WebSphere Business Integration Server
Express and Express Plus

Adapter for SWIFT ユーザーズ・ガイド

バージョン 4.3.1

WebSphere Business Integration Server
Express and Express Plus

Adapter for SWIFT ユーザーズ・ガイド

バージョン 4.3.1

お願い

本書および本書で紹介する製品をご使用になる前に、135 ページの『付録 D. 特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Server Express バージョン 4.3.1、IBM WebSphere Business Integration Server Express Plus バージョン 4.3.1、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： WebSphere Business Integration Server Express and Express Plus
Adapter for SWIFT User Guide
Version 4.3.1

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

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

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

目次

本書について

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

本書では、WebSphere Business Integration Server Express Adapter for SWIFT の構成、ビジネス・オブジェクト開発、およびトラブルシューティングについて説明します。

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

対象読者

本書は、WebSphere Business Integration Server Express 製品をお客様のサイトでサポートおよび管理する、コンサルタント、開発者、およびシステム管理者を対象としています。

本書の前提条件

本書の読者には以下に関する知識が必要です。

- WebSphere Business Integration Server Express システム
- ビジネス・オブジェクト開発
- WebSphere MQ アプリケーション
- SWIFT 製品スイートとプロトコル

関連資料

本書の対象製品の一連の関連文書には、WebSphere Business Integration Server Express のどのインストールにも共通する機能とコンポーネントの解説のほか、特定のコンポーネントに関する参考資料が含まれています。

関連文書は、http://www.ibm.com/websphere/wbiserverexpress/infocenter でダウンロード、インストール、および表示することができます。

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

情報や速報は、WebSphere Business Integration のサポート Web サイト (http://www.ibm.com/software/integration/websphere/support/) で参照できます。適切なコンポーネント領域を選択し、「Technotes (技術情報)」セクションと「Flashes (速報)」セクションを参照してください。

表記上の規則

本書では、以下のような規則を使用しています。

`Courier フォント`	コマンド名、ファイル名、入力情報、システムが画面に出力した情報などのリテラル値を示します。
太字	初出語を示します。
イタリック、***イタリック***	変数名または相互参照を示します。
青のアウトライン	オンラインで表示したときにのみ見られる青のアウトラインは、相互参照用のハイパーリンクです。アウトラインの内側をクリックすると、参照先オブジェクトにジャンプします。
ProductDir	IBM WebSphere Business Integration Server Express for Adapters 製品がインストールされているディレクトリーを表します。各プラットフォームのデフォルトは、次のとおりです。 Windows: IBM¥WebSphereServer OS/400: /QIBM/ProdData/WBIServer43/product Linux: /home/${username}/IBM/WebSphereServer
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、`option[,...]` は、複数のオプションをコンマで区切って指定できることを意味します。
< >	命名規則では、不等号括弧は名前の個々の要素を囲み、各要素を区別します。(例: `<server_name><connector_name>tmp.log`)
/、¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。OS/400 と Linux では、ディレクトリー・パスにスラッシュ (/) が使用されています。WebSphere Business Integration Server Express システム製品のパス名は、同製品がインストールされているシステムのディレクトリーからの相対位置になっています。
%text% および *$text*	% 記号で囲まれたテキストは、Windows(TM) の `text` システム変数またはユーザー変数の値を示します。UNIX 環境での同等の表記は *$text* です。これは、UNIX 環境変数 *text* の値を示します。

本リリースの新機能

リリース 4.3.1 の新機能

今回のリリースでは、次のオペレーティング・システムに対するサポートが追加されています。

- Microsoft Windows 2003
- IBM OS/400 V5R2、V5R3
- Red Hat Enterprise Linux AS 3.0, Update 1
- SuSE Linux Enterprise Server 8.1 with SP3

リリース 4.3 の新機能

本書の最初のリリースです。

第 1 章 概要

Connector for SWIFT は、WebSphere Business Integration Server Express and Express Plus Adapter for SWIFT のランタイム・コンポーネントです。このコネクターを使用すると、WebSphere InterChange Server Express 統合ブローカーと、SWIFT 対応のビジネス・プロセスとの間で、ビジネス・オブジェクトを交換できます。

注: 特に明記しない限り、本書全体を通じて、SWIFT メッセージは SWIFT FIN メッセージを意味します。

コネクターは、アプリケーション固有のコンポーネントとコネクター・フレームワークから成り立っています。アプリケーション固有のコンポーネントには、特定のアプリケーションに応じて調整されたコードが含まれます。コネクター・フレームワークは統合ブローカーとアプリケーション固有のコンポーネントの間の仲介役として機能し、そのコードはどのコネクターにも共通です。コネクター・フレームワークは、統合ブローカーとアプリケーション固有のコンポーネントとの間で以下のようなサービスを提供します。

- ビジネス・オブジェクトの受信と送信
- 始動メッセージや管理メッセージの交換の管理

本書では、アプリケーション固有のコンポーネントおよびコネクター・フレームワークについて説明します。本書では、この 2 つのコンポーネントをまとめてコネクターと呼びます。

統合ブローカーとコネクターの関係の詳細については、「*システム・インプリメンテーション・ガイド*」を参照してください。

すべての WebSphere Business Integration Server Express アダプターでは、統合ブローカーとして WebSphere InterChange Server Express を使用できます。

Connector for SWIFT を使用すると、InterChange Server Express は SWIFT メッセージの形式でデータを送受信するアプリケーションとビジネス・オブジェクトを交換できます。

アダプター環境

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

- 『アダプターの規格』
- 『アダプターのプラットフォーム』

アダプターのプラットフォーム

このアダプターは、次のプラットフォームで動作します。

- Microsoft Windows 2000
- Microsoft Windows 2003
- IBM OS/400 V5R2、V5R3
- Red Hat Enterprise Linux AS 3.0, Update 1
- SuSE Linux Enterprise Server 8.1 with SP3

アダプターの規格

アダプターは次の規格をサポートしています。

SWIFTAlliance Access

SWIFTAlliance Access Gateway は、IP を介してリモートの金融アプリケーションとの間で SWIFT メッセージを送受信する窓口となります。コネクターは SWIFTAlliance Access 5.0 をサポートしています。

ロケール依存データ

コネクターは、国際化対応済みであるため、2 バイト文字セットをサポートし、指定の言語でメッセージ・テキストを配信することができます。コネクターは、1 つの文字コード・セットを使用する場所から別のコード・セットを使用する場所にデータを転送するとき、データの意味を保存するように文字変換を実行します。

Java 仮想マシン (JVM) 内での Java ランタイム環境は、Unicode 文字コード・セットでデータを表します。Unicode には、ほとんどの既知の文字コード・セット (1 バイト系とマルチバイト系を含む) の文字に対応できるエンコード方式が組み込まれています。WebSphere Business Integration Server Express システムのコンポーネントは、ほとんどの場合、Java で書かれています。したがって、WebSphere Business Integration Server Express システム・コンポーネント間でデータが転送される場合は、通常、文字変換の必要がありません。

エラー・メッセージと通知メッセージを適切な言語で適切な国と地域に合わせて記録するには、該当する環境の Locale 標準構成プロパティーを設定します。構成プロパティーの詳細については、97 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

コネクター・アーキテクチャー

コネクターを使用すると、WebSphere ビジネス・プロセスは、データが変更された場合に SWIFT メッセージを発行または受信するアプリケーションとの間でビジネス・オブジェクトを非同期で交換できます (コネクターは、同期確認通知もサポートしています)。

SWIFT は、Society for Worldwide Interbank Financial Telecommunications の略語です。これは、金融メッセージング規格の作成と保守を目的とする、国連に認可された国際標準化機構 (ISO) です。

図 1 に示されているように、コネクターはいくつかのコンポーネント (WebSphere コンポーネントは**太字**で示されています) と対話します。これらのコンポーネントは、WebSphere ビジネス・オブジェクトの世界と SWIFT メッセージの世界の橋渡しをします。IBM WebSphere Business Integration Server Express and Express Plus の場合、図 1 の統合ブローカーは WebSphere InterChange Server Express です。

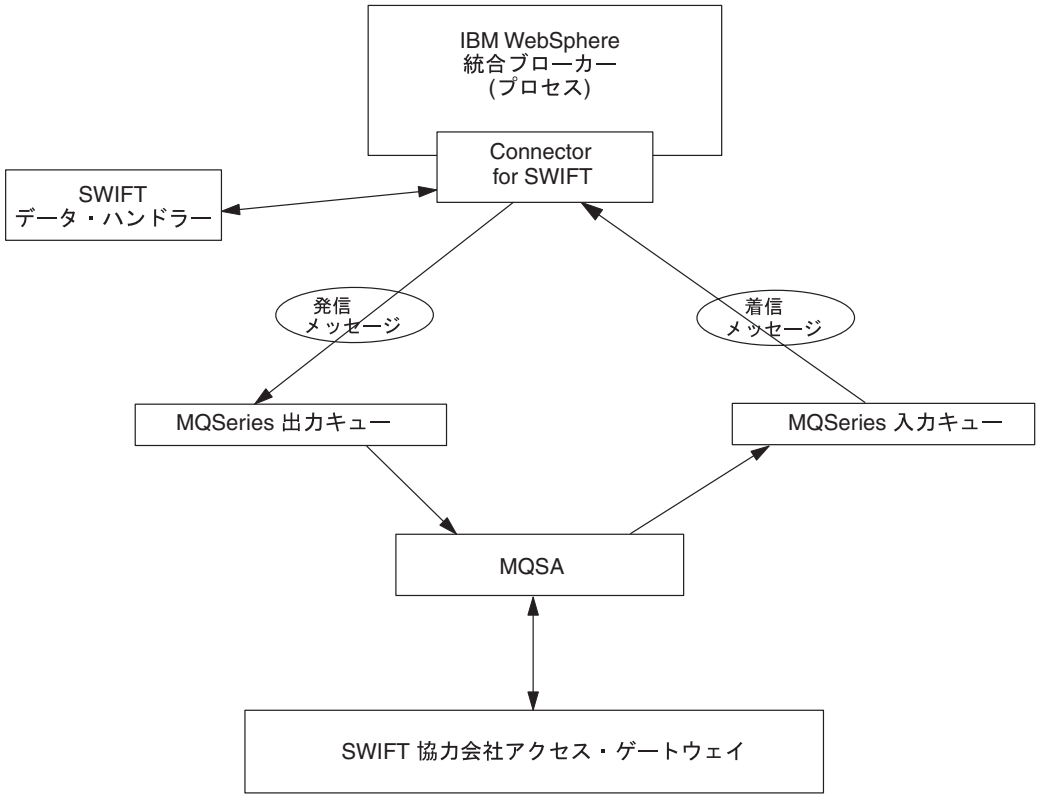


図 1. SWIFT アーキテクチャーのコネクター

SWIFT 環境を構成するさまざまなコンポーネントについて以下に説明します。

Connector for SWIFT

Connector for SWIFT はメタデータ主導型です。メッセージ・ルーティングおよびフォーマット変換は、イベント・ポーリング手法によって開始されます。コネクタ

ーはキューから WebSphere MQ メッセージを検索し、SWIFT データ・ハンドラーを呼び出してメッセージを対応するビジネス・オブジェクトに変換し、そのオブジェクトを対応するビジネス・プロセスにデリバリーします。反対方向の場合、コネクターは WebSphere InterChange Server Express からビジネス・オブジェクトを受け取り、同じデータ・ハンドラーを使用して SWIFT メッセージに変換し、WebSphere MQ キューにデリバリーします。

メッセージの処理に使用されるビジネス・オブジェクトのタイプと動詞は、WebSphere MQ メッセージのヘッダーに含まれる Format フィールドのメタデータによって決定されます。ビジネス・オブジェクト名と動詞を格納するメタオブジェクトを構築し、WebSphere MQ メッセージ・ヘッダーの Format フィールドのテキストに関連付けます。

オプションで、コネクターに渡されるビジネス・オブジェクトに子として追加される動的なメタオブジェクトを構成することもできます。子メタオブジェクトの値は、コネクター全体を対象として指定されている静的なメタオブジェクトに指定されている値をオーバーライドします。子メタオブジェクトが定義されていない場合や、必要な変換プロパティーが定義されていない場合は、コネクターはデフォルトで、その値に対する静的なメタオブジェクトを調べます。単一の静的なコネクター・メタオブジェクトの代わりに、またはそれを補足するために、1 つ以上の動的な子メタオブジェクトを指定することができます。

コネクターは複数の入力キューのポーリングが可能で、各キューをラウンドロビン方式でポーリングして、各キューから構成可能な数のメッセージを検索します。ポーリング中に検索するメッセージごとに、コネクターは動的な子メタオブジェクト (ビジネス・オブジェクトに指定されている場合) を追加します。子メタオブジェクトの値はコネクターに対し、メッセージ・フォーマットとメッセージが検索された入力キューの名前を、属性に取り込むよう指示できます。

入力キューからメッセージを検索する場合、コネクターは、FORMAT テキスト・フィールドに対応するビジネス・オブジェクト名を探します。次に、ビジネス・オブジェクトの名前とともに、メッセージがデータ・ハンドラーに渡されます。ビジネス・オブジェクトにメッセージの内容が正常に取り込まれると、コネクターはコラボレーションがそのビジネス・オブジェクトにサブスクライブしているかどうかをチェックしてから、gotApplEvents() メソッドを使用して InterChange Server Express にデリバリーします。

SWIFT データ・ハンドラー

コネクターは SWIFT データ・ハンドラーを呼び出して、ビジネス・オブジェクトを SWIFT メッセージに (あるいは SWIFT メッセージをビジネス・オブジェクトに) 変換します。SWIFT データ・ハンドラーの詳細については、91 ページの『第 4 章 SWIFT データ・ハンドラー』を参照してください。

WebSphere MQ

Connector for SWIFT は、エンタープライズ・メッセージング・システムにアクセスするための API である JavaTM Message Service (JMS) の MQ インプリメンテーションを使用します。これによって、着信および発信 WebSphere MQ イベント・キューとの対話が可能になります。

MQSA

WebSphere MQ イベント・キューは、WebSphere MQ Interface for SWIFTAlliance (MQSA) との間でメッセージを交換します。MQSA ソフトウェアは WebSphere MQ メッセージング機能と SWIFT メッセージ・タイプを統合し、デリバリー、確認通知、キュー管理、タイム・スタンプ、およびその他の機能を実行します。

SWIFTAlliance Access

SWIFTAlliance Access Gateway は、IP を介してリモートの金融アプリケーションとの間で SWIFT メッセージを送受信する窓口となります。コネクターは SWIFTAlliance Access 5.0 をサポートしています。

アプリケーションとコネクターの通信方式

コネクターは、IBM WebSphere MQ にインプリメントされている Java Message Service (JMS) を使用します。JMS は、エンタープライズ・メッセージング・システムにアクセスするためのオープン・スタンダード API です。これは、ビジネス・アプリケーションがビジネス・データとイベントを非同期で送受信できるように設計されています。

メッセージ要求

図 2 に、メッセージ要求通信を示します。

1. コネクター・フレームワークは ISO 15022 SWIFT メッセージを表すビジネス・オブジェクトを統合ブローカーから受け取ります。IBM WebSphere Business Integration Server Express and Express Plus の場合、図 2 の統合ブローカーは WebSphere InterChange Server Express です。
2. コネクターはそのビジネス・オブジェクトをデータ・ハンドラーに渡します。
3. データ・ハンドラーは、ISO 15022 ビジネス・オブジェクトを、ISO 15022 に準拠した SWIFT メッセージに変換します。
4. コネクターは ISO 15022 SWIFT に準拠したメッセージを WebSphere MQ 出力キューにディスパッチします。
5. JMS レイヤーは、適切な呼び出しによってキュー・セッションを開き、メッセージを MQSA に発送し、MQSA は SWIFT Alliance Gateway にメッセージを発行します。

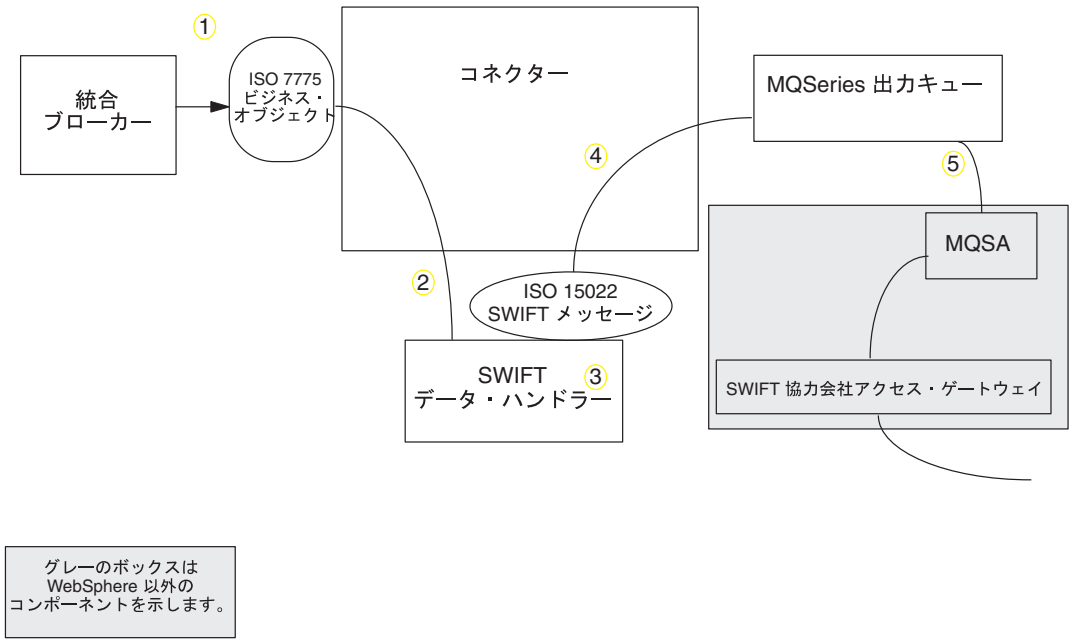


図 2. アプリケーションとコネクターの通信方式: メッセージ要求

イベント・デリバリー

図 3 に、メッセージ返送の通信を示します。

1. ポーリング・メソッドは、該当する次の ISO 15022 SWIFT メッセージを WebSphere MQ キューから検索します。
2. メッセージは、進行中キューにステージされ、処理が完了するまでそこにとどまります。
3. データ・ハンドラーはメッセージを ISO 15022 ビジネス・オブジェクトに変換します。
4. SWIFT データ・ハンドラーは ISO 15022 ビジネス・オブジェクトを受信し、その中の動詞を、データ・ハンドラー固有のメタオブジェクトで指定されたデフォルトの動詞に設定します。
5. コネクターは、ビジネス・オブジェクトが統合ブローカーによってサブスクライブされているかどうかを判別します。サブスクライブされている場合、コネクター・フレームワークがビジネス・オブジェクトを統合ブローカーにデリバリーし、進行中のキューからメッセージが削除されます。IBM WebSphere Business Integration Server Express and Express Plus の場合、図 3 の統合ブローカーは WebSphere InterChange Server Express です。

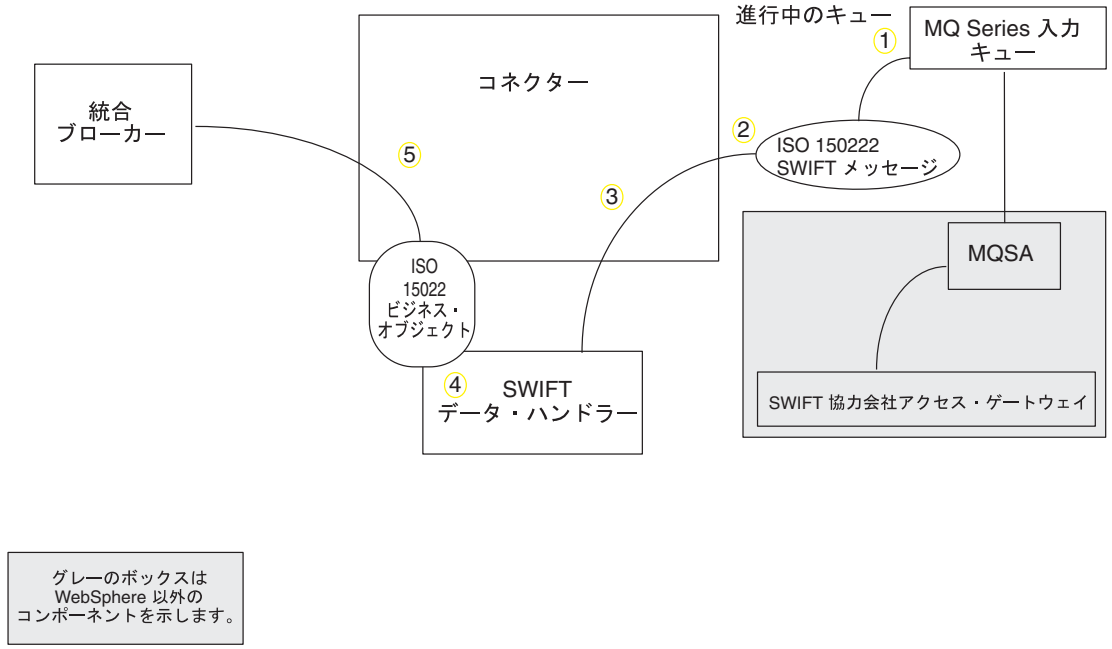


図 3. アプリケーションとコネクターの通信方式: イベント・デリバリー

イベント処理

イベント通知のため、コネクターは、データベース・トリガーではなくアプリケーションによってキューに書き込まれたイベントを検出します。イベントは、SWIFTAlliance が SWIFT メッセージを生成し、それを WebSphere MQ キューに保管したときに発生します。

検索

コネクターはポーリング・メソッドを使用して WebSphere MQ 入力キューからメッセージを定期的にポーリングします。メッセージが見つかった場合、コネクターはメッセージを WebSphere MQ 入力キューから取り出して検討し、そのフォーマットを判断します。フォーマットがコネクターの静的メタオブジェクトまたは子メタオブジェクトで定義されている場合、コネクターはデータ・ハンドラーを使用して動詞を持つ適切なビジネス・オブジェクトを生成します。

進行中のキュー

コネクターがメッセージを処理するときには、最初に WebSphere MQ キューへのトランザクション・セッションが開始されます。このトランザクションのアプローチによって、ビジネス・オブジェクトがビジネス・プロセスに 2 回デリバリーされる可能性があります。これは、コネクターがビジネス・オブジェクトのサブミットには成功するが、キュー内のトランザクションのコミットに失敗することが原因です。この問題を回避するために、コネクターは、すべてのメッセージを進行中キューに移動します。進行中キューでは、メッセージは処理が完了するまで保持されま

す。処理中にコネクターの予期しないシャットダウンが発生した場合、進行中キュー内のメッセージは、元の WebSphere MQ キューに復元されずにそのまま保持されます。

注: JMS サービス・プロバイダーとのトランザクション・セッションでは、キュー内の要求されたアクションがすべて実行され、キューからイベントが除去される前にコミットされる必要があります。そのため、コネクターはキューからメッセージを取り出すと、1) メッセージがビジネス・オブジェクトに変換されるか、2) ビジネス・オブジェクトが InterChange Server Express に送信されるか、3) 戻り値が受信されるまで、検索にコミットしません。

同期確認通知

Connector for SWIFT は、発行した要求に関するフィードバックを必要とするアプリケーションをサポートするために、レポート・メッセージをアプリケーションに返送します。このレポート・メッセージには、アプリケーションからの要求の処理が完了した時点での結果が詳細に記述されます。

この処理を実現するために、コネクターはこのような要求のビジネス・データを InterChange Server Express に同期的に通知します。ビジネス・オブジェクトを正常に処理した場合、コネクターは InterChange Server Express からの戻りコードとビジネス・オブジェクトのすべての変更を含むレポートを、要求を発行したアプリケーションに返送します。コネクターまたは InterChange Server Express がビジネス・オブジェクトの処理に失敗した場合、コネクターは、該当するエラー・コードとエラー・メッセージを含むレポートを返送します。

いずれの場合も、Connector for SWIFT に要求を送信するアプリケーションは、要求の結果について通知されます。

Connector for SWIFT が肯定確認通知レポートまたは否定確認通知レポート (PAN または NAN) を要求するメッセージを受け取った場合、コネクターはそのメッセージの内容を InterChange Server Express に同期的に通知し、レポート・メッセージに戻りコードと変更されたビジネス・データを組み込んで、要求を発行したアプリケーションに返送します。

表 1 に、コネクターに送信されたメッセージが同期的に処理されるために必要な構造を示します。

表 1. 同期 WebSphere MQ メッセージに必要な構造

MQMD フィールド (メッセージ記述子)	説明	サポートされる値 (複数の値がある場合は OR として扱います)
MessageType	メッセージ・タイプ	DATAGRAM

表 1. 同期 WebSphere MQ メッセージに必要な構造 (続き)

MQMD フィールド (メッセージ記述子)	説明	サポートされる値 (複数の値がある場合は OR として扱います)
Report	必要なレポート・メッセージのオプション	次のいずれか一方、または両方を指定できます。 • MQRO_PAN コネクターは、ビジネス・オブジェクトが正常に処理された場合にレポート・メッセージを送信します。 • MQRO_NAN コネクターは、ビジネス・オブジェクトの処理中にエラーが発生した場合にレポート・メッセージを送信します。 次のいずれかの値を指定すると、レポート・メッセージの相関 ID の設定方法を制御できます。 • MQRO_COPY_MSG_ID_TO_CORREL_ID コネクターは、要求メッセージのメッセージ ID をレポートの相関 ID にコピーします。これはデフォルトのアクションです。 • MQRO_PASS_CORREL_IDコネクターは、要求メッセージの相関 ID をレポートの相関 ID にコピーします。
ReplyToQueue	応答キューの名前	レポート・メッセージの送信先となるキューの名前。
ReplyToQueueManager	キュー・マネージャーの名前	レポート・メッセージの送信先となるキュー・マネージャーの名前。
メッセージ本文		コネクターに構成されているデータ・ハンドラーと互換性のあるフォーマットで直列化されたビジネス・オブジェクト。

表 1 で説明したメッセージを受け取ると、コネクターは以下の処理を行います。

1. 構成されているデータ・ハンドラーを使用して、メッセージの本文に含まれるビジネス・オブジェクトを再構成します。
2. ビジネス・オブジェクトに指定されたビジネス・プロセスおよび静的メタデータ・オブジェクトの動詞を検索します。
3. 指定されたプロセスに、ビジネス・オブジェクトを同期的に通知します。
4. 処理の結果とビジネス・オブジェクトのすべての変更またはエラー・メッセージをカプセル化したレポートを生成します。
5. 要求の replyToQueue および replyToQueueManager フィールド内で指定されたキューに、レポートを送信します。

表 2 に、コネクターから要求を発行したアプリケーションに送信されるレポートの構造を示します。

表 2. 要求を発行したアプリケーションに返送されるレポートの構造

MQMD フィールド	説明	サポートされる値 (複数の値がある場合は OR として扱います)
MessageType	メッセージ・タイプ	REPORT
feedback	レポートのタイプ	次のいずれかです。 • MQRO_PAN ビジネス・オブジェクトが正常に処理された場合に、レポートが返送されます。 • MQRO_NAN 要求の処理中にコネクターまたは InterChange Server Express がエラーを検出した場合に、レポートが返送されます。
メッセージ本文		ビジネス・オブジェクトが正常に処理された場合、コネクターはメッセージの本文に InterChange Server Express から戻されたビジネス・オブジェクトを取り込みます。このデフォルトの動作は、静的メタデータ・オブジェクトの DoNotReportBusObj プロパティーを true に設定することによりオーバーライドできます。 要求を処理できなかった場合、コネクターはメッセージの本文にコネクターまたは InterChange Server Express によって生成されたエラー・メッセージを取り込みます。

リカバリー

コネクターの初期化時には、コネクターのシャットダウンなどが原因で完全に処理されなかったメッセージが進行中キュー内にあるかどうかが検査されます。コネクター構成プロパティー InDoubtEvents を使用すると、このようなメッセージのリカバリー処理のための 4 つのオプション (fail on startup、reprocess、ignore、または log error) のうちの 1 つを指定できます。

始動時の失敗 (Fail on startup)

Fail on Startup オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、エラー・ログは記録されますが、コネクターは即時にシャットダウンします。メッセージを調べて適切な処置を行う (これらのメッセージを完全に削除するか、または別のキューに移動する) のは、ユーザーまたはシステム管理者の役割です。

再処理 (Reprocess)

Reprocess オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、以降のポーリング中にこれらのメッセージが最初に処理されます。コネクターは、進行中キュー内のメッセージをすべて処理した後で、入力キューからのメッセージの処理を開始します。

無視 (Ignore)

Ignore オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、これらのメッセージは無視されますが、コネクターはシャットダウンしません。

エラー・ログ記録 (Log Error)

Log Error オプションを使用すると、コネクターの初期化時に進行中キュー内のメッセージが検出された場合、エラー・ログが記録されますが、コネクターはシャットダウンしません。

アーカイブ

コネクター・プロパティー ArchiveQueue が指定されて有効なキューを示す場合、コネクターは正常に処理されたすべてのメッセージのコピーをアーカイブ・キューに入れます。ArchiveQueue が定義されていない場合は、メッセージは処理後に廃棄されます。

保証付きイベント・デリバリー

保証付きイベント・デリバリー機能により、コネクター・フレームワークは、イベントが逸失したり、コネクターのイベント・ストア、JMS イベント・ストア、および宛先の JMS キューの間でイベントが 2 度送信されたりするのを防ぐことができます。JMS を有効にするには、コネクターの DeliveryTransport 標準プロパティーを JMS に設定する必要があります。この構成により、コネクターは JMS トランスポートを使用することになり、コネクターと InterChange Server Express の間の後続の通信はすべて、このトランスポートを介して行われます。JMS トランスポートでは、最終的にメッセージが宛先に移送されることが保証されます。JMS トランスポートの役割は、トランザクションのキュー・セッションの開始後、コミットが発行されるまでメッセージが確実にキャッシュされるようにすることです。障害が起こったとき、あるいはロールバックが発行されたときには、メッセージは廃棄されます。

注: 保証付きイベント・デリバリー機能を使用しない場合、コネクターがイベントをパブリッシュする時間 (コネクターが gotApplEvent() メソッドを自身の pollForEvents() メソッド内部で呼び出す時間) と、コネクターがイベント・レコードを削除することによってイベント・ストアを更新する (または「イベント送付済み」状況を使用して更新する) 時間との間のわずかな期間に障害が発生する可能性があります。この期間に障害が発生すると、イベントは送信されますが、イベント・レコードは「進行中」状況でイベント・ストア内に残ります。コネクターは再始動時にイベント・ストア内に残ったイベント・レコードを検出して送信するため、結果的にイベントが 2 回送信されることになります。

保証付きイベント・デリバリー機能は、JMS イベント・ストアを持つ JMS 対応コネクター用、あるいは JMS イベント・ストアを持たない JMS 対応コネクター用に構成できます。保証付きイベント・デリバリー用にコネクターを構成する方法については、「*システム・インプリメンテーション・ガイド*」を参照してください。

コネクター・フレームワークがビジネス・オブジェクトを InterChange Server Express に送信できない場合、オブジェクトは UnsubscribedQueue や ErrorQueue で

はなく FaultQueue に置かれ、状況表示と問題の説明が生成されます。FaultQueue メッセージは MQRFH2 フォーマットで書き込まれます。

ビジネス・オブジェクト要求

InterChange Server Express がビジネス・オブジェクトを発行すると、ビジネス・オブジェクト要求が処理されます。

メッセージ処理

コネクターは、各ビジネス・オブジェクトの動詞に基づいた InterChange Server Express によって、渡されるビジネス・オブジェクトを処理します。サポートするビジネス・オブジェクトを処理するために、コネクターはビジネス・オブジェクト・ハンドラーを使用します。ビジネス・オブジェクト・ハンドラーには、アプリケーションと対話し、ビジネス・オブジェクト要求をアプリケーション操作に変換するメソッドがあります。

コネクターは以下のビジネス・オブジェクトの動詞をサポートします。

- Create
- Retrieve

Create

Create を含むビジネス・オブジェクトの処理は、オブジェクトが非同期または同期のどちらの方式で発行されているかによって異なります。

非同期デリバリー

Create の動詞を含むビジネス・オブジェクトに関しては、これがデフォルトのデリバリー・モードです。メッセージは、データ・ハンドラーを使用してビジネス・オブジェクトから作成され、出力キューに書き込まれます。メッセージが配信されると、コネクターは BON_SUCCESS、さもなければ BON_FAIL を戻します。

注: コネクターには、メッセージが受信されたかどうか、あるいはアクションが実行されたかどうかを確認する手段はありません。

同期確認通知

コネクター・プロパティーに replyToQueue が定義されていて、ビジネス・オブジェクトの変換プロパティーに responseTimeout が存在する場合、コネクターは同期モードで要求を発行します。コネクターはその後で応答を待機して、受信アプリケーションによって適切なアクションが実行されることを確認します。

WebSphere MQ の場合、コネクターは最初に、表 3 に示すヘッダーを持つメッセージを発行します。

表 3. 要求メッセージ記述子ヘッダー (MQMD)

フィールド	説明	値
Format	フォーマット名	変換プロパティーで定義された出力フォーマット。IBM 要件に合わせて 8 文字までに切り捨てられます (例: MQSTR)。
MessageType	メッセージ・タイプ	MQMT_DATAGRAM[a]

表 3. 要求メッセージ記述子ヘッダー (MQMD) (続き)

フィールド	説明	値
Report	必要なレポート・メッセージのオプション。	応答メッセージが期待される場合、このフィールドには次のように値が取り込まれます。 処理が成功した場合の肯定処理レポートが必要なことを示す、MQRO_PAN[a]。 処理が失敗した場合の否定処理レポートが必要なことを示す、MQRO_NAN[a]。 生成されるレポートの相関 ID が始めに発行された要求のメッセージ ID と等しくなければならないことを示す、MQRO_COPY_MSG_ID_TO_CORREL_ID[a]。
ReplyToQueue	応答キューの名前	応答メッセージが期待される場合は、このフィールドにはコネクター・プロパティー ReplyToQueue の値が取り込まれます。
Persistence	メッセージのパーシスタンス (永続性)	MQPER_PERSISTENT[a]
Expiry	メッセージの存続時間	MQEI_UNLIMITED[a]

[a] は、IBM によって定義されている定数を示します。

表 3 に示したメッセージ・ヘッダーの後ろに、メッセージ本文が続きます。メッセージ本文は、データ・ハンドラーを使用して直列化されたビジネス・オブジェクトです。

Report フィールドは、受信側アプリケーションからの肯定処理レポートと否定処理レポートの両方の返送が期待されていることを示すよう、設定されます。メッセージを発行したスレッドは、受信アプリケーションが要求を処理できるかどうかを示す応答メッセージを待機します。

アプリケーションは、コネクターから同期要求を受信すると、ビジネス・オブジェクトを処理して、表 4、表 5、および表 6 に示すレポート・メッセージを発行します。

表 4. 応答メッセージ記述子ヘッダー (MQMD)

フィールド	説明	値
Format	フォーマット名	変換プロパティーに定義された busObj の入力フォーマット。
MessageType	メッセージ・タイプ	MQMT_REPORT[a]

[a] は、IBM によって定義されている定数を示します。

表 5. 応答メッセージに含まれるデータ

動詞	フィードバック・フィールド	メッセージ本文
Create	SUCCESS VALCHANGE	(オプション) 変更を反映する直列化されたビジネス・オブジェクト
	VALDUPES FAIL	(オプション) エラー・メッセージ

表 6. フィードバック・コードおよび応答値

WebSphere MQ フィードバック・コード	同等な WebSphere Business Integration Server Express の応答[a]
MQFB_PANまたは MQFB_APPL_FIRST	SUCCESS

表 6. フィードバック・コードおよび応答値 (続き)

WebSphere MQ フィードバック・コード	同等な WebSphere Business Integration Server Express の応答[a]
MQFB_NANまたは MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	VALCHANGE
MQFB_APPL_FIRST + 3	VALDUPES
MQFB_APPL_FIRST + 4	MULTIPLE_HITS
MQFB_APPL_FIRST + 5	適用されない
MQFB_APPL_FIRST + 6	適用されない
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (コネクターが即時終了します)
MQFB_NONE	応答メッセージにフィードバック・コードが指定されていない場合にコネクターが受け取る情報

[a] 詳細については、「システム・インプリメンテーション・ガイド」を参照してください。

ビジネス・オブジェクトが処理できた場合、アプリケーションはフィードバック・フィールドを MQFB_PAN (または特定の WebSphere Business Integration Server Express システムの値) に設定して、レポート・メッセージを作成します。オプションで、アプリケーションはメッセージ本文に変更箇所を含むビジネス・オブジェクトを直列化して取り込むことができます。ビジネス・オブジェクトが処理されなかった場合、アプリケーションはフィードバック・フィールドを MQFB_NAN (または特定の WebSphere Business Integration Server Express システムの値) に設定して、レポート・メッセージを作成します。その後、オプションでメッセージ本文にエラー・メッセージを組み込みます。どちらの場合でも、アプリケーションはメッセージの correlationID フィールドをコネクター・メッセージの messageID に設定し、ReplyTo フィールドで指定されたキューにメッセージを発行します。

コネクターは、デフォルトでは、応答メッセージを検索する際に応答メッセージの correlationID と要求メッセージの messageID とを付き合わせます。次に、コネクターは要求を発行したスレッドを通知します。応答のフィードバック・フィールドによって、コネクターはメッセージ本文に、ビジネス・オブジェクトかエラー・メッセージのいずれかが含まれていることを、予測します。ビジネス・オブジェクトが要求され、メッセージ本文に値が含まれていない場合、コネクターは単に、Request 操作で最初に InterChange Server Express によって発行されたものと同じビジネス・オブジェクトを戻します。エラー・メッセージが要求され、メッセージ本文に値が含まれていない場合、汎用エラー・メッセージが応答コードとともに InterChange Server Express に戻されます。ただし、メッセージ・セレクターを使用して、要求に対する応答メッセージの識別、フィルター操作、およびアダプターでの識別方法を制御することもできます。このメッセージ・セレクターの機能は JMS の機能です。これは、同期要求処理のみに適用されます (下記参照)。

メッセージ・セレクターを使用した応答メッセージのフィルター操作: コネクターは、同期要求処理の対象となるビジネス・オブジェクトを受信するときに、動詞のアプリケーション固有情報に response_selector ストリングが存在するかどうかを検査します。response_selector が定義されていない場合は、コネクターは、上記の相関 ID を使用して応答メッセージを識別します。

response_selector が定義されている場合は、名前と値の組が以下の構文で格納されている必要があります。

```
response_selector=JMSCorrelationID LIKE 'selectorstring'
```

メッセージ・セレクター・ストリングは応答を一意的に識別しなければならず、値は以下のように単一引用符で囲まれていなければなりません。

```
response_selector=JMSCorrelationID LIKE 'Oshkosh'
```

上記の例では、要求メッセージを発行した後、アダプターは correlationID が「Oshkosh」に等しい応答メッセージがあるかどうか ReplyToQueue をモニターします。アダプターはこのメッセージ・セレクターに一致する最初のメッセージを検索し、応答としてディスパッチします。

オプションで、アダプターは実行時置換を行い、要求ごとに固有のメッセージ・セレクターを生成できるようにします。メッセージ・セレクターの代わりに、整数を中括弧で囲んだ形式 (例: '{1}') でプレースホルダーを指定します。この後にコロンを置き、置換に使用する属性をコンマで区切ってリストします。プレースホルダーの整数は、置換に使用する属性に対する指標として機能します。例えば、以下のメッセージ・セレクターを見てください。

```
response_selector=JMSCorrelationID LIKE '{1}': MyDynamicMO.CorrelationID
```

このメッセージ・セレクターでは、{1} をセレクターの最初の属性 (この例では子オブジェクト MyDynamicMO の属性 CorrelationId) の値で置換するようにアダプターに指示します。属性 CorrelationID の値が 123ABC の場合は、アダプターは以下の基準で作成されたメッセージ・セレクターを生成し、使用します。

```
JMSCorrelation LIKE '123ABC'
```

これにより、応答メッセージを識別します。

以下のように複数の置換を指定することもできます。

```
response_selector=PrimaryId LIKE '{1}' AND AddressId LIKE '{2}' :
PrimaryId, Address[4].AddressId
```

この例では、アダプターは {1} をトップレベルのビジネス・オブジェクトの属性 PrimaryId の値で置換し、{2} を子コンテナー・オブジェクト Address の 5 番目の AddressId の値で置換します。この方法では、応答メッセージ・セレクターでビジネス・オブジェクトおよびメタオブジェクトの任意の属性を参照できます。Address[4].AddressId を使用した深い検索の方法については、JCDK API のマニュアル (getAttribute メソッド) を参照してください。

以下の場合は、実行時にエラーが報告されます。

- {} 記号の間に整数以外の値を指定した場合
- 属性が定義されていない指標を指定した場合
- 指定された属性がビジネス・オブジェクトまたはメタオブジェクトに存在しない場合
- 属性パスの構文が誤っている場合

例えば、リテラル値「{」または「}」をメッセージ・セレクターに含める場合は、それぞれ「{{」または「{}」を使用してください。また、これらの文字を属性値に

入れることもできますが、この場合は最初の「{」は不要です。エスケープ文字の使用例を以下に示します。response_selector=JMSCorrelation LIKE '{1}' and CompanyName='A{{P': MyDynamicMO.CorrelationID

コネクターは、このメッセージ・セレクターを以下のように解決します。

```
JMSCorrelationID LIKE '123ABC' and CompanyName='A{P'
```

コネクターは、属性値で「{」、「}」、「:」、「;」などの特殊文字を検出した場合は、それらの文字を照会ストリングに直接挿入します。これにより、アプリケーション固有情報の区切り文字としても機能する特殊文字を照会ストリングに含めることができます。

属性値からリテラル・ストリング置換を抽出する方法を次の例に示します。

```
response_selector=JMSCorrelation LIKE '{1}' and CompanyName='A{{P':
MyDynamicMO.CorrelationID
```

MyDynamicMO.CorrelationID に値 {A:B}C;D が含まれている場合は、コネクターは、このメッセージ・セレクターを JMSCorrelationID LIKE '{A:B}C;D' and CompanyName='A{P' と解決します。

応答セレクター・コードの詳細については、JMS 1.0.1 仕様を参照してください。

カスタム・フィードバック・コードの作成: コネクター・プロパティー FeedbackCodeMappingMO を指定することにより、WebSphere MQ フィードバック・コードを拡張して表 6 に示されたデフォルトの解釈をオーバーライドすることができます。このプロパティーを使用すると、すべての WebSphere Business Integration Server Express システム固有の戻り状況値が WebSphere MQ のフィードバック・コードにマップされるメタオブジェクトを作成できます。(メタオブジェクトを使用して) フィードバック・コードに割り当てられた戻り状況は、InterChange Server Express に渡されます。詳細については、25 ページの『FeedbackCodeMappingMO』を参照してください。

Retrieve

Retrieve 動詞を持つビジネス・オブジェクトは、同期デリバリーのみをサポートします。コネクターはこの動詞を含むビジネス・オブジェクトを、Create 用に定義された同期デリバリーの場合と同様に処理します。ただし、Retrieve 動詞を使用している場合、responseTimeout と replyToQueue が必要になります。さらに、トランザクションを完了するには、メッセージ本文に直列化ビジネス・オブジェクトが取り込まれる必要があります。

表 7 に、これらの動詞に対応する応答メッセージを示します。

表 7. 応答メッセージに含まれるデータ

動詞	フィードバック・フィールド	メッセージ本文
Retrieve	FAIL	(オプション) エラー・メッセージ
	FAIL_RETRIEVE_BY_CONTENT MULTIPLE_HITS SUCCESS	直列化ビジネス・オブジェクト

エラー処理

コネクターが生成するエラー・メッセージはすべて、SWIFTConnector.txtという名前のメッセージ・ファイルに保管されます (ファイル名は、LogFileName 標準コネクター構成プロパティーによって決定されます)。各エラーには、エラー番号とそれに続くエラー・メッセージが含まれます。

Message number

Message text

コネクターは、以下の各セクションで説明するような特定のエラーを処理します。

アプリケーション・タイムアウト

以下の場合に、エラー・メッセージ「ABON_APPRESPONSETIMEOUT」が戻されます。

- メッセージの検索中、コネクターが JMS サービス・プロバイダーへの接続を確立できない場合。
- コネクターはビジネス・オブジェクトをメッセージに正常に変換したが、接続切断が原因でメッセージを出力キューにデリバリーできない場合。
- コネクターはメッセージを発行したが、変換プロパティー TimeoutFatal が True であるビジネス・オブジェクトからの応答の待機中にタイムアウトが発生した場合。
- コネクターが、APP_RESPONSE_TIMEOUT または UNABLE_TO_LOGIN に等しい戻りコードを持つ応答メッセージを受け取った場合。

アンサブスクライブされたビジネス・オブジェクト

コネクターは、以下の場合に UnsubscribedQueue プロパティーで指定されたキューにメッセージをデリバリーします。

- コネクターが、アンサブスクライブされたビジネス・オブジェクトに関連するメッセージを検索する場合。
- コネクターはメッセージを検索したが、Format フィールドのテキストをビジネス・オブジェクト名に関連付けることができない場合。

注: UnsubscribedQueue が定義されていない場合は、アンサブスクライブされたメッセージは廃棄されます。

データ・ハンドラーによる変換

データ・ハンドラーがメッセージをビジネス・オブジェクトに変換できなかった場合や (JMS プロバイダーではなく) ビジネス・オブジェクトに固有の処理エラーが発生した場合、メッセージは、ErrorQueue で指定されたキューに送信されます。ErrorQueue が定義されていない場合、エラーが原因で処理できないメッセージは廃棄されます。

データ・ハンドラーがビジネス・オブジェクトをメッセージに変換できない場合は、BON_FAIL が戻されます。

トレース

トレースはオプションのデバッグ機能であり、この機能をオンにするとコネクターの動作を密着して追跡できます。トレース・メッセージは、デフォルトでは STDOUT に書き込まれます。トレース・メッセージの構成の詳細については、19 ページの『第 2 章 コネクターのインストールと構成』に記載されている『コネクター構成プロパティー』を参照してください。トレースの有効化および設定の方法など、詳細については、「システム・インプリメンテーション・ガイド」を参照してください。

次に、コネクター・トレース・メッセージに推奨する内容を示します。

レベル 0　このレベルは、コネクターのバージョンを示すトレース・メッセージに使用されます。

レベル 1　このレベルは、処理された各ビジネス・オブジェクトについて主要な情報を提供するトレース・メッセージや、ポーリング・スレッドが入力キュー内で新規メッセージを検出したときにそれを記録するトレース・メッセージに使用されます。

レベル 2　このレベルは、ビジネス・オブジェクトが `gotApplEvent()` または `executeCollaboration()` のいずれかから InterChange Server Express に通知されるたびにログを記録するトレース・メッセージに使用されます。

レベル 3　このレベルは、メッセージからビジネス・オブジェクトへの変換およびビジネス・オブジェクトからメッセージへの変換に関する情報を提供するトレース・メッセージや、出力キューへのメッセージの送達に関する情報を提供するトレース・メッセージに使用されます。

レベル 4　このレベルは、コネクターがある関数を入力または出力する場合を示すトレース・メッセージに使用します。

レベル 5　このレベルは、コネクターの初期化を示すトレース・メッセージ、アプリケーション内で実行されるステートメントを示すトレース・メッセージ、メッセージがキューから取り出されたりキューに入れられたりしたときにそれを記録するトレース・メッセージ、ビジネス・オブジェクトのダンプを記録するトレース・メッセージなどに使用されます。

第 2 章 コネクターのインストールと構成

この章では、コネクターをインストールおよび構成する方法と、コネクターと連動するようにメッセージ・キューを構成する方法について説明します。

インストール作業の概要

Connector for SWIFT をインストールするには、次の作業を実行する必要があります。

アダプターの前提条件の確認

アダプターをインストールする前に、アダプターをインストールおよび実行するための環境の前提条件がご使用のシステムですべて満たされていることを確認してください。詳細については、2 ページの『アダプター環境』を参照してください。

統合ブローカーのインストール

統合ブローカーのインストール、すなわち、WebSphere Business Integration Server Express のインストールと始動を含むタスクは、「*WebSphere Business Integration Server Express インストールガイド (Windows 版)*」、「*WebSphere Business Integration Server Express インストール・ガイド (Linux 版)*」、または「*WebSphere Business Integration Server Express インストール・ガイド (OS/400 版)*」に説明されています。

Adapter for SWIFT と関連ファイルのインストール

アダプターのインストールについては、以下のサイトの WebSphere Business Integration Server Express InfoCenter に用意されている、ご使用のプラットフォームに対応した「*WebSphere Business Integration Server Express インストール・ガイド*」を参照してください。

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

インストール済みファイルの構造

以下のサブセクションでは、Windows、OS/400、および Linux の各システムにインストールされているファイル構造について説明します。

インストール済みの Windows ファイル

表 8 に、コネクターが使用する Windows ファイル構造の説明を示します。

注: このコネクターの Web リリースをインストールする場合は、リリース情報にあるインストール手順を参照してください。

表 8. コネクター用としてインストールされた Windows ファイル構造

ProductDir のサブディレクトリー	説明
connectors¥SWIFT¥CWSwift.jar	コネクター jar ファイル
connectors¥SWIFT¥start_SWIFT.bat	コネクターの始動ファイル
connectors¥messages¥SWIFTConnector.txt	コネクター・メッセージ・ファイル
repository¥SWIFT¥CN_SWIFT.txt	コネクター定義
DataHandlers¥CwDataHandler.jar	SWIFT データ・ハンドラー
repository¥DataHandlers¥MO_DataHandler_SWIFT.txt	SWIFT データ・ハンドラー用メタオブジェクト
repository¥DataHandlers¥MO_DataHandler_Default.txt	データ・ハンドラーのデフォルト・オブジェクト
connectors¥SWIFT¥samples¥Sample_SWIFT_MO_Config.txt	構成オブジェクトのサンプル
connectors¥SWIFT¥samples¥MO_SWIFT_MAPSUBSCRIPTIONS.txt	マッピング・メタオブジェクト
connectors¥SWIFT¥samples¥BO_Definitions¥SWIFT_objects.txt	ビジネス・オブジェクト定義
connectors¥SWIFT¥samples¥Map_Definitions¥Map_objects.txt	マップ定義

コネクター・コンポーネントのインストールの詳細については、「*WebSphere Business Integration Server Express Installation Guide for Windows*」を参照してください。

インストール済みの OS/400 ファイル

表 9 は、コネクターが使用する OS/400 ファイル構造を説明しています。

注: このコネクターの Web リリースをインストールする場合は、リリース情報にあるインストール手順を参照してください。

表 9. コネクター用としてインストールされた OS/400 ファイル構造

ProductDir のサブディレクトリー	説明
connectors/SWIFT/CWSwift.jar	コネクター jar ファイル
connectors/SWIFT/start_SWIFT.sh	コネクターの始動ファイル
connectors/messages/SWIFTConnector.txt	コネクター・メッセージ・ファイル
repository/SWIFT/CN_SWIFT.txt	コネクター定義
DataHandlers/CwDataHandler.jar	SWIFT データ・ハンドラー
repository/DataHandlers/MO_DataHandler_SWIFT.txt	SWIFT データ・ハンドラー用メタオブジェクト
repository/DataHandlers/MO_DataHandler_Default.txt	データ・ハンドラーのデフォルト・オブジェクト
connectors/SWIFT/samples/Sample_SWIFT_MO_Config.txt	構成オブジェクトのサンプル
connectors/SWIFT/samples/MO_SWIFT_MAPSUBSCRIPTIONS.txt	マッピング・メタオブジェクト

表 9. コネクター用としてインストールされた OS/400 ファイル構造 (続き)

ProductDir のサブディレクトリー	説明
connectors/SWIFT/samples/BO_Definitions/SWIFT_objects.txt	ビジネス・オブジェクト定義
connectors/SWIFT/samples/Map_Definitions/Map_objects.txt	マップ定義

コネクター・コンポーネントのインストールの詳細については、「*WebSphere Business Integration Server Express Installation Guide for OS/400*」を参照してください。

インストール済みの Linux ファイル

表 10 は、コネクターが使用する Linux ファイル構造を説明しています。

注: このコネクターの Web リリースをインストールする場合は、リリース情報にあるインストール手順を参照してください。

表 10. コネクター用としてインストールされた Linux ファイル構造

ProductDir のサブディレクトリー	説明
connectors/SWIFT/CWSwift.jar	コネクター jar ファイル
connectors/SWIFT/start_SWIFT.sh	コネクターの始動ファイル。このスクリプトは、汎用コネクター・マネージャー・スクリプトによって呼び出されます。System Manager の「Connector Configuration Express」画面の「インストール」をクリックすると、このコネクター・マネージャー・スクリプト用にカスタマイズされたラッパーがインストーラーによって作成されます。このカスタマイズ・ラッパーは、コネクターの始動と停止を行うためのみに使用してください。
connectors/messages/SWIFTConnector.txt	コネクター・メッセージ・ファイル
repository/SWIFT/CN_SWIFT.txt	コネクター定義
DataHandlers/CwDataHandler.jar	SWIFT データ・ハンドラー
repository/DataHandlers/MO_DataHandler_SWIFT.txt	SWIFT データ・ハンドラー用メタオブジェクト
repository/DataHandlers/MO_DataHandler_Default.txt	データ・ハンドラーのデフォルト・オブジェクト
connectors/SWIFT/samples/Sample_SWIFT_MO_Config.txt	構成オブジェクトのサンプル
connectors/SWIFT/samples/MO_SWIFT_MAPSUBSCRIPTIONS.txt	マッピング・メタオブジェクト
connectors/SWIFT/samples/BO_Definitions/SWIFT_objects.txt	ビジネス・オブジェクト定義
connectors/SWIFT/samples/Map_Definitions/Map_objects.txt	マップ定義

コネクター・コンポーネントのインストールの詳細については、「*WebSphere Business Integration Server Express Installation Guide for Linux*」を参照してください。

コネクターの構成

コネクターの構成プロパティーには、標準構成プロパティーとアダプター固有の構成プロパティーという 2 つのタイプがあります。アダプターを実行する前に、必ずこれらのプロパティーの値を設定してください。

Connector Configurator Express を使用して、コネクター・プロパティーを構成することができます。

- Connector Configurator Express の詳細および段階的な操作手順については、111 ページの『付録 B. Connector Configurator Express』を参照してください。
- 標準コネクター・プロパティーについては、『標準コネクター・プロパティー』および 97 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。
- コネクター固有プロパティーについては、『コネクター固有のプロパティー』を参照してください。

コネクターは、始動時に構成値を取得します。実行時のセッション中に、1 つ以上のコネクター・プロパティーの値を変更できます。AgentTraceLevel など一部のコネクター構成プロパティーへの変更は、即時に有効になります。その他のコネクター・プロパティーへの変更を有効にするには、変更後にコンポーネントまたはシステムを再始動する必要があります。あるプロパティーが動的 (即時に有効になる) か静的 (コネクター・コンポーネントまたはシステムを再始動する必要がある) かを判別するには、Connector Configurator Express の「コネクター・プロパティー」ウィンドウ内の「更新メソッド」列を参照してください。

標準コネクター・プロパティー

標準構成プロパティーにより、すべてのコネクターによって使用される情報が提供されます。標準構成プロパティーの資料については、97 ページの『付録 A. コネクターの標準構成プロパティー』を参照してください。

注: Connector Configurator Express で構成プロパティーを設定するときは、BrokerType プロパティーを InterChange Server Express に設定します。InterChange Server Express に関連するプロパティーが「Connector Configurator Express」ウィンドウに表示されます。

コネクター固有のプロパティー

コネクター固有の構成プロパティーは、コネクターが実行時に必要とする情報を提供します。また、コネクター固有のプロパティーを使用すれば、エージェントを再コーディングまたは再ビルドせずに、コネクター内の静的情報やロジックを変更できます。

注: WebSphere MQ が提供する値が誤っていたり、不明である可能性があるので、必ずこれらの値をチェックしてください。供給された値が不適切であれば、その値を明示的に指定してください。

表 11 に、Connector for SWIFT のコネクター固有の構成プロパティーを示します。プロパティーの説明については、以下の各セクションを参照してください。

表 11. コネクター固有のプロパティー

名前	指定可能な値	デフォルト値	必須
『ApplicationPassword』	*ログイン・パスワード*		いいえ
24 ページの『ApplicationUserID』	*ログイン・ユーザー ID*		いいえ
24 ページの『ArchiveQueue』	*正常に処理されたメッセージのコピーが送信されるキュー*	queue://CrossWorlds.QueueManager/MQCONN.ARCHIVE	いいえ
24 ページの『Channel』	*MQ サーバー・コネクター・チャネル*		はい
24 ページの『ConfigurationMetaObject』	*構成メタオブジェクトの名前*		はい
24 ページの『DataHandlerClassName』	*データ・ハンドラー・クラス名*	com.crossworlds.DataHandlers.swift.SwiftDataHandler	いいえ
24 ページの『DataHandlerConfigMO』	*データ・ハンドラー・メタオブジェクト*	MO_DataHandler_Default	はい
24 ページの『DataHandlerMimeType』	*ファイルの MIME タイプ*	swift	いいえ
24 ページの『DefaultVerb』	*コネクターによってサポートされる動詞*	Create	
25 ページの『ErrorQueue』	*未処理のメッセージのキュー*	queue://crossworlds.Queue.manager/MQCONN.ERROR	いいえ
25 ページの『FeedbackCodeMappingMO』	*フィードバック・コード・メタオブジェクト*		いいえ
25 ページの『HostName』	*WebSphere MQ サーバー*		いいえ
26 ページの『InDoubtEvents』	FailOnStartup Reprocess Ignore LogError	Reprocess	いいえ
26 ページの『InputQueue』	*ポーリング・キュー*	queue://CrossWorlds.QueueManager/MQCONN.IN	はい
27 ページの『InProgressQueue』	*進行中イベント・キュー*	queue://CrossWorlds.QueueManager/MQCONN.IN_PROGRESS	いいえ
27 ページの『PollQuantity』	*InputQueue プロパティーで指定された各キューから検索するメッセージの数*	1	いいえ
27 ページの『Port』	*WebSphere MQ リスナーのために確立するポート*		いいえ
27 ページの『ReplyToQueue』	*コネクターからの要求発行時に応答メッセージが配信されるキュー*	queue://CrossWorlds.QueueManager/MQCONN.REPLYTO	いいえ
27 ページの『UnsubscribedQueue』	*アンサブスクライブされたメッセージが送信されるキュー*	queue://CrossWorlds.QueueManager/MQCONN.UNSUBSCRIBE	いいえ
27 ページの『UseDefaults』	true または false	false	

ApplicationPassword

WebSphere MQ へのログイン時に ApplicationUserID とともに使用されるパスワード。

デフォルト = なし。

ApplicationPassword がブランクの場合または除去された場合、コネクターは、WebSphere MQ が提供するデフォルトのパスワードを使用します。

ApplicationUserID

WebSphere MQ へのログイン時に ApplicationPassword とともに使用されるユーザー ID。

デフォルト = なし。

ApplicationUserIDがブランクの場合または除去された場合、コネクターは、WebSphere MQ が提供するデフォルトのユーザー ID を使用します。

ArchiveQueue

正常に処理されたメッセージのコピーが送信されるキューです。

デフォルト = queue://crossworlds.Queue.manager/MQCONN.ARCHIVE

Channel

コネクターが WebSphere MQ と通信するために経由する MQ サーバー・コネクター・チャネルです。

デフォルト = なし。

Channel の値が空白のままか、あるいはこのプロパティーが除去された場合、コネクターは WebSphere MQ から提供されるデフォルト・サーバー・チャネルを使用します。

ConfigurationMetaObject

コネクターの構成情報を含む静的なメタオブジェクトの名前です。

デフォルト = なし。

DataHandlerClassName

ビジネス・オブジェクトとの間でのメッセージ変換に使用するデータ・ハンドラー・クラスです。

デフォルト = com.crossworlds.DataHandlers.swift.SwiftDataHandler

DataHandlerConfigMO

構成情報を提供するためにデータ・ハンドラーに渡されるメタオブジェクトです。

デフォルト = MO_DataHandler_Default

DataHandlerMimeType

使用すると、特定の MIME タイプに基づいたデータ・ハンドラーを要求できます。

デフォルト = swift

DefaultVerb

ポーリング中にデータ・ハンドラーが動詞を設定しなかった場合に、着信ビジネス・オブジェクト内に設定される動詞を指定します。

デフォルト = Create

ErrorQueue

処理されなかったメッセージが送信されるキューです。

デフォルト = queue://crossworlds.Queue.manager/MQCONN.ERROR

FeedbackCodeMappingMO

メッセージの受信を WebSphere InterChange Server Express に同期的に知らせるのに使用される、デフォルトのフィードバック・コードをオーバーライドして再割り当てすることができます。このプロパティーを使用すると、それぞれの属性名がフィードバック・コードを表すと解釈されるメタオブジェクトを指定できます。フィードバック・コードに対応する値は、InterChange Server Express に渡される戻り状況値です。デフォルトのフィードバック・コードのリストについては、8 ページの『同期確認通知』を参照してください。コネクターは、WebSphere MQ 固有のフィードバック・コードを表す、次の属性値を受け入れます。

- MQFB_APPL_FIRST
- MQFB_APPL_FIRST_OFFSET_*N*、*N* は整数 (MQFB_APPL_FIRST + *N*) の値として解釈される)

コネクターは、次の WebSphere Business Integration Server Express システム固有の状況コードを、メタオブジェクトの属性値として受け入れます。

- SUCCESS
- FAIL
- APP_RESPONSE_TIMEOUT
- MULTIPLE_HITS
- UNABLE_TO_LOGIN
- VALCHANGE
- VALDUPES

表 12 に、メタオブジェクトのサンプルを示します。

表 12. フィードバック・コードのメタオブジェクト属性の例

属性名	デフォルト値
MQFB_APPL_FIRST	SUCCESS
MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	UNABLE_TO_LOGIN

デフォルト = なし。

HostName

WebSphere MQ のホストであるサーバーの名前です。

デフォルト = なし。

HostName が空白のままか、あるいは除去された場合、コネクターは WebSphere MQ にホストを決定させます。

InDoubtEvents

コネクターの予期しないシャットダウンのために、処理が完了していない進行中イベントの処理方法を指定します。初期化中に進行中キューにイベントが見つかった場合に実行するアクションを、以下の 4 つから選択してください。

- FailOnStartup。 エラー・ログを記録して即時にシャットダウンします。
- Reprocess。 残りのイベントを先に処理してから、入力キューのメッセージを処理します。
- Ignore。 進行中キューのすべてのメッセージを無視します。
- LogError。 エラー・ログを記録しますが、シャットダウンはしません。

デフォルト = Reprocess。

InputQueue

コネクターが新規メッセージをポーリングするメッセージ・キューを指定します。SWIFTAlliance Gateway へのルーティング用に WebSphere MQ キューを設定する方法については、MQSA の資料を参照してください。

コネクターは、セミコロンで区切られた複数のキュー名を受け入れます。例えば、キュー MyQueueA、MyQueueB、および MyQueueC をポーリングするには、コネクター構成プロパティー InputQueue の値を MyQueueA;MyQueueB;MyQueueC とします。

コネクターは、ラウンドロビン方式でキューをポーリングして、各キューから最大 pollQuantity 個のメッセージを検索します。例えば、pollQuantity が 2 であり、MyQueueA に 2 件のメッセージがあり、MyQueueB に 1 件のメッセージがあり、MyQueueC に 5 件のメッセージがある場合は、コネクターは以下のようにメッセージを取得します。

pollQuanity が 2 に設定されていると、コネクターは、pollForEvents を呼び出すごとに各キューから最大で 2 つのメッセージを検索します。最初のサイクル (2 回のうちの 1 回目) では、コネクターは、MyQueueA、MyQueueB、および MyQueueC の各キューの 1 番目のメッセージを検索します。これで最初のポーリング巡回は完了です。コネクターは 2 回目のポーリング巡回 (2/2 回目) を開始し、MyQueueA および MyQueueC からメッセージを 1 つずつ検索します。MqQueueB は空になっているので、このキューはスキップされます。すべてのキューを 2 回ポーリングしたら、メソッド pollForEvents への呼び出しは完了します。このメッセージ検索の順序は以下のとおりです。

1. MyQueueA から 1 件のメッセージ
2. MyQueueB から 1 件のメッセージ
3. MyQueueC から 1 件のメッセージ
4. MyQueueA から 1 件のメッセージ
5. 空になったため、MyQueueB をスキップ
6. MyQueueC から 1 件のメッセージ

デフォルト = queue://crossworlds.Queue.manager/MQCONN.IN

InProgressQueue

処理中にメッセージを保持するメッセージ・キュー。System Manager を使用してコネクター固有のプロパティーからデフォルトの InProgressQueue 名を削除することによって、このキューなしで動作するようコネクターを構成できます。そのように設定すると、イベントの保留中にコネクターがシャットダウンされたときにイベント・デリバリーが影響を受ける場合があるという警告のプロンプトが、始動時に出されます。

デフォルト = queue://crossworlds.Queue.manager/MQCONN.IN_PROGRESS

PollQuantity

pollForEvents スキャン中に InputQueue プロパティーで指定された各キューから検索するメッセージの数。

デフォルト =1

Port

WebSphere MQ リスナーのために確立するポート。

デフォルト = なし。

Port の値が空白のままか、あるいはこのプロパティーが除去された場合、コネクターは WebSphere MQ に適切なポートを決定させます。

ReplyToQueue

コネクターからの要求発行時に応答メッセージが配信されるキューです。

デフォルト = queue://crossworlds.Queue.manager/MQCONN.REPLYTO

UnsubscribedQueue

サブスクライブされていないビジネス・オブジェクトについてのメッセージが送信されるキューです。

デフォルト = queue://crossworlds.Queue.manager/MQCONN.UNSUBSCRIBED

UseDefaults

Create 操作では、UseDefaults を true に設定した場合に、各 isRequired ビジネス・オブジェクト属性に対して有効な値またはデフォルト値が指定されているかどうかをコネクターがチェックします。値が指定されている場合、Create 操作は成功します。このパラメーターを false に設定すると、コネクターは有効値の有無だけをチェックし、有効値が指定されていない場合、Create 操作は失敗します。デフォルト値は false です。

キューの Uniform Resource Identifiers (URI)

URI はキューを一意的に識別します。キューの URI は、シーケンス queue:// で始まり、それに以下の項目が続きます。

- キューが存在しているキュー・マネージャーの名前
- スラッシュ (/)

- キューの名前
- 残りのキュー・プロパティーを設定する、名前と値のペアのリスト (オプション)

例えば、次の URI を指定すると、キュー・マネージャー crossworlds.queue.manager に存在するキュー IN に接続し、すべてのメッセージが優先順位 5 の SWIFT メッセージとして送信されます。

```
queue://crossworlds.Queue.manager
/MQCONN.IN?targetClient=1&priority=5
```

表 13 に、キュー URI のプロパティー名を示します。

表 13. キューの URI に対する SWIFT 固有のコネクター・プロパティー名

プロパティー名	説明	値
expiry	ミリ秒で表した、メッセージの存続時間	0 = 無制限。 正整数 = タイムアウト (ミリ秒)。
priority	メッセージの優先順位。	0-9。1 が最高の優先順位。値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクターが自分のデフォルト値を使用できることを意味します。
persistence	メッセージを永続メモリーに保存するかどうか。	1 = 非永続 2 = 永続 値 -1 は、このプロパティーがキューの構成によって決定されることを意味します。値 -2 は、コネクターが自分のデフォルト値を使用することを意味します。
CCSID[1] (29 ページ)	宛先の文字セット。	整数 - 有効値は、WebSphere MQ の基本資料にリストされています。
targetClient	受信アプリケーションが JMS 準拠であるかどうか。	1 = MQ (MQMD ヘッダーのみ) この値は、SWIFTAlliance を表す 1 に設定する必要があります。
encoding	数値フィールドの表示方法。	WebSphere MQ の基本資料に記載されている整数値。

表 13. キューの URI に対する SWIFT 固有のコネクター・プロパティー名 (続き)

プロパティー名	説明	値

注:

1. コネクターは、MQMessages 内のデータの文字セット (CCSID) やエンコード属性を制御しません。コネクターを適切に動作させるには、WebSphere MQ キューで ASCII 文字集合を使用し、MQSA で適切に構成する必要があります。データ変換は、データがメッセージ・バッファーから検索されるか、あるいはメッセージ・バッファーに送達されるときに行われるため、コネクターはデータ変換を、IBM WebSphere MQ にインプリメントされている JMS に依存します (IBM WebSphere MQ Java クライアント・ライブラリーの資料を参照してください)。したがって、これらの変換は、ネイティブの WebSphere MQ API がオプション `MQGMO_CONVERT` を使用して実行する変換と、双方向で等しくなければなりません。コネクターは、変換プロセスにおける差異または失敗を制御できません。これは、追加の変更 (MQSA によって課される変更など) なしに、任意の CCSID や WebSphere MQ によってサポートされるエンコードのメッセージ・データを検索できます。特定の CCSID またはエンコードのメッセージを送達するには、出力キューが完全修飾の URI で、`CCSID` と `encoding` の値を指定している必要があります。コネクターはこの情報を WebSphere MQ に渡し、WebSphere MQ は MQMessage を送達するためにデータをエンコードするときに、この情報を使用します (JMS API を介して)。CCSID およびエンコードがサポートされていない場合は、IBM の Web サイトから最新バージョンの IBM WebSphere MQ Java クライアント・ライブラリーをダウンロードすると、多くの場合、サポートされるようになります。MQSA 要件の詳細については、MQSA の資料を参照してください。CCSID およびエンコードに固有の問題がそれでも解決されない場合は、IBM ソフトウェア・サポートに連絡を取り、別の Java 仮想マシンを使用してコネクターを実行する可能性を検討してください。

メタオブジェクト属性構成

Connector for SWIFT は、2 種類のメタオブジェクトを認識および読み取ることができます。

- 静的なコネクター・メタオブジェクト
- 動的な子メタオブジェクト

動的な子メタオブジェクトの属性値は、静的なメタオブジェクトの属性値と重複し、それらをオーバーライドします。

静的メタオブジェクト

静的メタオブジェクトは、ビジネス・オブジェクトごとに定義された変換プロパティーのリストで構成されています。ビジネス・オブジェクトの変換プロパティーを定義するには、ストリング属性を作成し、構文 `busObj_verb` を使用してそれを命名します。例えば、動詞 Create を含む Customer オブジェクトの変換プロパティーを定義するには、`Swift_MT502_Create` という名前の属性を作成します。属性のアプリケーション固有テキストには、実際の変換プロパティーを指定します。

さらに、`Default` という名前の予約済み属性名を、メタオブジェクトに定義することもできます。この属性があると、そのプロパティーはすべてのビジネス・オブジェクトの変換プロパティーのデフォルト値として使用されます。

注: 静的メタオブジェクトが指定されないと、コネクターは、ポーリング時に所与のメッセージ・フォーマットを特定のビジネス・オブジェクト・タイプにマップできません。この場合、コネクターはビジネス・オブジェクトを指定せずに、メッセージ・テキストを構成済みのデータ・ハンドラーに渡します。デー

タ・ハンドラーがテキストのみに基づいたビジネス・オブジェクトを作成できない場合、コネクターはこのメッセージ・フォーマットが認識されていないことを表すエラーを報告します。

表 14 に、メタオブジェクトのプロパティーを示します。

表 14. 静的メタオブジェクト・プロパティー

プロパティー名	説明
CollaborationName	コラボレーション名は、ビジネス・オブジェクトと動詞の組み合わせに対する属性の、アプリケーション固有テキスト内で指定される必要があります。例えば、Create 動詞付きのビジネス・オブジェクト Customer の同期要求を処理するようにしたい場合は、静的メタデータ・オブジェクト内に Swift_MT*nnn_Verb* という名前の属性を設定します。ここで、*nnn* は SWIFT メッセージ・タイプを表します (例、Swift_MT502_Create)。Swift_MT502_Create 属性には、名前と値のペアを含む、アプリケーション固有テキストが入っていなければなりません。例えば、CollaborationName=MyCustomerProcessingCollab です。構文の詳細については、 31 ページの『アプリケーション固有の情報』 の節を参照してください。この条件が満たされていない場合は、コネクターが Customer ビジネス・オブジェクトに関する要求を同期処理しようとすると、ランタイム・エラーが発生します。 **注:** このプロパティーは同期要求にのみ使用可能です。
DoNotReportBusObj	オプションで、DoNotReportBusObj プロパティーを含めることができます。このプロパティーを true に設定すると、発行されるすべての PAN レポート・メッセージのメッセージ本文がブランクになります。このプロパティーは、要求が正常処理されたことは確認したいが、ビジネス・オブジェクト変更の通知は必要ない場合に使用することをお勧めします。このプロパティーは、NAN レポートには影響しません。静的メタオブジェクトにこのプロパティーがない場合、コネクターはデフォルトの false をとり、ビジネス・オブジェクトをメッセージ・レポートに取り込みます。 **注:** このプロパティーは同期要求にのみ使用可能です。
InputFormat	入力フォーマットは、特定のビジネス・オブジェクトと関連付けるメッセージ・フォーマットです。検索されたメッセージがこのフォーマットである場合、メッセージは可能であれば特定のビジネス・オブジェクトに変換されます。ビジネス・オブジェクトにこのフォーマットが指定されていない場合、コネクターは特定のビジネス・オブジェクトのサブスクリプション・デリバリーを処理しません。 静的メタオブジェクト内の InputQueue プロパティーは、アダプターでメッセージが特定のビジネス・オブジェクトにマップされる際に、InputFormat プロパティーとともに基準の役割を果たします。この機能は、Adapter for SWIFT Protocol では使用されません。

表 14. 静的メタオブジェクト・プロパティー (続き)

プロパティー名	説明
OutputFormat	出力フォーマットは、指定されたビジネス・オブジェクトから作成されたメッセージで設定されます。OutputFormat プロパティーの値が指定されていない場合、使用可能であれば入力フォーマットが使用されます。動的な子メタオブジェクトに定義された OutputFormat プロパティーは、静的なメタオブジェクトに定義された値をオーバーライドします。
InputQueue	コネクターが、新しいメッセージを検出するためにポーリングする入力キュー。コネクター固有のプロパティーとしての InputQueue プロパティーは、アダプターのポーリング先キューを定義します。これは、アダプターがポーリングするキューを決定するのに使用する唯一のプロパティーです。 静的メタオブジェクト内の InputQueue プロパティーは、アダプターでメッセージが特定のビジネス・オブジェクトにマップされる際に、InputFormat プロパティーとともに基準の役割を果たします。この機能は、Adapter for SWIFT Protocol では使用されません。
OutputQueue	出力キューは、特定のビジネス・オブジェクトから派生したメッセージが配信されるキューです。動的な子メタオブジェクトに定義された OutputQueue プロパティーは、静的なメタオブジェクトに定義された値をオーバーライドします。
ResponseTimeout	タイムアウトまでの応答の待ち時間 (ミリ秒)。このプロパティーが未定義、または負の値の場合、コネクターは応答を待たずに即時に SUCCESS を戻します。動的子メタオブジェクトに定義された ResponseTimeout プロパティーの値は、静的メタオブジェクトに定義された値をオーバーライドします。
TimeoutFatal	このプロパティーが定義されていて、値 true を含む場合、ResponseTimeout に指定された時間内に応答を受信しなければ、コネクターは APP_RESPONSE_TIMEOUT を戻します。応答メッセージを待機中のその他すべてのスレッドは、InterChange Server Express にすぐに APP_RESPONSE_TIMEOUT を戻します。これにより、InterChange Server Express はコネクターへの接続を終了します。動的な子メタオブジェクトに定義された TimeoutFatal プロパティーは、静的なメタオブジェクトに定義された値をオーバーライドします。

注: コネクター固有のプロパティーとしての InputQueue プロパティーは、アダプターのポーリング先キューを定義します。これは、アダプターがポーリングするキューを決定するのに使用する唯一のプロパティーです。静的メタオブジェクト内の InputQueue プロパティーは、アダプターでメッセージが特定のビジネス・オブジェクトにマップされる際に、InputFormat プロパティーとともに基準の役割を果たします。 Adapter for SWIFT に対しては、この機能を使用しないでください。

アプリケーション固有の情報

アプリケーション固有の情報は、名前と値のペアで構成され、それらはセミコロンで区切られています。例えば、次のようになります。

```
InputFormat=ORDER_IN;OutputFormat=ORDER_OUT
```

アプリケーション固有の情報を使用すれば、データ・ハンドラーを入力キューにマップできます。

InputQueue へのデータ・ハンドラーのマッピング

静的メタオブジェクトのアプリケーション固有情報で InputQueue プロパティーを使用することにより、データ・ハンドラーと入力キューを関連付けることができます。この機能は、異なる書式や変換要件を持つ複数の取引先と取り引きする場合に役立ちます。それには、以下の作業を行う必要があります。

1. コネクター固有プロパティー (26 ページの『InputQueue』を参照) を使用して、1 つ以上の入力キューを構成する。
2. それぞれの入力キューごとに、キュー・マネージャーおよび入力キュー名を指定し、またアプリケーション固有情報にデータ・ハンドラーのクラス名および MIME タイプを指定する。

例えば、次に示す静的メタオブジェクトの属性は、データ・ハンドラーと、CompReceipts という名前の InputQueue を関連付けています。

```
[Attribute]
Name = Swift_MT502_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputQueue=//queue.manager/CompReceipts;
  DataHandlerClassName=com.crossworlds.
DataHandlers.swift.disposition_notification;
  DataHandlerMimeType=message/
disposition_notification
IsRequiredServerBound = false
[End]
```

入力フォーマットの多重定義

コネクターは通常、メッセージ検索時に入力フォーマットを特定のビジネス・オブジェクトと動詞の組み合わせと付き合わせます。次に、コネクターはそのビジネス・オブジェクト名とメッセージの内容をデータ・ハンドラーに渡します。これにより、データ・ハンドラーは、メッセージの内容がユーザーの要求するビジネス・オブジェクトと対応しているかどうかを確認できます。

ただし、2 つ以上のビジネス・オブジェクトに同一の入力フォーマットが定義されている場合は、コネクターはデータ・ハンドラーにデータを渡す前にそのデータが表すビジネス・オブジェクトを判別することはできません。このような場合、コネクターはメッセージ内容のみをデータ・ハンドラーに渡してから、生成されるビジネス・オブジェクトに基づいた変換プロパティーを検索します。したがって、データ・ハンドラーはメッセージ内容のみに基づいてビジネス・オブジェクトを判別する必要があります。

生成されるビジネス・オブジェクトの動詞が設定されていない場合、コネクターはなんらかの動詞を含む同じビジネス・オブジェクトに定義されている変換プロパティーを検索します。変換プロパティーのセットが 1 つだけ検出された場合、コネク

ターは特定の動詞 を割り当てます。複数の変換プロパティーが検出された場合、コネクターは動詞を区別できないため、メッセージは失敗します。

静的メタオブジェクトのサンプル

以下に示す静的なメタオブジェクトは、Create および Retrieve の動詞を使用して SWIFT_MT502 ビジネス・オブジェクトを変換するようにコネクターを構成します。属性 Default はメタオブジェクトで定義されます。コネクターは以下の属性を持つ変換プロパティーを使用します。

```
OutputQueue=CustomerQueue1;ResponseTimeout=5000;
TimeoutFatal=true
```

この属性は、その他すべての変換プロパティーのデフォルト値として使用されます。したがって、ある属性によって別の指定をされたり動的な子メタオブジェクト値によってオーバーライドされる場合を除いて、コネクターはすべてのビジネス・オブジェクトをキュー CustomerQueue1 に発行し、その後応答メッセージを待機します。5000 ミリ秒内に応答が到着しない場合、コネクター はすぐに終了します。

動詞 Create を含むビジネス・オブジェクト: 属性 Swift_MT502_Create は、フォーマット NEW のメッセージはすべて、動詞 Create を含むビジネス・オブジェクトに変換する必要があることをコネクターに示します。出力フォーマットは定義されていないため、コネクターは入力用に定義されたフォーマット (この場合は NEW) を使用して、このオブジェクトと動詞の組み合わせを表すメッセージを送信します。

動詞 Retrieve を含むビジネス・オブジェクト: 属性 Swift_MT502_Retrieve は、動詞 Retrieve を含むビジネス・オブジェクトが、フォーマット RETRIEVE を持つメッセージとして送信される必要があることを示します。デフォルトの応答時間は、コネクターがタイムアウトまでに最大 10000 ミリ秒待機できるようにオーバーライドされているので注意してください (応答が受信されない場合も終了します)。

```
[ReposCopy]
Version = 3.1.0
Repositories = 1cHyILNuPTc=
[End]
[BusinessObjectDefinition]
Name = Sample_MO
Version = 1.0.0

[Attribute]
Name = Default
Type = String
Cardinality = 1
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = OutputQueue=CustomerQueue1;ResponseTimeout=5000;TimeoutFatal=true
IsRequiredServerBound = false
[End]
[Attribute]
Name = Swift_MT502_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputFormat=NEW
IsRequiredServerBound = false
```

```
[End]
[Attribute]
Name = Swift_MT502_Retrieve
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = OutputFormat=RETRIEVE;ResponseTimeout=10000
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]

[End]
```

動的な子メタオブジェクト

静的なメタオブジェクトに必要なメタデータを指定することが困難または実行不可能な場合、コネクターは、ビジネス・オブジェクト・インスタンスごとに実行時に指定されたメタデータをオプションで受け入れることができます。

コネクターは、コネクターに渡されるトップレベル・ビジネス・オブジェクトに子として追加される動的なメタオブジェクトから、変換プロパティーを認識し、読み取ります。この動的な子メタオブジェクトの属性値は、コネクターの構成に使用される静的なメタオブジェクトに指定可能であった変換プロパティーと重複します。

動的な子メタオブジェクトのプロパティーは静的なメタオブジェクトから検出されるプロパティーをオーバーライドするため、動的な子メタオブジェクトを指定する場合は、静的なメタオブジェクトを指定するコネクター・プロパティーを組み込む必要はありません。つまり、動的子メタオブジェクトと静的メタオブジェクトのいずれを使用することも、両方を使用することもできます。

表 15 に、ビジネス・オブジェクト Swift_MT502_Create の静的メタオブジェクト・プロパティーの例を示します。アプリケーション固有のテキストは、セミコロンで区切られた名前と値のペアで構成されます。

表 15. Swift_MT502_Create の静的メタオブジェクト構造

属性名	アプリケーション固有のテキスト
Swift_MT502_Create	InputFormat=ORDER_IN; OutputFormat=ORDER_OUT; OutputQueue=QueueA; ResponseTimeout=10000; TimeoutFatal=False

表 16 に、ビジネス・オブジェクト Swift_MT_Create の動的子メタオブジェクトの例を示します。

表 16. Swift_MT502_Create の動的子メタオブジェクト構造

プロパティー名	値
OutputFormat	ORDER_OUT
OutputQueue	QueueA
ResponseTimeout	10000
TimeoutFatal	False

コネクターは、受信したトップレベルのビジネス・オブジェクトのアプリケーション固有テキストをチェックし、タグ cw_mo_conn に子メタオブジェクトが指定されているかを判別します。子メタオブジェクトが指定されている場合、動的な子メタオブジェクトの値が静的なメタオブジェクトに指定された値をオーバーライドします。

ポーリング中の動的な子メタオブジェクトの含まれるデータ

ポーリング中に検索されたメッセージについてさらに詳しい情報を InterChange Server Express に提供するため、コネクターは、作成されたビジネス・オブジェクトに動的なメタオブジェクトが定義済みである場合、その特定の属性に値を取り込みます。

表 17 に、動的な子メタオブジェクトがポーリング用に構造化される方法を示します。

表 17. ポーリング用の JMS 動的子メタオブジェクト構造

プロパティー名	サンプル値
InputFormat	ORDER_IN
InputQueue	MYInputQueue
OutputFormat	CxIgnore
OutputQueue	CxIgnore
ResponseTimeout	CxIgnore
TimeoutFatal	CxIgnore

表 17 に示すように、動的子メタオブジェクトで追加のプロパティー InputQueue を定義できます。このプロパティーには特定のメッセージが検索されるキューの名前が含まれます。子メタオブジェクト内にこのプロパティーが定義されていない場合、これらには値が取り込まれません。

シナリオ例:

- コネクターは、WebSphere MQ キューからフォーマット ORDER_IN でメッセージを取得します。
- コネクターは、このメッセージを注文ビジネス・オブジェクトに変換し、アプリケーション固有のテキストをチェックして、メタオブジェクトが定義されているかを判別します。
- メタオブジェクトが定義されている場合、コネクターはこのメタオブジェクトのインスタンスを作成し、InputQueue および InputFormat プロパティーを適宜取り込んで、そのビジネス・オブジェクトを有効なプロセスにパブリッシュします。

動的な子メタオブジェクトのサンプル

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

[Attribute]
Name = OutputFormat
Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
DefaultValue = ORDER
IsRequiredServerBound = false
[End]
[Attribute]
Name = OutputQueue
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = OUT
IsRequiredServerBound = false
[End]
[Attribute]
Name = ResponseTimeout
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = -1
IsRequiredServerBound = false
[End]
[Attribute]
Name = TimeoutFatal
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
DefaultValue = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = InputFormat
Type = String
MaxLength = 1
IsKey = true
```

```
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = InputQueue
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]


[Verb]
Name = Retrieve
[End]

[End]
[BusinessObjectDefinition]
Name = Swift_MT502
Version = 1.0.0
AppSpecificInfo = cw_mo_conn=MyConfig

[Attribute]
Name = FirstName
Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = LastName
Type = String
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Telephone
Type = String
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = MyConfig
```

```
Type = MO_Sample_Config
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]

[End]
```

JMS ヘッダー、SWIFT メッセージ・プロパティー、および動的子メタオブジェクト属性

動的メタオブジェクトに属性を追加すると、メッセージ・トランスポートの詳細情報を取得したりメッセージ・トランスポートを詳細に制御したりすることができます。このような属性を追加すると、JMS プロパティーを変更し、(アダプター・プロパティーで指定されたデフォルト ReplyToQueue を使用せずに) 要求ごとに ReplyToQueue を制御したり、メッセージの CorrelationID を再ターゲットしたりすることができます。このセクションでは、これらの属性、および同期モードと非同期モードの両方におけるイベント通知と要求処理に対する影響について説明します。

以下の属性は JMS および SWIFT ヘッダー・プロパティーを反映しており、動的メタオブジェクトで認識されます。

表 18. 動的メタオブジェクト・ヘッダー属性

ヘッダー属性名	モード	対応する JMS ヘッダー
CorrelationID	読み取り/書き込み	JMSCorrelationID
ReplyToQueue	読み取り/書き込み	JMSReplyTo
DeliveryMode	読み取り/書き込み	JMSDeliveryMode
Priority	読み取り/書き込み	JMSPriority
Destination	読み取り	JMSDestination
Expiration	読み取り	JMSExpiration
MessageID	読み取り	JMSMessageID
Redelivered	読み取り	JMSRedelivered
TimeStamp	読み取り	JMSTimeStamp

表 18. 動的メタオブジェクト・ヘッダー属性 (続き)

ヘッダー属性名	モード	対応する JMS ヘッダー
Type	読み取り	JMSType
UserID	読み取り	JMSXUserID
AppID	読み取り	JMSXAppID
DeliveryCount	読み取り	JMSXDeliveryCount
GroupID	読み取り	JMSXGroupID
GroupSeq	読み取り	JMSXGroupSeq
JMSProperties	読み取り/書き込み	

読み取り専用属性は、イベント通知中にメッセージ・ヘッダーから読み取られ、動的メタオブジェクトに書き込まれます。これらのプロパティーは、要求処理中に応答メッセージが発行されたときに動的メタオブジェクトも設定します。読み取り/書き込み属性は、要求処理中に作成されるメッセージ・ヘッダーで設定されます。イベント通知中は、読み取り/書き込み属性はメッセージ・ヘッダーから読み取られ、動的メタオブジェクトを設定します。

以下のセクションでは、これらの属性の解釈および使用について説明します。

注: 上記の属性はいずれも必須ではありません。ビジネス・プロセスに関連する動的メタオブジェクトには任意の属性を追加できます。

JMS プロパティー: 動的メタオブジェクトの他の属性と異なり、JMSProperties は単一カーディナリティー子オブジェクトを定義する必要があります。この子オブジェクトの各属性は、以下のように JMS メッセージ・ヘッダーの可変部分で読み取り/書き込みを行う単一プロパティーを定義する必要があります。

1. 属性の名前はセマンティック値を持ちません。
2. 属性のタイプは、JMS プロパティー・タイプに無関係に必ず String でなければなりません。
3. 属性のアプリケーション固有情報は、属性をマップする JMS メッセージ・プロパティーの名前と形式を定義する 2 つの名前と値の組を含まなければなりません。

以下の表に、JMSProperties オブジェクトの属性に対して定義する必要があるアプリケーション固有情報プロパティーを示します。

表 19. JMS プロパティー属性のアプリケーション固有情報

名前	指定可能な値	コメント
Name	任意の有効な JMS プロパティー名	これは JMS プロパティーの名前です。ベンダーによっては、拡張機能を提供するために特定のプロパティーを予約している場合があります。一般に、ユーザーはベンダー固有の機能にアクセスする場合以外は、JMS で開始するカスタム・プロパティーを定義してはなりません。
Type	String、Int、Boolean、Float、Double、Long、Short	これは JMS プロパティーのタイプです。JMS API は、JMS メッセージに値を設定するための多くのメソッドを提供します (例: setIntProperty、setLongProperty、setStringProperty)。ここで指定される JMS プロパティーのタイプによって、これらのどのメソッドを使用してメッセージのプロパティー値を設定するかが決まります。

以下の図に、動的メタオブジェクトの属性 JMSProperties および JMS メッセージ・ヘッダーの 4 つのプロパティー (ID、GID、RESPONSE、および RESPONSE_PERSIST) の定義を示します。属性のアプリケーション固有情報はそれぞれの名前およびタイプを定義します。例えば、属性 ID はタイプ String の JMS プロパティー ID にマップされます。

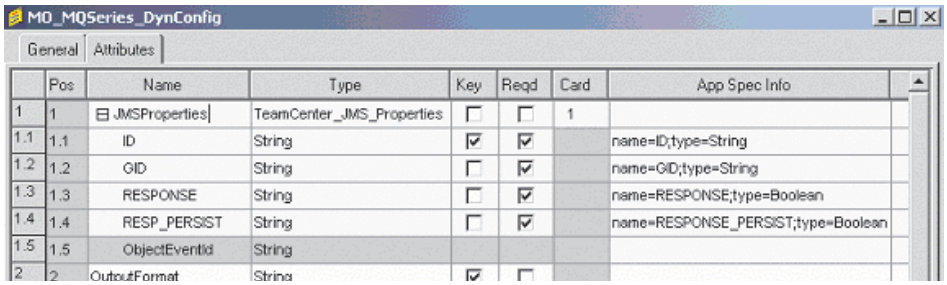


図 4. 動的メタオブジェクトの JMS プロパティー属性

非同期イベント通知: ヘッダー属性を持つ動的メタオブジェクトがイベント・ビジネス・オブジェクトに存在する場合は、コネクターは、(メタオブジェクトにトランスポート関連のデータを設定するほかに) 以下のステップを実行します。

1. メタオブジェクトの CorrelationId 属性に、メッセージの JMSCorrelationID ヘッダー・フィールドで指定された値を設定します。
2. メタオブジェクトの ReplyToQueue 属性に、メッセージの JMSReplyTo ヘッダー・フィールドで指定されたキューを設定します。このヘッダー・フィールド

はメッセージの Java オブジェクトによって表されるため、この属性にはキューの名前 (多くの場合は URI) が設定されます。

3. メタオブジェクトの DeliveryMode 属性に、メッセージの JMSDeliveryMode ヘッダー・フィールドで指定された値を設定します。
4. メタオブジェクトの Priority 属性に、メッセージの JMSPriority ヘッダー・フィールドを設定します。
5. メタオブジェクトの Destination 属性に、メッセージの JMSDestination ヘッダー・フィールドの名前を設定します。Destination はオブジェクトによって表されるため、この属性には Destination オブジェクトの名前が設定されます。
6. メタオブジェクトの Expiration 属性に、メッセージの JMSExpiration ヘッダー・フィールドの値を設定します。
7. メタオブジェクトの MessageID 属性に、メッセージの JMSMessageID ヘッダー・フィールドの値を設定します。
8. メタオブジェクトの Redelivered 属性に、メッセージの JMSRedelivered ヘッダー・フィールドの値を設定します。
9. メタオブジェクトの TimeStamp 属性に、メッセージの JMSTimeStamp ヘッダー・フィールドの値を設定します。
10. メタオブジェクトの Type 属性に、メッセージの JMSType ヘッダー・フィールドの値を設定します。
11. メタオブジェクトの UserID 属性に、メッセージの JMSXUserID プロパティー・フィールドの値を設定します。
12. メタオブジェクトの AppID 属性に、メッセージの JMSXAppID プロパティー・フィールドの値を設定します。
13. メタオブジェクトの DeliveryCount 属性に、メッセージの JMSXDeliveryCount プロパティー・フィールドの値を設定します。
14. メタオブジェクトの GroupID 属性に、メッセージの JMSXGroupID プロパティー・フィールドの値を設定します。
15. メタオブジェクトの GroupSeq 属性に、メッセージの JMSXGroupSeq プロパティー・フィールドの値を設定します。
16. メタオブジェクトの JMSProperties 属性に定義されたオブジェクトを検証します。アダプターは、メッセージの対応するプロパティーの値をこのオブジェクトの各属性に設定します。特定のプロパティーがメッセージで定義されていない場合は、アダプターはその属性の値を CxBlank に設定します。

同期イベント通知: 同期イベント処理の場合は、アダプターはイベントを通知し、InterChange Server Express からの応答を待った後、アプリケーションに応答メッセージを送信します。ビジネス・データに対する変更は、戻される応答メッセージに反映されます。イベントを通知する前に、アダプターは、非同期イベント通知の場合と同様に動的メタオブジェクトを設定します。動的メタオブジェクトに設定される値は、以下のように応答発行ヘッダーに反映されます (動的メタオブジェクトの他の読み取り専用属性は無視されます)。

- **CorrelationID** 動的メタオブジェクトが属性 CorrelationId を含む場合は、発信元アプリケーションが必要とする値に設定する必要があります。アプリケーションは、CorrelationID を使用してコネクターから戻されたメッセージと元の要求

を突き合わせます。CorrelationID が予期しない値または無効値の場合は、問題が発生します。これは、この属性を使用する前にアプリケーションが相関する要求および応答メッセージを処理する方法を判別するのに役立ちます。同期要求で CorrelationID を設定するには 4 つの方法があります。

1. 値を変更しない。応答メッセージの CorrelationID は、要求メッセージの CorrelationID と同じになります。これは、WebSphere MQ オプション MQRO_PASS_CORREL_ID と同等です。
2. 値を CxIgnore に変更する。コネクターは、デフォルトで要求のメッセージ ID を応答の CorrelationID にコピーします。これは、WebSphere MQ オプション MQRO_COPY_MSG_ID_TO_CORREL_ID と同等です。
3. 値を CxBlank に変更する。コネクターは応答メッセージの CorrelationID を設定しません。
4. 値をカスタム値に変更する。これを行うには、応答を処理するアプリケーションでカスタム値を認識する必要があります。

メタオブジェクトで属性 CorrelationID を定義しない場合は、コネクターは自動的に CorrelationID を処理します。

- **ReplyToQueue** 属性 ReplyToQueue に別のキューを指定することによって動的メタオブジェクトを更新する場合は、コネクターは、応答メッセージを指定のキューに送信します。これはお勧めしません。コネクターに対して応答メッセージを別のキューに送信させると、応答メッセージで特定の応答キューを設定するアプリケーションはそのキューで応答を待つと想定されるため、通信に干渉する場合があります。
- **JMS プロパティー** 更新されたビジネス・オブジェクトがコネクターに戻されるときに動的メタオブジェクトの JMS プロパティー属性に設定された値が応答メッセージに設定されます。

非同期要求処理: コネクターは、動的メタオブジェクト (存在する場合) を使用して要求メッセージを設定してから発行します。コネクターは、以下のステップを実行してから要求メッセージを送信します。

1. 属性 CorrelationID が動的メタオブジェクトに存在する場合は、コネクターは、アウトバウンド要求メッセージの CorrelationID をこの値に設定します。
2. 属性 ReplyToQueue が動的メタオブジェクトで指定されている場合は、コネクターは、応答メッセージでこのキューを渡し、このキューで応答を待ちます。これにより、コネクター構成プロパティーで指定されている ReplyToQueue 値をオーバーライドできます。さらに負の ResponseTimeout (コネクターが応答を待たないことを示す) を指定した場合は、コネクターは実際には応答を待ちませんが、応答メッセージで ReplyToQueue が設定されます。
3. 属性 DeliveryMode を 2 に設定すると、メッセージは永続的に送信されます。DeliveryMode を 1 に設定すると、メッセージは永続的に送信されません。その他の値を設定すると、コネクターに障害が発生します。MO に DeliveryMode を指定しないと、JMS プロバイダーが永続設定を確立します。
4. 属性 Priority を指定すると、コネクターが発信要求に値を設定します。Priority 属性には 0 から 9 までの値を設定できます。その他の値を指定すると、コネクターが終了します。

5. 動的メタオブジェクトで属性 JMSProperties を指定した場合は、コネクターによって送信されるアウトバウンド・メッセージに、子動的メタオブジェクトで指定された対応する JMS プロパティーが設定されます。

注: 動的メタオブジェクトのヘッダー属性が定義されていない場合または CxIgnore を指定した場合は、コネクターはデフォルト設定に従います。

同期要求処理: コネクターは、動的メタオブジェクト (存在する場合) を使用して要求メッセージを設定してから発行します。動的メタオブジェクトがヘッダー属性を含む場合は、コネクターは、応答メッセージで検出された対応する新しい値をそのヘッダー属性に設定します。コネクターは、応答メッセージを受信した後、(メタオブジェクトにトランスポート関連のデータを設定するほかに) 以下のステップを実行します。

1. 属性 CorrelationID が動的メタオブジェクトに存在する場合は、アダプターは、応答メッセージで指定された JMSCorrelationID でこの属性を更新します。
2. 属性 ReplyToQueue が動的メタオブジェクトで定義されている場合は、アダプターは、応答メッセージで指定された JMSReplyTo の名前でこの属性を更新します。
3. 属性 DeliveryMode が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSDeliveryMode ヘッダー・フィールドの値でこの属性を更新します。
4. 属性 Priority が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSPriority ヘッダー・フィールドの値でこの属性を更新します。
5. 属性 Destination が動的メタオブジェクトで定義されている場合は、アダプターは、応答メッセージで指定された JMSDestination の名前でこの属性を更新します。
6. 属性 Expiration が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSExpiration ヘッダー・フィールドの値でこの属性を更新します。
7. 属性 MessageID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSMessageID ヘッダー・フィールドの値でこの属性を更新します。
8. 属性 Redelivered が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSRedelivered ヘッダー・フィールドの値でこの属性を更新します。
9. 属性 TimeStamp が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSTimeStamp ヘッダー・フィールドの値でこの属性を更新します。
10. 属性 Type が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSType ヘッダー・フィールドの値でこの属性を更新します。
11. 属性 UserID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXUserID ヘッダー・フィールドの値でこの属性を更新します。
12. 属性 AppID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXAppID プロパティー・フィールドの値でこの属性を更新します。

13. 属性 DeliveryCount が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXDeliveryCount ヘッダー・フィールドの値でこの属性を更新します。
14. 属性 GroupID が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXGroupID ヘッダー・フィールドの値でこの属性を更新します。
15. 属性 GroupSeq が動的メタオブジェクトに存在する場合は、アダプターは、メッセージの JMSXGroupSeq ヘッダー・フィールドの値でこの属性を更新します。
16. 属性 JMSProperties が動的メタオブジェクトで定義されている場合は、アダプターは、子オブジェクトで定義されているすべてのプロパティーを、応答メッセージで検出された値で更新します。子オブジェクトで定義されているプロパティーがメッセージに存在しない場合は、値は CxBlank に設定されます。

注: 動的メタオブジェクトを使用して要求メッセージで設定された CorrelationID を変更しても、アダプターが応答メッセージを識別する方法には影響しません。アダプターは、デフォルトですべての応答メッセージの CorrelationID がアダプターによって送信された要求のメッセージ ID に等しいことを要求します。

エラー処理: JMS プロパティーをメッセージから読み取れない場合、またはメッセージに書き込めない場合は、コネクターはエラーをログに記録し、要求またはイベントは失敗します。ユーザー指定の ReplyToQueue が存在しないかアクセスできない場合は、コネクターはエラーをログに記録し、要求は失敗します。CorrelationID が無効であるか設定できない場合は、コネクターはエラーをログに記録し、要求は失敗します。いずれの場合も、ログに記録されたメッセージはコネクターのメッセージ・ファイルからのものです。

始動ファイルの構成

Connector for SWIFT を始動する前に、始動ファイルを構成する必要があります。以降のセクションでは、Windows、Linux、および OS/400 の各システムに対してこの作業を行う方法について説明します。

Windows

Windows 2000 のコネクターの構成を完了するには、start_SWIFT.bat ファイルを修正する必要があります。

1. start_SWIFT.bat ファイルを開きます。
2. 「Set the directory containing your WebSphere MQ Java client libraries,」で始まるセクションまでスクロールし、WebSphere MQ Java クライアント・ライブラリーの場所を指定します。

Linux

Linux プラットフォーム用のコネクターの構成を完了するには、start_SWIFT.sh ファイルを変更する必要があります。

1. start_SWIFT.sh ファイルを開きます。

2. 「Set the directory containing your WebSphere MQ Java client libraries」で始まるセクションまでスクロールし、WebSphere MQ Java クライアント・ライブラリーの場所を指定します。

OS/400

OS/400 では、始動ファイルを構成する必要はありません。start_SWIFT.sh スクリプトをインストールされた状態のまま使用し、アダプターを始動することができます。

アダプターの複数インスタンスの実行

注: このアダプター (または WebSphere Business Integration Server Express または Express Plus に付属する任意のアダプター) の追加インスタンスを作成すると、作成されたインスタンスは、配置可能なアダプターの総数を制限するライセンス交付機能によって、それぞれ別個のアダプターとしてカウントされます。

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

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

新規ディレクトリーの作成

それぞれのコネクター・インスタンスごとにコネクター・ディレクトリーを作成する必要があります。

- Windows プラットフォームの場合、コネクター・ディレクトリーには、次の名前を付けるようにします。

  ```
  ProductDir¥connectors¥connectorInstance
  ```

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

  ```
  ProductDir¥repository¥connectorInstance
  ```

  InterChange Server Express サーバー名は、例えば、次のように、startup.bat のパラメーターとして指定することができます。

  ```
  start_Swift.bat connName serverName
  ```

- OS/400 プラットフォームの場合、コネクター・ディレクトリーには、次の名前を付けるようにします。

  ```
  /QIBM/UserData/WBIServer43/WebSphereICSName/connectors
  /connectorInstance
  ```

ここで、connectorInstance はコネクター・インスタンスを一意的に示し、*WebSphereICSName* は、コネクターがともに稼動する InterChange Server Express サーバー・インスタンスの名前となります。

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

```
/QIBM/UserData/WBIServer43/WebSphereICSName/repository/connectorInstance
```

ここで、*WebSphereICSName* は、コネクターがともに稼動する InterChange Server Express サーバー・インスタンスの名前となります。

InterChange Server Express サーバー名は、例えば、次のように、startup.sh のパラメーターとして指定することができます。

```
start_Swift.sh connName serverName
```

- Linux プラットフォームの場合、コネクター・ディレクトリーには、次の名前を付けるようにします。

  ```
  ProductDir/connectors/connectorInstance
  ```

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

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

  ```
  ProductDir/repository/connectorInstance
  ```

  InterChange Server Express サーバー名は、例えば、次のように、connector_manager のパラメーターとして指定することができます。

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

ビジネス・オブジェクト定義の作成

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

1. 初期コネクターに関連付けられているビジネス・オブジェクト定義を変更する必要がある場合は、適切なファイルをコピーし、Business Object Designer Express を使用してそれらのファイルをインポートします。初期コネクターの任意のファイルをコピーできます。変更を加えた場合は、名前を変更してください。
2. 初期コネクターのファイルは、該当するディレクトリーにおきます。
   - Windows の場合

     ```
     ProductDir¥repository¥initialconnectorInstance
     ```

     作成した追加ファイルは、ProductDir¥repository の適切な connectorInstance サブディレクトリー内に存在している必要があります。
   - OS/400 の場合

     ```
     /QIBM/UserData/WBIServer43/WebSphereICSName/repository
     /initialConnectorInstance
     ```

ここで、*WebSphereICSName* は、コネクターがともに稼動する InterChange Server Express サーバー・インスタンスの名前となります。

作成した追加ファイル
は、/QIBM/UserData/WBIServer43/*WebSphereICSName*/repository の適切な connectorInstance サブディレクトリー内に存在している必要があります。

- Linux の場合

  ProductDir/repository/initialconnectorInstance

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname

   (Linux の場合のみ) 始動スクリプト CONJAR を
   CONJAR=${CONDIR}/CW${CONNAME}.jar から
   CONJAR=${CONDIR}/CWSWIFT.jar に変更します。
2. この始動スクリプトを、 45 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. (Windows の場合のみ) 始動スクリプトのショートカットを作成します。
4. (Windows の場合のみ) 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。
5. (OS/400 場合のみ) 次の情報を使用して、コネクターに対するジョブ記述を作成します。

   ```
   CRTDUPOBJ OBJ(QWBISWIFT) FROMLIB(QWBISVR43) OBJTYPE(*JOBD)
   TOLIB(QWBISVR43) NEWOBJ(newSWIFTCname)
   ```

   ここで、*newSWIFTCname* は 新規の Swift コネクターのジョブ記述に使用される 10 文字の名前となります。
6. (OS/400 の場合のみ) WebSphere Business Integration Console に新規のコネクターを追加します。この Console については、Console に用意されているオンライン・ヘルプを参照してください。

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

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 20 に示すように、オペレーティング・システム・プラットフォームによって異なります。

表 20. コネクターの始動スクリプト

オペレーティング・システム	始動スクリプト
Windows	start_*connName*.bat
OS/400	start_*connName*.sh
Linux	始動スクリプトの実行前に、環境変数を設定しておく必要があります。以下のコマンドが、環境変数を設定し、始動スクリプト start_*connName*.sh を自動的に実行します。 connector_manager -start *connName* *WebSphereICSName* [-cConfigFile]

コマンド行の始動オプションなどのコネクターの始動方法の詳細については、「*システム管理ガイド*」を参照してください。

Windows での始動スクリプトの起動

Windows プラットフォームでは、コネクター始動スクリプトは、以下の方法で起動することができます。

- **「スタート」**メニューから次のようにします。
  - **「プログラム」>「IBM WebSphere Business Integration Express」>「アダプター」>「コネクター」>**「***ご使用のコネクター名 (your_connector_name)***」を選択します。

    デフォルトでは、プログラム名は「IBM WebSphere Business Integration Express」となっています。ただし、これはカスタマイズすることができます。あるいは、ご使用のコネクターへのデスクトップ・ショートカットを作成することもできます。
  - Windows システムでは、Windows サービスとして始動するようにコネクターを構成することができます。この場合、Windows システムがブートしたとき (自動サービスの場合)、または Windows サービス・ウィンドウを通じてサービスを始動したとき (手動サービスの場合) に、コネクターが始動します。
- コマンド行から次を実行します。

  start_*connName* *connName* *WebSphereICSName* [-*cConfigFile*]

  ここで、*connName* はコネクターの名前、および *WebSphereICSName* は InterChange Server Express インスタンスの名前となります。デフォルトでは、InterChange Server Express インスタンスの名前は Web SphereICS です。

OS/400 での始動スクリプトの起動

OS/400 プラットフォームでは、コネクター始動スクリプトは、以下の方法で起動することができます。

- WebSphere Business Integration Server Express Console がインストールされている Windows システムから次のようにします。

  「プログラム」>「IBM WebSphere Business Integration Console」>「コンソール」を選択します。その後、OS/400 システム名または IP アドレス、およびユーザー・プロファイルと *JOBCTL 特殊権限をもつパスワードを指定します。アダプターのリストから、*connName* アダプターを選択してから、**「アダプターを始動」**ボタンを選択します。

- OS/400 コマンド行から次のようにします。
  - バッチ・モード

    CL Command QSH を実行し、QSHELL 環境から、次を実行します。

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

    ここで、*connName* はコネクター名、*WebSphereICSName* は InterChange Server Express サーバー名 (デフォルトは、QWBIDFT)、*pathToConnNameStartScript* はコネクター始動スクリプトの絶対パス、および *jobDescriptionName* は QWBISVR43 ライブラリーで使用するジョブ記述の名前となります。

  - 対話モード

    CL Command QSH を実行し、QSHELL 環境から、次を実行します。

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

    ここで、*connName* はコネクターの名前、および *WebSphereICSName* は InterChange Server Express インスタンスの名前となります。

注: TCP/IP サーバーから始動するには、次のコマンドを使用します。

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

ここで、*connName* はコネクターの名前、*WebSphereICSName* は InterChange Server Express インスタンスの名前、*pathToConnNameStartScript* はコネクター始動スクリプトの絶対パス、および *jobDescriptionName* は、アダプターのジョブ記述の名前となります。

Linux での始動スクリプトの起動

Linux プラットフォームでは、コネクター始動スクリプトは、以下の方法で起動することができます。

- コマンド行で、次を使用します。

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

  ここで、*connName* はコネクターの名前、および *WebSphereICSName* は InterChange Server Express インスタンスの名前となります。

コネクターの停止

コネクターを停止する方法は、コネクターが始動された方法によって異なります。

Windows でのコネクターの停止

Windows プラットフォームでは、コネクターは、以下の方法で停止することができます。

- コネクターのウィンドウをアクティブにし、「q」を入力した後、Enter キーを押します。
- コネクターが Windows サービスとして始動された場合は、**コントロール パネル (「コントロール パネル」>「管理ツール」>「サービス」> 「CWConnectorWBISwiftAdapter」)** を使用して停止することができます。

OS/400 でのコネクターの停止

OS/400 プラットフォームでは、コネクターは、以下の方法で停止することができます。

- Console またはコマンド行から次のようにします。

  Console または QSHELL 環境で「submit_adapter.sh」スクリプトを使用してコネクターを始動した場合は、OS/400 コマンド・エントリーから、CL Command *WRKACTJOB SBS(QWBISVR43)* を使用して、Server Express 製品のジョブを表示します。リストをスクロールし、コネクターのジョブ記述と一致するジョブ名をもつジョブを検出します。例えば、Swift コネクターの場合、ジョブ名は QWBISWIFTC です。

  このジョブの オプション 4 を選択し、F4 を押して、ENDJOB コマンド用のプロンプトを表示します。その後、Option パラメーターに *IMMED を指定し、Enter キーを押します。
- QSHELL から start_connName.sh スクリプトを使用してコネクターを始動した場合は、QSHELL 環境で F3 を押して、QSHELL とコネクターを終了します。

Linux でのコネクターの停止

Linux システムでは、コネクターはバックグラウンドで実行されているため、別個のウィンドウを持っていません。ウィンドウを使用しないで、次のコマンドを実行して、コネクターを停止します。

```
connector_manager -stop serverName connName
```

ここで、*serverName* は WebSphere ICS および *connName* はコネクターの名前となります。

第 3 章 ビジネス・オブジェクト

Connector for SWIFT はメタデータ主導型です。WebSphere ビジネス・オブジェクトでは、メタデータとはアプリケーションのデータに関するデータのことです。このデータはビジネス・オブジェクト定義に格納されており、コネクターとアプリケーションとの対話に役立ちます。メタデータ主導型コネクターは、サポートする各ビジネス・オブジェクトを処理する際に、コネクター内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義にエンコードされたメタデータに基づいて処理を行います。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、属性プロパティーの設定、およびアプリケーション固有テキストの内容が含まれています。コネクターは、メタデータ主導型であるため、コネクターのコードを修正せずに、新規または変更済みのビジネス・オブジェクトを処理できます。ただし、コネクターの構成済みデータ・ハンドラーでは、サポートされるビジネス・オブジェクトの構造、オブジェクト・カーディナリティー、アプリケーション固有のテキストの形式、およびビジネス・オブジェクトのデータベース表記に関する前提事項が想定されます。したがって、SWIFT 向けビジネス・オブジェクトを作成または変更する場合は、コネクターがそれに従うように設計されている規則に準拠して変更を行う必要があります。そうしないと、コネクターは新規のまたは変更されたビジネス・オブジェクトを適切に処理できません。

この章では、コネクターによるビジネス・オブジェクトの処理方法と、コネクターの前提事項について説明します。この情報は、新規のビジネス・オブジェクトをインプリメントするためのガイドとして使用できます。

コネクター・ビジネス・オブジェクトの要件

コネクターのビジネス・オブジェクト要件は、SWIFT データ・ハンドラーによる SWIFT メッセージと WebSphere ビジネス・オブジェクトの相互変換の方法を反映しています。

以下のセクションでは、WebSphere ビジネス・オブジェクトの要件と SWIFT メッセージ構造について説明します。SWIFT データ・ハンドラーが WebSphere ビジネス・オブジェクトおよび SWIFT メッセージとどのように対話するかの詳細については、91 ページの『第 4 章 SWIFT データ・ハンドラー』を参照してください。

「ビジネス・オブジェクト開発ガイド」を再度参照してください。

ビジネス・オブジェクト階層

WebSphere のビジネス・オブジェクトには、フラットなものと階層構造のものがあります。**フラット**・ビジネス・オブジェクトの属性はいずれも**単純**であり、各属性によって単一の値 (例えば `String`、`Integer`、`Date` など) が表されます。

単純属性に加えて、**階層**ビジネス・オブジェクトには、1 つの子ビジネス・オブジェクト、子ビジネス・オブジェクトの配列、またはその両方の組み合わせを表す属性があります。そして、子ビジネス・オブジェクトも、それぞれ自身の子ビジネス・オブジェクトまたはビジネス・オブジェクトの配列を持つことができます。この関係は階層の下に向かって続きます。

重要: ビジネス・オブジェクトの配列には、タイプがビジネス・オブジェクトであるデータを入れることができます。`String` や `Integer` などのほかのタイプのデータを入れることはできません。

親ビジネス・オブジェクトと子ビジネス・オブジェクト間の関係は 2 種類あります。

- **単一カーディナリティー —** 親ビジネス・オブジェクト内の属性が単一の子ビジネス・オブジェクトを表す場合。属性のタイプは子ビジネス・オブジェクトのタイプと同じです。
- **複数カーディナリティー —** 親ビジネス・オブジェクト内の属性が子ビジネス・オブジェクトの配列を表す場合。属性は、子ビジネス・オブジェクトと同じタイプの配列になります。

WebSphere では、ビジネス・オブジェクトに言及する場合に以下の用語を使用します。

- **階層型**: トップレベル・ビジネス・オブジェクトとすべてのレベルの子ビジネス・オブジェクトを含めた完全なビジネス・オブジェクトを指します。
- **親**: 少なくとも 1 つの子ビジネス・オブジェクトを含むビジネス・オブジェクトを指します。トップレベル・ビジネス・オブジェクトも親です。
- **個別**: 互いに包含関係にあると考えられるビジネス・オブジェクトのいずれの子ビジネス・オブジェクトからも独立した単一のビジネス・オブジェクトを指します。
- **トップレベル**: 階層のトップレベルにあって、それ自体は親ビジネス・オブジェクトを持たない個別ビジネス・オブジェクトを指します。
- **ラッパー**: 子ビジネス・オブジェクトの処理に使用する情報を含むトップレベル・ビジネス・オブジェクトを指します。例えば、XML コネクターの場合、ラッパー・ビジネス・オブジェクトは、子データ・ビジネス・オブジェクトのフォーマットを決定し、その子への経路を指定する情報を格納している必要があります。

ビジネス・オブジェクト属性のプロパティー

ビジネス・オブジェクト・アーキテクチャーで、属性に適用するさまざまなプロパティーを定義します。このセクションでは、コネクターがこれらのプロパティーのいくつかを解釈する方法を解説します。これらのプロパティーの詳細については、*「ビジネス・オブジェクト開発ガイド」の第 2 章 『ビジネス・オブジェクトの属性および属性プロパティー』を参照*してください。

Name プロパティー

ビジネス・オブジェクトの内部にある各ビジネス・オブジェクト属性は、一意の名前を持つ必要があります。この名前はその属性が格納するデータを示すものにしてください。

アプリケーション固有ビジネス・オブジェクトの場合は、特定の命名要件についてコネクターまたはデータ・ハンドラーの資料を確認してください。

名前は、80 文字までの英数字および下線にすることができます。スペース、句読点、特殊文字を入れることはできません。

Type プロパティー

Type プロパティーは属性のデータ型を定義します。

- 単純属性の場合、サポートされている型は、Boolean、Integer、Float、Double、String、Date、および LongText です。
- 属性が子ビジネス・オブジェクトを表す場合は、子ビジネス・オブジェクト定義の名前をタイプとして指定します (例えば、Type = MT502A としてカーディナリティーに 1 を指定します)。
- 属性が子ビジネス・オブジェクトの配列を表している場合、その子ビジネス・オブジェクト定義の名前をタイプとして指定し、カーディナリティーには n を指定してください。

注: また、子ビジネス・オブジェクトを表すすべての属性は、ContainedObjectVersion プロパティー (子のバージョン番号を指定する)、および Relationship プロパティー (値 Containment を指定する) も持ちます。

Cardinality プロパティー

各単純属性のカーディナリティーは 1 です。子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す各ビジネス・オブジェクト属性のカーディナリティーはそれぞれ 1 または n です。

注: 必須属性に指定する場合は、カーディナリティー 1 は子ビジネス・オブジェクトが存在する必要があることを示し、カーディナリティー n は 0 個以上の任意の数の子ビジネス・オブジェクトのインスタンスを示します。

Key プロパティー

各ビジネス・オブジェクトの少なくとも 1 つの属性をそのオブジェクトのキーとして指定する必要があります。属性をキーとして定義するには、このプロパティーを true に設定します。

子ビジネス・オブジェクトを表す属性をキーとして指定する場合は、そのキーは子ビジネス・オブジェクトのキーを連結したものです。子ビジネス・オブジェクトの配列を表す属性をキーとして指定する場合は、そのキーは配列の位置 0 にある子ビジネス・オブジェクトのキーを連結したものです。

注: キー情報は、コラボレーション・マッピング処理では使用できません。

Foreign key プロパティー

Foreign Key プロパティーは、一般的には、アプリケーション固有のビジネス・オブジェクトにおいて、ある属性の値が別のビジネス・オブジェクトの基本キーを保持することを指定するために使用します。この属性は、これら 2 つのビジネス・オブジェクトをリンクする手段として機能します。別のビジネス・オブジェクトの基本キーを保持する属性のことを、**foreign key** と呼びます。Foreign Key プロパティーは、外部キーを表す各属性に対して `true` に設定してください。

Foreign Key プロパティーは、他の処理命令にも使用することができます。例えば、このプロパティーは、コネクターが実行する外部キー検索の種類を指定する目的でも使用できます。 この場合、Foreign Key を `true` に設定すれば、コネクターは、データベース内に該当のエンティティーが存在するかどうかを調べ、そのエンティティーのレコードが存在している場合に限って関係を作成します。

Required プロパティー

Required プロパティーは、属性に対する値の指定が必要であるかどうかを指定します。作成しているビジネス・オブジェクトの特定の属性に値が必要な場合は、その属性の Required プロパティーを `true` に設定します。

属性に対する Required プロパティーの強制については、「*Connector Reference: C++ Class Library*」および「*Connector Reference: Java Class Library*」の『`initAndValidateAttributes()`』のセクションを参照してください。

AppSpecificInfo

AppSpecificInfo プロパティーは、主にアプリケーション固有ビジネス・オブジェクトに指定する 255 文字以下の `String` です。

注: アプリケーション固有テキストは、コラボレーション・マッピング処理では使用できません。

Max length プロパティー

Max Length プロパティーには、`String` 型の属性に格納できるバイト数を設定します。この値を WebSphere システムが強制することはありませんが、特定のコネクターまたはデータ・ハンドラーがこの値を使用する場合があります。許可されている最小長および最大長については、ビジネス・オブジェクトを処理するコネクターまたはデータ・ハンドラーの資料を参照してください。

注: Max Length プロパティーは、固定幅のデータ・ハンドラーを使用する場合にきわめて重要です。属性長は、コラボレーション・マッピング処理では使用できません。

Default value プロパティー

Default Value プロパティーは、属性に対してデフォルト値を指定するために使用します。

アプリケーション固有のビジネス・オブジェクトに対してこのプロパティーが指定されている場合、UseDefaults コネクター構成プロパティーが `true` に設定されていれば、コネクターは、実行時に値が指定されていない属性に値を設定するために、ビジネス・オブジェクト定義に指定されたデフォルト値を使用できます。

Default Value プロパティーの使用方法については、「*Connector Reference: C++ Class Library*」および「*Connector Reference: Java Class Library*」の『initAndValidateAttributes()』のセクションを参照してください。

Comments プロパティー

Comments プロパティーによって、人間にとって読みやすいコメントを属性に指定できます。 AppSpecificInfo プロパティーがビジネス・オブジェクトの処理に使用されるのに対し、Comments プロパティーは、文書情報のみを備えています。

特殊な属性値

ビジネス・オブジェクトの単純属性には、特殊値 CxIgnore を格納することができます。コネクターは統合ブローカーからビジネス・オブジェクトを受け取ると、CxIgnore という値を持つすべての属性を無視します。まるでコネクターにはそれらの属性が見えないかのように処理されます。

値が不要な場合、コネクターはデフォルトでその属性の値を CxIgnore に設定します。

属性レベルのアプリケーション固有テキスト

注: コネクターは、ビジネス・オブジェクト・レベルのアプリケーション固有テキストを使用しません。

ビジネス・オブジェクト属性の場合、アプリケーション固有テキストのフォーマットは名前と値のパラメーターからなります。名前と値のパラメーターにはそれぞれパラメーター名とその値が含まれています。属性アプリケーション固有テキストのフォーマットは次のとおりです。

name=*value*[:*name_n*=*value_n*][...]

各パラメーター・セットは、次のパラメーター・セットとコロン (:) 区切り文字によって区切られます。

表 21 で、属性アプリケーション固有テキストの名前と値のパラメーターについて説明します。

表 21. 属性の AppSpecificText 内の名前と値のパラメーター

パラメーター	必須	説明
block	トップレベル・オブジェクトの場合のみ必須	SWIFT メッセージ内のブロックの数。0 から 5 までの値の範囲。SWIFT メッセージ・ブロックの詳細については、56 ページの『SWIFT メッセージ構造の概要』を参照してください。
parse	トップレベル・オブジェクトの属性の場合のみ必須	SWIFT メッセージ・ブロックを解析するかどうかと、その方法を説明します。値は、fixlen (固定長として解析)、delim (区切られたテキストとして解析)、field (ブロック 4 のみ)、no (解析せず単一のストリングとして処理) です。

表 21. 属性の *AppSpecificText* 内の名前と値のパラメーター (続き)

パラメーター	必須	説明
`tag`	タイプ Tag ビジネス・オブジェクトの属性の場合は必須。	フィールドのタグ番号。SWIFT メッセージ・タグの詳細については、 127 ページの『付録 C. SWIFT メッセージ構造』を参照してください。Sequence ビジネス・オブジェクトおよび Field ビジネス・オブジェクトの詳細については、70 ページの『ブロック 4 ビジネス・オブジェクトの構造』を参照してください。
`letter=`*a*	Tag ビジネス・オブジェクトを指す各属性の場合は必須	SWIFT メッセージ・フォーマットでタグに付加される 1 つ以上のサポートされた文字。例えば、`20A` または `[A|B|NULL]` (A または B またはヌル)。`NULL` は、文字が使用される可能性のないタグや、文字オプションがまったくないタグに対して指定する必要があります。例えば、タグ 59 です。
`content`	いいえ	SWIFT メッセージ・フォーマットの修飾子。例えば、SWIFT メッセージ MT502、tag20C では、修飾子は `SEME`。

SWIFT メッセージ構造の概要

SWIFT メッセージは、5 つのデータ・ブロックからなります。さらに、MQSA コンポーネントは、キュー管理に使用するブロックを 2 つ追加します。SWIFT メッセージの上位構造は次のとおりです。

MQSA UUID

SWIFT 1: 基本ヘッダー・ブロック

SWIFT 2: アプリケーション・ヘッダー・ブロック

SWIFT 3: ユーザー・ヘッダー・ブロック

SWIFT 4: テキスト・ブロック化

SWIFT 5: トレーラー

MQSA S ブロック

注: MQSA コンポーネントは、UUID (User Unique Message Identifier) および S ブロックを追加します。どちらも SWIFT データ・ハンドラーによって解析されません。S ブロックの構造は SWIFT ブロック 5 と同じですが、フィールド・タグが 3 つの `char` ストリングから構成される点だけが異なります。例えば、`{S:{COP:P}}`。

SWIFT メッセージ構造の詳細については、 127 ページの『付録 C. SWIFT メッセージ構造』、および「*All Things SWIFT: the SWIFT User Handbook*」を参照してください。

SWIFT 用ビジネス・オブジェクトの概要

図 5 に示されているように、SWIFT には 5 種類のビジネス・オブジェクトがあります。

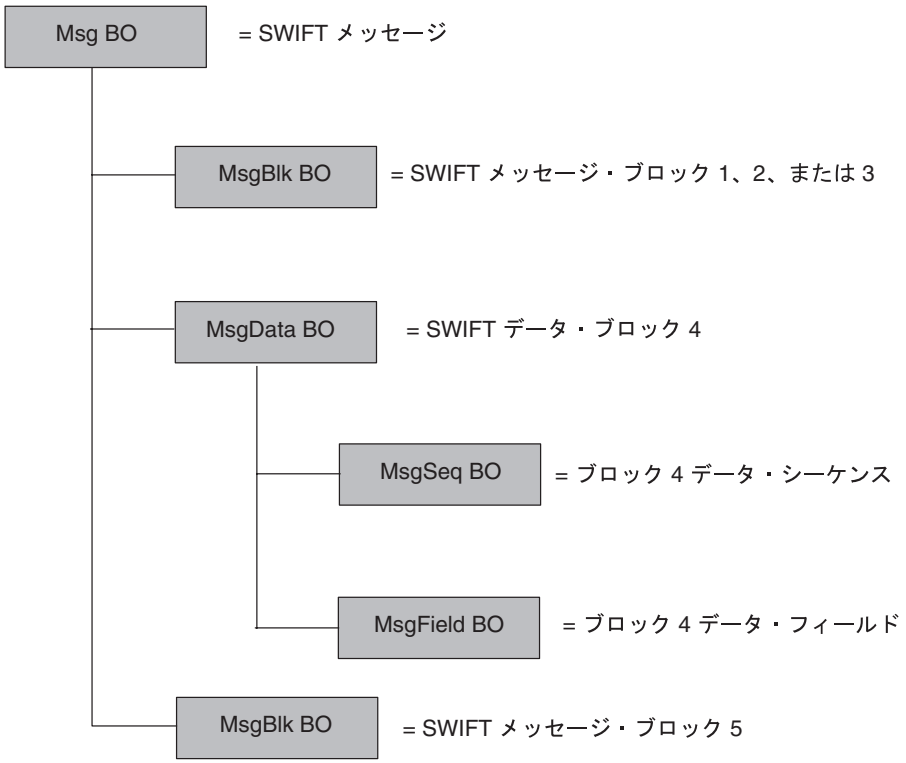


図 5. SWIFT メッセージ・コンポーネントへのビジネス・オブジェクト・マップ

- **Message ビジネス・オブジェクト (Msg BO)** これは、属性が SWIFT メッセージ内のブロックに対応しているトップレベルのビジネス・オブジェクトです。詳細については、58 ページの『トップレベル・ビジネス・オブジェクトの構造』を参照してください。
- **Message block ビジネス・オブジェクト (MsgBlk BO)** SWIFT メッセージ内でブロック 1、2、3、または 5 を表すことができる Msg BO の子オブジェクト。詳細については、62 ページの『ブロック 1 ビジネス・オブジェクトの構造』を参照してください。
- **Message data ビジネス・オブジェクト (MsgData BO)** SWIFT メッセージのブロック 4 を表す Msg BO の子オブジェクト。詳細については、70 ページの『ブロック 4 ビジネス・オブジェクトの構造』を参照してください。
- **Message sequence ビジネス・オブジェクト (MsgSeq BO)** MsgData BO または別の MsgSeq BO の子オブジェクト。MsgSeq BO は、SWIFT メッセージのブロック 4 で発生するフィールドのシーケンスを表します。詳細については、76 ページの『Sequence ビジネス・オブジェクトの構造』を参照してください。
- **Message field ビジネス・オブジェクト (MsgField BO)** フィールドの内容を含む MsgData BO または MsgSeq BO の子オブジェクト。フィールドは SWIFT メッセージ内のタグに対応しています。詳細については、79 ページの『Field ビジネス・オブジェクト定義』を参照してください。

これらのビジネス・オブジェクトはそれぞれ次のものから構成されます。

- **Name** ビジネス・オブジェクトの名前は、SWIFT メッセージ名、SWIFT メッセージ・シーケンス名、または SWIFT フィールド名から構成されます。より詳細

な命名規則がある場合は、以下にリストされた各種ビジネス・オブジェクトの各セクションでそれについて説明しています。例えば、次のようになります。

- Swift_MT502 は Msg BO の名前です。詳細については、58 ページの『トップレベル・ビジネス・オブジェクトの構造』を参照してください。
- Swift_ApplicationHeader は MsgBlk BO の名前です。詳細については、62 ページの『ブロック 1 ビジネス・オブジェクトの構造』、64 ページの『ブロック 2 ビジネス・オブジェクトの構造』、および68 ページの『ブロック 3 ビジネス・オブジェクトの構造』を参照してください。
- Swift_MT502Data は MsgData BO の名前です。詳細については、70 ページの『ブロック 4 ビジネス・オブジェクトの構造』を参照してください。
- Swift_MT502_B1 は MsgSeq BO の名前です。詳細については、76 ページの『Sequence ビジネス・オブジェクトの構造』を参照してください。
- Swift_Tag_22 は MsgField BO の名前です。詳細については、79 ページの『Field ビジネス・オブジェクト定義』を参照してください。

- **Version** ビジネス・オブジェクトのバージョンは 1.1.0 に設定されています。例えば、次のようになります。

  ```
  Version = 1.1.0
  ```

- **Attributes** 各ビジネス・オブジェクトには 1 つ以上の属性があります。詳細については、52 ページの『ビジネス・オブジェクト属性のプロパティー』および各種ビジネス・オブジェクトに関する以下の各セクションを参照してください。
- **Verbs** 各ビジネス・オブジェクトは、次の標準動詞をサポートしています。
  - Create
  - Retrieve

SWIFT メッセージとビジネス・オブジェクトのデータ・マッピング

IBM WebSphere Business Integration Server Express Adapter for SWIFT は、以下の種類のマッピングをサポートしています。

- **SWIFT メッセージから WebSphere ビジネス・オブジェクトへ** 以下の各セクションで、SWIFT メッセージと WebSphere ビジネス・オブジェクトとの間で行われるデータ・マッピングについて説明します。

トップレベル・ビジネス・オブジェクトの構造

SWIFT メッセージのトップレベル・ビジネス・オブジェクト、Msg BO の構造は、SWIFT メッセージの構造を反映します。WebSphere では、各 SWIFT ブロックにビジネス・オブジェクトが必要です。表 22 に示されているように、トップレベル・ビジネス・オブジェクトには、少なくとも各 SWIFT ブロックごとに 1 つずつ、合わせて 5 つの属性が必要です。

注: 表 22 には、結果の属性プロパティーだけが示されています。すべての属性プロパティーのリストは、60 ページの『トップレベル・ビジネス・オブジェクト (Msg BO) 定義のサンプル』を参照してください。

表 22. トップレベル・ビジネス・オブジェクトの構造

名前	タイプ	キー	必須	アプリケーション固有情報
UUID (MQSA が前に付加された)	String	はい	いいえ	block=0;parse=no
Swift_01Header	Swift_BasicHeader	なし	はい	block=1; parse=fixlen
Swift_02Header	Swift_Application Header	なし	いいえ	block=2; parse=fixlen
Swift_03Header	Swift_UserHeader	なし	いいえ	block=3; parse=delim
Swift_Data	Swift_Text	なし	いいえ	block=4;parse=field
Swift_05Trailer	String	なし	いいえ	block=5;parse=no
Swift_BlockS (MQSA が付加された)	String	なし	いいえ	block=6;parse=no

次の規則は、トップレベル・ビジネス・オブジェクトに適用されます。

- トップレベル・オブジェクトの名前は、次のように構成する必要があります。

  *BOPrefix*_MT*MessageType*

  ここで、以下のように説明されます。

  BOPrefix = メタオブジェクト (MO) の属性。メタオブジェクトの詳細については、29 ページの『静的メタオブジェクト』を参照してください。

  _MT = 定数ストリング。

  MessageType = SWIFT メッセージのブロック 2 の属性。詳細については、「*All Things SWIFT: the SWIFT User Handbook*」を参照してください。

  トップレベル・ビジネス・オブジェクト名は、例えば、Swift_MT502 のようになります。
- MQSA によってメッセージの前に付加される UUID は、String 属性で表されます。
- 1 から 4 までのブロックは、単一カーディナリティー・コンテナーで表されます。
- ブロック 5 は、ストリング属性であり、SWIFT データ・ハンドラーによってメッセージから抽出されることはありません。

注: ブロック 3 をテンプレートとして使用して、ブロック 5 およびブロック S のビジネス・オブジェクトを作成することもできます。

属性アプリケーション固有の情報については、表 21 を参照してください。

図 6 に、SWIFT メッセージのトップレベル・ビジネス・オブジェクトのビジネス・オブジェクト定義を示します。この Msg BO 定義は、WebSphere 開発環境で作成されました。

アプリケーション固有の情報には、各属性のブロック番号と構文解析パラメーターが含まれています。属性アプリケーション固有テキストの詳細については、表 21 を

参照してください。Swift_ 属性は、次の各セクションで説明する子ビジネス・オブジェクトに対応しています。このサンプル・ビジネス・オブジェクト定義の完全な仕様は、60 ページの『トップレベル・ビジネス・オブジェクト (Msg BO) 定義のサンプル』を参照してください。データ・ブロック属性 Swift_MT502Data のタイプに特に注意してください。これは、SWIFT メッセージ・タイプ 502 で、購買または販売のオーダーを示しています。この属性は、SWIFT メッセージのブロック 4 を表すトップレベル Msg BO の子オブジェクトに対応しています。子オブジェクトは、メッセージ・データ・ビジネス・オブジェクト (MsgData BO) です。

SWIFT トップレベル・ビジネス・オブジェクト定義はすべて、図 6 に示されている定義と同じですが、Swift_MT502Data として示されているブロック 4 が特定の SWIFT メッセージの実際のデータ定義を反映する点だけは異なります。

Name	Type	Key	Required	Application-Specifi...
UUID	String	Yes	No	block=0;parse=no
⊞ Swift_01Header	Swift_BasicHeader	No	Yes	block=1;parse=fixlen
⊞ Swift_02Header	Swift_ApplicationHeader	No	No	block=2;parse=fixlen
⊞ Swift_03Header	Swift_UserHeader	No	No	block=3;parse=delim
⊞ Swift_MT502Data	Swift_MT502Data	No	No	block=4;parse=field
Swift_05Trailer	String	No	No	block=5;parse=no
Swift_BlockS	String	No	No	block=6;parse=no
ObjectEventId	String	No	No	

図 6. SWIFT メッセージのトップレベル・ビジネス・オブジェクトの定義

注: SWIFT メッセージのトップレベル・ビジネス・オブジェクト定義を作成するには、Business Object Designer Express を起動し、まずすべての子オブジェクトを作成する必要があります。

トップレベル・ビジネス・オブジェクト (Msg BO) 定義のサンプル

このセクションでは、タイプ MT502 (購買または販売のオーダー) の SWIFT メッセージに対するトップレベル・ビジネス・オブジェクト (Msg BO) の定義のサンプルを示します。

```
[BusinessObjectDefinition]
Name = Swift_MT502
Version = 1.1.0

   [Attribute]
   Name = UUID
   Type = String
   Cardinality = 1
   MaxLength = 255
   IsKey = true
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = block=0;parse=no
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = Swift_01Header
   Type = Swift_BasicHeader
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 1
   IsKey = false
```

```
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = block=1;parse=fixlen
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = Swift_02Header
   Type = Swift_ApplicationHeader
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 1
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = block=2;parse=fixlen
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = Swift_03Header
   Type = Swift_UserHeader
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 1
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = block=3;parse=delim
   IsRequiredServerBound = false
   [End]
   [Attribute]
Name = Swift_MT502Data
Type = Swift_MT502Data
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 1
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = block=4;parse=field
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = Swift_05Trailer
   Type = String
   Cardinality = 1
   MaxLength = 1
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = block=5;parse=no
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = Swift_BlockS
   Type = String
   Cardinality = 1
   MaxLength = 1
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = block=6;parse=no
   IsRequiredServerBound = false
   [End]
   [Attribute]
```

```
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]
```

ブロック 1 ビジネス・オブジェクトの構造

MsgBlck BO、Swift_BasicHeader のフォーマットと属性が、表 23 に示されています。SWIFT データ・ハンドラーは、このブロック内の SWIFT フィールドをそれぞれ Swift_BasicHeader ビジネス・オブジェクト内の属性に変換します。このビジネス・オブジェクトには、属性アプリケーション固有の情報はありません。

注: 表 23 には、結果の属性プロパティーだけが示されています。すべての属性プロパティーのリストは、63 ページの『ブロック 1 ビジネス・オブジェクト定義のサンプル』を参照してください。

表 23. ブロック 1 ビジネス・オブジェクトの構造

名前	タイプ	キー	外部キー	必須	カーディナリティー	デフォルト	最大長
BlockIdentifier	String	はい	なし	はい	1	1:[a]	2
ApplicationIdentifier	String	なし	なし	はい	1		1
ServiceIdentifier	String	なし	なし	はい	1		2
LTIdentifier	String	なし	なし	はい	1		12
SessionNumber	String	なし	なし	はい	1		4
SequenceNumber	String	なし	なし	いいえ	1		4

[a] BlockIdentifier 属性には、「1:」のように、区切り文字「:」が含まれています。

属性アプリケーション固有の情報については、表 21 を参照してください。

図 7 に、WebSphere 開発環境で手動で作成されたブロック 1 ビジネス・オブジェクト定義を示します。各属性名 (ApplicationIdentifier、ServiceIdentifier など) は、この SWIFT メッセージ・ブロック内のフィールドに対応しています。この SWIFT メッセージ・ブロックの詳細については、127 ページの『付録 C. SWIFT メッセージ構造』および「*All Things SWIFT: the SWIFT User Handbook*」を参照してください。各名前付き属性に対して、型 String を指定してください。このビジネス・オブジェクトのコンポーネントには、属性アプリケーション固有の情報はありません。

注: この固定長ブロック・ビジネス定義では、属性名に必ず正しい MaxLength 値を指定してください。

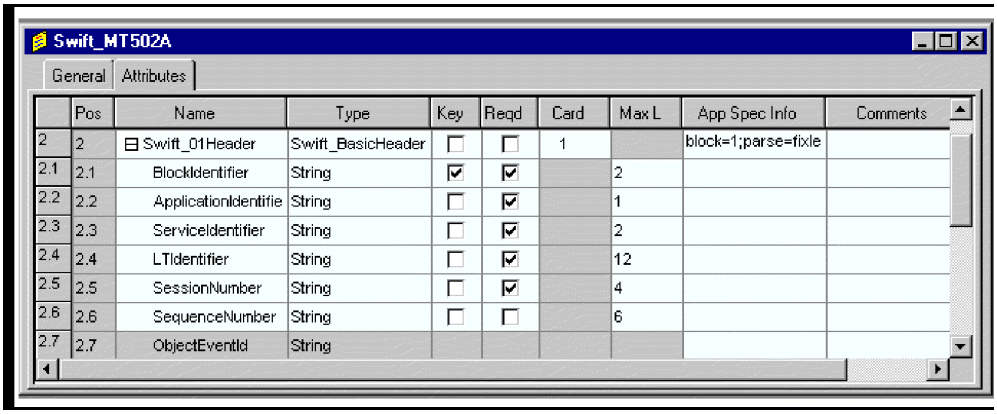


図 7. ブロック 1 ビジネス・オブジェクト定義

注: SWIFT メッセージに対してブロック 1 ビジネス・オブジェクト定義を作成するには、Business Object Designer Express を起動し、63 ページの『ブロック 1 ビジネス・オブジェクト定義のサンプル』に示された属性の値を入力します。

ブロック 1 ビジネス・オブジェクト定義のサンプル

このセクションでは、タイプ MT502 (購買または販売のオーダー) の SWIFT メッセージに対するブロック 1 ビジネス・オブジェクトの定義のサンプルを示します。

```
[BusinessObjectDefinition]
Name = Swift_BasicHeader
Version = 1.1.0

   [Attribute]
   Name = BlockIdentifier
   Type = String
   Cardinality = 1
   MaxLength = 2
   IsKey = true
   IsForeignKey = false
   IsRequired = true
   DefaultValue = 1:
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = ApplicationIdentifier
   Type = String
   Cardinality = 1
   MaxLength = 1
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = ServiceIdentifier
   Type = String
   Cardinality = 1
   MaxLength = 2
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = LTIdentifier
   Type = String
```

```
Cardinality = 1
MaxLength = 12
IsKey = false
IsForeignKey = false
IsRequired = true
IsRequiredServerBound = false
[End]
[Attribute]
Name = SessionNumber
Type = String
Cardinality = 1
MaxLength = 4
IsKey = false
IsForeignKey = false
IsRequired = true
IsRequiredServerBound = false
[End]
[Attribute]
Name = SequenceNumber
Type = String
Cardinality = 1
MaxLength = 6
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]
```

ブロック 2 ビジネス・オブジェクトの構造

ブロック 2 MsgBlk BO、Swift_ApplicationHeader のフォーマットと属性が 表 24 に示されています。SWIFT データ・ハンドラーは、このブロック内の SWIFT フィールドをそれぞれ Swift_ApplicationHeader ビジネス・オブジェクト内の属性に変換します。このビジネス・オブジェクトには、属性アプリケーション固有の情報はありません。

注: 表 24 には、結果の属性プロパティーだけが示されています。すべての属性プロパティーのリストは、66 ページの『ブロック 2 ビジネス・オブジェクト定義のサンプル』を参照してください。

表 24. ブロック 2 ビジネス・オブジェクトの構造

名前	タイプ	キー	必須	カーディナリティー	デフォルト	最大長
Block Identifier	String	なし	はい	1	2:[a]	2
IOIdentifier	String	なし	はい	1		1
MessageType	String	なし	はい	1		3
I_ReceiverAddress	String	なし	はい	1		12

表 24. ブロック 2 ビジネス・オブジェクトの構造 (続き)

名前	タイプ	キー	必須	カーディナリティー	デフォルト	最大長
I_MessagePriority	String	なし	はい	1		1
I_DeliveryMonitoring	String	なし	いいえ	1		1
I_ObsolescencePeriod	String	なし	いいえ	1		3
O_InputTime	String	なし	はい	1		4
O_MessageInputReference	String	なし	はい	1		28
O_OutputDate	String	なし	いいえ	1		6
O_OutputMessagePriority	String	なし	いいえ	1		6

[a] BlockIdentifier 属性には、「2」のように、区切り文字「:」が含まれています。

表 24 の最初の 3 つの属性は I/O 属性です。I_ で始まる属性は入力属性であり、SWIFT からビジネス・オブジェクトへの変換中にデータが取り込まれます。O_ で始まる属性は出力属性であり、ビジネス・オブジェクトから SWIFT への変換中にデータが取り込まれます。CxIgnore プロパティーは、ビジネス・オブジェクトから SWIFT への変換に対して設定する必要があります。

属性アプリケーション固有の情報については、表 21 を参照してください。

図 8 に、WebSphere 開発環境で手動で作成されたブロック 2 ビジネス・オブジェクト定義を示します。各属性名 (`BlockIdentifier`、`IOIdentifier` など) は、この SWIFT メッセージ・ブロック内のフィールドに対応しています。示されている定義は、入力属性 (I_) の定義であり、SWIFT からビジネス・オブジェクトへの変換中にデータが取り込まれます。この SWIFT メッセージ・ブロックの詳細については、127 ページの『付録 C. SWIFT メッセージ構造』および「*All Things SWIFT: the SWIFT User Handbook*」を参照してください。各名前付き属性に対して、型 `String` を指定してください。このビジネス・オブジェクトのコンポーネントには、属性アプリケーション固有の情報はありません。

注: この固定長ブロック・ビジネス定義では、属性名に必ず正しい MaxLength 値を指定してください。

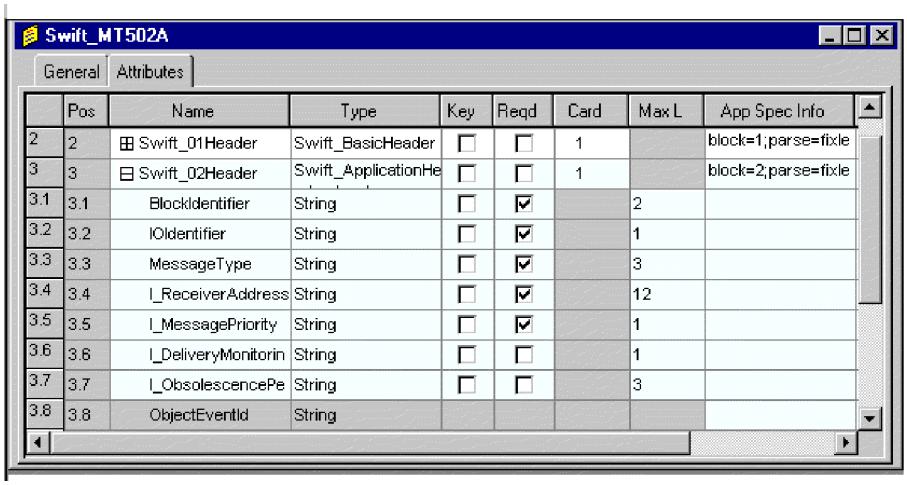


図 8. ブロック 2 ビジネス・オブジェクト定義

注: SWIFT メッセージに対してブロック 2 ビジネス・オブジェクト定義を作成するには、Business Object Designer Express を起動し、66 ページの『ブロック 2 ビジネス・オブジェクト定義のサンプル』に示された属性の値を入力します。

ブロック 2 ビジネス・オブジェクト定義のサンプル

このセクションでは、タイプ MT502 (購買または販売のオーダー) の SWIFT メッセージに対するブロック 2 ビジネス・オブジェクトの定義のサンプルを示します。

```
[BusinessObjectDefinition]
Name = Swift_ApplicationHeader
Version = 1.1.0

   [Attribute]
   Name = BlockIdentifier
   Type = String
   Cardinality = 1
   MaxLength = 2
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   DefaultValue = 2:
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = IOIdentifier
   Type = String
   Cardinality = 1
   MaxLength = 1
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   DefaultValue = O
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = MessageType
   Type = String
   Cardinality = 1
   MaxLength = 3
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = O_InputTime
   Type = String
   Cardinality = 1
   MaxLength = 4
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = O_MessageInputReference
   Type = String
   Cardinality = 1
   MaxLength = 28
   IsKey = false
   IsForeignKey = false
   IsRequired = true
   IsRequiredServerBound = false
   [End]
   [Attribute]
```

```
Name = O_OutputDate
Type = String
Cardinality = 1
MaxLength = 6
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = O_OutputTime
Type = String
Cardinality = 1
MaxLength = 4
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = O_OutputMessagePriority
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = I_ReceiverAddress
Type = String
Cardinality = 1
MaxLength = 12
IsKey = false
IsForeignKey = false
IsRequired = true
IsRequiredServerBound = false
[End]
[Attribute]
Name = I_MessagePriority
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = true
IsRequiredServerBound = false
[End]
[Attribute]
Name = I_DeliveryMonitoring
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = I_ObsolescencePeriod
Type = String
Cardinality = 1
MaxLength = 3
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]
```

ブロック 3 ビジネス・オブジェクトの構造

ブロック 3 MsgBlk BO、Swift_UserHeader のフォーマットと属性が 表 25 に示されています。このビジネス・オブジェクトには、属性アプリケーション固有の情報、Tag パラメーターがあります。Tag パラメーターについては、表 21 を参照してください。

注: 表 25 には、結果の属性プロパティーだけが示されています。すべての属性プロパティーのリストは、69 ページの『ブロック 3 ビジネス・オブジェクト定義のサンプル』を参照してください。

表 25. ブロック 3 ビジネス・オブジェクトの構造

名前	タイプ	キー	外部	必須	カーディナリティー	アプリケーション固有情報	最大長
Tag103	String	はい	なし	いいえ	1	Tag=103	6
Tag113	String	なし	なし	いいえ	1	Tag=113	6
Tag108	String	なし	なし	いいえ	1	Tag=108	6
Tag119	String	なし	なし	いいえ	1	Tag=119	6
Tag115	String	なし	なし	いいえ	1	Tag=115	6

図 9 に、WebSphere 開発環境で手動で作成されたブロック 3 ビジネス・オブジェクト定義を示します。各属性名 (Tag103、Tag113 など) は、この SWIFT メッセージ・ブロック内のフィールドに対応しています。この SWIFT メッセージ・ブロックの詳細については、 127 ページの『付録 C. SWIFT メッセージ構造』および「*All Things SWIFT: the SWIFT User Handbook*」を参照してください。各名前付き属性に対して、型 String を指定してください。このビジネス・オブジェクトのコンポーネントのアプリケーション固有の情報は、SWIFT タグです。

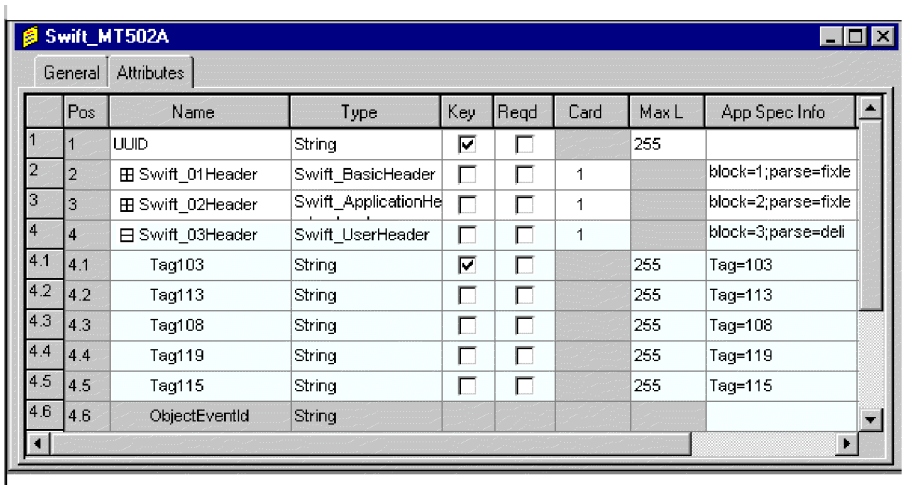


図 9. ブロック 3 ビジネス・オブジェクト定義

注: SWIFT メッセージに対してブロック 3 ビジネス・オブジェクト定義を作成するには、Business Object Designer Express を起動し、69 ページの『ブロック 3 ビジネス・オブジェクト定義のサンプル』に示された属性の値を入力します。

ブロック 3 ビジネス・オブジェクト定義のサンプル

このセクションでは、タイプ MT502 (購買または販売のオーダー) の SWIFT メッセージに対するブロック 3 ビジネス・オブジェクトの定義のサンプルを示します。

```
[BusinessObjectDefinition]
Name = Swift_UserHeader
Version = 1.1.0

   [Attribute]
   Name = Tag103
   Type = String
   Cardinality = 1
   MaxLength = 255
   IsKey = true
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = Tag=103
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = Tag113
   Type = String
   Cardinality = 1
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = Tag=113
   IsRequiredServerBound = false
   [End]
   [Attribute]
   Name = Tag108
   Type = String
   Cardinality = 1
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = Tag=108
```

```
IsRequiredServerBound = false
[End]
[Attribute]
Name = Tag119
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = Tag=119
IsRequiredServerBound = false
[End]
[Attribute]
Name = Tag115
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = Tag=115
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]
```

ブロック 4 ビジネス・オブジェクトの構造

SWIFT ブロック 4 には、SWIFT メッセージの本文が含まれています。ブロック 4 は、一方ではメッセージ・タグとその内容のフィールドから構成され、もう一方ではメッセージ・タグのシーケンスから構成されます。このデータ内容は、ブロック 1、2、および 3 の構造とは異なる、ブロック 4 ビジネス・オブジェクト構造を作成します。ブロック 4 ビジネス・オブジェクトは、メッセージ・データのビジネス・オブジェクト (MsgData BO) です。

SWIFT メッセージ内のすべてのタグとシーケンスが、MsgData BO の子ビジネス・オブジェクトとしてモデル化されます。したがって、MsgData BO は、Field ビジネス・オブジェクト (MsgField BO) と Sequence ビジネス・オブジェクト (MsgSeq BO) の 2 種類の子オブジェクトを持ちます。これらのビジネス・オブジェクトは、SWIFT データがブロック 4 でフォーマットされる方法を反映します。具体的には、これらのビジネス・オブジェクト内の属性は、SWIFT メッセージ・フォーマット仕様で指定された内容 (メッセージ・タグとその内容) と順序 (シーケンス) をモデル化します。メッセージ・タグのシーケンスは、ビジネス・オブジェクト定義が SWIFT メッセージを忠実に表す場合に重要です。MsgField BO および MsgSeq BO の詳細については、75 ページの『Sequence ビジネス・オブジェクトと Field ビジネス・オブジェクト』 を参照してください。

例として、「*SWIFT Standards Release Guide*」 から MT502 (購買または販売のオーダー) のフォーマット仕様を見てみましょう。後述の図 10 に、MT502 に対応するビジネス・オブジェクト定義の一部が示されています。ビジネス・オブジェクト定義は、SWIFT メッセージ内のメッセージ・タグとシーケンスの構造を反映しています。

- SWIFT メッセージ内の Status (M (必須) または O (オプション)) フィールドは、ビジネス・オブジェクト定義の Required プロパティーにマップされます。例えば、SWIFT Tag 98a (図 10 に示されています) の状況は、O (オプション) です。図 10 に、対応するビジネス・オブジェクト属性、Preparation_DateTime (タイプ `Swift_Tag_98`) を示します。これについては、Required プロパティーにチェックマークは付けられません。
- SWIFT メッセージの「タグ」、「修飾子」、および「内容/オプション (Content/Options)」フィールドは、ビジネス・オブジェクト定義内の属性アプリケーション固有テキストとしてマップされます。例えば、図 10 に示されている SWIFT メッセージでは、Start of Block は、Content が GENL である Tag16R です。図 10 に示されている対応する記入項目は、Tag、Tag の文字、Content (`Tag=16;Letter=R;Content=GENL`) を示すアプリケーション固有の情報プロパティー・パラメーターを持つタイプ `Swift_Tag_16` の属性 `Start_Of_Block` です。
- データ・フォーマットは、多くの場合、SWIFT メッセージの「内容/オプション (Content/Options)」フィールドに示されます。例えば、図 10 は、「Mandatory Sequence A General Information」に対する送信者の参照をデータ・フォーマット指示からなる SEME 修飾子と Content (`:4!c[/4!c]`) を持つ Tag20C として示しています。図 10 に、対応する属性アプリケーション固有テキストが示されています。「AppSpecInfo」フィールドには、Tag と Letter だけが示されています (`Tag=20;Letter=C`)。
- SWIFT メッセージ内でのシーケンスの繰り返しは、図 10 に示されているように、SWIFT フォーマット仕様で「`---->`」によって示されます。繰り返されないシーケンスには、「`-----|`」のマークが付けられます。ビジネス・オブジェクト定義では、シーケンスの繰り返しにカーディナリティー n が割り当てられます。例えば、図 10 に示されているシーケンス Tag22F の繰り返しは、カーディナリティー・プロパティー n でタイプ `Swift_Tag_22` の属性 Indicator にマップされます。

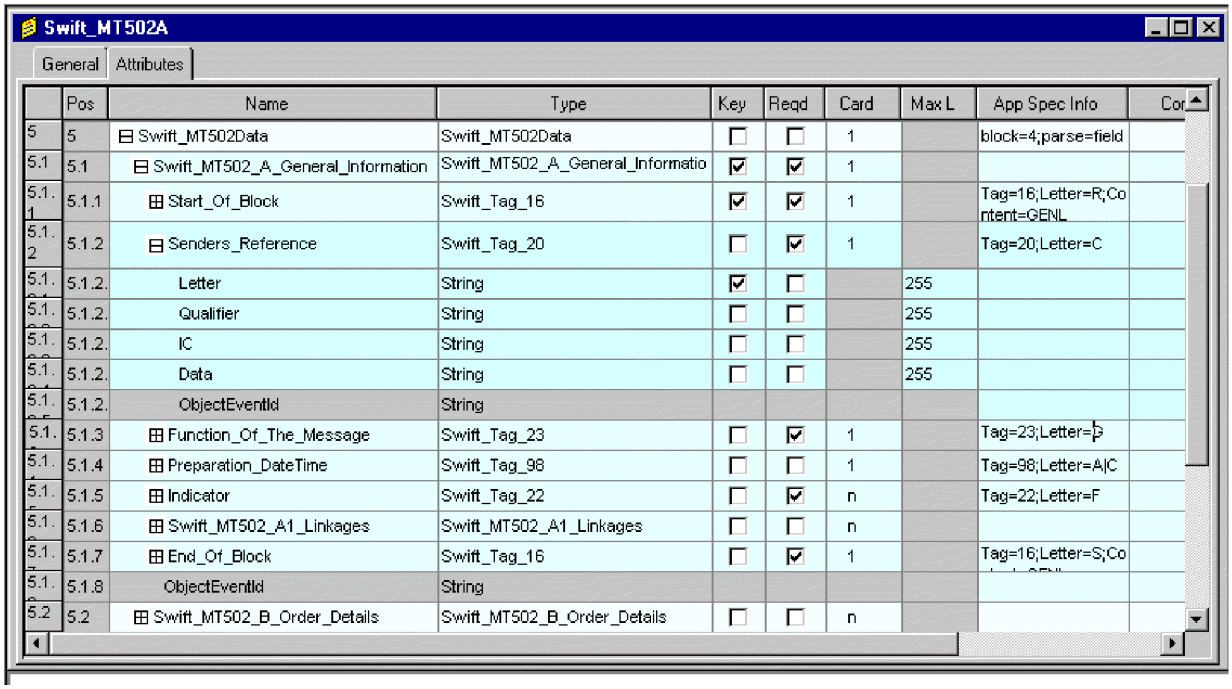


図 10. ブロック 4 ビジネス・オブジェクト定義の一部

MsgData BO フォーマット

MsgData BO のフォーマットの要約を以下の各セクションで示します。

MsgData BO 名: SWIFT メッセージのブロック 4 を表す MsgData BO の命名規則は次のとおりです。

```
Swift_MT<message_type>Data
```

例えば、次のようになります。

```
Name = Swift_MT502Data
```

MsgData BO 属性名: MsgData BO の各属性は、次のいずれかを表します。

- MsgSeq BO
- MsgField BO

したがって、属性名は MsgSeq BO および MsgField BO の属性名と同じになります。MsgField BO 属性の命名規則は次のとおりです。

```
Swift_<tag_number>_<position_in_the_SWIFT_message>
```

例えば、次のようになります。

```
Name = Swift_94_1
```

MsgSeq BO 属性の命名規則は次のとおりです。

```
Swift_MT<message_type>_<SWIFT_sequence_name>
```

例えば、次のようになります。

```
Name = Swift_MT502_B
```

詳細については、76 ページの『Sequence ビジネス・オブジェクトの構造』および 79 ページの『Field ビジネス・オブジェクト定義』を参照してください。

MsgData BO 属性のタイプ: MsgData 属性のタイプは次のとおりです。

MsgField BO 属性:

```
Swift_Tag_<tag_number>
```

例えば、次のようになります。

```
Type = Swift_Tag_94
```

MsgSeq BO 属性:

```
Swift_MT<message_type>_<SWIFT_sequence_name>
```

例えば、次のようになります。

```
Type = Swift_MT502_B
```

MsgData BO 属性の ContainedObjectVersion: MsgData BO およびその MsgSeq BO 属性に対して組み込まれたオブジェクト・バージョンは 1.1.0 です。例えば、次のようになります。

```
[Attribute]
Name = Swift_MT502_B
Type = Swift_MT502_B

...

ContainedObjectVersion = 1.1.0

...

[End]
```

注: MsgField BO 属性は単純な属性であり、ContainedObjectVersion を持ちません。

MsgData BO 属性の関係: MsgData BO およびその MsgSeq BO 属性の関係属性プロパティーは、Containment です。例えば、次のようになります。

```
[Attribute]
Name = Swift_MT502Data
Type = Swift_MT502Data

...

Relationship = Containment

...

[End]
```

MsgData BO 属性のカーディナリティー: MsgData BO およびその MsgSeq BO 属性のカーディナリティー・プロパティーは n です。フィールドの繰り返しを表す MsgField BO 属性のカーディナリティーも n です。その他の属性のカーディナリティーはすべて 1 です。例えば、次のようになります。

```
[Attribute]
Name = Swift_16_1
Type = Swift_Tag_16

...

Cardinality = n

...

[End]
```

MsgData BO 属性の IsKey: 各 MsgData BO 定義に、キー属性として定義された属性が少なくとも 1 つ含まれている必要があります (`IsKey = true`)。規則では、各 BO 定義内の最初の単一カーディナリティー属性をキー属性として定義する必要があります。

例えば、次のようになります。

```
[Attribute]
Name = Swift_16.1
Type = Swift_Tag_16

...

Cardinality = 1

IsKey = true

[End]
```

MsgData BO 属性の AppSpecificInfo: MsgData BO 定義では、MsgField BO 属性だけがアプリケーション固有の情報を持ちます。MsgSeq BO 属性では、このプロパティーは常にヌルです。MsgField BO 属性の場合、アプリケーション固有の情報の規則は次のとおりです。

```
Tag=nn;Letter=xx;Content=string
```

ここで、`nn` はフィールドの SWIFT タグ番号、*xx* はタグに対してサポートされる 1 つ以上の文字オプション、*string* は 55 ページの表 21 で説明されている非汎用フィールドの修飾子の値です。例えば、次のようになります。

```
[Attribute]
Name = Swift_16_22
Type = Swift_Tag_16

...

AppSpecificInfo = Tag=16;Letter=S;Content=OTHRPRTY

...

[End]
```

MsgField BO 属性が MsgSeq BO に表示され、アプリケーション固有の情報に次のものが示されている場合は、

```
...;Union=True
```

DataField 属性の代わりに MsgField 子オブジェクト (TagUnion ビジネス・オブジェクトとその子オブジェクトである TagLetterOption オブジェクト) に値が取り込まれ

ます。TagUnion ビジネス・オブジェクトの詳細については、79 ページの『Field ビジネス・オブジェクト定義』を参照してください。

Sequence ビジネス・オブジェクトと Field ビジネス・オブジェクト

前述のように、コネクターは SWIFT メッセージ内のシーケンスとタグをそれぞれ Sequence ビジネス・オブジェクト (MsgSeq BO) および Field ビジネス・オブジェクト (MsgField BO) としてモデル化します。図 11 に、これらのビジネス・オブジェクトの階層関係を示します。

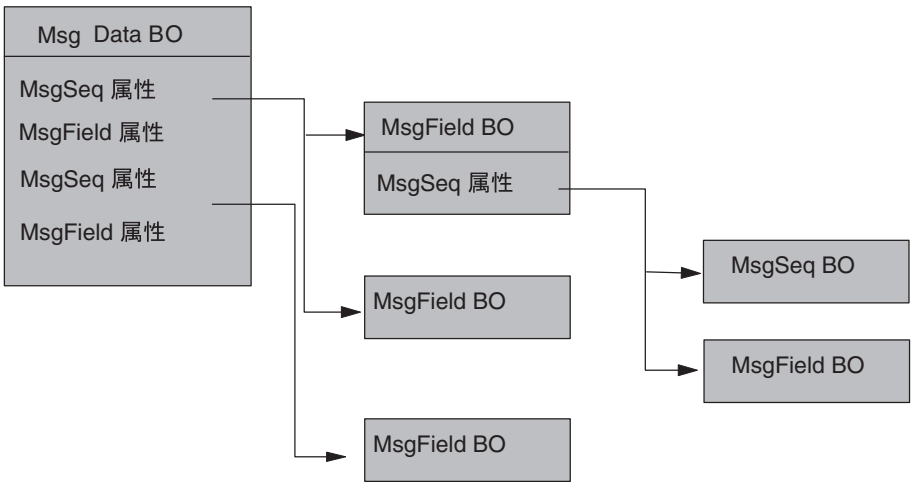


図 11. (ブロック 4) MsgData BO 内の Field ビジネス・オブジェクトと Sequence ビジネス・オブジェクト

図 12 に、フィールド属性とシーケンス属性を含むシーケンスを示す SWIFT メッセージ (MT502) の定義の一部を示します。シーケンス属性 `Swift_MT02_B_Order_Details` には、タイプ Tag のいくつかの属性 (例えば、`Swift_Tag_16`、`Swift_Tag_94`) だけでなく、サブシーケンス `Swift_MT502_B1_Price` も含まれています。このサブシーケンスはオプションのシーケンスの繰り返しであり、そのプロパティーはこれを反映します (`Required= no`; `Cardinality=n`)。サブシーケンスにはアプリケーション固有の情報は含まれていません。

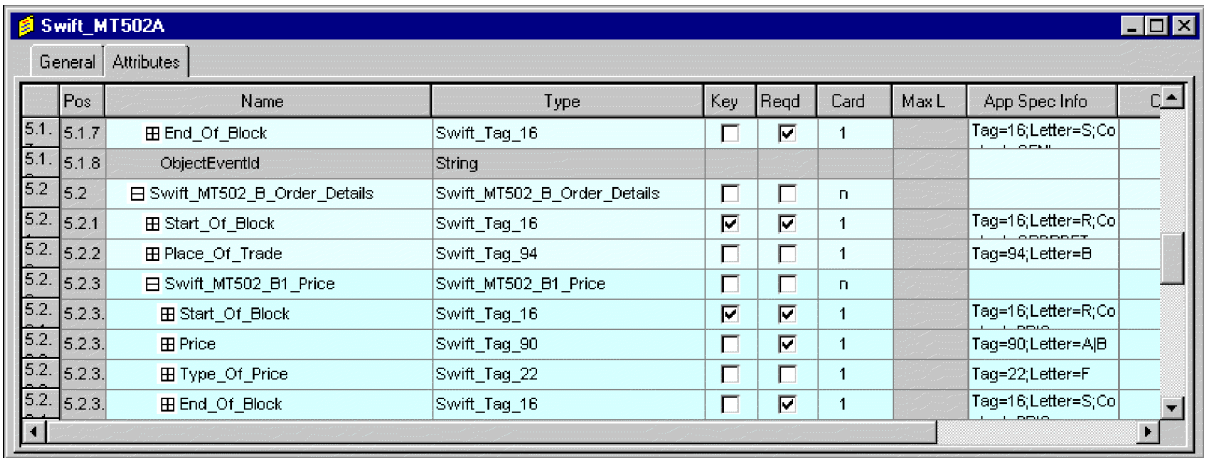


図 12. タグ属性とサブシーケンス属性を含むシーケンス

Sequence ビジネス・オブジェクトの構造

図 13 に示されているように、各 Sequence ビジネス・オブジェクト (MsgSeq BO) 属性は、次のいずれかのものを示します。

- 別の MsgSeq BO またはサブシーケンス
- MsgField BO

MsqSeq BO がネストできるサブシーケンスの数に制限はありません。

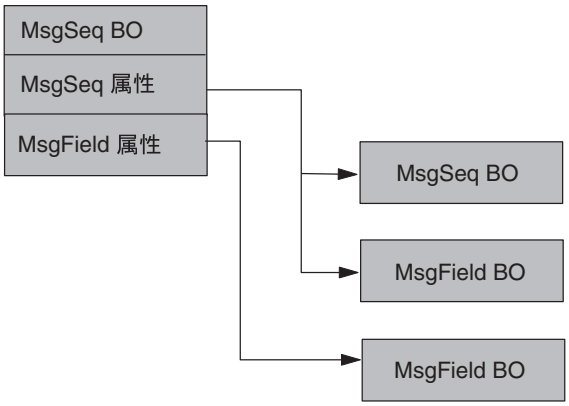
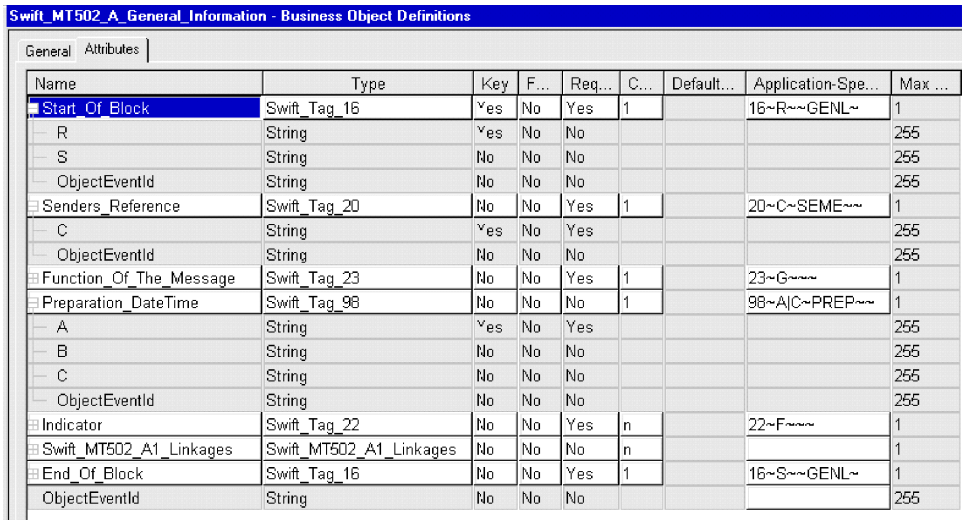


図 13. MsgSeq BO 内の Field ビジネス・オブジェクトと Subsequence ビジネス・オブジェクト

図 14 に、MsgSeq BO の別の抜粋を示します。この抜粋では、`Swift_Tag_` 属性が MsgField BO を示しています。`Swift_MT502_A1_Linkages` 属性は、サブシーケンス MsgSeq BO の子オブジェクトに対して使用されています。



Name	Type	Key	F...	Req...	C...	Default...	Application-Spe...	Max ...
Start_Of_Block	Swift_Tag_16	Yes	No	Yes	1		16~R~~GENL~	1
R	String	Yes	No	No				255
S	String	No	No	No				255
ObjectEventId	String	No	No	No				255
Senders_Reference	Swift_Tag_20	No	No	Yes	1		20~C~SEME~~	1
C	String	Yes	No	Yes				255
ObjectEventId	String	No	No	No				255
Function_Of_The_Message	Swift_Tag_23	No	No	Yes	1		23~G~~~	1
Preparation_DateTime	Swift_Tag_98	No	No	No	1		98~A|C~PREP~~	1
A	String	Yes	No	Yes				255
B	String	No	No	No				255
C	String	No	No	No				255
ObjectEventId	String	No	No	No				255
Indicator	Swift_Tag_22	No	No	Yes	n		22~F~~~	1
Swift_MT502_A1_Linkages	Swift_MT502_A1_Linkages	No	No	No	n			1
End_Of_Block	Swift_Tag_16	No	No	Yes	1		16~S~~GENL~	1
ObjectEventId	String	No	No	No				255

図 14. Sequence ビジネス・オブジェクト (MsgSeq BO) からの抜粋

Sequence ビジネス・オブジェクトには次の規則が適用されます。

- Subsequence ビジネス・オブジェクトは、特定の Sequence ビジネス・オブジェクト・タイプの属性です。
- 複数のフィールドの繰り返しの集合は、サブシーケンスとして扱われます。
- シーケンス属性のアプリケーション固有の情報は常に NULL です。

Sequence ビジネス・オブジェクトのサンプルは、77 ページの『Sequence ビジネス・オブジェクト定義のサンプル』を参照してください。

MsgSeq BO フォーマット

MsgData BO と同様に、MsgSeq BO は、MsgSeq BO または MsgField BO のいずれかの属性から構成されます。これらの属性のフォーマットについては、72 ページの『MsgData BO フォーマット』を参照してください。

Sequence ビジネス・オブジェクト定義のサンプル

このセクションでは、タイプ MT502 (購買または販売のオーダー) の SWIFT メッセージに対する MsgSeq BO の定義のサンプルを示します。この定義は、「Mandatory Sequence A Order to Buy or Sell」のものです。

```
[BusinessObjectDefinition]
Name = Swift_MT502_A_General_Information
Version = 1.0.0

   [Attribute]
   Name = Start_Of_Block
   Type = Swift_Tag_16
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 1
   IsKey = true
   IsForeignKey = false
   IsRequired = true
   AppSpecificInfo = Tag=16;Letter=R;Content=GENL
   IsRequiredServerBound = false
   [End]
   [Attribute]
```

```
Name = Senders_Reference
Type = Swift_Tag_20
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = Tag=20;Letter=C
IsRequiredServerBound = false
[End]
[Attribute]
Name = Function_Of_The_Message
Type = Swift_Tag_23
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = Tag=23;Letter=G
IsRequiredServerBound = false
[End]
[Attribute]
Name = Preparation_DateTime
Type = Swift_Tag_98
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = Tag=98;Letter=A|C
IsRequiredServerBound = false
[End]
[Attribute]
Name = Indicator
Type = Swift_Tag_22
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = Tag=22;Letter=F
IsRequiredServerBound = false
[End]
[Attribute]
Name = Swift_MT502_A1_Linkages
Type = Swift_MT502_A1_Linkages
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = End_Of_Block
Type = Swift_Tag_16
ContainedObjectVersion = 1.0.0
```

```
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = true
AppSpecificInfo = Tag=16;Letter=S;Content=GENL
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Create
[End]

[Verb]
Name = Retrieve
[End]
```

Field ビジネス・オブジェクト定義

WebSphere は、すべての SWIFT タグを Field ビジネス・オブジェクト (MsgField BO) として表します。各 MsgField BO は、フィールドが非汎用であっても、SWIFT 汎用フィールド構造を使用してモデル化されます。WebSphere は 2 つの追加のビジネス・オブジェクト・モデルを使用して、SWIFT メッセージ・コンポーネントの表示と結合に使用される文字とオプションの組み合わせをビジネス・オブジェクト内のサブフィールドとして表します。

- **Tag union ビジネス・オブジェクト (TagUnion BO)** これは、MsgField BO の子オブジェクトです。TagUnion BO には、特定のタグに使用できるすべての文字オプションが含まれています。これは、特定のメッセージ・タイプに固有のものではありません。
- **Tag letter option ビジネス・オブジェクト (TagLetterOption BO)** これは、サブフィールドの内容および区切り文字を含むフォーマットを定義する TagUnion BO の文字オプション子オブジェクトです。

MsgField BO フォーマット

図 15 に示されているように、各 MsgField BO には 5 つの属性があります。このうち、TagUnion BO が 1 つだけ含まれており、そのデータ型は以下で括弧 () 内に示されています。

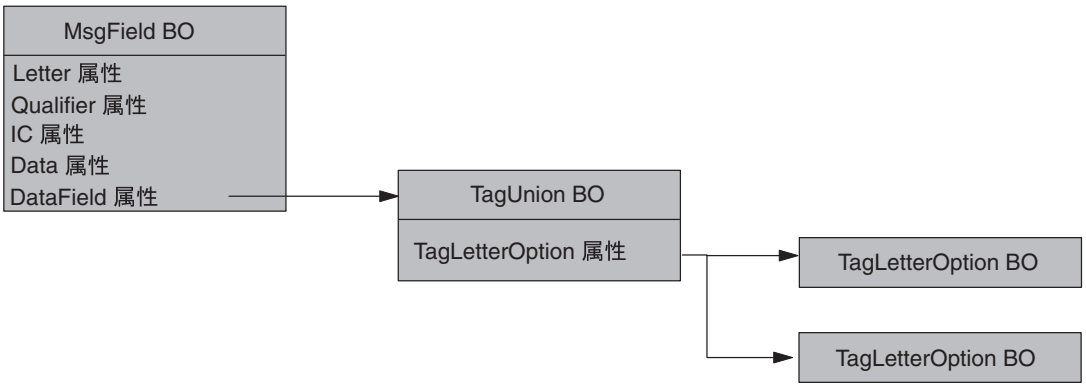


図 15. MsgField BO の属性とビジネス・オブジェクト

SWIFT Qualifier および Issuer Code (IC) 以外のすべてのサブフィールドの内容と順序が、DataField の子オブジェクトに取り込まれます。これは、TagUnion BO およびその子オブジェクト TagLetterOption BO です。図 15 に示されている属性およびビジネス・オブジェクトについては、以下で説明します。

MsgField BO、TagUnion BO、および TagLetterOption BO の名前: MsgField BO の命名規則は次のとおりです。

```
Swift_Tag_<N>
```

ここで、N はメッセージ番号を表します。例えば、次のようになります。

```
Name = Swift_Tag_22
```

TagUnion BO の命名規則は次のとおりです。

```
Swift_Tag_Union_<tag_number>
```

ここで、tag_number はタグ番号の数値表現です。例えば、次のようになります。

```
Name = Swift_Tag_Union_20
```

TagLetterOption BO の命名規則は次のとおりです。

```
Swift_Tag_Union_<tag_number>_Opt_[<letter_option>]
```

ここで、tag_number はタグ番号の数値表現、[<letter_option>] はタグが文字に関連付けられている場合の文字オプションです。タグに文字が関連付けられていない場合、名前は Opt で終わります。例えば、次のようになります。

```
Name = Swift_Tag_Union_20_Opt_C
```

MsgField BO、TagUnion BO、および TagLetter BO Attribute の名前:
MsgField BO 内の 5 つの属性の名前は次のとおりです。

- Letter
- Qualifier
- IC
- Data
- DataField

TagUnion BO 内の属性の名前は次のとおりです。

```
Swift_<tag_number>_[<letter_option>]
```

ここで、tag_number はタグ番号の数値表現、大括弧は、文字がタグに関連付けられている場合のみ付加されることを示します。例えば、次のようになります。

```
Swift_20_C
```

TagLetterOption BO 内の属性の名前は、SWIFT フォーマット仕様テーブルに示されているサブフィールド名のワードを連結したものです。SWIFT フォーマット仕様でのワードのスペルに関わりなく、連結されたストリング内の各ワードの先頭文字は常に大文字であり、後続の文字は小文字です。スペースおよび英字記号以外の記号は連結名から除外されます。フィールドにサブフィールドがない場合は、Subfield が属性名として使用されます。例えば、95R 内のサブフィールド「Proprietary」の場合、TagLetterOption BO Swift_Tag_Union_95_Opt_R の定義内の対応する属性名は次のようになります。

```
Name = ProprietaryCode
```

MsgField BO、TagUnion BO、および TagLetterOption BO 属性タイプ:
MsgField 属性のタイプは次のとおりです。

- Letter (String)
- Qualifier (String)
- Issuer Code (String)
- Data (String)
- DataField (*TagUnion_BO*)

例えば、MsgField BO 定義では、Swift_Tag_20 属性のタイプは次のように示されます。

```
   [Attribute]
   Name = DataField
Type = Swift_Tag_Union_20
```

TagUnion BO 内の属性のタイプは、TagLetterOption BO 子オブジェクトの名前になります。例えば、Swift_Tag_Union_20 の TagUnion BO 定義では、TagLetterOption 属性のタイプは次のようになります。

```
   [Attribute]
Name = Swift_20_C
Type = Swift_Tag_Union_20_Opt_C
```

TagLetterOption BO 内の属性のタイプは常に String です。

MsgField BO、TagUnion BO、および TagLetterOption BO ContainedObjectVersion: MsgField BO、TagUnion BO、および TagLetterOption BO に対して組み込まれているオブジェクト・バージョンは 1.1.0 です。例えば、次のようになります。

また、MsgSeq BO 属性の場合も 1.1.0 です。例えば、次のようになります。

```
[Attribute]
Name = Swift_20_C
Type = Swift_Tag_Union_20_Opt_C

...
```

```
ContainedObjectVersion = 1.1.0

...
[End]
```

注: MsgField BO 属性は単純な属性であり、ContainedObjectVersion を持ちません。

MsgField BO、TagUnion BO、および TagLetterOption BO 属性のカーディナリティー: TagUnion BO および TagLetterOption BO 内の属性のカーディナリティーは、常に 1 に設定されます。例えば、次のようになります。

```
[Attribute]
Name = Swift_20_C
Type = Swift_Tag_Union_20_Opt_C

...

Cardinality = 1

...

[End]
```

MsgField BO、TagUnion BO、および TagLetterOption BO 属性の IsKey: 各 MsgField BO で、属性 Letter をキー属性として定義する必要があります。

例えば、次のようになります。

```
[Attribute]

Name = Letter
Type = String
IsKey = true

...

[End]
```

TagUnionBO の最初の属性がキーとして定義されます。

TagLetterOption BO の最初の属性がキーとして定義されます。

TagLetterOption BO 属性の AppSpecificInfo: TagLetterOption BO の AppSpecificInfo 属性の定義は、ビジネス・オブジェクト・サブフィールドに重要な SWIFT メッセージ・フォーマット情報を提供します。AppSpecificInfo 属性には、次の情報が含まれている必要があります。

```
Format=***;Delim=$$$
```

ここで、以下のように説明されます。

*** は、区切り文字情報を除外した SWIFT サブフィールド・フォーマット仕様を表します。

$$$ は、現在のサブフィールドと次のサブフィールドの間の区切り文字を構成する 1 つ以上の文字を表します。

区切り文字が CrLf である場合は、シンボル・ストリング CrLf が、復帰文字のすぐ後に改行文字が続くことを指定します。

例えば、TagLetterOption BO、Swift_Tag_Union_95_Opt_C の AppSpecificInfo 属性は次のように表示されます。

```
[Attribute]

Name = CountryCode
Type = String

...

AppSpecificInfo = Format=2!a;Delim=/

...

[End]
```

サンプル・オブジェクトと属性定義は、83 ページの『MsgField BO、TagUnion BO、および TagLetterOption BO の定義のサンプル』を参照してください。

MsgField BO、TagUnion BO、および TagLetterOption BO の定義のサンプル

このセクションでは、TagUnion および TagLetterOption の属性とオブジェクトを示す MsgField BO 定義のサンプルを示します。

サンプル MsgField BO、Swift_Tag_21 は次のとおりです。

```
[BusinessObjectDefinition]
Name = Swift_Tag_21
Version = 3.0.0

   [Attribute]
   Name = Letter
   Type = String
   MaxLength = 255
   IsKey = true
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = Qualifier
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = IC
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = Data
   Type = String
   MaxLength = 255
```

```
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = DataField
   Type = Swift_Tag_Union_21
   ContainedObjectVersion = 3.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 0
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = ObjectEventId
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Verb]
   Name = Create
   [End]

   [Verb]
   Name = Delete
   [End]

   [Verb]
   Name = Retrieve
   [End]

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

DataField 属性が、Type 属性によって名前を定義された TagUnion BO、Swift_Tag_Union_21 を表していることに注意してください。次に、その TagUnion BO を示します。これは、Swift_Tag_21 のすべての文字オプションを属性として示します。

```
[BusinessObjectDefinition]
Name = Swift_Tag_Union_21
Version = 1.1.0

   [Attribute]
   Name = Swift_21
   Type = Swift_Tag_Union_21_Opt
   ContainedObjectVersion = 1.0.0
   Relationship = Containment
   Cardinality = 1
   MaxLength = 0
   IsKey = true
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
```

```
[End]

[Attribute]
Name = Swift_21_A
Type = Swift_Tag_Union_21_Opt_A
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_B
Type = Swift_Tag_Union_21_Opt_B
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_C
Type = Swift_Tag_Union_21_Opt_C
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_D
Type = Swift_Tag_Union_21_Opt_D
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_E
Type = Swift_Tag_Union_21_Opt_E
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = Swift_21_F
Type = Swift_Tag_Union_21_Opt_F
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_G
Type = Swift_Tag_Union_21_Opt_G
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_N
Type = Swift_Tag_Union_21_Opt_N
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_P
Type = Swift_Tag_Union_21_Opt_P
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Swift_21_R
Type = Swift_Tag_Union_21_Opt_R
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
```

```
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Verb]
   Name = Create
   [End]

   [Verb]
   Name = Retrieve
   [End]

[End]
```

上記の TagUnion BO の最初の属性 Swift_21 で、IsKey = true であることに注意してください。

属性 Swift_21_A は、子オブジェクト TagLetterOption BO を示しています。この子オブジェクトの名前は、属性の Type 属性によって定義されます (Swift_Tag_Union_21_Opt_A)。次に、その TagLetterOption BO を示します。

```
[BusinessObjectDefinition]
Name = Swift_Tag_Union_21_Opt_A
Version = 1.0.0

   [Attribute]
   Name = ReferenceOfTheIndividualAllocation
   Type = String
   MaxLength = 255
   IsKey = true
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = Format=16x
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = ObjectEventId
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Verb]
   Name = Create
   [End]

   [Verb]
   Name = Retrieve
   [End]
```

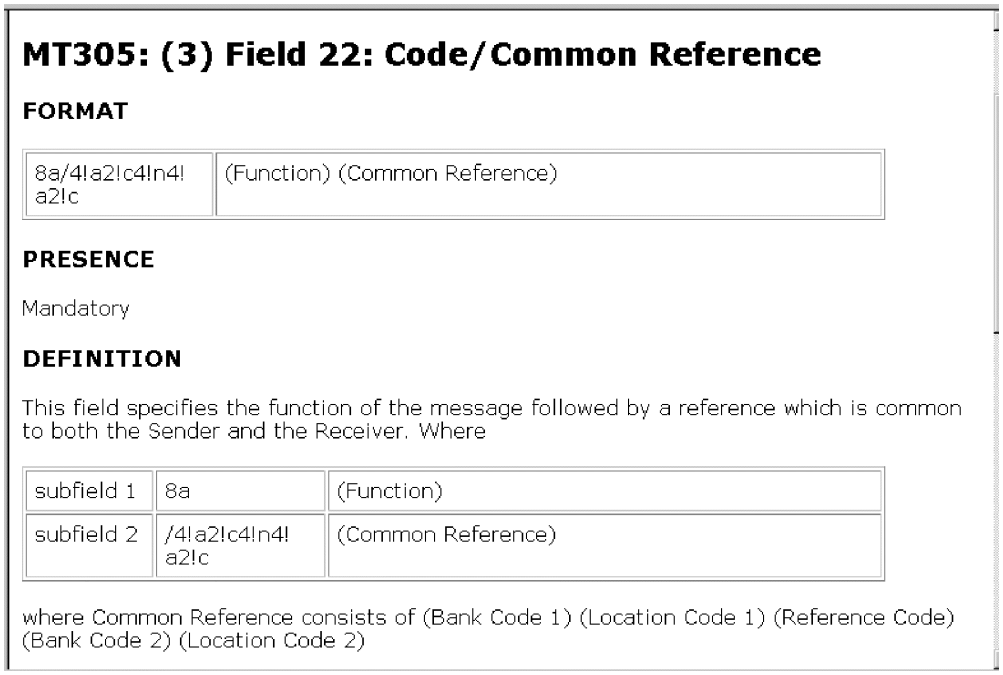
この TagLetterOption BO の唯一の属性である ReferenceOfTheIndividualAllocation が、このタグ・オプションの対応する SWIFT サブフィールド名の連結であることに注意してください。各ワードの先頭文

字は大文字です。Qualifier サブフィールドおよび Issuer Code サブフィールドは、TagLetterOption BO の属性から除外されます。この属性では、IsKey プロパティーも true です。

注: TagUnion BO には、汎用フィールドと非汎用フィールドの両方が含まれています。非汎用フィールドにはサブフィールドがありません。

TagLetterOption BO は、単純および複雑な SWIFT フィールドおよびサブフィールドのフォーマットを表すことができます。次に、Swift_Tag_Union_22_Opt のビジネス・オブジェクト定義、TagLetterOption BO を示します。この属性とアプリケーション固有の情報は、SWIFT Field 22、送信側と受信側との Common Reference 機能のサブフィールド・フォーマットを指定します。Function の AppSpecificInfo が、SWIFT メッセージでのデータの解析に使用されるフォーマットと区切り文字を指定することに注意してください。CommonReference は、サブフィールド名の連結です。CommonReference の AppSpecificInfo は、図 16 に示されている内容に対応します。

```
[BusinessObjectDefinition]
Name = Swift_Tag_Union_22_Opt
Version = 1.0.0

   [Attribute]
   Name = Function
   Type = String
   MaxLength = 255
   IsKey = true
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = Format=8a;Delim=/
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = CommonReference
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   AppSpecificInfo = Format=4!a2!c4!n4!a2!c
   IsRequiredServerBound = false
   [End]

   [Attribute]
   Name = ObjectEventId
   Type = String
   MaxLength = 255
   IsKey = false
   IsForeignKey = false
   IsRequired = false
   IsRequiredServerBound = false
   [End]

   [Verb]
   Name = Create
   [End]

   [Verb]
   Name = Retrieve
   [End]
[End]
```



MT305: (3) Field 22: Code/Common Reference

FORMAT

8a/4!a2!c4!n4! a2!c	(Function) (Common Reference)

PRESENCE

Mandatory

DEFINITION

This field specifies the function of the message followed by a reference which is common to both the Sender and the Receiver. Where

subfield 1	8a	(Function)
subfield 2	/4!a2!c4!n4! a2!c	(Common Reference)

where Common Reference consists of (Bank Code 1) (Location Code 1) (Reference Code) (Bank Code 2) (Location Code 2)

図 16. SWIFT フィールド定義

第 4 章 SWIFT データ・ハンドラー

SWIFT データ・ハンドラーは、ビジネス・オブジェクトから SWIFT メッセージへ、および SWIFT メッセージからビジネス・オブジェクトへの変換を主な役割とするデータ変換モジュールです。デフォルトのトップレベル・データ・ハンドラー・メタオブジェクトのコネクターとサーバーは、ともに、swift MIME タイプをサポートし、したがって、SWIFT データ・ハンドラーの使用もサポートします。

この章では、SWIFT データ・ハンドラーが SWIFT メッセージをどのように処理するかについて説明します。また、SWIFT データ・ハンドラーの構成方法についても説明します。

SWIFT データ・ハンドラーの構成

コネクターと使用するために SWIFT データ・ハンドラーを設定するには、以下の手順を実行する必要があります。

- SWIFT データ・ハンドラーのクラス名がコネクター・プロパティーで指定されていることを確認してください。
- SWIFT データ・ハンドラー子メタオブジェクトの属性に対して、該当する値を入力します。

注: SWIFT データ・ハンドラーが正常に機能するには、ビジネス・オブジェクト定義を、データ・ハンドラーをサポートするように作成または変更する必要があります。詳細については、 128 ページの『SWIFT フィールド構造』を参照してください。

コネクター・メタオブジェクトの構成

SWIFT データ・ハンドラーと対話するようにコネクターを構成するには、コネクター固有プロパティー DataHandlerClassName の値が、com.crossworlds.DataHandlers.swift.SwiftDataHandler であることを確認します。

コネクターを実行する前に、このプロパティーの値を設定する必要があります。それによって、SWIFT メッセージとビジネス・オブジェクトの間で相互変換を行う際にコネクターが SWIFT データ・ハンドラーにアクセスできるようになります。詳細については、22 ページの『コネクター固有のプロパティー』を参照してください。

データ・ハンドラー子メタオブジェクトの構成

WebSphere には、SWIFT データ・ハンドラー用のデフォルト・メタオブジェクトの MO_DataHandler_Default が添付されています。このメタオブジェクトでは、タイプ MO_DataHandler_Swift の子属性が指定されています。表 26 に、子メタオブジェクト MO_DataHandler_SWIFT 内の属性を説明します。

表 26. SWIFT データ・ハンドラーの子メタオブジェクト属性

属性名	説明	添付されているデフォルト値
BOPrefix	ビジネス・オブジェクト名を作成するためにデフォルトの NameHandler クラスによって使用される接頭部。デフォルト値は、ビジネス・オブジェクトのタイプに合わせて変更する必要があります。属性値は大文字小文字を区別します。	Swift
DefaultVerb	ビジネス・オブジェクト作成時に使用される動詞。	Create
ClassName	指定された MIME タイプで使用するためにロードするデータ・ハンドラー・クラスの名前。トップレベルのデータ・ハンドラー・メタオブジェクトの属性は、その名前が指定された MIME タイプと一致し、そのタイプが SWIFT 子メタオブジェクトとなります。	com.crossworlds.DataHandlers.swift.SwiftDataHandler
DummyKey	キー属性; データ・ハンドラーによっては使用されませんが、統合ブローカーに必要です。	1

表 26 の添付されているデフォルト値の列に、関連するメタオブジェクト属性のデフォルト値に対して WebSphere が提供する値が示されています。この子メタオブジェクトのすべての属性が、実際のシステムと SWIFT メッセージ・タイプに適切なデフォルト値を持っていることを確認する必要があります。また、少なくとも、ClassName 属性と BOPrefix 属性がデフォルト値を持っていることを確認してください。

注: このメタオブジェクト内の属性にデフォルト値を割り当てるには、Business Object Designer Express を使用します。

ビジネス・オブジェクトの要件

SWIFT データ・ハンドラーは、ビジネス・オブジェクトまたは SWIFT メッセージを変換するときにビジネス・オブジェクト定義を使用します。変換は、ビジネス・オブジェクトの構造とそのアプリケーション固有テキストを使用して実行されます。SWIFT データ・ハンドラーの要件に、ビジネス・オブジェクト定義を準拠させるには、51 ページの『第 3 章 ビジネス・オブジェクト』に説明されている指針に従います。

SWIFT メッセージへのビジネス・オブジェクトの変換

ビジネス・オブジェクトを SWIFT メッセージに変換するため、SWIFT データ・ハンドラーは、トップレベルのビジネス・オブジェクト内の属性を順番にループします。これは、属性がビジネス・オブジェクトとその子に現れる順序に基づいて、取り込まれた SWIFT メッセージのブロックを循環的に生成します。

ブロック番号のない属性、すなわちパーサー・プロパティーによって認識されない値を持つ属性は、無視されます。また、MQSA によって追加される UUID ヘッダーであるブロック 0 も無視されます。

parse=value アプリケーション固有の情報プロパティーは、ストリングのフォーマット方法を決定するために使用されます。このプロパティーは、次のようにビジネス・オブジェクトを解析します。

- `parse=no;` 属性はタイプ `String` である必要があります。これは、{*block number*:*attribute value*} としてフォーマットされます。*block number* は、`block=`*block value* アプリケーション固有テキスト・プロパティーの値です。
- `parse=fixlen;` この属性は、単一のカーディナリティー・コンテナーである必要があります。これは、{*block number*:*attr0 value* a*ttr1 value....attrn value*}としてフォーマットされます。ここで、*attrn value* は、*n* 番目の属性の属性値です。CxIgnore 属性と CxBlank 属性はすべて無視されます。
- `parse=delim;` この属性は、単一のカーディナリティー・コンテナーである必要があります。これは、{*block number*:[*Tag*:*attr1 data*]...[*Tag*:*attr1 data*]} としてフォーマットされます。ここで、*Tag* は属性アプリケーション固有テキストの Tag プロパティーの値で、*attrn data* は属性の値です。CxIgnore 属性と CxBlank 属性はすべて無視されます。
- `parse=field;` この設定は、ブロック 4 のメッセージでのみ使用できます。フィールドは、ビジネス・オブジェクトの CxIgnore と CxBlank 以外の属性によってループで印刷されます。
  - `appText == NULL` で、属性がコンテナーである場合は、`printBO(childBO)` を呼び出します。必要に応じて複数のカーディナリティーを処理します。
  - `appText != NULL` である場合は、`printFieldObj()` を呼び出します。これは、複数のカーディナリティーを処理し、タグを書き出すために `printFieldBO()` を呼び出します。
- すべてのフィールドが汎用フィールドまたは非汎用フィールドとしてフォーマットされます。タグ番号は、Tag ビジネス・オブジェクト属性の値として判別されます。Tag ビジネス・オブジェクトの非 CxIgnore 属性はすべて印刷されます。汎用フィールドまたは非汎用フィールドの詳細については、127 ページの『付録 C. SWIFT メッセージ構造』を参照してください。

ビジネス・オブジェクトへの SWIFT メッセージの変換

すべての SWIFT メッセージと、SWIFT フォーマットおよび構文との整合性が、SWIFT データ・ハンドラーによる処理の前に SWIFT によって検証されます。SWIFT データ・ハンドラーは、ビジネス・オブジェクトの構造と整合性だけを検証します。

SWIFT データ・ハンドラーは、次のように SWIFT メッセージからデータを抽出し、ビジネス・オブジェクト内の対応する属性を設定します。

1. SWIFT パーサーが呼び出され、最初の 4 ブロック (UUID およびブロック 1 から 3) が抽出されます。ブロック 2 の SWIFT アプリケーション・ヘッダーでは、入力属性だけが抽出されます。
2. SWIFT データ・ハンドラーが呼び出され、SWIFT メッセージのブロック 2 からビジネス・オブジェクトの名前が抽出されます。

3. SWIFT データ・ハンドラーは、トップレベル・オブジェクトのインスタンスを作成します。
4. データ・ハンドラーは、アプリケーション固有の情報パラメーターに基づいて、SWIFT メッセージ・ブロックを処理します。ブロックは、4 つの異なる方法のいずれかで解析されます。
   - `parse=no;` ブロック・データはタイプ `String` として扱われ、解析されません。
   - `parse=fixlen;` ブロック・データは、Block ビジネス・オブジェクトの最大長属性の値に基づき、固定長構造として解析されます。
   - `parse=delim;` ブロック・データは {*n*:*data*} という区切られたフォーマットとして解析されます。
   - `parse=field;` この設定は、ブロック 4 のデータでのみ使用されます。フィールドは、汎用または非汎用として解析されます。
5. ブロック 4 のデータ (`parse=field;`) の場合、データ・ハンドラーは、パーサーから Tag ビジネス・オブジェクト属性に戻された属性フィールドを突き合わせるか、あるいはフィールドが属す Sequence ビジネス・オブジェクトを検出します。
   a. 属性のアプリケーション固有情報が `NULL` である場合、子ビジネス・オブジェクトはシーケンスです。データ・ハンドラーは、最初に必要な子ビジネス・オブジェクトの属性がフィールドと一致するかどうかを調べます。
      - 一致する場合、データ・ハンドラーは属性に複数のカーディナリティーを割り当て、子ビジネス・オブジェクト用のシーケンスにデータを取り込みます。
      - 一致しない場合、データ・ハンドラーは、親ビジネス・オブジェクトの次の属性にスキップします。
   b. アプリケーション固有の情報が NULL ではない場合、その子は Tag ビジネス・オブジェクトです。アプリケーション固有の情報に一致する場合、フィールドは、複数のカーディナリティーで処理され、抽出されます。この場合、データ・ハンドラーは Tag ビジネス・オブジェクトの文字属性およびデータ属性を設定します。
6. 非 NULL フィールドが戻された場合、そのフィールドはログに書き込まれ、例外がスローされます。
7. データ・ハンドラーは SWIFT メッセージのブロック 5 を解析します。このブロックのアプリケーション固有情報は常に `block=5; parse=no` であり、タイプは `String` です。ブロック 5 は単一のストリングとして扱われます。

第 5 章 トラブルシューティング

この章では、コネクターの始動または稼働時に発生する可能性のある問題について説明します。

始動時の問題

問題	考えられる解決策と説明
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException...」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル jms.jar を検出できません。start_connector.bat の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/mq/jms/MQConnectionFactory...」が報告される。	コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーからファイル com.ibm.mqjms.jar を検出できません。start_connector.bat の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
初期化中にコネクターが予期せずシャットダウンし、メッセージ「Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...」が報告される。	コネクターが IBM WebSphere MQ Java クライアント・ライブラリーからファイル jndi.jar を検出できません。start_connector.bat の変数 MQSERIES_JAVA_LIB が IBM WebSphere MQ Java クライアント・ライブラリー・フォルダーを指していることを確認してください。
コネクターが「MQJMS2005: failed to create MQQueueManager for ':'」を報告する。	プロパティー HostName、Channel、および Port の値を明示的に設定します。

イベント処理

問題	考えられる解決策と説明
コネクターが、MQRFH2 ヘッダーの付いたすべてのメッセージを送達する。	MQMD WebSphere MQ ヘッダーの付いたメッセージのみを送達するには、?targetClient=1 を出力キュー URI の名前に追加します。例えば、メッセージをキュー queue://my.queue.manager/OUT に出力するには、URI を queue://my.queue.manager/OUT?targetClient=1 に変更します。詳細については、19 ページの『第 2 章 コネクターのインストールと構成』を参照してください。
コネクターが、コネクター・メタオブジェクトでのフォーマットの定義に関係なく、すべてのメッセージ・フォーマットを送達時に 8 文字に切り捨てる。	これは、コネクターではなく、WebSphere MQ MQMD メッセージ・ヘッダーの制約です。

付録 A. コネクターの標準構成プロパティー

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

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

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

標準コネクター・プロパティーの構成

Adapter コネクターには 2 つのタイプの構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有のプロパティー

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

Connector Configurator Express の使用

コネクター・プロパティーの構成は Connector Configurator Express から行います。Connector Configurator Express には、System Manager からアクセスします。Connector Configurator Express の使用方法の詳細については、付録 B『Connector Configurator Express』を参照してください。

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

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

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

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

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

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

標準プロパティーの要約

表 27 は、標準コネクター構成プロパティーの早見表です。標準プロパティーの依存関係は RepositoryDirectory に基づいているため、コネクターによっては使用されないプロパティーがあり、使用する統合ブローカーによってプロパティーの設定が異なる可能性があります。

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

表 27. 標準構成プロパティーの要約

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

表 27. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
ContainerManagedEvents	値なしまたは JMS	値なし	コンポーネント再始動	Delivery Transport は JMS
ControllerStoreAndForwardMode	true または false	true	動的	Repository Directory は <REMOTE>
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	コンポーネント再始動	JMS トランスポートのみ
DeliveryTransport	IDL または JMS	IDL	コンポーネント再始動	
DuplicateEventElimination	true または false	false	コンポーネント再始動	JMS トランスポートのみ: Container Managed Events は <NONE> でなければならない
EnableOidForFlowMonitoring	true または false	false	コンポーネント再始動	
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	crossworlds.queue.manager	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
JvmMaxHeapSize	ヒープ・サイズ (メガバイト単位)	128m	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMaxNativeStackSize	スタックのサイズ (キロバイト単位)	128k	コンポーネント再始動	Repository Directory は <REMOTE>
JvmMinHeapSize	ヒープ・サイズ (メガバイト単位)	1m	コンポーネント再始動	Repository Directory は <REMOTE>
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	

表 27. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
LogAtInterchangeEnd	`true` または `false`	`false`	コンポーネント再始動	
MaxEventCapacity	`1` から `2147483647`	`2147483647`	動的	Repository Directory は <REMOTE>
MessageFileName	パスまたはファイル名	`InterchangeSystem.txt`	コンポーネント再始動	
MonitorQueue	任意の有効なキュー名	*CONNECTORNAME*`/MONITORQUEUE`	コンポーネント再始動	JMS トランスポートのみ: DuplicateEvent Elimination は true でなければならない。
OADAutoRestartAgent	`true` または `false`	`false`	動的	Repository Directory は <REMOTE>
OADMaxNumRetry	正数	`1000`	動的	Repository Directory は <REMOTE>
OADRetryTimeInterval	正数 (単位: 分)	`10`	動的	Repository Directory は <REMOTE>
PollEndTime	`HH:MM` (HH は 0 から 23、MM は 0 から 59)	`HH:MM`	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) `no` (ポーリングを使用不可にする) `key` (コネクターのコマンド・プロンプト・ウィンドウで文字 `p` が入力された場合にのみポーリングする)	`10000`	動的	
PollQuantity	`1` から `500`	`1`	エージェント再始動	JMS トランスポートのみ: Container Managed Events を指定
PollStartTime	`HH:MM` (HH は 0 から 23、MM は 0 から 59)	`HH:MM`	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	<REMOTE> に設定する
RequestQueue	有効な JMS キュー名	*CONNECTORNAME*`/REQUESTQUEUE`	コンポーネント再始動	Delivery Transport は JMS
ResponseQueue	有効な JMS キュー名	*CONNECTORNAME*`/RESPONSEQUEUE`	コンポーネント再始動	Delivery Transport は JMS
RestartRetryCount	`0` から `99`	`3`	動的	
RestartRetryInterval	適切な正数 (単位: 分): 1 から 2147483547	`1`	動的	

表 27. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
SourceQueue	有効な JMS キュー名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue	有効な JMS キュー名	*CONNECTORNAME*/SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue	有効な JMS キュー名	*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwBO	CwBO	エージェント再始動	

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

使用する統合ブローカーを指定します。ICS を指定する必要があります。

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

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

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

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

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

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

ContainerManagedEvents

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

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

デフォルト値は No value です。

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

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

また、MimeType、DHClass、および DataHandlerConfigMOName (オプショナル) プロパティーを設定したデータ・ハンドラーも構成する必要があります。これらのプロパティーの値を設定するには、Connector Configurator Express の**「データ・ハンドラー」**タブを使用します。「データ・ハンドラー」タブの値のフィールドは、ContainerManagedEvents を JMS に設定した場合にのみ表示されます。

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

ControllerStoreAndForwardMode

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

JMS

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

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

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

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

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

EnableOidForFlowMonitoring

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

デフォルト値は false です。

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。

デフォルト値は 128M です。

JvmMaxNativeStackSize

エージェントの最大ネイティブ・スタック・サイズ (キロバイト単位)。

デフォルト値は 128K です。

JvmMinHeapSize

エージェントの最小ヒープ・サイズ (メガバイト単位)。

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、以下のように説明されます。

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

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

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

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

LogAtInterchangeEnd

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

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

MaxEventCapacity

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

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

MessageFileName

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

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

デフォルト値は false です。

OADMaxNumRetry

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

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

デフォルト値は 10000 です。

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

PollQuantity

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

PollStartTime

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

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

RequestQueue

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

ResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

JMS 応答キューを指定します。JMS 応答キューは、応答メッセージをコネクター・フレームワークから統合ブローカーへデリバリーします。WebSphere InterChange Server Express は、要求を送信した後、JMS 応答キューで応答メッセージを待機します。

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、103 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。設定値は CwBO です。

付録 B. Connector Configurator Express

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

この付録では、次のトピックについて説明します。

Connector Configurator Express の概要

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

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **コネクター構成ファイル**を作成します。インストールするコネクターごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティーを設定する。
  場合によっては、コネクター・テンプレートでプロパティーに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、コラボレーションとともに使用するマップを指定し、必要に応じてメッセージング、ロギングとトレース、およびデータ・ハンドラーに関するパラメーターを指定する必要があります。

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

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、113 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

注: Connector Configurator Express は、Windows 環境でのみ実行できます。このコネクターを別の環境で実行する場合は、Windows で Connector Configurator Express を使用して構成ファイルを変更した後、その別の環境に、変更したファイルをコピーします。

Connector Configurator Express の始動

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

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

スタンドアロン・モードでの Configurator Express の実行

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

これを行うには、以下のステップを実行します。

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

Connector Configurator Express を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存する方法が便利です (118 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator Express の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーでいずれかの構成ファイルを選択し、右クリックします。

2. 「標準のプロパティー」タブをクリックし、この構成ファイルに含まれているプロパティーを確認します。

コネクター固有のプロパティー・テンプレートの作成

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

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

- テンプレートの新規作成については、113 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でこのテンプレートを保管します。

新規テンプレートの作成

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

テンプレートは以下のように作成します。

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

     このテンプレートが使用されるコネクター (またはコネクターのタイプ) を表す固有の名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
   - **「旧テンプレート」**、**「変更する既存のテンプレートを選択してください」**

     「テンプレート名」表示に、現在使用可能なすべてのテンプレートの名前が表示されます。
   - テンプレートに含まれているコネクター固有のプロパティー定義を調べるには、**「テンプレート名」**表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティー定義のリストが**「テンプレートのプレビュー」**表示に表示されます。テンプレートを作成するときには、ご使用のコネクターに必要なプロパティー定義に類似したプロパティー定義が含まれている既存のテンプレートを使用できます。
3. **「テンプレート名」**表示からテンプレートを選択し、その名前を**「名前の検索」**フィールドに入力し (または**「テンプレート名」**で自分の選択項目を強調表示し)、**「次へ」**をクリックします。

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

一般特性の指定

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

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

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

値の指定

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

1. **「値」**タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. **「プロパティーを編集」**表示でプロパティーの名前を選択します。
3. **「最大長」**および**「最大複数値」**のフィールドで、変更を行います。次のステップで説明するように、プロパティーの**「プロパティー値」**ダイアログ・ボックスを開かない限り、そのプロパティーの変更内容は受け入れられませんので、注意してください。
4. 値テーブルの左上の隅にあるボックスを右マウス・ボタンでクリックしてから、**「追加」**をクリックします。**「プロパティー値」**ダイアログ・ボックスが表示されます。このダイアログ・ボックスではプロパティーのタイプに応じて、値だけを入力できる場合と、値と範囲の両方を入力できる場合があります。適切な値または範囲を入力し、**「OK」**をクリックします。
5. **「値」**パネルが最新表示され、**「最大長」**および**「最大複数値」**で行った変更が表示されます。以下のような 3 つの列があるテーブルが表示されます。

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

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

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

   値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。テーブルにある既存の値の変更を行うには、その行の行番号

をクリックして行全体を選択します。次に**「値」**フィールドを右マウス・ボタンでクリックし、**「値の編集 (Edit Value)」**をクリックします。

依存関係の設定

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

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

1. **「使用可能なプロパティー」**表示で、依存プロパティーとして指定するプロパティーを選択します。
2. **「プロパティーを選択」**フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティーを選択します。
3. **「条件演算子」**フィールドで以下のいずれかを選択します。

   == (等しい)

   != (等しくない)

   > (より大)

   < (より小)

   >= (より大か等しい)

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

新規構成ファイルの作成

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

コネクター固有のテンプレートからの構成ファイルの作成

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

1. **「ファイル」>「新規」>「コネクター構成」**をクリックします。

2. 以下のフィールドを含む**「新規コネクター」**ダイアログ・ボックスが表示されます。
   - **名前**

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

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

     デフォルトのブローカーは ICS です。この値は変更できません。
   - **コネクター固有プロパティー・テンプレートを選択**

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

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

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

既存ファイルの使用

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

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

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

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

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

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

構成ファイルの完成

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

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

- 119 ページの『標準コネクター・プロパティーの設定』
- 119 ページの『アプリケーション固有の構成プロパティーの設定』
- 120 ページの『サポートされるビジネス・オブジェクト定義の指定』
- 122 ページの『関連付けられたマップ』
- 124 ページの『トレース/ログ・ファイル値の設定』

注: コネクターが JMS メッセージングを使用するものである場合、データをビジネス・オブジェクトに変換するデータ・ハンドラーを構成できるように、追加のカテゴリーが表示されることがあります。詳細については、 124 ページの『データ・ハンドラー』を参照してください。

構成ファイル・プロパティーの設定

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

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

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

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

標準コネクター・プロパティーの設定

標準のプロパティーの値を変更するには、以下のステップを実行します。

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

アプリケーション固有の構成プロパティーの設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティーを追加するときは**「追加」**をクリックします。子プロパティーを追加するには、親の行番号で右マウス・ボタンをクリックし、**「子を追加」**をクリックします。
2. プロパティーまたは子プロパティーの値を入力します。
3. プロパティーを暗号化するには、**「暗号化」**ボックスを選択します。

4. 119 ページの『標準コネクター・プロパティーの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクター・プロパティーの暗号化

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

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

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

更新メソッド

付録 A『*コネクターの標準構成プロパティー*』の 97 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

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

サポートされるビジネス・オブジェクト定義の指定

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

サポートされるビジネス・オブジェクトを指定するときには、指定するビジネス・オブジェクトとそのオブジェクトに対応するマップが、システムに存在していなければなりません。ビジネス・オブジェクト定義 (データ・ハンドラー・メタオブジェクトのビジネス・オブジェクト定義を含みます) とマップ定義は、統合コンポー

ネント・ライブラリー (ICL) プロジェクトに保管されている必要があります。ICL プロジェクトの詳細については、「*WebSphere Business Integration Server Express ユーザーズ・ガイド*」を参照してください。

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

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

ビジネス・オブジェクト名

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

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

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

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

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

エージェント・サポート

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

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

ビジネス・オブジェクトがコネクター・エージェントによってサポートされるよう指定するには、**「エージェント・サポート」**ボックスにチェックマークを付けます。「Connector Configurator Express」ウィンドウでは、「エージェント・サポート」を選択しても問題ないかどうかの検証は行われません。

最大トランザクション・レベル

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

ほとんどのコネクターの場合、選択可能な項目は「最大限の努力」のみです。

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

関連付けられたマップ

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

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

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

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

  以下のをステップを実行して、マップを明示的にバインドします。

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

リソース

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

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下のステップを実行します。

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

     注: STDOUT オプションは、Windows プラットフォームで実行しているコネクターの**「トレース/ログ・ファイル」**タブでのみ使用できます。
   - ファイルに:
     ログ・メッセージまたはトレース・メッセージを指定されたファイルに書き込みます。ファイルを指定するには、ディレクトリー・ボタン (省略符号) をクリックし、指定する格納場所に移動し、ファイル名を指定し、**「保管」**をクリックします。(ご使用のコネクターが、Connector Configurator Express をインストールした Windows プラットフォーム上で稼動していない場合は、まず、ファイルを格納するシステム上の場所にドライブをマップする必要があります。) ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。このタブは、アダプターが保証付きイベント・デリバリーを利用するものである場合に使用可能になります。

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

構成ファイルの保管

構成ファイルの作成とそのファイルに含まれるプロパティーの設定が完了したら、使用するコネクターに応じた適切な場所にそのファイルを配置する必要があります。ICL プロジェクトに構成を保管し、保管されたファイルを System Manager から InterChange Server Express へロードしてください。

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

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- 指定したディレクトリーに保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。

System Manager でのプロジェクトの使用方法と、配置の詳細については、「*User Guide for IBM WebSphere Business Integration Server Express*」を参照してください。

構成の完了

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

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

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

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

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

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

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

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

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

```
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
                                <Value>it_IT</Value>
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>
                        <DefaultValue>en_US</DefaultValue>
                </ValidValues>
        </Property>
```

付録 C. SWIFT メッセージ構造

この付録では、SWIFT メッセージ構造について説明します。

SWIFT メッセージ・タイプ

SWIFT メッセージは、3 つのヘッダー、メッセージの内容、トレーラーを含む、5 つのデータ・ブロックからなります。メッセージ・タイプは、内容を識別する上で重要です。

すべての SWIFT メッセージにリテラル「MT」(メッセージ・タイプ) が含まれています。これに、メッセージのタイプ、カテゴリー、グループを示す 3 桁の数値が続きます。次の例は、サード・パーティーを介した購買または販売のオーダーです。

MT502

最初の桁 (5) はカテゴリーを表します。カテゴリーは、Precious Metals、Syndications、または Travelers Checks など、特定の金融証書やサービスに関連するメッセージを示します。5 で表されるカテゴリーは、Securities Markets です。

2 番目の桁 (0) は、トランザクションのライフ・サイクル内の関連パーツのグループを表します。0 で示されるグループは、Financial Institution Transfer です。

3 番目の桁 (2) は、特定のメッセージを表すタイプです。カテゴリー全体で数百のメッセージ・タイプがあります。2 で表されるタイプは Third-Party Transfer です。

各メッセージに、固有 ID が割り当てられます。ユーザーがログインするたびに 4 桁のセッション番号が割り当てられます。次に、各メッセージに 6 桁のシーケンス番号が割り当てられます。これらが結合されて、ユーザーのコンピューターから SWIFT への ISN (Input Sequence Number)、または SWIFT からユーザーのコンピューターへの OSN (Output Sequence Number) になります。用語は常に、ユーザーではなく SWIFT の視点からのものであることに注意してください。

Logical Terminal Address (12 文字 BIC)、Day、Session および Sequence numbers が結合され、それぞれ MIR (Message Input Reference) および MOR (Message Output Reference) になります。

SWIFT メッセージ・タイプの完全なリストは、「*All Things SWIFT: the SWIFT User Handbook*」を参照してください。

SWIFT フィールド構造

このセクションでは、SWIFT フィールド構造について説明します。フィールドはメッセージ・ブロック A の論理サブディビジョンです。これは、開始フィールド・タグと区切り文字を持つコンポーネントのシーケンスからなります。

フィールドの前には常に、2 桁のフィールド・タグが付きます。このタグの後には、オプションで英字が続きます。英字は、オプションと見なされます。例えば、16R は、ブロックの開始を示すオプション (R) を持つタグ (16) です。16S は、ブロックの終わりを示すオプション (S) を持つタグ (16) です。フィールドは常にフィールド区切り文字で終了します。区切り文字は、メッセージ・ブロックで使用されるフィールドのタイプによって変わります。

SWIFT メッセージで使用されるフィールドには、汎用と非汎用の 2 つのタイプがあります。SWIFT メッセージ・ブロックで使用されるフィールドのタイプは、メッセージ・タイプによって決まります。次に、これらの SWIFT フィールド構造について説明します。汎用フィールドと非汎用フィールドの詳細およびそれらを区別する方法については、「*SWIFT User Handbook*」の パート III の第 3 章を参照してください。

注: 以下に示されている記号 CRLF は制御文字であり、復帰文字/改行文字を表します (ASCII 16 進数では 0D0A、EBCDIC 16 進数では 0D25)。

非汎用フィールド

SWIFT メッセージ・ブロック内の非汎用フィールドの構造は、次のようになります。

:2!n[1a]: *data content*<CRLF>

ここで、以下のように説明されます。

: = 必須コロン

2!n= 数字、固定長

[1a] = オプションの英字、文字オプション

: = 必須コロン

data content = タグごとに個別に定義されるデータ内容

<CRLF> = フィールド区切り文字

次に、非汎用フィールドの例を示します。

:20:1234<CRLF> :32A:...<CRLF>

注: 場合によっては (タグ 15A...*n* がある場合など)、データ内容はオプションです。

汎用フィールド

SWIFT メッセージ内の汎用フィールドの構造は、次のようになります。

`:2!n1a::4!c'/'[8c]'/'`*data content*

ここで、以下のように説明されます。

`:2!n1a:` = `1a` が必須であることを除き、非汎用フィールドと同じフォーマット

`:=` 2 番目の必須コロン (すべての汎用フィールドに必要)

`4!c` = 修飾子

`'/'` = 最初の区切り文字

`[8c]` = 発行者コードまたは Data Source Scheme (DSS)

`'/'` = 2 番目の区切り文字

data content = フォーマット定義については、「*SWIFT User Handbook*」のパート III の第 3 章を参照してください。

注: 非汎用フィールドと汎用フィールドが同じフィールド・タグ文字オプション文字を共有することはできません。これらを容易に区別できるように、コロンが列 Component Sequence の最初の文字として定義されています。汎用フィールドは、非汎用フィールドと同じセクションで定義されます (「*SWIFT User Handbook*」のパート III の第 3 章)。

汎用フィールドのデータ内容には次の文字制限が適用されます。

- データ内容の 2 行目以降は、区切り文字 `CRLF` で開始する必要があります。
- データ内容の 2 行目以降をコロン (:) またはハイフン (-) で開始することはできません。
- データ内容は、区切り文字 `CRLF` で終了する必要があります。

SWIFT メッセージのブロック構造

コネクターは、SWIFT Financial Application (FIN) メッセージをサポートしています。これらの構造は次のとおりです。

{1: *Basic Header Block*} {2: *Application Header Block*} {3: *User Header Block*} {4: *Text Block or body*} {5: *Trailer Block*}

これら 5 つの SWIFT メッセージ・ブロックには、ヘッダー情報、メッセージの本文、およびトレーラーが含まれています。どのブロックも基本フォーマットは同じです。

{*n*:...}

中括弧 ({}) はブロックの開始と終了を示します。*n* はブロック ID です。この場合は、1 から 5 までの単一の整数です。各ブロック ID は、メッセージの特定の部分と関連付けられています。ブロック間に復帰文字や改行文字 (CRLF) はありません。

ブロック 3、4、および 5 には、フィールド・タグで区切られたサブブロックまたはフィールドが含まれる場合があります。ブロック 3 はオプションです。ただし、多くのアプリケーションは、SWIFT が確認通知を戻したときに調整に使用できるよう、ブロック 3 に参照番号を組み込んでいます。

注: SWIFT メッセージ・ブロックの詳細については、「*SWIFT User Handbook FIN System Messages Document*」の第 2 章を参照してください。

{1: Basic Header Block}

基本ヘッダー・ブロックは固定長であり、フィールド区切り文字なしで連続します。これは、次のような形式になります。

```
{1:    F    01   BANKBEBB   2222   123456}
(a)   (b)  (c)     (d)      (e)      (f)
```

a) 1: = ブロック ID (常に 1)

b) アプリケーション ID は以下のとおりです。

- F = FIN (金融アプリケーション)
- A = GPA (汎用アプリケーション)
- L = GPA (ログインなどに使用します)

c) サービス ID は以下のとおりです。

- 01 = FIN/GPA
- 21 = ACK/NAK

d) BANKBEBB = 論理端末 (LT) アドレス。これは 12 文字に固定されており、位置 9 に X を指定することはできません。

e) 2222 = セッション番号。これはユーザーのコンピューターによって生成され、ゼロが埋め込まれます。

f) 123456= ユーザーのコンピューターによって生成されるシーケンス番号。これには、ゼロが埋め込まれます。

{2: Application Header Block}

アプリケーション・ヘッダーには、入力と出力の 2 つのタイプがあります。どちらも固定長であり、フィールド区切り文字なしで連続します。

入力 (SWIFT への) 構造は次のとおりです。

```
{2:    I     100    BANKDEFFXXXX    U       3       003}
(a)   (b)    (c)      (d)          (e)     (f)      (g)
```

a) 2: = ブロック ID (常に 2)

b) I = 入力

c) 100 = メッセージ・タイプ

d) BANKDEFFXXXX = 位置 9 に X を持つ受信側のアドレス。枝が必要ない場合は、X が埋め込まれます。

e) U = 次のようなメッセージ優先順位

- S = システム
- N = 標準

- U = 緊急

f) 3 = デリバリー・モニター・フィールドは次のとおりです。
- 1 = デリバリー警告なし (MT010)
- 2 = デリバリー通知 (MT011)
- 3 = どちらも有効 = U1 または U3、N2 または N

g) 003 = 陳腐化期間。これは、次のように、ノンデリバリー通知をいつ生成するかを指定します。
- U に有効 = 003 (15 分)
- N に有効 = 020 (100 分)

出力 (SWIFT からの) 構造は次のとおりです。

```
{2:   O      100   1200   970103BANKBEBBAXXX2222123456   970103  1201   N}
(a)  (b)      (c)   (d)           (e)                      (f)    (g)   (h)
```

a) 2: = ブロック ID (常に 2)

b) O = 出力

c) 100 = メッセージ・タイプ

d) 1200 = 送信側の入力時間

e) 入力日付や送信側のアドレスを含む Message Input Reference (MIR)

f) 970103 = 受信側の出力日付

g) 1201 = 受信側の出力時間

h) N = 次のようなメッセージ優先順位
- S = システム
- N = 標準
- U = 緊急

{3: User Header Block}

これはオプションのブロックであり、構造は次のとおりです。

```
{3:  {113:xxxx}  {108:abcdefgh12345678}     }
(a)      (b)             (c)
```

a) 3: = ブロック ID (常に 3)

b) 113:xxxx = オプションの銀行用優先順位コード

c) これは、アクセス・ポイントが ACK の調整に使用する Message User Reference (MUR) です。

注: このブロックには、その他のタグもあります。これには、ISITC (Industry Standardization for Institutional Trade Communication) によって定められた、メッセージの本文を検証するための追加のコード・ワードとフォーマット規則を必要とするタグ (119 など。これには、MT521 にコード ISITC が含まれている場合もあります) も含まれます。詳細については、「*All Things SWIFT: the SWIFT User Handbook*」を参照してください。

{4: Text Block or body}

このブロックでは実際のメッセージ内容が指定されます。このブロックは、ほとんどのユーザーに示されます。通常、その他のブロックは表示前に除去されます。次のように、フォーマットは可変長で、フィールド区切り文字として CRLF を必要とします。

```
{4:CRLF
:20:PAYREFTB54302 CRLF
:32A:970103BEF1000000,CRLF
:50:CUSTOMER NAME CRLF
AND ADDRESS CRLF
:59:/123-456-789 CRLF
BENEFICIARY NAME CRLF
AND ADDRESS CRLF
-}
```

ブロック 4 では、記号 CRLF は必須の区切り文字です。

上記の例は、必須フィールドだけが入力されたタイプ MT100 (Customer Transfer) です。これは、ISO7775 メッセージ構造のフォーマットの例です。ブロック 4 フィールドは、「*SWIFT User Handbook*」の該当するボリュームでメッセージ・タイプに対して指定された順序である必要があります。

注: ISO7775 メッセージ規格は、新しいデータ・ディクショナリー規格 ISO15022 に徐々に置き換えられています。特に、新しいメッセージ規格では、SWIFT メッセージ構造のブロック 4 で汎用フィールドが可能になっています。詳細については、128 ページの『SWIFT フィールド構造』を参照してください。

テキスト・ブロック化の内容は、フィールドの集合です。SWIFT フィールドの詳細については、128 ページの『SWIFT フィールド構造』を参照してください。フィールドが論理的にシーケンスにグループ化される場合もあります。シーケンスは、必須またはオプションであり、繰り返しが可能です。シーケンスをサブシーケンスに分割することもできます。さらに、単一のフィールドおよび連続するフィールドのグループを繰り返すこともできます。例えば、SWIFT Tag 16R および 16S 内のシーケンスなどは、開始フィールドと終了フィールドを持つ場合があります。Tag 15 など、その他のシーケンスは、開始フィールドだけを持ちます。さらにその他のメッセージ・タイプでは、フィールド・シーケンスの開始や終了を示す特定のタグはありません。

ブロック 4 フィールドのタグのフォーマットは次のとおりです。

:*nna*:

nn = 番号

a = オプションの文字 (選択されたタグで表示される場合があります)

例えば、次のようになります。

:20: = トランザクション参照番号

:58A: = 受益者の銀行

フィールドの長さは次のとおりです。

nn = 最大長

nn! = 固定長

nn-*nn* = 最小長と最大長

nn * *nn* = 最大行数と行の最大長を掛けた数値

データのフォーマットは次のとおりです。

n = 桁

d = 10 進数のコンマを持つ桁

h = 大文字の 16 進数

a = 大文字

c = 大文字の英数字

e = スペース

x = SWIFT 文字セット

y = 大文字レベル A ISO 9735 文字

z = SWIFT 拡張文字セット

オプションとして定義されているフィールドもあります。メッセージでフィールドをブランクにすることはできないので、特定のメッセージでオプションのフィールドが必要ない場合は、それらを組み込まないでください。

/,word = 「そのままの」文字

[...] = 大括弧はオプションのエレメントを示します。

例えば、次のようになります。

4!c[/30x] これは、固定された 4 つの大文字の英数字であり、オプションでスラッシュと最大 30 の SWIFT 文字が続きます。

ISIN1!e12!c これは、コード・ワードであり、スペースと 12 の固定された大文字の英数字が後に続きます。

注: メッセージ・タイプによっては、特定のフィールドが条件付きとして定義されています。例えば、特定のフィールドが存在する場合に、別のフィールドがオプションから必須または禁止に変わることがあります。特定のフィールドにサブフィールドが含まれている場合もあります。その場合、サブフィールド間に CRLF はありません。検証はサポートされていません。

選択されたオプションに応じて異なるフォーマットを持つフィールドもあります。オプションは、次のように、タグ番号の後の文字によって指定されます。

:32A:000718GBP1000000,00 バリュー・デート、ISO 通貨、および数量

:32B:GBP1000000,00 ISO 通貨および数量

注: 数量のフォーマットに関する SWIFT 規格は、次のとおりです。3 桁ごとの区切りは使用できません (10,000 は使用できませんが、10000 は使用できます)。小数点には実際の小数点ではなくコンマを使用します (1000,45 = 千と百分の四十五)。

:58A:NWBKGB2L	受益者の SWIFT アドレス
:58D:NatWest Bank Head Office London	受益者の氏名と住所

{5: Trailer Block}

メッセージは常に次の形式のトレーラーで終了します。

```
{5: {MAC:12345678}{CHK:123456789ABC}
```

このブロックは SWIFT システムで使用するためのものであり、次のようなキーワードで示されるいくつかのフィールドを含みます。

MAC 宛先との間で交換されるキーと秘密のアルゴリズムを使用して、メッセージの内容全体に基づいて計算されるメッセージ確認コード。メッセージ・カテゴリー 1、2、4、5、7、8、ほとんどの 6s および 304 で検出されます。

CHK すべてのメッセージ・タイプに対して計算されるチェックサム。

PDE 同じメッセージが以前に送信されたとユーザーが考えた場合に追加される重複送信の可能性。

DLM 緊急メッセージ (U) が 15 分以内に配信されなかった場合や、標準メッセージ (N) が 100 分以内にデリバリーされなかった場合に SWIFT によって追加されます。

付録 D. 特記事項

特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

著作権使用許諾

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

経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

プログラミング・インターフェース情報

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

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。

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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

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

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12