェクト用に作成されたメタオブジェクト)
- ファイルから読み取られる、またはファイルに書き込まれるビジネス・オブジェクト
- データ・ハンドラーのメタオブジェクト (データ・ハンドラーは MO_JTextConnector_Default メタオブジェクトの DataHandlerConfigMO 属性で指定されたもの)

第 6 章 JText コネクターへのマイグレーションまたはアップグレード

本章では、5.3.0 バージョンの JText コネクターを 5.6.x へアップグレードする方法、および 3.2.0 バージョンの JText コネクターを から 4.0.x へアップグレードする方法について説明します。また、Text コネクターから JText コネクターへのアップグレード方法についても説明します。

注: JText コネクターの 4.4.x および 4.3.x のバージョンには、特定の構成変更はありません。これらのバージョンではオプションの構成変更がいくつかあるだけで、この新しいオプションを使用しない場合は、構成変更の必要はありません。詳細については、 vii ページの『本リリースの新機能』を参照してください。

この章には以下のトピックが含まれます。

- 『アップグレード・シナリオ』
- 『バージョン 5.3.x から 5.6.x へのアップグレード』
- 102 ページの『バージョン 3.2.0 から 4.0.x にアップグレードする理由』
- 103 ページの『バージョン 4.0.x へのアップグレード』
- 104 ページの『Text コネクターをアップグレードする理由』
- 105 ページの『JText コネクターへのアップグレード』

アップグレード・シナリオ

リリース 4.0.x の JText コネクターを 4.1.x へアップグレードする場合は、101 ページの『アップグレード・シナリオ』で説明する手順に従ってください。

リリース 3.2.0 の JText コネクターを 4.1.x へアップグレードする場合は、103 ページの『バージョン 4.0.x へのアップグレード』および 101 ページの『アップグレード・シナリオ』で説明する手順に従ってください。

Text コネクターから JText コネクターへのアップグレードの場合は、105 ページの『JText コネクターへのアップグレード』で説明する手順に従ってください。

バージョン 5.3.x から 5.6.x へのアップグレード

このセクションの内容は、以下のとおりです。

- 102 ページの『メタオブジェクトの変更点』
- 102 ページの『アーキテクチャーの変更点』

メタオブジェクトの変更点

MO_JTextConnector_Default メタ・オブジェクトには、6 つの新規属性 (FTPTransferType、FixedBOSize、DataProcessingMode、FTPDataStructure、MVSSiteCommand、および LargeObject) があり、これにより JText の機能性が拡張されています。

これらの属性を JText のトップレベル・メタオブジェクトに追加し、これらの値を構成する必要があります。Business Object Designer Express を使用して新規属性を追加し、そのデフォルト値を指定してから、定義に加えた変更を保管します。

この 6 つの新しい属性を MO_JTextConnector_Default を基にカスタマイズしたそれぞれのメタオブジェクトに追加します。例えば、Customer と Item という 2 つのビジネス・オブジェクト用にユーザー独自のメタオブジェクトを作成しているとすると、これらのメタオブジェクトに新しい属性を追加し、その属性にユーザー独自のデフォルト値を指定して、この変更をリポジトリーに保管します。

詳細については、 55 ページの表 8 を参照してください。

アーキテクチャーの変更点

このセクションの内容は、以下のとおりです。

- 『構成プロパティーの変更点』

構成プロパティーの変更点

5.6.0 リリースでは、新規のコネクター固有構成プロパティーが 2 つ追加されました。

- **SortFilesOnTimestamp** - アダプターには新規のコネクター固有プロパティーが追加されており、アダプターは、変更タイム・スタンプを基にしてイベント・ファイルを選出することができます。
- **NoPoll** - 新規のコネクター固有 boolean プロパティーが導入されました。オプションでポーリングをオフにできます。デフォルト値は false です。true に設定されている場合、アダプターは要求のみを処理し、ポーリングは処理しません。

バージョン 3.2.0 から 4.0.x にアップグレードする理由

バージョン 4.0.x の JText コネクターでは、コネクターの構成に必要なメタオブジェクトの構造が格段に単純化しています。したがって、構成プロセスも単純化しています。

以前のバージョンのコネクターが使用していたメタオブジェクトの構造は、3 階層と 10 個以上の異なるメタオブジェクトから構成されていたのに対して、バージョン 4.0.x の構造は 2 つのメタオブジェクトと 2 階層があるのみです。この新バージョンを利用すると、コネクターの構成方法は変わりますが、コネクターの機能は変わりません。

バージョン 4.0.x へのアップグレード

新しいメタオブジェクトでも以前のバージョンと同じ構成データが使用されるため、アップグレードで構成値を変更する必要はありません。ただし、新しいメタオブジェクトは以前より数がかなり少なくなったメタオブジェクトの中に違った名前の属性でデータを格納するため、アップグレードには以下の操作が必要です。

- メタオブジェクトを新規作成します。
- 新しいメタオブジェクトそれぞれの各属性の DefaultValue プロパティーの値を、既存のメタオブジェクト内のカスタマイズされたデフォルト値に置き換えます。
- 廃止となったメタオブジェクトすべてをリポジトリーから除去します。

IBM WBIS サポートには、上記の操作を自動で行うユーティリティーがあります。これらの操作を手動で実行する場合は、以下の手順を実行してください。

1. repos_copy ユーティリティーを使用して、リポジトリーのバックアップを作成します。例えば、以下のコマンドでは、Server1 リポジトリーの内容全体のバックアップを作成して、出力ファイル InterChangeRepository.out にコピーします。

   ```
   repos_copy -oInterChangeRepository.out -sServer1 -pmypassword
   ```

2. 既存の各トップレベル・メタオブジェクトに対して、新しく提供されたトップレベル・メタオブジェクト MO_JTextConnector_Default と同じ属性を持つメタオブジェクトを新規に作成します。例えば、以前の命名規則に従って名前を付けた Customer ビジネス・オブジェクト用のメタオブジェクト (MO_JTextConnector_Customer) を独自に生成していたとすると、新しい命名規則に従って Customer に対応する新しいメタオブジェクト (MO_JTextConnector_Customer) を作成します。
3. 新しいメタオブジェクトのデフォルト値を、元のメタオブジェクトの値に基づいて設定します。元のメタオブジェクトの属性と新しいメタオブジェクトの属性との対応については、表 12 を参照してください。
4. System Manager を使用して、元のメタオブジェクトの定義セットをリポジトリーから削除します。今回作成したメタオブジェクトと MO_JTextConnector_Default_DHFormatter だけを保持します。

表 12 に、元の属性と新しい属性との対応を、元のメタオブジェクトの名前と共に示します。元の属性を持つのが複数のメタオブジェクトであったのに対して、MO_JTextConnector_Default メタオブジェクトが新しい属性すべてを持ちます。

表 12. 元のメタオブジェクトおよび属性と新しい属性との対応

元のメタオブジェクトの名前	元の属性名	新しい属性名
MO_JTextConnector_*BOName*_Connector	DummyKey	該当なし
MO_JTextConnector_*BOName*_ArchiveDir	ディレクトリー	ArchiveDir
MO_JTextConnector_*BOName*_ArchiveFileExt	Success	SuccessArchiveExt
		PartialArchiveExt
	Fail	FailArchiveExt
		UnsubscribedArchiveExt
MO_JTextConnector_*BOName*_EventDir	ディレクトリー	EventDir
	FileExt	EventExt

表 12. 元のメタオブジェクトおよび属性と新しい属性との対応 (続き)

元のメタオブジェクトの名前	元の属性名	新しい属性名
MO_JTextConnector_*BOName*_OutputDir	ディレクトリー	OutputDir
	FileExt	OutputExt
	FileSequencingEnabled	FileSeqEnabled
MO_JTextConnector_*BOName*_FormatType		該当なし
MO_JTextConnector_*BOName*_ServicePolicy	OutputFileName	OutputFileName
	EndBODelimiter	EndBODelimiter
MO_JTextConnector_*BOName*_FormatService	EventService	EventFormat
	OutputService	OutputFormat

103 ページの表 12 には、以下の情報は含まれていません。

- `MO_JTextConnector_BOname_FormatType` メタオブジェクトに対応する属性

  以前のバージョンのコネクターでは、`MO_JTextConnector_BOname_FormatType` メタオブジェクトに可能なフォーマットをすべてリストしてから、イベント・ファイルや出力ファイルに使用するフォーマットを構成する必要がありました。新しいメタオブジェクトの構造では、イベント・ファイルや出力ファイルに使用できるようにフォーマットを構成するだけで済みます。 103 ページの表 12 で、`MO_JTextConnector_BOname_FormatType` メタオブジェクトに対応する属性がないことが、この変更を示しています。
- 個々のフォーマッター用のメタオブジェクト

  トップレベル・メタオブジェクトは、フォーマッター・メタオブジェクトを収容する 2 つの属性を持っています。フォーマッター・メタオブジェクトは、元のメタオブジェクトの構造と同じ属性を持ち、同じように使用されます。4 つのフォーマッターのうちの 3 つが使用すべきでないものとなったので、該当するフォーマッター・メタオブジェクトは `MO_JTextConnector_BOname__DHFormatter` のみです。

Text コネクターをアップグレードする理由

現在使用している Text コネクターによりアプリケーションと統合ブローカーの間の通信を処理している場合には、次の理由から JText コネクターへのアップグレードを検討してください。

- パフォーマンス。Text コネクターは、1 度に 1 つのファイルしか処理しません。このため、大容量のファイルや多数のファイルの処理時にパフォーマンスの妨げになる恐れがあります。
- フォーマットの可用性。Text コネクターは、少数のフォーマット・タイプしか処理しません。
- 変更の容易さ。Text コネクターは、JText コネクターに比べて変更が容易ではありません。

これに対して、JText コネクターは以下のように構成することができます。

- 1 度に複数のファイルを処理する。
- 特定のビジネス・オブジェクトを複数の場所で検索するため、パフォーマンスが向上する。

- 幅広いフォーマット・タイプに対応する。

JText コネクターへのアップグレード

Text コネクターから JText コネクターへアップグレードする場合は、以下のようにします。

1. 製品 CD から、JText ディレクトリーを %CROSSWORLD%connectors ディレクトリーにコピーします。
2. 「コマンド・プロンプト」ウィンドウを開き、repos_copy を使用して、CN_JText.txt と MO_JTextConnector_Default.txt の 2 つのファイルをリポジトリーに追加します。
3. ArchiveDir、EventDir、および OutputDir のメタオブジェクト属性用の指定のディレクトリーが作成されたことを確認します。これらのディレクトリーが作成されなかった場合は、新規に作成してください。
4. メタオブジェクトの属性を構成します。
5. 必要なビジネス・オブジェクトにサブスクライブします。

付録. 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 Server Express Adapter のコネクター・コンポーネントの標準構成プロパティーについて説明します。説明は、InterChange Server Express が対象となります。

このコネクター固有のプロパティーの詳細については、本書の該当するセクションを参照してください。

新規プロパティー

以下の標準プロパティーは、本リリースで追加されました。

- AdapterHelpName
- BiDi.Application
- BiDi.Broker
- BiDi.Metadata
- BiDi.Transformation
- ControllerEventSequencing
- jms.ListenerConcurrency
- jms.TransportOptimized
- TivoliTransactionMonitorPerformance

標準コネクター・プロパティーの概要

コネクターには、2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー。フレームワークが使用します。
- アプリケーション固有またはコネクター固有の構成プロパティー。エージェントが使用します。

これらのプロパティーは、アダプターのフレームワークおよびエージェントの実行時の振る舞いを決定します。

このセクションでは、Connector Configurator Express の始動方法について説明し、すべてのプロパティーに共通する特性について説明します。コネクター固有の構成プロパティーについては、該当するアダプターのユーザーズ・ガイドを参照してください。

Connector Configurator Express の始動

コネクター・プロパティーの構成は Connector Configurator Express から行います。Connector Configurator Express には、System Manager からアクセスします。Connector Configurator Express の使用法の詳細については、本書の Connector Configurator Express に関するセクションを参照してください。

Connector Configurator Express と System Manager は、 Windows システム上でのみ動作します。コネクターを Linux システム上で稼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。

Linux 上で動作するコネクターのコネクター・プロパティーを設定する場合は、Windows マシン上で System Manager を起動し、Linux の統合ブローカーに接続してから、コネクター用の Connector Configurator Express を開く必要があります。

構成プロパティー値の概要

コネクターは、以下の順序に従ってプロパティーの値を決定します。

1. デフォルト
2. InterChange Server Express 統合ブローカー用のリポジトリー
3. ローカル構成ファイル
4. コマンド行

プロパティー・フィールドのデフォルトの長さは 255 文字です。STRING プロパティー・タイプの長さに制限はありません。INTEGER タイプの長さは、アダプターを実行しているサーバーによって決まります。

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

プロパティーの更新特性 (すなわちコネクター・プロパティーへの変更を有効にする方法とタイミング) は、プロパティーの性質によって異なります。

標準コネクター・プロパティーには、以下の 4 種類の更新メソッドがあります。

- **動的**
  変更を System Manager に保管すると、新しい値が即時に有効になります。ただし、コネクターがスタンドアロン・モードの場合 (System Manager に依存しない) です。
- **エージェント再始動 (InterChange Server Express のみ)**
  コネクター・エージェントを停止して再始動しなければ、新規の値が有効になりません。
- **コンポーネント再始動**
  System Manager でコネクターを停止してから再始動しなければ、新しい値が有効になりません。エージェントまたはサーバー・プロセスを停止して再始動する必要はありません。
- **システム再始動**
  コネクター・エージェントおよびサーバーを停止して再始動しなければ、新規の値が有効になりません。

特定のプロパティーの更新方法を確認するには、「Connector Configurator Express」ウィンドウ内の「**更新メソッド**」列を参照するか、109 ページの表 13 の「更新メソッド」列を参照してください。

標準プロパティーが存在できる場所が 3 箇所あります。一部のプロパティーは複数の場所にあってもかまいません。

- **ReposController**
  このプロパティーはコネクター・コントローラー内にあり、その場所でのみ有効です。エージェント・サイドで値を変更した場合、コントローラーには影響しません。
- **ReposAgent**
  このプロパティーはエージェント内にあり、その場所でのみ有効です。プロパティーによっては、ローカル構成によってこの値をオーバーライドされることがあります。
- **LocalConfig**
  このプロパティーは、コネクターの構成ファイル内にあり、構成ファイルを通じてのみ機能することができます。コントローラーはこのプロパティーの値を変更することができず、システムが再配置されてコントローラーが明示的に更新されなければ、構成ファイルに加えられた変更を認識しません。

標準プロパティーの早見表

表 13 は、標準コネクター構成プロパティーの早見表です。すべてのコネクターでこれらのプロパティーすべてを必要とするわけではなく、プロパティー設定は異なる場合があります。

各プロパティーの説明については、表の次のセクションを参照してください。

注: 表 13 の注の欄で、「RepositoryDirectory が <REMOTE> に設定され」という句は、ブローカーが InterChange Server Express であることを示します。

表 13. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdapterHelpName	有効な <Regional Setting> ディレクトリーを含む <ProductDir>¥bin¥Data¥App¥Help 内の有効なサブディレクトリーのいずれか	テンプレート名 (有効な場合) またはブランク・フィールド	コンポーネント再始動	サポートされる地域設定。chs_chn、cht_twn、deu_deu、esn_esp、fra_fra、ita_ita、jpn_jpn、kor_kor、ptb_bra、および enu_usa (デフォルト) を含む。
AdminInQueue	有効な JMS キュー名	*<CONNECTORNAME>*/ADMININQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
AdminOutQueue	有効な JMS キュー名	*<CONNECTORNAME>*/ADMINOUTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。

表 13. 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
AgentConnections	1 から 4	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ または IDL であり、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
AgentTraceLevel	0 から 5	0	ICS では動的、その他の場合はコンポーネント再始動	
ApplicationName	アプリケーション名	コネクターのアプリケーション名に指定された値	コンポーネント再始動	
BiDi.Application	以下の双方向属性の任意の有効な組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント再始動	このプロパティーは、BiDi.Transformation の値が true の場合のみ有効です。
BiDi.Broker	以下の双方向属性の任意の有効な組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント再始動	このプロパティーは、BiDi.Transformation の値が true の場合のみ有効です。BrokerType の値が ICS の場合、プロパティーは読み取り専用です。
BiDi.Metadata	以下の双方向属性の任意の有効な組み合わせ 最初の文字: I、V 2 番目の文字: L、R 3 番目の文字: Y、N 4 番目の文字: S、N 5 番目の文字: H、C、N	ILYNN (5 文字)	コンポーネント再始動	このプロパティーは、BiDi.Transformation の値が true の場合のみ有効です。
BiDi.Transformation	true または false	false	コンポーネント再始動	このプロパティーは、BrokerType の値が WAS でない場合のみ有効です。
BrokerType	ICS	ICS	コンポーネント再始動	

表 13. 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
CharacterEncoding	サポートされる任意のコード。次のリストはその一部です。 ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437 .	ascii7	コンポーネント再始動	このプロパティーは、C++ コネクターでのみ有効です。
CommonEventInfrastructure	true または false	false	コンポーネント再始動	
CommonEventInfrastructureURL	URL ストリング。例えば、corbaloc:iiop:host:2809。	デフォルト値はありません。	コンポーネント再始動	このプロパティーは、CommonEvent Infrastructure の値が true の場合のみ有効です。
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定されて、BrokerType の値が ICS の場合にのみ有効です。
ContainerManagedEvents	ブランクまたは JMS	ブランク	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
ControllerEventSequencing	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ControllerStoreAndForwardMode	true または false	true	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ControllerTraceLevel	0 から 5	0	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定されて、BrokerType の値が ICS の場合にのみ有効です。
DeliveryQueue	任意の有効な JMS キュー名	<*CONNECTORNAME*>/DELIVERYQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
DeliveryTransport	IDL または JMS	RepositoryDirectory の値が <REMOTE> の場合は IDL。それ以外の場合は JMS。	コンポーネント再始動	RepositoryDirectory の値が <REMOTE> ではない場合、このプロパティーの有効な値は JMS のみです。

表 13. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	このプロパティーは、BrokerType の値が ICS である場合のみ有効です。
FaultQueue	任意の有効なキュー名。	*<CONNECTORNAME>*/FAULTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory、CxCommon.Messaging.jms.SonicMQFactory、または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.ListenerConcurrency	1 から 32767	1	コンポーネント再始動	このプロパティーは、jms.TransportOptimized の値が true の場合にのみ有効です。
jms.MessageBrokerName	jms.FactoryClassName の値が IBM の場合は、crossworlds.queue.manager を使用します。	crossworlds.queue.manager	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.Password	任意の有効なパスワード		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
jms.TransportOptimized	true または false	false	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS で、BrokerType の値が ICS である場合にのみ有効です。
jms.UserName	任意の有効な名前		コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合のみ有効です。
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。

表 13. 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
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
ListenerConcurrency	1 から 100	1	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が MQ の場合のみ有効です。
Locale	これは、サポートされるロケールの一部です。 en_US、ja_JP、ko_KR、 zh_CN、zh_TW、fr_FR、 de_DE、it_IT、 es_ES、pt_BR	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
MaxEventCapacity	1 から 2147483647	2147483647	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
MessageFileName	有効なファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	<*CONNECTORNAME*> /MONITORQUEUE	コンポーネント再始動	このプロパティーは、DuplicateEventElimination の値が true で、ContainerManagedEvents に値がない場合にのみ有効です。
OADAutoRestartAgent	true または false	false	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。

表 13. 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
OADMaxNumRetry	正整数	1000	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
OADRetryTimeInterval	正整数 (単位: 分)	10	動的	このプロパティーは、RepositoryDirectory の値が <REMOTE> に設定され、BrokerType の値が ICS である場合のみ有効です。
PollEndTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒)	10000	ブローカーが ICS の場合は動的。そうでない場合は、コンポーネント再始動。	
PollQuantity	1 から 500	1	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS の場合のみ有効です。
PollStartTime	HH = 0 から 23 MM = 0 から 59	HH:MM	コンポーネント再始動	
RepositoryDirectory	ブローカーが ICS の場合は <REMOTE>。それ以外の場合は任意の有効なローカル・ディレクトリー。	ICS の場合、値は <REMOTE> に設定されます。	エージェント再始動	
RequestQueue	有効な JMS キュー名	*<CONNECTORNAME>*/REQUESTQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
ResponseQueue	有効な JMS キュー名	*<CONNECTORNAME>*/RESPONSEQUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
RestartRetryCount	0 から 99	3	ICS の場合は動的、その他の場合はコンポーネント再始動	
RestartRetryInterval	1 から 2147483647 までの値 (分単位)。	1	ICS の場合は動的、その他の場合はコンポーネント再始動	

表 13. 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
RHF2MessageDomain	mrmまたは xml	mrm	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS であり、WireFormat の値が CwXML である場合のみ有効です。
SourceQueue	任意の有効な WebSphere MQ キュー名	<*CONNECTORNAME*> /SOURCEQUEUE	エージェント再始動	このプロパティーは、ContainerManagedEvents の値が JMS の場合のみ有効です。
SynchronousRequest Queue	任意の有効なキュー名。	<*CONNECTORNAME*> /SYNCHRONOUSREQUEST QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
SynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
SynchronousResponse Queue	任意の有効なキュー名	<*CONNECTORNAME*> /SYNCHRONOUSRESPONSE QUEUE	コンポーネント再始動	このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。
TivoliMonitorTransaction パフォーマンス	true または false	false	コンポーネント再始動	
WireFormat	CwXMLまたは CwBO	CwXML	エージェント再始動	RepositoryDirectory の値が <REMOTE> に設定されていない場合、このプロパティーの値は、CwXML でなければなりません。RepositoryDirectory の値が <REMOTE> に設定されている場合、値は CwBO でなければなりません。
WsifSynchronousRequest Timeout	0 から任意の数 (ミリ秒)	0	コンポーネント再始動	BrokerType の値が ICS の場合、このプロパティーは無効です。
XMLNameSpaceFormat	shortまたは long	short	エージェント再始動	BrokerType の値が ICS の場合、このプロパティーは無効です。

標準プロパティー

このセクションでは、標準コネクター構成プロパティーについて説明します。

AdapterHelpName

AdapterHelpName プロパティーは、コネクター固有の全般ヘルプ・ファイルがあるディレクトリーの名前です。ディレクトリーは、<*ProductDir*>¥bin¥Data¥App¥Help

内に配置される必要があり、少なくとも言語ディレクトリー enu_usa が含まれていなければなりません。ロケールに応じて、その他のディレクトリーが含まれることがあります。

デフォルト値は、テンプレート名が有効であればテンプレート名、有効でなければブランクです。

AdminInQueue

AdminInQueue プロパティーは、統合ブローカーがコネクターへ管理メッセージを送信するときに使用するキューを指定します。

デフォルト値は *<CONNECTORNAME>*/ADMININQUEUE です。

AdminOutQueue

AdminOutQueue プロパティーは、コネクターから統合ブローカーへ管理メッセージが送信されるときに使用されるキューを指定します。

デフォルト値は *<CONNECTORNAME>*/ADMINOUTQUEUE です。

AgentConnections

AgentConnections プロパティーは、ORB (オブジェクト・リクエスト・ブローカー) が初期化するときに開かれる ORB 接続の数を制御します。

このプロパティーのデフォルト値は 1 です。

AgentTraceLevel

AgentTraceLevel プロパティーは、アプリケーション固有のコンポーネントのトレース・メッセージのレベルを設定します。コネクターは、設定されたトレース・レベル以下の該当するトレース・メッセージをすべてデリバリーします。

デフォルト値は 0 です。

ApplicationName

ApplicationName プロパティーは、コネクター・アプリケーションの名前を一意的に識別します。この名前は、システム管理者が統合環境をモニターするために使用します。コネクターを実行する前に、このプロパティーに値を指定する必要があります。

デフォルトはコネクターの名前です。

BiDi.Application

BiDi.Application プロパティーは、このアダプターがサポートする任意のビジネス・オブジェクトの形式で、外部アプリケーションからアダプターに入ってくるデータの双方向フォーマットを指定します。このプロパティーは、アプリケーション・データの双方向属性を定義します。これらの属性は以下のとおりです。

- テキストのタイプ: 暗黙または可視 (I または V)
- テキストの方向: 左から右または右から左 (L または R)

- 対称スワッピング: オンまたはオフ (Y または N)
- 成形 (アラビア語): オンまたはオフ (S または N)
- 数字成形 (アラビア語): ヒンディ語、コンテキスト、または名詞 (H、C、または N)

このプロパティーは、BiDi.Transformation プロパティーの値が true に設定されている場合のみ有効です。

デフォルト値は ILYNN (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Broker

BiDi.Broker プロパティーは、サポートされる任意のビジネス・オブジェクトの形式で、アダプターから統合ブローカーに送信されるデータの双方向フォーマットを指定します。データの双方向属性を定義します。属性は、前述の BiDi.Application の下にリストされています。

このプロパティーは、BiDi.Transformation プロパティーの値が true に設定されている場合のみ有効です。BrokerType プロパティーが ICS の場合、プロパティー値は読み取り専用です。

デフォルト値は ILYNN (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Metadata

BiDi.Metadata プロパティーは、メタデータの双方向フォーマットまたは属性を定義します。メタデータは、外部アプリケーションへのリンクを確立および保守するために、コネクターが使用します。属性の設定は、双方向機能を使用する各アダプターに固有です。アダプターが双方向処理をサポートする場合、詳細についてはアダプター固有のプロパティーに関するセクションを参照してください。

このプロパティーは、BiDi.Transformation プロパティーの値が true に設定されている場合のみ有効です。

デフォルト値は ILYNN (暗黙、左から右、オン、オフ、名詞) です。

BiDi.Transformation

BiDi.Transformation プロパティーは、システムが実行時に双方向変換を実行するかどうかを定義します。

プロパティー値が true に設定されている場合、BiDi.Application、BiDi.Broker、および BiDi.Metadata プロパティーが使用可能です。プロパティー値が false に設定されている場合は、それらは非表示になります。

デフォルト値は false です。

BrokerType

BrokerType プロパティーは、使用している統合ブローカーのタイプを識別します。値は、ICS (InterChange Server Express) です。

CharacterEncoding

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

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

デフォルトでは、サポートされる文字エンコードの一部のみが表示されます。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 (*<ProductDir>*) にある ¥Data¥Std¥stdConnProps.xml ファイルを手動で変更する必要があります。詳細については、本書の付録『Connector Configurator Express』を参照してください。

ConcurrentEventTriggeredFlows

ConcurrentEventTriggeredFlows プロパティーは、コネクターがイベントのデリバリー時に並行処理できるビジネス・オブジェクトの数を決定します。この属性の値を、並行してマップおよびデリバリーされるビジネス・オブジェクトの数に設定します。例えば、このプロパティーの値を `5` に設定すると、5 個のビジネス・オブジェクトが並行して処理されます。

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

ソース・アプリケーションから宛先アプリケーションまでのフロー全体に並行処理を実装するには、以下のプロパティーを構成する必要があります。

- `Maximum number of concurrent events` プロパティーの値を増加して、複数のスレッドを使用できるようにコラボレーションを構成する必要があります。
- 宛先アプリケーションのアプリケーション固有コンポーネントを、複数の要求を並行して処理できるように構成する必要があります。

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

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は `1` です。

ContainerManagedEvents

ContainerManagedEvents プロパティーにより、JMS イベント・ストアを使用する JMS 対応コネクターが、保証付きイベント・デリバリーを提供できるようになります。保証付きイベント・デリバリーでは、イベントはソース・キューから除去され、1 つの JMS トランザクションとして宛先キューに配置されます。

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

- PollQuantity = 1 から 500
- SourceQueue = /SOURCEQUEUE

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

これらのプロパティーはアダプター固有ですが、以下に値の例をいくつか示します。

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

「**データ・ハンドラー**」 タブのこれらの値のフィールドは、ContainerManagedEvents プロパティーを JMS という値に設定した場合にのみ表示されます。

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

ContainerManagedEvents プロパティーは、DeliveryTransport プロパティーの値が JMS に設定されている場合のみ有効です。

デフォルト値はありません。

ControllerEventSequencing

ControllerEventSequencing プロパティーは、コネクター・コントローラーでイベント順序付けを使用可能にします。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType は ICS) のみ有効です。

デフォルト値は true です。

ControllerStoreAndForwardMode

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

このプロパティーを true に設定した場合、イベントが InterChange Server Express (ICS) に到達したときに宛先側のアプリケーション固有のコンポーネントが使用不可であれば、コネクター・コントローラーはそのアプリケーション固有のコンポーネ

ントへの要求をブロックします。アプリケーション固有のコンポーネントが作動可能になると、コネクター・コントローラーはアプリケーション固有のコンポーネントにその要求を転送します。

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

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

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType プロパティーの値が ICS) のみ有効です。

デフォルト値は `true` です。

ControllerTraceLevel

ControllerTraceLevel プロパティーは、コネクター・コントローラーのトレース・メッセージのレベルを設定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は `0` です。

DeliveryQueue

DeliveryQueue プロパティーは、コネクターが統合ブローカーへビジネス・オブジェクトを送信するときに使用するキューを定義します。

このプロパティーは、DeliveryTransport プロパティーの値が `JMS` に設定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*`/DELIVERYQUEUE` です。

DeliveryTransport

DeliveryTransport プロパティーは、イベントのデリバリーのためのトランスポート機構を指定します。Java Messaging Service の場合、値は JMS です。

- RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合、DeliveryTransport プロパティーの値には `IDL` または `JMS` を指定することができ、デフォルトは `IDL` です。
- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合、値に使用できるのは `JMS` のみです。

RepositoryDirectory プロパティーの値が `IDL` である場合、コネクターは、CORBA IIOP を使用してサービス呼び出し要求と管理メッセージを送信します。

デフォルト値は JMS です。

JMS

JMS トランスポート機構は、Java Messaging Service (JMS) を使用した、コネクターとクライアント・コネクター・フレームワークとの間の通信を可能にします。

JMS をデリバリー・トランスポートとして選択した場合は、jms.MessageBrokerName、jms.FactoryClassName、jms.Password、jms.UserName などの追加の JMS プロパティーが Connector Configurator Express 内にリストされます。jms.MessageBrokerName プロパティーおよび jms.FactoryClassName プロパティーは、このトランスポートの必須プロパティーです。

InterChange Server Espress (ICS) が統合ブローカーである場合、以下の環境では、コネクターに JMS トランスポート機構を使用すると、メモリー制限が発生することもあります。

この環境では、WebSphere MQ クライアント内でメモリーが使用されるため、(サーバー側の) 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 768MB 未満である場合には、次の変数およびプロパティーを設定してください。

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

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

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

  この行は、ヒープ・メモリーの使用量を最大 768 MB (3 セグメント * 256 MB) に制限します。プロセス・メモリーがこの制限値を超えると、ページ・スワッピングが発生し、システムのパフォーマンスに悪影響を与える場合があります。
- IPCCBaseAddress プロパティーの値を 11 または 12 に設定する。このプロパティーの詳細については、「*WebSphere Business Integration Server Express インストール・ガイド*」(Windows 版、Linux 版、または OS/400 および i5/OS 版) を参照してください。

DuplicateEventElimination

このプロパティーの値が true の場合、JMS 対応コネクターでは重複イベントがデリバリー・キューへデリバリーされないようにすることができます。この機能を使用するには、コネクター開発時に、コネクターに対し、アプリケーション固有のコード内でビジネス・オブジェクトの ObjectEventId 属性として一意のイベント ID が設定されている必要があります。

注: このプロパティーの値が true の場合、保証付きイベント・デリバリーを提供するには、MonitorQueue プロパティーを使用可能にする必要があります。

デフォルト値は false です。

EnableOidForFlowMonitoring

このプロパティーの値が `true` の場合、アダプター・ランタイムは、着信 ObjectEventID にフロー・モニターの外部キーのマークを付けます。

このプロパティーは、BrokerType プロパティーが ICS に設定されている場合のみ有効です。

デフォルト値は `false` です。

FaultQueue

コネクターでメッセージを処理中にエラーが発生すると、コネクターは、そのメッセージ (および状況標識と問題説明) を FaultQueue プロパティーで指定されているキューに移動します。

デフォルト値は *<CONNECTORNAME>*`/FAULTQUEUE` です。

jms.FactoryClassName

jms.FactoryClassName プロパティーは、JMS プロバイダーのためにインスタンスを生成するクラス名を指定します。DeliveryTransport プロパティーの値が `JMS` に設定されている場合、このプロパティーを設定する必要があります。

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

jms.ListenerConcurrency

jms.ListenerConcurrency プロパティーは、JMS コントローラーの並行リスナーの数を指定します。コントローラー内部で、並行してメッセージを取り出して処理するスレッドの数を指定します。

このプロパティーは、jms.OptimizedTransport プロパティーの値が `true` の場合のみ有効です。

デフォルト値は `1` です。

jms.MessageBrokerName

jms.MessageBrokerName は、JMS プロバイダーのために使用するブローカー名を指定します。`JMS` をデリバリー・トランスポート機構として (DeliveryTransport プロパティーで) 指定する場合、このコネクター・プロパティーを設定する必要があります。

リモート・メッセージ・ブローカーに接続した場合、このプロパティーでは以下の値を指定する必要があります。
QueueMgrName`:`*Channel*`:`*HostName*`:`*PortNumber*
ここで、以下のように説明されます。
QueueMgrName は、キュー・マネージャー名です。
Channel は、クライアントが使用するチャネルです。
HostName は、キュー・マネージャーの配置先のマシン名です。
PortNumber は、キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のようになります。
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456

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

jms.NumConcurrentRequests

jms.NumConcurrentRequests プロパティーは、コネクターに対して同時に送信することができる並行サービス呼び出し要求の数 (最大値) を指定します。この最大値に達した場合、新規のサービス呼び出しはブロックされ、処理を続行するには他のいずれかの要求が完了するのを待機する必要があります。

デフォルト値は 10 です。

jms.Password

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

デフォルト値はありません。

jms.TransportOptimized

jms.TransportOptimized プロパティーは、WIP (処理中の作業) が最適化されるかどうかを決定します。WIP を最適化するには、WebSphere MQ プロバイダーが必要です。最適化された WIP が作動するためには、メッセージング・プロバイダーが以下の操作を実行できなければなりません。

1. メッセージをキューから削除せずに読み取る。
2. メッセージ全体を受信側のメモリー空間に転送することなく、固有の ID を使用してメッセージを削除する。
3. 固有の ID を使用してメッセージを読み取る (リカバリーのために必要)。
4. 読み取られなかったイベントが現れるポイントを追跡する。

JMS API は、上記の条件 2 および 4 を満たさないため、最適化された WIP には使用できませんが、MQ Java API は 4 つの条件をすべて満たすため、最適化された WIP には必要です。

このプロパティーは、DeliveryTransport の値が JMS で、BrokerType の値が ICS である場合にのみ有効です。

デフォルト値は false です。

jms.UserName

jms.UserName プロパティーは、JMS プロバイダーの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。

デフォルト値はありません。

JvmMaxHeapSize

JvmMaxHeapSize プロパティーは、エージェントの最大ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128M です。

JvmMaxNativeStackSize

JvmMaxNativeStackSize プロパティーは、エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 128K です。

JvmMinHeapSize

JvmMinHeapSize プロパティーは、エージェントの最小ヒープ・サイズ (メガバイト単位) を指定します。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合のみ有効です。

デフォルト値は 1M です。

ListenerConcurrency

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

このプロパティーは、MQ トランスポートを使用するコネクターのみで有効です。DeliveryTransport プロパティーの値は `MQ` でなければなりません。

デフォルト値は 1 です。

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。
ll は、2 文字の言語コード (小文字を使用) です。

TT は、2 文字の国または地域コード (大文字を使用) です。
codeset は、関連文字コード・セットの名前です (オプションの場合があります)。

デフォルトでは、サポートされるロケールの一部のみがリストされます。サポートされる他の値をリストに追加するには、<*ProductDir*>¥bin ディレクトリーにある ¥Data¥Std¥stdConnProps.xml ファイルを変更します。詳細については、本書の付録『Connector Configurator Express』を参照してください。

コネクターが国際化に対応していない場合、このプロパティーの有効な値は en_US のみです。特定のコネクターがグローバル化に対応しているかどうかを判別するには、そのアダプターのユーザーズ・ガイドを参照してください。

デフォルト値は en_US です。

LogAtInterchangeEnd

LogAtInterchangeEnd プロパティーは、統合ブローカーのログ宛先にエラーを記録するかどうかを指定します。

ログ宛先にログを記録すると、E メール通知もオンになります。これにより、エラーまたは致命的エラーが発生すると、InterchangeSystem.cfg ファイルで MESSAGE_RECIPIENT の値として指定された宛先に対する E メール・メッセージが生成されます。例えば、LogAtInterChangeEnd の値を true に設定した場合にコネクターからアプリケーションへの接続が失われると、指定されたメッセージ宛先に、E メール・メッセージが送信されます。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は false です。

MaxEventCapacity

MaxEventCapacity プロパティーは、コントローラー・バッファー内のイベントの最大数を指定します。このプロパティーは、フロー制御機能によって使用されます。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

値は 1 から 2147483647 の間の正整数です。

デフォルト値は 2147483647 です。

MessageFileName

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

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

注: コネクターについて、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。

デフォルト値は `InterchangeSystem.txt` です。

MonitorQueue

MonitorQueue プロパティーは、コネクターが重複イベントをモニターするために使用する論理キューを指定します。

このプロパティーは、DeliveryTransport プロパティーの値が `JMS` であり、DuplicateEventElimination の値が `true` である場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*`/MONITORQUEUE` です。

OADAutoRestartAgent

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

自動再始動機能およびリモート再始動機能を使用可能にするには、このプロパティーを `true` に設定する必要があります。WebSphere MQ により起動される OAD 機能の構成方法については、「*WebSphere Business Integration Server Express インストール・ガイド (Windows 版)*」、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」または「*WebSphere Business Integration Server Express インストール・ガイド (OS/400 および i5/OS 版)*」を参照してください。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は `false` です。

OADMaxNumRetry

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

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は `1000` です。

OADRetryTimeInterval

OADRetryTimeInterval プロパティーは、WebSphere MQ によりトリガーされる Object Activation Daemon (OAD) の再試行時間間隔の分数を指定します。コネクター・エージェントがこの再試行時間間隔内に再始動しない場合は、コネクター・コ

ントローラーはコネクター・エージェントを再び再始動するように OAD に要求します。OAD はこの再試行プロセスを OADMaxNumRetry プロパティーで指定された回数だけ繰り返します。このプロパティーを有効にするためには、OADAutoRestartAgent プロパティーを `true` に設定する必要があります。

このプロパティーは、RepositoryDirectory プロパティーの値が <REMOTE> に設定されている場合 (BrokerType の値が ICS) のみ有効です。

デフォルト値は `10` です。

PollEndTime

PollEndTime プロパティーは、イベント・キューのポーリングを停止する時刻を指定します。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

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

アダプター・ランタイムが以下のことを検出した場合、

- PollStartTime が設定されて、PollEndTime が設定されていない、または
- PollEndTime が設定されて、PollStartTime が設定されていない

PollFrequency プロパティーに構成された値を使用してポーリングします。

PollFrequency

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

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

このプロパティーでは、以下の値が有効です。

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

デフォルト値は `10000` です。

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

PollQuantity

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

このプロパティーは、`DeliveryTransport` プロパティーの値が `JMS` であり、`ContainerManagedEvents` プロパティーに値がある場合のみ有効です。

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、E メールに関するポーリングを受けたときには次のように動作します。

- 一度ポーリングされると、コネクターはメッセージの本文を検出し、それを添付ファイルとして読み取ります。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターはメッセージを無視します。
- コネクターは最初の BO 添付ファイルを処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- 二度目にポーリングされると、コネクターは 2 番目の BO 添付ファイルを処理します。この MIME タイプには対応するデータ・ハンドラーがあるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。
- それが受け入れられると、3 番目の BO 添付ファイルが送信されます。

PollStartTime

PollStartTime プロパティーは、イベント・キューのポーリングを開始する時刻を指定します。形式は *HH:MM* です。ここで、*HH* は 0 から 23 時を表し、*MM* は 0 から 59 分を表します。

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

アダプター・ランタイムが以下のことを検出した場合、

- PollStartTime が設定されて、PollEndTime が設定されていない、または
- PollEndTime が設定されて、PollStartTime が設定されていない

PollFrequency プロパティーに構成された値を使用してポーリングします。

RepositoryDirectory

RepositoryDirectory プロパティーは、コネクターが XML スキーマ文書を読み取るリポジトリーの場所です。この XML スキーマ文書には、ビジネス・オブジェクト定義のメタデータが保管されています。

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

統合ブローカーが WebSphere Message Broker または WAS の場合は、この値はデフォルトで <*ProductDir*>¥repository に設定されます。ただし、これには任意の有効なディレクトリー名を設定することができます。

RequestQueue

RequestQueue プロパティーは、統合ブローカーが、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を指定します。

このプロパティーは、DeliveryTransport プロパティーの値が JMS の場合のみ有効です。

デフォルト値は <*CONNECTORNAME*>/REQUESTQUEUE です。

ResponseQueue

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

このプロパティーは、DeliveryTransport プロパティーの値が JMS の場合のみ有効です。

デフォルト値は <*CONNECTORNAME*>/RESPONSEQUEUE です。

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

RestartRetryInterval プロパティーは、コネクターによるコネクター自体の再始動の試行間隔を分単位で指定します。このプロパティーを並列にリンクされたコネクターに対して使用する場合、コネクターのマスター側のアプリケーション固有のコンポーネントがクライアント側のアプリケーション固有のコンポーネントの再始動を試行する間隔が指定されます。

プロパティーに使用可能な値の範囲は 1 から 2147483647 です。

デフォルト値は 1 です。

RHF2MessageDomain

RHF2MessageDomain プロパティーにより、JMS ヘッダーのドメイン名フィールドの値を構成できます。JMS トランスポートを介してデータを WebSphere Message Broker に送信するときに、アダプター・フレームワークにより JMS ヘッダー情

報、ドメイン名、および固定値 mrm が書き込まれます。構成可能ドメイン名によって、WebSphere Message Broker がメッセージ・データを処理する方法を追跡できます。

ヘッダーの例を示します。

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

BrokerType の値が ICS の場合、このプロパティーは無効です。また、このプロパティーは、DeliveryTransport プロパティーの値が JMS であり、WireFormat プロパティーの値が CwXML である場合のみ有効です。

可能な値は、mrm および xml です。デフォルト値は mrm です。

SourceQueue

SourceQueue プロパティーは、JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、118 ページの『ContainerManagedEvents』を参照してください。

このプロパティーは、DeliveryTransport の値が JMS であり、ContainerManagedEvents の値が指定されている場合のみ有効です。

デフォルト値は *<CONNECTORNAME>*/SOURCEQUEUE です。

SynchronousRequestQueue

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

このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。

デフォルト値は *<CONNECTORNAME>*/SYNCHRONOUSREQUESTQUEUE です。

SynchronousRequestTimeout

SynchronousRequestTimeout プロパティーは、コネクターが同期要求への応答を待機する時間をミリ秒単位で指定します。コネクターは、指定された時間内に応答を受信できなかった場合、元の同期要求メッセージ (およびエラー・メッセージ) を障害キューに移動します。

このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。

デフォルト値は 0 です。

SynchronousResponseQueue

SynchronousResponseQueue プロパティーは、同期要求に対する応答メッセージを、ブローカーからコネクター・フレームワークにデリバリーします。このキューは、コネクターが同期実行を使用する場合にのみ必要です。

このプロパティーは、DeliveryTransport の値が JMS の場合にのみ有効です。

デフォルトは *<CONNECTORNAME>*/SYNCHRONOUSRESPONSEQUEUE です。

TivoliMonitorTransactionPerformance

TivoliMonitorTransactionPerformance プロパティーは、IBM Tivoli Monitoring for Transaction Performance (ITMTP) を実行時に起動するかどうかを指定します。

デフォルト値は false です。

WireFormat

WireFormat プロパティーは、トランスポートのメッセージ・フォーマットを指定します。

- RepositoryDirectory プロパティーの値がローカル・ディレクトリーの場合、値は CwXML です。
- RepositoryDirectory プロパティーの値がリモート・ディレクトリーの場合、値は CwBO です。

索引

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

[ア行]

アーキテクチャー 2
アダプター環境 17
アダプターをサポートするプラットフォーム 18
アダプター・コンポーネント 2
アップグレード 101
  アーキテクチャーの変更点 102
  シナリオ 101
  5.3.0 から 5.6.0 101
イベント通知 7
  指定 66
イベントの起動
  問題 94
イベントを処理
  構成操作 82
イベント・アーカイブ 9
  指定 67
イベント・ログ・ファイル 95
インストール 17, 19
  検証 19
  タスクの概要 17
  Linux での検証 20
  OS/400 および i5/OS での検証 20
  Windows での検証 19
  Windows ファイル構造 19, 20
エラー・タイプ 94
エラー・メッセージ
  ロギング 93

[カ行]

構成タスク 66
構成ファイル
  完成 31
  新規作成 29
  テンプレートからの作成 29
  プロパティーの設定 32
  変更 44
  保管 43
コネクター固有の構成プロパティー
  設定 34
コネクター固有のプロパティー 35
  ArchivingEnabled 35, 36
コネクター固有のプロパティー *(続き)*
  EventLog 35, 36
  EventRecovery 35, 36
  FTPPollFrequency 35, 36
  GenerateTemplate 35, 36
  NoPoll 38
  OutputLog 35, 37
  PollQuantity 36, 37
  SortFilesOnTimestamp 36, 37
コネクター固有のプロパティー・テンプレート
  作成 25
コネクター・インスタンス
  複数作成 47

[サ行]

サブスクリプション・エラー
  リカバリー 98
サポートされるプラットフォーム 18
サンプル・ファイル
  生成 90
出力ファイル
  構成 72
  名前の指定 64
障害リカバリー 96
新規テンプレート
  作成 26
静的なファイルの命名 70
セキュア FTP
  構成 85
前提条件 18
送信エラー
  リカバリー 99

[タ行]

データ・ハンドラー 2
  区切り文字での区切り 3
  サポートされるビジネス・オブジェクト 99
  指定 86
  処理 14
  変更 86
  FixedWidth 3
  NameValue 3
動的なファイルの命名 70
トラブルシューティング 93

[ハ行]

パフォーマンス
  調整 89
ビジネス・オブジェクト 3
  区切り文字エラー・リカバリー 98
  サポートの追加 50
  動詞処理の要求 15
ビジネス・オブジェクトのサンプル
  生成 91
標準構成プロパティー 107
標準コネクター・プロパティー
  設定 33
フォーマット・エラー
  リカバリー 98
複数のイベント・ディレクトリー
  指定 74
複数のイベント・ファイル
  指定 74
ポーリング
  固有のビジネス・オブジェクト 74
  振る舞いの構成 67

[マ行]

マイグレーション 101
メタオブジェクト 3
  アップグレードの変更点 102
  階層構造 53
  カスタムの作成 53
  構造 52
  固有のビジネス・オブジェクトの作成 87
  使用 51
  定義済み 51
  動的子の使用 4
  動的子の属性 5
  命名規則 52
  命名の問題 93

[ヤ行]

要求処理 12
  構成操作 84
  指定 69

[ラ行]

リモート FTP ファイル・システム
  指定 78

リモートのイベント処理　78
リモートの要求処理　83
リモート・アーカイブ
　指定　80
リモート・サイト
　イベントを処理　81
リモート・ポーリング
　指定　81
ローカル・アーカイブ
　構成　69
　ファイル名　68
ローカル・ディレクトリー
　指定　79
ロケール依存データ　18
　処理　16

C

Connector Configurator
　概要　23
　グローバル化環境　44
　始動　24
　スタンドアロン・モード　24
　System Manager、実行元　25
　UNIX　24

E

EndBODelimiter
　印刷不可能文字の使用　76
EndBODelimiter 属性
　使用　75
EndBODelimiter ベースの解析手法
　使用　75

F

FixedBOSize ベースの解析手法
　使用　78
FTP URL
　指定　79
FTP 転送用
　構成情報　84

J

JText アダプター
　アーキテクチャー　2
　アップグレード　101
　アップグレード 手順　105
　インストール　17, 19
　インストールの検証　19
　エラー・タイプ　94
　概要　1
　環境　17
JText アダプター *(続き)*
　機能　15
　構成　23
　コンポーネント　2
　始動　45
　障害の処理　94
　前提条件　18
　違い　16
　停止　47
　トラブルシューティング　93
　パフォーマンスの調整　89
　プラットフォーム　18
　マイグレーション　101
　2 番目のインスタンスの設定　88

M

MO_JTextConnector_Default
　属性　54

O

ObjectEventID 属性
　値の指定　88

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。 IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。 IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換および (ii) 交換された情報の相互利用を可能にすることを目的として本プログラムに関する情報を必要とする方は、下記に連絡してください。

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

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。 IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。 IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。著作権使用許諾: 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。汎用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

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

商標

以下は、IBM Corporation の商標です。
IBM
IBM ロゴ
AIX
CICS
CrossWorlds
DB2
DB2 Universal Database
i5/OS
IMS
Informix
iSeries
Lotus
Lotus Domino
Lotus Notes
MQIntegrator
MQSeries
MVS
OS/400
Passport Advantage
SupportPac
WebSphere
z/OS
Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。
MMX、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。
Linux は、Linus Torvalds の米国およびその他の国における商標です。
他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Server Express バージョン 4.4、および WebSphere Business Integration Server Express Plus バージョン 4.4

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12