IBM WebSphere Business Integration
Adapters

Adapter for Healthcare Data Protocols ユーザーズ・ガイド

バージョン 1.1.0

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、 143 ページの『付録 E. 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 Integration Adapter for Healthcare Protocols バージョン 1.1.0 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： IBM WebSphere Business Integration Adapters
Adapter for Healthcare Data Protocols User Guide
Version 1.1.0

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 2004.1

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

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

目次

本書について

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

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

対象読者

本書は、お客様のサイトで WebSphere Business Integration システムのサポートおよび管理を担当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 システム (InterChange Server を統合ブローカーとしてご使用の場合)
- WebSphere MQ Integrator Broker (WebSphere MQ Integrator Broker を統合ブローカーとしてご使用の場合)
- ビジネス・オブジェクト開発
- WebSphere MQ アプリケーション
- Healthcare データ・プロトコル

関連文書

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

本資料では、「システム・インストール・ガイド (Windows 版)」または「システム・インストール・ガイド (UNIX 版)」および「WebSphere InterChange Server インプリメンテーション・ガイド」への参照が数多く含まれています。本書を印刷する場合には、これらの資料も印刷すると便利です。

以下のサイトから、資料をインストールすることができます。

- アダプターの一般情報が必要な場合、アダプターを WebSphere Message Broker (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere

Business Integration Message Broker) とともに使用する場合、およびアダプターを WebSphere Application Server とともに使用する場合は、以下のサイトを参照してください。

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

- アダプターを InterChange Server とともに使用する場合は、以下のサイトを参照してください。

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

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

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

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

- WebSphere Application Server の詳細については、以下を参照してください。

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

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

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、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。
青い文字	オンラインで表示したときにのみ見られる青の部分は、相互参照用のハイパーリンクです。青い文字ストリングをクリックすることにより、参照先オブジェクトに飛ぶことができます。
ProductDir	製品ファミリーは WBIA です。これは、IBM WebSphere Business Integration Adapters 製品がインストールされているディレクトリーを示します。CROSSWORLDS 環境変数には、ProductDir のディレクトリー・パスが含まれます。このパスは、デフォルトでは IBM¥WebSphereAdapters です。
{ }	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。
< >	命名規則により、1 つの名前の各要素を個々に判別できるようにするために、不等号括弧で囲みます。例えば、`<server_name><connector_name>tmp.log` のように使用します。

/、¥	本書では、ディレクトリー・パスに円記号 (¥) を使用します。UNIX システムの場合には、円記号をスラッシュ (/) に置き換えてください。すべての製品のパス名は、使用システムに製品をインストールしたディレクトリーを基準とした相対パス名です。
%*文字列*% および $*文字列*	パーセント (%) 記号で囲まれた文字列は、Windows *文字列* のシステム変数またはユーザー変数を示します。UNIX 環境での同等表記は、$*文字列*です。*文字列*の値は、UNIX 環境変数を示します。

本リリースの新機能

リリース 1.1 の新機能

2003 年 12 月更新。このアダプターには、以下の要素が含まれます。

- HP-UX のサポート。
- 繰り返し HL7 データ・エレメントのマッピングに使用するビジネス・オブジェクト・ラッパー (各繰り返しデータ・エレメントをそれぞれプリミティブ・ビジネス・オブジェクト属性にマッピングする代わりに使用)。
- HL7 データ・ハンドラーでの拡張長さセグメント用長さ ID のサポート。
- ビジネス・オブジェクト抽出ユーティリティー。これは、業界固有ビジネス・オブジェクトのマスター・ファイルに対して実行するバッチ・ファイルです。このユーティリティーは、業界固有ビジネス・オブジェクトのサブセットを抽出できるので、アプリケーション固有のビジネス・オブジェクトを構成するための新しい手段として利用できます。
- アダプターのインストールに関する情報が本書から除去されました。この情報の新たな入手先については、第 2 章を参照してください。

リリース 1.0 の新機能

リリース 1.0 は、本製品の最初のリリースです。

第 1 章 概要

Healthcare データ・ハンドラー用のコネクターの Healthcare Level Seven (HL7) および National Council for Drug Control programs (NCPDP) メッセージ規格は、WebSphere Business Integration Adapter for Healthcare Data Protocols のランタイム・コンポーネントの 1 つです。HL7 および NCPDP メッセージ規格は両方とも、明確な他の規格 (さまざまな ISO 規格や ANSI 規格など) に言及しています。このコネクターを使用すると、WebSphere 統合ブローカーと、Healthcare 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。

コネクターは、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・フレームワークで構成されます。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用に調整されたコードが含まれています。コネクター・フレームワークのコードはすべてのコネクターに共通なので、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仲介役の機能を果たします。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

本書には、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・フレームワークに関する情報が含まれます。本書では、これらのコンポーネントをどちらも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、「IBM WebSphere InterChange Server システム管理ガイド」または「IBM WebSphere Business Integration Adapters WebSphere MQ Integrator Broker 用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 アダプターは、統合ブローカーと連携して動作します。Healthcare データ・プロトコル対応コネクターは、WebSphere InterChange Server (ICS) および WebSphere MQ Integrator Broker 統合ブローカーとともに動作します。

Healthcare データ・プロトコル対応コネクターを使用すると、IBM InterChange Server または IBM WebSphere MQ Integrator Broker と、HL7/NCPDP 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の間で、ビジネス・オブジェクトを交換できます。

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・プロセスと、データの変更が発生したときに Healthcare (HL7/NCPDP) データ・プロトコル・メッセージを送受信するアプリケーションとの間で、非同期的にビジネス・オブジェクトを交換できます。

図 1 に示すように、コネクターは複数のコンポーネントと相互作用しますが、その共通目的は WebSphere ビジネス・オブジェクトと Healthcare メッセージをつなぐ媒体となることです。

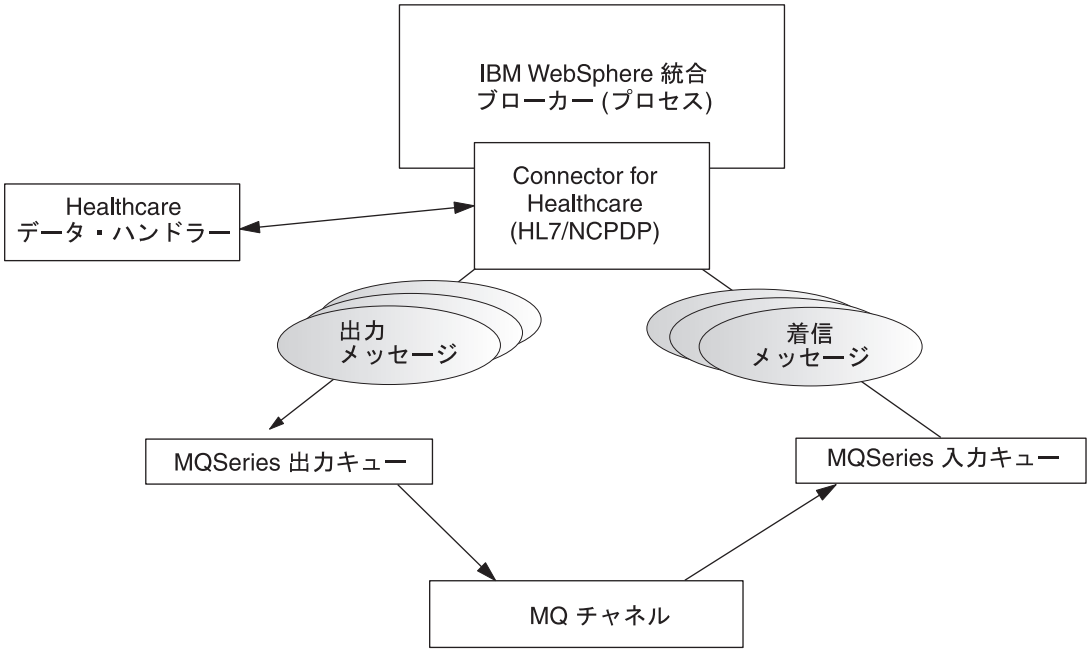

図 1. Healthcare データ・ハンドラーのアーキテクチャー

Healthcare 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 Healthcare

Connector for Healthcare は、メタデータ主導型です。メッセージのルーティングおよびフォーマット変換は、イベント・ポーリング技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、Healthcare 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、コネクターは統合ブローカーからビジネス・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 Healthcare 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。

オプションで動的メタオブジェクトを構築し、コネクターに渡されるビジネス・オブジェクトの子として追加することもできます。この子メタオブジェクトの値は、コネクター全体に対して指定されている静的メタオブジェクトの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、または子メタオブジェクトが必要な変換プロパティーを定義していない場合、デフォルトでは、コネクターは静的メタオブジェクトの値を調べます。1 つの静的コネクター・メタオブジェクトの代わりに、またはその補足として、1 つ以上の動的子メタオブジェクトを指定できます。

コネクターは複数の入力キューをポーリングできます。その際、各入力キューをラウンドロビン方式でポーリングし、各入力キューからメッセージを検索します (検索するメッセージの数は構成可能です)。コネクターは、ポーリング中に検索された各メッセージに、動的子メタオブジェクト (ビジネス・オブジェクトで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、メッセージのフォーマットおよび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に指示できます。

入力キューからメッセージが検索されると、コネクターは、FORMAT テキスト・フィールドに関連付けられているビジネス・オブジェクト名を調べます。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統合ブローカーにデリバリーします。

Healthcare データ・ハンドラー

コネクターは Heathcare 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 HL7/NCPDP メッセージに (あるいは HL7/NCPDP メッセージをビジネス・オブジェクトに) 変換します。

WebSphere MQ

Healthcare データ・ハンドラー用コネクターは、JavaTM Message Service (JMS) の MQ インプリメンテーションを使用します。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 API です。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。

MQ チャネル

WebSphere MQ イベント・キューは WebSphere MQ Interface とメッセージを交換します。ソフトウェアは WebSphere MQ メッセージング機能と HL7/NCPDP 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。

アプリケーションとコネクター間の通信方法

コネクターは、IBM WebSphere MQ に実装されている Java Message Service (JMS) を使用して通信します。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。JMS 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的に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、メッセージ要求の通信を示します。

1. コネクター・フレームワークは Healthcare 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。
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、Healthcare ビジネス・オブジェクトを HL7/NCPDP に準拠したメッセージに変換します。
4. コネクターは HL7/NCPDP 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 層は適切な呼び出しを実行してキュー・セッションを開き、メッセージを MQ Series 入力キューに経路指定します。

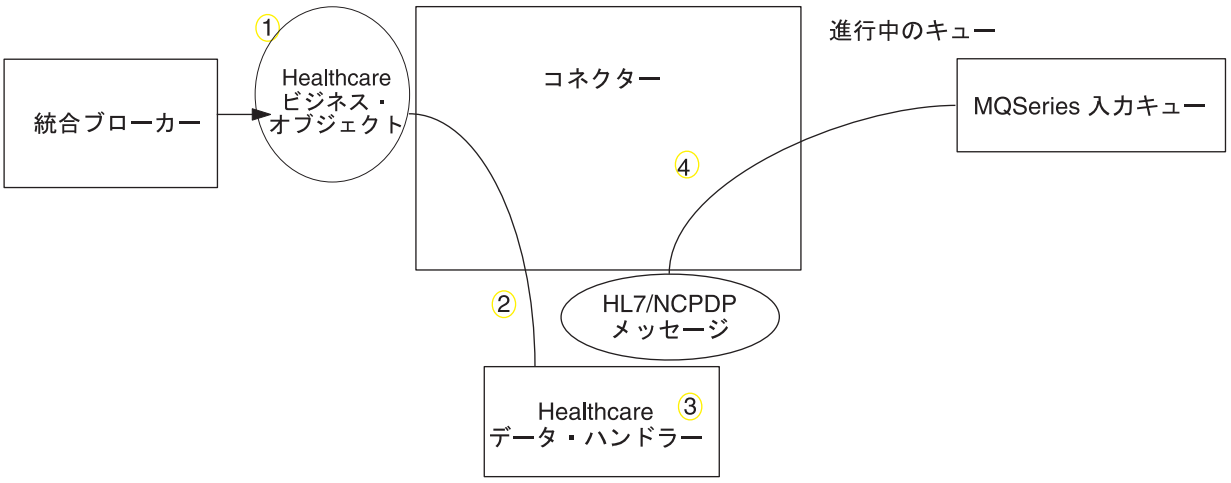

図 2. アプリケーションとコネクターの間の通信方法: メッセージ要求

イベント・デリバリー

図 2 に、メッセージ返送の通信を示します。

1. ポーリング・メソッドが、WebSphere MQ 入力キューから次に適用するメッセージを検索します。
2. 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します。
3. HL7 データ・ハンドラーはビジネス・オブジェクトを受け取り、そのビジネス・オブジェクトの動詞をデータ・ハンドラー固有のメタオブジェクトに指定されたデフォルト動詞に設定します
4. 次に、コネクターは、そのビジネス・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調べます。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。

イベント処理

WebSphere MQ キューにメッセージが格納された場合、イベント通知のために、データベース・トリガーではなくアプリケーションがキューに書き込んだイベントが、コネクターによって検出されます。

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。メッセージを検出すると、コネクターはそれ

を WebSphere MQ 入力キューから検索して調べ、メッセージのフォーマットを判別します。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

コネクターは、最初に WebSphere MQ キューとのトランザクション・セッションを開いて、メッセージを処理します。このトランザクション・アプローチを使用すると、コネクターがビジネス・オブジェクトを正常にサブミットしたにもかかわらず、キューでトランザクションをコミットできなかった場合に、ビジネス・プロセスにビジネス・オブジェクトが 2 回デリバリーされてしまう可能性が若干あります。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のキューに移動します。その結果、メッセージは、処理が完了するまでキュー内に保留されます。処理中にコネクターが予期しないエラーでシャットダウンした場合、メッセージは元の WebSphere MQ キューには復元されず、進行中のキュー内に残されます。

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、キュー上の要求されたすべての処理が、キューからイベントが削除される前に実行され、コミットされる必要があります。したがって、コネクターがキューからメッセージを検索するときには、次の 3 つの処理が実行されるまでは検索がコミットされません。1) メッセージからビジネス・オブジェクトへの変換、2) 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、および 3) 戻り値の受信。

同期確認通知

Connector for Healthcare 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・データを統合ブローカー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、コネクターは統合ブローカー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。コネクターまたは統合ブローカーがビジネス・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 Healthcare 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 Healthcare 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統合ブローカーに同期的に通知し、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
Report	要求されたレポート・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。
メッセージの本文		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統合ブローカーがエラーを検出した場合に、レポートが返送されます。
メッセージの本文		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統合ブローカーから戻されたビジネス・オブジェクトを取り込みます。静的メタデータ・オブジェクト内の DoNotReportBusObj プロパティーを true に設定すると、このデフォルトの振る舞いを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統合ブローカー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

コネクターは初期化の際に進行中のキューを調べ、コネクターのシャットダウンが原因で未処理のまま残っているメッセージがないかどうかを調べます。コネクターの構成プロパティー InDoubtEvents を使用すると、そのようなメッセージのリカバリー処理に関する 4 つのオプション (fail on startup、reprocess、ignore、または log error) のうち、いずれかを指定できます。

fail on startup

fail on startup オプションを指定した場合、コネクターが初期化の際、進行中のキュー内にメッセージを検出すると、コネクターはエラーを記録し、即時にシャットダウンします。ユーザーまたはシステム管理者は、検出されたメッセージを調べ、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などの適切な処置を取る必要があります。

reprocess

reprocessing オプションを指定した場合、コネクターが初期化の際、進行中のキュー内にメッセージを検出すると、コネクターは以降のポーリングでそのメッセージを最初に処理します。進行中のキュー内にあったすべてのメッセージの処理が完了すると、コネクターは入力キューからのメッセージの処理を開始します。

ignore

ignore オプションを指定した場合、初期化の際、コネクターが進行中のキュー内にメッセージを検出すると、コネクターはそれを無視しますが、シャットダウンはしません。

log error

log error オプションを指定した場合、初期化の際、コネクターが進行中のキュー内にメッセージを検出すると、コネクターはエラーを記録しますが、シャットダウンはしません。

アーカイブ

コネクターのプロパティー ArchiveQueue が指定されており、かつ有効なキューを示している場合には、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・キューに格納します。ArchiveQueue が未定義の場合、メッセージは処理後に破棄されます。

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、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 キューとの間で、イベントを失ったり 2 度送信したりせずに、確実に送信することができます。JMS 対応にするためには、コネクター DeliveryTransport 標準プロパティーに JMS を設定する必要があります。このように構成されたコネクターは、JMS トランスポートを使用し、コネクターと統合ブローカーとの間の以降の通信は、すべてこのトランスポートを介して行われます。JMS トランスポートにより、メッセージは最終的に宛先に確実に配送されます。JMS トランスポートの役割は、トランザクション・キュー・セッションが開始されると、コミットが発行されるまでメッセージがキャッシュされるようにすることです。障害が発生するかまたはロールバックが発行されると、メッセージは破棄されます。

注: 保証付きイベント・デリバリー機能を使用しないと、コネクターがイベントをパブリッシュして (コネクターが pollForEvents() メソッド内の gotApplEvent() メソッドを呼び出して) から、イベント・レコードを削除してイベント・ストアを更新する (または「イベント通知済み」状況に更新する) までの間に、障害の可能性を示す短い間が空きます。このすき間で障害が発生すると、イベントは送信されますが、イベント・レコードはイベント・ストアで「進行中」状況のままになっています。コネクターは再始動時に、このイベント・ストアに残されたイベント・レコードを検出して送信するので、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・デリバリー機能を、JMS イベント・ストアあり、またはなしで、JMS 対応コネクターのために構成することができます。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には、26 ページの『保証付きイベント・デリバリーの使用可能化』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 ICS 統合ブローカーに配送できない場合、オブジェクトは (UnsubscribedQueue と ErrorQueue ではなく) FaultQueue に配置されて、状況表示と問題の説明を生成します。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

統合ブローカーがビジネス・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。コネクターは Healthcare データ・ハンドラーを使用し、サブスクリプションのメタオブジェクトに指定された要件に従って、HL7 オブジェクトおよび NCPDP オブジェクトを変換してから発行することができます。

メッセージ処理

コネクターは、統合ブローカーから渡されたビジネス・オブジェクトを、各ビジネス・オブジェクトの動詞に基づいて処理します。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 動詞を持つビジネス・オブジェクトの処理は、ビジネス・オブジェクトが非同期的に送信されたか同期的に送信されたかによって決まります。

非同期デリバリー

これは、Create 動詞を持つビジネス・オブジェクトのデフォルト・デリバリー・モードです。データ・ハンドラーを使用して、ビジネス・オブジェクトからメッセージが作成され、出力キューに書き込まれます。メッセージがデリバリーされた場合、コネクターは BON_SUCCESS を戻します。それ以外の場合は、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、または、処置が行われたかどうかを確認する方法はありません。

同期確認通知

コネクターのプロパティーで `replyToQueue` が定義されており、かつビジネス・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送信します。続いて、コネクターは、受信側のアプリケーションで適切な処置が行われたかどうかを確認するために応答を待ちます。

WebSphere MQ では、コネクターは表 3 に示すようなヘッダーを持つメッセージを最初に発行します。

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	変換プロパティーに定義されている出力フォーマット。IBM の要件に合わせて、8 文字を超える部分が切り捨てられます (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	要求されたレポート・メッセージのオプション	応答メッセージの返送が予測される場合、このフィールドには以下の値が取り込まれます。 処理が成功したときに肯定処理レポートが必要な場合は、MQRO_PAN[a]。 処理が失敗したときに否定処理レポートが必要な場合は、MQRO_NAN[a]。 生成されるレポートの相関 ID が最初に発行された要求のメッセージ ID と同じになる必要がある場合は、MQRO_COPY_MSG_ID_TO_CORREL_ID[a]。
ReplyToQueue	応答キューの名前	応答メッセージの返送が予測される場合、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	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・ヘッダーの後に、メッセージの本文が続きます。メッセージの本文は、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肯定処理レポートと否定処理レポートの両方の返送が予測されることを示すために設定されます。メッセージを発行したスレッドは、受信側アプリケーションが要求を処理できたかどうかを示す応答メッセージを待ちます。

コネクターから同期要求を受け取ると、アプリケーションはビジネス・オブジェクトを処理し、表 4、表 5、および表 6 に示すようなレポート・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	変換プロパティー内で定義された 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 によって定義される定数を示します。

表 5. 応答メッセージへの取り込み

動詞	Feedback フィールド	メッセージの本文
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 システムの応答[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 システムの応答[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	該当なし
MQFB_APPL_FIRST + 6	該当なし
MQFB_APPL_FIRST + 7	UNABLE_TO_LOGIN
MQFB_APPL_FIRST + 8	APP_RESPONSE_TIMEOUT (この応答後、コネクターは即時に終了します)
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応答

[a] 詳細については、「*コネクター開発ガイド*」を参照してください。

ビジネス・オブジェクトを処理できる場合、アプリケーションは、feedback フィールドが MQFB_PAN (または特定の WebSphere Business Integration システムの値) に設定されたレポート・メッセージを作成します。また、オプションで、すべての変更を含む直列化されたビジネス・オブジェクトをメッセージ本文に取り込みます。ビジネス・オブジェクトを処理できない場合、アプリケーションは、feedback フィールドが MQFB_NAN (または特定の WebSphere Business Integration システムの値) に設定されたレポート・メッセージを作成します。オプションで、このレポート・メッセージの本文にエラー・メッセージを含むこともできます。いずれの場合も、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送信します。

コネクターは、応答メッセージを検索すると、デフォルトでは、応答の correlationID を要求メッセージの messageID と突き合わせます。続いて、要求を発行したスレッドに通知を送信します。コネクターは、応答の feedback フィールドの設定によって、メッセージの本文にビジネス・オブジェクトとエラー・メッセージのどちらが含まれているかを予測します。ビジネス・オブジェクトが含まれていると予測したにもかかわらず、メッセージの本文にビジネス・オブジェクトが取り込まれていなかった場合、コネクターは統合ブローカーが Request 操作のために最初に発行したのと同じビジネス・オブジェクトを単純に返送します。エラー・メッセージが含まれていると予測したにもかかわらず、メッセージの本文にエラー・メッセージが取り込まれていなかった場合、統合ブローカーには応答コードと汎用エラー・メッセージが返送されます。ただし、メッセージ選択子を使用して、識別やフィルター操作を行うこともできます。あるいは、アダプターが特定の要求に対して応答メッセージを識別する方法を制御できます。このメッセージ選択子機能は、JMS 機能です。この機能は同期要求処理にのみ適用されます。以下に詳細を説明します。

メッセージ選択子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、同期要求処理用のビジネス・オブジェクトを受け取ると、動詞のアプリケーション固有情報に response_selector ストリングが含まれていないかどうかをチェックします。response_selector が未定義の場合、コネクターは、前述のように、相関 ID を使用して応答メッセージを識別します。

response_selector が定義されていると、コネクターは次の構文に基づく名前と値のペアを探します。

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

メッセージ選択子ストリングは、応答を一意的に識別する必要があります。また、次の例に示すように、値は単一引用符で囲む必要があります。

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

上記の例の場合、アダプターは、要求メッセージを発行した後、「Oshkosh」に等しい 相関 ID を持つ応答メッセージの ReplyToQueue をモニターします。アダプターは、このメッセージ選択子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。

また、オプションで、アダプターによる実行時置換を実行して、各要求ごとに固有のメッセージ選択子を生成することもできます。メッセージ選択子の代わりに、'{1}' のように、整数を中括弧で囲んだ形式でプレースホルダーを指定します。この後にコロンを記入し、置換に使用する属性をコンマで区切ってリストします。プレースホルダー内の整数は、置換時に使用される属性のインデックスとして機能します。次のメッセージ選択子を例に考えてみます。

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

このメッセージ選択子は、アダプター {1} を選択子に続く最初の属性の値 (この例では、子オブジェクト MyDynamicMO の属性 CorrelationId) と置換するように通知します。属性 CorrelationID の値が 123ABC である場合には、アダプターは以下の基準によって作成されたメッセージ選択子を生成し、使用します。

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

これで、応答メッセージが識別されます。

また、次のように、複数の置換を指定することも可能です。

```
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・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 番目の位置にある AddressId の値で置換します。この方法により、応答メッセージ選択子に指定されたビジネス・オブジェクトおよびメタオブジェクト内の、すべての属性を参照することができます。Address[4].AddressId を使用した詳細検索の実行方法について詳しくは、「JCDK API マニュアル」(getAttribute メソッド) を参照してください。

次のいずれかの状況が発生すると、実行時にエラーが報告されます。

- '{}' シンボルの間に非整数の値を指定した場合
- 属性が定義されていないインデックスを指定した場合
- 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

- 属性パスの構文が不正の場合

例えば、メッセージ選択子にリテラル値 '{' または '}' を組み込む場合には、それぞれ '{{' または "{}" を使用できます。また、属性値にこれらの文字を組み込むこともできますが、その場合、最初 の "{" は不要です。エスケープ文字を使用した次の例について考えてみます。`response_selector=JMSCorrelation LIKE '{1}' and CompanyName='A{{P': MyDynamicMO.CorrelationID`

コネクターはこのメッセージ選択子を次のように解決します。

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

コネクターが属性値内で検出した特殊文字 ('{'、'}'、':'、または ';' など) は、照会ストリングに直接挿入されます。このため、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、照会ストリングに組み込むことができます。

次の例は、リテラル・ストリングの置換値が属性値から抽出される方法を示しています。

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

`MyDynamicMO.CorrelationID` に値 `{A:B}C;D` が含まれていると、コネクターはメッセージ選択子を次のように解決します。 `JMSCorrelationID LIKE '{A:B}C;D' and CompanyName='A{P'`

応答選択子コードについて詳しくは、「JMS 1.0.1 仕様書」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、WebSphere Business Integration システム固有のすべての戻り状況値を WebSphere MQ フィードバック・コードにマップした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値は、統合ブローカーに渡されます。詳細については、23 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、この動詞を持つビジネス・オブジェクトを、Create 動詞に対して定義されている同期デリバリーと同様に処理します。ただし、Retrieve 動詞を使用する場合には、`responseTimeout` と `replyToQueue` が必須です。さらに、トランザクションを完了するためにはメッセージの本文に直列化されたビジネス・オブジェクトが取り込まれている必要があります。

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. 応答メッセージへの取り込み

動詞	Feedback フィールド	メッセージの本文
Retrieve	FAIL FAIL_RETRIEVE_BY_CONTENT	(オプション) エラー・メッセージ。

表 7. 応答メッセージへの取り込み (続き)

動詞	Feedback フィールド	メッセージの本文
	MULTIPLE_HITS SUCCESS	直列化されたビジネス・オブジェクト。

エラー処理

コネクターによって生成されたすべてのエラー・メッセージは、BIA_HealthcareConnector.txt という名前のメッセージ・ファイルに格納されます。(このファイルの名前は、コネクター構成標準プロパティー LogFileName によって決定されます。) 各エラーはエラー番号が付けられ、その後にエラー・メッセージが表示されます。

Message number

Message text

コネクターは、以降のセクションで説明する方法で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

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

- コネクターが、メッセージの検索中に JMS サービス・プロバイダーとの接続を確立できない場合。
- コネクターがビジネス・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。
- コネクターが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、戻りコードが APP_RESPONSE_TIMEOUT または UNABLE_TO_LOGIN の応答メッセージを受信した場合。

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付けられているメッセージを検索した場合。
- コネクターが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、アンサブスクライブされたメッセージは破棄されます。

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からビジネス・オブジェクトへの変換に失敗した場合、または、(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デリバリーされます。ErrorQueue が定義されていない場合、エラーが原因で処理できなかったメッセージは破棄されます。

データ・ハンドラーがビジネス・オブジェクトからメッセージへの変換に失敗した場合、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、オンにするとコネクターの動作を詳細にトレースできます。デフォルトでは、トレース・メッセージは 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、17 ページの『第 2 章 コネクターの構成』のコネクター構成プロパティーの説明を参照してください。トレースの使用可能化や設定方法などの詳細については、「*コネクター開発ガイド*」を参照してください。

コネクターのトレース・メッセージに推奨される内容を以下に示します。

レベル 0　コネクターのバージョンを確認するトレース・メッセージに使用します。

レベル 1　処理される各ビジネス・オブジェクトについての重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しいメッセージを検出するたびに記録されるトレース・メッセージに使用します。

レベル 2　ビジネス・オブジェクトが gotApplEvent() または executeCollaboration() から統合ブローカーに送付されるたびに記録されるトレース・メッセージに使用します。

レベル 3　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デリバリーに関する情報を提供するトレース・メッセージに使用します。

レベル 4　コネクターが動作を開始または終了した時間を記録するトレース・メッセージに使用します。

レベル 5　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表すトレース・メッセージ、メッセージが除去されるかキューに送出されるたびに記録されるトレース・メッセージ、または、ビジネス・オブジェクトのダンプを記録するトレース・メッセージに使用します。

第 2 章 コネクターの構成

この章では、コネクターのインストール方法と構成方法、およびコネクターと共に動作するメッセージ・キューの構成方法について説明します。

互換性

アダプターが使用するアダプター・フレームワークは、アダプターと通信する統合ブローカーのバージョンとの互換性を備えている必要があります。Adapter for Healthcare Data Protocols バージョン 1.1 は、以下のアダプター・フレームワークと統合ブローカーでサポートされています。

- **アダプター・フレームワーク**: WebSphere Business Integration Adapter Framework、バージョン 2.1、2.2、2.3.x、および 2.4
- **統合ブローカー:**
  - WebSphere InterChange Server、バージョン 4.2、4.21、および 4.22
  - WebShpere MQ Integrator、バージョン 2.1.0
  - WebSphere MQ Integrator broker、バージョン 2.1.0
  - WebSphere Business Integration Message Broker、バージョン 2.1 および 5.0
  - WebSphere Application Server Enterprise、バージョン 5.0.2 (WebSphere Studio Application Developer Integration Edition、バージョン 5.0.1 と併用)

例外については、「リリース情報」を参照してください。

注:

統合ブローカーのインストールおよびその前提条件に関する説明については、以下の資料を参照してください。

WebSphere InterChange Server (ICS) については、「システム・インストール・ガイド (UNIX 版)」または「システム・インストール・ガイド (Windows 版)」を参照してください。

Message Brokers (WebSphere MQ Integrator Broker、WebSphere MQ Integrator、および WebSphere Business Integration Message Broker) の場合は、「WebSphere Message Brokers 使用アダプター・インプリメンテーション・ガイド」およびそ

れぞれの Message Brokers のインストールに関する資料を参照してください。一部の資料は次の Web サイトにあります。

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

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

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

前提条件

ソフトウェアの前提条件

Adapter for Healthcare Data Protocols をインストールして構成するには、あらかじめ以下のソフトウェアをインストールしておく必要があります。

- Windows システムの場合: Windows 2000

  注: バージョン 1.1 以降、Adapter for Healthcare Data Protocols は Microsoft Windows NT ではサポートされなくなりました。

- UNIX システムの場合: Solaris 7.0 または 8.0、AIX 5.1 または 5.2、あるいは HP-UX11i

アダプターおよび関連ファイルのインストール

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

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

インストール済みファイルの構造

以降のサブセクションでは、コネクターが使用する、UNIX システム上または Windows システム上のファイルの構造について説明します。

UNIX のインストール済みファイルの構造

表 8 に、コネクターが使用する UNIX ファイル構造の説明を示します。

表 8. コネクター用としてインストールされた UNIX ファイル構造

***ProductDir* のサブディレクトリー**	**説明**
`connectors/Healthcare/BIA_CW_Healthcare.jar`	コネクター jar ファイル。

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

***ProductDir* のサブディレクトリー**	**説明**
connectors/Healthcare/start_Healthcare.sh	コネクターの始動スクリプト。このスクリプトは、汎用コネクター管理スクリプトから呼び出されます。「Connector Configurator からインストール (Install from Connector Configurator)」(WebSphere MQ Integrator Broker を統合ブローカーとして使用する場合)、または System Manager の「コネクターの構成 (Connector Configuration)」画面 (ICS を統合ブローカーとして使用する場合) をクリックすると、インストーラーがこのコネクター管理スクリプト用にカスタマイズされたラッパーを作成します。コネクターが ICS と連動する場合、このカスタマイズされたラッパーはコネクターの始動と停止にのみ使用します。コネクターが WebSphere MQ Integrator Broker と連動する場合は、このカスタマイズされたラッパーはコネクターの始動にのみ使用し、コネクターの停止には mqsiremotestopadapter を使用します。
connectors/messages/BIA_HeathcareConnector.txt	コネクター・メッセージ・ファイル。
repository/Healthcare/dependencies/ BIA_HL7MTEventMap.cfg	構成ファイル。
repository/Healthcare/dependencies/ BIA_HL7I18N.cfg	構成ファイル。
connectors/Heathcare/ BIA_CWHealthcareDataHandler.jar	Healthcare データ・ハンドラー
/lib	WBIA.jar ファイルが含まれています。
/bin	CWConnEnv.sh ファイルが含まれています。

Windows のインストール済みファイルの構造

表 9 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。

表 9. 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¥Healthcare¥BIA_CW_Healthcare.jar	コネクター jar ファイル。
connectors¥Healthcare¥start_Healthcare.bat	コネクターの始動ファイル。
connectors¥messages¥BIA_HeathcareConnector.txt	コネクター・メッセージ・ファイル。
connectors¥Healthcare¥BIA_CW_HealthcareDataHandler.jar	Healthcare データ・ハンドラー
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。
repository¥Healthcare/dependencies¥BIA_HL7MTEventMap.cfg	構成ファイル。
repository¥Healthcare¥dependencies¥BIA_HL7I18N.cfg	構成ファイル。
¥lib	WBIA.jar ファイルが含まれています。
¥bin	CWConnEnv.bat ファイルが含まれています。

コネクターの構成

コネクターには、標準構成プロパティーとコネクター固有の構成プロパティーという 2 種類の構成プロパティーがあります。これらのプロパティーの値は、コネクターを実行する前に設定する必要があります。次のいずれかのツールを使用して、コネクターの構成プロパティーを設定してください。

- Connector Configurator (統合ブローカーが ICS の場合) — このツールには System Manager からアクセスします。
- Connector Configurator (統合ブローカーが WebSphere MQ Integrator Broker の場合) — このツールには WebSphere Business Integration Adapter プログラム・フォルダーからアクセスします。詳細については、109 ページの『付録 B. Connector Configurator』 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。これらのプロパティーの詳細については、91 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。

重要: このコネクターは ICS と WebSphere MQ Integrator Broker の両方をサポートしているため、このコネクターには両方のブローカーの構成プロパティーが関係します。

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、コネクター内の静的情報やロジックを、エージェントの再コーディングおよび再ビルドをせずに変更できます。

注: WebSphere MQ から供給される値は、不適切または不明な場合があるため、この値を常に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 10 に、Connector for Heathcare 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 10. コネクター固有の構成プロパティー

名前	指定可能な値	デフォルト値	必要
22 ページの『ApplicationPassword』	*ログイン・パスワード*		いいえ
22 ページの『ApplicationUserID』	*ログイン・ユーザー ID*		いいえ
22 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`	いいえ
22 ページの『BOPref』	*ビジネス・オブジェクト設定*	HL7	はい
22 ページの『Channel』	*MQ サーバー・コネクター・チャネル*		はい
22 ページの『ClassName』	*データ・ハンドラー・クラス名*	`com.ibm.adapters.datahandlers.hl7.HL7DataHandler`	いいえ

表 10. コネクター固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必要
22 ページの『ConfigurationMetaObject』	*構成メタオブジェクトの名前*		はい
22 ページの『Component delimiter』	ユーザー定義	^	
23 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
23 ページの『DataHandlerMimeType』	*MIME* ファイル・タイプ	HL7	いいえ
23 ページの『DefaultVerb』	コネクターがサポートする任意の動詞	Create	
23 ページの『Dummy string』	ユーザー定義		
23 ページの『ErrorQueue』	未処理のメッセージ・キュー	queue://crossworlds.Queue.manager/MQCONN.ERROR	いいえ
24 ページの『Field delimiter』	全タイプ用のデフォルトのフィールド区切り文字。	デフォルト値は「|」です。	
24 ページの『HostName』	*WebSphere MQ* サーバー	デフォルトは「|」です。	いいえ
24 ページの『InDoubtEvents』	FailOnStartup Reprocess Ignore LogError	Reprocess	いいえ
24 ページの『InputQueue』	ポーリング・キュー	queue://CrossWorlds.QueueManager/MQCONN.IN	はい
25 ページの『InProgressQueue』	進行中イベント・キュー	queue://CrossWorlds.QueueManager/MQCONN.IN_PROGRESS	いいえ
25 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
25 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
26 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
26 ページの『Representation』	構文解析するヘッダー・オブジェクトまたはメッセージ・オブジェクト	「Simplified」または「Native」のどちらかです。 Simplified の場合、構文解析メソッドを使用して、メッセージ・ヘッダーにのみ詳細オブジェクトを表示します。 Native の場合、構文解析メソッドを使用して、メッセージ全体に詳細オブジェクトを表示します。	
26 ページの『Repetition delimiter』	ユーザー定義	デフォルト値は「~」です。	
26 ページの『Subcomponent delimiter』	ユーザー定義	デフォルト値は「&」です。	
26 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
26 ページの『UseDefaults』	true または false	false	

ApplicationPassword
WebSphere MQ にログインするときに ApplicationUserID と共に使用するパスワードです。

デフォルト: なし

ApplicationPassword 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 から提供されるデフォルト・パスワードを使用します。

ApplicationUserID
WebSphere MQ にログインするときに ApplicationPassword と共に使用するユーザー ID です。

デフォルト: なし

ApplicationUserID 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 から提供されるデフォルト・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

BOPref
ビジネス・オブジェクトのプリファレンスの接頭部。デフォルトは HL7 です。

Channel
コネクターが WebSphere MQ との通信に使用する 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。

ClassName
メッセージからビジネス・オブジェクトへの変換、およびその逆の変換のときに使用するデータ・ハンドラー・クラスです。

デフォルト: com.ibm.adapters.DataHandlers.hl7.HL7DataHandler

Component delimiter
全フライト用のデフォルトのコンポーネント区切り文字。デフォルトは「^」です。

ConfigurationMetaObject
コネクターの構成情報を格納した静的メタオブジェクトの名前です。

デフォルト: なし

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
特定の MIME タイプに基づくデータ・ハンドラーを要求することができます。

デフォルト: `HL7`

DefaultVerb
ポーリングのときにデータ・ハンドラーによって着信ビジネス・オブジェクト内に動詞が設定されていない場合に設定すべき動詞を指定します。

デフォルト: `Create`

Dummy string
ダミーのストリングです。

ErrorQueue
処理でき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
メッセージの受領を同期的に統合ブローカーに通知するときに使用されるデフォルト・フィードバック・コードをオーバーライドし、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れば、各属性名がフィードバック・コードを表すメタオブジェクトを指定できます。このフィードバック・コードの対応する値は、統合ブローカーに渡される戻り状況です。デフォルト・フィードバック・コードのリストについては、6 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・コードを表す以下の属性値を受け入れます。

- MQFB_APPL_FIRST
- MQFB_APPL_FIRST_OFFSET_*N*。*ここで N は整数を表します (*MQFB_APPL_FIRST+ *N の値と解釈されます)*。

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

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

表 11 にメタオブジェクトの例を示します。

表 11. 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

Field delimiter
全タイプ用のデフォルトの区切り文字。デフォルトは「|」(パイプ記号) です。

HostName
WebSphere MQ のホストとな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
コネクターの予期せぬシャットダウンのために不完全に処理された進行中のイベントをどのように扱うかを指定します。初期化のときに進行中のキューにイベントが見つかった場合に、以下の 4 つのうちのどのアクションを実行するかを選択してください。

- FailOnStartup。 エラーをログに記録して即時にシャットダウンします。
- Reprocess。 残っているイベントを最初に処理してから、入力キュー内のメッセージを処理します。
- Ignore。 進行中のキューにメッセージがあっても無視します。
- LogError。 エラーはログに記録されるが、シャットダウンは行われません。

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。MQSA 文書を参照して、WebSphere MQ キューを構成してください。

コネクターに複数のキュー名を指定する場合はセミコロンで区切ります。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、各キューからは最大で pollQuantity 数のメッセージを検索します。例えば、pollQuantity が 2 で、MyQueueA に 2 つのメッセージ、MyQueueB に 1 つのメッセージ、MyQueueC に 5 つのメッセージがあるとします。

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 (1/2 回

目) では、コネクターは MyQueueA、MyQueueB、および MyQueueC それぞれから 1 つめ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 回ずつポーリングしたら、メソッド pollForEvents の呼び出しは完了です。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 つのメッセージ
2. MyQueueB から 1 つのメッセージ
3. MyQueueC から 1 つのメッセージ
4. MyQueueA から 1 つのメッセージ
5. MyQueueB は空のためスキップ
6. MyQueueC から 1 つのメッセージ

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

InProgressQueue

処理中にメッセージが保留されたメッセージ・キューです。このキューなしでコネクターが動作するように構成するには、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 名を除去します。このように構成すると、コネクターがシャットダウンしてイベントが保留になった場合はイベント・デリバリーに影響する可能性があるという警告が始動時に出されます。

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

I118N

BIA_HL7I18N.cfg ファイルのパス名を指定します。
..¥Healthcare¥dependencies¥hl7¥BIA_HL7I18N.cfg

MTEventMap

この構成には、(メッセージ X イベント・タイプ) -> メッセージ構造のマップが含まれます。ファイルは..¥Healthcare¥dependencies¥hl7¥HL7MTEventMap.cfg にあります。

PollQuantity

pollForEvents のスキャン時に、InputQueue プロパティーに指定された各キューから検索するメッセージ数です。

デフォルト: 1

Port

WebSphere MQ リスナー用に確立するポートです。

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。

Repetition delimiter
全タイプ用のデフォルトの反復区切り文字。デフォルトは「~」です。

ReplyToQueue
コネクターが要求を出したときに応答メッセージがデリバリーされるキューです。

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

Representation
Representation 次第で、メッセージの構文解析方法は「Simplified」または「Native」のどちらかに決定されます。Simplified の場合、メッセージ・ヘッダーのみが表示されるように構文解析されます。Native の場合、メッセージ全体が表示されるように構文解析されます。デフォルトは「Simplified」です。

Subcomponent delimiter
デフォルトのサブコンポーネント区切り文字。デフォルトは「&」です。

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: 全タイプ用のデフォルトのサブコンポーネント区切り文字。
`queue://crossworlds.Queue.manager/MQCONN.UNSUBSCRIBED`

UseDefaults
Create 操作では、UseDefaults が `true` に設定されている場合、コネクターは、isRequired となっている各ビジネス・オブジェクト属性に有効な値またはデフォルト値が指定されているかをチェックします。値が指定されていれば、Create 操作は正常に実行されます。このパラメーターが `false` に設定されている場合、コネクターは有効な値についてのチェックしか行わないため、有効な値が指定されていないと Create 操作は失敗します。デフォルト値は `false` です。

保証付きイベント・デリバリーの使用可能化
JMS 対応コネクターの保証付きイベント・デリバリー機能を構成するには、次のいずれかの方法を行います。

- コネクターが JMS イベント・ストア (JMS ソース・キューとして実装される) を使用する場合は、コネクター・フレームワークが JMS イベント・ストアを管理できます。詳細については、 27 ページの『JMS イベント・ストアを持つコネクターの保証付きイベント・デリバリー』を参照してください。
- コネクターが非 JMS イベント・ストア (例えば JDBC テーブル、E メールのメールボックス、またはフラット・ファイルとして実装される) を使用する場合、コネクター・フレームワークは JMS モニター・キューを使用して、重複イベントが発生しないようにすることができます。詳細については、 29 ページの『非 JMS イベント・ストアを持つコネクターの保証付きイベント・デリバリー』を参照してください。

JMS イベント・ストアを持つコネクターの保証付きイベント・デリバリー

JMS 対応コネクターが JMS キューを使用してイベント・ストアを実装する場合は、コネクター・フレームワークが「コンテナー」となって JMS イベント・ストア (JMS ソース・キュー) を管理することができます。単一の JMS トランザクションで、コネクターはソース・キューからメッセージを取り出し、それを宛先キューに配置できます。このセクションでは、JMS イベント・ストアを持つ JMS 対応コネクターの保証付きイベント・デリバリー機能の使用に関する以下の情報について説明します。

- 『JMS イベント・ストアを持つコネクターの機能の使用可能化』
- 28 ページの『イベント・ポーリングへの影響』

JMS イベント・ストアを持つコネクターの機能の使用可能化

JMS イベント・ストアを持つ JMS 対応コネクターの保証付きイベント・デリバリー機能を使用可能にするには、コネクター構成プロパティーを表 12 に示す値に設定します。

表 12. JMS イベント・ストアを持つコネクターの保証付きイベント・デリバリー・コネクター・プロパティー

コネクター・プロパティー	値
DeliveryTransport	JMS
ContainerManagedEvents	JMS
PollQuantity	イベント・ストアの 1 回のポーリングで処理するイベント数。
SourceQueue	コネクター・フレームワークがポーリングし、処理するイベントを検索する JMS ソース・キュー (イベント・ストア) の名前。 **注:** ソース・キューやその他の JMS キューは同じキュー・マネージャーの一部でなければなりません。コネクターのアプリケーションが別のキュー・マネージャーに保管されるイベントを生成する場合は、リモート・キュー・マネージャー上にリモート・キュー定義を設定してください。これにより WebSphere MQ は、そのイベントをリモート・キューから、JMS 対応コネクターが統合ブローカーへの送信に使用するキュー・マネージャーに転送できます。リモート・キュー定義の構成方法については、IBM WebSphere MQ 文書を参照してください。

コネクターの構成に加えて、JMS ストア内のイベントとビジネス・オブジェクトとの間の変換を行うデータ・ハンドラーも構成する必要があります。このデータ・ハンドラー情報は、表 13 に示すコネクター構成プロパティーから成っています。

表 13. 保証付きイベント・デリバリーのデータ・ハンドラー・プロパティー

データ・ハンドラー・プロパティー	値	必須
MimeType	データ・ハンドラーが処理する MIME タイプ。この MIME タイプによって、呼び出すデータ・ハンドラーを識別します。	はい
DHClass	データ・ハンドラーを実装する Java クラスの絶対パス名。	はい
DataHandlerConfigMOName	MIME タイプとそのデータ・ハンドラーを関連付けるトップレベルのメタオブジェクト名。	オプション

注: データ・ハンドラー構成プロパティーは、他のコネクター構成プロパティーと共に、コネクター構成ファイルに格納されます。

JMS イベント・ストアを持つコネクターについて、保証付きイベント・デリバリーを使用するように構成する場合は、表 12 および 表 13 に従ってコネクター・プロパティーを設定する必要があります。これらのコネクター構成プロパティーを設定するには、Connector Configurator ツールを使用します。Connector Configurator では、表 12 のコネクター・プロパティーは「標準のプロパティー」タブに表示されます。表 13 のコネクター・プロパティーは、「データ・ハンドラー」タブに表示されます。

注: Connector Configurator の「データ・ハンドラー」タブにあるフィールドは、DeliveryTransport コネクター構成プロパティーが JMS に設定され、ContainerManagedEvents が JMS に設定されている場合のみアクティブになります。

Connector Configurator については、 109 ページの『付録 B. Connector Configurator』 を参照してください。

イベント・ポーリングへの影響

ContainedManagedEvents を JMS に設定することによって、コネクターが保証付きイベント・デリバリーを使用するようにした場合、そのコネクターの動作は、この機能を使用しないコネクターとは少し異なります。Container Managed Events を提供するため、コネクター・フレームワークは以下の手順でイベント・ストアをポーリングします。

1. JMS トランザクションを開始します。
2. イベント・ストアから JMS メッセージを読み取ります。

   イベント・ストアは JMS ソース・キューとして実装されます。JMS メッセージにはイベント・レコードが格納されています。JMS ソース・キューの名前は、SourceQueue コネクター構成プロパティーから取得します。
3. データ・ハンドラーを呼び出して、イベントをビジネス・オブジェクトに変換します。

   コネクター・フレームワークは、表 13 に示されたプロパティーが構成済みであるデータ・ハンドラーを呼び出します。

4. 統合ブローカーが WebSphere MQ Integrator Broker である場合は、構成されたワイヤー形式 (XML) に基づいてビジネス・オブジェクトをメッセージに変換します。
5. これにより得られるメッセージを JMS 宛先キューに送信します。
   WebSphere ICS 統合ブローカーを使用している場合は、JMS 宛先キューに送信されるメッセージはビジネス・オブジェクトです。WebSphere MQ Integrator Broker を使用している場合は、JMS 宛先キューに送信されるメッセージは、(データ・ハンドラーが生成した) XML メッセージ です。
6. JMS トランザクションをコミットします。
   JMS トランザクションがコミットされると、同じトランザクションの中でメッセージは JMS 宛先キューに書き込まれ、JMS ソース・キューから除去されます。
7. ステップ 1 から 6 までを 1 ループとして繰り返します。このループを何回繰り返すかは、`PollQuantity` コネクター・プロパティーによって決まります。

重要: `ContainerManagedEvents` プロパティーが `JMS` に設定されているコネクターは、イベント・ポーリングを実行する際に `pollForEvents()` メソッドは呼び出しません。このコネクターの基底クラスに `pollForEvents()` メソッドが含まれていても、このメソッドは呼び出されません。

非 JMS イベント・ストアを持つコネクターの保証付きイベント・デリバリー

JMS 対応コネクターがJMS 以外のソリューションを使用してイベント・ストア (JDBC イベント表、E メールのメールボックス、またはフラット・ファイルなど) を実装する場合、コネクター・フレームワークは Duplicate Event Elimination を使用して、重複イベントが発生しないようにすることができます。このセクションでは、非 JMS イベント・ストアを持つ JMS 対応コネクターの保証付きイベント・デリバリー機能の使用に関する以下の情報について説明します。

- 『非 JMS イベント・ストアを持つコネクターの機能の使用可能化』
- 28 ページの『イベント・ポーリングへの影響』

非 JMS イベント・ストアを持つコネクターの機能の使用可能化: 非 JMS イベント・ストアを持つ JMS 対応コネクターの保証付きイベント・デリバリー機能を使用可能にするには、コネクター構成プロパティーを表 14 に示す値に設定します。

表 14. 非 JMS イベント・ストアを持つコネクターの保証付きイベント・デリバリー・コネクター・プロパティー

コネクター・プロパティー	値
`DeliveryTransport`	`JMS`
`DuplicateEventElimination`	`true`
`MonitorQueue`	JMS モニター・キューの名前。コネクター・フレームワークは、処理済みビジネス・オブジェクトの ObjectEventId をこのキューに保管します。

保証付きイベント・デリバリーを使用するようにコネクターを構成する場合は、表 14 に従ってコネクター・プロパティーを設定する必要があります。これらのコネク

ター構成プロパティーを設定するには、Connector Configurator ツールを使用します。Connector Configurator では、これらのコネクター・プロパティーは「標準のプロパティー」タブに表示されます。Connector Configurator については、109 ページの『付録 B. Connector Configurator』 を参照してください。

イベント・ポーリングへの影響: DuplicateEventElimination を true に設定することによって、コネクターが保証付きイベント・デリバリーを使用するようにした場合、そのコネクターの動作は、この機能を使用しないコネクターとは少し異なります。Duplicate Event Elimination を提供するため、コネクター・フレームワークは JMS モニター・キューを使用してビジネス・オブジェクトを追跡します。JMS モニター・キューの名前は、MonitorQueue コネクター構成プロパティーから取得します。

コネクター・フレームワークは、(pollForEvents() メソッド内の gotApplEvent() を呼び出すことによって) アプリケーション固有のコンポーネントからビジネス・オブジェクトを受信した後、(gotApplEvents() から受信した) 現行のビジネス・オブジェクトが重複イベントを表しているかどうかを判別する必要があります。この判別を行うため、コネクター・フレームワークは JMS モニター・キューからビジネス・オブジェクトを検索し、その ObjectEventId を現行ビジネス・オブジェクトの ObjectEventId と比較します。

- これら 2 つの ObjectEventIds が同じであれば、現行のビジネス・オブジェクトは重複イベントを表します。その場合、コネクター・フレームワークは現行ビジネス・オブジェクトが表すイベントを無視します。したがって、このイベントは統合ブローカーに送信されません。
- これらの ObjectEventIds が異なる 場合、そのビジネス・オブジェクトは重複イベントを表していません。この場合、コネクター・フレームワークは現行ビジネス・オブジェクトを JMS モニター・キューにコピーし、それを JMS デリバリー・キューにデリバリーします (この処理はすべて同じ JMS トランザクションの中で行われます)。JMS デリバリー・キューの名前は、DeliveryQueue コネクター構成プロパティーから取得します。gotApplEvent() メソッドの呼び出し後は、コネクターの pollForEvents() メソッドに制御が戻ります。

JMS 対応コネクターで Duplicate Event Elimination をサポートするには、コネクターの pollForEvents() メソッドに以下のステップが含まれていることを確認してください。

- 非 JMS イベント・ストアから検索されたイベント・レコードからビジネス・オブジェクトを作成するときは、そのイベント・レコードの一意のイベント ID をビジネス・オブジェクトの ObjectEventId 属性として保管します。

  アプリケーションはこのイベント ID を生成することによって、イベント・ストア内のイベント・レコードを一意に識別します。イベントが統合ブローカーに送信されてから、このイベント・レコードの状況が変更可能になるまでの間にコネクターが停止した場合、そのイベント・レコードは進行中の状況のままイベント・ストアに残ります。コネクターは、バックアップされるときに、進行中のイベントがあればそのイベントをリカバリーする必要があります。コネクターは、ポーリングを再開するときに、イベント・ストアにまだ残っているイベント・レコードのビジネス・オブジェクトを生成します。ただし、すでに送信されたビジネス・オブジェクトと、新規ビジネス・オブジェクトの両方に同じ ObjectEventId

のイベント・レコードがあるため、コネクター・フレームワークは新規ビジネス・オブジェクトを複製と認識し、統合ブローカーに送信しません。

- コネクターのリカバリー時は、コネクターが新規イベントのポーリングを開始する*前に* 進行中のイベントを処理するようにしてください。

  コネクターが始動時に進行中のイベントを Ready-for-Poll 状況に変更しない限り、ポーリング・メソッドは再処理するイベント・レコードを選出しません。

メタオブジェクト属性の構成

HL7 用コネクターは次の 2 種類のメタオブジェクトを認識し、読み取ることができます。

- 静的コネクター・メタオブジェクト
- 動的子メタオブジェクト

動的子メタオブジェクトの属性値は、静的メタオブジェクトの属性値を複製し、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、さまざまなビジネス・オブジェクト用に定義された変換プロパティーのリストで構成されます。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、まずストリング属性を作成し、構文 `busObj_verb` を使用して名前を付けます。例えば、動詞 Create が付いたカスタマー・オブジェクトの変換プロパティーを定義するには、`HL7_MTADT_A03_Create` という名前の属性を作成します。この属性のアプリケーション固有のテキストに、実際の変換プロパティーを指定します。

さらに、予約済みの属性 `Default` をメタオブジェクトに定義できます。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。このような場合、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・ハンドラーがこのテキストのみではビジネス・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できないことを示すエラーをレポートします。

表 15 にメタオブジェクト・プロパティーの詳細を示します。

表 15. 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を組み合わせた属性のアプリケーション固有テキスト内に指定する必要があります。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、静的メタオブジェクト内に HL7_MT*nnn_Verb* という名前の属性を設定します。ここで、*nnn* は HL7 メッセージ・タイプを表します (例、HL7_MTADT_A03_Create)。HL7_MTADT_A03_Create 属性は、名前と値のペアを含むアプリケーション固有のテキストを含んでいる必要があります。例えば、CollaborationName=MyCustomerProcessingCollab です。構文の詳細については、 33 ページの『アプリケーション固有の情報』 セクションを参照してください。上記のように設定しないと、コネクターが Customer ビジネス・オブジェクトにかかわる要求を同期処理するときにランタイム・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 プロパティーを設定することができます。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・ボディが空白になります。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。これは 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	入力フォーマットとは、所与のビジネス・オブジェクトと関連付けられるメッセージ・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、そのメッセージは (変換可能であれば) 所与のビジネス・オブジェクトに変換されます。このフォーマットがビジネス・オブジェクト用に指定されていない場合、コネクターは、所与のビジネス・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。
OutputFormat	出力フォーマットは、所与のビジネス・オブジェクトから作成されるメッセージに設定されます。OutputFormat プロパティーの値が指定されていない場合は、入力フォーマットが使用可能であれば、それが使用されます。動的子メタオブジェクトに定義された OutputFormat 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。
InputQueue	コネクターが新規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です。コネクター固有のプロパティーを使用すれば、複数の InputQueues を構成し、オプションで各キューに異なるデータ・ハンドラーをマップすることができます。

表 15. 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
OutputQueue	出力キューとは、所与のビジネス・オブジェクトから派生したメッセージがデリバリーされるキューです。動的子メタオブジェクトに定義された OutputQueue 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。
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、その値が true の場合は、ResponseTimeout に指定された時間内に応答が受信されないと、コネクターは APP_RESPONSE_TIMEOUT を戻します。応答メッセージを待っているその他のすべてのスレッドは、即時に統合ブローカーに APP_RESPONSE_TIMEOUT を戻します。これにより、統合ブローカーはコネクターへの接続を終了します。動的子メタオブジェクトに定義された TimeoutFatal 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。

注: コネクター固有のプロパティーにある InputQueue プロパティーは、アダプターがポーリングする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 MO では、InputQueue プロパティーおよび InputFormat プロパティーは、アダプターが指定されたメッセージを特定のビジネス・オブジェクトにマップする条件として使用できます。この機能は、Adapter for Healthcare Data Protocols では使用されていません。

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、名前と値のペアをセミコロンで区切る形式で構成されます。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。

データ・ハンドラーの InputQueues へのマッピング

静的メタオブジェクトのアプリケーション固有の情報に InputQueue プロパティーを使用すれば、データ・ハンドラーを入力キューに関連付けることができます。この機能は、フォーマットや変換要件の異なる取引先が複数ある場合に有効です。このように設定するには、以下の作業を行います。

1. コネクター固有のプロパティー (24 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します。
2. アプリケーション固有の情報の中で、各入力キューのキュー・マネージャーおよび入力キュー名、さらにはデータ・ハンドラー・クラス名、MIME タイプを指定します。

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

```
[Attribute]
Name = HL7_MTADT_A03_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputQueue=//queue.manager/CompReceipts;
  DataHandlerClassName=com.crossworlds.
DataHandlers.HL7.disposition_notification;
  DataHandlerMimeType=message/
disposition_notification
IsRequiredServerBound = false
[End]
```

入力フォーマットの多重定義

メッセージを検索するとき、コネクターは通常は、1 組の特定ビジネス・オブジェクトと動詞の組み合わせに対して入力フォーマットを突き合わせます。そして、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、そのメッセージの内容はユーザーが期待するビジネス・オブジェクトと対応するかを検証できます。

ただし、複数のビジネス・オブジェクトに同じ入力フォーマットが定義されていると、コネクターは、データ・ハンドラーに渡す前に、そのデータがどのビジネス・オブジェクトを表すのかを判別できません。このような場合、コネクターはメッセージの内容のみをデータ・ハンドラーに渡し、生成されたビジネス・オブジェクトを基にして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の内容のみに基づいてビジネス・オブジェクトを判別しなければなりません。

生成されたビジネス・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、このビジネス・オブジェクトと任意の動詞の組み合わせに対して定義された変換プロパティーを検索します。ここで検出される変換プロパティーが 1 セットのみであれば、コネクターはそこで指定された動詞を割り当てます。複数のプロパティーが検出された場合、コネクターはそれらの動詞を識別できないため、メッセージの処理に失敗します。

静的構成メタオブジェクト

静的構成メタオブジェクトは、MQ 接続性に固有のものであり、以下のサービスを提供します。

- 出力キューを定義する
- BO ごとに入力キューからメッセージを取り出すときの動詞を定義します。この BO_Verb と入力キューの関連は、属性名で指定します。
- BO ごとに入力キューと出力キューの両方からのデータの形式を指定します。この情報は、ASI プロパティーで指定します。

表 16. ここで、IK=Is key、FK=Foreign key、IR=Is required、C=Cardinality、ML=Maximum length、および Def=Default です。

属性名	Type	IK	FK	IR	C	ML	Def	ASI	コメント
Default	String	はい	いいえ	はい	1	1		OutputQueue =queue: //crossworlds .queue. manager/ MQCONN.OUT	出力キューの定義
HL7 _Message _Create	String	いいえ	いいえ	いいえ	1	7	Create	InputFormat =MQSTR; OutputFormat =MQSTR	
Object Event ID		いいえ		いいえ	1				システムが使用するために予約済みです。

静的メタオブジェクトの例

下記に示す静的メタオブジェクトは、動詞 Create および Retrieve を使用して HL7_MTADT_A03 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 がメタオブジェクト内に定義されている点に注目してください。コネクターはこの属性の変換プロパティー

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

を、その他のすべての変換プロパティーのデフォルト値として使用します。したがって、属性に他の値が指定されたり、動的子メタオブジェクトの値によってオーバーライドされない限り、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 に送信し、応答メッセージを待ちます。そして、5000 ミリ秒以内に応答がなければ、コネクターは即時に終了します。

動詞 Create 付きのビジネス・オブジェクト: 属性 HL7_MTADT_A03_Create は、NEW フォーマットのメッセージがあれば、そのメッセージを動詞 Create 付きのビジネス・オブジェクトに変換するようにコネクターに指示します。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 付きのビジネス・オブジェクト: 属性 HL7_MTADT_A03_Retrieve は、動詞 Retrieve 付きのビジネス・オブジェクトを RETRIEVE フォーマットのメッセージとして送信するように指定します。デフォルトの応答時間がオーバーライドされ、コネクターはタイムアウトまで最長で 10000 ミリ秒間待機するようになっている点に注意してください (応答が受信されなければコネクターが終了する点は同じです)。

```
[ReposCopy]
Version = 3.0.0
Repositories = 1cHyILNuPTc=
[End]
[BusinessObjectDefinition]
```

```
Name = Sample_MO
Version = 3.0.0

[Attribute]
Name = Default
Type = String
Cardinality = 1
MaxLength = 1
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = OutputQueue=queue://crossworlds.queue.manager/MQCONN.OUT;
   ResponseTimeout=5000;TimeoutFatal=true
IsRequiredServerBound = false
[End]
[Attribute]
Name = HL7_MTADT_A03_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputFormat=NEW
IsRequiredServerBound = false
[End]
[Attribute]
Name = HL7_MTADT_A03_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]
```

動的子メタオブジェクト

静的メタオブジェクトを介して必要なメタデータを指定することが困難または不可能な場合は、オプションとして、コネクターは各ビジネス・オブジェクト・インスタンスに対して実行時に指定されたメタデータを受け入れることもできます。

コネクターは、コネクターに渡されるトップレベルのビジネス・オブジェクトに子として追加された動的メタオブジェクトから、変換プロパティーを認識して読み取

ります。動的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た静的メタオブジェクトを介して指定できる変換プロパティーから複製されます。

動的子メタオブジェクトのプロパティーは静的メタオブジェクトにあるプロパティーをオーバーライドするので、動的子メタオブジェクトを指定する場合は、静的メタオブジェクトを指定するコネクター・プロパティーを含める必要はありません。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 17 に、ビジネス・オブジェクト HL7_MTADT_A03_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 17. HL7_MTADT_A03_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
HL7_MTADT_A03_Create	InputFormat=ORDER_IN; OutputFormat=ORDER_OUT; OutputQueue=QueueA; ResponseTimeout=10000; TimeoutFatal=False

表 18 に、ビジネス・オブジェクト HL7_MT_Create の動的子メタオブジェクトの例を示します。

表 18. HL7_MTADT_A03_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、動的子メタオブジェクトの値は静的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。

ポーリング時の動的子メタオブジェクトの取り込み

ポーリング時に検索されるメッセージに関して、より多くの情報を統合ブローカーに提供するため、コネクターは、作成されたビジネス・オブジェクトについて動的メタオブジェクトの特定の属性がすでに定義されている場合は、その属性を取り込みます。

表 19 に、ポーリング用の動的子メタオブジェクトの構成方法を示します。

表 19. 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

表 19. ポーリング用の JMS 動的子メタオブジェクト構造 (続き)

プロパティー名	サンプル値
TimeoutFatal	CxIgnore

表 19 に示すように、動的子メタオブジェクトには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、所与のメッセージが検索されたキューの名前が入ります。このプロパティーは、子メタオブジェクトに定義されていない場合は取り込まれません。

シナリオ例:

- コネクターは WebSphere MQ キューから、ORDER_IN フォーマットのメッセージを検索します。
- 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。

MO_DataHandler_Healthcare

次のメタオブジェクトは、どの Healthcare データ・ハンドラーの構成にも使用されます。

表 20. ここで、IK=Is key、FK=Foreign key、IR=Is required、C=Cardinality、ML=Maximum length、および Def=Default です。

属性名	Type	IK	FK	IR	C	ML	Def	コメント
HL7	MO_DataHandler_HL7	いいえ		いいえ	1	使用不可		属性名には、HL7 メッセージの MIME タイプの名前を指定します。
Dummy キー	String	はい	いいえ	はい	1	1		

MO_DataHandler_HL7

次のメタオブジェクトには、HL7 データ・ハンドラーの構成プロパティーが含まれています。BO_Prefix、Default Verb、および ClassName は、コネクターがデータ・ハンドラーの起動に使用します。

表 21. ここで、IK=Is key、FK=Foreign key、IR=Is required、C=Cardinality、ML=Maximum length、および Def=Default です。

属性名	Type	IK	FK	IR	C	ML	Def	コメント
BOPrefix	String	はい	いいえ	はい	1	4	HL7	BO の接頭部

表 21. ここで、*IK=Is key*、*FK=Foreign key*、*IR=Is required*、*C=Cardinality*、*ML=Maximum length*、および *Def=Default* です。 *(続き)*

属性名	Type	IK	FK	IR	C	ML	Def	コメント
Default Verb	String	いいえ	いいえ	いいえ	1	7	Create	BO の中の動詞セット
Class Name	String	いいえ	いいえ	いいえ	1	255	com.ibm.adapters.datahandlers.hl7.HL7DataHandle	HL7 データ・ハンドラーの Java クラス名
Representation	String	いいえ	いいえ	いいえ	1	1	Simplified	「Simplified」または「Native」のどちらかです。Simplified の場合、構文解析メソッドを使用して、メッセージ・ヘッダーにのみ詳細オブジェクトを表示します。Native の場合、構文解析メソッドを使用して、メッセージ全体に詳細オブジェクトを表示します。
Field delimiter	String	いいえ	いいえ	いいえ	1	1	|	全タイプ用のデフォルトのフィールド区切り文字。
Repetition Delimiter	String	いいえ	いいえ	いいえ	1	1	~	全タイプ用のデフォルトの反復区切り文字。
Component Delimiter	String	いいえ	いいえ	いいえ	1	1	^	全タイプ用のデフォルトのコンポーネント区切り文字。

表 21. ここで、*IK=Is key*、*FK=Foreign key*、*IR=Is required*、*C=Cardinality*、*ML=Maximum length*、および *Def=Default* です。 *(続き)*

属性名	Type	IK	FK	IR	C	ML	Def	コメント
Sub-component Delimiter	String	いいえ	いいえ	いいえ	1	1	&	全タイプ用のデフォルトのサブコンポーネント区切り文字。
MTEventMap	String	いいえ	いいえ	いいえ	1	255	file=...¥Healthcare¥dependencies¥ HL7¥BIA_ HL7MTEventMap.cfg	この構成には、(メッセージ・タイプ X イベント・タイプ) -> メッセージ構造のマップが含まれます。
I118N	String	いいえ	いいえ	いいえ	1	255	file=...¥Healthcare¥dependencies¥HL7¥BIA_HLI118N.cfg	BIA_HL7I118N.cfg ファイルのパス名を指定します。このファイルには、ISO 文字セット、エスケープ・シーケンス、および Java 名に関するマップ情報が含まれています。
DummyKey	String	はい	いいえ	いいえ	1			

動的子メタオブジェクトの例

```
[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 = HL7_MTADT_A03
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、HL7 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

メッセージ・トランスポートに関して、より多くの情報を取得したり、制御の範囲を広げるための属性を動的メタオブジェクトに追加することができます。このような属性を追加すれば、JMS プロパティーの変更、要求単位での ReplyToQueue の制

御 (アダプター・プロパティーに指定されたデフォルトの ReplyToQueue 値を使用するのでなく)、およびメッセージ CorrelationID の宛先変更が可能になります。このセクションでは、これらの属性の詳細と、同期、非同期の両モードでこれらの属性がイベント通知および要求処理に及ぼす影響について説明します。

以下に示す属性は、JMS および HL7 のヘッダー・プロパティーを反映しており、これらは動的メタオブジェクトで認識されます。

表 22. 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	JMSDeliveryMode
Priority	読み取り	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
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	

読み取り専用の属性は、イベント通知時にメッセージ・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。要求処理の中で応答メッセージが発行されるときに、これらのプロパティーには動的メタオブジェクトが取り込まれます。読み取り/書き込み属性は、要求処理の中で作成されるメッセージ・ヘッダーに設定されます。イベント通知時には、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が取り込まれます。

これらの属性の解釈および使用方法については、以降のセクションで説明します。

注: 上記の属性はいずれも必須ではありません。目的のビジネス・プロセスに関連する属性があれば、それを動的メタオブジェクトに追加してください。

JMS プロパティー: 動的メタオブジェクトの他の属性とは異なり、JMSProperties には単一カーディナリティーの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内のすべての属性には、JMS メッセージ・ヘッダーの変数部分で読み/書きされる単一プロパティーを以下のように定義します。

1. 属性の名前は、セマンティック値ではありません。
2. 属性のタイプは、JMS プロパティー・タイプにかかわらず、常に String です。
3. 属性のアプリケーション固有の情報には、この属性がマップする JMS メッセージ・プロパティーの名前とフォーマットを定義した 2 組の名前と値のペアを記述します。

次の表は、JMSProperties オブジェクトの属性として定義する必要のあるアプリケーション固有の情報のプロパティーを示します。

表 23. JMS プロパティー属性のアプリケーション固有の情報

名前	指定可能な値	コメント
名前	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が予約されていることがあります。このようなベンダー固有の機能にアクセスしようとする場合を除き、一般にユーザーは JMS で始まるカスタム・プロパティーを定義しないでください。
タイプ	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 には、JMS Message に値を設定するメソッドとして setIntProperty、setLongProperty、setStringProperty などがあります。ここで指定する JMS プロパティーのタイプは、これらのうちのどのメソッドを使用してメッセージにプロパティー値を設定するかを示します。

次の図は、動的メタオブジェクトの属性 JMSProperties と、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 の定義を表します。この属性のアプリケーション固有の情報には、それぞれの名前とタイプが定義されています。例えば属性 ID は、String タイプの JMS プロパティー ID と対応しています。

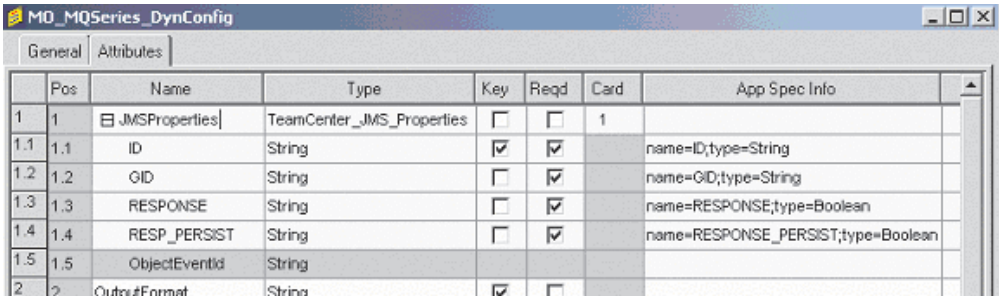

図 3. 動的メタオブジェクトの JMS プロパティー属性

非同期イベント通知: ヘッダー属性の付いた動的メタオブジェクトがイベント・ビジネス・オブジェクトにある場合、コネクターは以下のステップを実行します (下記の処理のほかに、トランスポート関連データをメタオブジェクトに取り込みます)。

1. メッセージの JMSCorrelationID ヘッダー・フィールドに指定された値をメタオブジェクトの CorrelationId 属性に取り込みます。
2. メッセージの JMSReplyTo ヘッダー・フィールドに指定されたキューをメタオブジェクトの ReplyToQueue 属性に取り込みます。このヘッダー・フィールドはメッセージ内の Java オブジェクトによって表されるので、属性にはキューの名前 (通常は URI) が取り込まれます。
3. メッセージの JMSDeliveryMode ヘッダー・フィールドに指定された値をメタオブジェクトの DeliveryMode 属性に取り込みます。
4. メッセージの JMSPriority ヘッダー・フィールドをメタオブジェクトの Priority 属性に取り込みます。
5. メッセージの JMSDestination ヘッダー・フィールドの名前をメタオブジェクトの Destination 属性に取り込みます。Destination はオブジェクトによって表されるので、この属性には Destination オブジェクトの名前が取り込まれます。
6. メッセージの JMSExpiration ヘッダー・フィールドの値をメタオブジェクトの Expiration 属性に取り込みます。
7. メッセージの JMSMessageID ヘッダー・フィールドの値をメタオブジェクトの MessageID 属性に取り込みます。
8. メッセージの JMSRedelivered ヘッダー・フィールドの値をメタオブジェクトの Redelivered 属性に取り込みます。
9. メッセージの JMSTimeStamp ヘッダー・フィールドの値をメタオブジェクトの TimeStamp 属性に取り込みます。
10. メッセージの JMSType ヘッダー・フィールドの値をメタオブジェクトの Type 属性に取り込みます。
11. メッセージの JMSXUserID プロパティー・フィールドの値をメタオブジェクトの UserID 属性に取り込みます。
12. メッセージの JMSXAppID プロパティー・フィールドの値をメタオブジェクトの AppID 属性に取り込みます。
13. メッセージの JMSXDeliveryCount プロパティー・フィールドの値をメタオブジェクトの DeliveryCount 属性に取り込みます。

14. メッセージの JMSXGroupID プロパティー・フィールドの値をメタオブジェクトの GroupID 属性に取り込みます。
15. メッセージの JMSXGroupSeq プロパティー・フィールドの値をメタオブジェクトの GroupSeq 属性を取り込みます。
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調べます。アダプターは、メッセージ内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取り込みます。メッセージ内の特定のプロパティーが未定義の場合、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: イベントの同期処理では、アダプターはイベントを POST して、統合ブローカーからの応答を待ち、この応答を受けてから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・データが変更されれば、その変更内容は、戻される応答メッセージに反映されます。イベントを POST する前にアダプターは、前述の非同期イベント通知と同様に、動的メタオブジェクトを取り込みます。動的メタオブジェクトに設定された値は、応答が発行されるヘッダーに以下のように反映されます (動的メタオブジェクト内のその他の読み取り専用ヘッダー属性は、すべて無視されます)。

- **CorrelationID** 動的メタオブジェクトに属性 CorrelationId が含まれる場合は、発信元のアプリケーションが期待する値を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を元の要求と突き合わせます。CorrelationID の値が予期しない値または無効な値であると、問題が発生します。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メッセージと応答メッセージをどのように処理するかを決定しておくと便利です。同期要求での CorrelationID の取り込みには、以下の 4 つの方法があります。
  1. 値を変更せず、そのままにします。この場合、応答メッセージの CorrelationID は、要求メッセージの CorrelationID と同じです。これは、WebSphere MQ オプション MQRO_PASS_CORREL_ID に相当します。
  2. 値を CxIgnore に変更します。この場合、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 に相当します。
  3. 値を CxBlank に変更します。この場合、コネクターは応答メッセージに CorrelationID を設定しません。
  4. 値をカスタム値に変更します。この場合は、応答を処理するアプリケーションがカスタム値を認識する必要があります。

  メタオブジェクトに属性 CorrelationID を定義しなければ、コネクターは CorrelationID を自動的に処理します。
- **ReplyToQueue** 属性 ReplyToQueue に別のキューを指定して動的メタオブジェクトを更新すると、コネクターは指定されたキューに応答メッセージを送信します。この方法はお勧めできません。コネクターが別のキューに応答メッセージを送信するようにすると、通信に障害が発生する可能性があります。これは、要求メッセージに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っているとみられるためです。

- **JMS properties** 動的メタオブジェクト内の JMS Properties 属性の値セットは、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、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があればそれを使用して、要求メッセージを取り込み、発行します。コネクターは要求メッセージを送信する前に以下のステップを実行します。

1. 動的メタオブジェクトに属性 CorrelationID がある場合、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. 動的メタオブジェクトに属性 ReplyToQueue が指定されている場合、コネクターは要求メッセージを介してこのキューを渡し、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に指定された ReplyToQueue 値をオーバーライドできます。さらに ResponseTimeout (コネクターが応答待ちをやめる時間) に負の値を指定した場合、ReplyToQueue は応答メッセージに設定されますが、コネクターは実際には応答を待ちません。
3. 動的メタオブジェクトに属性 JMSProperties が指定されている場合は、動的子メタオブジェクト内の対応する JMS プロパティーが、コネクターから送信されるアウトバウンド・メッセージに設定されます。

注: 動的メタオブジェクトのヘッダー属性が未定義または CxIgnore と指定されている場合、コネクターはデフォルト設定に従って処理します。

同期要求処理: コネクターは、動的メタオブジェクトがあればそれを使用して、要求メッセージを取り込み、発行します。動的メタオブジェクトにヘッダー属性があれば、コネクターは応答メッセージ内の対応する新規の値をメタオブジェクトに取り込みます。コネクターは、応答メッセージを受信した後、以下のステップを実行します (下記の処理のほかに、トランスポート関連データをメタオブジェクトに取り込みます)。

1. 動的メタオブジェクトに属性 CorrelationID がある場合、アダプターはこの属性を、応答メッセージに指定された JMSCorrelationID の値に更新します。
2. 動的メタオブジェクトに属性 ReplyToQueue が定義されている場合、アダプターはこの属性を、応答メッセージに指定された JMSReplyTo の名前に更新します。
3. 動的メタオブジェクトに属性 DeliveryMode がある場合、アダプターはこの属性を、メッセージの JMSDeliveryMode ヘッダー・フィールドの値に更新します。
4. 動的メタオブジェクトに属性 Priority がある場合、アダプターはこの属性を、メッセージの JMSPriority ヘッダー・フィールドの値に更新します。
5. 動的メタオブジェクトに属性 Destination が定義されている場合、アダプターはこの属性を、応答メッセージに指定された JMSDestination の名前に更新します。
6. 動的メタオブジェクトに属性 Expiration がある場合、アダプターはこの属性を、メッセージの JMSExpiration ヘッダー・フィールドの値に更新します。
7. 動的メタオブジェクトに属性 MessageID がある場合、アダプターはこの属性を、メッセージの JMSMessageID ヘッダー・フィールドの値に更新します。
8. 動的メタオブジェクトに属性 Redelivered がある場合、アダプターはこの属性を、メッセージの JMSRedelivered ヘッダー・フィールドの値に更新します。

9. 動的メタオブジェクトに属性 TimeStamp がある場合、アダプターはこの属性を、メッセージの JMSTimeStamp ヘッダー・フィールドの値に更新します。
10. 動的メタオブジェクトに属性 Type がある場合、アダプターはこの属性を、メッセージの JMSType ヘッダー・フィールドの値に更新します。
11. 動的メタオブジェクトに属性 UserID がある場合、アダプターはこの属性を、メッセージの JMSXUserID ヘッダー・フィールドの値に更新します。
12. 動的メタオブジェクトに属性 AppID がある場合、アダプターはこの属性を、メッセージの JMSXAppID プロパティー・フィールドの値に更新します。
13. 動的メタオブジェクトに属性 DeliveryCount がある場合、アダプターはこの属性を、メッセージの JMSXDeliveryCount ヘッダー・フィールドの値に更新します。
14. 動的メタオブジェクトに属性 GroupID がある場合、アダプターはこの属性を、メッセージの JMSXGroupID ヘッダー・フィールドの値に更新します。
15. 動的メタオブジェクトに属性 GroupSeq がある場合、アダプターはこの属性を、メッセージの JMSXGroupSeq ヘッダー・フィールドの値に更新します。
16. 動的メタオブジェクトに属性 JMSProperties が定義されている場合、アダプターは子オブジェクトに定義されたプロパティーを、応答メッセージにある値に更新します。子オブジェクトに定義されたプロパティーがメッセージに存在しなければ、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、要求メッセージに設定された CorrelationID を変更しても、アダプターによる応答メッセージの識別方法に影響はありません。デフォルトではアダプターは、応答メッセージの CorrelationID は、アダプターが送信した要求のメッセージ ID と同一であると想定しています。

エラー処理: JMS プロパティーをメッセージから読み取ったりメッセージに書き込むことができない場合、コネクターはエラーをログに記録し、その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、コネクターはエラーをログに記録し、その要求は失敗します。CorrelationID が無効または設定できない場合、コネクターはエラーをログに記録し、その要求は失敗します。いずれの場合でも、ログに記録されるメッセージはコネクター・メッセージ・ファイルから取得されます。

始動ファイル構成

HL7 用コネクターを始動する前に、MQ Java クライアント・ライブラリーのパス情報を指定して始動ファイルを構成しておく必要があります。始動ファイル内に MQ Java クライアント・ライブラリーがリストされていない場合、以下のセクションの説明に従って、これらのファイルを Windows システム用または UNIX システム用に構成してください。

Windows

Windows プラットフォーム用のコネクターを構成するには、以下の手順に従って start_HL7.bat ファイルを変更する必要があります。

1. start_HL7.bat ファイルを開きます。

2. 「Set the directory containing your MQ Java client libraries」で始まるセクションにスクロールし、使用する MQ Java クライアント・ライブラリーのロケーションを指定します。

UNIX

UNIX プラットフォーム用のコネクターを構成するには、以下の手順に従って start_HL7.sh ファイルを変更する必要があります。

1. start_HL7.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にスクロールし、使用する WebSphere MQ Java クライアント・ライブラリーのロケーションを指定します。

複数のコネクター・インスタンスの作成

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

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

新規ディレクトリーの作成

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

```
ProductDir¥connectors¥connectorInstance
```

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

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

```
ProductDir¥repository¥connectorInstance
```

ビジネス・オブジェクト定義の作成

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

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

ProductDir¥repository¥initialConnectorInstance

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

コネクター定義の作成

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

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

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

   dirname
2. この始動スクリプトを、49 ページの『新規ディレクトリーの作成』で作成したコネクター・ディレクトリーに格納します。
3. 始動スクリプトのショートカットを作成します (Windows のみ)。
4. 初期コネクターのショートカット・テキストをコピーし、新規コネクター・インスタンスの名前に一致するように (コマンド行で) 初期コネクターの名前を変更します。

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

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

コネクターの始動

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

ProductDir¥connectors¥*connName*

ここで、*connName* はコネクターを示します。始動スクリプトの名前は、表 24 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 24. コネクターの始動スクリプト

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

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

- Windows システムで**「スタート」**メニューから。

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

    `start_connName connName brokerName [-cconfigFile ]`
  - UNIX ベースのシステム:

    `connector_manager_connName -start`

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

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

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

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

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

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

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

コネクターの停止

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

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

    ```
    connector_manager_connName -stop
    ```

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

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

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

第 3 章 ビジネス・オブジェクト

Connector for Healthcare Data Protocols は、メタデータ主導型コネクターです。WebSphere ビジネス・オブジェクトでは、メタデータはアプリケーションのデータに関するデータです。このデータはビジネス・オブジェクト定義に格納され、コネクターとアプリケーションの対話を支援します。メタデータ主導型コネクターでは、そのサポートする各ビジネス・オブジェクトはコネクター内にハードコーディングされた命令ではなく、ビジネス・オブジェクト定義内にエンコードされたメタデータに基づいて処理されます。

ビジネス・オブジェクト・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値、およびアプリケーション固有テキストの内容が含まれます。コネクターはメタデータ主導型なので、コネクター・コードを変更しなくても、新しいビジネス・オブジェクトまたは変更されたビジネス・オブジェクトを処理することができます。しかし、コネクターの構成済みデータ・ハンドラーは、そのビジネス・オブジェクトの構造、オブジェクトのカーディナリティー、アプリケーション固有テキストのフォーマット、およびビジネス・オブジェクトのデータベース表記について前提事項を設けています。そのため、Healthcare データ・プロトコル用ビジネス・オブジェクトを作成または変更する場合は、コネクターが従うものとして設計されているルールに変更内容が適合していないと、コネクターが新しいビジネス・オブジェクト、または変更されたビジネス・オブジェクトを正しく処理できません。

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することができます。

コネクターのビジネス・オブジェクト要件

コネクターのビジネス・オブジェクト要件は、以下の Healthcare データ・ハンドラーの変換方法を反映しています。

- HL7 メッセージと WebSphere ビジネス・オブジェクトの相互変換
- NCPDP メッセージと WebSphere ビジネス・オブジェクトの相互変換

以下のセクションでは、WebSphere ビジネス・オブジェクトの要件および HL7 メッセージ構造について説明します。

以下の WebSphere 資料も参照してください。

- *テクニカル入門 (IBM WebSphere InterChange Server)*（ICS が統合ブローカーの場合)

- *IBM WebSphere Business Integration Adapters WebSphere MQ Integrator Broker 用インプリメンテーション・ガイド* (MQ Integrator Broker が統合ブローカーの場合)
- *ビジネス・オブジェクト開発ガイド*

ビジネス・オブジェクト階層

WebSphere ビジネス・オブジェクトには、フラットなビジネス・オブジェクトと階層のあるビジネス・オブジェクトがあります。**フラットな**ビジネス・オブジェクトの属性はすべて**単純**であり、各属性は単一の値 (`String`、`Integer`、`Date` など) を表します。

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。そのため、それぞれの子ビジネス・オブジェクトには、1 つの子ビジネス・オブジェクト、またはビジネス・オブジェクトの配列など、いろいろと含めることができます。

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・オブジェクトの間には、以下の 2 種類の関係があります。

- **単一カーディナリティー:** 親ビジネス・オブジェクトの属性が単一の子ビジネス・オブジェクトを表す場合です。属性のタイプは子ビジネス・オブジェクトのタイプと同じです。
- **複数カーディナリティー:** 親ビジネス・オブジェクトの属性が子ビジネス・オブジェクトの配列を表す場合です。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 コネクターでは、その子データ・ビジネス・オブジェクトの形式を判別し、子をルーティングする情報をラッパー・ビジネス・オブジェクトに入れる必要があります。

ビジネス・オブジェクトの属性プロパティー

ビジネス・オブジェクト・アーキテクチャーは、属性に適用され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、これらのプロパティーの一部についてコネクターの解釈方法を説明します。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

各ビジネス・オブジェクト属性は、ビジネス・オブジェクト内で固有の名前を持つ必要があります。名前は、属性が含むデータを説明する必要があります。

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、以下のように属性のデータ型を定義します。

- 単純属性の場合にサポートされるタイプは、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としてタイプを指定します (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す場合は、子ビジネス・オブジェクト定義の名前としてタイプを指定し、カーディナリティーに n を指定します。

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・オブジェクトで、少なくとも 1 つの属性を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 (ICS が統合ブローカーの場合にのみ関係します)。

Foreign key プロパティー

通常、Foreign Key プロパティーはアプリケーション固有ビジネス・オブジェクトで使用し、属性の値が別のビジネス・オブジェクトの 1 次キーを保持し、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することを指定します。別のビジネス・オブジェクトの 1 次キーを保持する属性を**外部キー**と呼びます。外部キーを表す各属性の Foreign Key プロパティーは `true` として定義します。

また、ほかの処理命令の Foreign Key プロパティーを使用することもできます。例えば、このプロパティーを使用してコネクターが実行する外部キー検索の種類を指定できます。この場合は Foreign Key を `true` に設定し、コネクターがデータベースにエンティティーが存在するかどうかを確認し、エンティティーのレコードが存在する場合にのみ関係を作成することを指定します。

Required プロパティー

Required プロパティーは、属性に値が含まれていなければならない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 (ICS が統合ブローカーの場合にのみ関係します)。

Max length プロパティー

Max Length プロパティーは、`String` タイプの属性に格納可能なバイト数に設定します。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、固定幅データ・ハンドラーを使用する場合に非常に重要です。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 (ICS が統合ブローカーの場合にのみ関係します)。

Default value プロパティー

Default Value プロパティーは属性のデフォルト値を指定できます。

このプロパティーをアプリケーション固有ビジネス・オブジェクトに指定し、UseDefaults コネクター構成プロパティーを `true` に設定した場合は、コネクターは、ビジネス・オブジェクト定義で指定されたデフォルト値を使用して実行時に値がない属性に値を指定します。

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 プロパティーはビジネス・オブジェクトの処理に使用しますが、Comments プロパティーはドキュメンテーション情報のみを提供します。

特殊属性値

ビジネス・オブジェクト内の単純属性は、特殊値 CxIgnore を持つことができます。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き、値が CxIgnore の属性をすべて無視します。すなわち、これらの属性はコネクターには不可視と同様です。

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。

HL7 メッセージ構造の概要

HL7 規格は、主にバッチ・モードおよび対話モードの 2 つの操作モードをサポートしています。システム間でやり取りされるデータは、大きく 2 つのグループ (個別およびバルク) に分類されます。これらのグループは、2 つの操作モードに対応しています。

定義上、バルク・メッセージは個々のメッセージの集まりです。ヘッダー、順次にリストされた個々のメッセージ、およびトレーラーから構成されます。組み込まれた個々のメッセージは、あたかも対話式操作モードになっているかのように、メッセージ構造を保持します。2 つの異なる操作モードで動作している個々のメッセージの間には、構造的な差異はいっさいありません。ただし、バッチ操作の本質上、ダイアログ・メッセージは許可されないため、個々のメッセージの一部のカテゴリーは、対話式操作モードにのみ適しています。

個々のメッセージの構造にはそれぞれ、ヘッダーが含まれていますが、本文はあるものとないものがあります。個々のメッセージは、この規格によると「システム間で転送される、アトミック単位のデータ」です。ただし、この同規格では、論理メッセージの表記もできます。メッセージのデータは、物理的に複数の個別メッセージに分割することが可能であり、メッセージ・ヘッダー内の論理メッセージ ID を使用して相互に関連します。あるメッセージから個別メッセージへの細分化は、主として、メッセージのやり取りに従事する当事者間で折衝されたメッセージ長を基準にして駆動されます。

HL7 のメッセージ構造および HL7 のメッセージ・コンポーネントについて詳しくは、以下を参照してください。

HL7 用ビジネス・オブジェクトの概要

HL7 のメッセージ表記方法には、次の 2 通りがあります。

- サンプルのオブジェクト表示
  HL7 のメッセージ本文によって、単にヘッダーにのみオブジェクトを表示します。メッセージ本文は BLOB 、つまり全メッセージ・タイプを表す 1 つのビジネス・オブジェクトとして扱われます。
- 固有のオブジェクト表示
  HL7 メッセージのこのレベルによって、メッセージ・ヘッダーとメッセージ本文の両方にオブジェクトを表示します。対応するビジネス・オブジェクトは、事実上すべての HL7 データ・エレメント (メッセージ構造またはクエリーをすべて含む) を表します。

以降のセクションでは、HL7 データ・エレメントを表すビジネス・オブジェクトを構成するときに従う必要があるルールについて説明します。

サポートされている固有メッセージ

サポートされている固有メッセージの作成、構成、または変更の際は、以下の点に留意してください。

- サンプル以外のメッセージ・タイプおよびクエリーはすべて、バージョン 2.4 の HL7 メッセージの標準仕様に基づいて定義されます。
- 標準仕様で用いられているテンプレートに対応する、業界固有ビジネス・オブジェクト (ISBO) は、参照用にのみ提供されています。一部のデータ・エレメントは完全には固定化されていないため、そのビジネス・オブジェクトに対してはサポートがいっさい提供されていません。
- HL7 標準の以前のバージョンでは、ユーザーは独自のオーダーメイド (CM) データ型を作成することができました。CM データ型は、標準に規定されていないため、業界固有ビジネス・オブジェクト (ISBO) のパブリッシュ済みコレクションには組み込まれていません。ただし、パブリッシュされた ISBO に用いられているメタデータ・モデルを使用すれば、そのような CM データ型に対応する ASBO 表記を作成することができます。

プリミティブ・データ型のマッピング

プリミティブ・データ型をマップする際には、次の表を使用してください。

表 25. プリミティブ・データ型のマッピング

HL7 データ・エレメント	ビジネス・オブジェクト属性のデータ型
SI	String
ID	String
NM	Float
DT	String
IS	String
ST	String
FT	String
TX	String

繰り返しデータ・エレメントのマッピング

繰り返しデータ・エレメントをマップする際には、次の表を使用してください。

表 26. 繰り返しデータ・エレメントのマッピング

繰り返し HL7 データ・エレメント	BO 属性のデータ型	カーディナリティー	ASI
SI、ID、DT、IS、ST、FT、TX	CW_Array_String	N	DataTypeID=HL7 データ型 ID
NM	CW_Array_Float	N	DataTypeID=NM

ISBO の定義

HL7 の全データ・エレメント・カタログを表すのに必要なビジネス・オブジェクトをすべて文書化するのは、法外なことです。次のセクションでは、ビジネス・オブジェクトの構成方法について説明します。また、従来の BO 構成規則に従わないいくつかの HL7 データ・エレメントに必要な特別処理については注記を付けてあります。

ビジネス・オブジェクトの構造は、HL7 データ・エレメントの構造 (ただし、HL7_DTEncodedText、HL7_DTMA、HL7_DTNA、HL7_DTQIP、および HL7_DTUnionALL を除く) に非常によく似ています。

注: HL7 データ・エレメントから BO を組み立てる規則については、以降のサブセクションで詳しく説明します。命名規則によると、[] を使うことにより、選択肢を示すことができます。また、<> を使うことにより、不等号括弧で囲まれた内容 (命名規則において必須とされる文字の一部とはなっていない場合) に関する説明を示すことができます。各ビジネス・オブジェクトは、ObjectEventID BO を必ず含んでいなければなりません。この理由から、以下のビジネス・オブジェクトは、この ObjectEventID ビジネス・オブジェクトは明確には挙げてありません。

BO の名前

メッセージ BO

メッセージ BO は、以下のパラメーターを使用して構成します。

- 対応する HL7 データ・エレメント
  HL7 メッセージ・タイプとイベント/クエリー ID のペアで識別されるメッセージ構造。
- 命名規則
  HLT_MT<message type>_<Event code/query id>
  ここで、message type/query には、一字一語法による最小値を使用したイベント・コードが適用されます。
- マーク付け規則
  適用外

- 文法
  適用外
- 例:
  HL7_MTADT_A01。ADT メッセージ・タイプ、および ACK イベント (イベント・コード A01、A04、A08、および A13 を使用) を表します。
  HL7_MTQBP_Q21。照会ステートメント ID Q21 および Q24 を表します。

グループ BO

グループ BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  複数のセグメントを持つ任意のセグメント・グループ。
- 命名規則
  HLT_MT<message type>_<Event code/query id>_GP<group index>
  ここで、message type/query には、最小の字句値を使用したイベント・コードが適用されます。また、メッセージ構造のスコープ内では index は一意です。
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  HL7_MTADT_A03 は、グループ名 HL7_MTADT_A03_GP1 を持ちます。

表 27. セグメント・グループの命名規則

ADT^A03^ADT_A03	ADT メッセージ
MSH	メッセージ・ヘッダー
EVN	イベント・タイプ
PID	患者の識別番号
[PD1]	付加的所在情報
[{ ROL }]	役割
[PV2]	患者の来院 (追加情報)
[{ ROL }]	役割
[{ DB1 }]	身体障害に関する情報
[{ DG1 }]	診断に関する情報
[DRG]	診断関連のグループ
[{	
PR1	手続き
[{ ROL }]	役割
}]	
[{ OBX }]	測定値/結果
[PDA }	患者の死/病理解剖

セグメント BO

セグメント BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  HL7 セグメントと、QPD、QED、RCP、QAK セグメントに関するセグメント定義 (各セグメントには、適合ステートメントのパラメーター・テーブルが使用されます)。
- 命名規則
  HL7_SG<segment ID>_[<Query ID>]
  ここで、segment ID は HL7 によって定義されるセグメント ID です。Query ID は HL7 またはユーザーによって定義されるクエリー ID です。QPD、QED、RCP、QAK セグメントにのみ Query ID が必要です。
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  MSH 用のセグメント BO の名前は HL7_SGMSH です。QBP_Q21 内の QBD セグメント用のセグメント BO の名前は HL7_SGQBD_Q21 です。

複合データ型 BO

複合データ型 BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  複合 HL7 データ型 (MA、NA、および QIP データ型を除く)。
- 命名規則
  HL7_DT<data type>
  ここで、data type は、HL7 により定義された、複合 HL7 データ型用のデータ型 ID です。

  CE データ型については、次の規則を使用します。 HL7_DT<Data Type>_<Table Number>
  ここで、Table Number は、CE データ型のセグメント・フィールド用テーブルの ID です。CE データ型はセグメント定義内に定義されます。

  CM データ型は、そのポリモアフィック性によって、別個の処理を保証しています。次の表の中に指定されているのが、CM データ型の各インスタンスです。データ構造体名は、HL7 メッセージ標準 Access データベース内のデータ構造体の表に指定されている名前です。

表 28. CM データ用の複合データ型に対応する全ビジネス・オブジェクトのリスト

DT BO	データ構造名	説明
HL7_DTAUI	AUI	権限情報
HL7_DTCCD	CCD	チャージ時間
HL7_DTCCP	CCP	チャネル調整パラメーター
HL7_DTCSU	CSU	チャネル感度/ユニット
HL7_DDI	DDI	1 日あたりの控除額
HL7_DIN	DIN	活動化コード
HL7_DLD	DLD	チャージ解除のロケーション
HL7_DLT	DLT	デルタ・チェック
HL7_DTN	DTN	日付、タイプ、および番号

表 28. CM データ用の複合データ型に対応する全ビジネス・オブジェクトのリスト (続き)

DT BO	データ構造名	説明
HL7_DEIP	EIP	親オーダー
HL7_DTELD	ELD	エラー
HL7_DTLA1	LA1	アドレス情報が指定されたロケーション (バリアント 1)
HL7_DTLA2	LA2	アドレス情報が指定されたロケーション (バリアント 2)
HL7_DMOC	MOC	実施チャージ
HL7_DMSG	MSG	メッセージ・タイプ
HL7_DNDL	NDL	観察実施者
HL7_DNR	NR	値の範囲
HL7_DOCD_	OCD	オカレンス
HL7_DOSD	OSD	オーダー・シーケンス
HL7_DOSP	OSP	オカレンス・スパン
HL7_DPCF	PCF	事前許可
HL7_DPEN	PEN	ペナルティー
HL7_DTPI	PI	個人 ID
HL7_DTPIP	PIP	特権
HL7_DTPLN	PLN	実施者 ID 番号
HL7_DTPRL	PRL	親の結果リンク
HL7_DTPTA	PTA	ポリシー・タイプ
HL7_DTRFR	RFR	参照範囲
HL7_DTRMC	RMC	ルーム・カバレッジ
HL7_DTSPD	SPD	専門
HL7_DTSPS	SPS	試料ソース
HL7_DTUVC	UVC	値コードおよび量
HL7_DTVR	VR	値修飾子
HL7_DTWVI	WFI	チャネル ID
HL7_DTWVS	WVS	Waveform ソース

- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  CQ データ型用の BO の名前は HL_DTCQ です。
  次の抜粋に示すように、OBR セグメントの第 46 フィールドは CE データ型であり、テーブル 0411 を使用しています。この CE データ型の BO の名前は HL7_DTCE_0411 です。

表 29. OBR セグメントの第 46 フィールド

SEQ	LEN	DT	OPT	RP/#	TBL#	ITEM #	エレメント名
46	250	CE	O	Y	0411	01474	依頼者補給サービス情報

データ型 Union BO

データ型 Union BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  データ型に応じて異なります。
- 命名規則
  `HL7_DTUnionAll`
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  適用外

BO AppSpecificInfo

一般的なビジネス・オブジェクト用のアプリケーション固有プロパティーについて、以下に詳述します。

メッセージ BO

AppSpecificInfo 用のメッセージ BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  HL7 メッセージ・タイプとイベント/クエリー ID のペアで識別されるメッセージ構造。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  適用外

グループ BO

AppSpecificInfo 用のグループ BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  複数のセグメントを持つ任意のセグメント・グループ。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  `StructType=Group`
  `StructType` ASI プロパティーの値は、次の表に定義されています。

表 30. StructType の値定義テーブル

StructType の値	説明
Group	BO がセグメント・グループ用であることを示します。
Segment	BO がセグメント用であることを示します。
DataType	BO が HL7 データ型用であることを示します。
Union	BO が、ポリモアフィックデータ型を表す共用体 BO であることを示します。各データ型の可変範囲内で許可されるデータ型は、Union BO の属性で表されます。実行時には共用体 BO 内の属性の 1 つにのみデータが取り込まれます。値付き属性の選択方法へのコンテキストは、Type Context ASI プロパティーで示されます。
Array	BO がコンポーネント配列コンテナー用であることを示します。

- 例:
  適用外

セグメント BO

AppSpecificInfo 用のセグメント BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  HL7 セグメントと、QPD、QED、RCP、QAK セグメントに関するセグメント定義 (各セグメントには、適合ステートメントのパラメーター・テーブルが使用されます)。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  `StructType=Segment;SegID=<segment ID>`
  ここで、segment ID は HL7 セグメント ID です。
- 例:
  適用外

複合データ型の BO

BO AppSpecificInfo 用の複合データ型の BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  複合 HL7 データ型 (MA、NA、および QIP データ型を除く)。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  適用外

データ型 Union BO

BO AppSpecificInfo 用のデータ型 Union BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  データ型に応じて異なります。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  適用外

BO の属性構造

いくつかの一般的なビジネス・オブジェクトの属性構造については、次のセクションで詳述します。

メッセージ BO

BO の属性構造用のデータ型 Message BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  セグメント BO またはセグメント・グループ BO のタイプの属性。属性の実際のリストは、HL7 メッセージ構造の定義によって指定されます。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  メッセージ・タイプ ADT およびイベント・タイプ A61 用のメッセージ構造には、以下の属性タイプがあります。

表 31. HL7_MTADT_A61 の属性、および HL7 の対応部分それぞれの属性タイプ

属性タイプ	属性タイプ	HL7 の対応部分
1	HL7_SGMSH	MSH
2	HL7_SGEVN	EVN
3	HL7_SGPID	PID
4	HL7_SGPD1	[PD1]
5	HL7_SGV1	PV1
6	HL7_SGROL	[{ROL}]
7	HL7_SGPV2	[PV2}

グループ BO

BO の属性構造用のデータ型 Group BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  複数のセグメントを持つ任意のセグメント・グループ。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  BO HL7_MTADT_A03_GP1 の属性については、次の表に HL7 の対応部分と併せて示します。

表 32. HL7_MTADT_A03_GP1 の各属性の属性シーケンス、および HL7 の対応部分

属性シーケンス	属性タイプ	HL7 の対応部分
1	HL7_SGPR1	PR1
2	HL7_SGROL	[{ROL}]

セグメント BO

BO AttributeStructure 用のデータ型 Segment BO の構造は、以下のように要約されます。

- HL7 セグメントと、QPD、RDT セグメントに関するセグメント定義 (各セグメントには、適合ステートメントのパラメーター・テーブルが使用されます)。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  BO HL7_SGQBD_Q21 (クエリー Q21 用の HL7 メッセージ標準仕様定義を使用した場合)。
-

表 33. BO HL7_SGQBD_Q21 内のクエリー定義 Q21 用の属性シーケンス

属性シーケンス	属性タイプ	HL7 の対応部分
1	HL7_DTCE	MessageQueryName (CE)
2	HL7_DTENDCODEDTEXT	QueryTag (ST)
3	HL7_DTCX	PersonIdentifier (CX)
4	HL7_DTCX	WhatDomainsReturned (CSX)

複合データ型の BO

BO AttributeStructure 用の複合データ構造体の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  複数のセグメントを持つ任意のセグメント・グループ。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  BO HL7_MTADT_A03_GP1 の属性については、次の表に HL7 の対応部分と併せて示します。

表 34. HL7_MTADT_A03_GP1 の各属性の属性シーケンス、および HL7 の対応部分

属性シーケンス	属性タイプ	HL7 の対応部分
1	HL7_SGPR1	PR1
2	HL7_SGROL	[{ ROL }]

データ型 Union BO

Union BO は、標準仕様により定義されている全データ型の全集合を表します。

注: Union BO に別の共用体 BO を再帰的に含めることはできません。

Union BO の構造は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  - CM および Varies 以外の HL7 データ型
  - 61 ページの表 28からの CM 値
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  適用外

BO の属性プロパティー名

一般的なビジネス・オブジェクトの属性プロパティー名について、以下に詳述します。

メッセージ BO

メッセージ BO のプロパティー名の属性は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。
- 命名規則
  <説明の中の単語の連結>
  ここで、連結の際は以下の規則に従います。

表 35. 連結規則

ルール名	ルール記述
単語の選択	説明の中のすべての単語 (ただし、句読点および補助的な単語を除く) は、文字の大文字化を顧慮せずに組み込まれて連結されます。例えば、″a″、″an″、″the″、″in″、″of″、″for″、″with″、″at″、″/″、″¥″、″-″、″(″、″)″,″[″、″]″、″'″、スペース、タブなどが連結対象です。
データ型	説明の中のデータ型 ID が括弧で囲まれていることもありますが、これらのデータ型は括弧と一緒にストリップされます。
単語の大文字化	残った各単語については、先頭文字が大文字化され、先頭文字以外の文字は小文字に抑制されます。

注: HL7 標準仕様は、テーブルを使用して、メッセージ構造またはクエリー・メッセージ構造を伝えます。ただし、各列に説明的な短い単語が含まれている場合であっても、どのテーブル列にも列タイトル内の 「説明」は付けられません。これは「属性説明」の参照先です。次の例では、「ADT メッセージ」列は「説明」列と同等です。

- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  次の例では、BO `HL7_MTADT_A61` を使用しています。

表 36. ADT_A61 のメッセージ構造から構築された BO HL7_MTADT_A61 の属性名。「ADT メッセージ」列は、メッセージ構造体定義内の各属性の説明です。

HL7 メッセージ構造 ADT^A61^ADT^A61	ADT メッセージ	メッセージ BO の属性名
MSH	メッセージ・ハンドラー	Message Header
EVN	イベント・タイプ	Phenotype
PID	患者の識別番号	Patient Identification
[PD1]	付加的所在情報	Additional Demographics
PV1	患者の来院	Potentialities
[{ROL}]	役割	役割
[PV2}	患者の来院 (追加情報)	Patient Visit Additional Info

グループ BO

グループ BO のプロパティー名の属性は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  HLT_MT<message type>_<Event code/query id>_GP<group index>
  ここで、message type/query には、一字一語法に基づく最小の値を使用したイベント・コードが適用されます。また、メッセージ構造のスコープ内では group index は一意です。つまり、グループ BO の属性名はそのタイプと同じです。
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  BO HL_MTADT_A03 内の唯一のグループの属性名は、HL_MTADT_A03 です。詳しくは、見出し『*複合データ型 BO*』の下の表を参照してください。

セグメント BO

セグメント BO のプロパティー名の属性は、以下のように要約されます。

- 対応する HL7 データ・エレメント
  - 通常の HL7 セグメントのフィールド名またはエレメント名
  - 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  <通常のセグメントのエレメント名、またはパラメーター・テーブル内のパラメーター名の中の単語の連結>
  ここで、message type/query には、一字一語法に基づく最小の値を使用したイベント・コードが適用されます。また、メッセージ構造のスコープ内では group index は一意です。つまり、グループ BO の属性名はそのタイプと同じです。
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  BO HL_SGQBD_Q21 の使用

表 37. HL7_SGQBD_Q21 の属性名および HL7 の対応部分

属性シーケンス	セグメント BO の属性名	BO セグメントの属性タイプ	HL7 の対応部分
1	MessageQueryName	HL7_DTCE	MessageQueryName (CE)
2	QueryTag	HL7_DTENDCODEDTEXT	QueryTag (ST)
3	PersonIdentifier	HL7_DTCX	PersonIdentifier
4	WhatDomains Returned	HL7_DTCX	WhatDomainsReturned

MSH のセグメントには、以下の値が含まれます。

表 38.

SEQ	MSH セグメントの属性名	エレメント名
1	フィールド区切り文字	フィールド区切り文字
2	コード化文字	コード化文字
3	送信アプリケーション	送信アプリケーション
4	送信施設	送信施設

複合データ型の BO

複合データ型 BO のプロパティー名の属性は、以下のように要約されます。

対応する HL7 データ・エレメント
CM および Varies 以外の定義データ型の HL7 コンポーネント

- 命名規則
  <コンポーネント名の通常のデータ型、通常のセグメントの「説明」列、またはパラメーター・テーブル内のパラメーター名の中の単語の連結>
  ここで、連結の際は 68 ページの表 35 にリストされた規則に従います。加えて、コンポーネントの末尾にある括弧は、コンポーネントのデータ型を示すのに使用されますが、フィルターにより除外されます。
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  BO HL7_DTCP の使用。この BO は、CP データ型に対応していて、次のように記述されます。
  `<price (MO)> ^ <price type (ID)> ^ <from value (NM)> ^ <to value (NM)> ^ <range units (CE)> ^ <range type (ID)>`

表 39. CP データ型のコンポーネント名から派生した、BO 複合データ型 HL7_DTCP の属性名

HL7_DTCP の属性名	CP データ型のコンポーネント名
Price	Price
PriceType	Price type
FromValue	From value
ToValue	To value
RangeUnits	Range units
RangeType	Range type

データ型 Union BO

Union BO のプロパティー名の属性は、以下のように要約されます。

HL7 データ・エレメントに対応しています。

- CM および Varies 以外の HL7 データ型

- 61 ページの表 28にリストされたデータ構造体
- プリミティブ・データ型
- 命名規則
  <複合データ型の名前>
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  タイプ HL7_DTCP の属性名は HL7_DTCP、属性 NM の名前は NM です。

BO の属性プロパティー・タイプ

一般的なビジネス・オブジェクトの属性プロパティー・タイプについて、以下に詳述します。

メッセージ BO

メッセージ BO のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。
- 命名規則
  <対応するセグメント・グループまたはセグメントの BO 名>
  BO の名前の付け方の詳細は、 63 ページの『メッセージ BO』 を参照してください。
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  属性タイプの名前については、68 ページの表 36の BO HL7_MTADT_A61を参照してください。

グループ BO

グループ BO のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  <セグメント・グループ内のセグメント BO の名前>
  完全なリストについては、 59 ページの『メッセージ BO』を参照してください。
- マーク付け規則
  適用外

- 文法
  適用外
- 例:
  68 ページの表 36の BO HL7_MTADT_A61 を参照してください。

セグメント BO

セグメント BO のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - 通常の HL7 セグメントのフィールド
  - 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  <複合データ型または適切なプリミティブ・データ型の BO 名>
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  69 ページの表 37 を参照してください。

複合データ型の BO

複合データ型 BO のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - 通常の HL7 セグメントのフィールド
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
- 命名規則
  <サブコンポーネントに対応する複合データ型、または適切なプリミティブ・データ型の BO 名>
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  65 ページの表 31 を参照してください。

データ型 Union BO

Union BO データ型のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の HL7 データ型
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
  - プリミティブ・データ型

- 命名規則
  <サブコンポーネントに対応する複合データ型、または適切なプリミティブ・データ型の BO 名>
- マーク付け規則
  適用外
- 文法
  適用外
- 例:
  タイプ HL7_DTCP の属性名は HL7_DTCP です。タイプ NM の属性名は NM です。

BO の属性プロパティー Iskey

一般的なビジネス・オブジェクトの属性プロパティー Iskey について、以下に詳述します。

メッセージ BO

メッセージ BO のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。
- 命名規則
  適用外
- マーク付け規則
  メッセージ BO の最初の属性については、キーを true に設定します。
- 例:
  適用外

グループ BO

グループ BO のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  適用外
- マーク付け規則
  グループ BO の最初の属性については、キーを true に設定します。
- 例:
  適用外

セグメント BO

セグメント BO 内のキー属性は、通常のセグメントから派生するものであり、何の意味もありません。これらのキー属性は、アーキテクチャーのシステム要件を満たすためにのみ存在していますが、QPD、RDT セグメント内のパラメーターにとってはきわめて重要な存在です。

- HL7 データ・エレメントに対応しています。
- 通常の HL7 セグメントのフィールド
- 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  適用外
- マーク付け規則
  - タイプ QPD、RDT のセグメントを除くすべてのセグメント BO については、最初の属性キーを true に設定します。
  - タイプ QPD、RDT のセグメントについては、対応するパラメーターが「Key」とマークされた属性のキーを true に設定します。
- 文法
  適用外
- 例:
  適用外

複合データ型の BO

複合データ型の BO 内のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の定義データ型の HL7 コンポーネント
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
- 命名規則
  適用外
- マーク付け規則
  - 複合データ型のすべての BO (61 ページの表 28 にリストされたデータ型、およびプリミティブ型の属性を含む)。
  - 最初の属性については、IsKey プロパティーを true に設定します。
- 文法
  適用外
- 例:
  適用外

BO 属性プロパティー IsForeignKey

一般的なビジネス・オブジェクトの属性プロパティー IsKey について、以下に詳述します。

メッセージ BO

メッセージ BO のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。

- 命名規則
  適用外
- マーク付け規則
  IsForeignKey を false に設定します。
- 文法
  適用外
- 例
  適用外

グループ BO

- HL7 データ・エレメントに対応しています。
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  適用外
- マーク付け規則
  IsForeignKey を false に設定します。
- 例:
  適用外

セグメント BO

- HL7 データ・エレメントに対応しています。
  - 通常の HL7 セグメントのフィールド
  - 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  適用外
- マーク付け規則
  IsForeignKey を false に設定します。
- 文法
  適用外
- 例:
  適用外

複合データ型の BO

複合データ型の BO 内のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
-
  - CM および Varies 以外の定義データ型の HL7 コンポーネント
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
- 命名規則
  適用外

- マーク付け規則
  IsForeignKey を false に設定します。
- 文法
  適用外
- 例:
  適用外

データ型 Union BO

Union BO データ型のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の HL7 データ型
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
  - プリミティブ・データ型
- 命名規則
  適用外
- マーク付け規則
  IsForeignKey を false に設定します。
- 文法
  適用外
- 例:
  適用外

BO 属性プロパティーのカーディナリティー

一般的なビジネス・オブジェクトの属性プロパティー Cardinality について、以下に詳述します。

メッセージ BO

メッセージ BO のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。
- 命名規則
  適用外
- マーク付け規則
  セグメント・グループまたはセグメントに {} 中括弧が含まれている場合、カーディナリティーを N に設定します。それ以外の場合は 1 に設定します。
- 例:
  メッセージ・タイプ ADT およびイベント・タイプ A61 用の ADT_A61 HL7 メッセージ構造に対応するメッセージ BO には、以下の属性が備わっています。

表 40. この例は、HL7_MTADT_A61 の属性および HL7 の対応部分それぞれの属性カーディナリティーを示しています。

属性シーケンス	属性タイプ	HL7 の対応部分	カーディナリティー
1	HL7_SGMSH	MSH	1
2	HL7_SGEVN	EVN	1
3	HL7_SGPID	PID	1
4	HL7_SGPD1	[PD1]	1
5	HL7_SGV1	PV1	1
6	HL7_SGROL	[{ ROL }]	N
7	HL7_SGPV2	[PV2]	1

グループ BO

- HL7 データ・エレメントに対応しています。
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  適用外
- マーク付け規則
  IsForeignKey を false に設定します。
- 例:
  適用外

セグメント BO

- HL7 データ・エレメントに対応しています。
  - 通常の HL7 セグメントのフィールド
  - 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  適用外
- マーク付け規則
  セグメント・フィールドの RP# または RP 列が ″Y″ または ″y″ でマークされている場合、カーディナリティーを N に設定します。それ以外の場合は 1 に設定します。
- 文法
  適用外
- 例:
  BO HL7_SGQBD_Q21。HL7 メッセージ標準仕様のカーディナリティー属性 (クエリー Q21 のパラメーター定義用)。

表 41. この例は、HL7_SGQBD_Q21 の属性および HL7 の対応部分それぞれの属性カーディナリティーを示しています。

属性シーケンス	シーケンス BO の属性名	RP#/RP	カーディナリティー
1	MessageQueryName		1

表 41. この例は、HL7_SGQBD_Q21 の属性および HL7 の対応部分それぞれの属性カーディナリティーを示しています。 (続き)

属性シーケンス	シーケンス BO の属性名	RP#/RP	カーディナリティー
2	QueryTag		1
3	PersonIdentifier	N	1
4	WhatDomainReturned	Y	N

複合データ型の BO

複合データ型の BO 内のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の定義データ型の HL7 コンポーネント
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
- 命名規則
  適用外
- マーク付け規則
  カーディナリティーを 1 に設定します。
- 文法
  適用外
- 例
  適用外

データ型 Union BO

Union BO データ型のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の HL7 データ型
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
  - プリミティブ・データ型
- 命名規則
  適用外
- マーク付け規則
  カーディナリティーを 1 に設定します。
- 文法
  適用外
- 例:
  適用外

BO 属性プロパティー MaxLength

一般的なビジネス・オブジェクトの属性プロパティー MaxLength について、以下に詳述します。

アーキテクチャー BO は、BO サイズの表記がないうえ、HL7 データ・エレメントはすべて特性が強く現れて、単なるプリミティブ・タイプのデータ・エレメントではなくなっているため、最大長の表記は MaxLength BO 属性プロパティーによる従来の方法では伝えることができません。この属性プロパティーは、HL7 データ・エレメントの最大長の表記統一を達成するという目的に、HL7 データ・エレメント表記の全スコープの BO 内に使用されているわけではありません。代わりに、HL7 データ・エレメントの最大長の表記は、AppSpecificInfo を介して伝えられるようになりました。このバージョンのデータ・ハンドラーは検証をサポートしていないため、AppSpecificInfo プロパティーとしての Maxlength は、今回のリリースでは BO のどの部分にも現れません。

MaxLength は、一定サイズの BO を特定ブローカー用の通信チャネル内に格納できるかどうかの決定要因となるため、この属性プロパティーに正しい値を設定することは従来と同様に重要です。特に言及されていない場合、どのプリミティブ属性についても、この属性プロパティーのデフォルト値は 255 になります。

BO の属性プロパティー IsRequired

一般的なビジネス・オブジェクトの属性プロパティー IsRequired について、以下に詳述します。

メッセージ BO

- HL7 データ・エレメントに対応しています。
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。
- 命名規則
  適用外
- マーク付け規則
  HL7 メッセージ構造体定義のセグメントまたはセグメント・グループが大括弧 [] で囲まれている場合は、IsRequired を false に設定します。[] で囲まれていない場合は、このプロパティーを true に設定します。
- 例:
  メッセージ・タイプ ADT およびイベント・タイプ A61 用の ADT_A61 HL7 メッセージ構造に対応するメッセージ BO には、以下の属性が備わっています。

表 42. この例は、HL7_MTADT_A61 の属性および HL7 の対応部分それぞれの属性カーディナリティーを示しています。

属性シーケンス	属性タイプ	HL7 の対応部分	IsRequired
1	HL7_SGMSH	MSH	True
2	HL7_SGEVN	EVN	True
3	HL7_SGPID	PID	True
4	HL7_SGPD1	[PD1]	False

表 42. この例は、HL7_MTADT_A61 の属性および HL7 の対応部分それぞれの属性カーディナリティーを示しています。 (続き)

属性シーケンス	属性タイプ	HL7 の対応部分	IsRequired
5	HL7_SGV1	PV1	True
6	HL7_SGROL	[{ ROL }]	False
7	HL7_SGPV2	[PV2]	False

グループ BO

- HL7 データ・エレメントに対応しています。
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  適用外
- マーク付け規則
  セグメント・グループまたはセグメント BO に [] 中括弧が含まれている場合は IsRequired を false に設定します。[] が含まれていない場合は true に設定します。
- 例:
  適用外

セグメント BO

- HL7 データ・エレメントに対応しています。
  - 通常の HL7 セグメントのフィールド
  - 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  適用外
- マーク付け規則
  セグメント・フィールドの OPT 列またはクエリー・パラメーターが O に設定されていない場合は、対応する属性の IsRequired プロパティーを false に設定します。O に設定されている場合は true に設定します。
- 文法
  適用外
- 例:
  BO HL7_SGQBD_Q21。HL7 メッセージ標準仕様の IsRequired 属性 (クエリー Q21 のパラメーター定義用)。

表 43. この例は、HL7_SGQBD_Q21 の IsRequired 属性および HL7 の対応部分を示しています。

属性シーケンス	シーケンス BO の属性名	OPT	IsRequired
1	MessageQueryName	R	True
2	QueryTag	R	True
3	PersonIdentifier	R	True
4	WhatDomainReturned	O	False

複合データ型の BO

複合データ型の BO 内のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の定義データ型の HL7 コンポーネント
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
- 命名規則
  適用外
- マーク付け規則
  IsRequired を false に設定します。
- 文法
  適用外
- 例:
  適用外

データ型 Union BO

Union BO データ型のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の HL7 データ型
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
  - プリミティブ・データ型
- 命名規則
  適用外
- マーク付け規則
  IsRequired を false に設定します。
- 文法
  適用外
- 例:
  適用外

BO の属性プロパティー Relationship

一般的なビジネス・オブジェクトの属性プロパティー Relationship について、以下に詳述します。

メッセージ BO

- HL7 データ・エレメントに対応しています。
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。
- 命名規則
  適用外

- マーク付け規則
  Relationship を Containment に設定します。
- 例:
  適用外

グループ BO

- HL7 データ・エレメントに対応しています。
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  適用外
- マーク付け規則
  Relationship を Containment に設定します。
- 例:
  適用外

セグメント BO

- HL7 データ・エレメントに対応しています。
  - 通常の HL7 セグメントのフィールド
  - 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  適用外
- マーク付け規則
  Relationship を Containment に設定します。
- 文法
  適用外
- 例:
  適用外

複合データ型の BO

複合データ型の BO 内のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の定義データ型の HL7 コンポーネント
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
- 命名規則
  適用外
- マーク付け規則
  属性タイプが BO の場合は、Relationship を Containment に設定してください。
  属性タイプが BO 以外の場合は、定義ファイルを処理する際に属性プロパティーを除外してください。
- 文法
  適用外

- 例:
  適用外

データ型 Union BO

Union BO データ型のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の HL7 データ型
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
  - プリミティブ・データ型
- 命名規則
  適用外
- マーク付け規則
  属性タイプが BO の場合は、Relationship を Containment に設定してください。
  属性タイプが BO 以外の場合は、定義ファイルを処理する際に属性プロパティーを除外してください。
- 文法
  適用外
- 例:
  適用外

BO の属性プロパティー AppSpecificInfo

一般的なビジネス・オブジェクトの属性プロパティー Relationship について、以下に詳述します。

メッセージ BO

- HL7 データ・エレメントに対応しています。
  HL7 メッセージ構造体定義またはクエリー構造体定義を構成するセグメントまたはセグメント・グループ内の属性。
- 命名規則
  適用外
- マーク付け規則
  適用外
- 例:
  適用外

グループ BO

- HL7 データ・エレメントに対応しています。
  複数のセグメントを持つ特定のセグメント・グループのセグメントまたはセグメント・サブグループ。
- 命名規則
  適用外

- マーク付け規則
  適用外
- 例:
  適用外

セグメント BO

- HL7 データ・エレメントに対応しています。
  - 通常の HL7 セグメントのフィールド
  - 適合ステートメントの QPD、RDT セグメントのパラメーター名
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  - CM 以外のすべてのデータ型: `StructType=DataType;DataTypeID=<HL7 データ型 ID>`
  - 61 ページの表 28 (`StructType=DataType;DataTypeID=CM`) にリストされた DT BO
- 例:
  - ST データ型に対応する属性は、AppSpecificInfo が `StructType=DataType;DataTypeID=ST; MaxLength=199` です。
  - 61 ページの表 28 に定義されている属性型が HL7_DTMSG の場合、AppSpecificInfo は `StructType=DataType;DataTypeID=CM` です。

複合データ型の BO

複合データ型の BO 内のキー属性には何の意味もありません。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の定義データ型の HL7 コンポーネント
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
- 命名規則
  適用外
- マーク付け規則
  属性タイプが BO の場合は、Relationship を Containment に設定してください。
  属性タイプが BO 以外の場合は、定義ファイルを処理する際に属性プロパティーを除外してください。
- 文法
  - CM、MA、NA、QIP および Varies 以外のすべてのデータ型 `StructType=DataType;DataTypeID=<HL7 データ型 ID>`
  - 61 ページの表 28 (`StructType=DataType;DataTypeID=CM`) にリストされた DT BO
  - MA、NA および QIP 型 BO

– Varies データ型の共用体 BO StructType=Union;TypeContext=<BO 構造体の中のデータ型が指定されている位置にある参照パス>

TypeContext は BO 構造体の中のどこにあるデータ型アナウンサー情報 (例えばデータ・キャリアの場合は CM や NM など)を取得すべきかを示すために使用されます。

パスは、ルートが最上位の親 BO となっている場合は、相対パスにも絶対パスにもできます。

RDT セグメント定義は、各列ごとにユーザーが定義を与える必要があり、Varies データ型の列を構成していないという理由から、絶対パスをサポートする必要性は認められません。今回のリリースでは相対パスだけがサポートされています。

今回のリリースでは、相対パス名はデータ型アナウンサーのいずれかの BO 属性名として定義されるようになりました。

- 例:

表 44. この例は、HL7_DTCQ の属性および HL7 の対応部分それぞれの属性 AppSpecificInfo を示しています。

属性シーケンス	属性名	属性タイプ	ASI
1	数量	Float	DataTypeID=NM
2	単位	HL7_DTCE	DataTypeID=CM

BO HL7_SGOBX の ObservationValue 属性は、OBX-5 (Varies) に対応していて、OBX-2 にあるデータ型アナウンサーに依存しているため、AppSpecificInfo StructType=Union;TypeContext=ValueType が設定されています。

ここでは、データ型アナウンサーへのパスは「ValueType」です。この ValueType は単に属性名となっています。「ValueType」属性および「ObservationValue」属性は両方とも同一の BO 内にあるため、このパスは相対パスです。

データ型 Union BO

Union BO データ型のプロパティー・タイプの属性は、以下のように要約されます。

- HL7 データ・エレメントに対応しています。
  - CM および Varies 以外の HL7 データ型
  - 本書の 61 ページの表 28 にリストされた CM 型のデータ構造体のコンポーネント
  - プリミティブ・データ型
- 命名規則
  適用外
- マーク付け規則
  適用外
- 文法
  - CM 以外のすべてのデータ型: StructType=DataType;DataTypeID=<HL7 データ型 ID>

– 61 ページの表 28 (StructType=DataType;DataTypeID=CM) にリストされた CM 型 BO
– 61 ページの表 28 (StructType=Array;DataTypeID=<MA または NA のうちどちらか適切なデータ型 ID>) にリストされた MA、NA 型 BO

• 例:

属性が ST データ型に対応している場合、AppSpecificInfo は StructType=DataType;DataTypeID=ST です。61 ページの表 28 に定義されている属性型が HL7_DTMSG の場合、AppSpecificInfo は StructType=DataType;DataTypeID=CM です。

HL7 ビジネス・オブジェクト

サンプルの Healthcare ビジネス・オブジェクトとして次の 2 種類が組み込まれています。

• BIA_MO_DataHandler_Healthcare.txt
• BIA_MO_DataHandler_Hl7.txt

BIA_MO_DataHandler_Healthcare.txt

BIA_MO_DataHandler_Health.txt は、WebSphereBI/connectors/Healthcare/samples/ フォルダーにあります。

```
[ReposCopy]
Version = 3.0.0
[End]

[BusinessObjectDefinition]
Name = BIA_MO_DataHandler_Healthcare
Version = 1.0.0

[Attribute]
Name = ncpdp
Type = BIA_MO_DataHandler_NCPDP
ContainedObjectVersion = 3.0.0
Relationship = Containment
Cardinality = 1
MaxLength =
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = hl7
Type = BIA_MO_DataHandler_HL7
ContainedObjectVersion = 3.0.0
Relationship = Containment
Cardinality = 1
MaxLength =
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
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]
```

BIA_MO_DataHandler_HL7.txt

BIA_MO_DataHandler_HL7.txt は、WebSphereBI/connectors/Healthcare/samples/ フォルダーにあります。

```
[ReposCopy] Version = 3.0.0 [End] [BusinessObjectDefinition] Name =
BIA_MO_DataHandler_HL7 Version = 3.0.0 [Attribute] Name = ClassName Type =
String Cardinality = 1 MaxLength = 255 IsKey = false IsForeignKey = false
IsRequired = false DefaultValue =
com.ibm.adapters.datahandlers.hl7.HL7DataHandler IsRequiredServerBound =
false [End] [Attribute] Name = BOPrefix Type = String Cardinality = 1
MaxLength = 255 IsKey = false IsForeignKey = false IsRequired = false
DefaultValue = HL7 IsRequiredServerBound = false [End] [Attribute] Name =
Representation Type = String Cardinality = 1 MaxLength = 255 IsKey = false
IsForeignKey = false IsRequired = false DefaultValue = simplified
IsRequiredServerBound = false [End] [Attribute] Name = FieldDelimiter Type
= String Cardinality = 1 MaxLength = 255 IsKey = false IsForeignKey = false
IsRequired = false DefaultValue = | IsRequiredServerBound = false [End]
[Attribute] Name = ComponentDelimiter Type = String Cardinality = 1
MaxLength = 255 IsKey = false IsForeignKey = false IsRequired = false
DefaultValue = ^ IsRequiredServerBound = false [End] [Attribute] Name =
RepetitionDelimiter Type = String Cardinality = 1 MaxLength = 255 IsKey =
true IsForeignKey = false IsRequired = false DefaultValue = ~
IsRequiredServerBound = false [End] [Attribute] Name = EscapeDelimiter Type
= String Cardinality = 1 MaxLength = 255 IsKey = true IsForeignKey = false
IsRequired = false DefaultValue = ¥ IsRequiredServerBound = false [End]
[Attribute] Name = SubcomponentDelimiter Type = String Cardinality = 1
MaxLength = 255 IsKey = false IsForeignKey = false IsRequired = false
DefaultValue = & IsRequiredServerBound = false [End] [Attribute] Name =
MTEventMap Type = String Cardinality = 1 MaxLength = 255 IsKey = false
IsForeignKey = false IsRequired = false DefaultValue =
file=C:¥x¥WebSphereAdapters¥connectors¥Healthcare¥dependencies¥
hl7¥BIA_HL7MTEventMap.cfg IsRequiredServerBound = false [End] [Attribute]
```

```
Name = I18N Type = String Cardinality = 1 MaxLength = 255 IsKey = false
IsForeignKey = false IsRequired = false DefaultValue =
file=C:¥x¥WebSphereAdapters¥connectors¥Healthcare¥dependencies¥
hl7¥BIA_HL7I18N.cfg IsRequiredServerBound = false [End] [Attribute] Name =
DummyKey Type = String Cardinality = 1 MaxLength = 255 IsKey = false
IsForeignKey = false IsRequired = false DefaultValue = dummy
IsRequiredServerBound = false [End] [Attribute] Name = DefaultVerb Type =
String Cardinality = 1 MaxLength = 255 IsKey = false IsForeignKey = false
IsRequired = false DefaultValue = Create IsRequiredServerBound = false
[End] [Attribute] Name = EnableStackTrace Type = String Cardinality = 1
MaxLength = 255 IsKey = false IsForeignKey = false IsRequired = false
DefaultValue = true IsRequiredServerBound = false [End] [Attribute] Name =
ObjectEventId Type = String Cardinality = 1 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]
```

第 4 章 トラブルシューティング

この章では、コネクターを始動または実行するときに発生する可能性がある問題について説明します。

始動時の問題

問題	考えられる処置/説明
初期化中にコネクターが予期しないエラーでシャットダウンし、次のメッセージが報告されました: `Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException...`	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 `jms.jar` を見つけることができません。`start_connector.bat` 内の変数 `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ます。
初期化中にコネクターが予期しないエラーでシャットダウンし、次のメッセージが報告されました: `Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/mq/jms/MQConnectionFactory...`	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` を見つけることができません。`start_connector.bat` 内の変数 `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ます。
初期化中にコネクターが予期しないエラーでシャットダウンし、次のメッセージが報告されました: `Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...`	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 `jndi.jar` を見つけることができません。`start_connector.bat` 内の変数 `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ます。
初期化中にコネクターが予期しないエラーでシャットダウンし、次の例外が報告されました: `java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path`	コネクターは、IBM WebSphere MQ Java クライアント・ライブラリーから必要なランタイム・ライブラリー (`mqjbnd01.dll` [NT] または `libmqjbnd01.so` [Solaris]) を見つけることができません。パスに IBM WebSphere MQ Java クライアント・ライブラリーのフォルダーが含まれていることを確認します。
コネクターから次の例外が報告されました: `MQJMS2005: failed to create MQQueueManager for ':'`	次のプロパティーの値を明示的に設定します: `HostName`、`Channel`、および `Port`。

イベント処理

問題	考えられる処置/説明
コネクターは、MQRFH2 ヘッダーを持つすべてのメッセージをデリバリーします。	MQMD WebSphere MQ ヘッダーを持つメッセージのみをデリバリーするには、出力キューの URI の名前に `?targetClient=1` を付加します。例えば、メッセージをキュー `queue://my.queue.manager/OUT` に出力する場合は、URI を `queue://my.queue.manager/OUT?targetClient=1` に変更します。詳細については、17 ページの『第 2 章 コネクターの構成』を参照してください。

問題	考えられる処置/説明
コネクターは、コネクター・メタオブジェクト内でのメッセージ・フォーマットの定義にかかわらず、デリバリー時にすべてのメッセージ・フォーマットの 8 文字を超える部分を切り捨てます。	これは WebSphere MQ MQMD メッセージ・ヘッダーの制限であり、コネクターの制限ではありません。

付録 A. コネクターの標準構成プロパティー

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

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

コネクターによっては、一部の標準プロパティーが使用されないことがあります。Connector Configurator から統合ブローカーを選択するときには、そのブローカーで実行されるアダプターについて構成する必要のある標準プロパティーのリストが表示されます。

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

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

新規プロパティーと削除されたプロパティー

以下の標準プロパティーは、本リリースで追加されました。

新規プロパティー

- XMLNameSpaceFormat

削除されたプロパティー

- RestartCount

標準コネクター・プロパティーの構成

アダプター・コネクターには、以下の 2 種類の構成プロパティーがあります。

- 標準構成プロパティー
- コネクター固有の構成プロパティー

このセクションでは、標準の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。

Connector Configurator の使用

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

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

プロパティー値の設定と更新

プロパティー・フィールドのデフォルトの長さは 255 文字です。

コネクターは、以下の順序に従って、プロパティーの値を決定します (この場合、最も番号の大きい項目が他の項目の値をオーバーライドします)。

1. デフォルト
2. リポジトリー (統合ブローカーとして WebSphere InterChange Server を使用する場合のみ)
3. ローカル構成ファイル
4. コマンド行

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

- **動的**
  変更を System Manager に保管すると、変更が即時に有効になります。コネクターが System Manager から独立してスタンドアロン・モードで稼働している場合 (例えば、いずれかの WebSphere Message Brokers と連携している場合) は、構成ファイルでのみプロパティーを変更できます。この場合、動的更新は実行できません。
- 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 (ICS のみ)
  アプリケーション固有のコンポーネントを停止して再始動しなければ、変更が有効になりません。

特定のプロパティーの更新方法を確認するには、「Connector Configurator」ウィンドウ内の**「更新メソッド」**列を参照するか、次に示す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 45 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 45. 標準構成プロパティーの要約

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	*CONNECTORNAME* /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	*CONNECTORNAME*/ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は MQ および IDL、Repository Directory は <REMOTE>
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名。	コネクター・アプリケーション名として指定された値	コンポーネント再始動	
BrokerType	ICS、WMQI、WAS			
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>
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	MQ、IDL、または JMS	JMS	コンポーネント再始動	Repository Directory がローカルの場合は、値は JMS のみ

表 45. 標準構成プロパティーの要約 (続き)

プロパティー名	指定可能な値	デフォルト値	更新メソッド	注
DuplicateEventElimination	True または False	False	コンポーネント再始動	JMS トランスポートのみ、Container Managed Events は <NONE> でなければならない
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	コンポーネント再始動	JMS トランスポートのみ
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory または CxCommon.Messaging.jms.SonicMQFactory または任意の Java クラス名	CxCommon.Messaging.jms.IBMMQSeriesFactory	コンポーネント再始動	JMS トランスポートのみ
jms.MessageBrokerName	FactoryClassName が IBM の場合は crossworlds.queue.manager を使用。FactoryClassName が Sonicの場合 localhost:2506 を使用。	crossworlds.queue.manager	コンポーネント再始動	JMS トランスポートのみ
jms.NumConcurrentRequests	正整数	10	コンポーネント再始動	JMS トランスポートのみ
jms.Password	任意の有効なパスワード		コンポーネント再始動	JMS トランスポートのみ
jms.UserName	任意の有効な名前		コンポーネント再始動	JMS トランスポートのみ
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>
ListenerConcurrency	1 から 100	1	コンポーネント再始動	Delivery Transport は MQ でなければならない
Locale	en_US、ja_JP、ko_KR、zh_CN、zh_TW、fr_FR、de_DE、it_IT、es_ES、pt_BR **注:** これは、サポートされるロケールの一部です。	en_US	コンポーネント再始動	
LogAtInterchangeEnd	True または False	False	コンポーネント再始動	Repository Directory は <REMOTE> でなければならない

表 45. 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
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない
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> でなければならない
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければならない
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければならない
PollEndTime	HH:MM	HH:MM	コンポーネント再始動	
PollFrequency	正整数 (単位: ミリ秒) no (ポーリングを使用不可にする) key (コネクターのコマンド・プロンプト・ウィンドウで文字 p が入力された場合にのみポーリングする)	10000	動的	
PollQuantity	1 から 500	1	エージェント再始動	JMS トランスポートのみ、Container Managed Events を指定
PollStartTime	HH:MM(HH は 0 から 23、MM は 0 から 59)	HH:MM	コンポーネント再始動	
RepositoryDirectory	メタデータ・リポジトリーの場所		エージェント再始動	ICS の場合は <REMOTE> に設定する。WebSphere MQ Message Brokers および WAS の場合、C:¥crossworlds¥repository に設定する

表 45. 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
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 が JMS の場合、Repository Directory が <REMOTE> の場合のみ必要
RestartRetryCount	0 から 99	3	動的	
RestartRetryInterval	適切な正数 (単位: 分) : 1 から 2147483547	1	動的	
RHF2MessageDomain	mrm、xml	mrm	コンポーネント再始動	Delivery Transport が JMS であり、かつ WireFormat が CwXML である。
SourceQueue	有効な WebSphere MQ 名	*CONNECTORNAME*/SOURCEQUEUE	エージェント再始動	Delivery Transport が JMS であり、かつ Container Managed Events が指定されている場合のみ
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	コンポーネント再始動	Delivery Transport は JMS
SynchronousRequestTimeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	Delivery Transport は JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	コンポーネント再始動	Delivery Transport は JMS
WireFormat	CwXML、CwBO	CwXML	エージェント再始動	Repository Directory が <REMOTE> でない場合は CwXML。Repository Directory が <REMOTE> であれば CwBO
WsifSynchronousRequest Timeout	0 以上の任意の数値 (ミリ秒)	0	コンポーネント再始動	WAS のみ
XMLNameSpaceFormat	short、long	short	エージェント再始動	WebSphere MQ Message Brokers および WAS のみ

標準構成プロパティー

このセクションでは、各標準コネクター構成プロパティーの定義を示します。

AdminInQueue

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

デフォルト値は CONNECTORNAME/ADMININQUEUE です。

AdminOutQueue

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

デフォルト値は CONNECTORNAME/ADMINOUTQUEUE です。

AgentConnections

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

AgentConnections プロパティーは、orb.init[] により開かれる ORB 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・タイプを指定します。オプションは ICS、WebSphere Message Brokers (WMQI、WMQIB または WBIMB) または WAS です。

CharacterEncoding

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

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

デフォルトでは、ドロップ・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、Connector Configurator に関する付録を参照してください。

ConcurrentEventTriggeredFlows

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

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

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

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

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

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

ContainerManagedEvents

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

デフォルト値は 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 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。

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

ControllerStoreAndForwardMode

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

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

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

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

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

デフォルト値は true です。

ControllerTraceLevel

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

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

DeliveryQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は CONNECTORNAME/DELIVERYQUEUE です。

DeliveryTransport

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

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

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

WebSphere MQ および IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

このプロパティーは、false に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

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

デフォルト値は 128M です。

JvmMaxNativeStackSize

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

デフォルト値は 128K です。

JvmMinHeapSize

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

デフォルト値は 1M です。

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

デフォルト値は crossworlds.queue.manager です。

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

ListenerConcurrency

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

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

Locale

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

ロケール名は、次の書式で指定します。

ll_TT.codeset

ここで、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、Connector Configurator に関する付録を参照してください。

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

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

LogAtInterchangeEnd

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

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

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

MaxEventCapacity

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

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

MessageFileName

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

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

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

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

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

デフォルト値は false です。

OADMaxNumRetry

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

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

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

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

デフォルト値は CONNECTOR/REQUESTQUEUE です。

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

RHF2MessageDomain

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

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

サンプル・ヘッダーを以下に示します。

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

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

SourceQueue

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

JMS イベント・ストアを使用する JMS 対応コネクターでの保証付きイベント・デリバリーをサポートするコネクター・フレームワークに、JMS ソース・キューを指定します。詳細については、98 ページの『ContainerManagedEvents』を参照してください。

デフォルト値は CONNECTOR/SOURCEQUEUE です。

SynchronousRequestQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE です。

SynchronousResponseQueue

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルトは CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE です。

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

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

WsifSynchronousRequest Timeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

XMLNameSpaceFormat

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

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

デフォルト値は short です。

付録 B. Connector Configurator

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

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

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

注:

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

この付録では、次のトピックについて説明します。

Connector Configurator の概要

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

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

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

- コネクターを構成するための**コネクター固有のプロパティー・テンプレート**を作成する。
- **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、ICS の場合はコラボレーションとともに使用するマップを

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

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

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

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

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

ただし**コネクター固有プロパティー**の場合は、最初にプロパティーを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクター固有プロパティーのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、111 ページの『新規テンプレートの作成』のステップに従い、テンプレートを新規に作成します。

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

Connector Configurator の始動

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

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

スタンドアロン・モードでの Configurator の実行

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

これを行うには、以下のステップを実行します。

- **「スタート」>「プログラム」**から、**「IBM WebSphere InterChange Server」>「IBM WebSphere Business Integration Toolset」>「開発」>「Connector Configurator」**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。

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

Connector Configurator を個別に実行して構成ファイルを生成してから、System Manager に接続してこの構成ファイルを System Manager プロジェクトに保存することもできます (116 ページの『構成ファイルの完成』を参照)。

System Manager からの Configurator の実行

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

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

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

既存の構成ファイルを編集するには、以下のステップを実行します。

1. 「System Manager」ウィンドウの「コネクター」フォルダーで構成ファイルを選択し、右クリックします。Connector Configurator が開き、この構成ファイルの統合ブローカー・タイプおよびファイル名が上部に表示されます。
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、『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」**とクリックします。
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、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 (Find Name)」**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、変更を行ってください。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」**パネルがリフレッシュされ、**「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、作成した以前の値が表示されます。

   「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・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」**表示で強調表示された依存プロパティーで、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 (Finish)」**をクリックします。Connector Configurator により、XML 文書として入力した情報が、Connector Configurator がインストールされている ¥bin ディレクトリーの ¥data¥app の下に保管されます。

新しい構成ファイルを作成

構成ファイルを新規に作成するには、最初に統合ブローカーを選択します。選択したブローカーによって、構成ファイルに記述されるプロパティーが決まります。

ブローカーを選択するには、以下のステップを実行します。

- Connector Configurator のホーム・メニューで、**「ファイル」>「新規」>「コネクター構成」**をクリックします。**「新規コネクター」**ダイアログ・ボックスが表示されます。
- **「Integration Broker」**フィールドで、ICS 接続、WebSphere Message Brokers 接続、WAS 接続のいずれかを選択します。
- この章で後述する説明に従って**「新規コネクター」**」ウィンドウの残りのフィールドを入力します。

また、以下の作業も実行できます。

- 「System Manager」ウィンドウで**「コネクター」**フォルダーを右クリックし、**「新規コネクターの作成」**を選択します。Connector Configurator が開き、**「新規コネクター」**ダイアログ・ボックスが表示されます。

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

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

- **「コネクター固有プロパティー・テンプレート」**を選択します。

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

  使用するテンプレートを選択し、**「OK」**をクリックします。

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

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

既存ファイルの使用

使用可能な既存ファイルは、以下の 1 つまたは複数の形式になります。

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

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

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

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

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

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

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

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

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

構成ファイルの完成

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

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

1. **「標準のプロパティー」**タブで、BrokerType プロパティーの値フィールドを選択します。ドロップダウン・メニューで、値 ICS、WMQI、または WAS を選択します。
2. 選択したブローカーに関連付けられているプロパティーが「標準のプロパティー」タブに表示されます。ここでファイルを保管するか、または 119 ページの

『サポートされるビジネス・オブジェクト定義の指定』の説明に従い残りの構成フィールドに値を入力することができます。

3. 構成が完了したら、**「ファイル」>「保管」>「プロジェクトに」**を選択するか、または**「ファイル」>「保管」>「ファイルに」**を選択します。

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

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

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

構成ファイル・プロパティーの設定

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

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

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

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

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

- 関連マップ
- リソース
- メッセージング (該当する場合)

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

標準プロパティーとコネクター固有プロパティーの違いは、以下のとおりです。

- コネクターの標準プロパティーは、コネクターのアプリケーション固有のコンポーネントとブローカー・コンポーネントの両方によって共用されます。すべてのコネクターが同じ標準プロパティーのセットを使用します。これらのプロパティーの説明は、各アダプター・ガイドの付録 A にあります。変更できるのはこれらの値の一部のみです。
- アプリケーション固有プロパティーは、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。

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

- 背景がグレーのフィールドは、標準のプロパティーを表します。値を変更することはできますが、名前の変更およびプロパティーの除去はできません。
- 背景が白のフィールドは、アプリケーション固有のプロパティーを表します。これらのプロパティーは、アプリケーションまたはコネクターの特定のニーズによって異なります。値の変更も、これらのプロパティーの除去も可能です。
- 「値」フィールドは構成できます。
- **「更新メソッド」**フィールドは通知を行うもので、構成できません。このフィールドでは、値が変更されたプロパティーをアクティブにするために必要なアクションを指定します。

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、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。
2. 値を入力するか、ドロップダウン・メニューが表示される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すると、以下のいずれかを実行することができます。
   - 変更内容を破棄し、元の値を保持したままで Connector Configurator を終了するには、**「ファイル」>「終了」**をクリックし (またはウィンドウを閉じ)、変更内容を保管するかどうかを確認するプロンプトが出されたら**「いいえ」**をクリックします。
   - Connector Configurator 内の他のカテゴリーの値を入力するには、そのカテゴリーのタブを選択します。**「標準のプロパティー」** (またはその他のカテゴリー) で入力した値は、次のカテゴリーに移動しても保持されます。ウィンドウを閉じるときに、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・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. 118 ページの『標準コネクター・プロパティーの設定』で説明したように、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

アプリケーション固有のプロパティーは、**「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、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、暗号化が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・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『コネクターの*標準構成プロパティー*』の 92 ページの『プロパティー値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

Connector Configurator の**「サポートされているビジネス・オブジェクト」**タブで、コネクターが使用するビジネス・オブジェクトを指定します。汎用ビジネス・オブ

ジェクトと、アプリケーション固有のビジネス・オブジェクトの両方を指定する必要があり、またそれらのビジネス・オブジェクト間のマップの関連を指定することが必要です。

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

ご使用のブローカーが ICS の場合

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

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

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

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

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator」ウィンドウの**「編集」**メニューから、**「行を削除」**をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. **「ファイル」**メニューから、**「プロジェクトに保管」**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・オブジェクトにエージェント・サポートがある場合、システムは、コネクター・エージェントを介してアプリケーションにデータを配布する際にそのビジネス・オブジェクトの使用を試みます。

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

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

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

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

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

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

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

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

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

ご使用のブローカーが WAS の場合

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

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

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

関係付けられたマップ (ICS のみ)

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

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

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

  場合によっては、関連マップを明示的にバインドすることが必要になります。

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

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

  以下の手順を実行して、マップを明示的にバインドします。

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

リソース (ICS)

「リソース」タブでは、コネクター・エージェントがコネクター・エージェント並列処理を使用して、同時に複数のプロセスを処理するかどうか、またどの程度処理するかを決定する値を設定することができます。
すべてのコネクターでこの機能がサポートされるわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

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

トレース/ログ・ファイル値の設定

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

ログとトレースの値を変更するには、以下の手順を実行します。

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

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

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

データ・ハンドラー

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

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

構成ファイルの保管

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

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

- System Manager から、統合コンポーネント・ライブラリーに *.con 拡張子付きファイルとして保管します。
- System Manager から、指定したディレクトリーに *.con 拡張子付きファイルとして保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。

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

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

構成ファイルの変更

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

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

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

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

構成の完了

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

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

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

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

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

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

「CharacterEncoding」および「ロケール」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サブセットのみです。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 「ロケール」 プロパティーの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太字で示される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. HL7 メッセージ構造

この付録では、HL7 Healthcare メッセージ構造について説明します。

HL7 メッセージ

HL7 メッセージは、以下のデータ・エレメントで構成されています。

メッセージ・タイプ

HL7 のメッセージ・タイプは、メッセージのビジネス目的に対する固有 ID です。各メッセージには、メッセージの目的を宣言する手段として、メッセージ・タイプ ID が含まれていなければなりません。例えば、ADT は、患者管理に対する固有のメッセージ ID です。

ただし、メッセージの構造に対しては、どちらかと言えば固有の分類ではありません。1 つのメッセージ・タイプが、複数のメッセージ構造を持つことができます。

メッセージ・タイプは、メッセージ・ヘッダー・セグメントで通知します。

メッセージ・イベント

メッセージ・イベントは、メッセージ生成のコンテキストに対する固有の ID であり、トリガーとも呼ばれます。メッセージ・イベントは、1 つの大文字と 2 つの数字で構成されます。例えば、A01 は入院/来院通知に対応し、A61 は顧問医師の変更に対応します。A01 および A61 は両方とも、ADT メッセージで使用します。

イベント・タイプは、メッセージ・ヘッダー・セグメントで通知します。

メッセージ構造

メッセージ構造は、メッセージのクラスに対するイベントとメッセージ・タイプの関連を表すために使用するデータ構造です。各メッセージ構造にも、固有の ID が含まれます。

メッセージ構造は、明確に定義された HL7 セグメントのリストで構造的に構成されています。セグメントはオプションであり、繰り返しが可能です。セグメントの繰り返し可能回数に制限はありません。

セグメントを 1 つに集約して、セグメント・グループを形成することができます。セグメント・グループもまた、繰り返しが可能です。標準規格では、セグメント・グループを {} または [] を使用して表します。この場合の {} は繰り返しを示し、[] はオプションであることを示します。

メッセージ構造定義では、{} または [] を使用して単一セグメントを囲むことができるため、単一セグメントで構成されるセグメント・グループを解釈することも可能です。ただし、データ・ハンドラーや関連する ISBO を説明するために、ここで

は、セグメント・グループという用語を、{} または [] で囲まれた複数のセグメントの集合体を表すものとして使用します。

セグメント・グループおよびメッセージ構造におけるセグメントの相対位置は、明確に定義されています。メッセージ構造レベルでのセグメントは、アトミック・データ型です。

メッセージ構造は、メッセージ・タイプとイベントの両方を使用して定義します。1 つのメッセージ・タイプに複数のイベントを関連付けることができますが、1 つのイベントには、1 つのメッセージ・タイプしか関連付けることができません。また、特定のメッセージ・タイプを持つ複数のイベントは、同じメッセージ構造に関連付けられます。例えば、イベント A01 とイベント A04 の両方を持つメッセージ・タイプ ADT は、メッセージ構造 ADT_A01 を使用します。

表 46. ADT_A61 のメッセージ構造

ADT^A61^ADT_A61	ADT メッセージ
MSH	メッセージ・ヘッダー
EVN	イベント・タイプ
[PD1]	付加的所在情報
[{ROL}]	役割
[PV2]	患者来院 - 補足情報

セグメント

セグメントは、明確に定義されたリスト属性であり、各属性は HL7 データ型を持ちます。セグメントはすべて、3 つの大文字から成るセグメント ID で始まります。セグメント属性はオプションであり、繰り返しが可能です。属性繰り返し可能回数の最大数は指定されています。さまざまなテーブルを使用して、特定の属性値の妥当性を定義します。

注: セグメント属性の相対位置は、どのように定義するかという点で重大な意味を持ちます。

各セグメントは、ASCII の復帰または x0D のいずれかのセグメント終了記号で終了します。

HL7 では、セグメント属性をフィールドと呼びます。標準では、フィールドを区切るために、区切り文字を指定します。区切り文字の値は、ユーザーによって、メッセージ・インスタンスごとに定義されます。

フィールドを繰り返す場合、標準ではユーザー定義の区切り文字を指定してします。この値は、メッセージ・インスタンスごとに個々の繰り返しフィールドへ定義されます。フィールドの繰り返しインスタンスを分離する区切り文字も用途別に定義され、この区切り文字により反復が呼び出されます。

表 47. MSH セグメント規格

SEQ	LEN	DT	OPT	RP#	TBL	ITEM	エレメント名
1	1	ST	R			00001	フィールド区切り文字
2	4	ST	R			00002	エンコード文字

表 47. MSH セグメント規格 (続き)

SEQ	LEN	DT	OPT	RP#	TBL	ITEM	エレメント名
3	180	HD	O		0361	00003	送信アプリケーション
4	180	HD	O		0362	00004	送信施設
5	180	HD	O		0361	00005	受信アプリケーション
6	180	HD	O		0362	00006	受信施設
7	26	TS	R			00007	メッセージの日付/時刻
8	40	ST	O			00008	セキュリティー
9	13	CM	R		0076/0003	00009	メッセージ・タイプ
10	20	ST	R			00010	メッセージ制御 ID
11	3	PT	R			00011	処理 ID
12	60	VID	R		0104	00012	バージョン ID
13	15	NM	O			00013	シーケンス番号
14	180	ST	O			00014	継続ポインター
15	2 ID	O			0155	00015	受諾肯定応答型
16	2	ID	O		0155	00016	アプリケーション肯定応答型
17	3	ID	O		0399	0017	国別コード
18	16	ID	O	Y	0211	00018	文字コード
19	250	CE	O			00692	メッセージの主要言語
20	20	ID	O		0356	01317	代替文字セット処理
21	10	ID	O	Y	0449	01598	適合ステートメント ID

注: 128 ページの表 47 中の「Len」は文字数の最大長、「DT」はデータ型、「OPT」はオプションであること、「RP#」はフィールド繰り返し可能回数の最大値、「TBL#」はテーブルの ID をそれぞれ意味します。テーブルには、検証対象となる正当値の有限リストが含まれます。また、「ITEM#」はこの規格全体で共通するデータ・エレメントの固有 ID を意味します。「エレメント名」は、フィールドについて簡単に説明した名前です。

多くのセグメントの構造は、静的に定義されており、複数のメッセージ構造で共用することができます。同じセグメント ID を共用しますが、異なる構造、つまり名前およびデータ型の点で異なる属性を持つセグメントもあります。QPD、QED、RCP、および QAK は、このカテゴリーのセグメントに属します。

データ型

HL7 では、データ型について多項目リストを定義しています。プリミティブ型として定義されているデータ型もあれば、複合型として定義されているデータ型もあります。複合データ型は、プリミティブ型の複数の属性で構成されます。HL7 では、複合データ型の属性を**コンポーネント**と呼びます。

例えば、以下の HD データ型は、3 つのコンポーネント (namespace ID、universal ID、および universal ID type) を使用しています。

```
<namespace ID (IS)>^<universal ID (ST)>^
  <universal ID type )ID)>
```

標準では区切り文字を指定しています。この値は、ユーザーによって、メッセージ・インスタンスごとに個々のコンポーネントへ定義されます。

特定のデータ型のコンポーネントを、より詳細に定義できる複合データ型もあります。HL7 では、このような複合データ型の属性を**サブコンポーネント**と呼びます。複合データ型は、プリミティブ型のすべての属性を持つ必要があります。

標準ではサブコンポーネントを分離する区切り文字も指定しています。この値は、ユーザーによって、メッセージ・インスタンスごとに定義されます。

コンポーネントを定義するデータ型のコンポーネント区切り文字は、サブコンポーネント区切り文字に降格します。

HL7 では、各種の区切り文字に等しい文字をエスケープするためのエスケープ文字も指定しています。これらの文字は、ユーザーによってメッセージ・インスタンスごとに定義されます。

データ型の大部分は、標準における個別のデータ構造と同様、静的に指定されています。動的な振る舞いを示すデータ型は少数ですが、特別な注意が必要です (特に CM および * データ型)。

オーダーメイド・データ型

CM は、コンポジット・データ型とも呼ばれます。このデータ型は、固有の状態ごとにあらかじめ定義されたデータ型を、異なるセグメントまたは同じセグメントの異なるフィールドで使用するカスタム構造です。CM データ型に関連したデータ構造は多数あります。

CM 型のデータ構造は、他のデータ型と大幅に異なる場合がありますが、従属先のセグメントを定義すると固定化されます。CM データ型は、設計時に明らかになります。

HL7 標準の以前のバージョンでは、CM は、オーダーメイド・データ型、つまり、配置されるローカル・サイトで定義されるデータ型を意味していました。このため、CM データ型は、すぐに利用できるように定義されたデータ型のさまざま組み合わせと順列を使用することにより、構造を無限に表すことができます。

HL7 のバージョン 2.4 に限り、CM データ型を示すラベルの付いたデータ構造に数の制限があります。

ポリモアフィック・データ型

通常使用する 2 つまたは 3 つの大文字ではなく、* 記号または「varies」データ型ラベルを持つデータ型です。

このデータ型は、すでに定義されたデータ型のいずれにもなり得る共通の振る舞いを示します。また、データ型は他の型を使用して宣言されます。例えば、OBX-2 は、このポリモアフィック・データ型の実際のデータ・キャリアである、OBX-5 というデータ型を宣言します。わかりやすくするために、ポリモアフィック・データ型のデータ型を宣言するフィールドを「データ型アナウンサー」、ポリモアフィック・データ型の実際のデータを含むフィールドを「データ・キャリア」と呼びます。

データ型アナウンサーとデータ・キャリアは、同じセグメントに共存する場合もあれば、異なるセグメントに存在する場合もあります。例えば、OBX セグメントは、データ型アナウンサーである OBX-2 フィールドと、データ・キャリアである OBX-5 フィールドを持ちます。RDF セグメントには、RCD データ型のフィールドが含まれます。RCD フィールドの各インスタンスは、RDT セグメント内のデータ・キャリアに対するデータ・アナウンサーとして機能します。RDF セグメントと RDT セグメントの関係は、関係データベースの表記述メタデータと実データの行との関係に類似します。

データ型アナウンサーとデータ・キャリアの両方が同じセグメントに存在する場合、アナウンサー・フィールドとデータ・キャリア・フィールドの相対位置が変わる場合もあります。データ型アナウンサーがデータ・キャリアより先行する場合もあれば、その逆もあります。第 8 章のセクション 8.5.2.4 では、データ型アナウンサー MFE-5 がデータ・キャリア MFE-4 の下位にある例を説明しています。

データ型アナウンサーとデータ・キャリアを個々に設定するフィールドの固定番号はありません。以下の図に、データ型アナウンサーとデータ・キャリアの関係を示します。

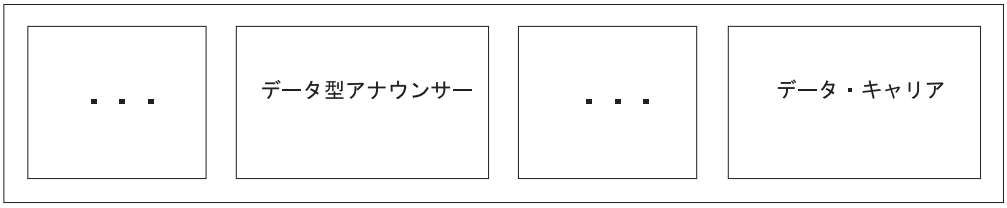

図 4. ポリモアフィック・データ型におけるデータ・キャリアとデータ型アナウンサーの関係 ("...: はセグメント内の他のフィールドを示す)

バリアント・データ型の範囲は、それがポリモアフィックであったとしても、既知であり、有限です。

区切り文字および拡張データ・モデル

セグメント、フィールド、コンポーネント、およびサブコンポーネントの区切り文字は、すべてユーザー定義されます。区切り文字は、個々のメッセージのメッセージ・ヘッダーで宣言します。

セグメント区切り文字は、ASCII 復帰または x0A として定義されます。これらの区切り文字と同一の文字がデータに含まれる場合は、エスケープ・シーケンスを使用して、その文字にマークを付けます。

コンポーネント区切り文字は、データ型が別のデータ型のコンポーネントで使用されている場合に、サブコンポーネント区切り文字に降格することがあります。ほとんどのデータ・エレメントにおいて、反復区切り文字の使い方は、データ・エレメントの性質に依存しますが、幾つかの例外があります。

反復区切り文字は、データ型エレメント・セグメント・フィールドの繰り返しインスタンス間で使用します。共通で使用する文字は、「~」です。データ型エレメントのコンポーネントは、通常、データ型が MA、NA、および QIP である場合を除き、繰り返しは想定されていません。

NA はコンポーネントとして次のように定義されます。

```
<value1> ^ <value2> ^ <value3> ^ <value4> ^ ...
```

コンポーネントそれぞれのデータ型は NM です。このデータ型を使用して、数の配列を表します。「...」は、NA が理論上、無数の値を含むことができることをを示しています。つまり、フィールドの反復を示し、反復区切り文字がフィールド区切り文字になります。以下にその例を示します。

図 5. OBX セグメントにおける NA の使用例 (太字の部分)

```
OBX|3|NA|5&WAV^^99SVL|1|0^1^2^3^4^5^6^7^8^7^6^5^4^3^2^1^0^-1^-2^-3^-4^-5^-
6^-7^-8||||||F|...<cr>
```

MA はコンポーネントとして次のように定義されます。

```
<sample 1 from channel 1 (NM)> ^ <sample 1 from channel 2
  (NM)> ^ <sample 1 from channel 3 (NM)> ...~<sample 2 from channel 1
   (NM)> ^ <sample 2 from channel 2 (NM)> ^ <sample 2 from channel 3
    (NM)> ...~ ...
```

コンポーネント区切り文字「^」によって区切られ、クラスター化されたコンポーネントのグループが繰り返され、反復はセグメント反復区切り文字「~」によって区切られる、という構造のパターンを持ちます。

この定義例では、コンポーネント・クラスター 1 つあたり 3 つのチャネルがあることを示しています。コンポーネント・クラスターは 1 つあたり 6 つまでチャネルを持つことができます。インプリメンテーション要件に応じて、コンポーネント数を増やすことも可能です。

図 6. OBX セグメントにおける MA の使用例 (太字の部分)

```
OBX|3|NA|5&WAV^^99SVL|1|0^1^2^3^4^5^6^7^8^7^6^5^4^3^2^1^0^-1^-2^-3^-4^-5^-
6^-7^-8~0^1^2^3^4^5^6^7^8^7^6^5^4^3^2^1^0^-1^-2^-3^-4^-5^-6^-7^-
8~0^1^2^3^4^5^6^7^8^7^6^5^4^3^2^1^0^-1^-2^-3^-4^-5^-6^-7^-8||||||F|...<cr>
```

NA とは異なり、コンポーネント・クラスターのサイズは有限です。

QIP はコンポーネントとして次のように定義します。`<segment field name (ST) > ^ <value1 (ST) & value2 (ST) & value3 (ST) ...>`

このデータ型の定義には、2 つのコンポーネントが含まれています。1 つは標準的なコンポーネントであり、もう 1 つは無限に繰り返し可能なサブコンポーネントです。

図 7. QIP の繰り返しサブコンポーネント (太字の部分)

```
|@PID.5.1^EVANS&Param2&Param3|
```

上記のように、データ型エレメントには、3 種類の反復があります。NA には繰り返し可能なコンポーネントが含まれます。QIP には繰り返し可能なサブコンポーネ

ントをが含まれます。MA には繰り返し可能なカスタム定義のコンポーネント・クラスターが含まれます。これらの反復はそれぞれ、異なる区切り文字を使用します。

コンポーネントが、コンポーネント区切り文字の使用により繰り返し可能であると想定した場合、コンポーネント区切り文字の降格規則をコンポーネント反復に適用すると、これらの反復は再分類されます。つまり、サブコンポーネント反復は、コンポーネント反復と同じカテゴリー内のものとして、より詳細に認識されます。

HL7 データ・エレメントの統一データ・モデルに近づくためには、従来の HL7 データ・モードを幾つか拡張する必要があります。拡張されたデータ・モードを使用することにより、古いデータ・モデルではできなかった、コンポーネントの反復が可能になります。

拡張されたモデルを使用すると、コンポーネント・クラスターと呼ばれるデータ・エレメントをデータ型エレメントに追加できます。さらに、コンポーネントとコンポーネント・クラスター・データ・エレメントの両方の繰り返しが可能になります。

以下の図に、この拡張されたデータ・モデルにおけるデータ・エレメントの階層を示します。

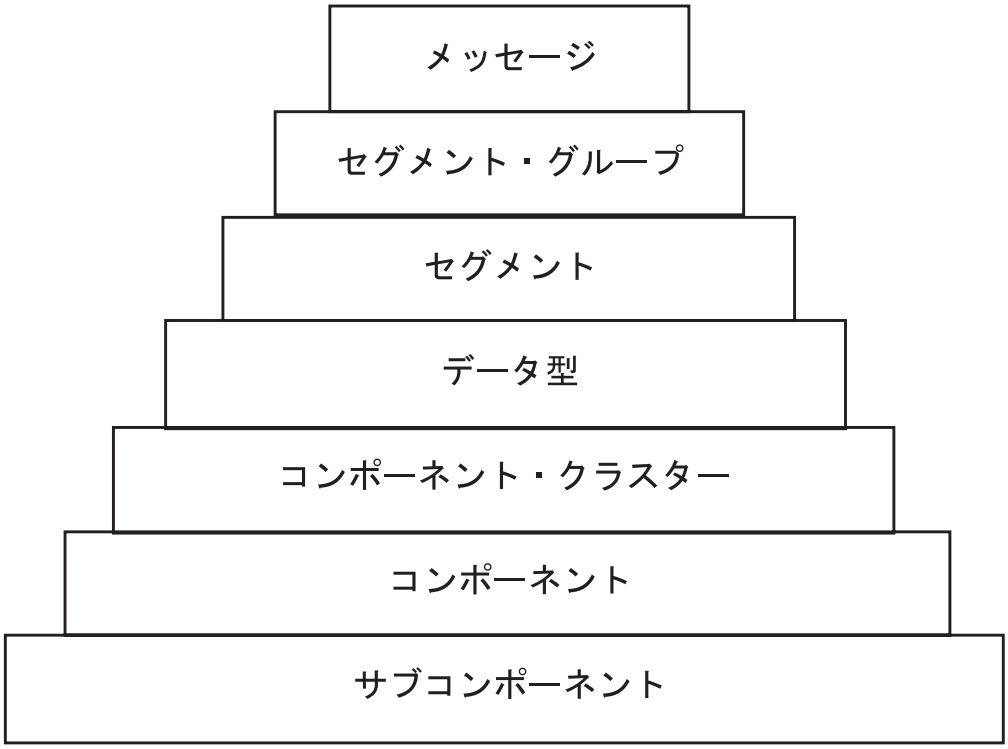

図 8. 拡張されたデータ・モデルにおけるデータ・エレメント階層

以下の表に、各データ・エレメントの区切り文字を要約します。

表 48. 各データ・エレメントの区切り文字

拡張されたデータ・モデルにおけるデータ・エレメント	反復可能度	区切り文字	説明
データ型	はい	セグメント・フィールド区切り文字 (通常 ~)	セグメント・フィールド区切り文字でデータ型エレメントの繰り返しインスタンスを区切ります。
コンポーネント・クラスター	はい	セグメント・フィールド区切り文字 (通常 ~)	セグメント・フィールド区切り文字でデータ型エレメントの繰り返しインスタンスを区切ります。コンポーネント・クラスターを含むデータ型は、繰り返し不可能である必要があります。
コンポーネント	はい	コンポーネント区切り文字 (通常 ^)	コンポーネント区切り文字でデータ型エレメントの繰り返しインスタンスを区切ります。
サブコンポーネント	はい	サブコンポーネント区切り文字 (通常 &)	サブコンポーネント区切り文字でデータ型エレメントの繰り返しインスタンスを区切ります。

国際化対応

HL7 では、メッセージごとにさまざまな文字およびエンコード方式を使用することにより、メッセージをエンコードすることができます。

以下の表に、サポートされる文字セットの詳細を示します。

表 49. アダプターで使用する文字セット ID と標準名のリスト

HL7 文字セット	説明	BIA_Healthcare Adapter でサポートする該当の標準名
ASCII	印刷可能な 7 ビット ASCII 文字集合 (このフィールドを省略した場合のデフォルト値です)。	(通常 ASCII7)
8859/1	IDO 8859/1 文字セットの印刷可能文字。	ISO8859_1
8859/	IDO 8859/2 文字セットの印刷可能文字。	ISO8859_2
8859/3	IDO 8859/3 文字セットの印刷可能文字。	ISO8859_3
8859/4	IDO 8859/4 文字セットの印刷可能文字。	ISO8859_4
8859/5	8859/5 文字セットの印刷可能文字。	ISO8859_5
8859/6	IDO 8859/6 文字セットの印刷可能文字。	ISO8859_6
8859/7	IDO 8859/7 文字セットの印刷可能文字。	ISO8859_7

表 49. アダプターで使用する文字セット ID と標準名のリスト (続き)

HL7 文字セット	説明	BIA_Healthcare Adapter でサポートする該当の標準名
8859/8	IDO 8859/8 文字セットの印刷可能文字。	ISO8859_8
8859/9	IDO 8859/9 文字セットの印刷可能文字。	ISO8859_9
ISO IR14	情報交換用のコード (1 バイト)(JIS X 0201-1976)。「ISO IR14」のように、コードにはスペースが含まれます。	JIS0201
ISO IR87	情報交換用に設定された日本語図形文字セットのコード (JIS X 0208 1990)。	JIS0208
ISO IR159	情報交換用に設定された補足的な日本語図形文字セットのコード (JIS X 0212 1990)。	JISO21‘2
UNICODE	ISO/IEC 10646-1-1993 に基づく万国共通文字規格。	UTF-8 (IDO1010646 と同じ)
ISO-IR xxxx	ISO 2375 で定められたその他の文字セット命名規則。上にリストした同じ文字セット ID とは表記が異なります。	

HL7 では、デフォルトの文字セットを、常に G0 領域で ISO IR6 (ISO 646) または ISO IR14 (JIS X 0201 1976) が指定された、1 バイト文字セットとして定義しています。

複数文字セット・データに対するエスケープ・シーケンス

メッセージは、複数の文字セットおよびエンコード・スキームを使用して、エンコードすることができます。エスケープ・シーケンスを使用してフラグを立てることにより、異なる文字セットに切り替えたり、デフォルトの文字セットに戻したりします。以下の例に、代替文字セットでエンコードされたデータについてエスケープ・シーケンスを使用する場合のデータ・パターンを示します。

```
<Escape sequence>
   <encoded data using alternate character set>
```

代替文字セットからデフォルトの文字セットに戻す場合は、デフォルトの文字セットのエスケープ・セットを使用する同じ手法を用います。

```
<Escape sequence>
  <encoded data using alternate character set> <Default escape sequence>
```

ISO 2002-1994 では、エスケープ・シーケンスを構造化する手法について定義しています。ISO 2002-1994 標準によると、エスケープ・シーケンスは、文字セットの識別に使用するビット・パターンのシーケンスであるとされています。ISO 2002-1994 では、エスケープ・シーケンスを表す場合に、10 進数 xx/yy 表記を使用しています。本書では、16 進数表記を使用するバイトのシーケンスにおけるエスケープ・シーケンスのビット・シーケンスを表します。以下は、HL7 標準規格からの引用であり、エスケープ・シーケンスを使用する場所と時期を説明したものです。

PN フィールド、XPN フィールド、XON フィールド、XCN フィールド、または XAD フィールドの各反復は、デフォルトの文字セットで始まることを前提とします。別の文字セットを使用する場合、文字セットの宣言に使用される、HL7 定義のエスケープ・シーケンスが反復の先頭にある必要があります。また、デフォルトの文字セットの開始に使用される、HL7 定義のエスケープ・シーケンスが反復の末尾にある必要があります。反復が、デフォルトの文字セットへのリターンで終了する場合は、幾つかの文字セットが単一反復内で混合することもあります。

複数文字セットに対するエスケープ・シーケンスのモード

HL7 では、エスケープ・シーケンスを指定および処理する、以下の 2 つのモードを提供しています。

- ISO 2002-1994 標準
- 2.3 (HL7 v. 2.3 で説明しています)

どちらのモードも、エスケープ・シーケンスの指定に、同じ ISO 2002-1994 手法を採用しています。この 2 つのモードの相違点は、エスケープ・シーケンスでエスケープ文字を表す方法です。

ISO 2002 モードのエスケープ・シーケンス

このモードでは、前に述べた HL7 定義のエスケープ文字ではなく、ASCII エスケープ文字を使用します。以下の表に、HL7 でサポートされる文字セットに対するエスケープ・シーケンスのリストを示します。

表 50.

エスケープ・シーケンス	HL7 で使用する文字セット
ESC 2842	ISO-IR6 G0 または ASCII (ISO 646 : ASCII))
ESC 2D41	ISO-IR100 (ISO 8859: ラテン・アルファベット 1)
ESC 2D42	ISO-IR101 (ISO 8859: ラテン・アルファベット 2)
ESC 2D43	ISO-IR109 (ISO 8859: ラテン・アルファベット 3)
ESC 2D44	ISO-IR110 (ISO 8859: ラテン・アルファベット 4)
ESC 2D4C	ISO-IR144 または 8859/5 (ISO 8859: キリル文字)
ESC 2D47	ISO-IR127 または 8859/6 (ISO 8859: アラビア語)
ESC 2D46	ISO-IR126 または 8859/7 (ISO 8859: ギリシャ語)
ESC 2D48	ISO-IR138 または 8859/8 (ISO 8859: ヘブライ語)
ESC 2D4D	ISO-IR148 または 8859/9 (ISO 8859: ラテン・アルファベット 5)
ESC 284A	ISO-IR14 (JIS X 0201 -1976: ローマ字)
ESC 2949	ISO-IR13 (JIS X 0201 : カタカナ)
ESC 2442	ISO-IR87 (JIS X 0208: 漢字、ひらがな、およびカタカナ)
ESC 242844	ISO-IR159 (JIS X 0212: 補助漢字)

HL7 で定義されたモードのエスケープ・シーケンス

別の文字セットを使用する場合は、反復の先頭に、HL7 定義のエスケープ・シーケンスを使用する必要があります。また、反復の末尾には、デフォルトの文字セットの開始に使用される、HL7 定義のエスケープ・シーケンスを使用する必要があります。

エスケープ・シーケンスは、1 つの文字と、ゼロ個または 1 つ以上のデータ文字から成るエスケープ・コード ID の両側をエスケープ文字で囲んだものです。このエスケープ文字は、ユーザーによってメッセージ内で定義されます。

以下の表に、HL7 定義のエスケープ文字として「¥」を使用する文字セットごとにエスケープ・シーケンスをリストします。

表 51. HL7 でサポートする文字セットに対するエスケープ・シーケンスのリスト

エスケープ・シーケンス	HL7 で使用する文字セット
¥C2842¥	ISO-IR6 G0 または ASCII (ISO 646 : ASCII))
¥C2D41¥	ISO-IR100 (ISO 8859: ラテン・アルファベット 1)
¥C2D42¥	ISO-IR101 (ISO 8859: ラテン・アルファベット 2)
¥C2D43¥	ISO-IR109 (ISO 8859: ラテン・アルファベット 3)
¥C2D44¥	ISO-IR110 (ISO 8859: ラテン・アルファベット 4)
¥C2D4C¥	ISO-IR144 または 8859/5 (ISO 8859: キリル文字)
¥C2D47¥	ISO-IR127 または 8859/6 (ISO 8859: アラビア語)
¥C2D46¥	ISO-IR126 または 8859/7 (ISO 8859: ギリシャ語)
¥C2D48¥	ISO-IR138 または 8859/8 (ISO 8859: ヘブライ語)
¥C2D4D¥	ISO-IR148 または 8859/9 (ISO 8859: ラテン・アルファベット 5)
¥C284A¥	ISO-IR14 (JIS X 0201 -1976: ローマ字)
¥C2949¥	ISO-IR13 (JIS X 0201 : カタカナ)
¥M2442¥	ISO-IR87 (JIS X 0208: 漢字、ひらがな、およびカタカナ)
¥M242844¥	ISO-IR159 (JIS X 0212: 補助漢字)

16 進数のエスケープ・シーケンスおよびローカル・シーケンス

HL7 では、`¥Xdddd...¥` の形式でエンコードされたバイナリー・データを転送することができます。この場合の `dddd` は、ASCII 文字の 1 から 9、および A から F を使用して示されるバイナリー値のペアを表します。これは 16 進数のエスケープ・シーケンスです。

また、HL7 コミュニケーションの関係者間における相互契約について、ユーザーは、`¥Zdddd...¥` 形式のカスタム・エスケープ・シーケンスを使用することにより、自分のデータをエンコードすることもできます。この場合の `dddd` は、TX データ型で許可された有効な文字です。

その他のエンコード・スキーム

文字セット・エンコード方式の他にも、HL7 メッセージ標準で使用するエンコード・スキームがあります。

場合によっては、メッセージが他のシステムのデータを参照することもあります。HL7 では、ユーザーが 2 つの方法 (参照による方法と値による方法) でデータにアクセスできるようになっています。参照によるデータの引き渡しを実行すると、受信システムの Reference Pointer (RP) データ型を使用して、参照データが追跡されます。ただし、受信システムへのワイヤーをまたいだ物理的なデータ転送は行われません。

受信システムで、データの実コピーを取得する必要がある場合は、データを受信システムに転送する必要があります。サード・パーティー・アプリケーションからのこのデータが、HL7 メッセージ構造の役割に従っていないことがよくあります。この種のデータを HL7 メッセージに適合させるには、特別なエンコード・スキームが必要です。

特殊なデータをエンコードするための方法として、Hex および Base64 の 2 つがあります。

- Hex エンコード方式では、バイナリー・データの連続的な単一オクテットを表すために 16 進数字の連続ペアを使用します。
- Base64 エンコード方式は、MIME 規格 RFC 1521 に準拠します。この方式では、4 つの連続する ASCII 文字を使用し、値を文字に直接置換する方法をとることにより、バイナリー・データの 3 つの連続するオクテットを表します。

以下の表は、ASCII 検索に対する値の HL7 テーブル 0290 の正確なコピーです。

表 52. Base64 MIME エンコード・スキームでの ASCII 値に対するバイナリーの HL7 テーブル 0290

値	コード	値	コード	値	コード	値	コード
3	D	20	U	37	l	54	54 2
4	E	21	V	38	m	55	55 3
5	F	22	W	39	n	56	56 4
6	G	23	X	40	o	57	57 5
7	H	24	Y	41	p	58	58 6
8	I	25	Z	42	q	59	59 7
9	J	26	a	43	r	60	60 8
10	K	27	b	44	s	61	61 9
11	L	28	c	45	t	62	62 +
12	M	29	d	46	u	63	63 /
13	N	30	e	47	v		
14	O	31	f	48	w	(埋め込み)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

表示データ

HL7 では、表示レンダリングを目的とする、フォーマットされたテキスト (FT) データの交換も許可しています。FT データは、マシンではなく、ユーザーにとって読みやすいレポートを転送するために使用されます。

表示データには、組み込み表示命令が含まれます。HTML のように、受信 HL7 アプリケーションに対するデータのレンダー方法についての命令として機能する事前定義タグのグループがあります。表示命令は明確に定義されていますが、表示命令を FT ストリングに組み込むロケーションは、完全にレポートの設計主導です。そのため、事前に明らかになることはありません。

メッセージ構造規則

メッセージに対して定義された順序でセグメントを構成してください。各メッセージは次のように構成されます。

1. 最初の 3 文字は、セグメント ID コードです。
2. シーケンスの各データ・フィールドは、以下の方法でセグメントに挿入されます。
   a. フィールド区切り文字をセグメント内に入力します。
   b. 値が存在しない場合は、これ以上文字を入力する必要はありません。
   c. 値は存在するが、その値が null の場合は、文字 "" (2 つの連続する二重引用符) をフィールドに入力します。
   d. それ以外の場合は、値の文字をセグメントに入力します。文字は、データ・フィールドに対して定義された最大数まで入力できます。フィールドを固定長まで埋め込む必要はありません。またそれは望ましくありません。ただし、フィールドを固定長まで埋め込むことは可能です。
   e. フィールド定義で、フィールドをコンポーネントに分ける必要がある場合は、以下の規則を使用します。
      1) 複数のコンポーネントが含まれている場合は、コンポーネント分離文字でコンポーネントを分離します。
      2) 存在はするが、値が null のコンポーネントは、文字 "" で表します。
      3) 存在しないコンポーネントは、コンポーネントに文字を含めないものとして処理します。
      4) 存在しないコンポーネントは、コンポーネントの末尾にコンポーネント分離文字を使用する必要がありません。例えば、以下の 2 つのデータ・コンポーネントは同じです。

         ```
         |ABC^DEF^^| and |ABC^DEF|
         ```

   f. コンポーネント定義で、1 つのサブコンポーネントを複数のサブコンポーネントに分ける必要がある場合は、以下の規則を使用します。
      1) 複数のサブコンポーネントが含まれる場合は、コンポーネント分離文字でサブコンポーネントを分離します。
      2) 存在はするが、値が null のサブコンポーネントは、文字 "" で表します。
      3) 存在しないサブコンポーネントは、コンポーネントに文字を含めないものとして処理します＞
      4) 存在しないサブコンポーネントは、コンポーネントの末尾にコンポーネント分離文字を使用する必要がありません。例えば、以下の 2 つのデータ・コンポーネントは同じです。

         ```
         ^XXX&YYY&&^ and ^XXX&YYY^
         ```

   g. フィールド定義でフィールドの反復を許可すると、反復分離文字は、複数のオカレンスが伝送される場合にのみ使用されます。このようなケースでは、

反復分離文字は、オカレンス間に挿入されます。つまり、3 つのオカレンスが伝送される場合は、2 つの反復分離文字が使用されます。
以下の例では、電話番号を構成する 2 つのオカレンスが送信されています。

```
|234-7120~599-1288B1234
```

送信対象のフィールドが残っている間は、ステップ 1b を繰り返します。セグメント定義に残存するデータ・フィールドが 1 つもない場合、これ以上の区切り文字の使用は必要ありません。

以下の規則は、HL7 メッセージの受信およびデータ値へのメッセージ内容の変換に適用します。

1. セグメント、フィールド、コンポーネント、サブコンポーネント、および存在はするが、予想していなかったフィールドの追加反復は無視します
2. 予想はしていたが、存在しないセグメントは、存在しないフィールドのみで構成されるものとして処理します。
3. 予想しているが、セグメントに含まれなかったフィールドおよびコンポーネントは、存在しないものとして処理します。

付録 D. ビジネス・オブジェクト最小抽出ユーティリティー

ビジネス・オブジェクト最小抽出ユーティリティーは Java ベースのバッチ・ツールであり、このツールを使用すると、業界固有ビジネス・オブジェクトのマスター・ファイルから、ビジネス・オブジェクト定義のサブセットを抽出することができます。これは、BO Designer の代わりに利用できます。このユーティリティーで処理できるのは、名前と値のペアを含むテキスト・ファイルのみです。また、このユーティリティーは、以下のいずれかの用途で実行できます。

- ビジネス・オブジェクト定義を個々に抽出する。この場合、マスター・ファイルに対して抽出ユーティリティーが連続的に複数回実行されてから、ビジネス・オブジェクト定義の小規模なコレクションが抽出されることもあります。子オブジェクトは、親ビジネス・オブジェクト定義内に出現するたびに抽出されます。つまり、同じ子オブジェクト定義が複数回抽出される可能性があります。この用途で使用する場合は、引き数 -b を指定します。
- 特定の 1 つの入力ファイルから、複数のビジネス・オブジェクトの定義を抽出する。抽出ユーティリティーをこの用途で使用すると、1 回の実行でビジネス・オブジェクト定義の小規模なコレクションを抽出できます。複数の親オブジェクトが同じ子オブジェクトを共用している場合、抽出ユーティリティーはその子オブジェクトの定義の出現回数を 1 回に減らして、重複をなくします。この用途で使用する場合は、引き数 -f を指定します。

  引き数 -b を指定して抽出ユーティリティーを実行したときに、数回の実行の後で出力が得られた場合は、その出力に対し、引き数 -f を指定して抽出ユーティリティーを実行すると、複数回抽出された子オブジェクト定義の出現回数を 1 回に減らすことができます。
- 複数の入力ファイルにまたがってビジネス・オブジェクトの依存関係が定義されている場合に、そのビジネス・オブジェクトの最小定義を抽出する。この用途で使用する場合は、引き数 -c を指定します。

呼び出し例

次の例は、呼び出し構文と、有効なパラメーターの説明を示しています。

```
java com.ibm.crossworlds.BusObjMinimalExtractor [-h]
[-i <InputBusObjFileName>] [-b <TargetBusObject>]
[-o <BusObjectOutputFileName>]
[-d] [-w <OutputSortedFileName>] [-r <InputSortedFileName>]
[-f <BusObjListFile>] [-c <BusObjDefFileListFile>]
```

表 53. 呼び出し例のエレメントの説明

パラメーター	説明
[-h]	呼び出し構文を出力します。
[-i <InputBusObjFileName>]	抽出するビジネス・オブジェクト定義が含まれている .in ファイルを指定します。
[-b <TargetBusObject>]	定義を抽出する必要があるビジネス・オブジェクトを指定します。

表 53. 呼び出し例のエレメントの説明 (続き)

パラメーター	説明
[-o <BusObjectOutputFileName>]	抽出した定義を書き込む .in ファイルを指定します。
[-d]	オプションのブール型パラメーターです。子オブジェクトを親オブジェクトの前にリストする必要があることを示すために使用します。使用されていない場合には、親オブジェクトが先にリストされます。
[-w <OutputSortedFileName>]	<InputBusObjFileName> の .in ファイル (抽出するビジネス・オブジェクト定義が含まれるファイル) から作成されるトポロジカル・ソート済みファイルの名前を指定します。
[-r <InputSortedFileName>]	ビジネス・オブジェクト定義の抽出元にするトポロジカル・ソート済みファイルの名前を指定します。
[-f <BusObjListFile>]	定義を抽出する必要があるビジネス・オブジェクトのリストが含まれているファイルです。各ビジネス・オブジェクトの名前は、それぞれ別の行に記述されている必要があります。
[-c <BusObjDefFileListFile>]	BO 定義の抽出元にするファイルのリストが含まれているファイルの名前を指定します。

付録 E. 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。

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

IBM WebSphere Business Integration Adapter Framework V2.4.0