

**WebSphere Business Integration Server
Express and Express Plus**



Adapter for JText ユーザーズ・ガイド

バージョン 4.3.1

**WebSphere Business Integration Server
Express and Express Plus**



Adapter for JText ユーザーズ・ガイド

バージョン 4.3.1

お願い

本書および本書で紹介する製品をご使用になる前に、117ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Server Express バージョン 4.3.1、および IBM WebSphere Integration Server Express Plus バージョン 4.3.1 に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原 典： WebSphere Business Integration Sever Express and Express Plus
Adapter for JText User Guide
Version 4.3.1

発 行： 日本アイ・ビー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2004.8

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

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

© Copyright International Business Machines Corporation 2000, 2003, 2004. All rights reserved.

© Copyright IBM Japan 2004

目次

本書について	v
対象読者	v
関連資料	v
表記上の規則	vi
本リリースの新機能	vii
リリース 4.3.1 の新機能	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 コネクタが使用するビジネス・オブジェクト	3
コネクタの動作	7
コネクタの機能	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	18
インストールの検証	18
第 3 章 JText アダプターの構成	23
コネクタの構成の概要	23
Connector Configurator Express の概要	24
Connector Configurator Express の始動	25
コネクタ固有のプロパティ・テンプレートの作成	26
新しい構成ファイルを作成	29
既存の構成ファイルの使用	30
構成ファイルの完成	31
構成ファイル・プロパティの設定	31
構成ファイルの保管	40
構成の完了	41
グローバル化環境における Connector Configurator Express の使用	41
コネクタの始動	42
コネクタの停止	44
複数のコネクタ・インスタンスの作成	45
サポートされるビジネス・オブジェクトの追加	48
第 4 章 JText コネクタ・メタオブジェクトの使用	49
JText メタオブジェクトの命名規則	50
JText メタオブジェクトの構造	50
共通の構成タスク	65
第 5 章 JText コネクタのトラブルシューティング	91
エラー・メッセージのロギング	91
メタオブジェクトの名前に関する問題	91
イベントの起動に関する問題	91
JText での障害の処理	92
付録. コネクタの標準構成プロパティ	99

標準コネクタ・プロパティの構成	99
標準プロパティの要約	100
標準構成プロパティ	103
索引	115
特記事項.	117
特記事項	117

本書について

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

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一連の資料では、すべての WebSphere Business Integration Server Express Adapter インストールに共通の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。

関連する文書は、次のサイトからインストールできます。

<http://www.ibm.com/websphere/wbiserverexpress/infocenter>

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

<http://www.ibm.com/software/integration/websphere/support/>

適切なコンポーネント領域を選択し、「Technotes (技術情報)」セクションと「Flashes (速報)」セクションを参照してください。

表記上の規則

本書は、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、大括弧はオプションの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、 <code>option[,...]</code> は、複数のオプションをコマンド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、 <code><server_name><connector_name>tmp.log</code> 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。OS/400 および Linux 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 Server Express システム製品のパス名は、ご使用のシステムでの製品インストール・ディレクトリーからの相対パスです。
%text% および \$text	パーセント (%) 記号で囲まれたテキストは、Windows の <code>text</code> システム変数またはユーザー変数の値を示します。UNIX 環境においてこれに相当する表記は <code>\$text</code> で、 <code>text</code> UNIX 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。 <ul style="list-style-type: none">• Windows: <code>IBM¥WebSphereServer</code>• OS/400: <code>/QIBM/ProdData/WBIServer43/product</code>• Linux: <code>/home/\${username}/IBM/WebSphereServer</code>

本リリースの新機能

リリース 4.3.1 の新機能

本書の最初のリリースです。

第 1 章 JText アダプターの概要

この章では、IBM WebSphere Business Integration 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、InterChange Server Express が統合ブローカーとして使用される場合の JText コネクターのアーキテクチャーを示しています。

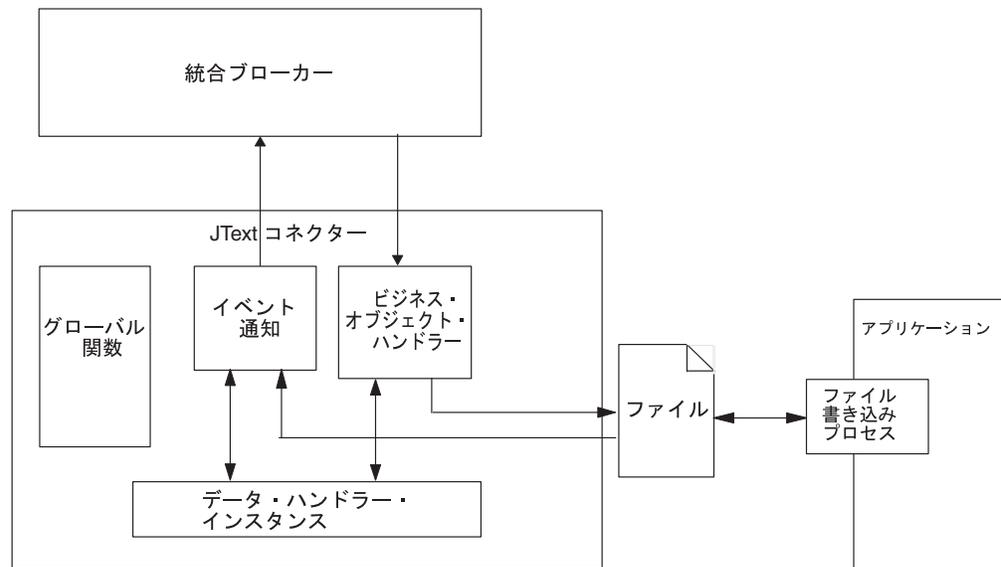


図 1. JText コネクター・アーキテクチャー

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、各種ファイル进行操作します。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行います。

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行われます。

データ変換を行うには、IBM WebSphere Business Integration 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=B0name)、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せん。

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、49 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、その名前ですべて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に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に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に必要なデータとデータ・ハンドラーに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 IBM WebSphere Business Integration 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法

コネクタ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、コネクタは指定されたファイルを、コネクタ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、77 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。

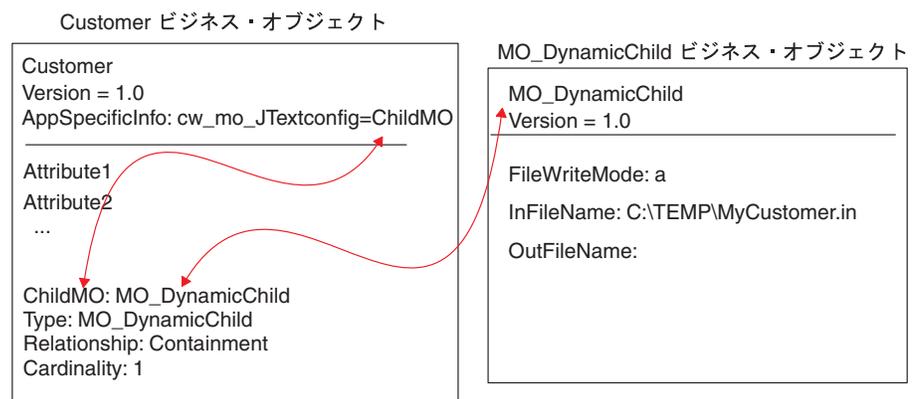


図2. 動的子メタオブジェクトの例

コネクタの動作

JText コネクタは、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 コネクタは、他のコネクタとは異なる方法でイベントを処理します。

JText コネクタは、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. コネクタは、データ・ハンドラー・インスタンスの `getB0()` または `getB0ByteArray()` 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

ーカイクします。データ・ハンドラーの処理が成功すると、コネクタはビジネス・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、73 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 プロパティにより、JText コネクタは、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、35 ページの『SortFilesOnTimestamp』を参照してください。

PollQuantity プロパティは、コネクタ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 プロパティを使用する際にパフォーマンスを調整する場合は、88 ページの『JText コネクタ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)。

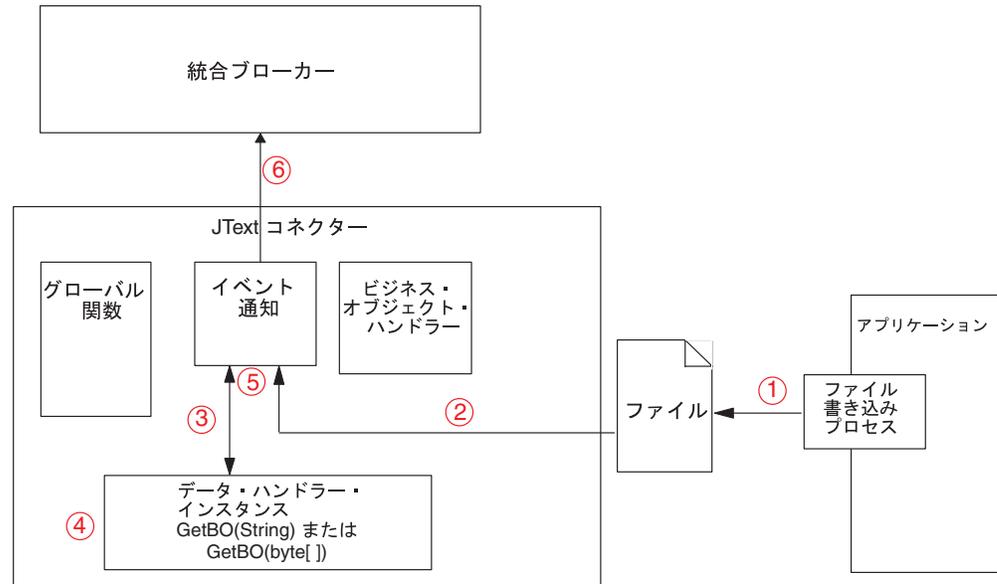


図3. イベント通知操作

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文字列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と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、コネクタがサブスクライブしない場合は、JText コネクタは、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、IBM WebSphere Business Integration Adapter for JText がイベントおよびアーカイブ・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	Windows: C:¥temp¥JTextConn¥Default¥Event Linux: /tmp/JTextConn/Default/Event OS/400: 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	Windows: C:¥temp¥JTextConn¥Default¥Archive Linux: /tmp/JTextConn/Default/Archive OS/400: 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	Windows: C:¥temp¥JTextConn¥Default¥Archive Linux: /tmp/JTextConn/Default/Archive OS/400: デフォルトなし
アーカイブ	unsubscribed	unsub	Windows: C:¥temp¥JTextConn¥Default¥Archive Linux: /tmp/JTextConn/Default/Archive OS/400: 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	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	Windows: C:¥temp¥JTextConn¥Default¥Archive Linux: /tmp/JTextConn/Default/Archive OS/400: デフォルトなし
アーカイブ	fail	fail	Windows: C:¥temp¥JTextConn¥Default¥Archive Linux: /tmp/JTextConn/Default/Archive OS/400: デフォルトなし
出力	out	out	Windows: C:¥temp¥JTextConn¥Default¥Out Linux: /tmp/JTextConn/Default/Out OS/400: 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、53 ページの表 9 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、コネクタは動的に出力ファイルの名前を付けたり、コネクタ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』
- 12 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. コネクタは、ビジネス・オブジェクト・レベルの `AppSpecificInfo` プロパティ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. コネクタはビジネス・オブジェクト・レベルの `AppSpecificInfo` プロパティ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「0」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、統合ブローカーからの宛先アプリケーションに対する要求を処理しているときのコネクター・コンポーネントを示します。

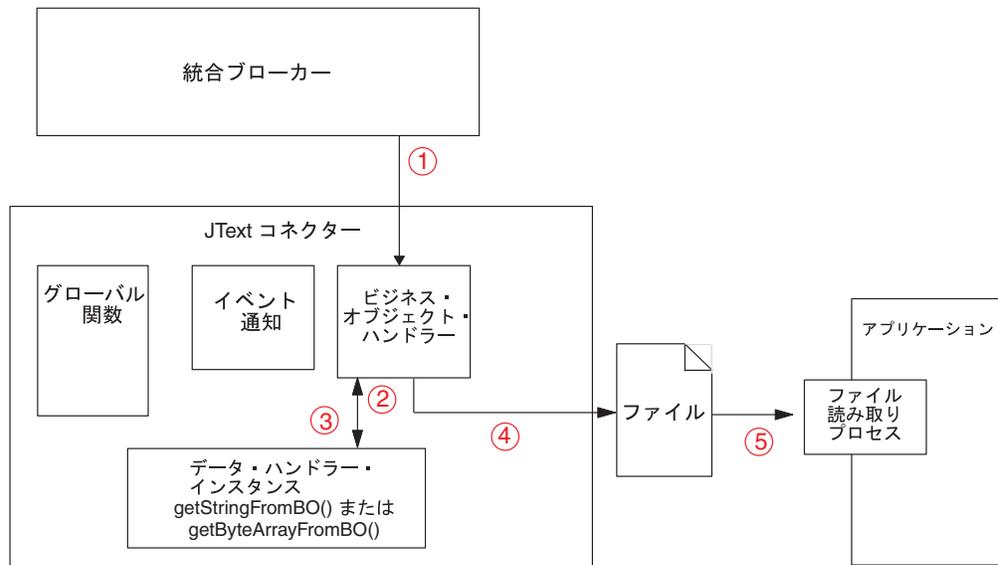


図4. ビジネス・オブジェクト要求操作

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文字列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。

コネクタは、`DataProcessingMode` メタオブジェクトの設定を基に、インターフェース (文字列またはバイト配列) を判別します。このメタオブジェクトの詳細については、53 ページの表9 を参照してください。

コネクタ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、コネクタはデータ・ハンドラー上の `getB0(String)` または `getB0(byte[])` メソッドを呼び出します。コネクタは、ビジネス・オブジェクトに変換される文字列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。
- コネクタは要求処理のためにデータ・ハンドラーの `getStringFromB0()` または `getBytesFromB0()` メソッドを呼び出します。コネクタは、文字列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、文字列またはバイト配列の形で戻します。

`getB0(String)` (または `getB0(byte[])`) メソッドと `getStringFromB0()` (または `getBytesFromB0()`) 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。
- コネクタの `UseDefaults` プロパティが `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。

コネクタの機能

JText コネクタは、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・ローケーション、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・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』、49 ページの『第 4 章 JText コネクタ・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、49 ページの『第 4 章 JText コネクタ・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、コネクタは構成済みのイベント・ディレクトリ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 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を設定します。これらのプロパティに関する詳細は、99 ページの『コネクタの標準構成プロパティ』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』
- 18 ページの『JText アダプターのインストール』
- 18 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。

- 統合ブローカーをインストールします。このタスクでは、WebSphere Business Integration システムをインストールして統合ブローカーを始動します。詳しくは、インストール資料を参照してください。
- アダプターと関連ファイルをインストールします。このタスクでは、アダプター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。18 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。

この章の内容

この章で説明するタスクは以下のとおりです。

表 2. アダプターのインストール: タスクのロードマップ

タスク	関連手順 (参照先...)	詳細について (参照先...)
アダプターのインストール	18 ページの『JText アダプターのインストール』	Windows 版、OS/400 版、または Linux 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	18 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。

- 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』
- 18 ページの『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。

- Windows 2000
- Windows 2003
- IBM OS/400 V5R2、V5R3
- Red Hat Enterprise Linux AS 3.0
- SuSE Linux Enterprise Server 8.1 (SP3 を適用)

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を設定します。構成プロパティの詳細については、付録を参照してください。

JText アダプターのインストール

アダプターのインストールの詳細については、Windows、Linux、または OS/400 用の「WebSphere Business Integration Server Express インストール・ガイド」を参照してください。これらの資料は、以下のサイトの WebSphere Business Integration Server Express Infocenter にあります。

<http://www.ibm.com/websphere/wbiserverexpress/infocenter>

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。

コネクター・コンポーネントのインストール方法の詳細については、以下の資料を参照してください。

- Windows 版、OS/400 版、または Linux 版の「WebSphere Business Integration Server Express クイック・スタート・ガイド」
- Windows 版、OS/400 版、または Linux 版の「WebSphere Business Integration Server Express インストール・ガイド」

Windows システムでのインストールの検証

ユーティリティーは、コネクターを `ProductDir¥connectors¥` ディレクトリーにインストールします (ここで、`ProductDir` はデフォルトで `IBM¥WebSphereServer`)。

表 3 には、アダプターが使用する Windows ファイルの構造が説明されており、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が示されています。インストールを検証するには、ディレクトリーを、アダプターをインストールしたディレクトリーに変更し、ディレクトリーの内容を表 3 のリストと比較します。

表 3. コネクターの Windows ファイル構造

ディレクトリー	説明
<code>ProductDir¥connectors¥JText</code>	コネクターの <code>CWJText.jar</code> および <code>start_JText.bat</code> ファイルが格納されています。
<code>ProductDir¥connectors¥JText ¥dependencies</code>	サード・パーティーの <code>commons-net-1.1.0.jar</code> ファイルと <code>jakarta-oro-2.0.8.jar</code> ファイルが格納されています。
<code>ProductDir¥connectors ¥messages</code>	<code>JTextConnector.txt</code> メッセージ・ファイルと、 <code>JTextConnector_II_TT.txt</code> ファイル (言語に固有なメッセージ・ファイル (<i>II</i>) と国/地域に固有なメッセージ・ファイル (<i>TT</i>)) が格納されています。
<code>ProductDir¥repository¥JText</code>	<code>BIA_CN_JText.txt</code> リポジトリ定義が格納されています。
<code>ProductDir¥lib</code>	<code>WBIA.jar</code> ファイルが格納されています。
<code>ProductDir¥bin</code>	<code>CWConnEnv.bat</code> ファイルが格納されています。

インストーラーにより、コネクター・ファイルのアイコンが「プログラム」メニューに追加されます (「プログラム」>「IBM WebSphere Business Integration Express」->「アダプター」>「コネクター」をクリック)。コネクターをすばやく始動するには、このファイルへのショートカットをデスクトップに作成してください。

OS/400 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/WBIServer43/product` ディレクトリー (`ProductDir` と呼ぶ) にインストールされます。20 ページの表 4 に、コネクターが使用する OS/400 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。OS/400 システムでのアダプターのインストールを検証するには、ディレクトリーを、アダプターをインストールしたディレクトリーに変更し、内容を 20 ページの表 4 のリストと比較します。

表 4. OS/400 での JText アダプターのインストール・ファイル構造

ディレクトリー	説明
<i>ProductDir</i> /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。
<i>ProductDir</i> /repository/JText	BIA_CN_JText.txt リポジトリ定義が格納されています。
<i>ProductDir</i> /connectors/JText/Dependencies	使用許諾契約書、README ファイル、およびサード・パーティーの commons-net-1.1.0.jar ファイルと jakarta-oror-2.0.8.jar ファイルが格納されています。
<i>ProductDir</i> /connectors/messages	JTextConnector.txt ファイルと、JTextConnector_ll_TT.txt ファイル (言語に固有なメッセージ・ファイル (ll) と国/地域に固有なメッセージ・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。

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。

Linux 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 /home/\${username}/IBM/WebSphereServer ディレクトリー (*ProductDir* と呼ぶ) にインストールされます。表 5 に、コネクターが使用する Linux 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。アダプターのインストールを検証するには、ディレクトリーを、アダプターをインストールしたディレクトリーに変更し、ディレクトリーの内容を表 5 のリストと比較します。

表 5. Linux での JText アダプターのインストール・ファイル構造

ディレクトリー	説明
<i>ProductDir</i> /bin/Data/App	JText ファイルが格納されています。
<i>ProductDir</i> /connectors/JText	アダプター・アプリケーション固有のコンポーネント・クラス・ファイル CWEJText.jar、アダプターの始動ファイル start_JText.sh、および ReadMe.htm ファイルが格納されています。
<i>ProductDir</i> /connectors/JText/Dependencies	使用許諾契約書、ReadMe.htm ファイル、および NetComponents サブディレクトリーが格納されています。

表 5. Linux での JText アダプターのインストール・ファイル構造 (続き)

ディレクトリー	説明
<i>ProductDir</i> /connectors/JText/Dependencies /NetComponents/1.3.8/src/java/com/ oroinc/net/ftp	以下のオープン・ソース・ファイルを含め、FTP 機能の依存関係ファイルが格納されています。 <ul style="list-style-type: none"> • .java • FTPFileListParser.java • FTPClient.java
<i>ProductDir</i> /connectors/messages	JTextConnector.txt ファイルと、JTextConnector_II_TT.txt ファイル (言語に固有なメッセージ・ファイル (II) と国/地域に固有なメッセージ・ファイル (TT)) が格納されています。
<i>ProductDir</i> /repository/JText	MO_JText_Default.txt ファイルが格納されています。
<i>ProductDir</i> /lib	WBIA.jar ファイルが格納されています。
<i>ProductDir</i> /bin	CWConnEnv.sh ファイルが格納されています。

Linux システムでコネクタを始動するには、connector_manager コマンドを使用する必要があります。

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。

- 『コネクターの構成の概要』
- 45 ページの『複数のコネクター・インスタンスの作成』
- 48 ページの『サポートされるビジネス・オブジェクトの追加』

コネクターの構成の概要

コネクター・プロパティの構成は、Connector Configurator Express から行います。また、メタオブジェクトを構成して、コネクターが異なるビジネス・オブジェクトを異なる方法で処理できるようにします。

注: 「Parallel Process Degree Resource」を 1 より大きい値に設定して、JText コネクターを実行することはお勧めできません。「並列処理の度合い」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・プロパティの値を変更することができます。コネクター・プロパティに加える変更には以下のものがあります。

- 動的 — この変更は、変更後即時に変更内容が有効になります。
- 静的 — この変更では、変更内容を有効にするには、コネクター・コンポーネントの再始動またはシステムの再始動が必要です。

プロパティが動的であるか静的であるかを判断するには、ご使用の統合ブローカーの構成ユーティリティを参照してください。

コネクターには、2 つのタイプの構成プロパティがあります。

- 標準構成プロパティ
- コネクター固有の構成プロパティ

コネクターを実行する前に、これらのプロパティの一部の値を設定する必要があります。

コネクターの構成: タスクの概要

Adapter for JText の構成には、次のセクションで説明する標準コネクター・プロパティを使用します。また、その次のセクションで説明するコネクター固有のプロパティも使用します。

このセクションでは、以下のトピックについて説明します。

- 24 ページの『Connector Configurator Express の概要』
- 25 ページの『Connector Configurator Express の始動』
- 25 ページの『System Manager からの Configurator Express の実行』
- 26 ページの『コネクター固有のプロパティ・テンプレートの作成』

- 29 ページの『新しい構成ファイルを作成』
- 31 ページの『構成ファイル・プロパティの設定』
- 40 ページの『構成ファイルの保管』
- 41 ページの『構成の完了』
- 41 ページの『グローバル化環境における Connector Configurator Express の使用』
- 42 ページの『コネクタの始動』
- 44 ページの『コネクタの停止』

ビジネス・オブジェクトの処理の構成

メタオブジェクトを使用して、コネクタの動作の以下の性質を構成します。

- どのデータ・ハンドラーを使用するか。
- JText の処理をバイナリーとテキストのどちらのモードで実行するか。
- イベント・ファイルを調べるポーリングを、どのディレクトリーから行うか。
- イベント・ファイルにどのファイル拡張子を使用するか。
- アーカイブを行う場合、ファイルのアーカイブにどのディレクトリーを使用するか。
- アーカイブを行う場合、処理済みのファイルのアーカイブ時に、どのファイル拡張子を使用するか。
- 異なるディレクトリーにある異なるオブジェクトからイベントを選出するか、同一のイベント・ディレクトリーから複数のファイルを選出するか。

メタオブジェクトの詳細については、49 ページの『第 4 章 JText コネクタ・メタオブジェクトの使用』を参照してください。

Connector Configurator Express の概要

Connector Configurator Express では、WebSphere InterChange Server Express で使用するアダプターのコネクタ・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。

- コネクタを構成するための**コネクタ固有のプロパティ・テンプレート**を作成する。
- **コネクタ構成ファイル**を作成する。インストールするコネクタ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を設定する。
場合によっては、コネクタ・テンプレートでプロパティに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、コラボレーションとともに使用するマップを指定し、必要に応じてメッセージング、ロギングとトレース、およびデータ・ハンドラーに関するパラメーターを指定する必要があります。

コネクタ構成プロパティには、標準の構成プロパティ (すべてのコネクタがもつプロパティ) と、コネクタ固有のプロパティ (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に必要なプロパティ) とが含まれます。

標準プロパティは、すべてのコネクタで使用されるので、新規に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の値を設定する必要があります。

標準プロパティの範囲は、ブローカーと構成によって異なる可能性があります。特定のプロパティに特定の値が設定されている場合にのみ使用できるプロパティがあります。Connector Configurator Express の「標準のプロパティ」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が表示されます。

ただしコネクタ固有プロパティの場合は、最初にプロパティを定義し、その値を設定する必要があります。このため、特定のアダプタのコネクタ固有プロパティ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。

注: Connector Configurator Express は、Windows 環境でのみ実行できます。別の環境でコネクタ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、このファイルを別の環境へコピーしてください。

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 Configurator Express の実行

Connector Configurator Express をブローカーと連携させずに別個に実行して、コネクタ構成ファイルを編集することができます。

これを行うには、以下のステップを実行します。

- 「スタート」>「プログラム」から、「IBM WebSphere Business Integration Server Express」>「Toolset Express」>「開発」>「Connector Configurator Express」をクリックします。
- 「ファイル」>「新規」>「構成ファイル」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 Configurator Express の実行

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

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

1. System Manager を開きます。

2. 「System Manager」ウィンドウで、「統合コンポーネント・ライブラリー」アイコンを展開し、「コネクタ」を強調表示します。
3. System Manager メニュー・バーから、「ツール」>「Connector Configurator Express」をクリックします。「Connector Configurator Express」ウィンドウが開き、「新規コネクタ」ダイアログ・ボックスが表示されます。

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクタ」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。
2. 「標準のプロパティ」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を確認します。

コネクタ固有のプロパティ・テンプレートの作成

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

コネクタ固有プロパティのテンプレートを新規に作成するか、または既存のファイルをテンプレートとして使用します。

- テンプレートの新規作成については、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、新しい名前でのこのテンプレートを保管します。

新規テンプレートの作成

このセクションでは、テンプレートでプロパティを作成し、プロパティの一般特性および値を定義し、プロパティ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、新規コネクタ構成ファイルを作成するためのベースとして使用します。

テンプレートは以下のように作成します。

1. 「ファイル」>「新規」>「コネクタ固有プロパティ・テンプレート」をクリックします。
2. 以下のフィールドを含む「コネクタ固有プロパティ・テンプレート」ダイアログ・ボックスが表示されます。
 - 「テンプレート」、「名前」

このテンプレートが使用されるコネクタ（またはコネクタのタイプ）を表す固有の名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。

- 「旧テンプレート」、「変更する既存のテンプレートを選択してください」

「テンプレート名」表示に、現在使用可能なすべてのテンプレートの名前が表示されます。

- テンプレートに含まれているコネクタ固有のプロパティ定義を調べるには、「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定義のリストが「テンプレートのプレ

ビュー」表示に表示されます。テンプレートを作成するときには、ご使用のコネクターに必要なプロパティ定義に類似したプロパティ定義が含まれている既存のテンプレートを使用できます。

3. 「**テンプレート名**」表示からテンプレートを選択し、その名前を「**名前の検索**」フィールドに入力し (または「**テンプレート名**」で自分の選択項目を強調表示し)、「**次へ**」をクリックします。

ご使用のコネクターで使用するコネクター固有のプロパティが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。

一般特性の指定

「次へ」をクリックしてテンプレートを選択すると、「**プロパティ: コネクター固有プロパティ・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・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. 「**プロパティを編集**」表示でプロパティの名前を選択します。
3. 「**最大長**」および「**最大複数値**」のフィールドで、変更を行います。次のステップで説明するように、プロパティの「**プロパティ値**」ダイアログ・ボックスを開かない限り、そのプロパティの変更内容は受け入れられませんので、注意してください。
4. 値テーブルの左上の隅にあるボックスを右マウス・ボタンでクリックしてから、「**追加**」をクリックします。「**プロパティ値**」ダイアログ・ボックスが表示されます。このダイアログ・ボックスではプロパティのタイプに応じて、値だけを入力できる場合と、値と範囲の両方を入力できる場合があります。適切な値または範囲を入力し、「**OK**」をクリックします。
5. 「**値**」パネルが最新表示され、「**最大長**」および「**最大複数値**」で行った変更が表示されます。以下のような 3 つの列があるテーブルが表示されます。

「**値**」の列には、「**プロパティ値**」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・テンプレート」ダイアログ・ボックスが表示されます。

依存プロパティは、別のプロパティ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。プロパティを依存プロパティとして指定し、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」表示で、依存プロパティとして指定するプロパティを選択します。
2. 「プロパティを選択」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」表示で依存プロパティを強調表示させて矢印をクリックし、「依存プロパティ」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。

新しい構成ファイルを作成

コネクタ構成ファイルを作成するには、コネクタ固有のテンプレートから作成するか、既存の構成ファイルを変更します。

コネクタ固有のテンプレートからの構成ファイルの作成

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

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

- **名前**

コネクタ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**

デフォルトのブローカーは ICS です。この値は変更できません。

- **「コネクタ固有プロパティ・テンプレート」を選択**

ご使用のコネクタ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・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固有のプロパティが表示されます。

使用するテンプレートを選択し、「OK」をクリックします。

3. 構成しているコネクタ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の名前が表示されます。ここですべてのフィールドに値を入力して定義を完了するか、ファイルを保管して後でフィールドに値を入力するかを選択できます。

4. ファイルを保管するには、「ファイル」>「保管」>「ファイルに」をクリックするか、「ファイル」>「保管」>「プロジェクトに」をクリックします。プロジェクトに保管するには、System Manager が実行中でなければなりません。ファイルとして保管する場合は、「ファイル・コネクタを保管」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、コネクタの始動ファイルで指定するコネクタ構成ファイルのパスおよび名前に一致している必要があります。

5. この章で後述する手順に従って、「Connector Configurator Express」ウィンドウの各タブにあるフィールドに値を入力し、コネクタ定義を完了します。

既存の構成ファイルの使用

既存ファイルを使用してコネクタを構成するには、Connector Configurator Express でそのファイルを開き、構成を修正してから、構成ファイル (*.cfg) として保管する必要があります。

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。

- コネクタ定義ファイル。
コネクタ定義ファイルは、特定のコネクタのプロパティと、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。コネクタの配布パッケージの `¥repository` ディレクトリ内には、このようなファイルが格納されていることがあります (通常、このファイルの拡張子は `.txt` です。例えば、XML コネクタの場合は `CN_XML.txt` です)。
- InterChange Server Express リポジトリ・ファイル。
以前にコネクタの 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、そのコネクタの構成に使用されたりポジトリ・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、コネクタのコネクタ固有プロパティのほとんど、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、コネクタ構成ファイルは、ファイルを開いて、プロパティ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)
 - InterChange Server Express リポジトリ (*.in, *.out) (InterChange Server Express Repository (*.in, *.out))

これまでリポジトリ・ファイルを使用してコネクタを構成していた場合は、このオプションを選択します。リポジトリ・ファイルに複数のコネクタ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。

Connector Configurator Express では、以下のセクションに記載されているプロパティの値を設定する必要があります。

- 32 ページの『標準コネクタ・プロパティの設定』
- 33 ページの『アプリケーション固有の構成プロパティ』
- 37 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』
- 38 ページの『関連付けられたマップ』
- 40 ページの『トレース/ログ・ファイル値の設定』

注: コネクタが JMS メッセージングを使用するものである場合、データをビジネス・オブジェクトに変換するデータ・ハンドラーを構成できるように、追加のカテゴリが表示されることがあります。詳細については、40 ページの『データ・ハンドラー』を参照してください。

構成ファイル・プロパティの設定

新規のコネクタ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に対応する複数のタブがあります。

標準プロパティとコネクタ固有プロパティの違いは、以下のとおりです。

- コネクタの標準プロパティは、コネクタ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有されます。すべてのコネクタが同じ標準プロパティのセットを使用します。これらのプロパティの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。

- アプリケーション固有のプロパティは、コネクタ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。各コネクタには、そのコネクタ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があります。これらのプロパティには、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および推奨値が記述されています。

「標準プロパティ」と「コネクタ固有プロパティ」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を表します。これらのプロパティは、アプリケーションまたはコネクタ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の除去も可能です。
- 「値」フィールドは構成可能です。
- 各プロパティごとに表示される「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。

標準コネクタ・プロパティの設定

標準構成プロパティにより、すべてのコネクタによって使用される情報が提供されます。これらのプロパティの資料が必要な場合は、99 ページの『コネクタの標準構成プロパティ』を参照してください。

付録に記載の構成プロパティに関する情報のうち、このコネクタに固有のものを表 6 に示します。

表 6. このコネクタ固有のプロパティ情報

プロパティ	注
CharacterEncoding	このコネクタは Java ベースのコネクタであるため、このプロパティは使用しません。
Locale	このコネクタは国際化されているため、このプロパティの値は変更できます。現在サポートされているロケールを知るには、コネクタのリリース情報を参照してください。

標準のプロパティの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示された場合にはメニューから値を選択します。
3. 標準のプロパティ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。

アプリケーション固有の構成プロパティー

アプリケーション固有 (またはコネクタ固有) の構成プロパティーは、コネクタが実行時に必要とする情報です。また、このプロパティーには、コネクタ内の静的情報やロジックを、コネクタ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 7 に、コネクタ固有のコネクタ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 7. コネクタ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
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

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。このプロパティを `retry` に設定した場合、コネクタは `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。このプロパティを `abort` に設定した場合、コネクタ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、93 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が 10000 に設定され、`FTPPollFrequency` が 6 に設定されている場合、コネクタはローカル・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。コネクタが FTP ポーリングを実行するのは、このプロパティに値を指定した場合のみです。`FTPPollFrequency` に 0 または空白が設定された場合、コネクタは 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。

このプロパティのデフォルト値はありません。

GenerateTemplate

コネクタの始動後に、コネクタ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、89 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、コネクタはこのシーケンス番号を使用して、ビジネス・オブジェクトごとに作成する出力ファイルに固有の名前を付けます。コネクタ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、88 ページの『JText コネクタ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、Windows プラットフォームでのみサポートされています。また、JText アダプターは、FTP サーバーが `mdtm` (変更日付と時刻) コマンドをサポートする場合にのみ、FTP サイトからのタイム・スタンプに基づいてリモート・ファイルをポーリングできます。

アプリケーション固有の構成プロパティの設定

アプリケーション固有の構成プロパティの場合、プロパティ名の追加または変更、値の構成、プロパティの削除、およびプロパティの暗号化が可能です。プロパティ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。プロパティを追加するときは「追加」をクリックします。子プロパティを追加するには、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. プロパティまたは子プロパティの値を入力します。
3. プロパティを暗号化するには、「暗号化」ボックスを選択します。
4. 32 ページの『標準コネクタ・プロパティの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

各プロパティごとに表示される「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・プロパティ名を変更すると、コネクタに障害が発生する可能性があります。コネクタ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、特定のプロパティ名が必要である場合があります。

コネクタ・プロパティの暗号化: 「プロパティを編集」ウィンドウの「暗号化」チェック・ボックスにチェックマークを付けると、アプリケーション固有のプロパティ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。

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

プロパティに複数の値がある場合には、プロパティの最初の値に「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、そのプロパティのすべての値が暗号化されます。プロパティの複数の値を暗号化解除するには、そのプロパティの最初の値の「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、すべての複数值が暗号化解除されます。

更新メソッド: 付録『コネクタの標準構成プロパティ』の 99 ページの『プロパティ値の設定と更新』にある更新メソッドの説明を参照してください。

コネクタ・プロパティはほとんどが静的なプロパティであり、それらの更新メソッドはコンポーネント再始動です。変更を有効にするには、変更したコネクタ構成ファイルを保管した後、コネクタを再始動する必要があります。

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。

サポートされるビジネス・オブジェクトを指定するときには、指定するビジネス・オブジェクトとそのオブジェクトに対応するマップが、システムに存在していなければなりません。ビジネス・オブジェクト定義 (データ・ハンドラー・メタオブジェクトのビジネス・オブジェクト定義を含みます) とマップ定義は、統合コンポーネント・ライブラリー (ICL) プロジェクトに保管されている必要があります。ICL プロジェクトの詳細については、「*WebSphere Business Integration Server Express ユーザーズ・ガイド*」を参照してください。

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

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

1. 「ビジネス・オブジェクト名」リストで空のフィールドをクリックします。
System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの「エージェント・サポート」(以下で説明) を設定します。
4. 「Connector Configurator 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定義が、System Manager のプロジェクトに保管されます。

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

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ず。
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・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・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

コネクタの最大トランザクション・レベルは、そのコネクタ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の場合、選択可能な項目は「最大限の努力」のみです。

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

関連付けられたマップ

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

場合によっては、関連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 つのマップを見つけて、バインドしようとします。

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

以下の手順を実行して、マップを明示的にバインドします。

1. 「明示的 (Explicit)」列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
2. ビジネス・オブジェクトに関連付けるマップを選択します。
3. 「Connector Configurator 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。

トレース/ログ・ファイル値の設定

コネクタ構成ファイルまたはコネクタ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、指定する格納場所へ移動し、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。このタブは、アダプターが保証付きイベント・デリバリーを利用するものである場合に使用可能になります。

これらのプロパティに使用する値については、標準プロパティに関する付録の『ContainerManagedEvents』の説明を参照してください。

構成ファイルの保管

コネクタの構成が完了したら、コネクタ構成ファイルを保管します。

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。

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、「*WebSphere InterChange Server 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、以下に太文字で示した行を追加してください。

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

```

        <Value>it_IT</Value>
        <Value>es_ES</Value>
        <Value>pt_BR</Value>
        <Value>en_US</Value>
        <Value>en_GB</Value>
        <DefaultValue>en_US</DefaultValue>
    </ValidValues>
</Property>

```

コネクタの始動

コネクタは、**コネクタ始動スクリプト**を使用して明示的に開始する必要があります。始動スクリプトは、次に示すようなコネクタのランタイム・ディレクトリに存在していなければなりません。例えば、Windows システムでは、始動スクリプトは `ProductDir¥connectors¥connName` ディレクトリにあります。ここで、`connName` はコネクタを識別します。

始動スクリプトの名前は、表 8 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 8. コネクタの始動スクリプト

オペレーティング・システム	始動スクリプト
Windows	start_connName.bat
OS/400	start_connName.sh
Linux	start_connName.sh

注: start_connName.sh 始動スクリプトを手動で実行する必要はありません。代わりに、以下のコマンドを使用して必要な環境変数を設定し、始動スクリプトを自動的に起動します。

```
connector_manager -start connName [-cConfigFile]
```

始動スクリプトの起動 (Windows システムの場合)

以下の方法でコネクタの始動スクリプトを起動することができます。

- **System Monitor から**
System Monitor を介して、コネクタをロード、活動化、非活動化、休止、シャットダウン、または削除できます。
- **Windows の「スタート」メニューから**
「プログラム」 → 「IBM WebSphere Business Integration Express」 → 「アダプター」 → 「コネクタ」 → 「your_connector_name」をクリックします。

デフォルトでは、プログラム名は「IBM WebSphere Business Integration 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、ご使用のコネクタへのデスクトップ・ショートカットを作成することもできます。

- **コマンド行から**

以下のコマンドを入力します。

```
start_connName connName WebSphereICSName [-cConfigFile]
```

ここで、*connName* はコネクタの名前であり、*WebSphereICSName* は InterChange Server Express インスタンスの名前 (デフォルトでは WebSphereICS) です。

- **Windows のサービスとして**

コネクタを 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、コネクタが始動します。

始動スクリプトの起動 (OS/400 システムの場合)

以下の方法でコネクタの始動スクリプトを起動することができます。

- **Windows システムから**

以下の手順を実行します。

1. WebSphere Business Integration Server Express Console がインストールされた場所から、「プログラム」→「**IBM WebSphere Business Integration Console**」→「コンソール」をクリックします。
2. OS/400 システム名または IP アドレス、および *JOBCTL 特殊権限を持つユーザーのユーザー・プロファイルとパスワードを指定します。
3. リストから正しいコネクタを選択し、「**アダプターを始動**」をクリックします。

- **OS/400 コマンド行から**

バッチ・モードまたは対話モードを使用します。

- バッチ・モード

CL コマンド QSH を実行し、QSHELL 環境を起動します。QSHELL 内から、以下のスクリプトを実行します。

```
/QIBM/ProdData/WBIServer43/bin/submit_adapter.sh connName  
WebSphereICSName pathToConnNameStartScript jobDescriptionName
```

ここで、*connName* はコネクタの名前であり、*WebSphereICSName* は InterChange Server Express インスタンスの名前、*pathToConnNameStartScript* はコネクタの始動スクリプトの絶対パス、*jobDescription Name* は QWBISVR43 ライブラリーで使用されるジョブ記述名です。

- 対話モード

CL コマンド QSH を実行し、QSHELL 環境を起動します。QSHELL 内から、以下のスクリプトを実行します。

```
/QIBM/ProdData/WBIServer43/WebSphereICSName/connectors  
/connName/start_connName.sh connName WebSphereICSName [-cConfigFile]
```

ここで、*WebSphereICSName* は InterChange Server Express インスタンスの名前であり、*connName* はコネクタの名前です。

- **System Monitor** から

System Monitor を介して、コネクターをロード、活動化、非活動化、休止、シャットダウン、または削除できます。

注: OS/400 システム上で自動的に始動された TCP/IP サーバーを使用してコネクターを始動するには、以下のスクリプトを使用します。

```
/QIBM/ProdData/WBIServer43/bin/add_autostart_adapter.sh connName  
WebSphereICSName pathToConnNameStartScript jobDescriptionName
```

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、*pathToConnNameStartScript* はコネクターの始動スクリプトの絶対パス、*jobDescription Name* は QWBISVR43 ライブラリーで使用されるジョブ記述名です。

始動スクリプトの起動 (Linux システムの場合)

以下のコマンドを入力することにより、Linux システムでコネクターの始動スクリプトを起動できます。

```
connector_manager -start connName WebSphereICSName [-cConfigFile]
```

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 (デフォルトでは WebSphereICS) です。

コネクターの停止

コネクターの停止方法は、使用しているプラットフォームによって異なります。以下のいずれかのセクションを参照してください。

コネクターの停止 (Windows システムの場合)

以下のいずれかの方法でコネクターを停止できます。

- **System Monitor** から
- コネクター・プロセスを実行しているウィンドウから

「コネクター」ウィンドウで q を押します。

- 「**Windows サービス**」パネルから

コネクターが Windows サービスとして構成されている場合は、サービスを停止できます。

コネクターの停止 (OS/400 システムの場合)

以下のいずれかの方法でコネクターを停止できます。

- **Console** またはコマンド行から

Console を使用してコネクターを始動した場合、あるいは、QSHHELL 環境で `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、以下を実行してコネクターを停止します。

1. CL コマンド `WRKACTJOB SBS(QWBISVR43)` を使用して、WebSphere Business Integration Server Express のジョブを表示します。

2. リストをスクロールして、ご使用のコネクターのジョブ記述 (例えば、JText コネクターのジョブ記述は QWBIJTEXTC) に一致するジョブ名を持つジョブを探し出します。
3. このジョブに対してオプション 4 を選択します。
4. F4 を押して、ENDJOB コマンドのプロンプトを表示します。
5. 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。

- **QSHHELL 環境から**

`start_connName.sh` スクリプトを使用して QSHHELL 環境からコネクターを始動した場合は、F3 を押してコネクターを停止できます。

- **System Monitor から**

コネクターの停止 (Linux システムの場合)

コネクターは、Linux システムのバックグラウンドで実行されます。コネクター用の個別のウィンドウはありません。コネクターを停止するには、以下のコマンドを実行します。

```
connector_manager connName -stop WebSphereICSName
```

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。

複数のコネクター・インスタンスの作成

注: このアダプター (あるいは WebSphere Business Integration Server Express または Express Plus に付属の任意のアダプター) の追加インスタンスを作成すると、そのアダプター・インスタンスは、配置できるアダプターの総数を制限するライセンス機能によって、別のアダプターとしてカウントされます。

以下に示すステップを実行することによって、コネクターの複数のインスタンスを作成して実行するように、ご使用のシステムを設定することができます。次のようにする必要があります。

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

新規ディレクトリーの作成

それぞれのコネクター・インスタンスごとにコネクター・ディレクトリーを作成する必要があります。このコネクター・ディレクトリーには、使用しているプラットフォームに応じた名前が付けられます。

Windows システム

ディレクトリーは、`ProductDir%connectors%connectorInstance` と命名する必要があります。ここで、`connectorInstance` はコネクター・インスタンスを固有に識別する名前です。

コネクタに、コネクタ固有のメタオブジェクトがある場合、コネクタ・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、このファイルを格納するために、`ProductDir\repository\connectorInstance` ディレクトリを作成します。

InterChange Server Express サーバー名を `startup.bat` ファイルのパラメーターとして指定できます。以下に例を示します。

```
start_JText.bat connName WebSphereICSName
```

OS/400 システム

新規ディレクトリは、

`/QIBM/UserData/WBIServer43/WebSphereICSName/connectors/connectorInstance` と命名する必要があります。ここで、`connectorInstance` は、コネクタ・インスタンスを一意的に識別し、`WebSphereICSName` は InterChange Server Express インスタンスの名前です。

コネクタに固有のメタオブジェクトがある場合は、そのコネクタ・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、このファイルを保管するために、

`/QIBM/UserData/WBIServer43/WebSphereICSName/repository/connectorInstance` ディレクトリを作成します。

InterChange Server Express サーバー名を `start_JText.sh` スクリプトのパラメーターとして指定できます。以下に例を示します。

```
start_JText.sh connName WebSphereICSName [-cConfigFile]
```

Linux システム

新規ディレクトリは、`ProductDir/connectors/connectorInstance` と命名する必要があります。ここで、`connectorInstance` はコネクタ・インスタンスを固有に識別する名前です。

コネクタに、コネクタ固有のメタオブジェクトがある場合、コネクタ・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、このファイルを保管するために、

`ProductDir/repository/connectorInstance` ディレクトリを作成します。

InterChange Server Express サーバー名を `connector_manager` コマンドのパラメーターとして指定できます。以下に例を示します。

```
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・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. 初期コネクタのファイルは、プラットフォームに応じて、次のディレクトリに入っていないければなりません。

- **Windows システム:** *ProductDir¥repository¥initialConnectorInstance*

追加で作成するファイルは、*ProductDir¥repository* ディレクトリーの適切な *connectorInstance* サブディレクトリーに格納する必要があります。

- **OS/400 システム:** *QIBM/UserData/WBIServer43/WebSphereICSName/repository/initialConnectorInstance*。ここで、*WebSphereICSName* は InterChange Server Express インスタンスの名前です。

追加で作成するファイルは、

QIBM/UserData/WBIServer43/WebSphereICSName/repository ディレクトリーの適切な *connectorInstance* サブディレクトリーに格納する必要があります。

- **Linux システム:** *ProductDir/repository/initialConnectorInstance*

追加で作成するファイルは、*ProductDir/repository* ディレクトリーの適切な *connectorInstance* サブディレクトリーに格納する必要があります。

コネクタ定義の作成

Connector Configurator Express 内で、コネクタ・インスタンスの構成ファイル (コネクタ定義) 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. 初期コネクタの構成ファイル (コネクタ定義) をコピーし、名前変更します。
2. 各コネクタ・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・プロパティ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. 初期コネクタの始動スクリプトをコピーし、コネクタ・ディレクトリーの名前を含む名前を付けます。

`dirname`

2. (*Linux* システムのみ) 始動スクリプト `CONJAR` を、`CONJAR=${CONDIR}/CW${CONNAME}.jar` から以下のスクリプトに変更する必要があります。

`CONJAR=${CONDIR}/CWJText.jar`

3. この始動スクリプトを、45 ページの『新規ディレクトリーの作成』で作成したコネクタ・ディレクトリーに格納します。
4. (*Windows* システムのみ) 始動スクリプトのショートカットを作成します。
5. (*Windows* システムのみ) 初期コネクタのショートカット・テキストをコピーして、新規コネクタ・インスタンス名に一致するように初期コネクタの名前を (コマンド行で) 変更します。
6. (*OS/400* システムのみ) 次の情報を使用して、コネクタのジョブ記述を作成します。

```
CRTDUPOBJ OBJ(QWBIJTEXTC) FROMLIB(QWBISVR43)
OBJTYPE(*JOB) TOLIB(QWBISVR43) NEWOBJ(newConName)
```

ここで、*newConName* は新規 JText コネクターのジョブ記述として使用する 10 文字の名前です。

7. (OS/400 システムのみ) 新規コネクターを Console に追加します。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。

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

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。コネクターの使用方法によっては、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、84 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_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、85 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 Server Express 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进行处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・タイプ进行处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、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。

- 50 ページの『JText メタオブジェクトの命名規則』
- 50 ページの『JText メタオブジェクトの構造』
- 65 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・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・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。例えば、`Oracle_Customer` ビジネス・オブジェクトを `MO_JTextConnector_Oracle_Customer` メタオブジェクトと関連付けられます。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、85 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_DefaultNameValueCollection`) 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。
- EventHandler 属性および OutputHandler 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。

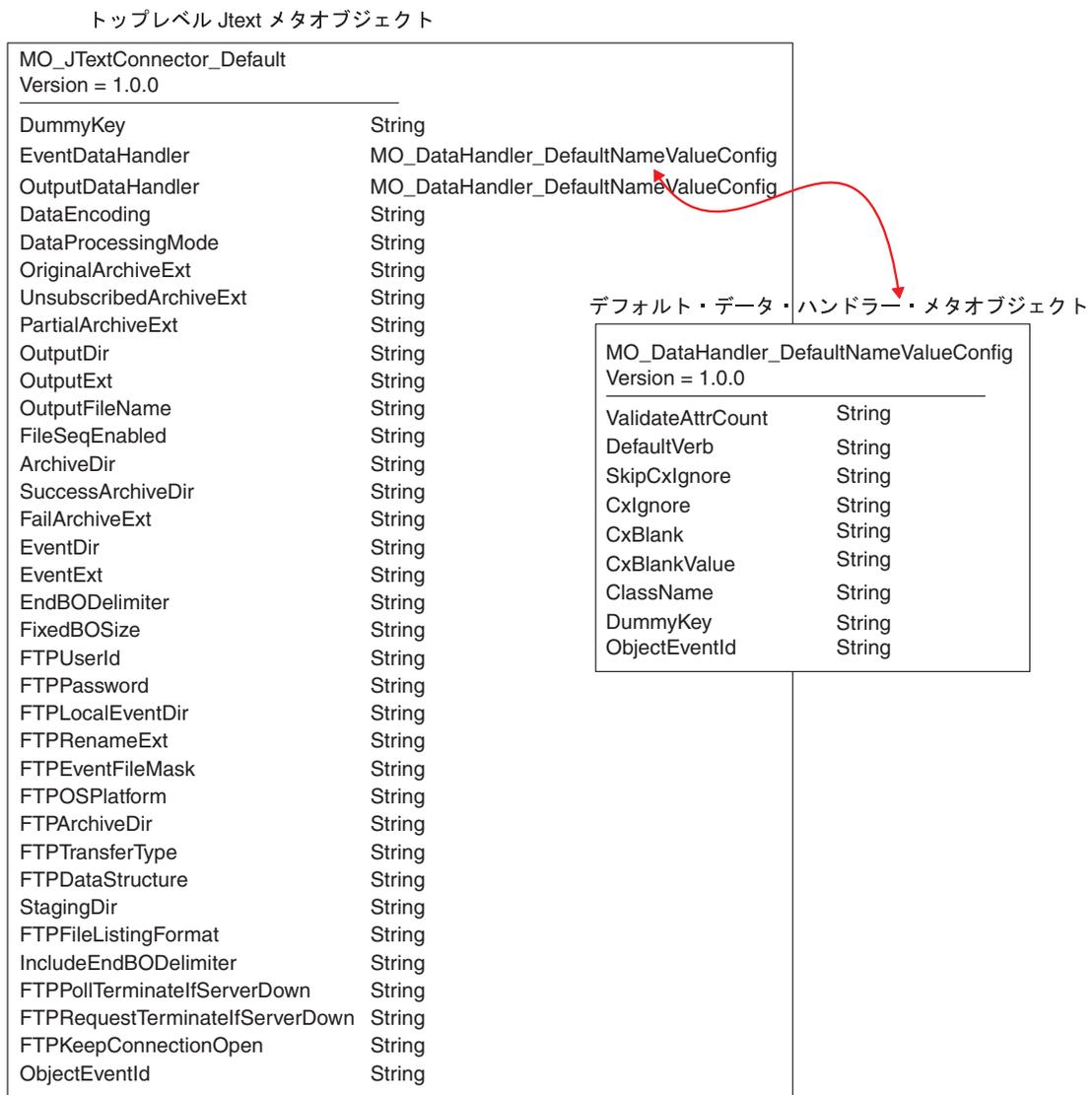


図 5. JText メタオブジェクトの階層構造

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。

53 ページの表 9 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 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性

属性名	説明
ArchiveDir	<p>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。</p> <p>Windows: C:%temp%\JTextConn\Default\Archive</p> <p>Linux: /tmp/JTextConn/Default/archive</p>
DataEncoding	<p>OS/400: デフォルトなし</p> <p>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。</p>
DataProcessing Mode	<p>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、getB0(byte[]) および getByteArrayFromB0() 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。</p>
DummyKey	<p>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。</p>

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
EndBODelimiter	<p>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。</p> <p>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、このプロパティのデフォルト値は FF01 になります。</p> <p>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tringが有効な EndBODelimiter の値です。</p>
EndOfFileDelimiter	<p>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。</p>
EventDataHandler	<p>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・String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。</p>

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
EventDir	<p data-bbox="857 258 1458 657">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 属性に固有の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。</p> <ul data-bbox="857 678 1458 1024" style="list-style-type: none"> • 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。 <p data-bbox="857 1052 1458 1108">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。</p> <pre data-bbox="857 1129 1458 1186">ftp://[UserId:password@]FTPserver[:port] [RemoteEventDirectory]</pre> <p data-bbox="857 1213 1458 1375">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。</p> <pre data-bbox="857 1396 1458 1423">[file://]FullPathname</pre> <p data-bbox="857 1451 1458 1478">出荷時のデフォルト値は以下のとおりです。</p> <pre data-bbox="857 1514 1458 1541">Windows: C:%temp%JTextConn%Default%Event</pre> <pre data-bbox="857 1577 1458 1604">Linux: /tmp/JTextConn/Default/event</pre> <p data-bbox="857 1631 1458 1659">OS/400: デフォルトなし</p>

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
EventExt	<p>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、73 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。</p> <p>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 <code>in</code> です。</p>
FailArchiveExt	<p>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 <code>fail</code> です。</p>
FileSeqEnabled	<p>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 <code>true</code> です。</p>
FixedB0Size	<p>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 <code>EndB0Delimiter</code> 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 <code>BO</code> 解析に代わる解析を行うことができます</p>

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
FTPArchiveDir	<p>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。</p> <ul style="list-style-type: none"> 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。
FTPDataStructure	<p>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。</p> <p>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。</p>

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
FTPEventFileMask	<p>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 その他の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・マスクを識別するときだけです。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、81 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。</p> <p>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せん。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ため、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。</p> <p>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。JText コネクタがファイルを読み取る時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> <p>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</p> <p>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。</p>
FTPFileListingFormat	

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
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。
FTPKeepConnectionOpen	この属性の Default Value プロパティを値 true に設定すると、JText コネクタは 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、コネクタは、コネクタ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。コネクタは、FTP サーバ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チェックします。接続がクローズされている場合は、コネクタは接続を再確立します。この属性の Default Value プロパティを値 false に設定すると、JText コネクタは、操作を実行するたびに FTP サーバ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	このプロパティは、FTP がイベント通知または要求応答で使用される場合に必須です。コネクタが 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・ディレクトリを指定します。コネクタが 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。
FTPPlatform	この属性を使用するのは、コネクタを構成して、リモート FTP サーバが 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、77 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、78 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。

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
FTPPollTerminateIfServerDown	<p>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。</p> <p>FTPPollTerminateIfServerDown 属性の Default Value プロパティを値 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。</p> <p>FTPPollTerminateIfServerDown 属性の Default Value プロパティを値 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。</p> <p>出荷時のデフォルト値はありません。</p>
FTPRenameExt	<p>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、81 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。</p>
FTPRequestTerminateIfServerDown	<p>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。</p> <p>FTPRequestTerminateIfServerDown 属性の Default Value プロパティを値 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。</p> <p>FTPRequestTerminateIfServerDown 属性の Default Value プロパティを値 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。</p> <p>出荷時のデフォルト値はありません。</p>
FTPTransferType	<p>この JText メタオブジェクト・プロパティは、イベント処理時と要求処理時の両方で使用されます。このプロパティに指定可能な値は、Binary と ASCII です。このプロパティは、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。</p>

表 9. MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)

属性名	説明
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、コネクタはこの属性を無視します。詳細については、78 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 プロパティを true に設定した場合は、コネクタは、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 プロパティを false に設定した場合は、コネクタは、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。
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、アーカイバの振る舞いが次のように変更されます。 <ol style="list-style-type: none"> 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。 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。 <p>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。</p>
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・コマンドの引き数を表します。
ObjectEventID	メタオブジェクト内のコネクタは使用しないが、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。

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
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 <code>orig</code> 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 <code>MO_DataHandler_DefaultNameValueConfig</code> です。
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。 <ul style="list-style-type: none"> • 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。 <p>OutputDir 属性で FTP 情報を指定する構文は、<code>ftp://[UserId:password@]FTPserver[:port]</code> です。詳細については、82 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。</p> <p><code>[file://]FullPathname</code></p> <p>出荷時のデフォルト値は以下のとおりです。</p> <p>Windows: <code>C:%temp%JTextConn%Default%Out</code></p> <p>Linux: <code>/tmp/JTextConn/Default/out</code></p> <p>OS/400: デフォルトなし</p>

表 9. MO_JTextConnector_Default メタオブジェクト定義の属性 (続き)

属性名	説明
OutputExt	<p>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。</p> <p>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。</p>
OutputFileName	<p>要求処理時にコネクタが着信ビジネス・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。</p> <p>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。</p>
PartialArchiveExt	<p>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。</p>
StagingDir	<p>コネクタが、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。</p> <p>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・ディレクトリーの場合があります。</p> <p>出荷時のデフォルト値はありません。</p>

表 9. *MO_JTextConnector_Default* メタオブジェクト定義の属性 (続き)

属性名	説明
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 <code>success</code> 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 <code>unsub</code> 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 `String Native` に設定され、`FileSeqEnabled` 属性が `true` に設定された場合、コネクタはビジネス・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 `String 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、コネクタはその値も使用します。値が含まれない場合、コネクタは拡張子なしでファイル名を作成します。

注: コネクタ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』

- 73 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』
- 77 ページの『リモート FTP ファイル・システムの指定』
- 84 ページの『データ・ハンドラーの指定』
- 85 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』
- 86 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』
- 86 ページの『ObjectEventID 属性値の指定』
- 86 ページの『JText コネクタの 2 番目のインスタンスのセットアップ』
- 88 ページの『JText コネクタのパフォーマンス調整』
- 89 ページの『テスト用サンプル・ファイルの生成』
- 89 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』
- 『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に送信して処理するとき、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)。

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』
- 68 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。ここでは、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 `M0_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。

- 『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』
- 71 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することに加え、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、および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、11 ページの『要求処理』を参照してください。

コネクタ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を再始動する必要があります。コネクタ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、34 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 10を参照してください。
 - OutputFileName— コネクタが、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 10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、コネクタが出力ファイル名を生成または戻すようにすることに加え、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、および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、79 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 10 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 10. 出力ファイルの指定

要求される出力条件	属性/プロパティに必要な構成	属性/プロパティの値
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	ユーザー指定のパス名
	FileSeqEnabled	true
	OutputFileName	Native
コネクタ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) InFileName (動的子メタオブジェクトの場合) FileWriteMode (動的子メタオブジェクトの場合) 次のメタオブジェクト構成を使用する。 MO_JTextConnector_businessobjectname: OutputDir	cw_mo_JTextConfig = DynChildMOName CxIgnore 該当なし 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

表 10. 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
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 = DynChildMOMName
	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:	
	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、85 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。

- 『EndBODelimiter ベースの解析手法の使用』
 - 75 ページの『EndBODelimiter の値としての印刷不可能文字の使用』
- 77 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 <EndB0: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 <EndB0: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、コネクタは <EndB0: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。

75 ページの表 11 は区切り文字のオプションを示しています。

表 11. 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	<ul style="list-style-type: none"> 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
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 またはユーザー指定値	<ul style="list-style-type: none"> 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。	None	<p>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。</p> <p>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。</p>

注: ソース・ファイルに空の行が含まれる場合、コネクタ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。

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、86 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、CustomEndB0;EOL となる。

```
Sample_BO~Create~1~TableGenKey5~strange~TextConnector_924055528_0
CustomEndB0
Sample_BO~Create~2~TableGenKey5~strange~TextConnector_924055528_0
CustomEndB0
Sample_BO~Create~3~TableGenKey5~strange~TextConnector_924055528_0
CustomEndB0
Sample_BO~Create~4~TableGenKey5~strange~TextConnector_924055528_0
CustomEndB0
```

- 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_B0~Create~1~TableGenKey5~strange~TextConnector_924055528_0

Sample_B0~Create~2~TableGenKey5~strange~TextConnector_924055528_0

Sample_B0~Create~3~TableGenKey5~strange~TextConnector_924055528_0

Sample_B0~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_B0~Create~1~TableGenKey5~strange~TextConnector_924055528_0

Sample_B0~Create~2~TableGenKey5~strange~TextConnector_924055528_0

Sample_B0~Create~3~TableGenKey5~strange~TextConnector_924055528_0

Sample_B0~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文字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、85 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、このプロパティが EndBODelimiter プロパティに優先し、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。

- 78 ページの『リモートのイベント処理』
- 82 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。

- 『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』
- 79 ページの『リモート・アーカイブの指定』
- 80 ページの『リモート・ポーリングの指定』
- 80 ページの『コネクタがリモート・サイトからイベントを処理する方法』
- 81 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』
- 81 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

MVS 関連 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、コネクタは始動しません。コネクタ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 プラットフォームやその他の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にタイム・スタンプが

追加されます。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: JText アダプターは、MVS の PDS をサポートしていません。これは、PDS のメンバーは、アーカイブ時にタイム・スタンプを追加してリネームすることができないからです。

リモート・ポーリングの指定: 34 ページの『FTPPollFrequency』構成プロパティを使用して、コネクタが 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、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。コネクタが FTP ポーリングを実行するのは、このプロパティ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、コネクタは FTP ポーリングを実行しません。

詳細については、88 ページの『JText コネクタのパフォーマンス調整』を参照してください。

コネクタがリモート・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。

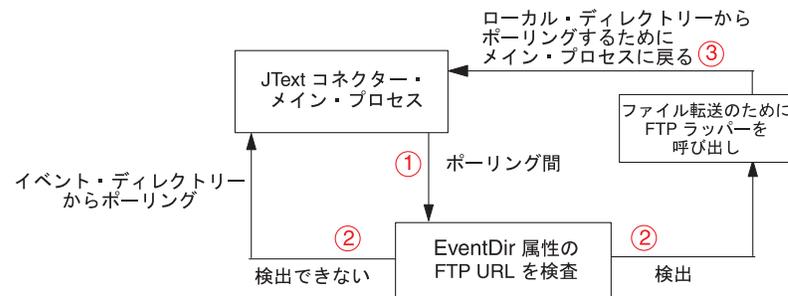


図 6. ローカルおよびリモートのイベント通知操作

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 またはその他の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 またはその他の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』
- 83 ページの『コネクタがリモート・サイトへのサービス呼び出し要求を処理する方法』
- 83 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。

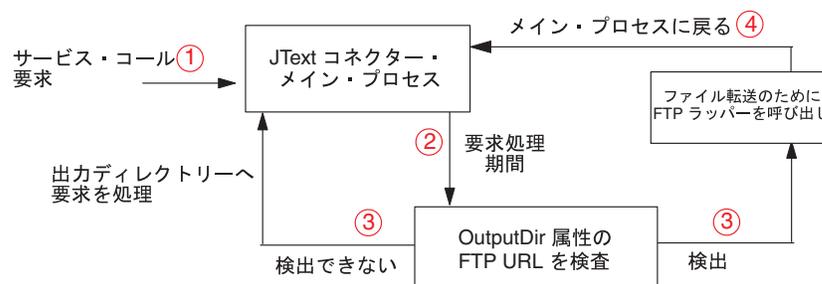


図 7. ローカルおよびリモートの要求操作

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を構成するには、FTPOSPPlatform 属性に MVS 属性を指定します。

FTP 転送用にコネクタ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時、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。コネクタ始動時にはシャットダウンしません。
- FTP サーバーによるファイルの処理のためにコネクタを構成する場合は、Apache Commons Net API に必要な固有の設定を使用するように FTP サーバーを構成する必要があります。

データ・ハンドラーの指定

JText コネクタ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、コネクタがサポートするかどうかを検証します。
- コネクタの始動時に、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 `EndB0Deliimiter` 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、コネクタによって ID が取り込まれないと、コネクタ・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 コネクタの 2 番目のインスタンスのセットアップ

`JText` コネクタに 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・ディレクトリーおよびリポジトリ・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 コネクタのすべてのメタオブジェクト (少なくとも 2 つあるはず) 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 コネクタの場合、名前を `MO_JTextConnector_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_` 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. Windows システムの場合、JText コネクタ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. OS/400 システムの場合は、コンソールに新規コネクタ・インスタンスを追加します。
8. Linux システムの場合は、既存のコネクタ・マネージャー・スクリプトのコピーを作成し、`JText2` を参照するようにパラメーターを変更します。
9. 統合ブローカーを再始動します。
10. ご使用のプラットフォームに応じて、以下のいずれかの操作を実行します。
 - Windows: 新規ショートカットをクリックします。
 - Linux: コネクタ・マネージャー・スクリプトを実行します。
 - OS/400: コンソールを使用してコネクタを始動します。

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構成プロパティを設定します。

- **PollQuantity** - このプロパティは、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 に適切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** - このコネクタ構成プロパティは、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いずれも頻繁に行われます。

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** - このコネクタ構成プロパティは、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、このプロパティに値を指定した場合のみです。FTPPollFrequency に 0 または空白が設定された場合、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 が互いに補い合う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』
- 『イベントの起動に関する問題』
- 92 ページの『JText での障害の処理』
- 93 ページの『イベント・ログ・ファイル』
- 94 ページの『障害リカバリー』
- 96 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』
- 96 ページの『サブスクリプション・エラーからのリカバリー』
- 96 ページの『フォーマット・エラーからのリカバリー』
- 97 ページの『送信エラーからのリカバリー』
- 97 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、93 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。

```
Wrong subscription: JText_Customer doesn't have supporting MO:  
this B0 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_JText2Connector_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。

```
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 12. 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 <code>EndBODElimiter</code> 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 <code>filename_timestamp.fail</code> 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 <code>filename_timestamp.unsub</code> 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。
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 <code>filename_timestamp.fail</code> 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。

表 12. JText エラー・タイプ (続き)

エラーのタイプ	説明
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が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 <code>filename_timestamp.fail</code> 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 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:

```
$ProdDir/JText/Event/Customer.in:: 1,3,4
```

OS/400:

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

Windows:

```
C:¥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、コネクタは以下の手順を行います。

1. コネクタはイベント・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、コネクタは以下を行います。
 - サブスクリプション・エラー— コネクタはアーカイブ・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。コネクタがログ・ファイルを参照するときに、ログ・ファイル内に新規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 34 ページの『EventRecovery』 構成プロパティを `retry` に設定している場合は、コネクタは始動時に、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、このプロパティを `abort` に設定している場合は、コネクタは、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、コネクタを再始動する必要があります。ただし、コネクタ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。コネクターが最も頻繁に戻すエラーは、`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)

付録. コネクターの標準構成プロパティ

この付録では、WebSphere InterChange Server Express で動作する、WebSphere Business Integration Server Express のアダプターに含まれるコネクター・コンポーネントの標準構成プロパティについて説明します。

コネクターによっては、一部の標準プロパティが使用されないことがあります。Connector Configurator Express から統合ブローカーを選択すると、ご使用のアダプターに対して構成する必要がある標準プロパティのリストが表示されます。

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

標準コネクター・プロパティの構成

アダプター・コネクターには 2 つのタイプの構成プロパティがあります。

- 標準構成プロパティ
- コネクター固有の構成プロパティ

このセクションでは、標準構成プロパティについて説明します。コネクター固有の構成プロパティについては、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 の使用

コネクター・プロパティの構成は 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 の使用方法の詳細については、付録の『Connector Configurator Express』を参照してください。

プロパティ値の設定と更新

プロパティ・フィールドのデフォルトの長さは 255 文字です。

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

1. デフォルト
2. リポジトリ
3. ローカル構成ファイル
4. コマンド行

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

- **動的**
変更を System Manager に保管すると、変更が即時に有効になります。
- **コンポーネント再始動**
System Manager でコネクターを停止してから再始動しなければ、変更が有効になりません。アプリケーション固有コンポーネントまたは統合ブローカーを停止、再始動する必要はありません。
- **サーバー再始動**
アプリケーション固有のコンポーネントおよび統合ブローカーを停止して再始動しなければ、変更が有効になりません。
- **エージェント再始動**
アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

特定のプロパティ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、次に示すプロパティの要約の表の「更新メソッド」列を参照してください。

標準プロパティの要約

表 13 は、標準コネクター構成プロパティの早見表です。標準プロパティの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティがあり、使用する統合ブローカーによってプロパティの設定が異なる可能性があります。

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

表 13. 標準構成プロパティの要約

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	CONNECTORNAME /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	CONNECTORNAME /ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は IDL
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクター・アプリケーション名として指定された値	コンポーネント再始動	
BrokerType	ICS	ICS		
CharacterEncoding	ascii7、ascii8、SJIS、Cp949、GBK、Big5、Cp297、Cp273、Cp280、Cp284、Cp037、Cp437 注: これは、サポートされる値の一部です。	ascii7	コンポーネント再始動	

表 13. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
ConcurrentEventTriggeredFlows	1 から 32,767	1	コンポーネント再始動	Repository Directory は <REMOTE>
ContainerManagedEvents	値なしまたは JMS	値なし	コンポーネント再始動	Delivery Transport は JMS
ControllerStoreAndForwardMode	true または false	true	動的	Repository Directory は <REMOTE>
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE>
DeliveryQueue		CONNECTORNAME/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	IDL または JMS	IDL	コンポーネント再始動	
DuplicateEventElimination	true または false	false	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならぬ
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	
FaultQueue		CONNECTORNAME/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	crossworlds.queue.manager	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>

表 13. 標準構成プロパティの要約 (続き)

プロパティ名	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	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR 注: これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	true または false	false	コンポーネント再始動	
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE>
MessageFileName	パスまたはファイル名	InterchangeSystem.txt	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	CONNECTORNAME/MONITORQUEUE	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は true でなければならぬ。
OADAutoRestartAgent	true または false	false	動的	Repository Directory は <REMOTE>
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE>
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE>
PollEndTime	HH:MM (HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクタのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	

表 13. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
PollQuantity	1 から 500	1	エージェント 再始動	JMS トランスポート のみ: Container Managed Events を指定
PollStartTime	HH:MM(HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント 再始動	
RepositoryDirectory	メタデータ・リポジトリ の場所		エージェント 再始動	<REMOTE> に設定する
RequestQueue	有効な JMS キュー名	CONNECTORNAME/REQUESTQUEUE	コンポーネント 再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	CONNECTORNAME/RESPONSEQUEUE	コンポーネント 再始動	Delivery Transport は JMS
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	1	動的	
SourceQueue	有効な JMS キュー名	CONNECTORNAME/SOURCEQUEUE	エージェント 再始動	Delivery Transport が JMS でありかつ Container Managed Events が指定されてい る場合のみ
SynchronousRequestQueue	有効な JMS キュー名	CONNECTORNAME/ SYNCHRONOUSREQUESTQUEUE	コンポーネント 再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミ リ秒)	0	コンポーネント 再始動	Delivery Transport は JMS
SynchronousResponseQueue	有効な JMS キュー名	CONNECTORNAME/ SYNCHRONOUSRESPONSEQUEUE	コンポーネント 再始動	Delivery Transport は JMS
WireFormat	CwBO	CwBO	エージェント 再始動	

標準構成プロパティ

このセクションでは、各標準コネクタ構成プロパティの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は `CONNECTORNAME/ADMINOUTQUEUE` です。

AgentConnections

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

使用する統合ブローカーを指定します。ICS を指定する必要があります。

CharacterEncoding

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

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

デフォルトでは、ドロップ・リストには、サポートされる文字エンコードの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、本書の Connector Configurator Express の使用方法に関する付録を参照してください。

ConcurrentEventTriggeredFlows

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

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

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

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

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

ContainerManagedEvents

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

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

デフォルト値は No value です。

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

- **PollQuantity** = 1 から 500
- **SourceQueue** = CONNECTORNAME/SOURCEQUEUE

また、**MimeType**、**DHClass**、および **DataHandlerConfigMOName** (オプション) プロパティを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティ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。「データ・ハンドラー」タブの値のフィールドは、**ContainerManagedEvents** を JMS に設定した場合にのみ表示されます。

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

ControllerStoreAndForwardMode

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。

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

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

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

デフォルト値は `true` です。

ControllerTraceLevel

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

DeliveryQueue

`DeliveryTransport` が `JMS` の場合のみ適用されます。

コネクターから `WebSphere InterChange Server Express` へビジネス・オブジェクトが送信されるときに使用されるキューです。

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

DeliveryTransport

イベントのデリバリーのためのトランスポート機構を指定します。指定可能な値は、`IDL` (`CORBA IIOP`) または `JMS` (`Java Messaging Service`) です。デフォルトは `IDL` です。

`DeliveryTransport` プロパティに指定されている値が `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。

JMS

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

`JMS` をデリバリー・トランスポートとして選択すると、`jms.MessageBrokerName`、`jms.FactoryClassName`、`jms.Password`、`jms.UserName` などの追加の `JMS` プロパテ

イヤーが Connector Configurator Express に表示されます。このうち最初の 2 つは、このトランスポートの必須プロパティです。

重要: WebSphere InterChange Server Express で動作しているコネクタで JMS トランスポート機構を使用すると、メモリ制限が発生することがあります。

この環境では、WebSphere MQ クライアント内でメモリが使用されるため、(サーバー側の) コネクタ・コントローラーと (クライアント側の) コネクタの両方を始動するのは困難な場合があります

DuplicateEventElimination

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

このプロパティは、false に設定することもできます。

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

EnableOidForFlowMonitoring

このプロパティを true に設定すると、アダプター・フレームワークは、フロー・モニターを使用できるようにするため、着信 **ObjectEventId** を外部キーとしてマークします。

デフォルト値は false です。

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。

デフォルト値は 128M です。

JvmMaxNativeStackSize

エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位)。

デフォルト値は 128K です。

JvmMinHeapSize

エージェントの最小ヒープ・サイズ (メガバイト単位)。

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は `crossworlds.queue.manager` です。

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

Locale

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

ロケール名は、次の書式で指定します。

```
ll_TT.codeset
```

ここで、以下のように説明されます。

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

デフォルトでは、ドロップ・リストには、サポートされるロケールの一部のみが表示されます。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、本書の `Connector Configurator Express` の使用方法に関する付録を参照してください。

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

<http://www.ibm.com/software/websphere/wbiadapters/infocenter>、または
<http://www.ibm.com/websphere/integration/wicsserver/infocenter>

LogAtInterchangeEnd

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

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

MaxEventCapacity

コントローラー・バッファー内のイベントの最大数。このプロパティーは、フロー制御で使用されます。

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

MessageFileName

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

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

コネクタが自動再始動およびリモート再始動機能を使用するかどうかを指定します。この機能では、MQ により起動される Object Activation Daemon (OAD) を使用して、異常シャットダウン後にコネクタを再始動したり、System Monitor からリモート・コネクタ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、このプロパティを true に設定する必要があります。MQ により起動される OAD 機能の構成方法については、「システム・インストール・ガイド (Windows 版)」を参照してください。

デフォルト値は false です。

OADMaxNumRetry

異常シャットダウンの後で MQ によりトリガーされる OAD がコネクタ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。このプロパティを有効にするためには、OADAutoRestartAgent プロパティを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

MQ によりトリガーされる OAD の再試行時間間隔の分数を指定します。コネクタ・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・コントローラーはコネクタ・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 プロパティで指定された回数だけ繰り返します。このプロパティを有効にするためには、OADAutoRestartAgent プロパティを true に設定する必要があります。

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

ポーリング・アクション間の時間の長さです。PollFrequency は以下の値のいずれかに設定します。

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

デフォルト値は `10000` です。

重要: 一部のコネクタでは、このプロパティの使用が制限されています。このプロパティが使用されるかどうかを特定のコネクタについて判断するには、該当するアダプター・ガイドのインストールと構成についての章を参照してください。

PollQuantity

コネクタ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固有のポーリング数設定プロパティがある場合、標準プロパティの値は、このコネクタ固有のプロパティの設定値によりオーバーライドされます。

PollStartTime

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

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

RequestQueue

WebSphere InterChange Server Express からコネクタへビジネス・オブジェクトが送信されるときに使用されるキューです。

デフォルト値は `CONNECTOR/REQUESTQUEUE` です。

RepositoryDirectory

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

この値は `<REMOTE>` に設定する必要があります。これは、コネクタが InterChange Server Express リポジトリからこの情報を取得するためです。

ResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクタ・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。WebSphere InterChange Server Express は、要求を送信した後、JMS 応答キューで応答メッセージを待機します。

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

SourceQueue

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

`JMS` イベント・ストアを使用する `JMS` 対応コネクタ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・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、105 ページの『`ContainerManagedEvents`』を参照してください。

デフォルト値は `CONNECTOR/SOURCEQUEUE` です。

SynchronousRequestQueue

`DeliveryTransport` が `JMS` の場合のみ適用されます。

同期応答を要求する要求メッセージを、コネクタ・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・フレームワークは、`SynchronousRequestQueue` にメッセージを送信し、`SynchronousResponseQueue` でブローカーから戻される応答を待機します。コネクタに送信される応答メッセージには、元のメッセージの ID を指定する 相関 ID が含まれています。

デフォルトは `CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE` です。

SynchronousResponseQueue

`DeliveryTransport` が `JMS` の場合のみ適用されます。

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

デフォルトは `CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE` です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。設定値は 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]

アダプター環境 17
アダプター・コンポーネント 2
イベント通知 7
 指定 66
イベント・アーカイブ 9
 指定 67
イベント・ログ・ファイル 93
インストール
 検証 18
 Linux での検証 20
 OS/400 での検証 19
 Windows での検証 19
エラー・メッセージ
 ロギング 91

[カ行]

構成タスク 65
構成ファイル
 完了 41
 プロパティの設定 31
 保管 40
コネクタ固有のプロパティ 33
 ArchivingEnabled 33
 EventLog 33, 34
 EventRecovery 33, 34
 FTPPollFrequency 33, 34
 GenerateTemplate 33, 34
 OutputLog 33, 34
 PollQuantity 33, 35
 SortFilesOnTimestamp 33, 35
コネクタの機能 15
コネクタの構成 70
 概要 23
コネクタの動作 7
コネクタの標準構成プロパティ 99
コネクタ・インスタンス
 複数作成 45

[サ行]

サンプル・ファイル
 生成 89

出力ファイル
 名前の指定 64
 障害リカバリー 94

[タ行]

データ・ハンドラー 2
 指定 84
 処理 14
 変更 85

[ハ行]

パフォーマンス
 調整 88
ビジネス・オブジェクト 3
 構成処理 24
 サポートの追加 48
 動詞の処理 15
ビジネス・オブジェクトのサンプル
 生成 89
標準コネクタ・プロパティ
 設定 32
複数イベント・ディレクトリー
 指定 73
複数イベント・ファイル
 指定 73
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 66

[マ行]

メタオブジェクト 3
 カスタムの作成 52
 構造 50
 固有のビジネス・オブジェクトの作成 85
 使用 49
 動的子 4
 命名規則 50

[ヤ行]

要求処理 11
 指定 69

[ラ行]

リモート FTP ファイル・システム
 指定 77
リモートのイベント処理 78
リモートの要求処理 82
ローカル・アーカイブ
 構成 68
ローカル依存データ 18
 処理 16

C

Connector Configurator
 グローバル化環境 41

E

EndBODelimiter ベースの解析手法
 使用 74

F

FixedBOSize ベースの解析手法
 使用 77

J

JText アダプター
 インストール 17, 18
 インストールの検証 18
 インストール・タスクの概要 17
 概要 1
 環境 17
 構成 23
 コネクタの始動 42
 コネクタの停止 44
 コンポーネント 2
 差異 16
 障害の処理 92
 前提条件 18
 トラブルシューティング 91
 パフォーマンスの調整 88
 プラットフォーム 18
 2 番目のインスタンスの設定 86

M

MO_JTextConnector_Default 属性 52

O

ObjectEventID 属性
値の指定 86

特記事項

特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

著作権使用許諾

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほめかしたり、保証することはできません。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

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

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。



IBM WebSphere Business Integration Server Express V4.3.1 および IBM WebSphere Business Integration Server Express Plus V4.3.1



Printed in Japan