IBM WebSphere Business Integration
Connect Enterprise and Advanced Editions

Connect プログラマーズ・ガイド

バージョン 4.2.2

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 159 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 Connect Enterprise Edition (5724-E87) および Advanced Edition (5724-E75) バージョン 4.2.2 に適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Connect Enterprise and Advanced Editions
Programmer Guide for Connect
Version 4.2.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.7

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

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

目次

本書について

IBM(R) WebSphere(R) Business Integration Connect Enterprise and Advanced Edition は、企業間 (B2B) 通信を管理するための、堅固でスケーラブルなプラットフォームを提供します。

本書では、日々のシステム管理の一部の自動化や、システムのプログラマチックなカスタマイズを使用可能にする、新規ツール・セットについて説明します。

対象者

本書は、お客様のサイトの WebSphere Business Integration Connect Enterprise Edition および Advanced Edition のコンサルタント、開発者、およびシステム管理者を対象としています。

関連資料

この製品で使用可能な文書の完全セットには、WebSphere Business Integration Connect Enterprise Edition および Advanced Edition のインストール、構成、管理、および使用に関する包括的な情報が含まれます。

次のサイトから文書をダウンロード、インストール、および表示することができます。 http://www.ibm.com/software/integration/wbiconnect/library/infocenter

注: 本書が発行された後にも、テクニカル・サポートの Technote および Flash で重要な情報を入手できる場合があります。これらの情報は、次の WebSphere Business Integration Support Web サイトで参照できます。
http://www.ibm.com/software/integration/websphere/support/

書体の規則

本書では、以下の規則を使用します。

`Courier フォント`	コマンド名、ファイル名、ユーザー入力情報、あるいはシステムが画面に出力する情報などのリテラル値を示します。
italic (イタリック)	変数名、相互参照の用語が初めて出現したことを示します。
青のアウトライン	資料をオンラインで参照した場合にのみ表示される青のアウトラインは、相互参照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して、参照オブジェクトにジャンプします。
{ }	構文行で、オプション・セットが中括弧で囲まれている場合、そこから 1 つのみを選択します。
|	構文行で、オプション・セットが縦線で分離されている場合、そこから 1 つのみを選択します。
[ ]	構文行で、オプショナル・パラメーターは大括弧で囲まれます。

. . .	構文行で、省略符号は直前のパラメーターの繰り返しを示します。例えば、option[,...] の場合、オプションを複数、コンマで区切って入力できます。
< >	名前の個々のエレメントは、不等号括弧で囲み、他のものと区別します。例えば、<server_name><connector_name>tmp.log となります。
/、¥	本書では、ディレクトリー・パスとして円記号 (¥) を使用するという規則を使用します。 UNIX をインストールする際は、円記号の代わりにスラッシュ (/) を使います。製品のパス名はすべて、ご使用のマシンの Connector のインストール先ディレクトリーの相対パスです。
ProductDir	製品のインストール先ディレクトリーを示します。

このリリースの新機能

バージョン 4.2.2 は『*Connect* プログラマーズ・ガイド』の最初のリリースです。

第 1 部 Business Integration Connect のカスタマイズ: ユーザー出口

Business Integration Connect は、企業間 (B2B) コミュニティー管理ソリューションです。Business Integration Connect を使用すると、企業の境を跨ぎ、ビジネス・インテグレーションを企業範囲を超えてコミュニティーにまで拡張し、取引コミュニティー内部でデータや処理を交換できます。取引コミュニティーは、通常、ハブ (コミュニティー・マネージャーとして機能する企業) を中心に展開します。コミュニティー参加者はハブに文書を送信し、ハブはその文書に必要な処理を実行して、処理された文書が適切な宛先に配送されます。

バージョン 4.2.1 と以前の Connect でハブの処理は、製品にハードコーディングされた構成可能なオプション・セットに限定されていました。これらのオプションはかなり広範囲に渡っていましたが、一部の取引シナリオが可能性があるにもかかわらずサポートされていませんでした。現行バージョン (4.2.2) から、Connect 製品で提供されるオプションの範囲外のシナリオを必要とするユーザーは、新たに開発された API セットに基づいて、追加のトランスポートやビジネス・プロトコルなどをサポートするプラグイン・モジュールを開発および配置して、いくつかの重要なステージでの処理をカスタマイズできるようになりました。これらのプラグイン・モジュールを呼び出すことのできるプロセスのポイントを、*ユーザー出口* と呼びます。

この章で、これらのユーザー出口を使用する WBI-C のカスタマイズ方法を説明します。

第 1 章 ユーザー出口の概要

ユーザー出口プラグインと標準 Connect コード本体がどのように相互作用するかを理解するには、ハブの処理フローを次の 3 つのステージに分割すると便利です。

- 『文書の受信』
- 4 ページの『文書処理』
- 6 ページの『文書の伝送』

この章では、これらのタスクを実行する WBI Connect コンポーネントと、ユーザー・カスタマイズに使用できるプロセスの機能について簡単に説明します。

文書の受信

ハブの処理システムにコンポーネントを経由して入力される文書は、*受信側* と呼ばれます。受信側は、インバウンド文書のトランスポートのモニター、到着した文書の検索、それに対する基本的な処理の実行、およびメイン処理エンジンが検索できるストレージ・キューへの配置を行います。

受信側は、トランスポート固有です。WBI Connect には、FTP/S、JMS、File、SMTP、および HTTP/S トランスポートを処理するように設計された受信側が付属しています。バージョン 4.2.2 には、ユーザーが特定のニーズに基づいて独自の受信側を開発できるように API も組み込まれています。

使用するには、受信側をターゲット と呼ばれるトランスポート構成に関連付けます。ターゲットには、Business Integration Connect に着信する文書のエントリー・ポイントの URI を指定します。ここでは、Document Manager が文書を検索するときのリポジトリー・ロケーションを指定します。各ターゲットは、単一トランスポート・タイプを使用して、送信する文書をサポートします。複数の文書フォーマットがサポートされる場合は、指定されたトランスポートごとに各タイプのターゲットが 1 つ存在する可能性があります。ターゲットは、WBI Connect の GUI ベース制御コンポーネントである、Community Console を使用して構成されます。提供されている受信側と同じ方法で、ユーザーが作成した受信側にターゲットを 1 つ以上持つことができます。ターゲットを構成して受信側と関連付けるためのコンソールの使用方法の詳細については、「*ハブ構成ガイド*」を参照してください。

Connect 4.2.2 のユーザーは、全く新しい受信側の追加に加え、ユーザー出口プラグイン・モジュールを開発して着信文書を受信側が処理する方法をカスタマイズできます。これらのモジュールは、ハンドラー と呼ばれます。追加処理を行うためにユーザー出口ハンドラーを呼び出すことのできる受信側の処理シーケンス内の場所として、前処理、同期検査、および後処理の 3 つがあります。*構成ポイント* とも呼ばれるこれらの各場所で、ユーザーはコンソールを使用して 1 つ以上のハンドラーを指定できます。このリリースでハンドラーを追加できるのは、ユーザー設計の受信側または WBI-C 提供の HTTP 受信側だけです。

前処理ハンドラーは、メイン処理が行われる Document Manager に文書を送信する前に、文書の必要な前処理を実行するために使用されます。例えば、複数のレコー

ドが 1 つのラッパー・メッセージで送信される場合があります。前処理で個別のメッセージを相互に分離してから、実際の処理のために送信することができます。

文書の伝送は、同期でも非同期でもかまいません。同期伝送では、開始パートナーは伝送する文書に応答する文書を予期します。非同期伝送では、応答文書は予期しません (しかし、HTTP 状況コードなどの一部の単純な情報が戻されることがあります)。ユーザー出口は、同期状況についての文書の検査を特定のハンドラーで処理するために使用できます。WBI-C には、`DefaultSynchronousSyncCheckHandler` と `DefaultAsynchrounousSyncCheckHandler` の 2 つのデフォルトの同期検査ハンドラーが付属していますが、ユーザーは独自のハンドラーを設定できます。これは、別個の同期および非同期ターゲット (別のハンドラーを使用) を定義するとスループットを向上できる一部の文書タイプの場合に、特に役立ちます。

後処理ハンドラーは、同期要求が出されたときに、開始パートナーに WBI-C が戻す応答文書を処理するために使用されます。

文書処理

コミュニティー参加者間で交換される文書は、すべてのパートナーのニーズに適合できるように、いくつかの方法で変形が必要な場合が頻繁に発生します。Document Manager コンポーネントの中核であるビジネス・プロセス・エンジン (BPE) が、これらの変形を実行します。この目的のために、BPE は、固定インバウンド・ワークフロー、可変ワークフロー、および固定アウトバウンド・ワークフローの 3 つの基本単位にグループ化される、一連のステップで行います。ユーザー出口により、ユーザー定義プロセスをこれらの各基本単位にプラグインすることができます。

固定インバウンド・ワークフロー

固定インバウンド・ワークフローは、受信側から Document Manager に着信するすべての文書に行う標準処理を対象とします。これは、ステップの数とタイプが常に同じであるので、固定です。しかしユーザーは、ユーザー出口を使用して、最初の 2 つのステップであるプロトコル・アンパックおよびプロトコル処理ステップの両方での処理に関するカスタマイズされたハンドラーを提供できます。

WBI-C に送信されるすべてのメッセージは、固有のビジネス・プロトコルの仕様に基づいてパッケージされます。例えば、RosettaNet 文書は RNIF 仕様に基づいてパッケージされます。プロトコル・アンパックには、メッセージをさらに処理するためのメッセージのアンパックが含まれます。このプロセスには、暗号化解除、圧縮解除、シグニチャー検証、ルーティング情報の抽出、ユーザー認証、ビジネス文書パーツの抽出が含まれる場合があります。WBI-C には、
RNIF、AS2、MIME、EAI、および NONE パッケージのハンドラーが用意されています。他のパッケージ・タイプのハンドラーが必要な場合は、ユーザー出口として作成できます。

プロトコル処理には、プロトコル固有情報の判別が含まれます。これには、ルーティング情報 (送信側 ID、宛先 ID)、プロトコル情報 (Rosettanet バージョン V02.02 などのビジネス・プロトコルとバージョン)、および文書フロー/プロセス情報 (3A4 バージョン V02.02 など) を判別するための構文解析が含まれる場合があります。

WBI-C には、XML、RosettaNet、および EDI の処理が用意されています。他のプロトコル (例えば CSV) の処理は、ユーザー出口プラグインを使用して用意できます。

ユーザー出口を使用して新規のパッケージ・タイプや新規のプロトコル・タイプをセットアップする場合は、新規のパッケージまたは新規のプロトコル情報をコンソールにもセットアップする必要があります。詳しくは、「*ハブ構成ガイド*」を参照してください。

可変ワークフロー

BPE での中核の変形処理は可変ワークフローで行われます。可変ワークフローは、順次に整列された複数のステップで構成され、それらすべてが集まってアクションとなります。アクションは、参加者接続の作成プロセスの一部として、コンソールで指定されます。WBI-C には、11 個の定義済みアクションが用意されています。ユーザー出口を使用すると、新規のシーケンスに配置する全く新しい一連のステップを開発するか、あるいは既存のアクションをコピーし、既存のステップをユーザー定義のステップで置換するか、または既存のシーケンスにユーザー定義ステップを追加し、既存のアクションを変更して新規のアクションを作成できます。

注: 内部的な WBI-C 特定の目的に使用されている可能性があるため、WBI-C が用意したステップすべてを新しいユーザー定義に使用できるわけではありません。詳しくは、44 ページの『テンプレートとしての WBI-C 提供アクション、およびその状況』を参照してください。

アクションは、一連のステップから構成されます。これらのステップの作成には、ユーザー出口プラグインを使用できます。通常、ステップには、次のタイプが含まれます。

- **検証**: ビジネス文書のフォームを検査します。例えば、XML 文書は XML スキーマに対して検証できます。WBI-C には、XML 検証ステップが用意されていますが、その他については作成することができます。
- **変形**: ビジネス文書のフォームを変更します。XML 文書は、XSLT を使用して別の XML 文書に変形できます。WBI-C には、XML 変形ステップが用意されていますが、その他については作成することができます。
- **変換**: ビジネス文書のフォーマット全体をあるタイプから別のタイプに変更します。
- **プロセス・ロギング**: この文書に固有のログを作成します。一部のビジネス・プロトコルには、固有のロギング要件があります。
- **シーケンス検査**: 文書のシーケンスを検査します。一部のビジネス・プロトコルには、固有の順序付け要件があります。

注: これらのステップでは、典型的な例のみを示します。可変ワークフローは、ビジネス・プロセス・ロジックの実装を意図しています。ロジックで、実際に必要なステップを指示します。

ステップが定義されると、ステップが実行されるシーケンスをアクションに指定する必要があります。例えば、定義ステップが検証と変形の場合、検証のみ、別の検証に続いて変形、および 3 番目の検証に続いて変形してさらに変形された文書の検

証、から構成されるアクションを順序付けることができます。ステップのシーケンスは、アクションとしてまとめてコンソールにリンクされます。詳しくは、「ハブ*構成ガイド*」を参照してください。

固定アウトバウンド

文書が該当する可変ワークフローで処理されると、その宛先に送信する準備としてパッケージ化する必要があります。WBI-C には、RNIF、バックエンド統合、AS、NONE パッケージ用と、cXML および SOAP プロトコル用のハンドラーが用意されています。他のパッケージ・ハンドラーが必要な場合は、ユーザー出口ステップとして作成できます。通常、これらのステップは次の 1 つ以上の処理を行います。

- アセンブルまたはエンベロープ
- 暗号化
- 署名
- 圧縮
- ビジネス・プロトコル固有のトランスポート・ヘッダーの設定

文書がパッケージされるとストレージ・キューに入れられ、送信側に引き渡すために Delivery Manager によって取り出されるまでそこに置かれます。

文書の伝送

多くの場合、WBI-C 文書の伝送プロセスは文書の受信プロセスを逆にしただけです。処理された文書は BPE ストレージ・キューから取り出され、処理を少し行って、その宛先に送信されます。文書は、トランスポート固有のコンポーネントである *送信側* を使用し、ゲートウェイ と呼ばれるコンソール・ベースの構成に基づき、トランスポートを介して送信されます。

このリリースから、ユーザーはこのステージを次のようにしてカスタマイズできます。a) まったく新しい送信側を作成するか (新規のトランスポートをサポートするため)、または b) 2 つの固有のポイントで標準またはユーザー定義の送信側の処理フローに配置される前処理および後処理ハンドラーを作成します (あるいはこの両方を作成します)。前処理ハンドラーは、文書が送信される前に文書の処理に影響を与えます。後処理ハンドラーは、文書が送信された後のデリバリー結果に影響を与えます (SenderResult オブジェクトにデリバリー状況を保持します。例えば、SOAP ベースの伝送が失敗すると、後処理ハンドラーは SOAP 障害メッセージを作成して、SenderResult オブジェクトに設定できます)。

第 2 章 受信側のカスタマイズ

受信側は、WBI Connect データ・フローの最初のステージを処理します。トランスポートから文書を取り出し、それらに基本的な処理をいくつか実行して、メインの処理コンポーネント (Document Manager) が取り出すストレージ・キューに入れます。同期要求の場合は、発信元の参加者に応答文書も戻します。 4.2.2 リリースでは、ユーザーは、新規受信側の作成、または新規受信側ハンドラーの作成および構成の 2 つの方法で受信側ステージの処理をカスタマイズできます。この章では、受信側カスタマイズについて、この 2 つの方法を説明します。

- 『受信側の新規作成の概要』
- 10 ページの『新規の受信側ハンドラー作成の概要』

追加セクションで、開発および配置について説明します。

- 11 ページの『開発と配置』

API リスト、またコードおよび疑似コード・サンプルのアウトラインは、次の章で説明します。

受信側の新規作成の概要

受信側は、トランスポート固有です。WBI-C には、FTP ディレクトリー、JMS、File ディレクトリー、SMTP (POP3)、および HTTP/S トランスポート用の受信側が付属しています。 WBI-C システムに新規トランスポートを追加するために、ユーザーは 4.2.2 リリースで提供される API を使用して、独自の受信側を作成できます。これらの新規受信側は、Community Console を使用して構成でき、また通常の方法で処理フローに統合できます。このセクションでは、新規受信側の開発プロセスを説明します。次について説明します。

- 『受信側フロー』
- 10 ページの『受信側タイプ』
- 10 ページの『受信側アーキテクチャー』

受信側フロー

受信側の内部の処理フローの本質はある程度は特定のトランスポートのニーズによって指示されますが、すべての受信側が実現する必要のある基本的なタスクがあります。ここでは、これらのタスクの上位について、一般的な方法で説明します。

1. トランスポートに到達するメッセージの検出

受信側は次の 2 つの方法のうちのいずれかを使用して、要求メッセージの到着を検出します。つまり、そのトランスポートに定義されたターゲットのポーリング (提供された JMS 受信側が実行) かまたはトランスポートからのコールバックの受信 (提供された HTTP/S 受信側が実行) です。

2. トランスポートからのメッセージの検索

受信側はトランスポートから、要求メッセージとトランスポート属性 (ヘッダーなど) を検索します。これには、ファイル・システムに一時ファイルを作成する場合があります。

3. WBI-C 必須ヘッダーの生成

文書の以降の処理に使用される、特別な WBI-C ヘッダーがあります。受信側には、トランスポート・メッセージ属性から、または他の方法で、該当するこれらのヘッダーのいずれかを作成するメカニズムが必要です。ヘッダーのリストは以下のとおりです。

- `InboundCharSet`: インバウンド文書の文字セット
- `MsgLengthIncHeaders`: メッセージの長さ (トランスポート・ヘッダーを含む)
- `requestURI`: HTTP の場合、受信したサーブレットの URI
- `timestamp`: 受信した文書のタイム・スタンプ
- `fromIP`: 文書が送信される宛先の IP アドレス
- `certDN`: SSL クライアント証明書の DN 名

さらに処理するために Document Manager に転送される*受信側要求*文書は、トランスポート・メッセージ、トランスポート・ヘッダー、およびこれらの WBI-C ヘッダーから構成されます。

注: 以下のステップ (4 および 5) は、いずれの順序でも実行できます。

4. 前処理の実行

受信側は WBI-C コンポーネント、受信側フレームワークを呼び出して、実際に前処理を行います。フレームワークは、コンソールを介してターゲットに指定されている Connect 提供またはユーザー定義のハンドラーを、コンソール構成画面に表示されている順に実行します。要求文書が入力として最初のハンドラーに渡され、戻された処理済みの文書が入力として次のハンドラーに渡される、という流れです。このステップでは、1 つ以上の文書を戻すことがあり、すべての受信側は、複数の戻り文書を処理するように設計する必要があります。

5. 同期状況の検査

受信側は受信側フレームワークを呼び出して、受信した要求が同期しているかどうかを判別します。フレームワークは、該当するハンドラーを見付けるまでハンドラーの構成済みリストを移動します。このステップの結果で、受信側の次ステップを判別します。要求が非同期の場合は、パス A に進みます。要求が同期の場合は、パス B に進みます。

6A. 非同期要求処理

要求が非同期の場合 (発生元の取引先に応答文書を戻す必要がない)、受信側はフレームワークを呼び出して要求文書を処理します。フレームワークは、Document Manager が検索する場所に情報を保管します。

6B. 同期要求処理

要求が同期の場合 (発生元の取引先に応答文書を戻す必要がある)、受信側はフレームワークを呼び出して要求文書を処理します。同期要求には、ブロッキングと非ブロッキングの 2 つのタイプがあります。ブロッキング・モードでは、フレームワークが Document Manager からの応答文書を受信側に

戻すまで、受信側の呼び出しスレッドはブロックされます。非ブロッキング・モードでは、受信側の呼び出しスレッドがすぐに戻されます。後で応答文書を受け取ったとき、フレームワークは呼び出し側で processResponse メソッドを呼び出し、再び応答文書を渡します。この応答に発生元の要求を同期させるため、相関オブジェクトが使用されます。

7. 後処理の実行

同期要求の場合、受信側はフレームワークを呼び出して、応答文書が発生元の取引先に戻される前に、その応答文書に対して後処理を実行します。フレームワークは、コンソールを介してターゲットに指定されている Connect 提供またはユーザー定義のハンドラーを、コンソール構成画面に表示されている順に実行します。応答文書が入力として最初のハンドラーに渡され、戻された処理済みの文書が入力として次のハンドラーに渡される、という流れです。

8. 処理の完了

同期要求の場合、応答文書がトランスポートを介して発生元の取引先に戻されます。受信側はフレームワークで setResponseStatus メソッドを呼び出し、応答のデリバリーが成功したか失敗したかを報告します。その後、受信側は要求メッセージをトランスポートから除去します。

例外

エラー条件が、次の場合に発生することがあります。

- トランスポートからのメッセージの検索に失敗
- 前処理の呼び出しに失敗
- 同期検査に失敗
- 非同期または同期処理実行の呼び出しに失敗
- 応答文書の後処理の呼び出しに失敗
- トランスポートに応答文書を戻そうとして失敗

これらのいずれかの障害が発生すると、受信側は次のアクションを実行します。

トランスポートからのメッセージを拒否

メッセージをトランスポートから除去する必要があります。例えば、JMS 受信側の場合、メッセージはキューから除去されます。HTTP 受信側の場合は、状況コード 500 が取引先に戻されます。

拒否されたメッセージのアーカイブ

これはオプションのステップです。メッセージは、後で再発信されるキューか、またはローカル・ファイル・システムの拒否済みフォルダーにアーカイブされます。後者の場合、フレームワークは拒否された要求文書について該当するストレージ・ファイルを生成できることがあります。

イベントの生成

これはオプションのステップです。受信側開発者は、受信側にイベントを作成させるか、またはエラー条件の場合にアラートを発生させるか (またはその両方) を選択できます。イベントには 3 つのレベル (通知、警告、およびエラー) があります。通知イベントが実行フローに提供するのは、その詳細だけです。警告イベントは、実行フローを停止させるほどではない問題を記録します。エラー・イベントは、文書の処理がエラーで終わり、それ以上の

文書の処理が停止したときに記録されます。例えば、同期要求で、フレームワークから受信した応答文書を受信側が発生元の取引先に戻すことができない場合、エラー・イベントが記録されます。受信側ステージで問題を記録するために使用できるイベントのリストは、後述の API の章にあります。

受信側タイプ

トランスポートでの着信メッセージの検出方法により、一般的な 2 つの受信側タイプがあります。 1 つの受信側タイプはポーリング・ベースで、トランスポートを定期的にポーリングして、新規のメッセージが到着したかどうかを判別します。この受信側タイプの Connect 提供のサンプルには、JMS と FTP があります。もう 1 つの受信側タイプはコールバック・ベースで、メッセージが到着すると、トランスポートから通知を受信します。HTTP 受信側は、コールバック・ベース受信側の例です。

注: 受信側はマルチボックス・モードで配置できます。この場合、複数の受信側とその構成済みターゲットは、同じトランスポート・ロケーションからメッセージを取り出します。このような配置モデルでは、並行管理アクセス調整機能が受信側に組み込まれている必要があります。

受信側アーキテクチャー

受信側の開発は、次の 2 つの主要部分をベースとします。つまり、API に `ReceiverInterface` インターフェースで表される受信側自体と、API に `ReceiverFrameworkInterface` インターフェースで表される受信側フレームワークです。フレームワークは、受信側と WBI-C コード本体との間のインターフェースを提供します。受信側はフレームワーク上のメソッドを呼び出して、さまざまな処理ステップを実現します。

API に指定されるその他のコンポーネントには次のものがあります。 Community Console に作成されたさまざまな構成属性を照会および設定する方法の `ReceiverConfig`、受信側がトランスポート・メッセージから作成する受信側要求文書である `ReceiverDocumentInterface`、非ブロッキング同期シナリオで同期を維持するために用意されたメカニズムである `ResponseCorrelation`、汎用例外クラスの `BCGReceiverException`、および Framework オブジェクトの取得、要求文書の作成、適切なファイル保管情報の検出のための静的メソッドを提供するユーティリティー・クラスである `BCGReceiverUtil` などがあります。これらはすべて、受信側 API の章で詳しく説明されています。

新規の受信側ハンドラー作成の概要

受信側は、受信側処理フロー内の、前処理、同期検査、および後処理の 3 つの時点で、受信側フレームワークに特殊な処理を行うよう要求することができます。文書処理フローにおけるこれらのポイントは、**構成ポイント**と呼ばれます。前処理では、文書がメインの処理システムに送信される前に実行が必要な処理を対象とし、発生元の取引先から 1 つのラッパーで送信された複数のメッセージの分離などの処理が含まれます。同期検査では、文書を同期または非同期要求として処理することを判別します。後処理では、同期要求の結果として Document Manager から戻される応答文書についての必要な処理を行います。

フレームワークは、ハンドラーに基づいてこれらの処理要求を実行します。ユーザーはその特定のニーズを満たすため、バージョン 4.2.2 に付属の API を使用してハンドラーを開発することができます。このリリースでは、ハンドラーをサポートできる WBI-C 提供の受信側は HTTP 受信側だけです。ハンドラーを作成し、配置したら、ユーザーはコンソールを使用してこれらを構成する必要があります。このプロセスの詳細については、「*ハブ構成ガイド*」を参照してください。

開発と配置

以下のセクションでは、ユーザー作成の受信側とハンドラーの両方の開発および配置について説明します。

開発環境

受信側および受信側ハンドラーの開発 API は、次の 2 つのパッケージにあるクラスとインターフェースに基づきます。

- `com.ibm.bcg.bcgdk.receiver`
- `com.ibm.bcg.bcgdk.common`

これらのパッケージは、次の WBI-C インストール可能ディレクトリーに存在する bcgsdk.jar の一部です。

- `<install dir>¥router¥was¥wbic`
- `<install dir>¥receiver¥was¥wbic`
- `<install dir>¥console¥was¥wbic`

すべての配置済みインスタンスで、この .jar ファイルはアプリケーション・サーバー・クラスパスにインストールされています。

開発のためには、bcgsdk.jar ファイルはユーザー出口クラスを含むプロジェクトのビルド・パス (つまり、クラスパス) に組み込まれている必要があります。

配置とパッケージ

カスタム受信側やカスタム・ハンドラーなどのすべてのユーザー作成コードは、次のいずれかの方法でパッケージ化し、配置する必要があります。

- `¥receiver¥was¥wbic¥userexits` の JAR ファイルに配置
- `¥receiver¥was¥wbic¥userexits¥classes` のクラスとして追加

さらに、すべてのハンドラー・クラスには、ハンドラーに関する重要なデータを Console GUI にインポートできる XML 記述子ファイルが付属している必要があります。以下の簡潔なアウトラインにおいて、`HandlerClassName` はハンドラー・クラスの名前、 `Description` はクラスの要旨、`HandlerTypeValues` はコンポーネント名 (RECEIVER - 受信側ハンドラー、GATEWAY - 送信側ハンドラー、FIXEDWORKFLOW - 固定インバウンドまたはアウトバウンド・ハンドラー、または ACTION - 可変ワークフロー・ステップ) で、ハンドラーのタイプが後に続きます (例えば PREPROCESS.<transport>、PROTOCOL.UNPACKAGING、など)。また、`ComponentAttribute` は、ランタイムにおけるハンドラーの振る舞いを変更する、ハンドラー・クラス内の構成可能メンバー変数です。

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:HandlerDefinition
     xmlns:tns="http://www.ibm.com/websphere/bcg/2004/v0.1/import/external"
     xmlns:tns2="http://www.ibm.com/websphere/bcg/2004/v0.1/import
                                                          /external/types"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.ibm.com/websphere/bcg/2004/v0.1
                            /import/external bcghandler.xsd
                        http://www.ibm.com/websphere/bcg/2004/v0.1
                            /import/external/types bcgimport.xsd ">
 <tns:HandlerClassName>com.ibm.SampleHandler </tns:HandlerClassName>
 <tns:Description>A Sample Handler</tns:Description>
 <tns:HandlerTypes>
     <tns:HandlerTypeValue>COMPONENT.TYPE</tns:HandlerTypeValue>
 </tns:HandlerTypes>
 <tns:HandlerAttributes>
       <tns2:ComponentAttribute>
               <tns2:AttributeName>Attribute 1</tns2:AttributeName>
       </tns2:ComponentAttribute>
       <tns2:ComponentAttribute>
           <tns2:AttributeName>Attribute 2</tns2:AttributeName>
            <tns2:AttributeDefaultValue>Attribute2DefaultValue
           </tns2:AttributeDefaultValue>
            </tns2:ComponentAttribute>
 </tns:HandlerAttributes>
</tns:HandlerDefinition>
```

第 3 章 受信側および受信側ハンドラー用の API およびコード例

この章では、カスタム受信側およびカスタム受信側ハンドラーを開発するために用意されている API の注釈付きのリストを示します。次のクラスとインターフェースを説明します。

- 14 ページの『ReceiverInterface』
- 17 ページの『ReceiverDocumentInterface』
- 21 ページの『ReceiverFrameworkInterface』
- 25 ページの『ReceiverConfig』
- 27 ページの『ResponseCorrelation』
- 28 ページの『BCGReceiverException』
- 29 ページの『ReceiverPreProcessHandlerInterface』
- 31 ページの『ReceiverSyncCheckHandlerInterface』
- 33 ページの『ReceiverPostProcessHandlerInterface』
- 35 ページの『BCGReceiverUtil』
- 36 ページの『イベント』
- その他のユーティリティー、セキュリティー、およびコンポーネント間の共用クラスについては、ワークフロー API の章にあるリストも参照してください。

このリストには、カスタム受信側の実装の概要を示すコードと疑似コードの簡単な例も含まれています。

ReceiverInterface

このインターフェースは、各受信側が実装する必要があります。これには、次のメソッドがあります。

- init
- refreshConfig
- startReceiving
- processResponse
- stopReceiving

メソッド

init

メソッドの説明

ReceiverConfig オブジェクトの内容に基づいて、受信側を初期化します。

構文

```
public void init (Context context, ReceiverConfig config)
       throws BCGReceiverException
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

config コンソールで指定された構成の詳細

メソッド

refreshConfig

メソッドの説明

この受信側の構成で変更を検出すると、受信側フレームワークによって呼び出されます。

構文

```
public void refreshConfig(ReceiverConfig config)
              throws BCGReceiverException
```

パラメーター

config コンソールで指定された構成の詳細内容を持つオブジェクト

メソッド

startReceiving

メソッドの説明

スレッドで受信側フレームワークによって呼び出されます。このメソッドが呼び出されると、受信側は文書の受信を開始できます。受信側がコールバック・タイプの場合、この時点の後でのみ独自のスレッドでコールバックの処理を開始します。このメソッドは、迅速に戻るはずです。

注: 受信側は、独自のスレッド管理を行います。

構文

```
public void startReceiving()
            throws BCGReceiverException
```

パラメーター

なし

メソッド

processResponse

メソッドの説明

非ブロッキング同期要求の場合、要求文書が Document Manager から戻されると、受信側フレームワークによって呼び出されます。呼び出しはフレームワーク (または内部クラス) スレッドで行われます。受信側は、この呼び出しを迅速に戻すはずです。

構文

```
public void processResponse(ResponseCorrelation respCorr,
                    ReceiverDocumentInterface response)
            throws BCGReceiverException
```

パラメーター

respCorr
応答文書を元の要求文書に同期することが必要な情報を含むオブジェクト

response
応答文書

メソッド

stopReceiving

メソッドの説明

WBI-C 内部コンポーネントによって呼び出されます。このメソッドは、迅速に戻るはずです。メソッドが呼び出されたら、受信側は文書の受信を停止して終結処理を実行します。以下の場合に行われます。

- 内部コンポーネントが終了要求を受信した場合
- 受信側が、使用不可かまたはコンソールから除去された場合
- 受信側が、受信側フレームワークに受信側の除去を要求した場合

構文

```
public void stopReceiving()
             throws BCGReceiverException
```

パラメーター

なし

ReceiverDocumentInterface

文書を表します。このオブジェクトは、受信側がフレームワークを呼び出す前に、受信側によって作成されます。これには、次のメソッドがあります。

- getTempObject
- setTempObject
- getAttribute
- setAttribute
- getAttributes
- getTransportHeaders
- setTransportHeaders
- getDocument
- setDocument
- getDocumentUUID

メソッド

```
getTempObject
```

メソッドの説明

ハンドラー間で引き渡す一時情報を取得します。

構文

```
public Object getTempObject(String objectName)
```

パラメーター

objectName
一時情報を保持するオブジェクトの名前

メソッド

```
setTempObject
```

メソッドの説明

ハンドラー間で引き渡す一時情報を設定します。

構文

```
public void setTempObject(String objectName, Object objectValue)
```

パラメーター

objectName
一時情報を保持するオブジェクトの名前

objectValue
一時情報

メソッド

getAttribute

メソッドの説明

コンソール定義属性を取得します。

構文

```
public Object getAttribute(String name)
```

パラメーター

name 属性

メソッド

setAttribute

メソッドの説明

コンソール定義属性を設定します。

構文

```
public void setAttribute(String name, Object value)
```

パラメーター

name 属性

value 属性に設定される値

メソッド

getAttributes

メソッドの説明

属性マップ全体を取得します。

構文

```
public Map getAttributes()
```

パラメーター

なし

メソッド

getTransportHeaders

メソッドの説明

トランスポート・ヘッダーを取得します。

構文

```
public HashMap getTransportHeaders()
```

パラメーター

なし

メソッド

```
setTransportHeaders
```

メソッドの説明

トランスポート・ヘッダーを設定します。

構文

```
public setTransportHeaders(Hashmap transportHeaders)
```

パラメーター

transportHeader
トランスポート・ヘッダー

メソッド

```
getDocument
```

メソッドの説明

文書の内容をファイルとして取得します。

構文

```
public File getDocument()
```

パラメーター

なし

メソッド

```
setDocument
```

メソッドの説明

文書の内容を File オブジェクトに設定します。

構文

```
public void setDocument(File document)
```

パラメーター

document
メッセージ内容

メソッド

getDocumentUUID

メソッドの説明

この文書に固有な参照 ID を取得します。文書にはすべて固有な ID が割り当てられています。

構文

```
getDocumentUUID()
```

パラメーター

なし

ReceiverFrameworkInterface

このインターフェースで、受信側フレームワークで使用可能なメソッドを指定します。そのメソッドは次のとおりです。

- remove
- preProcess
- syncCheck
- postProcess
- process
- setResponseStatus

メソッド

remove

メソッドの説明

受信側が致命的な条件を検出すると呼び出されます。受信側が受信を続行できない場合、このメソッドを呼び出すことしか行いません。フレームワークはこの受信側に除去のマークを付けて、即時に戻します。後で、内部 WBI-C コンポーネントが受信側オブジェクトについて stopReceiving メソッドを呼び出します。

構文

```
public void remove(String transportType)
                   throws BCGReceiverException
```

パラメーター

transportType
サポートするトランスポートによって受信側を識別するストリング

メソッド

preProcess

メソッドの説明

コンソールを使用してどの前処理ハンドラー (Connect 提供またはユーザー定義) がこのターゲットに指定されているかに基づいて、文書の前処理を行うために受信側によって呼び出されます。フレームワークは、要求文書を入力として引き渡すことにより、これらのハンドラーを実行します。処理された文書は、あるハンドラーから戻されると、2 番目のハンドラーに入力として送信される、というようになります。ハンドラーは、コンソールのターゲット構成画面で指定された順序で呼び出されます。結果の文書は、配列として戻されます。

構文

```
public ReceiverDocumentInterface[] preProcess(
                                    String transportType,
                                    String target,
                                    ReceiverDocumentInterface request)
              throws BCGReceiverException
```

パラメーター

transportType
サポートするトランスポートによって受信側を識別するストリング

target ターゲット、つまり受信側構成を識別するストリング

request
処理される要求文書

メソッド

`syncCheck`

メソッドの説明

コンソールに構成された同期検査ハンドラー (ユーザー提供のハンドラーを含む) のリストにアクセスするために受信側によって呼び出されます。フレームワークは、該当するハンドラーを見付けるまでリスト内を繰り返します。*true* は、要求が同期であることを示します。*false* は、要求が非同期に構成されているか、またはこの受信側に同期検査ハンドラーが構成されていないことを示し、要求は非同期に処理されることを示します。

構文

```
public boolean syncCheck(String transportType, String target,
                         ReceiverDocumentInterface request)
                  throws BCGReceiverException
```

パラメーター

transportType
サポートするトランスポートによって受信側を識別するストリング

target ターゲット、つまり受信側構成を識別するストリング

request
処理される要求文書

メソッド

`postProcess`

メソッドの説明

同期要求の場合、受信側はフレームワークを呼び出して、コンソールを使用して、ターゲットに指定された後処理ハンドラー (Connect 提供またはユーザー定義) に基づいて、応答文書の後処理を行います。フレームワークは、応答文書を入力として引き渡すことにより、これらのハンドラーを実行します。処理された文書は、あるハンドラーから戻されると、2 番目のハンドラーに入力として送信される、というようになります。ハンドラーは、コンソールのターゲット構成画面で指定された順序で呼び出されます。結果の文書は、配列として戻されます。

構文

```
public ReceiverDocumentInterface[] postProcess(
                                    String receiverType,
                                    String target,
                                    ReceiverDocumentInterface request)
               throws BCGReceiverException
```

パラメーター

receiverType
受信側を識別するストリング

target ターゲット、つまり受信側構成を識別するストリング

request
処理される応答文書

メソッド

process

メソッドの説明

メインの処理メソッドです。呼び出されると、フレームワークは要求に対して固有な ID (UUID) を生成し、適切な入力ディレクトリーの必要な内部のファイル・セットにデータを書き込みます。要求文書にも UUID を設定します。メソッドには、要求された処理タイプ (非同期、ブロッキング同期、非ブロッキング同期) に基づき、3 つの別個のシグニチャーがあります。

注: メソッドは、一度には 1 つの要求文書のみ使用します。前処理の結果、複数の文書が存在する場合、受信側が配列内を繰り返して各文書の処理を呼び出す役割を担います。

構文

非同期要求

```
public void process(String transportType, ReceiverDocumentInterface request)
               throws BCGReceiverException
```

ブロッキング同期要求

```
public void process(String transportType,
                    ReceiverDocumentInterface request,
                    ReceiverDocumentInterface response)
               throws BCGReceiverException
```

非ブロッキング同期要求

```
public void process(String transportType,
                    ReceiverDocumentInterface request,
                    ResponseCorrelation responseCorr)
               throws BCGReceiverException
```

パラメーター

transportType
受信側を識別するストリング

request
入力文書

response
Document Manager からの応答を保持する空白文書

responseCorr
情報を保持する Response Correlation オブジェクトで、受信側が元の要求文書を Document Manager から戻される応答文書と同期化できます。

メソッド

setResponseStatus

メソッドの説明

同期応答文書が発生元のホストに戻された後、その状況をフレームワークに通知します。

構文

```
public void setResponseStatus(String documentUUID,
                        boolean status, String statusMessage)
              throws BCGReceiverException
```

パラメーター

documentUUID
文書の固有 ID

status 応答文書の状態を表すブール値

statusMessage
応答文書の状況に関連した情報

ReceiverConfig

このオブジェクトで受信側構成情報を格納します。次のメソッドがあります。

- getTransportType
- getConfigs
- getAttribute
- setAttribute
- getTargetConfig
- getTargetConfigs

メソッド

getTransportType

メソッドの説明

受信側のタイプを取得します。

構文

```
public String getTransportType()
```

パラメーター

なし

メソッド

getConfigs

メソッドの説明

受信側の構成プロパティーを取得します。

構文

```
public Map getConfigs()
```

パラメーター

なし

メソッド

getAttribute

メソッドの説明

構成プロパティーの値を取得します。

構文

```
public Object getAttribute(String configName)
```

パラメーター

configName
プロパティーの名前

メソッド

setAttribute

メソッドの説明

構成プロパティーの値を設定します。

構文

```
public void setAttribute(String configName, Object value)
```

パラメーター

configName
プロパティーの名前

value 定義

メソッド

getTargetConfig

メソッドの説明

ターゲット構成を取得します。

構文

```
public Config getTargetConfig(String targetName)
```

パラメーター

targetName
ターゲットの名前

メソッド

getTargetConfigs

メソッドの説明

すべてのターゲットの構成を取得します。

構文

```
public List getTargetConfigs()
```

パラメーター

なし

ResponseCorrelation

非ブロッキング同期処理が呼び出されたときに、要求と応答との同期に必要な情報を永続する一般的な方法は、このインターフェースが提供します。次のメソッドがあります。

- set
- get

メソッド

set

メソッドの説明

同期使用可能データを設定します。

構文

```
public Object set(Serializable key, Serializable value)
              throws NullPointerException
```

パラメーター

key 同期使用可能データのキー。例えば、JMS 受信側は JMS 相関 ID を格納するので、呼び出しは次のようになります。

```
ResponseCorrelation respCorrel = new ResponseCorrelation()
respCorrel.set (CORREL_ID_STRING, correlID);
```

トランスポート・タイプにより、格納する必要な情報に複数のタイプがある場合があります。

value 設定する値

メソッド

get

メソッドの説明

格納された同期使用可能データを取得します。

構文

```
public Object get(Serializable key)
```

パラメーター

key 同期使用可能データが格納されるキー。トランスポート・タイプにより、検索に必要な情報が複数タイプある場合があります。

BCGReceiverException

BCGException を拡張するユーティリティー・クラスです。BCGException は Exception を拡張します。これには、4 つの別個のコンストラクターがあります。

```
public class BCGReceiverException extends BCGException {
                public BCGReceiverException()
                {}

                public BCGReceiverException(String s)
                    {}

                public BCGReceiverException(String s,
                                            String args,
                                                    String eventCode,
                                                    String resourceId)
                    {}

                 public BCGReceiverException(String s,
                                                     String args,
                                                     String eventCode,
                                                     String resourceId,
                                                     Throwable th)
                    {}
}
```

パラメーター

s 例外の一般的な説明

args 例外を発生させるために使用する引き数

eventCode
例外と関連したイベント・コード

resourceID
イベントが発生したときのリソース ID

th イベントを発生させるために使用する Throwable オブジェクト

ReceiverPreProcessHandlerInterface

このインターフェースは、すべての前処理ハンドラーが実装する必要のある 3 つのメソッドを記述します。これには次が含まれます。

- init
- applies
- process

メソッド

init

メソッドの説明

Config オブジェクトの構成プロパティーを読み取ってハンドラーを初期化します。

構文

```
public void init(Context context, Config handlerConfig)
             throws BCGReceiverException
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

handlerConfig
構成情報を格納するオブジェクト

メソッド

applies

メソッドの説明

ハンドラーが要求文書を処理できるかどうかを判別します。

構文

```
public boolean applies(ReceiverDocumentInterface request)
            throws BCGReceiverException
```

パラメーター

request
要求文書

メソッド

process

メソッドの説明

前処理を実行します。文書の配列を戻します。

構文

```
public ReceiverDocumentInterface[] process(
                                        ReceiverDocumentInterface request)
              throws BCGReceiverException
```

パラメーター

request
要求文書

ReceiverSyncCheckHandlerInterface

このインターフェースは、すべての同期検査ハンドラーが実装する必要のある 3 つのメソッドを記述します。これには次が含まれます。

- init
- applies
- syncCheck

同期検査は、他のハンドラーと同様に、Community Console を使用して構成されます。

メソッド

init

メソッドの説明

Config オブジェクトの構成プロパティーを読み取ってハンドラーを初期化します。

構文

```
public void init(Context context, Config handlerConfig)
             throws BCGReceiverException
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

handlerConfig
構成情報を格納するオブジェクト

メソッド

applies

メソッドの説明

ハンドラーが要求文書を処理できるかどうかを判別します。

構文

```
public boolean applies(ReceiverDocumentInterface request)
            throws BCGReceiverException
```

パラメーター

request
要求文書

メソッド

syncCheck

メソッドの説明

文書を同期的に処理する必要があるかどうかの検査を行います。メソッドは、要求が同期の場合は *true* を戻し、非同期の場合は *false* を戻します。

構文

```
public boolean syncCheck(ReceiverDocumentInterface request)
              throws BCGReceiverException
```

パラメーター

request
要求文書

同期検査は、他のハンドラーと同様に、Community Console を使用して構成されます。

ReceiverPostProcessHandlerInterface

このインターフェースは、すべての後処理ハンドラーが実装する必要のある 3 つのメソッドを記述します。これには次が含まれます。

- init
- applies
- process

メソッド

init

メソッドの説明

Config オブジェクトの構成プロパティーを読み取ってハンドラーを初期化します。

構文

```
public void init(Context context, Config handlerConfig)
             throws BCGReceiverException
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

handlerConfig
構成情報を格納するオブジェクト

メソッド

applies

メソッドの説明

ハンドラーが応答文書を処理できるかどうかを判別します。

構文

```
public boolean applies(ReceiverDocumentInterface response)
            throws BCGReceiverException
```

パラメーター

response
応答文書

メソッド

process

メソッドの説明

応答文書に後処理を実行します。文書配列を戻します。

構文

```
public ReceiverDocumentInterface[] process(
                              ReceiverDocumentInterface response)
              throws BCGReceiverException
```

パラメーター

response
応答文書

BCGReceiverUtil

さまざまな静的ユーティリティー・メソッドを示します。

- `public static ReceiverDocumentInterface createReceiverDocument()`

  応答文書を作成します。

- `public static ReceiverFrameworkInterface getReceiverFramework()`

  受信側フレームワーク・オブジェクトを取得します。

- `public static File getTempDir()`

  着信データの一時ストレージのロケーションを取得します。

- `public static File getRejectDir()`

  拒否されたメッセージのアーカイブのロケーションを取得します。

- `public static File getArchiveDir(String receiverType, String targetName)`

  受信側のアーカイブ・ディレクトリーのロケーションを取得します。パラメーターは受信側のタイプ (http、JMS など) で、この受信側のターゲットの名前です。

イベント

以下は受信側の実行フローで使用可能なイベントのリストです。

通知イベント

BCG_103207 - 受信側入り口

イベント・テキスト: 受信側 ({0}) の入り口。
引き数の予想値: {0} - 受信側のクラス名

BCG_103208 - 受信側出口

イベント・テキスト: 受信側 ({0}) の出口。
引き数の予想値: {0} - 受信側のクラス名

警告イベント

BCG_103204 - ターゲット処理の警告

(これは、通常の操作可能な警告で、受信側の実行が予想されます。例: キュー接続を終了する場合のエラー)
イベント・テキスト: ターゲット '{0}、{1}' の文書処理警告。理由: {2}。
引き数の予想値: {0} - ターゲット名
{1} - ターゲット (受信側) タイプ
{2} - ターゲットに固有の警告理由

エラー・イベント

BCG_103203 - ターゲット処理エラー

イベント・テキスト: ターゲット '{0}、{1}' が文書の処理に失敗しました。エラー: {2}。
引き数の予想値: {0} - ターゲット名
{1} - ターゲット (受信側) タイプ
{2} - 例外メッセージ

BCG_103205 - ターゲット・エラー

(例: キュー接続を開始するときの失敗)
イベント・テキスト: ターゲット '{0}、{1}' がターゲットの処理に失敗しました: {2}。
引き数の予想値: {0} - ターゲット名
{1} - ターゲット (受信側) タイプ
{2} - 例外メッセージ

受信側実装例の概要

次のコードおよび疑似コードは、JMS 受信側の実装例の概要を示します。

```
public class CustomJMSReceiver implements ReceiverInterface {
         private Context m_context = null;
         private ReceiverConfig m_rcvConfig = null;
         public void init(Context context, ReceiverConfig receiverConfig) {
           this.m_context = context;
           this.m_rcvConfig = receiverConfig;
return;
}
public void refreshConfig(ReceiverConfig rcvconfig)
                 throws BCGReceiverException {
          this.m_rcvConfig = rcvconfig;

// Check which receiver targets are updated, added newly or deleted
// If new target is added, create a new thread and start polling the target
// If current target is updated, stop the thread which is polling the
// target, and using the updated configuration information, start polling
// If current target is deleted, stop the thread which is polling the
// target and delete the thread which is responsible for polling the
//target.
...
return;
}

 public void startReceiving() throws BCGReceiverException {
// Read the list of targets in the ReceiverConfig object
// For each target create a UserJMSThread and start the thread
return;
}
public void processResponse(ResponseCorrelation respCorr,
                            ReceiverDocumentInterface response)
                    throws BCGReceiverException {
// get the correlation information like reply-to-queue, correlation id
// and send the response to that queue
return;
}
public void stopReceiving() throws BCGReceiverException {
// get the list of UserJMSReceiverThreads associated with each target
// call stop method.
...
return;
}
private class UserJMSReceiverThread extends Thread {
 public UserJMSReceiverThread(Config targetConfig) {
// create the queue session, connection, queue receiver
...
}
public void run() {
 while (true) {
       try{
          // call receive method on the queue
          // if message is available read the message and process the
          // document

             processDocument(data);

          // else continue to poll the queue.

        ...

       } catch(Exception e) {

          ...
      }
```

```
  }
}
//Upon receiving the document from the queue, start processing the
//documenting using Receiver FrameWork APIs

public void processDocument(byte[] data) throws BCGReceiverException{
             //Get the temporary location where data can be written
         File tempDir = BCGReceiverUtil.getTempDir();
            // Now create the temp file and write the data into it
         File fileLocation = new File(tempDir, fileStr);
         FileOutputStream fos = new FileOutStream(fileLocation);
         fow.write(data);
         fos.close();

// Create the ReceiverDocument object
        ReceiverDocumentInterface request =
                              BCGReceiverUtil.createReceiverDocument();
       // set document, transport headers and BCG headers in the
       // request
        request.setDocument(fileLocation);
     ...
     //Now start processing the document using ReceiverFrameWork APIs
         ReceiverFrameWorkInterface rcvFramework =
                                 BCGReceiverUtil.getReceiverFramework();

         ReceiverDocumentInterface requestDocs[] =
               rcvFramework.preprocess(transportType,target,request);
     //Check if the requestDocs length is 1, if yes document is not
     //split into multiple documents
        boolean sync =
                     rcvFramework.syncCheck(transportType,target,request);
     ...
        if(!sync) {
           //request is not synchronous message
             rcvFramework.process(transportType,target,request);
     }
 }
```

第 4 章 固定ワークフローおよび可変ワークフローのカスタマイズ

Document Manager コンポーネントの中核であるビジネス・プロセス・エンジン (BPE) は、WBI Connect プロセスの中心で、メイン変形を実行します。 BPE はこのために、固定インバウンド・ワークフロー、可変ワークフロー、および固定アウトバウンド・ワークフローの 3 つの基本単位にグループ化される、一連の識別処理ステップで行います。ユーザー出口により、ユーザー定義プロセスをこれらの各基本単位にプラグインすることができます。

固定インバウンドおよび固定アウトバウンド・ワークフローは、メイン処理ステージに出入りするとき、文書すべてに対して行われる標準処理をカバーしています。処理ステップの数とタイプが常に同じであるため、これらは「固定」と呼ばれます。メイン処理は可変ワークフローで行われますが、ここでは、処理ステップの数とタイプは特定の状態における処理要件に完全に依存します。 4.2.2 リリースでは、固定ワークフローでの特定ステップ用のカスタム・ハンドラーの作成、また可変ワークフロー・ステージでの新規アクション (ステップのシーケンス) の定義、という 2 つの方法で、ユーザーは WBI-Connect 処理のワークフロー・ステージをカスタマイズできます。この章では、ワークフローのカスタマイズについて、この 2 つの方法を説明します。

- 『固定インバウンド・ワークフローにおけるハンドラー作成の概要』
- 42 ページの『可変ワークフローにおけるアクション作成の概要』
- 46 ページの『固定アウトバウンド・ワークフローにおけるハンドラー作成の概要』

追加セクションで、開発および配置について説明します。

- 47 ページの『開発と配置』

API リストとサンプル・コードは、次の章で説明します。このリストには、すべての WBI-C コンポーネントに共通の、多くのユーティリティー、セキュリティー、およびクラスに関する情報もあります。

固定インバウンド・ワークフローにおけるハンドラー作成の概要

着信メッセージを処理できるようにするには、それらがバンドルされているパッケージ化すると共に、プロトコル特定情報すべてを抽出し、処理しておく必要があります。固定インバウンド・ワークフローにおける最初の 2 つのステップは、これらのタスクの実行を目的としています。 WBI-C には、RNIF、AS2、MIME、EAI、および NONE パッケージ化、さらに XML、RosettaNet、および EDI プロトコルを処理するためのコードが用意されています。新しいパッケージ化タイプを追加したり新しいプロトコルをサポートするため、ユーザーは、このリリースで提供されている API `BusinessProcessHandlerInterface` を使用し、独自のハンドラーを作成することができます。これらの新規ハンドラーは、Community Console を使用して構成し、また通常の方法で処理フローに統合する必要があります。構成プロセスの詳細については、「*ハブ構成ガイド*」を参照してください。このセクションには、固定

インバウンド・ワークフロー・プロセスでの 2 つのユーザー・カスタマイズ可能ステップに関する機能の概要があります。次について説明します。

- 『プロトコル・アンパック・ハンドラー』
- 41 ページの『プロトコル処理ハンドラー』

プロトコル・アンパック・ハンドラー

BPE で受信されたメッセージは、BusinessDocumentInterface を実装しているオブジェクトにラップされます。このラッパー・オブジェクト (多くの場合、単にビジネス文書と呼ばれる) には、受信側プロセスの間に追加されたトランスポート・レベルおよび WBI-C 定義ヘッダーを含む受信済みメッセージ、またメッセージに関連するさまざまなメタデータの両方が入っています。 BPE は、その処理でこのメタデータを使用します。固定処理の最初のステップは、ビジネス文書のアンパック (パッケージから取り出すこと) です。 ここには、以下のいくつか、またはすべてのステップが含まれます。

暗号化解除
メッセージが暗号化されている場合

圧縮解除
メッセージが圧縮されている場合

シグニチャー検証
メッセージに署名がある場合

ルーティング情報抽出

- パッケージ化で提供されている場合、元のビジネス ID、宛先ビジネス ID、および発信元のビジネス ID
- 元パッケージおよびバージョン (例えば RNIF, v02.00)

ユーザー認証
パッケージ化でユーザー情報が提供されている場合

ビジネス文書パーツ抽出
パッケージ化でさまざまなメッセージ・パーツ (ペイロード、添付など) のロケーションが指定されている場合

注: ユーザー定義ハンドラーで暗号化解除およびシグニチャー検証をサポートするため、API で使用可能なセキュリティー・サービス・ユーティリティー・クラスがあります。

完了すると、このステップはメッセージをアンパックします。さらにハンドラーは、ビジネス文書内のメタ情報を以下の方法で更新しなければなりません。

- **パッケージ・レベル情報属性の更新**:

パッケージ化属性名	説明
FRPACKAGINGNAME	元パッケージ名。 Console に定義されているパッケージ化の名前でなければなりません。
FRPACKAGINGVER	元パッケージ・バージョン。 Console に定義されているバージョンでなければなりません。
PKG_FRBUSINESSID	パッケージ化で送信側のビジネス ID が提供されている場合、ここに指定しなければなりません。

パッケージ化属性名	説明
PKG_TOBUSINESSID	パッケージ化で宛先のビジネス ID が提供されている場合、ここに指定しなければなりません。

- **文書の状況の更新**: ハンドラーがビジネス文書をアンパックできない、あるいは他の致命的エラーがある場合、ハンドラーは以下を行う必要があります。
  - イベントをビジネス文書に追加
  - 文書の状態を「失敗」に設定
  - BCGUserException または BCGException から派生した例外をスロー
- **イベントの生成 (オプション)**: ハンドラーは、文書関連のイベントを以下の 3 つのカテゴリーで生成することがあります。
  - 通知: これらのイベントは、実行フローにおける追加情報を提供します。文書の処理には影響しません。
  - 警告: 処理フローを停止させるほどではない問題。例として、文書が受信および処理された後、キュー接続をクローズしているときにスローされた例外があります。
  - エラー: それ以上の文書の処理を停止させる問題。

  ユーザー出口が制御を BPE に戻すと、BPE はビジネス文書に新しいイベントがないかどうか調べます。 BPE はイベントを記録し、WBI-C はアクティビティー・テーブルを更新、また必要なアラートを生成します。ワークフロー・ステージで問題を記録するために使用できるイベントのリストは、後述の API の章にあります。

プロトコル処理ハンドラー

アンパック・ハンドラーによって処理された後、ビジネス文書は 2 番目の固定ステップ、プロトコル処理ハンドラーに渡されます。このハンドラーはビジネス文書を解析し、以下の項目を調べます。

ルーティング情報
送信側 ID および宛先 ID

プロトコル情報
そのビジネス文書に関連するビジネス・プロトコルおよびバージョン。例: Rosettanet version V02.02

文書フロー/プロセス情報
そのビジネス文書に関連する文書フローおよびバージョン。例: 3A4 version V02.02

この情報が見つかると、ハンドラーは、ビジネス文書内のメタ情報を以下の方法で更新しなければなりません。

- **プロトコル・レベル情報属性の更新**:

プロトコル属性名	説明
FRBUSINESSID	メッセージからの元ビジネス ID
TOBUISNESSID	メッセージからの宛先ビジネス ID

プロトコル属性名	説明
INITBUSINESSID	これは FRBUSINESSID と同じ場合もあれば、プロトコルによっては異なる ID になる場合もあります。
FRPROTOCOLNAME	着信ビジネス文書に関連するプロトコル名。 Console に定義されている有効なプロセス名でなければなりません。例として、Rosettanet があります。
FRPROTOCOLVER	着信ビジネス文書に関連するプロトコル・バージョン。 Console に定義されている有効なプロトコル・バージョンでなければなりません。例として、V02.00 があります。
FRPROCESSCD	着信ビジネス文書に関連するプロセス・コード。 Console に定義されている有効なコードでなければなりません。例として、3A4 があります。
FRPROCESSVER	着信ビジネス文書に関連するプロセス・バージョン。 Console に定義されている有効なプロセス・バージョンでなければなりません。例として、V02.00 があります。

- **文書の状況の更新**: ハンドラーがビジネス文書を解析できない、あるいは他の致命的エラーがある場合、ハンドラーは以下の処理を行います。
  - イベントをビジネス文書に追加
  - 文書の状態を「失敗」に設定
  - BCGUserException または BCGException から派生した例外をスロー
- **イベントの生成 (オプション)**: ハンドラーは、文書関連のイベントを以下の 3 つのカテゴリーで生成することがあります。
  - 通知: これらのイベントは、実行フローにおける追加情報を提供します。文書の処理には影響しません。
  - 警告: 処理フローを停止させるほどではない問題。例として、文書が受信および処理された後、キュー接続をクローズしているときにスローされた例外があります。
  - エラー: それ以上の文書の処理を停止させる問題。

  ユーザー出口が制御を BPE に戻すと、BPE はビジネス文書に新しいイベントがないかどうか調べます。 BPE はイベントを記録し、WBI-C はアクティビティー・テーブルを更新、また必要なアラートを生成します。ワークフロー・ステージで問題を記録するために使用できるイベントのリストは、後述の API の章にあります。

可変ワークフローにおけるアクション作成の概要

インバウンド・ワークフロー・パスが完了すると、参加者接続の参照によって適切な可変ワークフロー・パスが決定します。このワークフロー・パス (アクション) は、Console 構成プロセスに指定されています。参加者接続および可変ワークフローを構成するためのコンソールの使用方法の詳細については、「*ハブ構成ガイド*」を参照してください。

アクションとは、シーケンスに整列された複数のステップです。 WBI-C には、11 個の定義済みアクションが用意されています。その他のオプションが必要であれば、ユーザーは新規のアクションを定義することで可変ワークフローをカスタマイズできます。この定義には、新規のシーケンスに配置する全く新しい一連のステップを開発するか、あるいは既存のアクションをコピーして、そのアクションにステップを追加したり、既存のステップを削除/ 置換したり、ステップの順序を変更するなどがあります。

注: 内部的な WBI-C 特定の目的に使用されている可能性があるため、WBI-C が用意したステップすべてを新しいユーザー定義に使用できるわけではありません。詳しくは、44 ページの『テンプレートとしての WBI-C 提供アクション、およびその状況』を参照してください。

アクションは、一連のステップから構成されます。通常、ステップには、次のタイプが含まれます。

- **検証**: ビジネス文書のフォームを検査します。例えば、XML 文書は XML スキーマに対して検証できます。WBI-C には、XML 検証ステップが用意されていますが、その他については作成することができます。
- **変形**: ビジネス文書のフォームを変更します。XML 文書は、XSLT を使用して別の XML 文書に変形できます。WBI-C には、XML 変形ステップが用意されていますが、その他については作成することができます。
- **変換**: ビジネス文書のフォーマット全体をあるタイプから別のタイプに変更します。
- **プロセス・ロギング**: この文書に固有のログを作成します。一部のビジネス・プロトコルには、固有のロギング要件があります。
- **シーケンス検査**: 文書のシーケンスを検査します。一部のビジネス・プロトコルには、固有の順序付け要件があります。

注: これらのステップでは、典型的な例のみを示します。可変ワークフローは、ビジネス・プロセス・ロジックの実装を意図しています。ロジックで、実際に必要なステップを指示します。

ステップの作成

ステップの作成は、2 つの部分からなるプロセスです。実際の処理ロジックは、`BusinessProcessInterface` インターフェースを実装しているクラスにあります。ただし BPE は、別のユーザー提供クラス (`BusinessProcessFactoryInterface` インターフェースを実装している Factory クラス) で `getBusinessProcess` メソッドを呼び出すことによって、これらのオブジェクトのいずれかを取得し、処理ロジック・オブジェクトの組み立てを行います。ステップは、単一の Factory クラスによって定義されています。

定義されると、ステップは Community Console で指定されている順に並べられます。詳細については、「*ハブ構成ガイド*」を参照してください。

固定インバウンド・ハンドラーの場合のように、致命的な可変ワークフロー処理エラーによって、ビジネス文書内で更新 (その状態が「失敗」に設定され、イベントが設定される) が行われます。 `BCGUserException` または `BCGException` もスロー

されます。必要に応じて、その他のイベントも同様に設定される場合があります。ワークフロー・ステージで問題を記録するために使用できるイベントのリストは、後述の API の章にあります。

テンプレートとしての WBI-C 提供アクション、およびその状況

WBI-C には、11 個の定義済みアクションが用意されています。すべてではありませんが、これらのアクションのいくつか、またそれらを作成するステップをユーザーがカスタマイズするために使用することができます。提供されているアクション、およびユーザー・カスタマイズ・アクションにそれらを使用できる程度を次のリストに示します。

1. パススルー

このアクションは、コピーして変更することができます。個々の 2 つの既存ステップを変更することはできませんが、追加ステップ (例えば XML 検証ステップ) をアクション・シーケンスの前に付加することができます。

2. RN プロセスの HubOwner 取り消し

このアクションをコピーして変更することはできません。これは、RosettaNet プロトコルに特定のものです。ただし、この中の `ValidationFactory` ステップをユーザー・アクションで再使用することができます。

3. RN パススルー (プロセス・ロギングあり)

このアクションは、コピーして変更することができます。提供されたままの状態では、このアクションは RosettaNet 文書フローのロギング用です。初期ステップ (`ProcessLoggingFactory`) をユーザー定義ステップで置き換えることによって、ユーザー定義のプロトコル文書フローのロギング用に変更できます。

4. RN と RNSC の間の双方向変換

このアクションをコピーして変更することはできません。これは、RNIF メッセージのために使用されます。 `ValidationFactory` ステップをユーザー・アクションで再使用することができます。

5 RN と XML の間の双方向変換

このアクションをコピーして変更することはできません。これは、RNIF メッセージのために使用されます。ただし、`ValidationFactory` および `OutboundValidationFactory` ステップをユーザー・アクションで再使用することができます。

6. カスタム XML の双方向変換 (検証あり)

このアクションは、コピーして変更することができます。提供されている 3 つのステップ `ValidationFactory`、`XSLTTranslationFactory`、および `OutboundValidationFactory` をユーザー定義ステップで置き換えることができます。

`ValidationFactory` ステップは、受信したカスタム XML 文書を検証します。置換ステップでは、X_PAYLOAD_ROOT_TAG (x-aux-payload-root-tag) を既存の XML 文書名に設定し、成功した場合、検証成功イベントを配置します。

XSLTTranslationFactory は、受信した XML 文書をそのアウトバウンド XML フォーマットに変形します。置換ステップでは、TRANSFORMED_DOC 属性を 'true' に設定し、存在する場合に、変形された文書をビジネス文書に戻す必要があります。

OutboundValidationFactory は、変形された文書を検証します。置換ステップでは、X_PAYLOAD_ROOT_TAG (x-aux-payload-root-tag) を既存の XML 文書名に設定し、成功した場合の検証成功イベントを配置します。

前述の置換可能ステップはすべて、ユーザー・アクションでも使用できます。

7. カスタム XML の双方向変換 (重複検査および検証あり)

このアクションは、コピーして変更することができます。提供されている 3 つのステップ ValidationFactory、XSLTTranslationFactory、および OutboundValidationFactory をユーザー定義ステップで置き換えることができます。

ValidationFactory ステップは、受信したカスタム XML 文書を検証します。置換ステップでは、X_PAYLOAD_ROOT_TAG (x-aux-payload-root-tag) を既存の XML 文書名に設定し、成功した場合、検証成功イベントを配置します。

XSLTTranslationFactory は、受信した XML 文書をそのアウトバウンド XML フォーマットに変形します。置換ステップでは、TRANSFORMED_DOC 属性を 'true' に設定し、存在する場合に、変形された文書をビジネス文書に戻す必要があります。

OutboundValidationFactory は、変形された文書を検証します。置換ステップでは、X_PAYLOAD_ROOT_TAG (x-aux-payload-root-tag) を既存の XML 文書名に設定し、成功した場合の検証成功イベントを配置します。

前述の置換可能ステップはすべて、ユーザー・アクションでも使用できます。また、ContentDuplicateProcessFactory もユーザー・アクションで使用できます。

8. 所有者カスタム XML から RN への双方向変換 (重複検査および検証あり)

このアクションをコピーまたは変更することはできません。これは、RNIF メッセージに特定のものです。ただし、ValidationFactory および ContentDuplicateProcessFactory ステップをユーザー・アクションで使用することができます。

9. カスタム XML パススルー (重複検査および検証あり)

このアクションは、コピーして変更することができます。 ValidationFactory ステップをユーザー定義ステップで置き換えることができます。

ValidationFactory ステップは、受信したカスタム XML 文書を検証します。置換ステップでは、X_PAYLOAD_ROOT_TAG (x-aux-payload-root-tag) を既存の XML 文書名に設定し、成功した場合、検証成功イベントを配置します。

ここに含まれている ValidationFactory ステップおよび ContentDuplicateProcessFactory ステップをユーザー・アクションで使用することができます。

10. カスタム XML パススルー (重複検査あり)

このアクションをコピーして変更することはできません。ただし、ContentDuplicationProcessFactory ステップをユーザー・アクションで使用することができます。

11. カスタム XML パススルー (検証あり)

このアクションは、コピーして変更することができます。 ValidationFactory ステップをユーザー定義ステップで置き換えることができます。

ValidationFactory ステップは、受信したカスタム XML 文書を検証します。置換ステップでは、X_PAYLOAD_ROOT_TAG (x-aux-payload-root-tag) を既存の XML 文書名に設定し、成功した場合、検証成功イベントを配置します。

ここに含まれる ValidationFactory ステップをユーザー・アクションで使用することができます。

固定アウトバウンド・ワークフローにおけるハンドラー作成の概要

BPE での処理の最後のステップは、宛先パッケージ化設定に指定されているように、アウトバウンド・メッセージをパッケージ化する (パッケージに組み込む) ことです。 WBI-C には、RNIF、EAI、AS、および NONE パッケージ化、また cXML および SOAP プロトコル用のハンドラーが用意されています。新しいパッケージ化 (例えば、添付付き SOAP のパッケージ化) を追加するため、ユーザーは、このリリースで提供されている API BusinessProcessHandlerInterface を使用し、独自のハンドラーを作成することができます。このパッケージ化ハンドラーは、Community Console を使用して構成し、また通常の方法で処理フローに統合する必要があります。構成プロセスの詳細については、「*ハブ構成ガイド*」を参照してください。このセクションには、固定アウトバウンド・ワークフロー・プロセスでのユーザー・カスタマイズ可能ステップに関する機能の概要があります。次について説明します。

- 『プロトコル・パッケージ化ハンドラー』

プロトコル・パッケージ化ハンドラー

プロトコル・パッケージ化には、以下の 1 つまたは複数のステップが含まれます。

アセンブルまたはエンベロープ
ビジネス・プロトコルが、メッセージを異なるパーツ (ペイロード、添付など) としてパッケージすることを必要としている場合

暗号化 パッケージ化タイプが暗号化を必要としている場合

署名 パッケージ化タイプがシグニチャーを必要としている場合

圧縮 パッケージ化タイプが圧縮を必要としている場合

トランスポート・ヘッダーの指定
該当する場合

このプロセスの出力は、パッケージ化されたメッセージです。メッセージは in_process ディレクトリーに書き込まれ、処理および宛先への伝送のため、そこで取り出されて送信側コンポーネントに渡されます。そのロケーションは、ビジネス文書オブジェクト内で更新されます。またハンドラーは、ビジネス文書内のメタ情報を以下の方法で更新する必要があります。

- **プロトコル・パッケージ化属性の更新**:

プロトコル属性名	説明
OUTBOUNDTRANSPORTHEADERS	この属性の値は、アウトバウンド・メッセージに設定されているトランスポート・ヘッダーのリストが入った HashMap オブジェクトです。例えば、RNIF 2.0 メッセージが取引先に送信されると、以下の RN ヘッダーが設定されます。 • x-rn-response-type=async • x-rn-version=RosettaNet/V02.00

- **文書の状況の更新**: ハンドラーがメッセージをパッケージ化できない、あるいは他の致命的エラーがある場合、ハンドラーは以下を行います。
  - イベントをビジネス文書に追加
  - 文書の状態を「失敗」に設定
  - BCGWorkflowException から派生した例外をスロー
- **イベントの生成 (オプション)**: 可変ワークフローで使用できるイベントのリストは、ワークフロー API の章にあります。ユーザー出口が制御を BPE に戻すと、BPE はビジネス文書に新しいイベントがないかどうか調べます。 BPE はイベントを記録し、WBI-C はアクティビティー・テーブルを更新、また必要なアラートを生成します。

開発と配置

以下のセクションでは、固定ワークフローにおけるユーザー作成のハンドラーと可変ワークフローにおけるユーザー作成のステップの両方の開発および配置について説明します。

開発環境

ワークフローの開発 API は、次の 3 つのパッケージにあるクラスとインターフェースに基づきます。

- com.ibm.bdg.bcgdk.workflow
- com.ibm.bcg.bcgdk.common
- com.ibm.bcg.bcgdk.services

これらのパッケージは、次の WBI-C インストール可能ディレクトリーに存在する bcgsdk.jar の一部です。

- <*install dir*>¥router¥was¥wbic
- <*install dir*>¥receiver¥was¥wbic
- <*install dir*>¥console¥was¥wbic

すべての配置済みインスタンスで、この .jar ファイルはモジュール・クラスパスでなく、アプリケーション・サーバー・クラスパスで使用できる必要があります。

開発のためには、bcgsdk.jar ファイルはユーザー出口クラスを含むプロジェクトのビルド・パス (つまり、クラスパス) に組み込まれている必要があります。

配置とパッケージ

カスタム・ハンドラーやカスタム・ステップなどのすべてのユーザー作成コードは、次のいずれかの方法でパッケージ化し、配置する必要があります。

- ¥router¥was¥wbic¥userexits の .jar ファイルに配置
- ¥router¥was¥wbic¥userexits¥classes のクラスとして追加

さらに、すべてのハンドラー・クラスには、ハンドラーに関する重要なデータを Console GUI にインポートできる XML 記述子ファイルが付属している必要があります。以下の簡潔なアウトラインにおいて、HandlerClassName はハンドラー・クラスの名前、 Description はクラスの要旨、HandlerTypeValues はコンポーネント名 (RECEIVER - 受信側ハンドラー、GATEWAY - 送信側ハンドラー、FIXEDWORKFLOW - 固定インバウンドまたはアウトバウンド・ハンドラー、または ACTION - 可変ワークフロー・ステップ) で、ハンドラーのタイプが後に続きます (例えば PREPROCESS.<transport>、PROTOCOL.UNPACKAGING、など)。また、ComponentAttribute は、ランタイムにおけるハンドラーの振る舞いを変更する、ハンドラー・クラス内の構成可能メンバー変数です。

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:HandlerDefinition
     xmlns:tns="http://www.ibm.com/websphere/bcg/2004/v0.1/import/external"
     xmlns:tns2="http://www.ibm.com/websphere/bcg/2004/v0.1/import
                                                        /external/types"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.ibm.com/websphere/bcg/2004/v0.1
                              /import/external bcghandler.xsd
                         http://www.ibm.com/websphere/bcg/2004/v0.1
                             /import/external/types bcgimport.xsd ">
 <tns:HandlerClassName>com.ibm.SampleHandler </tns:HandlerClassName>
 <tns:Description>A Sample Handler</tns:Description>
 <tns:HandlerTypes>
     <tns:HandlerTypeValue>COMPONENT.TYPE</tns:HandlerTypeValue>
 </tns:HandlerTypes>
 <tns:HandlerAttributes>
       <tns2:ComponentAttribute>
               <tns2:AttributeName>Attribute 1</tns2:AttributeName>
           </tns2:ComponentAttribute>
       <tns2:ComponentAttribute>
           <tns2:AttributeName>Attribute 2</tns2:AttributeName>
            <tns2:AttributeDefaultValue>Attribute2DefaultValue
          </tns2:AttributeDefaultValue>
           </tns2:ComponentAttribute>
 </tns:HandlerAttributes>
</tns:HandlerDefinition>
```

第 5 章 ワークフロー・ハンドラーおよびワークフロー・ステップ用の API およびコード例

この章では、カスタム・ハンドラーを開発するために用意されている API の注釈付きのリストを示します。カスタム・ハンドラーは、固定インバウンド・ワークフローと固定アウトバウンド・ワークフロー、および可変ワークフローのアクションにアセンブルするステップを処理するために用いられます。これには、ユーティリティー、セキュリティー、その他のコンポーネント間で共用する共通クラスのリストが含まれます。

次のクラスとインターフェースを説明します。

com.ibm.bcg.bcgdk.workflow パッケージ

com.ibm.bcg.bcgdk.services パッケージ

com.ibm.bcg.bcgdk.common パッケージ

ワークフロー・イベントについて

このリストには、プロトコル処理とプロトコル・アンパックのハンドラーのサンプル実装の概要を示すコードと疑似コードの簡単な例に加えて、検証と変形のステップも含まれています。検証と変形のステップのより完全なコード・サンプルは、`DevelopmentKits/UserExits/samples/` ディレクトリーの配布イメージで入手できます。これらのサンプルの詳細は、製品の README ファイルに記載されています。

com.ibm.bcg.bcgdk.workflow パッケージ

処理のワークフロー・ステージと直接関連したクラスおよびインターフェースがあります。これには次が含まれます。

BusinessProcessFactoryInterface

各可変ワークフロー・ステップはこのファクトリー・インターフェースを実装する必要があります。これには、次のメソッドがあります。

- getBusinessProcess
- returnBusinessProcess

メソッド

getBusinessProcess

メソッドの説明

BusinessProcessInterface のインスタンスを取得します。ファクトリー・クラスは、引き渡される構成情報に基づき、適切なコンストラクターを呼び出すことにより、BusinessProcess インスタンスを構成します。

構文

```
public BusinessProcessInterface getBusinessProcess(
                                    Context context,
                                    Config workflowConfig,
                                    BusinessDocumentInterface bDoc)
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

workflowConfig
コンソールで指定された構成の詳細

bDoc 処理されるビジネス文書

メソッド

returnBusinessProcess

メソッドの説明

BusinessProcess をファクトリーに戻します。BPE によって呼び出されます。

構文

```
public void returnBusinessProcess(BusinessProcessInterface bp)
```

パラメーター

bp 戻されるビジネス・プロセス

BusinessProcessInterface

各可変ワークフロー・ステップはこのインターフェースを実装する必要があります。ファクトリーが、実装されたクラスのインスタンスを生成します。このクラスは文書の実際のビジネス・ロジックを実行します。これには、次のメソッドがあります。

- process
- reset

メソッド

process

メソッドの説明

引き渡されるビジネス文書のビジネス・ロジックを実行します。

構文

```
public BusinessDocumentInterface process(
                                    Context context,
                                    BusinessDocumentInterface bDoc)
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

bDoc 処理されるビジネス文書

メソッド

reset

メソッドの説明

BusinessProcess をリセットします。BusinessProcessFactory によって呼び出されます。

構文

```
public boolean reset()
```

パラメーター

なし

BusinessProcessHandlerInterface

固定インバウンドおよび固定アウトバウンドのワークフロー・ハンドラーは、このインターフェースを実装する必要があります。これには、次の 3 つのメソッドがあります。

- init
- applies
- process

メソッド

init

メソッドの説明

Config オブジェクトの構成プロパティーを読み取ってハンドラーを初期化します。

構文

```
public void init(Context context,
                           Config config)
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

config コンソールで設定された構成情報

メソッド

applies

メソッドの説明

ハンドラーがビジネス文書を処理できるかどうかを判別します。

構文

```
public boolean applies(BusinessDocumentInterface bDoc)
```

パラメーター

bDoc 処理されるビジネス文書

メソッド

process

メソッドの説明

文書の必要なビジネス・ロジックを実行します。

構文

```
public BusinessDocumentInterface process(BusinessDocumentInterface bDoc)
```

パラメーター

bDocTerm
処理されるビジネス文書

AttachmentInterface

これは添付ファイルを処理するユーティリティー・インターフェースです。これには、次の 10 個のメソッドがあります。

- setContentType
- getContentType
- setDescription
- getDescription
- setURI
- getURI
- setEncoding
- getEncoding
- setFile
- getFile

メソッド

setContentType

メソッドの説明

添付ファイルのコンテンツ・タイプを設定します。

構文

```
public void setContentType(String contentType)
```

パラメーター

contentType
コンテンツ・タイプ

メソッド

getContentType

メソッドの説明

添付ファイルのコンテンツ・タイプを取得します。

構文

```
public String getContentType()
```

パラメーター

なし

メソッド

setDescription

メソッドの説明

添付ファイルを説明するストリングを設定します。

構文

```
public void setDescription(String desc)
```

パラメーター

desc　記述

メソッド

```
getDescription
```

メソッドの説明

記述を取得します。

構文

```
public String getDescription()
```

パラメーター

なし

メソッド

```
setURI
```

メソッドの説明

添付ファイルの URI を設定します。

構文

```
public void setURI(String URI)
```

パラメーター

URI　URI

メソッド

```
getURI
```

メソッドの説明

URI を取得します。

構文

```
public String getURI()
```

パラメーター

なし

メソッド

`setEncoding`

メソッドの説明

添付ファイルの文字エンコードを設定します。

構文

```
public void setEncoding(String encoding)
```

パラメーター

encoding
エンコード

メソッド

`getEncoding`

メソッドの説明

添付ファイルの文字エンコードを取得します。

構文

```
public String getEncoding()
```

パラメーター

なし

メソッド

`setFile`

メソッドの説明

添付ファイルのファイルを設定します。

構文

```
public void setFile(File file)
```

パラメーター

file　ファイル

メソッド

`getFile`

メソッドの説明

ファイルを取得します。

構文

```
public File getFile()
```

パラメーター

なし

BusinessProcessUtil

これは WBI-C 提供のユーティリティー・クラスです。これには、次のメソッドがあります。

- getSecurityService
- getPartnerService
- getMapService

ここでは以下の定数も指定します。

- BCG_DOCSTATE_FAILED = ″Failed″
- BCG_DOCSTATE_IN_PROCESS = ″In Process″

メソッド

```
getSecurityService
```

メソッドの説明

Security Service オブジェクトを取得します。

構文

```
public SecurityServiceInterface getSecurityService()
```

パラメーター

なし

メソッド

```
getPartnerService
```

メソッドの説明

Partner Service オブジェクトを取得します。

構文

```
public PartnerServiceInterface getPartnerService()
```

パラメーター

なし

メソッド

```
getMapService
```

メソッドの説明

MapService オブジェクトを取得します。

構文

```
public MapServiceInterface getMapService()
```

パラメーター

なし

定数

これらはビジネス文書の状況を更新するために使用されます。

```
public static final String BCG_DOCSTATE_FAILED = "Failed"
public static final String BCG_DOCSTATE_IN_PROCESS = "In Process"
```

com.ibm.bcg.bcgdk.services パッケージ

以下のインターフェースおよびクラスを使用して、セキュリティー・サービスおよびマッピング・サービスに一般的なアクセスができます。

SecurityServiceInterface

このインターフェースは、セキュリティー関連のメソッドおよび定数へのアクセスを提供します。 5 つのメソッドがあり、メソッドごとに 2 つのシグニチャーがあります。

- encryptBytes
- verifySignature
- decryptBytes
- signMessage
- generateDigest

5 つの定数もあります。

- BCG_ENC_ALG_DES =″3des″
- BCG_ENC_ALG_RC5 = ″rc5″
- BCG_ENC_ALG_RC = ″rc2-40″
- BCG_SIGN_ALG_SHA1 = ″sha1″
- BCG_SIGN_ALG_MD5 = ″md5″

メソッド

encryptBytes

メソッドの説明

入力を暗号化します。

構文

バイト配列入力

```
public byte[] encryptBytes(BusinessDocumentInterface doc,
                           byte[] buf, String alg)
          throws BCGSecurityException
```

入力ストリーム入力

```
public InputStream encryptBytes(BusinessDocumentInterface doc,
                                InputStream in, String alg)
          throws BCGSecurityException
```

パラメーター

doc ビジネス文書

buf 暗号化するデータ (バイト配列)

in 暗号化するデータ (入力ストリーム)

alg 暗号化のタイプ

メソッド

verifySignature

メソッドの説明

シグニチャーを検証します。

構文

バイト配列入力

```
public SignInfo verifySignature (BusinessDocumentInterface doc,
                                 byte[] signature, byte[] buff,
                                 String businessId,
                                 String signatureAlgo)
          throws BCGSecurityException
```

入力ストリーム入力

```
public SignInfo verifySignature (BusinessDocumentInterface doc,
                                 byte[] signature,
                                 InputStream in,
                                 String businessId,
                                 String signatureAlgo)
          throws BCGSecurityException
```

パラメーター

doc ビジネス文書

signature
検証するシグニチャー

buff シグニチャーが検証されるデータ (バイト配列)

in シグニチャーが検証されるデータ (入力ストリーム)

businessID
シグニチャーが検証されるパートナーのビジネス ID

signatureAlgo
シグニチャーで使用されるアルゴリズム

メソッド

decryptBytes

メソッドの説明

入力を暗号化解除します。

構文

バイト配列入力

```
public byte[] decryptBytes(BusinessDocumentInterface doc,
                           byte[] buff, String alg)
          throws BCGSecurityException
```

入力ストリーム入力

```
public InputStream decryptBytes(BusinessDocumentInterface doc,
                                InputStream is, String alg)
           throws BCGSecurityException
```

パラメーター

doc ビジネス文書

buf 暗号化解除するデータ (バイト配列)

is 暗号化解除するデータ (入力ストリーム)

alg 暗号化のタイプ

メソッド

signMessage

メソッドの説明

データに署名します。

構文

バイト配列入力

```
public SignInfo signMessage(BusinessDocumentInterface doc,
                            byte[] data, String alg)
        throws BCGSecurityException
```

入力ストリーム入力

```
public SignInfo signMessage(BusinessDocumentInterface document,
                             InputStream is, String alg)
        throws BCGSecurityException
```

パラメーター

doc ビジネス文書

data 署名するデータ (バイト配列)

is 署名するデータ (入力ストリーム)

alg シグニチャーのタイプ

メソッド

generateDigest

メソッドの説明

ダイジェストを生成します。

構文

バイト配列入力

```
public byte[] generateDigest(byte[] data, String alg)
              throws BCGSecurityException
```

入力ストリーム入力

```
public byte[] generateDigest(InputStream dataStream, String alg)
              throws BCGSecurityException
```

パラメーター

data　バイト配列のデータ

dataStream
入力ストリームのデータ

alg　暗号化のタイプ

SecurityService メソッドは、適用できる場合に、以下の情報でビジネス文書を更新します。

属性名	説明
SIGNING_PUBLIC_CERT_ID	文書の署名に使用する証明書 ID。
SIGNING_PRIVATE_KEY_ID	文書の署名に使用するキー ID。これは、ハブの所有者の秘密鍵です。
ENCRYPTION_PUBLIC_CERT_ID	アウトバウンド文書を暗号化するための参加者の公開鍵の ID。
ENCRYPTION_PRIVATE_KEY_ID	インバウンド文書を暗号化解除するためのハブの所有者の秘密鍵の ID。
DIGSIGALGORITHM	タイプ 'sha1' または 'md5' のメッセージへの署名に使用するアルゴリズム。
DOC_ENCRYPT_ALGO	タイプ '3des'、'rc5' などのメッセージに使用する暗号化アルゴリズム。

以下は、(上記のいくつかのメソッドで戻される) `SignInfo` クラスの例で、参照用です。

```
package com.ibm.bcg.bcgdk.services;
public class SignInfo {
 private byte[] data;// signature data
 private byte[] digest;// message digest
 public byte[] getData();
 public void setData(byte[] data);
 public byte[] getDigest();
 public void setDigest(byte[] digest);
}
```

定数

これらは、次の暗号化タイプおよびシグニチャー・タイプを定義します。

```
public final String BCG_ENC_ALG_DES="3des"
public final String BCG_ENC_ALG_RC5 = "rc5"
public final String BCG_ENC_ALG_RC2 = "rc2-40"
public final String BCG_SIGN_ALG_SHA1="sha1"
public final String BCG_SIGN_ALG_MD5 = "md5"
```

MapServiceInterface

このインターフェースは、検証マップおよび変形マップへのアクセスを提供します。以下の 3 つのメソッドがあります。

- getFromValidationMap
- getToValidationMap
- getTransformationMap

メソッド

getFromValidationMap

メソッドの説明

検証マップから該当するものを取得します。

構文

```
public byte[] getFromValidationMap(Context context,
                           BusinessDocumentInterface document)
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

document
ビジネス文書

メソッド

getToValidationMap

メソッドの説明

検証マップに該当するものを取得します。

構文

```
public byte[] getToValidationMap(Context context,
                           BusinessDocumentInterface document)
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

document
ビジネス文書

メソッド

getTransformationMap

メソッドの説明

該当する変形マップを取得します。

構文

```
public byte[] getTransformationMap(Context context,
                                   BusinessDocumentInterface document)
```

パラメーター

context

一時ディレクトリー・パスなどのランタイム情報

document

ビジネス文書

BCGSecurityException

BCGException を拡張するユーティリティー・クラスです。BCGException は Exception を拡張します。これには、2 つのコンストラクターがあります。

```
BCGSecurityException()
```

```
BCGSecurityException(String s)
```

com.ibm.bcg.bcgdk.common パッケージ

これらは、WBI-C 処理のすべてのステージに共通の汎用ユーティリティー・クラスおよびインターフェースです。これには次が含まれます。

Context

このクラスは、一時ディレクトリー・パスなどのさまざまな種類のランタイム情報の格納に使用されます。例えば、BPE は、このクラスを使用してデータベース接続オブジェクトへの参照を格納します。これには、2 つのメソッドが含まれます。

- getContext
- setContext

メソッド

```
getContext
```

メソッドの説明

格納された情報を取得します。

構文

```
public Object getContext(String contextName)
```

パラメーター

contextName
情報に関連付けられた名前

メソッド

```
setContext
```

メソッドの説明

情報を設定します。

構文

```
public void setContext(String contextName, Object context)
```

パラメーター

contextName
情報に関連付けられた名前

context
情報

Config

このクラスはターゲット構成情報を保持します。これには、4 つのメソッドがあります。

注: このクラスはスレッド・セーフではありません。

- getName
- getAttribute
- setAttribute
- getAttributes

メソッド

`getName`

メソッドの説明

ターゲットの名前を取得します。

構文

```
public String getName()
```

パラメーター

なし

メソッド

`getAttribute`

メソッドの説明

構成プロパティーの値を取得します。

構文

```
public Object getAttribute(String name)
```

パラメーター

name

プロパティーの名前

メソッド

`setAttribute`

メソッドの説明

構成プロパティーの値を設定します。

構文

```
public void setAttribute(String name, Object value)
```

パラメーター

name
プロパティーの名前

value 設定する値

メソッド

`getAttributes`

メソッドの説明

このターゲットのすべてのプロパティーを保持する Map オブジェクトを取得します。

構文

```
public Map getAttributes()
```

パラメーター

なし

BusinessDocumentInterface

このインターフェースは、処理中のビジネス文書を表します。これには、18 個のメソッドがあります。

- getDocumentUUID
- getDocumentParentUUID
- createFile
- getDocument
- setDocument
- getOriginalFile
- getDocumentState
- setDocumentState
- addEvents
- getEvents
- getAttribute
- setAttribute
- getTempObject
- setTempObject
- getAttachments
- addAttachments
- getTransportHeaders

メソッド

```
getDocumentUUID
```

メソッドの説明

この文書に関連付けられた固有 ID を取得します。

構文

```
public String getDocumentUUID()
```

パラメーター

なし

メソッド

```
getDocumentParentUUID
```

メソッドの説明

この文書の親に関連付けられた固有 ID を取得します。

構文

```
public String getDocumentParentUUID()
```

パラメーター

なし

メソッド

createFile

メソッドの説明

ビジネス文書以外の ファイルをファイル・システムに書き込む場合に使用する File オブジェクトを作成します。ビジネス文書名に基づいて名前を作成します。このメソッドは、例えば、添付ファイルの処理に使用されます。また、同期応答文書が Sender Result オブジェクトで設定される必要がある場合に、File オブジェクトの作成にも使用されます。

構文

```
public File createFile()
```

パラメーター

なし

メソッド

getDocument

メソッドの説明

WBI-C 作業ディレクトリーにあるビジネス文書のファイル参照を取得します。

構文

```
public File getDocument()
```

パラメーター

なし

メソッド

setDocument

メソッドの説明

文書を WBI-C 作業ディレクトリーに書き込みます。

構文

```
public void setDocument(File document)
```

パラメーター

document
ビジネス文書

メソッド

getOriginalFile

メソッドの説明

オリジナル・ファイルのロケーション、つまり、in_process ディレクトリー (WBI-C 作業ディレクトリー) に取り込まれたロケーションを取得します。

構文

```
public File getOriginalFile()
```

パラメーター

なし

メソッド

getDocumentState

メソッドの説明

文書の状況を取得します。

構文

```
public String getDocumentState()
```

パラメーター

なし

メソッド

setDocumentState

メソッドの説明

文書の状況を設定します。

構文

```
public String setDocumentState(String state)
```

パラメーター

state　設定する状況

メソッド

addEvents

メソッドの説明

この文書と関連付けるイベントを追加します。これらのイベントは、イベント・ビューアーおよび文書ビューアーで表示されます。

構文

```
public void addEvents(EventInfo[] events)
```

パラメーター

events
追加する EventInfo オブジェクトの配列

メソッド

```
getEvents
```

メソッドの説明

この文書と関連付けられた EventInfo オブジェクトの配列を取得します。

構文

```
public EventInfo[] getEvents()
```

パラメーター

なし

メソッド

```
clearEvents
```

メソッドの説明

これまでのイベント・セットをクリアします。BPE がビジネス・プロセスのすべてのイベントの処理を終了すると、このメソッドが呼び出され、同じイベントを再度処理することがないよう、すべてのイベントをクリアします。

構文

```
public void clearEvents()
```

パラメーター

なし

メソッド

```
getAttribute
```

メソッドの説明

指定された属性を取得します。パッケージの名前やバージョンなどの情報の取得に使用されます。使用可能な属性のリストについては、以下の DocumentConstant クラスを参照してください。

構文

```
public Object getAttribute(String attrName)
```

パラメーター

attrName
要求された属性の名前

メソッド

setAttribute

メソッドの説明

指定された属性をこの文書に設定します。使用可能な属性のリストについては、以下の DocumentConstant クラスを参照してください。

構文

```
public void setAttribute(String attrName, Object attrValue)
```

パラメーター

attrName
設定する属性の名前

attrValue
設定する値

メソッド

getTempObject

メソッドの説明

このフローに関連付けられた一時オブジェクトを取得します。

構文

```
public Object getTempObject(String objectName)
```

パラメーター

objectName
要求されたオブジェクトの名前

メソッド

setTempObject

メソッドの説明

このフローに関連付けられた一時オブジェクトを設定します。

構文

```
public void setTempObject(String objectName, Object objectValue)
```

パラメーター

objectName
設定するオブジェクトの名前

objectValue
設定する値

メソッド

`getAttachments`

メソッドの説明

この文書の添付ファイルのリストを取得します。

構文

```
public ListIterator getAttachments()
```

パラメーター

なし

メソッド

`addAttachment`

メソッドの説明

この文書に添付ファイルを追加します。

構文

```
public void addAttachment(AttachmentInterface attachment)
```

パラメーター

attachment
追加する添付ファイル

メソッド

`getTransportHeaders`

メソッドの説明

受信側によって設定されたトランスポート・ヘッダーを取得します。

構文

```
public ListIterator getTransportHeaders()
```

パラメーター

なし

exception.BCGException

これは Exception を拡張するユーティリティー・クラスです。

```
public class BCGException extends Exception {}
```

BCGUtil

このクラスは 3 つのユーティリティー・メソッドを提供し、いくつかの共通定数を定義します。メソッドは次のとおりです。

- generateUUID()
- logEvent
- トレース [2 つのシグニチャー]

定数は次のとおりです。

- BCG_TRACE_SEVERITY_DEBUG = ″Debug″
- BCG_TRACE_SEVERITY_INFO = ″Info″
- BCG_TRACE_SEVERITY_WARNING = ″Warning″
- BCG_TRACE_SEVERITY_ERROR = ″Error″
- BCG_TRACE_SEVERITY_CRITICAL = ″Critical″

メソッド

```
generateUUID()
```

メソッドの説明

固有 ID を生成します。

構文

```
public String generateUUID()
```

パラメーター

なし

メソッド

```
logEvent
```

メソッドの説明

このイベントをログに記録するかどうかを決定します。

構文

```
public boolean logEvent(EventInfo eventInfo)
```

パラメーター

eventInfo
イベント情報

メソッド

```
trace
```

メソッドの説明

トレースを設定します。

構文

例外オブジェクトがない場合

```
public void trace(String severity, String category, String msg)
```

例外オブジェクトがある場合

```
public void trace(String severity, String category, String msg, Throwable t)
```

パラメーター

severity
重大度レベルを示す定数。以下を参照してください。

category
影響を受けるモジュール名

msg トレース・メッセージ

t 例外

定数

これらはトレース重大度レベルを示します。

```
public static final String BCG_TRACE_SEVERITY_DEBUG = "Debug"
public static final String BCG_TRACE_SEVERITY_INFO = "Info"
public static final String BCG_TRACE_SEVERITY_WARNING = "Warning"
public static final String BCG_TRACE_SEVERITY_ERROR = "Error"
public static final String BCG_TRACE_SEVERITY_CRITICAL = "Critical"
```

EventInfo

このクラスはイベント情報を格納します。これは、4 つの個別の方法で初期化することができます。これには、9 つのメソッドが含まれます。

- getEventCode
- getBusinessDocument
- getDocumentUUID
- getParams
- getStackTrace
- getSourceClass
- setSourceClass
- setFaultType
- getFaultType

4 つの定数も定義します。

- FAULTTYPE_UNKNOWN = ″0″
- FAULTTYPE_SOURCE = ″1″
- FAULTTYPE_TARGET = ″2″
- FAULTTYPE_SYSTEM = ″3″

コンストラクター

オブジェクトは 4 つの別個の方法で初期化できます。

ビジネス文書を使用する

```
public EventInfo(int eventCode,
                 BusinessDocumentInterface document,
                 String[] params)
```

ビジネス文書と例外またはエラーを使用する

```
public EventInfo(int eventCode,
                 BusinessDocumentInterface document,
                 String[] params,
                 Throwable t)
```

文書 UUID を使用する

```
public EventInfo(int eventCode,
                 String documentUUID,
                 String[] params)
```

文書 UUID とエラーまたは例外を使用する

```
public EventInfo(int eventCode,
                 String documentUUID,
                 String[] params,
                 Throwable t)
```

メソッド

```
getEventCode
```

メソッドの説明

イベント・コードを取得します。

構文

```
public int getEventCode()
```

パラメーター

なし

メソッド

```
getBusinessDocument
```

メソッドの説明

ビジネス文書を取得します。

構文

```
public BusinessDocument getBusinessDocument()
```

パラメーター

なし

メソッド

```
getDocumentUUID
```

メソッドの説明

文書 UUID を取得します。

構文

```
public String getDocumentUUID
```

パラメーター

なし

メソッド

```
getParams
```

メソッドの説明

パラメーターの配列を取得します。

構文

```
public String[] getParams()
```

パラメーター

なし

メソッド

getStackTrace

メソッドの説明

スタック・トレースを取得します。

構文

```
public Throwable getStackTrace()
```

パラメーター

なし

メソッド

getSourceClass

メソッドの説明

ソース・クラスを取得します。

構文

```
public String getSourceClass()
```

パラメーター

なし

メソッド

setSourceClass

メソッドの説明

ソース・クラスを設定します。

構文

```
public void setSourceClass(String sourceClass)
```

パラメーター

sourceClass

ソース・クラス

メソッド

setFaultType

メソッドの説明

障害のタイプを設定します。以下の定数を参照してください。

構文

```
public void setFaultType(String faultType)
```

パラメーター

faultType
障害のタイプ

メソッド

```
getFaultType
```

メソッドの説明

障害のタイプを取得します。以下の定数を参照してください。

構文

```
public String getFaultType()
```

パラメーター

なし

定数

これらの定数は障害のタイプの定義に使用されます。

```
public static final String FAULTTYPE_UNKNOWN = "0"
public static final String FAULTTYPE_SOURCE = "1"
public static final String FAULTTYPE_TARGET = "2"
public static final String FAULTTYPE_SYSTEM = "3"
```

BCGDocumentConstants

このクラスは、ビジネス文書からの属性値の取得に使用する定数を設定します。

定数

```
public static final String BCG_FRBUSINESSID = "FROMBUSINESSID"
public static final String BCG_TOBUSINESSID = "TOBUSINESSID"
public static final String BCG_INITBUSINESSID = "INITIATINGBUSINESSID"
public static final String BCG_FRPROTOCOLNAME = "FROMPROTOCOLNAME"
public static final String BCG_FRPROTOCOLVER = "FROMPROTOCOLVERSION"
public static final String BCG_FRPROCESSCD = "FROMPROCESSCODE"
public static final String BCG_FRPROCESSVER = "FROMPROCESSVERSION"
public static final String BCG_INPROCESSDIR = "InProcessDirectory"
public static final String BCG_FROM_IPADDRESS = "fromIP"
public static final String BCG_INBOUND_CHARSET = "InboundCharset"
public static final String BCG_REQUEST_URI = "requestURI"
public static final String BCG_CERT_DN = "CertDN"
```

イベント

以下はワークフローの実行フローで使用可能なイベントのリストです。

通知イベント

BCG_240603 - ビジネス・プロセス入り口のパッケージ化

イベント・テキスト: ビジネス・プロセス ({0}) 入り口のパッケージ化

引き数の予想値:

{0} - BusinessProcess クラス名のパッケージ化

BCG_240604- ビジネス・プロセス出口のパッケージ化

イベント・テキスト: ビジネス・プロセス ({0}) 出口のパッケージ化

引き数の予想値:

{0} - BusinessProcess クラス名

BCG_240607 - ビジネス・プロセス入り口のアンパック

イベント・テキスト: ビジネス・プロセス ({0}) 入り口のパッケージ化

引き数の予想値:

{0} - BusinessProcess クラス名のアンパック

BCG_240608- ビジネス・プロセス出口のアンパック

イベント・テキスト: ビジネス・プロセス ({0}) 出口のパッケージ化

引き数の予想値:

{0} - BusinessProcess クラス名

BCG_240612 - ビジネス・プロセス入り口のプロトコル解析

イベント・テキスト: ビジネス・プロセス ({0}) 入り口のプロトコル解析

引き数の予想値:

{0} - BusinessProcess クラス名のプロトコル解析

BCG_240613- ビジネス・プロセス出口のプロトコル解析

イベント・テキスト: ビジネス・プロセス ({0}) 出口のプロトコル解析

引き数の予想値:

{0} - BusinessProcess クラス名

警告イベント

BCG_240605- パッケージ化の警告

イベント・テキスト: パッケージ化の警告 -{0}

引き数の予想値:

{0} - パッケージ化の警告情報

BCG_240609- アンパックの警告

イベント・テキスト: アンパックの警告 -{0}

引き数の予想値:

{0} - アンパックの警告情報

BCG_240614 - プロトコル解析の警告

イベント・テキスト: プロトコル解析の警告 -{0}

引き数の予想値:

{0} - プロトコル解析の警告情報

エラー・イベント

BCG_240418 - ダイジェスト生成の失敗

イベント・テキスト: {0}

引き数の予想値:

{0} - ダイジェスト失敗メッセージ

BCG_240419 - サポートされないシグニチャー・フォーマット (署名された受取プロトコルは pkcs7-signature ではありません)

イベント・テキスト: {0}。

引き数の予想値:

{0} - シグニチャー・フォーマットを含む例外メッセージ

BCG_240420 - サポートされないシグニチャー・アルゴリズム (シグニチャー・アルゴリズムは MD5 または SHA1 ではありません)

イベント・テキスト: {0}。

引き数の予想値:

{0} - シグニチャー・アルゴリズムを含む例外メッセージ

BCG_240606 - パッケージ化のエラー

イベント・テキスト: パッケージ化のエラー -{0}

引き数の予想値:

{0} - パッケージ化のエラー情報

BCG_240611- 暗号化の失敗

イベント・テキスト: {0}

引き数の予想値:

{0} - 暗号化の失敗メッセージ

BCG_240610 - アンパックのエラー

イベント・テキスト: アンパックのエラー -{0}

引き数の予想値:

{0} - アンパックのエラー情報

BCG_210014 - MIME メッセージのアンパックのエラー

イベント・テキスト: MIME multipart 文書のアンパックに失敗しました: {0}

引き数の予想値:

{0} - 例外メッセージ

BCG_240417 - 暗号化解除の失敗

イベント・テキスト: {0}

引き数の予想値:

{0} - 暗号化解除の失敗メッセージ

BCG_240424 - メッセージ・セキュリティー不足のエラー

イベント・テキスト: {0}。

引き数の予想値:

{0} - 欠落しているものの詳細。例えば、受信した文書は暗号化されているが、パートナー契約では暗号化と署名を必要としている、などです。

BCG_240615 - プロトコル解析エラー

イベント・テキスト: プロトコル解析エラー: -{0}

引き数の予想値:
{0} - プロトコル解析のエラー・メッセージ

ハンドラーおよびステップの実装例の概要

次のコードおよび疑似コードは、固定ワークフロー・ハンドラーおよび可変ワークフロー・ステップの実装例の概要を示します。

プロトコル処理ハンドラー

固定インバウンド・プロトコル処理ハンドラー (ここでは、CSV 処理をサポートするハンドラー) の実装の概要を示します。開発者によってプロトコル固有のコードを追加する必要があります。

```
public class MyCSVProtocolProcess implements
                                    BusinessProcessHandlerInterface {
   public boolean applies(BusinessDocumentInterface document) {
      //obtain from_protocol from BusinessDocument
       if (from_protocol.equals("CSV_PROTOCOL"))
          return true;
       return false;
      }
  public BusinessDocumentInterface process(
                                   BusinessDocumentInterface document) {
   try {
        //obtain the file contents in a String
        StringTokenizer tokenizer = new StringTokenizer(fileContents, ",");
        String fromBusinessId = tokenizer.nextToken();
        if (fromBusinessId == null) {
                                          EventInfo event = new EventInfo();
           BusinessProcessUtil.logEvent(eventInfo);
        }
        String toBusinessId = tokenizer.nextToken();
        String fromPackaging = tokenizer.nextToken();
        String customerId = tokenizer.nextToken();
        String customerName = tokenizer.nextToken();
        String documentType = tokenizer.nextToken();
        String documentVersion = tokenizer.nextToken();
         ...
        //log obtained parameters
         ...
         document.setValue(DocumentConstant.FRBUSINESSID, fromBusinessId);
         document.setValue(DocumentConstant.TOBUSINESSID, toBusinessId);
         document.setValue(DocumentConstant.INITBUSINESSID, customerId);
         document.setValue(DocumentConstant.FRPROTOCOLNAME,
                                              "CSV_PROTOCOL ");
         document.setValue(DocumentConstant.FRPROCESS, documentType);
         document.setValue(DocumentConstant.FRPROCESSVER, documentVersion);
          ...
          document.setDocumentState(BusinessProcessUtil.BCG_DOC_IN_PROCESS);
      } catch (Exception e) {
                              EventInfo event = new EventInfo();
        document.addEvent(event);
        document.setDocumentState(BusinessProcessUtil.BCG_DOC_FAILED);
         }
          return document;
      }
  }
```

プロトコル・アンパック・ハンドラー

固定インバウンド・プロトコル・アンパック・ハンドラー (ここでは、Websphere Commerce Business Edition からパッケージ化するカスタム XML をサポートするハンドラー) の実装の概要を示します。開発者によってプロトコル固有のコードを追加する必要があります。

```
public class WCBEXMLProtocolUnpackagingHandler implements
                                    BusinessProcessHandlerInterface {

   private Context m_context = null;
   private rConfig m_config = null;

   public void init(Context context,Config config) {
              this.m_context = context;
      this.m_config = config;
      return;
    }
    public boolean applies() {
          //obtain schema for this XML
          if (schemaLocation.startsWith("http://www.ibm.com/WCBE/schemas/"))
                 return true;
           return false;
    }

    public BusinessDocumentInterface process(
                                    BusinessDocumentInterface document) {
          try {
                 document.setValue(DocumentConst.FRPACKAGING, "WCBEPackaging");
                 /*
                  * Parse the file and obtain the following information:
                  * 1. Recipient Name
                  * 2. Type of document - PurchaseOrder, RFQ
                  */

            //obtain receiver_id
            PartnerService partnerService = BusinessProcessUtil.getPartnerService();
            String receiver_id = partnerService.getBusinessId(recipientName,
                                    PartnerService.COMMUNITY_PARTICIPANT);
                 if (receiver_id == null) {
             EventInfo event = new EventInfo();
                     BusinessProcessUtil.logEvent(msg);
             document.setDocumentState(
                       BusinessProcessUtil.BCG_DOCSTATE_FAILED);
                 }
                document.setValue(DocumentConstant.TOBUSINESSID, receiver_id);
                document.setValue(DocumentConst.FRPROCESSCD, documentType);

    document.setDocumentState(BusinessProcessUtil.BCG_DOCSTATE_IN_PROCESS);
       }catch (Exception e) {
            EventInfo event = new EventInfo();
            document.addEvents(event);
document.setDocumentState(BusinessProcessUtil.BCG_DOCSTATE_FAILED);
 }
         return document;
     }
}
```

検証ステップ

可変ワークフロー・ステップ (ここでは、WCBE 仕入れ注文文書を検証するステップ) の実装の概要を示します。 2 つの部分があります。最初の部分は、ファクトリー・インターフェース BusinessProcessFactoryInterface を実装し、2 番目の部分はプロセス・インターフェースBusinessProcessInterface を実装します。開発者によってプロトコル固有のコードを追加する必要があります。

ファクトリー・クラス:

```
public class WCBEXMLValidationFactory implements
                                  BusinessProcessFactoryInterface {

    public BusinessDocumentInterface getBusinessProcess(Context context,
                                                        Config config,
                                          BusinessDocumentInterface bDoc)
{

// Can use any configuration values from config as necessary.  These
//are set via the Console.

         WCBEValidationBusinessProcess bp = new WCBEValidationBusinessProcess();
            // Set any items in this class as specific to the implementation
// between the factory and the business process class.


            return bp;
     }

public static void returnBusinessProcess(BusinessProcessInterface bp)
                                     throws BCGWorkflowException {
      // if not reusing Business Processes then do nothing.
 }
}
```

プロセス・クラス:

```
public class WCBEValidationBusinessProcess implements
                                      BusinessProcessInterface {

    public BusinessDocumentInterface process(BusinessDocumentInterface bDoc,
                                       Context context) {
           /*
            * Obtain document's contents.
            */
         File document = bDoc.getDocument();
   // Read in file.


           //obtain validation map
         MapService mapService = BusinessProcessUtil.getMapService();
         byte[] fromValidationMap = mapService.getFromValidationMap(bDoc, context);

   /* Obtain a validating XML parser instance.
    * Set the validation map location in the parser.
            * Validate the XML by parsing it.
            */


/*
         * Validate the PurchaseOrder:
 *
         * if document type is PurchaseOrder
         *     check if there is at least one orderitem
         *             if there is atleast one orderitem
```

```
         *                    check if the quantity ordered is atleast one
         *                    if the quantity ordered is less than 1
         *                           set document status to DOC_FAILED
         *                           throw BCGInvalidDocumentException
         *             else
         *                     set document status to DOC_FAILED
         *                     throw BCGInvalidDocumentException
         *
         */

 return bDoc
    }

    public boolean reset() {
        /*
         * reset internal variables.
         */
     }
}
```

変形ステップ

可変ワークフロー・ステップ (ここでは、文書をあるフォーマットから別のフォーマットに変形するステップ) の実装の概要を示します。このサンプルには、プロセス実装のコードおよび疑似コードのみが含まれていますが、関連するファクトリーも必要です。開発者によってプロトコル固有のコードを追加する必要があります。

プロセス・クラス:

```
public class WCBETransformationBusinessProcess implements
                                                BusinessProcessInterface {

    public BusinessDocumentInterface process(BusinessDocumentInterface bDoc,
                                        Context context) {
            //obtain transformation map
             MapService mapService = BusinessProcessUtil.getMapService();
              byte[] transformationMap = mapService.getTransformationMap(
                                                        bDoc, context);
    // Transformer is for example only and is part of customer
    // implementation.
     Transformer transformer = new Transformer(transformationMap);

  // Get the Business document file.
    File message = bDoc.getDocument();
  //get contents of File
    byte[] transformOutput = transformer.transform(fileContents);
    File transformedFile = bDoc.createFile();
  //write transformedOutput to transformedFile
   bDoc.setDocument(transformedFile);
         return bDoc;
    }

    public boolean reset() {
         /*
          * reset internal variables.
          */
      }
}
```

第 6 章 送信側のカスタマイズ

送信側は、WBI Connect データ・フローの最終ステージを処理します。 BPE から文書を取り出し、それらをパッケージ化し、コンソール構成ゲートウェイの情報に基づいて宛先に送信します。同期要求の場合、応答文書の処理も行います。4.2.2 リリースでは、ユーザーは、新規送信側または新規送信側ハンドラーの作成の、2 つの方法で送信側ステージの処理をカスタマイズできます。この章では、送信側カスタマイズについての両方の方法を説明します。

- 『送信側の新規作成の概要』
- 96 ページの『新規の送信側ハンドラー作成の概要』

追加セクションで、開発および配置について説明します。

- 96 ページの『開発と配置』

API リストとサンプル・コードは、次の章で説明します。

送信側の新規作成の概要

送信側は、トランスポート固有です。WBI-C には、FTP/S、/JMS、File、SMTP、および HTTP/S トランスポート用の送信側が付属しています。 WBI-C システムに新規機能を追加するために (例えば、WAP トランスポートの追加)、ユーザーは 4.2.2 リリースで提供される API を使用して、独自の送信側を作成できます。これらの新規送信側は、Community Console を使用してトランスポートとの関連付けが可能であり、また通常の方法で処理フローに統合できます。このセクションでは、新規送信側の開発プロセスを説明します。次について説明します。

- 『送信側/送信側フレームワーク・フロー』
- 96 ページの『送信側アーキテクチャー』

送信側/送信側フレームワーク・フロー

WBI-C 側の送信側の処理フローの本質はある程度は特定の状態およびトランスポートのニーズによって指示されますが、実現する必要のある基本的なタスクがあります。ここでは、これらのタスクの上位について、一般的な方法で説明します。

1. **文書の受信**
   処理されるビジネス文書がゲートウェイ入力ディレクトリーに置かれます。
2. **前処理の実行**
   送信側フレームワーク (内部コンポーネント) が起動され、その `send` メソッドが呼び出されます。フレームワークは、文書を処理できる Connect 提供かユーザー定義のハンドラーを見つけるまで、コンソールに構成されたハンドラーのリスト内を順番に移動します。文書が処理されます。
3. **送信側の初期化**
   フレームワークは送信側の `init` メソッドを呼び出します。
4. **文書の送信**
   フレームワークは送信側の `send` メソッドを呼び出します。

送信側は SenderResult オブジェクトを作成して伝送および状況情報を格納し、ゲートウェイに指定された宛先、または設定によりメッセージにその情報が含まれる場合はメッセージ自体に指定された宛先を使用して、送信します。

4A. 応答文書の格納

同期要求の場合、送信側は応答文書をファイルに書き込んで、SenderResult オブジェクトに File オブジェクトを設定します。

5. 後処理の実行

フレームワークは、文書に該当する Connect 提供かユーザー定義のハンドラーを見つけるまで、コンソールに構成されたハンドラーのリスト内を移動します。SenderResult オブジェクトが処理されます。

8. 処理の完了

要求文書がゲートウェイ・キューから除去されます。同期要求の場合、応答文書は応答処理のために取り出されるディレクトリーに入れられます。

送信側アーキテクチャー

送信側の開発は次の 2 つの主要部分をベースとします。つまり、API に `SenderInterface` インターフェースで表される送信側自体と、送信側の管理を行う Connect 提供のクラスである送信側フレームワークです。送信側はメッセージを宛先に実際に送信すると共に、SenderResult オブジェクトを作成して最初にデータを取り込みます。同期要求の場合、送信側は応答文書をファイルに書き込んで、File オブジェクトへの参照を SenderResult オブジェクトに設定します。フレームワークは、文書の前処理および後処理の実行と、送信側をインスタンス化して使用します。

新規の送信側ハンドラー作成の概要

SenderFramework は送信側処理フロー内の、前処理および後処理の 2 つのステージでハンドラーを呼び出すことができます。これらのステージは、**構成ポイント**とも呼ばれます。前処理は、要求文書を宛先に送信するために送信側に渡す前に行うことを対象とし、後処理は、要求文書が宛先に送信されて、要求の状況を文書化するために SenderResult オブジェクトが作成された後を対象とします。

WBI-C には、いくつかの事前定義ハンドラーが付属していますが、提供されたハンドラーではユーザー特有のニーズに対応できない場合は、独自のハンドラーを開発することもできます。例えば、要求文書が設定済み取引先からの場合、前処理ハンドラーはパートナーの状況を判別し、それに応じてトランスポート・ヘッダーを設定できます。ハンドラーを作成し、配置したら、ユーザーは Connect 提供のハンドラーの場合のように、コンソールを使用してこれらを構成する必要があります。このプロセスの詳細については、「*ハブ構成ガイド*」を参照してください。

開発と配置

以下のセクションでは、ユーザー作成の送信側とハンドラーの両方の開発および配置について説明します。

開発環境

送信側および送信側ハンドラーの開発 API は、次のパッケージにあるクラスとインターフェースに基づきます。

- com.ibm.bcg.bcgdk.gateway

このパッケージは、次の WBI-C インストール可能ディレクトリーに存在する bcgsdk.jar の一部です。

- <*install dir*>¥router¥was¥wbic
- <*install dir*>¥receiver¥was¥wbic
- <*install dir*>¥console¥was¥wbic

すべての配置済みインスタンスで、この .jar ファイルはモジュール・クラスパスでなく、アプリケーション・サーバー・クラスパスで使用できる必要があります。

開発のためには、bcgsdk.jar ファイルはユーザー出口クラスを含むプロジェクトのビルド・パス (つまり、クラスパス) に組み込まれている必要があります。

配置とパッケージ

カスタム送信側やカスタム・ハンドラーなどのすべてのユーザー作成コードは、次のいずれかの方法でパッケージ化し、配置する必要があります。

- ¥router¥was¥wbic¥userexits の .jar ファイルに配置
- ¥router¥was¥wbic¥userexits¥classes のクラスとして追加

さらに、すべてのハンドラー・クラスには、ハンドラーに関する重要なデータを Console GUI にインポートできる XML 記述子ファイルが付属している必要があります。以下の簡潔なアウトラインにおいて、HandlerClassName はハンドラー・クラスの名前、 Description はクラスの要旨、HandlerTypeValues はコンポーネント名 (RECEIVER - 受信側ハンドラー、GATEWAY - 送信側ハンドラー、FIXEDWORKFLOW - 固定インバウンドまたはアウトバウンド・ハンドラー、または ACTION - 可変ワークフロー・ステップ) で、ハンドラーのタイプが後に続きます (例えば PREPROCESS.<transport>、PROTOCOL.UNPACKAGING、など)。また、ComponentAttribute は、ランタイムにおけるハンドラーの振る舞いを変更する、ハンドラー・クラス内の構成可能メンバー変数です。

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:HandlerDefinition
     xmlns:tns="http://www.ibm.com/websphere/bcg/2004/v0.1/import/external"
     xmlns:tns2="http://www.ibm.com/websphere/bcg/2004/v0.1/import
                                                          /external/types"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.ibm.com/websphere/bcg/2004/v0.1
                            /import/external bcghandler.xsd
                        http://www.ibm.com/websphere/bcg/2004/v0.1
                            /import/external/types bcgimport.xsd ">
 <tns:HandlerClassName>com.ibm.SampleHandler </tns:HandlerClassName>
 <tns:Description>A Sample Handler</tns:Description>
 <tns:HandlerTypes>
     <tns:HandlerTypeValue>COMPONENT.TYPE</tns:HandlerTypeValue>
 </tns:HandlerTypes>
 <tns:HandlerAttributes>
       <tns2:ComponentAttribute>
               <tns2:AttributeName>Attribute 1</tns2:AttributeName>
       </tns2:ComponentAttribute>
```

```
        <tns2:ComponentAttribute>
            <tns2:AttributeName>Attribute 2</tns2:AttributeName>
             <tns2:AttributeDefaultValue>Attribute2DefaultValue
            </tns2:AttributeDefaultValue>
         </tns2:ComponentAttribute>
  </tns:HandlerAttributes>
</tns:HandlerDefinition>
```

第 7 章 送信側および送信側ハンドラー用の API およびコード例

この章では、カスタム送信側およびカスタム送信側ハンドラーを開発するために用意されている API の注釈付きのリストを示します。次のクラスとインターフェースを説明します。

- 100 ページの『SenderInterface』
- 102 ページの『SenderResult』
- 107 ページの『SenderPreProcessHandlerInterface』
- 109 ページの『SenderPostProcessHandlerInterface』
- 111 ページの『BCGSenderException』
- 112 ページの『イベント』
- その他のユーティリティー、セキュリティー、およびコンポーネント間の共用クラスについては、ワークフロー API の章にあるリストも参照してください。

このリストには、送信側および前処理と後処理のハンドラーの例の実装の概要を示すコードと疑似コードの簡単な例が含まれています。

SenderInterface

各送信側はこのインターフェースを実装する必要があります。これには、次のメソッドがあります。

- init
- send
- cancel

メソッド

init

メソッドの説明

ゲートウェイ構成情報を含む deliveryConfig オブジェクトの内容に基づいて、送信側を初期化します。

構文

```
public void init (Context context, Config deliveryConfig)
       throws BCGSenderException
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

deliveryConfig
コンソールで指定されたゲートウェイ構成の詳細

メソッド

send

メソッドの説明

SenderFramework によって呼び出されます。deliveryConfig オブジェクトに指定された情報を使用して宛先に文書を送信します。SenderResult オブジェクトを作成し、デリバリー状況、WBI-C ヘッダーおよびトランスポート・ヘッダー、また同期フローの場合には応答文書で更新します。デリバリーに失敗すると、送信側はトランスポートを再試行できます。

構文

```
public SenderResult send(BusinessDocumentInterface document)
```

パラメーター

document
送信されるビジネス文書

メソッド

cancel

メソッドの説明

SenderFramework によって呼び出されます。メッセージのデリバリーおよびすべてのトランスポート再試行を停止します。

構文

```
public SenderResult cancel()
```

パラメーター

なし

SenderResult

SenderResult オブジェクトは、この提供されたクラスに基づいて送信側によって作成されます。要求ビジネス文書の状況についてのメタ情報を保持し、また同期フローの場合は応答文書を含む File オブジェクトへの参照を保持します。これには、次のメソッドが含まれます。

- addEvent
- getEvents
- setSendStatus
- getSendStatus
- setResponseDocument
- getResponseDocument
- setTransportStatusCode
- getTransportStatusCode
- setTransportHeaders
- getTransportHeaders
- setAttribute
- getAttribute
- get Attributes

メソッド

addEvent

メソッドの説明

SenderResult オブジェクトにイベントを追加します。

構文

```
public void addEvent(EventInfo)
```

パラメーター

EventInfo

EventInfo は、com.ibm.bcg.bcgdk.common パッケージからの専門化されたクラスで、WBI-C システム全体のイベント情報を保持するために使用されます。これについては、ワークフロー API の章で説明します。

メソッド

getEvents

メソッドの説明

このオブジェクト内のイベント・セットを検索します。

構文

```
public EventInfo[] getEvents()
```

パラメーター

なし

メソッド

```
setSendStatus
```

メソッドの説明

デリバリー状況を設定します。伝送状況に応じて、成功または失敗となります。

構文

```
public void setSendStatus(String status)
```

パラメーター

status 該当状況

メソッド

```
getSendStatus
```

メソッドの説明

デリバリー状況を検索します。

構文

```
public String getSendStatus()
```

パラメーター

なし

メソッド

```
setResponseDocument
```

メソッドの説明

応答文書を保持するファイルを設定します。

構文

```
public void setResponseDocument(File responseFile)
```

パラメーター

responseFile
応答文書が格納される File オブジェクト

メソッド

getResponseDocument

メソッドの説明

応答文書を保持する File オブジェクトを検索します。

構文

```
public File getResponseDocument()
```

パラメーター

なし

メソッド

setTransportStatusCode

メソッドの説明

トランスポート戻り状況コード (例えば、HTTP 200 OK) を設定します。

構文

```
public void setTransportStatusCode(Object transportStatusCode)
```

パラメーター

transportStatusCode
コード

メソッド

getTransportStatusCode

メソッドの説明

トランスポート戻り状況コードを検索します。

構文

```
public Object getTransportStatusCode()
```

パラメーター

なし

メソッド

setTransportHeaders

メソッドの説明

同期応答を受信すると、これらのヘッダーを設定します。

構文

```
public void setTransportHeaders(HashMap transportHeaders)
```

パラメーター

transportHeaders
トランスポート・ヘッダーを含む HashMap

メソッド

```
getTransportHeaders
```

メソッドの説明

送信側によって設定されたトランスポート・ヘッダーを検索します。

構文

```
public HashMap getTransportHeaders()
```

パラメーター

なし

メソッド

```
setAttribute
```

メソッドの説明

WBI-C 固有の属性を設定します。これらの属性には、送信側に固有のヘッダーが含まれます。これらはフレームワークによってメタデータ・ファイルへの入力 (デリバリー期間、トランスポート状況説明など) として使用されます。

構文

```
public void setAttribute(String name, Object obj)
```

パラメーター

name 属性を格納するオブジェクトの名前

obj オブジェクト

メソッド

```
getAttribute
```

メソッドの説明

WBI-C 固有の属性を検索します。

構文

```
public Object getAttribute()
```

パラメーター

なし

メソッド

```
getAttributes
```

メソッドの説明

すべての属性セットの HashMap を検索します。

構文

```
public Map getAttributes set()
```

パラメーター

なし

SenderPreProcessHandlerInterface

このインターフェースは、すべての前処理ハンドラーが実装する必要のある 3 つのメソッドを記述します。これには次が含まれます。

- init
- applies
- process

メソッド

init

メソッドの説明

Config オブジェクトの構成プロパティーを読み取ってハンドラーを初期化します。

構文

```
public void init(Context context, Config handlerConfig)
              throws BCGSenderException
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

handlerConfig
構成情報を格納するオブジェクト

メソッド

applies

メソッドの説明

ハンドラーがビジネス文書を処理できるかどうかを判別します。

構文

```
public boolean applies(BusinessDocumentInterface doc)
              throws BCGSenderException
```

パラメーター

doc 処理中の BusinessDocument

メソッド

process

メソッドの説明

BusinessDocument を更新して前処理を実行します。

構文

```
public BusinessDocumentInterface process(BusinessDocumentInterface doc)
                throws BCGSenderException
```

パラメーター

doc　処理中の BusinessDocument

SenderPostProcessHandlerInterface

このインターフェースは、すべての前処理ハンドラーが実装する必要のある 3 つのメソッドを記述します。これには次が含まれます。

- init
- applies
- process

メソッド

init

メソッドの説明

Config オブジェクトの構成プロパティーを読み取ってハンドラーを初期化します。

構文

```
public void init(Context context, Config handlerConfig)
             throws BCGSenderException
```

パラメーター

context
一時ディレクトリー・パスなどのランタイム情報

handlerConfig
構成情報を格納するオブジェクト

メソッド

applies

メソッドの説明

ハンドラーが適用可能文書を処理できるかどうかを判別します。

構文

```
public boolean applies(BusinessDocumentInterface doc)
            throws BCGRSenderException
```

パラメーター

doc 処理中の BusinessDocument

メソッド

process

メソッドの説明

SenderResult オブジェクトを更新してデリバリー応答に後処理を実行します。

構文

```
public SenderResult process(SenderResult response,
                            BusinessDocumentInterface doc)
            throws BCGReceiverException
```

パラメーター

response
更新される Sender Result オブジェクト

doc 処理中の BusinessDocument

BCGSenderException

BCGException を拡張するユーティリティー・クラスです。BCGException は Exception を拡張します。

イベント

以下は送信側の実行フローで使用可能なイベントのリストです。

通知イベント

BCG_240616 - 送信側入り口

イベント・テキスト: 送信側 ({0}) の入り口

引き数の予想値:

{0} - 送信側のクラス名

BCG_240617 - 送信側出口

イベント・テキスト: 送信側 ({0}) の出口

引き数の予想値:

{0} - 送信側のクラス名

BCG_250007 - 文書がデリバリーされました

イベント・テキスト: 文書が正常にデリバリーされました。応答: {0}

引き数の予想値:

{0} - <<ターゲット応答状況>>

警告イベント

BCG_240618 - 送信側の警告

イベント・テキスト: 送信側の警告 -{0}

引き数の予想値:

{0} - 送信側の警告情報

エラー・イベント

BCG_250008 - 文書のデリバリーに失敗しました

イベント・テキスト: 参加者ゲートウェイへの文書デリバリーに失敗しました: {0}

引き数の予想値:

{0} - <<応答状況およびエラー・メッセージ>>

送信側および送信側ハンドラーの実装例の概要

次のコードおよび疑似コードは、送信側および送信側ハンドラーの実装例の概要を示します。

送信側の例

送信側ハンドラーの実装の概要です。開発者によってプロトコル固有のコードを追加する必要があります。

```
public class SampleSenderUserExit implements SenderInterface
{
        SenderResult result = new SenderResult ();
        Connection connection = null;
        public SampleSenderUserExit()
        {
             ...
    }
        public void init(Context context, Config deliveryConfig)
                 throws BCGSenderException
        {
                //initialization code
           ...
    }
        public SenderResult send(BusinessDocumentInterface document)
                  throws BCGSenderException
        {
               //destination url set in the BusinessDocumentInterface
               //is preferred over url set in the Gateway
               URL url = (String)document.getAttribute(DocumentConstant.URI);
                if (url == null)
                  url = (String)config.getConfig(DocumentConstant.URI);
                //obtain other configuration information from the
                //DeliveryConfig which holds the configuration information
                //set in the Gateway
                File documentFile = document.getDocument();
                try
                {
                        //read contents from File
                    ...
            }
                catch(FileIOException exception)
                {
                        BCGSenderException e = new BCGSenderException(...);
                        throw e;
            }
                try
                {
                        //establish connection with the destination
                    ...
                        //send contents to destination
                    ...
                        //if DocumentConst.GET_SYNC_RESPONSE is true,
                        //read the response
                        if (String.valueOf(document.getAttribute(
                         DocumentConst.GET_SYNC_RESPONSE).
                                 equalsIgnoreCase(DocumentConstants.TRUE))
                        {
                           //read the response
                        ...
                String uuid = BCGUtil.getDocumentUUID();
                String tmpDir = BCGUtil.TMP_DIR_PATH;
                File responseFile = document.createFile(uuid+".resp");
                //write response to file
                         ...
```

```
                result.setResponse(responseFile);
                        ...
                    }
                      else
                      {
                            //read the transport response and
                            //set in the SenderResult object.
                            ...
                      }
                }
                  catch(Exception exception)
                  {
                          //create an event and add to the SenderResult
                          Object[] params = {exception};
                          EventInfo eventInfo = EventInfo(eventCode,
                                                          document, params);
                          result.addEvent(eventInfo);
                  result.setStatus(DocumentConstant.BCG_DOC_FAILED);
                          return result;
                }
                  //close the connection
                ...
             result.setStatus(DocumentConstant.BCG_DOC_SENT);
                  return result;
      }

        public SenderResult cancel() throws BCGSenderException
        {
                    connection.close();
                    EventInfo eventInfo = EventInfo(eventCode,
                                                    document, params);
                    result.setEvent(eventInfo);
                    return result;
      }
}
```

前処理ハンドラーの例

次のコードおよび疑似コードは、送信側の前処理ハンドラーの実装例の概要を示します。設定済みカスタマーが toPartner の場合、それに応じて BusinessDocument に一部のトランスポート属性を設定します。

```
public class SampleSenderPreProcessUserExit implements
SenderPreProcessHandlerInterface
{
    public SampleSenderPreProcessUserExit ()
    {
        ...
    }
    public void init(Config handlerConfig) throws BCGSenderException
    {
        //initialization code
        ...
    }
    public boolean applies(Config deliveryConfig,
                           BusinessDocumentInterface doc)
                    throws BCGSenderException
    {
        String toProtocol =
              document.getAttribute(DocumentConstants.TOPROTOCOLNAME);
        boolean isAS2 = (toProtocol!= null &&
                         toProtocol.equalsIgnoreCase(DocumentConstants.AS2));
        return isAS2;
    }
    public BusinessDocumentInterface process (Config deliveryConfig,
```

```
                                    BusinessDocumentInterface doc)
                        throws BCGSenderException
    {
        String toPartnerId = (String)doc.getAttribute
                                    (DocumentConstants.TOBUSINESSID);
        //check if partner is a PreferredCustomer
        ...
        if (preferredCustomer)
        {
            //set transport attributes for PreferredCustomer
            doc.setAttribute(PRIORITY, "HIGH");
doc.setAttribute(ACKNOWLEDGEMENT_REQUIRED, "true");
            ...
        }
    }
}
```

後処理ハンドラーの例

次のコードおよび疑似コードは、送信側の後処理ハンドラーの実装例の概要を示します。このサンプルでは、SOAP メッセージを想定しています。応答が例外または他のタイプのエラーであるが SOAP 障害ではない場合、ハンドラーは SOAP 障害を作成して、BusinessDocument に応答を設定します。

```
public class SampleSOAPPostProcessUserExit implements
                                      SenderPostProcessHandlerInterface
{
    public void init(Config handlerConfig)
              throws BCGSenderException
    {
        //initialization code
        ...
    }
    public SampleSOAPPostProcessUserExit ()
    {
        ...
    }
    public boolean applies(Config deliveryConfig,
                                    BusinessDocumentInterface doc)
                throws BCGSenderException
    {
        String fromProtocol =
                 (String)document.getAttribute(
                                      DocumentConst.FRPROTOCOLNAME);
        boolean isSoap = (fromProtocol != null &&
        fromProtocol.equalsIgnoreCase(
                                      DocumentConst.WEB_SERVICE_PROTOCOL));
        return isSoap;
    }

    public SenderResult process (SenderResult response,
                                  BusinessDocumentInterface doc)
                    throws BCGSenderException
    {
        SoapFault sf = null;
        String deliveryStatus= response. getDeliveryStatus();
        If (deliveryStatus.equalsIgnoreCase("FAILED")){
            //Get transport header, read the http status code,
//    and create SOAP fault message
            String faultMessage="Server error";
            sf = new SoapFault(faultString);
            response.setResponse(sf.getBytes());
        }
        return response;
    }
}
```

第 8 章 エンドツーエンド同期フロー: ユーザー出口使用の概要

この章では、ユーザー提供のコードが機能する場所に重点を置き、同期要求が WBI-C システムを介して移動することに関わるステップの概要を説明します。

文書の受信

WBI-C 提供またはユーザー提供の受信側は、開始パートナーから要求を受信します。以下を行います。

- a) メッセージを一時ロケーションに書き込みます。このロケーションは、受信側が BCGReceiverUtil クラスのメソッド getTempDir を使用して取得します。
- b) (ReceiverDocumentInterface を実装するクラスに基づいて) 要求文書オブジェクトを作成し、受信側にある受信側フレームワークおよび一時ロケーションから受信した UUID を設定します。
- c) 受信したトランスポート・ヘッダーを HashMap オブジェクトに書き込み、これを要求文書で設定します。

次に、WBI-C 固有のヘッダー (BCGHeaders) を作成し、ヘッダーごとに setAttribute メソッドを呼び出すことにより、要求文書でこれらのヘッダーを設定します。

受信側は受信側フレームワークで preProcess を呼び出します。フレームワークは、コンソールを介してターゲットに指定されている Connect 提供またはユーザー定義のハンドラーを、コンソール構成画面に表示されている順に実行します。要求文書が入力として最初のハンドラーに渡され、戻された処理済みの文書が入力として次のハンドラーに渡される、という流れです。文書が処理されます。

受信側は受信側フレームワークで syncCheck を呼び出します。フレームワークは、このトランスポートおよびターゲットで定義された (WBI-C 提供またはユーザー提供の) ハンドラーのリスト内を繰り返します。文書を管理できるハンドラーが見つかると、同期検査を実行します。ブロッキング同期要求の場合、'true' が戻されます。

受信側はフレームワークで、ターゲット・タイプ、要求文書、および作成した空の応答文書を渡すことにより、process を呼び出します。

フレームワークは 2 つの属性を、要求文書 (RESPONSE_URL および REPLY_TO_MSG_ID) の BCG ヘッダーに追加します。RESPONSE_URL には WBI-C 内部同期応答サーブレット URL が含まれ、REPLY_TO_MSG_ID には要求文書の UUID が含まれます。

フレームワークは sync_in ディレクトリーに、必要な BPE 入力ファイルを作成します。次に、SyncEngine の waitForSyncResponse を呼び出すことにより、同期サーブレットが DeliveryManger (送信側を含むコンポーネント) から応答を受信すると、通知を受信するようになります。

WBI-C 内部処理では、sync_in ディレクトリーからファイルを読み取り、ビジネス文書オブジェクトを作成し、データを受け入れます。文書および関連情報をアクティビティー・テーブルに記録し、ビジネス文書を BPE 同期インバウンド・キューに書き込みます。

文書処理

BPE は文書を同期インバウンド・キューから取得します。

固定インバウンド

BPE は固定インバウンド・フローを実行します。固定インバウンド・フローの最初のステップはプロトコルのアンパックです。 BPE は、文書を管理できるハンドラーを見つけるまで、ユーザー提供のハンドラーを含む構成済みハンドラー内を繰り返します。ここではアンパックを実行します。

固定インバウンド・フローの 2 番目のステップはプロトコル処理です。 BPE は、文書を管理できるハンドラーを見つけるまで、ユーザー提供のハンドラーを含む構成済みハンドラー内を繰り返します。ここではプロトコル処理を実行します。これは同期要求であるため、ハンドラーはビジネス文書の GET_SYNC_RESPONSE 値を 'true' に設定します。固定インバウンド・ステップの残りが処理されます。

可変

BPE は構成済みアクションを実行します。これらは、WBI-C 提供であるか、WBI-C 提供のアクションをユーザーが変更したバージョンであるか、あるいはユーザーが定義した完全に新しいアクションである可能性があります。ユーザーが変更したステップ、あるいは新しいアクションのステップは、ユーザー定義の可能性があります。可変ワークフロー処理中に、同期状況に関係する定数を更新する必要はありません。

固定アウトバウンド

BPE は固定アウトバウンド・フローを実行します。固定アウトバウンド・フローの最初のステップはプロトコルのパッケージ化です。 BPE は、文書を管理できるハンドラーを見つけるまで、ユーザー提供のハンドラーを含む構成済みハンドラー内を繰り返します。ここではプロトコルのパッケージ化を実行します。固定アウトバウンド・ワークフロー処理中に、同期状況に関係する定数を更新する必要はありません。

BPE はビジネス文書を Delivery Manager のインバウンド・キューに書き込みます。

注: 同期要求と非同期要求の両方が、同じインバウンド・キューに書き込まれます。

文書の伝送

Delivery Manager は、ビジネス文書をインバウンド・キューから該当するゲートウェイ・フォルダーに転送します。 DeliveryWorker はこのフォルダーをポーリングし、ビジネス文書を検索、取得し、送信側フレームワークの processSend メソッドを呼び出します。

フレームワークは、ゲートウェイで構成された前処理ハンドラーを実行します。

フレームワークは、SenderInterface を実装するクラスで送信側オブジェクトを作成します。このクラスはユーザー提供である可能性があります。フレームワークは送信側オブジェクトの send メソッドを呼び出します。文書が送信されます。send メソッドは、ビジネス文書の GET_SYNC_RESPONSE 定数が 'true' に設定されているかどうかも確認します。true に設定されている場合は、送信側は同期応答を接続から読み取り、そのデータを使用してメソッドが戻した SenderResult オブジェクトを取り込みます。フレームワークは、ゲートウェイで構成された後処理ハンドラーを実行します。

フレームワークは SenderResult オブジェクトを読み取り、応答の UUID を作成し、BPE の入力ファイル・セットを作成します。メタデータ・ファイルには 2 つの定数が含まれます。応答トランスポート状況コードを保持する BCG_RESPONSE_STATUS と、'true' に設定される BCG_SYNC_RESP です。ファイルは BPE の sync_in ディレクトリーに書き込まれます。

応答処理

BPE は応答ビジネス文書を同期インバウンド・キューから取得します。

固定インバウンド

BPE は固定インバウンド・フローを実行します。固定インバウンド・フローの最初のステップはプロトコルのアンパックです。 BPE は、文書を管理できるハンドラーを見つけるまで、ユーザー提供のハンドラーを含む構成済みハンドラー内を繰り返します。ここでは必要なアンパックを実行します。

固定インバウンド・フローの 2 番目のステップはプロトコル処理です。 BPE は、文書を管理できるハンドラーを見つけるまで、ユーザー提供のハンドラーを含む構成済みハンドラー内を繰り返します。ここではプロトコル処理を実行します。ハンドラーは BCG_RESPONSE_URL が null かどうかを確認します。値が null であっても、ハンドラーが応答文書を受信した場合は、ハンドラーは GET_SYNC_RESPONSE を 'false' に、BCG_SYNC_RESP を null に設定します。そうでない場合は、残りの固定インバウンド処理ステップを続行します。

可変

BPE は構成済みアクションを実行します。これらは、WBI-C 提供であるか、WBI-C 提供のアクションをユーザーが変更したバージョンであるか、あるいはユーザーが定義した完全に新しいアクションである可能性があります。ユーザーが変更したステップ、あるいは新しいアクションのステップは、ユーザー定義の可能性があります。可変ワークフロー処理中に、同期状況に関係する定数を更新する必要はありません。

固定アウトバウンド

BPE は固定アウトバウンド・フローを実行します。固定アウトバウンド・フローの最初のステップはプロトコルのパッケージ化です。 BPE は、文書を管理できるハンドラーを見つけるまで、ユーザー提供のハンドラーを含む構成済みハンドラー内を繰り返します。ここではプロトコルのパッケージ化を実行します。固定アウトバウンド・ワークフロー処理中に、同期状況に関係する定数を更新する必要はありません。

すべてのワークフロー・ステップを正常に実行したら、BPE は応答文書の SYNC_RESP 属性を確認します。応答ビジネス文書は Delivery Manager の同期インバウンド・キューに書き込まれます。

応答伝送

同期 Delivery Manager は、応答を応答ビジネス文書の RESPONSE_URL セットに送信します。同期エンジンは永続的なオリジナル接続を取得します。受信側フレームワークは、(ReceiverDocumentInterface を実装するクラスから) 同期応答情報と一緒に応答オブジェクトを取り込みます。

受信側は RESPONSE_STATUS 属性を読み取り、状況コードと同期応答をオリジナル接続の開始パートナーに戻します。

第 9 章 ユーザー出口のトラブルシューティング

この章では、ユーザー出口の設定と使用における一般的なトラブルシューティング状態をいくつか取り扱います。

ロギングの設定

文書フロー全体の内部アクティビティーのロギングのセットアップに、com.ibm.bcg.bcgdk.common パッケージ内の BCGUtil クラスのトレース・メソッドが使用されます。メソッドの全文書が 80 ページの『BCGUtil』 にあります。以下は、可変ワークフローにおける XML 変換ステップでのロギングのセットアップを行うコード例の一部です。

```
BCGUtil bcgUtil = new BCGUtil ();
:
:
:
    bcgUtil.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
                          "CustomXMLTranslation",
                           "The Schema is present",
                           null)
```

受信側のログは以下の場所にあります。

<Receiver_installed_dir>/was/logs/server1/wbic_receiver.log

固定ワークフロー、可変ワークフロー、および送信側のログ (ルーター・コンポーネントの一部としてグループ化される) は以下の場所にあります。

<Router_installed_Dir>/was/logs/server1/wbic_router.log

デフォルトでは、デバッグ・ロギングは使用できません。これをオンにするには、log4j プロパティー・ファイルを設定する必要があります。受信側のプロパティー・ファイルは以下の場所にあります。

<Receiver_installed_dir>/was/wbic/Config/receiver-was.logging.properties

ルーター・コンポーネントのプロパティー・ファイルは以下の場所にあります。

<Router_installed_dir>/was/wbic/Config/ router-was.logging.properties.

どちらの場合も、設定するプロパティーは log4j.rootCategory プロパティーです。デフォルトでは、これは error, RollingFile と設定されます。この値を debug, RollingFile に変更する必要があります。変更を有効にするために、サーバーを再始動する必要があります。

一般的なエラー・ソース

以下は、ユーザー出口の設定で頻繁に発生する 5 つの一般的なエラー・タイプと、これらのエラーを訂正するためのステップです。

ファイル・ロケーション・エラー

WBI-C システムがユーザー出口クラスを検索できることは重要です。以下の場合に、いずれかのログに Class Not Found (クラスが見つかりません) 例外が発生する可能性があります。

- ユーザー出口クラスのファイルがクラスパスにロードされていない。

または

- ユーザー出口クラスのファイルが、ユーザーが Community Console を使用してアップロードした XML ファイル内のパッケージ階層の指定された場所にない。

マルチボックスの分割トポロジー・セットアップで、適切なユーザー出口が受信側またはルーターの必要なすべてのインスタンスと一緒に配置されない場合は、追加のファイル・ロケーション問題が発生する可能性があります。

解決策: クラス・ファイルがクラスパスに正しくロードされ、ユーザー出口のクラス・ファイルの名前とロケーションが、XML 記述子ファイルのコンソールへのアップロードにおいて指定された詳細と完全に一致することを確認してください。該当するファイルがすべて適切な場所に存在することを確認してください。

ハンドラーの失敗エラー

受信側コンポーネントの前処理ハンドラーの失敗、送信側コンポーネントのいずれかのタイプの失敗、またはルーター・コンポーネントのハンドラーのアンパック、プロトコル処理、パッケージ化の失敗により、該当するログおよびコンソールにエラーが生成されます。デバッグ・モードをオンにすると、エラー・レポートの詳細が生成されます。エラーが発生すると、メッセージまたはビジネス文書はそれ以上処理されず、HTTP 受信側の前処理の失敗の場合は、500 応答コードが開始ホストに戻されます。

解決策: ユーザー出口コード内の問題を訂正し、クラス・ファイルを再ロードし、コンポーネントを再始動してください。

処理モード・エラー

文書プロトコルが同期処理をサポートする場合、定義されたターゲットは指定された同期検査ハンドラーを *持つ* 必要があります。プロトコルが同期処理をサポートしない場合は、後処理ハンドラーを指定してはなりません。

解決策: 指定したユーザー出口が、定義されている処理モードに適していることを確認してください。

ファイル更新エラー

システムでユーザー出口情報を更新する 2 つの方法があります。

- クラス・ファイル (または .jar) 自身を更新する
- XML 記述子ファイルを更新する

クラス・ファイルを更新する場合は、変更を確実に有効にするために、該当するコンポーネントを再始動する必要があります。既存のユーザー出口の新規 XML 記述子ファイルを更新すると (ファイルに同じ名前があり、同じクラスを指定すると仮

定した場合)、設定された属性および属性値は即座に変更されます。この場合は、新規記述子ファイルがアップロードされた後に処理された文書は、これらの新規ファイルの指定どおりに処理されます。

解決策: クラス・ファイルの更新を有効にするためにはコンポーネントの再始動を行ってください。XML 記述子ファイルの更新は即座に有効になります。

ファイル削除エラー

ユーザー定義の受信側 (トランスポート) をターゲット定義で使用した後は、たとえそのターゲットが削除されても、この受信側を削除することはできません。ユーザー定義の送信側 (トランスポート) を使用するゲートウェイが定義された後は、たとえそのゲートウェイが**オフライン**または**使用不可**状態に設定されていても、この送信側を削除することはできません。一方、固定ワークフロー・ハンドラーは、これが特定のワークフローの一部であっても、削除できます。ハンドラーが削除されると、**構成済みリスト**からも即時に除去されます。このハンドラーに従属している文書フローは、ハンドラーの削除後に処理されると失敗します。

解決策: ターゲットまたはゲートウェイの定義に使用されたトランスポートを削除しないようにしてください。固定ワークフローのハンドラーがどこからも使用されていないことを確信している場合を除いては、これらのハンドラーを削除しないようにしてください。

第 2 部 Business Integration Connect のカスタマイズ: 管理 API および外部イベント・デリバリー

これまでは、さまざまなタスクを日々のコミュニティー・ハブ管理と関連付ける処理を行うのに、Community Console を使用する必要がありました。WebSphere Business Integration Connect 4.2.2 がリリースされて、ハブ管理者は、新たに開発された API を使用することにより、簡単な XML ベースの HTTP POST メカニズムを使用し、特定の共通管理用タスクをプログラマチックに確立できるようになりました。さらに、WBI-C が変更されて、文書に関連したイベントと汎用システム・ベースのイベントの両方を、内部 WBI-C イベント・ストアに送信するのと同様に外部 JMS キューにデリバリーできるようになりました。

この章で、これらの WBI-C の新機能について説明します。

第 10 章 管理 API の使用

この章では、管理 API について説明します。管理 API によって、特定のハブ管理機能をプログラマチックに実行することができます。この章は 2 つのセクションに分かれています。

- 『管理 API について理解する』
- 128 ページの『管理 API』

管理 API について理解する

WBI-C 管理 API によって、Community Console GUI を使用することなく、特定の共通管理機能を実行できます。

注: API 呼び出しを処理するには Console それ自身が稼動していなければならず、呼び出しを行う前に GUI で API 機能をオンにしておく必要があります。 GUI を使用して API をオンにする際の詳細については、「*WBI-C ハブ構成ガイド*」を参照してください。

メソッドは、本文として該当する XML 文書を持つ HTTP POST 要求を送信することによって呼び出されます。この要求は、Console インスタンス (相対 URL `/console/bcgpublicapi`) で実行されているサーブレットに送信されます。

一般に、XML 要求文書には以下のデータが入っています。

- ユーザー情報 (これはコンソールを介してログインした際に使用される情報と同じで、セッション管理の観念がないため、すべての要求に指定しなければなりません)。
  - ユーザー名
  - パスワード
  - パートナー・ログイン名
- API 情報
  - メソッド名 (通常、名詞と動詞の連結によって与えられます。例えば ParticipantCreate)
  - パラメーター (通常、項目です。例えば Partner)

現在、以下の 11 のメソッドがサポートされています。

- 130 ページの『ParticipantCreate』
- 132 ページの『ParticipantUpdate』
- 134 ページの『ParticipantSearchByName』
- 135 ページの『ParticipantAddBusinessId』
- 136 ページの『ParticipantRemoveBusinessId』
- 137 ページの『ContactCreate』
- 145 ページの『ListTargets』
- 141 ページの『ListParticipantCapabilities』

- 143 ページの『ListParticipantConnections』
- 145 ページの『ListTargets』
- 147 ページの『ListEventDefinitions』

システムは要求を処理し、応答 (または例外) XML を同時に (つまり、同じ HTTP 接続で) 戻します。各メソッドには、対応する応答があります。 API を使用すると、コンソールを使用して作成するものと同じ内部プロセスが作成されます。コンソールを介して実行される特定の操作がイベントを生成する場合、API を介して実行される同じ操作も同じイベントを生成します。

『管理 API』に、これらの API について詳しく記述されています。提供されている以下の 2 つのスキーマにある `$(WBICINSTALLROOT)/publicapi` ディレクトリーを調べることによって、さらに詳しい情報を収集できます。

- `bcgpublicapi_v0.1.xsd`: API シグニチャー
- `bcgpublicapi_vocabulary_v0.1.xsd`: スキーマが構成されるときのボキャブラリー

実際の応答に加え、サーブレットそれ自身も 表 1に示されている標準 HTTP 状況コードを提供します。

表 1. サーブレット状況コード

HTTP 状況コード	このコードが戻される状態
500	• 要求 XML を構文解析できない • 要求の処理にエラーがある • 内部エラーがある
405	• POST 以外の HTTP 要求が受信された (サーブレットは POST メソッドしかサポートしていない)
200	• API が正常に実行された
501	• 実装されていない要求が受信された • 管理 API がオンになっていない

セキュリティーは SSL の使用によって、また場合によってはクライアント許可によって提供されています。文字エンコードが、予期されている標準エンコードの UTF-8 に設定されているかぎり、ユーザーのロケールに基づいてデータ (API それ自身のエレメントではない) をローカライズすることができます。

管理 API

以下のセクションには、11 の XML メソッド呼び出しの構造と応答、またエラーを報告するために使用される例外 XML についての概説があります。 XML の一般的な構造は以下のとおりです。

- ルート・エレメントは常に `BCGPublicAPI` エレメントです。
- 要求文書内のルートの最初の子は `<MethodName>` エレメントです。

- <MethodName> エレメントの最初の子は UserInfo エレメントです。このエレメントには、要求を行うユーザーのための Console ログイン情報が入っています。そのユーザーには、タスクを試みるために十分な許可がなければなりません。
- <MethodName> エレメントの 2 番目の子は、入力パラメーターを表します。

- 応答文書内のルートの最初の子は <MethodName>Response エレメントです。このエレメントは、要求 API の実行結果を表します。
- 例外文書内のルートの最初の子は BCGPublicAPIException エレメントです。

ParticipantCreate

参加者をハブ・コミュニティーに追加します。参加者とは、ハブ・コミュニティーを介して Community Manager とビジネスを行う会社のことです。接続されると、参加者は電子ビジネス文書を Community Manager と交換することができます。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantCreate

ParticipantCreate の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ParticipantCreate の 2 番目の子

ParticipantCreateInfo エレメント。 7 つのエレメントが含まれます。

- ParticipantLogin: 参加者のログイン名。
- ParticipantName: 参加者がハブ・コミュニティーに表示する名前。
- ParticipantType: コミュニティー内での参加者の機能を定義します。使用可能な値は Community Operator、Community Manager、または Community Participant です。
- ParticipantStatus: Enabled または Disabled。 Disabled (使用不可) にすると、参加者は検索条件およびドロップダウン・リストに表示されなくなります。デフォルト値は Enabled です。
- CompanyURL: 参加者の Web サイトの URL。これはオプションのエレメントです。
- ClassificationId: 参加者の役割を識別します。使用可能な値は Supplier、Contract Manufacturer、Distributor、Logistic Provider、または Other です。これはオプションのエレメントです。
- Password: その参加者がシステムにアクセスするときに使用するパスワード。

ParticipantCreateResponse

ParticipantCreate メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantCreateResponse

ParticipantCreateResponse の最初の子

ParticipantCreateResponseInfo エレメント。 7 つのエレメントが含まれます。

- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- ParticipantLogin: 参加者のログイン名。
- ParticipantName: 参加者がハブ・コミュニティーに表示する名前。
- ParticipantType: コミュニティー内での参加者の機能を定義します。使用可能な値は Community Operator、Community Manager、または Community Participant です。
- ParticipantStatus: Enabled または Disabled。 Disabled (使用不可) にすると、参加者は検索条件およびドロップダウン・リストに表示されなくなります。
- CompanyURL: 参加者の Web サイトの URL。これはオプションのエレメントです。
- ClassificationId: 参加者の役割を識別します。使用可能な値は Supplier、Contract Manufacturer、Distributor、Logistic Provider、または Other です。これはオプションのエレメントです。

ParticipantUpdate

システム内の参加者のプロファイルを更新します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantUpdate

ParticipantUpdate の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ParticipantUpdate の 2 番目の子

ParticipantUpdateInfo エレメント。 6 つのエレメントが含まれます。

- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- ParticipantName: 参加者がハブ・コミュニティーに表示する名前。
- ParticipantType: コミュニティー内での参加者の機能を定義します。使用可能な値は Community Operator、Community Manager、または Community Participant です。
- ParticipantStatus: Enabled または Disabled。 Disabled (使用不可) にすると、参加者は検索条件およびドロップダウン・リストに表示されなくなります。
- CompanyURL: 参加者の Web サイトの URL。これはオプションのエレメントです。
- ClassificationId: 参加者の役割を識別します。使用可能な値は Supplier、Contract Manufacturer、Distributor、Logistic Provider、または Other です。これはオプションのエレメントです。

ParticipantUpdateResponse

ParticipantUpdate メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantUpdateResponse

ParticipantUpdateResponse の最初の子

ParticipantUpdateResponseInfo エレメント。 7 つのエレメントが含まれます。

- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- ParticipantLogin: 参加者のログイン名。
- ParticipantName: 参加者がハブ・コミュニティーに表示する名前。

- `ParticipantType`: コミュニティー内での参加者の機能を定義します。使用可能な値は Community Operator、Community Manager、または Community Participant です。
- `ParticipantStatus`: Enabled または Disabled。 Disabled (使用不可) にすると、参加者は検索条件およびドロップダウン・リストに表示されなくなります。
- `CompanyURL`: 参加者の Web サイトの URL。これはオプションのエレメントです。
- `ClassificationId`: 参加者の役割を識別します。使用可能な値は Supplier、Contract Manufacturer、Distributor、Logistic Provider、または Other です。これはオプションのエレメントです。

ParticipantSearchByName

表示名で参加者のプロファイルを検索します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantSearchByName

ParticipantSearchByName の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ParticipantSearchByName の 2 番目の子

ParticipantName エレメント: 参加者がハブ・コミュニティーに表示する名前。

ParticipantSearchByNameResponse

ParticipantSearchByName メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantSearchByNameResponse

ParticipantSearchByNameResponse の最初の子エレメント

Participants エレメント

ゼロ以上の Participants の子

ParticipantInfo。 5 つのエレメントが含まれます。

- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- ParticipantLogin: 参加者のログイン名。
- ParticipantName: 参加者がハブ・コミュニティーに表示する名前。
- ParticipantType: コミュニティー内での参加者の機能を定義します。使用可能な値は Community Operator、Community Manager、または Community Participant です。
- ParticipantStatus: Enabled または Disabled。 Disabled (使用不可) にすると、参加者は検索条件およびドロップダウン・リストに表示されなくなります。

ParticipantAddBusinessId

ビジネス ID を参加者のプロファイルに追加します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantAddBusinessId

ParticipantAddBusinessId の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ParticipantAddBusinessId の 2 番目の子

ParticipantAddBusinessIdInfo エレメント。 3 つのエレメントが含まれます。

- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- BusinessId: システムがルーティングに使用する DUNS、DUNS+4、または Freeform 番号。 DUNS 番号は 9 桁で、DUNS+4 は 13 桁でなければなりません。 Freeform ID 番号では、60 文字までの英字、数字、および特殊文字が受け入れられます。
- BusinessIdType: 使用される ID のタイプ。使用可能な値は DUNS、DUNS+4、または Freeform です。

ParticipantAddBusinessIdResponse

ParticipantAddBusinessId メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantAddBuinessIdResponse

ParticipantAddBuinessIdResponse の最初の子エレメント

ParticipantAddBuinessIdResponseInfo エレメント。4 つのエレメントが含まれます。

- BusinessIdentifierId: BusinessId をシステムに対して識別する内部数値 ID。
- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- BusinessId: システムがルーティングに使用する DUNS、DUNS+4、または Freeform 番号。 DUNS 番号は 9 桁で、DUNS+4 は 13 桁でなければなりません。 Freeform ID 番号では、60 文字までの英字、数字、および特殊文字が受け入れられます。
- BusinessIdType: 使用される ID のタイプ。使用可能な値は DUNS、DUNS+4、または Freeform です。

ParticipantRemoveBusinessId

ビジネス ID を参加者のプロファイルから除去します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantRemoveBusinessId

ParticipantRemoveBusinessId の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ParticipantRemoveBusinessId の 2 番目の子

BusinessIdentifierId エレメント: BusinessId をシステムに対して識別する内部数値 ID。

ParticipantRemoveBusinessIdResponse

ParticipantRemoveBusinessId メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ParticipantRemoveBuinessIdResponse

ContactCreate

連絡先を作成します。連絡先とは、システムで指定されたイベントの結果として、そのシステムがアラートを生成したときに通知を受けなければならない重要な担当者です。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ContactCreate

ContactCreate の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ContactCreate の 2 番目の子

ContactCreateInfo エレメント。 13 のエレメントが含まれます。

- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- GivenName: 連絡先の名。
- FamilyName: 連絡先の姓。
- Address: 連絡先の住所。これはオプションのエレメントです。
- ContactType: 連絡先の役割。これはオプションのエレメントです。使用可能な値:
  - Project Manager
  - Business Lead
  - Technical Lead
  - B2B Lead
  - Data Content Lead
  - Backend Application Lead
  - Network Firewall Lead
- Email: 連絡先の E メール・アドレス。これはオプションのエレメントです。
- Telephone: 連絡先の電話番号。これはオプションのエレメントです。
- FaxNumber: 連絡先の FAX 番号。これはオプションのエレメントです。
- LanguageLocale: 連絡先の言語ロケール。これはオプションのエレメントです。
- FormatLocale: 連絡先の追加ロケール情報。これはオプションのエレメントです。
- TimeZone: 連絡先の時間帯。これはオプションのエレメントです。
- AlertStatus: 連絡先がアラートを受け取るべきかどうかを指示します。使用可能な値は Enabled または Disabled です。デフォルトは Disabled です。

- Visibility: 可視性を示します。使用可能な値は、Local (組織に制限) および Global (組織および Community Manager) です。デフォルトは Local です。

ContactCreateResponse

ContactCreate メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ContactCreateResponse

ContactCreateResponse の最初の子

ContactCreateResponseInfo エレメント。 14 のエレメントが含まれます。

- ContactId: 連絡先をシステムに対して識別する内部数値 ID。
- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- GivenName: 連絡先の名。
- FamilyName: 連絡先の姓。
- Address: 連絡先の住所。これはオプションのエレメントです。
- ContactType: 連絡先の役割。これはオプションのエレメントです。使用可能な値:
  - Project Manager
  - Business Lead
  - Technical Lead
  - B2B Lead
  - Data Content Lead
  - Backend Application Lead
  - Network Firewall Lead
- Email: 連絡先の E メール・アドレス。これはオプションのエレメントです。
- Telephone: 連絡先の電話番号。これはオプションのエレメントです。
- FaxNumber: 連絡先の FAX 番号。これはオプションのエレメントです。
- LanguageLocale: 連絡先の言語ロケール。これはオプションのエレメントです。
- FormatLocale: 連絡先の追加ロケール情報。これはオプションのエレメントです。
- TimeZone: 連絡先の時間帯。これはオプションのエレメントです。
- AlertStatus: 連絡先がアラートを受け取るべきかどうかを指示します。使用可能な値は Enabled または Disabled です。デフォルトは Disabled です。
- Visibility: 可視性を示します。使用可能な値は、Local (組織に制限) および Global (組織および Community Manager) です。デフォルトは Local です。

ContactUpdate

連絡先情報を更新します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ContactUpdate

ContactUpdate の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ContactUpdate の 2 番目の子

ContactUpdateInfo エレメント。 13 のエレメントが含まれます。

- ContactId: 連絡先をシステムに対して識別する内部数値 ID。
- GivenName: 連絡先の名。
- FamilyName: 連絡先の姓。
- Address: 連絡先の住所。これはオプションのエレメントです。
- ContactType: 連絡先の役割。これはオプションのエレメントです。使用可能な値:
  - Project Manager
  - Business Lead
  - Technical Lead
  - B2B Lead
  - Data Content Lead
  - Backend Application Lead
  - Network Firewall Lead
- Email: 連絡先の E メール・アドレス。これはオプションのエレメントです。
- Telephone: 連絡先の電話番号。これはオプションのエレメントです。
- FaxNumber: 連絡先の FAX 番号。これはオプションのエレメントです。
- LanguageLocale: 連絡先の言語ロケール。これはオプションのエレメントです。
- FormatLocale: 連絡先の追加ロケール情報。これはオプションのエレメントです。
- TimeZone: 連絡先の時間帯。これはオプションのエレメントです。
- AlertStatus: 連絡先がアラートを受け取るべきかどうかを指示します。使用可能な値は Enabled または Disabled です。デフォルトは Disabled です。

- Visibility: 可視性を示します。使用可能な値は、Local (組織に制限) および Global (組織および Community Manager) です。デフォルトは Local です。

ContactUpdateResponse

ContactUpdate メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ContactUpdateResponse

ContactUpdateResponse の最初の子

ContactUpdateResponseInfo エレメント。 14 のエレメントが含まれます。

- ContactId: 連絡先をシステムに対して識別する内部数値 ID。
- ParticipantId: 参加者をシステムに対して識別する内部数値 ID。
- GivenName: 連絡先の名。
- FamilyName: 連絡先の姓。
- Address: 連絡先の住所。これはオプションのエレメントです。
- ContactType: 連絡先の役割。これはオプションのエレメントです。使用可能な値:
  - Project Manager
  - Business Lead
  - Technical Lead
  - B2B Lead
  - Data Content Lead
  - Backend Application Lead
  - Network Firewall Lead
- Email: 連絡先の E メール・アドレス。これはオプションのエレメントです。
- Telephone: 連絡先の電話番号。これはオプションのエレメントです。
- FaxNumber: 連絡先の FAX 番号。これはオプションのエレメントです。
- LanguageLocale: 連絡先の言語ロケール。これはオプションのエレメントです。
- FormatLocale: 連絡先の追加ロケール情報。これはオプションのエレメントです。
- TimeZone: 連絡先の時間帯。これはオプションのエレメントです。
- AlertStatus: 連絡先がアラートを受け取るべきかどうかを指示します。使用可能な値は Enabled または Disabled です。デフォルトは Disabled です。
- Visibility: 可視性を示します。使用可能な値は、Local (組織に制限) および Global (組織および Community Manager) です。デフォルトは Local です。

ListParticipantCapabilities

参加者の機能を照会します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListParticipantCapabilities

ListParticipantCapabilities の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ListParticipantCapabilities の 2 番目の子

ParticipantId: 参加者をシステムに対して識別する内部数値 ID。

ListParticipantCapabilitiesResponse

ListParticipantCapabilities メソッドの応答文書

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListParticipantCapabilitiesResponse

ListParticipantCapabilitiesResponse の最初の子

ParticipantCapabilities エレメント。

ゼロ以上の ParticipantCapabilities の子

ParticipantCapability エレメント。 8 つのエレメントが含まれます。

- CapabilityId: その機能をシステムに対して識別する内部数値 ID。
- ParticipantId: その参加者をシステムに対して識別する内部数値 ID。
- ParticipantName: 参加者がハブ・コミュニティーに表示する名前。
- CapabililtyRole: 参加者がシステム内で持っている機能役割。使用可能な値:
  - Source
  - Target
  - SourceAndTarget
- CapabilityEnabled: ブール値。
- RoutingObjectRefId: その機能に関連するルーティング・オブジェクト参照をシステムに対して識別する内部数値 ID。
- RoutingObjectRefInfo: WBI-C 内のルーティング・オブジェクトは階層になっています。定義されるのは一度ですが、複数の場所で参照されることがあります。ルーティング・オブジェクト参照は、どこでルーティング・オブジェクトが参照されているかを一意的に識別します。これは、以下のエレメントを含む複合タイプです。

- RoutingObjectRefId: ルーティング・オブジェクト参照の内部数値 ID。
- RoutingObjectId: 参照されるルーティング・オブジェクトの内部数値 ID。
- RoutingObjectName: ルーティング・オブジェクトの名前。
- RoutingObjectVersion: ルーティング・オブジェクト・バージョン。
- RoutingObjectType: ユーザーのロケールにローカライズされる、そのルーティング・オブジェクトのタイプ。
- RoutingObjectTypeKey: そのルーティング・オブジェクトのタイプへのキー。例: Package、Protocol など
- RoutingObjectEnabled: ブール値。
- RoutingObjectParentRefId: 親ルーティング・オブジェクト参照の内部数値 ID。これはオプションのエレメントです。

• CapabilityChildren エレメント。これはオプションのエレメントです。ゼロ以上の CapabilityChild エレメントが含まれます。それぞれの CapabilityChild エレメントには、ParticipantCapability エレメントと同じ 8 つのエレメントが含まれます。

ListParticipantConnections

参加者の接続を照会します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListParticipantConnections

ListParticipantConnections の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ListParticipantConnections の 2 番目の子

SourceParticipantId: ソースとして参加者をシステムに対して識別する内部数値 ID。

ListParticipantConnections の 3 番目の子

TargetParticipantId: ターゲットとして参加者をシステムに対して識別する内部数値 ID。

ListParticipantConnectionsResponse

ListParticipantCapabilities メソッドの応答文書

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListParticipantConnectionsResponse

ListParticipantConnectionResponse の最初の子

ParticipantConnections エレメント。

ゼロ以上の ParticipantConnections の子

ParticipantConnection エレメント。 9 つのエレメントが含まれます。

- ConnectionId: その接続をシステムに対して識別する内部数値 ID。
- SourceParticipantId: ソースとして参加者をシステムに対して識別する内部数値 ID。
- SourceCapabilityId: ソース機能をシステムに対して識別する内部数値 ID。
- TargetParticipantId: ターゲットとして参加者をシステムに対して識別する内部数値 ID。
- TargetCapabilityId: ターゲット機能をシステムに対して識別する内部数値 ID。
- ActionId: アクションをシステムに対して識別する内部数値 ID。
- ActionName: アクションの表示名。

- TransformMapId: そのアクションに関連する変形マップを識別する内部数値 ID。これはオプションのエレメントです。
- ConnectionEnabled: ブール値。

ListTargets

システムで構成されているターゲットを照会します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListTargets

ListTargets の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ListTargetsResponse

ListTargets メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListTargetsResponse

ListTargetsResponse の最初の子

Targets

ゼロ以上の Targets の子

Target エレメント。 5 つのエレメントが含まれます。

- TargetId: ターゲットをシステムに対して識別する内部数値 ID。
- Description: ターゲットを記述するストリング。
- ClassName: ターゲット・クラスの名前。これはオプションのクラスです。
- TransportType Name: トランスポート・タイプ。
- TargetAttributes: 以下のいずれも含まないか、1 つ以上を含む複合タイプ。
  - TargetAttribute: 以下の属性を含む複合タイプです。これはオプションのエレメントです。
    - AttributeName: ターゲット属性の名前。
    - AttributeValue: ターゲット属性の値。これはオプションの値です。
- TargetConfigPoints: 3 つのターゲット構成ポイント: PreProcess、PostProcess、および SyncCheck があります。これらはそれぞれ、以下のエレメントを含む複合タイプで表されます。
  - <ConfigPointName>: 以下のエレメントを含む複合タイプです。
    - Handlers: 以下のいずれも含まないか、1 つ以上を含む複合タイプ。
      - Handler: 以下の 3 つのエレメントを含む複合タイプです。これはオプションのエレメントです。

- ClassName: ハンドラー・クラスの名前。
- HandlerType: ハンドラー・タイプ。
- HandlerAttributes: 以下のいずれも含まないか、1 つ以上を含む複合タイプ。
  - HandlerAttribute: 以下の 2 つのエレメントを含む複合タイプです。これはオプションのエレメントです。
    - AttributeName: 属性の名前。
    - AttributeValue: 属性の値。これはオプションのエレメントです。

ListEventDefinitions

システムで構成されているイベントを照会します。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListEventDefinitions

ListEventDefinitions の最初の子

UserInfo エレメント。これはコンソールを介してログインした際に使用される情報と同じです。これには、3 つのエレメントが含まれます。

- UserName: Console ログイン・ユーザー名。
- Password: Console ログイン・パスワード。
- ParticipantLogin: 参加者 (会社) ログイン名。

ListEventDefinitionsResponse

ListEventsDefinitions メソッドの応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

ListEventsDefinitionsResponse

ListEventsDefinitionsResponse の最初の子

EventDefinitions

ゼロ以上の EventDefinitions の子

EventDefinition エレメント。以下の 6 つのエレメントを含む複合タイプです。これはオプションのエレメントです。

- EventCode: そのイベントのコード。
- Event name: イベントの名前。
- InternalDescription: イベントの内部特定記述が入っているストリング。
- Visibility: システムでのイベントの可視性。 3 つのエレメントを含む複合タイプです。
  - CommunityManager: ブール値
  - CommunityOperator: ブール値
  - CommunityParticipant: ブール値
- Severity: イベントのセキュリティー。使用可能な値:
  - Info
  - Debug
  - Warning
  - Error
  - Critical
- Alertable: ブール値

BCGPublicAPIException

例外の際の応答文書。

ルート・エレメント

BCGPublicAPI

最初の子エレメント

BCGPublicAPIException

BCGPublicAPIException の最初の子

ErrorMsg: エラー・メッセージが入っているストリング

第 11 章 外部イベント・デリバリーの使用法

WebSphere Business Integration Connect は、Connect システム内部でアクティビティーをモニターするのと同様の方法で、イベントを生成、保管します。イベントは内部キューにパブリッシュされ、WBI-C イベント・サーバーは内部キューからそれらのイベントを取り出します。次に、イベント・サーバーはイベントを内部イベント・ストアに送信します。このリリース以前は、Community Console Event Viewer を使用してのみ、これらのイベントのレコードにアクセスすることができました。Connect の 4.2.2 がリリースされて、イベントを外部 JMS キューにもデリバリーできるようになりました。アプリケーションのモニターなどのその他の関連する処理から、この外部 JMS キューのイベントを取り出すことができます。この章では、これらの処理の概要を説明します。この章には 2 つのセクションがあります。

- 『外部イベント・デリバリー・プロセス』
- 151 ページの『デリバリーされたイベントの構造』

外部イベント・デリバリー・プロセス

Connect システムには、文書イベントとメッセージ・イベントの 2 つの異なるイベント・タイプがあります。文書イベントは、ビジネス文書と直接関連したイベントです。ビジネス・プロセス・エンジンは、これらのイベントを WBI-C 内部キューにパブリッシングします。文書の状態が送信または失敗のいずれかの場合は、Delivery Manager はビジネス文書イベントもこのキューにパブリッシュします。一方、メッセージ・イベントは、WBI-C のすべてのコンポーネントによってパブリッシュされます。メッセージ・イベントは、それらのいくつかがビジネス文書と関連している場合もあります が、必ずしもビジネス文書と関連しているわけではありません。

内部キューにパブリッシュされたイベントは、イベント・サーバーによって Connect のイベント・ストアに送信されます。ただし、リリース 4.2.2 では、ユーザーはイベントを外部 JMS キューにデリバリーするかどうかを決定する必要があります。外部デリバリーのオン/オフや外部キューの構成は、Community Console を使用して行われます。これらの設定については、「*ハブ構成ガイド*」を参照してください。

イベントは CBE XML 形式の JMS キューにデリバリーされます。CBE (Common Base Event) 形式は、IBM イニシアチブが発展した CEI (Common Event Infrastructure) の一部であり、アプリケーション間のイベントの処理を標準化する目的で設計されています。 CBE 構造は情報の 3 つの基本タイプをカバーします。

- CBE 標準プロパティーは、作成時刻、イベント・タイプ、ソース、重大度などの詳細情報から構成されます。
- CBE コンテキスト・データは、イベントが生成される環境に関する情報を含みます。
- CBE 拡張データは、イベント・タイプに固有の汎用データを保持します。

CBE 形式が外部イベント・デリバリーで使用される際の特性については、 151 ページの『デリバリーされたイベントの構造』に詳細が記載されています。

外部デリバリーがオンの場合、すべてのイベントが外部キューにデリバリーされます。 Console Event Viewer でユーザーが表示できるイベント・タイプを制限する可視/不可視のフラグは、外部デリバリーでは使用されません。外部デリバリー内のイベントの名前と説明は、イベント・ビューアー内と同じ方法でローカライズされます。

誤った JMS 構成または JMS プロバイダーの問題により、外部イベント・デリバリーでエラーが発生する場合があります。開始時には検出されず、外部デリバリー・エラーが発生した場合は、次のようなことが起きます。

- 以降のイベント・デリバリーはオフになります。
- ユーザーが以下の方法でシステムを再度初期化した場合にのみ、イベントはシステムの再始動時に再度デリバリーされます。
  - コンソールのイベント・パブリッシュ・プロパティー 画面で JMS プロパティーを修正、更新した場合 (詳しくは、「*ハブ構成ガイド*」およびコンソール・オンライン・ヘルプ情報を参照してください)。

  または
  - JMS プロバイダーの問題を修正し、コンソールの *イベント・パブリッシュ・プロパティー* 画面で「**保管**」をクリックした場合。
- WBI-C アラート・エンジンがアラートを生成できるようにするため、アラート可能なイベントはログに記録されます。何らかの理由でアラート・イベントをログに記録できない場合は、そのイベントは無視されます。このイベントのロギングの再試行は行われません。
- 通常の内部イベント処理は正常に続行します。

デリバリーされたイベントの構造

このセクションでは、外部 JMS キューにデリバリーされたイベントの構造について取り扱います。

CBE イベント文書構造

CBE イベント文書構造は複雑であるため、2 つの部分に分けて説明します。

- 『CBE イベント文書構造』
- 『WBI-C メッセージ・イベントおよびビジネス文書イベントの CBE イベント構造』

CBE 構造の完全な説明については、
\B2BIntegrate\events\schemas\commonbaseevent1_0_1.xsd にあるスキーマ・ファイルを参照してください。同じディレクトリーに、eventdelivery.xsd という追加のスキーマ・ファイルがあります。このファイルは、メイン・スキーマを拡張する WBI-C を定義します。ここでは、メイン・スキーマの situation エレメントで使用される SituationType タイプの OtherSituation タイプを定義します。CBE およびスキーマに関する詳細情報については、Hyades プロジェクト www.eclipse.org/hyades/ の Eclipse.org Web サイトを参照してください。

基本的な CBE 文書構造

CommonBaseEvent 文書のルート・エレメントは CommonBaseEvent エレメントです。CommonBaseEvent の子は以下のとおりです。

- contextDataElements: イベントのコンテキストを提供します。これはオプションのイベントです。 WBI-C では提供していません。
- extendedDataElements: 基本的な CBE 構造から直接取り込まない情報を取り込みます。これはオプションのエレメントです。 WBI-C で提供します。
- associatedEvents: 関連したイベントを取り込みます。これはオプションのエレメントです。 WBI-C では提供していません。
- reporterComponentId: イベントを報告するコンポーネントを指定します。これはオプションのエレメントです。 WBI-C では提供していません。
- sourceComponentId: イベントを生成したコンポーネントを指定します。これは必須のエレメントです。 WBI-C で提供します。
- msgDataElement: このイベントが保持しているメッセージと関連したすべての関連情報の指定に使用されるデータを指定します。これはオプションのエレメントです。メッセージ・イベント用に作成された CBE イベントで生成されます。ビジネス文書イベントの場合は、このエレメントは生成されません 。 WBI-C は常時、このエレメントを次のように生成します。
  ```
  <msgDataElement msgLocale="en-US"></msgDataElement>
  ```
- situation: イベントの原因となった状態のタイプを説明します。これは必須のエレメントです。 WBI-C で提供します。

WBI-C メッセージ・イベントおよびビジネス文書イベントの CBE イベント構造

このセクションでは、WBI-C 外部イベント・デリバリー・システムで生成されるイベント文書で提供される CBE エレメントの説明を、エレメントごとに取り扱いま

す。ここには、メインのエレメントの詳細な属性リストが含まれます。説明の中に、メッセージ・イベントおよびビジネス文書イベントに CBE XML で表示される簡単なエレメント例が含まれるものもあります。

CommonBaseEvent エレメント: これはすべての CBE イベント文書のルート・エレメントです。以下の表で、このエレメントとその属性について説明します。

表 2. CommonBaseEvent エレメント

プロパティー名	説明
Version	1.0.1 WBI-C 4.2.2 はこのバージョンのスキーマをサポートします。
localInstanceId	WBI-C ストア内の固有 ID: • メッセージ・イベント: ソース・イベントのイベント ID • ビジネス文書イベント: ビジネス文書の UUID
creationTime	この CBE イベントの作成時刻: • メッセージ・イベント: イベントの作成時刻 • ビジネス文書イベント: ビジネス文書にロギング時刻が保管されないため、現在時刻を設定します。
severity	• メッセージ・イベント: – デバッグ: 8 – 通知: 10 – 警告: 30 – エラー: 50 • ビジネス文書イベント: ビジネス文書には重大度がないため、10 (通知) を設定します。
priority	WBI-C には優先順位の概念がありません。常時 50 に設定します。
msg	• メッセージ・イベント: ローカライズされたこのイベントの説明 • ビジネス文書イベント: 指定されません
repeatCount	WBI-C では指定されません
elapsedTime	WBI-C では指定されません
extensionName	メッセージ・イベントとビジネス文書イベントの区別に使用されます。 • メッセージ・イベント: BCG_EVENT • ビジネス文書イベント: BCG_BUSINESSDOCUMENT
sequenceNumber	WBI-C では指定されません

これはメッセージ・イベントの CommonBaseEvent エレメントのサンプルです。

```
<cbe:CommonBaseEvent
         creationTime="2004-06-20T06:26:01"
         extensionName="BCG_EVENT"
```

```
         localInstanceId="1087712761674000C766F006F0178601DF89630A39DF6CA"
         msg="ASValidation"
          priority="50"
          severity="10"
          version="1.0.1"
         xsi:schemaLocation=
                        "http://www.ibm.com/AC
                        /commonbaseevent1_0_1commonbaseevent1_0_1.xsd">
:
:
<cbe:CommonBaseEvent />
```

これはビジネス文書イベントの CommonBaseEvent エレメントのサンプルです。

```
<cbe:CommonBaseEvent
          creationTime="2004-06-20T06:26:02"
          extensionName="BCG_BUSINESSDOCUMENT"
          localInstanceId="1087712759944000C766F006F017860B7583EB51E26A336"
          priority="50"
          severity="10"
          version="1.0.1"
          xsi:schemaLocation="http://www.ibm.com/AC
             /commonbaseevent1_0_1 commonbaseevent1_0_1.xsd">
:
:
<cbe:CommonBaseEvent />
```

sourceComponentId エレメント: このエレメントはイベントを生成したコンポーネントを指定します。 WBI-C は CBE の標準的な方法でこれを入力します。詳しくは、スキーマを参照してください。

situation エレメント: このエレメントはイベントを生成した状態のタイプを説明します。以下の表で、このエレメントとその属性について説明します。

表 3. situation エレメント

プロパティー名	説明
categoryName	OtherSituation
reasoningScope	INTERNAL
faultType	WBI-C はこの属性を eventdelivery.xsd の OtherSituation に定義します。 メッセージ・イベント: • SOURCE • TARGET • SYSTEM • UNKNOWN ビジネス文書イベント: • UNKNOWN

これはメッセージ・イベントの situation エレメントの例です。

```
<cbe:situation categoryName="OtherSituation">
               <cbe:situationType
               reasoningScope="INTERNAL"
                xsi:type="cbe:OtherSituation">
               <bcg:faultType/>
           </cbe:situationType>
</cbe:situation>
```

extendedDataElements エレメント: このエレメントは基本的な CBE 構造から直接取り込まない情報を取り込みます。以下の 3 つの表は、このエレメント、その属性、および特殊な子エレメントについて説明します。ここでは、メッセージ・イベント拡張エレメントおよびビジネス文書イベント拡張エレメントを取り上げます。

表 4. extendedDataElements エレメント

プロパティー名	説明
name	メッセージ・イベントとビジネス文書イベントの区別に使用されます。 • メッセージ・イベント: BCG_EVENT • ビジネス文書イベント: BCG_BUSINESSDOCUMENT
type	WBI-C はこれを noValue に設定します。
children	イベントのタイプ (メッセージまたはビジネス文書) に基づき、1 つ以上のエレメントが作成されます。以下の表の説明

メッセージ・イベント拡張データ・エレメント:

表 5. メッセージ・イベント拡張データ・エレメント

名前	値
BCG_EVENTCD	メッセージ・イベントからのイベント・コード。
BCG_HOSTIPADDRESS	ホスト IP アドレス。適用できる場合に指定します。
BCG_PARTNERID1	参加者の内部 ID。適用できる場合に指定します。
BCG_PARTNERID2	参加者の内部 ID。適用できる場合に指定します。
BCG_STACKTRACE	スタック・トレース。適用できる場合に指定します。
BCG_FRIPADDRESS	発信元 IP アドレス。適用できる場合に指定します。
BCG_PARENTBCGDOCID	親のビジネス文書の固有 ID。適用できる場合に指定します。
BCG_BCGDOCID	このメッセージ・イベントが関連付けられるビジネス文書の ID。 **注:** アプリケーションのモニターで、このイベントを関連したビジネス文書と関係付けるのに、このエレメントを使用できます。
BCG_USERID	ユーザー ID。適用できる場合に指定します。
BCG_BUSINESSID1	発信元ビジネス ID。適用できる場合に指定します。
BCG_INITBUSINESSID	開始ビジネス ID。適用できる場合に指定します。
BCG_INITASMESSAGEID	開始 AS メッセージ ID。適用できる場合に指定します。

表 5. メッセージ・イベント拡張データ・エレメント (続き)

名前	値
BCG_BUSINESSID2	宛先ビジネス ID。適用できる場合に指定します。

メッセージ・イベントの extendedDataElements エレメントの例の一部です。

```
<cbe:extendedDataElements name="BCG_EVENT" type="noValue">
        <cbe:values/>
               <cbe:children name="BCG_EVENTTIMESTAMP" type="string">
               <cbe:values>1087712761674</cbe:values>
           </cbe:children>
              <cbe:children name="BCG_PARENTBCGDOCID" type="string">
              <cbe:values>10877127599440000C766F006F017860B7583EB51E26A336
              </cbe:values>
          </cbe:children>
              <cbe:children name="BCG_ARGUMENTSTRING" type="string">
              <cbe:values>ASValidation</cbe:values>
          </cbe:children>
              <cbe:children name="BCG_HOSTIPADDRESS" type="string">
             <cbe:values>127.0.0.1</cbe:values>
          </cbe:children>

  :
  :
  :

   </cbe:extendedDataElements>
```

ビジネス文書イベント拡張データ・エレメント:

表 6. ビジネス文書イベント拡張データ・エレメント

属性	値
BCG_BCGDOCID	ビジネス文書の固有文書 ID。
BCG_PARENTBCGDOCID	親のビジネス文書の文書 ID。適用できる場合に指定します。
BCG_DOCLOCATION	ビジネス文書の絶対パスによるロケーション。適用できる場合に指定します。
BCG_DOCSTATE	ビジネス文書の現行状態。 • DOC_IN_PROCESS=″In Process″ • DOC_SENT = ″Sent″ • DOC_RECEIVED = ″Received″ • DOC_FAILED = ″Failed″
BCG_DOCSIZE	ビジネス文書から取得。適用できる場合に指定します。

表 6. ビジネス文書イベント拡張データ・エレメント (続き)

属性	値
ビジネス文書関連データ。	さらに、ビジネス文書イベントには、その他の情報が含まれる場合があります。 • ルーティング関連データ • フロー関連データ • ビジネス・プロトコル関連データ 子エレメントの名前の属性は、BCGDocumentConstants クラスに指定された定数の 1 つに設定されます。詳しくは、 86 ページの『BCGDocumentConstants』 を参照してください。 適用できる場合にのみ指定します。

ビジネス文書イベントの extendedDataElements エレメントの例の一部です。

```
<cbe:extendedDataElements name="BCG_BUSINESSDOCUMENT" type="noValue">
   <cbe:values/>
          <cbe:children name="BCG_BCGDOCID" type="string">
         <cbe:values>1087712755684000C766F006F01786046684D6EAC6FAC22
         </cbe:values>
      </cbe:children>
          <cbe:children name="BCG_DOCLOCATION" type="string">
            <cbe:values>
              /opt/IBM/WBIConnect/common/data/Inbound/process
              /520/D9
              /1087712753565000C766F006F003149F07FF1FC6C41D8D9.ascontent
        </cbe:values>
      </cbe:children>
          <cbe:children name="BCG_PARENTBCGDOCID" type="string">
        <cbe:values>1087712753565000C766F006F003149F07FF1FC6C41D8D9
        </cbe:values>
      </cbe:children>
          <cbe:children name="BCG_DOCSTATE" type="string">
        <cbe:values>In Process</cbe:values>
      </cbe:children>
          <cbe:children name="BCG_DOCRESTARTED" type="string">
        <cbe:values>false</cbe:values>
      </cbe:children>
          <cbe:children name="BCG_FRPARTNERTYPE" type="string">
        <cbe:values>0</cbe:values>
      </cbe:children>

      :
      :
      :

   </cbe:extendedDataElements>
```

索引

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

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

C

X

特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

著作権ライセンス

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

Websphere Business Integration Connect には、ICU4J というコードが含まれています。ICU4J のコードは、IBM の「プログラムのご使用条件」に基づきその「適用除

外コンポーネント」の条項に従うことを条件に使用許諾されます。ただし、IBM は以下の条項を明示することを義務付けられています。

著作権および許可に関する注意事項

本「プログラム」は、IBM 社およびその他の著作権により保護されています。
Copyright (c) 1995-2003

All rights reserved.

このソフトウェアおよびその関連文書ファイル (以下「ソフトウェア」といいます) を取得する人には、この「ソフトウェア」の、使用、複製、変更、結合、出版、配布またはソフトウェアの複製を販売する権利を含め、制約なく取引する権利を無償で許可し、また、「ソフトウェア」を与えられた人にも、この権利が与えられます。ただし、上記の著作権表示およびこの許可通知が、すべてのこの「ソフトウェア」の複製に記載され、また上記の著作権表示およびこの許可通知が、関連文書に記載されている場合に限ります。

ソフトウェアは、特定物として現存するままの状態で提供され、第三者の権利の不侵害の保証、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または保証条件を負わないものとします。さらに、著作権者またはこの注意事項に含まれている権利の所有者は、このソフトウェアの使用または実行に起因するものであれ、関連するものであれ、契約、過失、不法行為のいずれによるものであれ、使用、データまたは利益の喪失から発生する請求、あるいは特別、直接的、間接的、結果的損害、または他の一切の損害について、何等の責任も負いません。

この通知に記されているもの、および事前の書面による承認がある場合を除き、著作権者の名前を、このソフトウェアの広告、または販売、使用、取引の促進のためにご使用にな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。

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

WebSphere Business Integration Connect Enterprise and Advanced Editions には、Eclipse Project (www.eclipse.org) で開発されたソフトウェアが含まれます。

IBM WebSphere Business IntegrationConnect Enterprise and Advanced Editions バージョン 4.2.2