

**IBM WebSphere Business Integration
Adapters**



**Adapter for SAP Exchange Infrastructure (SAP XI)
ユーザーズ・ガイド**

V 2.0.x

**IBM WebSphere Business Integration
Adapters**



**Adapter for SAP Exchange Infrastructure (SAP XI)
ユーザーズ・ガイド**

V 2.0.x

お願い

本書および本書で紹介する製品をご使用になる前に、197ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Business Integration Adapter for SAP Exchange Infrastructure (SI) (5724-G84) バージョン 2.0.x 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 SAP Exchange Infrastructure (SAP XI) User Guide
V 2.0.x

発 行： 日本アイ・ピー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2004.7

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

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

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

© Copyright IBM Japan 2004

目次

本書について	v
対象読者	v
本書を読むための前提条件	v
関連文書	v
表記上の規則	vi
本リリースの新機能	vii
リリース 2.0.x の新機能	vii
第 1 章 アダプターの概要	1
Adapter for SAP XI の環境	1
用語	3
Connector for SAP XI のコンポーネント	5
Connector for SAP XI のアーキテクチャー	9
インストール、構成、および設計のチェックリスト	10
第 2 章 インストールおよび始動	13
インストール作業の概要	13
コネクタおよび関連ファイルのインストール	13
インストール済みファイルの構造	14
構成作業の概要	16
アダプターの複数インスタンスの実行	17
コネクタの始動と停止	18
第 3 章 ビジネス・オブジェクトの要件	21
ビジネス・オブジェクトのメタデータ	21
コネクタ・ビジネス・オブジェクトの構造	21
ビジネス・オブジェクトの開発	50
第 4 章 SAP XI コネクタ	51
コネクタ処理	51
HTTP(S) サービス	53
イベント処理	54
要求処理	61
SSL	67
コネクタの構成	69
始動時のコネクタ	79
ロギング	80
トレース	81
第 5 章 SOAP データ・ハンドラー	83
SOAP データ・ハンドラーの構成	83
SOAP データ・ハンドラーの処理	91
アプリケーション固有情報機能の使用	98
プラグ可能な名前ハンドラーの指定	119
制約事項	121
第 6 章 要求処理のためのコラボレーションの有効化	125
要求処理コラボレーションのチェックリスト	125

第 7 章 Web サービスとしてのコラボレーションの公開	127
手順のチェックリスト	127
ビジネス・オブジェクトの識別または開発	128
コラボレーション・テンプレートの選択または開発	128
新規コラボレーション・オブジェクトのポートのバインディング	129
WSDL 構成ウィザード	130
第 8 章 SAP XI ODA の使用	139
BIA_XIWSDLUtil ツールの実行	140
SAP XI ODA の始動	141
SAP XI ODA の実行	141
エージェントの構成	142
WSDL 文書の指定	145
選択内容の確認	145
オブジェクトの生成	146
制約事項	147
第 9 章 トラブルシューティング	149
始動時の問題	149
ランタイム・エラー	151
付録 A. コネクタの標準構成プロパティ	153
新規プロパティと削除されたプロパティ	153
標準コネクタ・プロパティの構成	153
標準プロパティの要約	155
標準構成プロパティ	160
付録 B. Connector Configurator	173
Connector Configurator の概要	173
Connector Configurator の始動	174
System Manager からの Configurator の実行	175
コネクタ固有のプロパティ・テンプレートの作成	175
新規構成ファイルの作成	178
既存ファイルの使用	179
構成ファイルの完成	181
構成ファイル・プロパティの設定	181
構成ファイルの保管	188
構成ファイルの変更	189
構成の完了	189
グローバル化環境における Connector Configurator の使用	190
付録 C. クイック・ステップ	191
要求処理	191
イベント処理	191

付録 D. HTTPS/SSL の構成 193
鍵ストアのセットアップ 193
トラストストアのセットアップ 194
公開鍵証明書用の証明書署名要求 (CSR) の生成 195

特記事項. 197
プログラミング・インターフェース情報 199
商標 199

本書について

IBM(R) WebSphere(R) Business Integration Adapter ポートフォリオは、先進の e-business テクノロジーやエンタープライズ・アプリケーションに統合コネクティビティを提供します。本書では、Web サービス用アダプターのインストール、構成、およびビジネス・オブジェクト開発について説明します。

対象読者

本書の対象読者は、IBM WebSphere の顧客、コンサルタント、開発者など、WebSphere Business Integration Adapter for SAP Exchange Infrastructure (SI) をインプリメントする人々です。

本書を読むための前提条件

本書の中では、いろいろな前提条件が挙げられています。その多くは、Web サービスに関する情報が掲載された Web サイトや Web サービスのためのリソースが提供されている Web サイトへの参照です。また、WebSphere Business Integration システムのインプリメントに関する知識も必要です。最初に「テクニカル入門 (IBM WebSphere InterChange Server)」を読むことをお勧めします。この資料には、詳細な情報が記載された文献に関する相互参照が示されています。

関連文書

この製品と共に提供される資料の完全なセットには、すべての WebSphere アダプター・インストールに共通の機能およびコンポーネントの説明と、個別のコンポーネントの参考資料が含まれています。

以下のサイトから、関連資料をインストールすることができます。

- 一般的なアダプター情報については、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>

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。

注: 本書の発行後に公開されたテクニカル・サポートの技術情報や速報に、本書の対象製品に関する重要な情報が記載されている場合があります。これらの情報は、WebSphere Business Integration Support Web サイト (<http://www.ibm.com/software/integration/websphere/support/>) にあります。関心のあるコンポーネント・エリアを選択し、「Technotes」セクションと「Flashes」

セクションを参照してください。また、IBM Redbooks (<http://www.redbooks.ibm.com/>) にもその他の有効な情報があります。

表記上の規則

本書では、以下の規則を使用します。

Courier フォント	コマンド名、ファイル名、入力する情報、システムが画面に出力する情報などのリテラル値を示します。
太字	初出語を示します。
イタリック、イタリック 青のアウトライン	変数名または相互参照を示します。 青のアウトラインは、マニュアルをオンラインで表示するときのみ見られるもので、相互参照用のハイパーリンクを示します。アウトラインの内側をクリックすることにより、参照先オブジェクトにジャンプできます。
{ }	構文内で、複数のオプションが中括弧で囲まれている場合、その中の 1 つのオプションのみを選択することが必要です。
[]	構文内で、大括弧で囲まれているパラメーターはオプションです。
...	構文内での省略符号は、直前のパラメーターの繰り返しを示します。例えば、option[,...] は、複数のオプションをコマンドで区切って指定できることを示します。
< >	命名規則において不等号括弧で囲まれた部分は、名前を構成する複数の要素の 1 つを示します。例: <server_name><connector_name>tmp.log。
/, ¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。UNIX システムの場合には、円記号はスラッシュ (/) に置き換えてください。すべての IBM 製品パス名は、製品がシステムにインストールされているディレクトリーを基準にした相対パス名です。
%text% および \$text	パーセント記号で囲まれたテキストは、Windows の text システム変数またはユーザー変数の値を示します。UNIX 環境ではこれを \$text で表します。すなわち、text UNIX 環境変数の値を示します。
ProductDir	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表します。CROSSWORLDS 環境変数には、ProductDir ディレクトリー・パスが含まれます。このディレクトリー・パスは、デフォルトでは IBM¥WebSphereAdapters になっています。
”	メニューから次のように項目を選択します。「ファイルの選択 (Choose File)」 「更新 (Update)」 「SGML リファレンス (SGML References)」

本リリースの新機能

リリース 2.0.x の新機能

このリリースには、以下の機能拡張が含まれます。

- アダプターは JMS トランスポートを使用しなくなりました。代わりに、HTTP(S) トランスポートおよび SOAP データ・ハンドラーを使用します。
- SAP XI Object Discovery Agent (ODA) を実行してビジネス・オブジェクト定義を生成する前に SAP XI WSDL ファイルを変換するには、WSDL 変換プログラム・ユーティリティー (BIA_XIWSDLUtil) を使用します。

アダプターは Solaris 7.0 プラットフォームをサポートしなくなりました。

第 1 章 アダプターの概要

- 『Adapter for SAP XI の環境』
- 3 ページの『用語』
- 5 ページの『Connector for SAP XI のコンポーネント』
- 9 ページの『Connector for SAP XI のアーキテクチャー』
- 10 ページの『インストール、構成、および設計のチェックリスト』

コネクタは、WebSphere Business Integration Adapter for SAP Exchange Infrastructure (XI) のランタイム・コンポーネントです。コネクタを使用すると、各企業は、自社の組織内部や取引先で使用するための SAP XI メッセージを集約、公開、および利用することができます。この資料で説明されているコネクタやその他のコンポーネントは、HTTP プロトコルおよび HTTPS プロトコルを介して転送できるメッセージの本文に記述されているビジネス・オブジェクトの情報を交換するために必要な機能を提供します。

この章では、WebSphere Business Integration Adapter for SAP XI のインプリメントに必要なスコープ、コンポーネント、設計ツール、およびアーキテクチャーについて説明します。また、本書で述べられている SAP XI コンポーネントをインストールおよび構成するために行う必要のあるタスクについても概説します。コンポーネントのインストールおよび構成については、10 ページの『インストール、構成、および設計のチェックリスト』を参照してください。

注: Adapter for SAP XI は標準 Adapter Framework API を実装しています。このため、アダプターは Framework がサポートする任意の統合ブローカーと連動して動作します。ただし、アダプターが提供する機能は、厳密には IBM WebSphere InterChange Server (ICS) 統合ブローカーをサポートするように設計されています。

Adapter for SAP XI の環境

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

- 『ブローカーの互換性』
- 2 ページの『ソフトウェア前提条件』
- 2 ページの『アダプターのプラットフォーム』
- 2 ページの『規格および API』
- 3 ページの『ロケール依存データ』

ブローカーの互換性

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 SAP XI バージョン 2.0 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
の以下のバージョン:
 - 2.2.0
 - 2.3.0
 - 2.3.1
 - 2.4.0
- WebSphere InterChange Server のみ、以下のバージョン:
 - 4.2
 - 4.2.1
 - 4.2.2

例外については、「リリース情報」を参照してください。

ソフトウェア前提条件

Connector for SAP XI をインストールするには、その前に以下の前提事項およびソフトウェア要件をよく確認してください。

- HTTPS/SSL を使用する場合は、鍵ストアおよびトラストストアを作成するために、ユーザー独自のサード・パーティー・ソフトウェアが必要になります。
- コネクタは、SAP Exchange Infrastructure 2.0 SR1 および SAPDB 7.3 と連携して使用する目的で設計されています。
- 次のサード・パーティー・アプリケーションが必要です。
 - Java WebStart 1.0
 - Microsoft Internet Explorer 5.01、Service Pack 1 以上

アダプターのプラットフォーム

アダプターは、以下のプラットフォーム (オペレーティング・システム) 上で動作します。

- Microsoft Windows 2000
- Solaris 8 または AIX 5.1、5.2 または HP-UX 11i

規格および API

さまざまな規格や技術の機能には、ネットワークを介してアクセスできます。

アダプターが使用する規格は、以下のとおりです。

- HTTP 1.0

アダプターが使用する API は、以下のとおりです。

- IBM JSSE 1.0.2

構成によっては、追加のソフトウェアをインストールする必要がある場合もあります。構成別の要件については、以下のセクションで説明します。

SSL

SSL の使用を計画している場合は、鍵ストア、証明書、および鍵生成を管理するために、サード・パーティーのソフトウェアを使用する必要があります。鍵ストア、

証明書のセットアップ、または鍵生成用のツールは提供されていません。Keytool (IBM JRE に同梱) の使用を選択して、自己署名証明書を作成し、鍵ストアを管理することもできます。詳しくは、67 ページの『SSL』を参照してください。

ロケール依存データ

コネクタは、2 バイト文字セットをサポートできるようにグローバル化対応されています。コネクタがある文字コード・セットを使用する場所から別の文字コード・セットを使用する場所にデータを転送する場合、文字変換を行ってデータの意味を保持します。

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

注: コネクタは、国際化対応されていません。このことは、トレースおよびログ・メッセージが変換されないということを意味します。

SAP XI コネクタ

このセクションでは、グローバリゼーションとコネクタについて説明します。

イベント通知: コネクタは、プラグ可能なプロトコル・リスナーを使用してイベント通知を行います。プロトコル・リスナーはトランスポートからメッセージを抽出し、メッセージのメタデータに指定したデータ・ハンドラーを呼び出します。リスナー処理の詳細については、55 ページの『HTTP および HTTPS プロトコル・リスナー処理』を参照してください。

要求処理: コネクタは、プラグ可能な HTTP-HTTPS プロトコル・ハンドラー・フレームワークを使用して、要求を処理します。プロトコル・ハンドラーはデータ・ハンドラーを呼び出します。詳しくは、62 ページの『HTTP-HTTPS プロトコル・ハンドラー処理』を参照してください。

データ・ハンドラー

SOAP データ・ハンドラーを使用するには、SAP XI Adapter を構成します。データ・ハンドラー構成の概要については、11 ページの『SOAP データ・ハンドラーの構成』を参照してください。

用語

本書では、以下の用語を使用します。

- **ASI (アプリケーション固有情報)** は、特定のアプリケーションまたはテクノロジーに合わせて調整されたコードです。ASI は、ビジネス・オブジェクト定義の属性レベルとビジネス・オブジェクト・レベルの両方のレベルで存在します。
- **ASBO (アプリケーション固有ビジネス・オブジェクト)** ASI を含めることのできるビジネス・オブジェクト。

- **BIA_XIWSDL 変換プログラム・ユーティリティー** SAP XI によって生成された WSDL 文書を、標準に準拠した文書に変換するツール。このツールは、SAP XI ODA を実行する前に使用します。
- **BO (ビジネス・オブジェクト)** ビジネス・エンティティー (Customer など) およびデータへの処置 (作成または更新操作など) を表す属性の集合。IBM WebSphere システムのコンポーネントは、情報を交換したりアクションを起動したりするためにビジネス・オブジェクトを使用します。
- **Content-Type** タイプサブタイプ およびオプション・パラメーターが含まれる HTTP プロトコル・ヘッダー。例えば、Content-Type 値 `text/xml; charset=ISO-8859-1` では、`text/xml` はタイプサブタイプであり、`charset=ISO-8859-1` はオプションの Charset パラメーターです。
- **ContentType** Content-Type ヘッダー値のタイプサブタイプ 部分のみを指します。例えば、Content-Type 値 `text/xml; charset=ISO-8859-1` の `text/xml` は、本書では、ContentType とみなされます。
- **GBO (汎用ビジネス・オブジェクト)** ASI を含まず、どのアプリケーションとも関連していないビジネス・オブジェクト。
- **MO_DataHandler_Default** コネクター・エージェントが、どのデータ・ハンドラーのインスタンスを生成するかを決定するために使用するデータ・ハンドラー・メタオブジェクトです。これは、コネクターの `DataHandlerMetaObjectName` 構成プロパティで指定されます。
- **MO_DataHandler_DefaultSOAPConfig** SOAP データ・ハンドラー固有の子データ・ハンドラー・メタオブジェクトです。
- **プロトコル構成 MO** HTTP-HTTPS プロトコル・ハンドラーは、要求の処理中にプロトコル構成 MO を使用して宛先を決定します。イベント処理時にコラボレーションを公開する場合、コネクターはプロトコル構成 MO を使用して、メッセージ・ヘッダー情報を HTTP/HTTPS プロトコル・リスナーからコラボレーションへ転送します。
- **SAP XI ODA (Object Discovery Agent)** ビジネス・オブジェクトを生成するための自動化ツール。詳細については、141 ページの『SAP XI ODA の始動』を参照してください。
- **SOAP ビジネス・オブジェクト** SOAP ビジネス・オブジェクトは TLO の子であり、SOAP 要求、SOAP 応答、および SOAP 障害ビジネス・オブジェクトのいずれかになります。SOAP ビジネス・オブジェクトには、SOAP 構成 MO (SOAP ビジネス・オブジェクトの子) などの、SOAP データ・ハンドラーによる処理のために必要な情報が含まれ、また、SOAP ヘッダー・コンテナ・ビジネス・オブジェクトも含まれています。
- **SOAP 構成 MO (構成メタオブジェクト)** データ・ハンドラーは、(例えば SOAP メッセージから SOAP ビジネス・オブジェクトなどへの) 単一の変換に関する構成情報を含むオブジェクトを必要とします。この情報は、SOAP ビジネス・オブジェクトの子にメタデータとして保管されます。この子オブジェクトが SOAP 構成 MO です。
- **SOAP ヘッダー子ビジネス・オブジェクト** SOAP メッセージ内の単一のヘッダー要素を表すビジネス・オブジェクト。ヘッダー要素は、SOAP メッセージの `SOAP-Env:Header` 要素の直接の子です。ヘッダー・コンテナ・ビジネス・オブジェクトのすべての属性は、この型に属しています。これらのビジネス・オブジ

エクトには、actor および mustUnderstand 属性が割り当てられることがあります。これらの属性は、SOAP ヘッダー要素の actor および mustUnderstand 属性に対応しています。

- **SOAP ヘッダー・コンテナ・ビジネス・オブジェクト** SOAP メッセージのヘッダーに関する情報が格納されているビジネス・オブジェクト。このビジネス・オブジェクトには 1 つ以上の子ビジネス・オブジェクトが含まれます。それぞれの子ビジネス・オブジェクトは、SOAP メッセージのヘッダー項目を表します。SOAP データ・ハンドラー・ビジネス・オブジェクトには、SOAP ヘッダー・コンテナ・ビジネス・オブジェクトと同じタイプの属性が割り当てられることがあります。この属性は、SOAP ヘッダー属性とも呼ばれます。このような属性は、83 ページの『第 5 章 SOAP データ・ハンドラー』で述べるように、特別なアプリケーション固有情報を必要とします。この属性は、SOAP ビジネス・オブジェクトの直接の子でなければなりません。
- **トップレベル・ビジネス・オブジェクト** トップレベル・ビジネス・オブジェクトには、1 つの要求、1 つの応答 (オプション)、および 1 つ以上の障害 (オプション) ビジネス・オブジェクトが含まれます。TLO は、コネクタがイベント処理および要求処理の両方で使用します。
- **UDDI (Universal Description, Discovery and Integration)** とは、Web サービスに関する情報の公開方法や検索方法を定義する仕様のことです。UDDI 仕様では、XML ベースのインターフェース (API) を提供することにより、UDDI 登録情報に対して、方針に基づいたアクセスが可能になります。SOAP は、これらの API の基盤となる RPC 機能です。
- **WSDL (Web サービス記述言語)** とは、Web サービス向けのソフトウェア・インターフェースを定義する XML 用語のことです。WSDL では、プログラミング・レベルでの自動的な統合に必要な Web サービス技術の詳細がすべて系統化されています。また、WSDL は、IBM WebSphere コラボレーションを Web サービスとして公開するときにも使用します。WSDL と Web サービスとの関係は、IDL と CORBA オブジェクトとの関係に相当します。

WSDL について詳しくは、次の URL にアクセスしてください。

<http://www.w3.org/TR/wsdl>

Connector for SAP XI のコンポーネント

図 1 には、Connector for SAP XI が、プロトコル・ハンドラー・フレームワークとプロトコル・リスナー・フレームワークを含む形で図示されています。

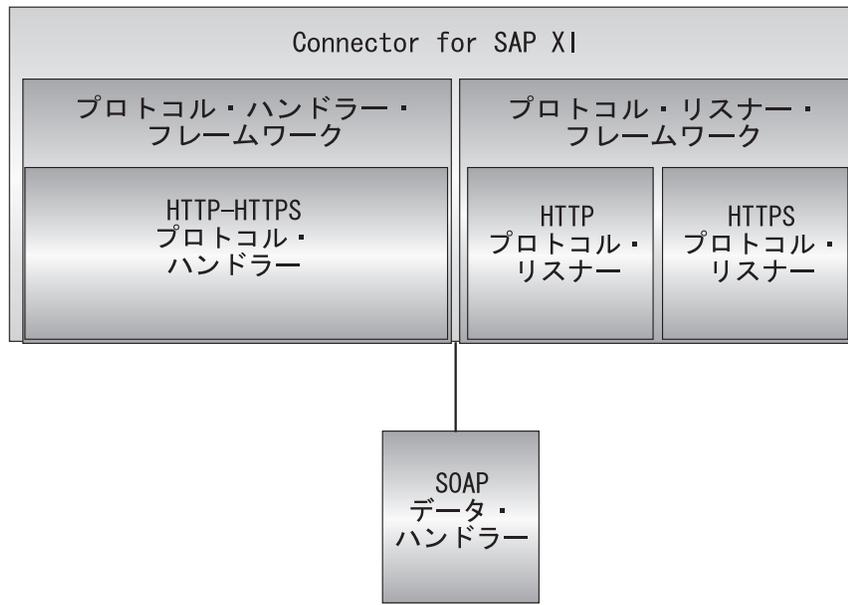


図 1. Connector for SAP XI

以下のコンポーネントは、相互に作用することにより、インターネットを介したデータ交換を可能にします。

- SOAP データ・ハンドラーおよびプロトコルのリスナーとハンドラーを含む、SAP XI サービス・コネクタ
- SAP XI 対応のコラボレーション
- ビジネス・オブジェクトおよび HTTP(S) メッセージ
- WebSphere Business Integration InterChange Server

Connector for SAP XI

コネクタは、要求の処理中にビジネス・オブジェクトを要求メッセージに変換し、指定の宛先に転送することによって、コラボレーション・サービス呼び出しに応答します。オプションで (同期要求処理の場合)、コネクタは、応答メッセージを 応答ビジネス・オブジェクトに変換し、これらをコラボレーションに戻します。

イベント処理中に、コネクタは、要求メッセージを要求ビジネス・オブジェクトに変換し、それらを処理するためにコラボレーションに引き渡すことによって、クライアントからの要求メッセージを処理します。コネクタは、オプションで応答ビジネス・オブジェクトをコラボレーションから受け取ります。これらの応答ビジネス・オブジェクトは、応答メッセージに変換されてから、クライアントに戻されます。

詳しくは、51 ページの『第 4 章 SAP XI コネクタ』を参照してください。

注: この資料でコネクタと言う場合、特に指定のない限り、SAP XI コネクタのことを指します。

プロトコル・リスナーおよびハンドラー

コネクタには、以下のプロトコル・リスナーおよびハンドラーが組み込まれています。

- HTTP プロトコル・リスナー
- HTTPS プロトコル・リスナー
- HTTP-HTTPS プロトコル・ハンドラー

プロトコル・リスナーは、SAP XI クライアントからの HTTP 形式または HTTPS 形式のイベントを検出します。プロトコル・リスナーは、コラボレーションによる処理が必要なイベントをコネクタに通知します。その後、プロトコル・リスナーは、ビジネス・オブジェクト・レベルおよび属性レベル ASI、コネクタ・プロパティ、およびプロトコル構成オブジェクトに組み込まれた変換規則を読み取って、コラボレーション、データ・ハンドラー、処理モード (同期/非同期)、およびトランザクションのトランスポート固有の性質を判別します。プロトコル・リスナー処理の詳細な説明については、54 ページの『プロトコル・リスナー』を参照してください。

プロトコル・ハンドラーは、HTTP 形式または HTTPS 形式の SAP XI サービスをコラボレーションの代わりに呼び出します。HTTP(S) プロトコル・ハンドラーは、TLO ASI と、プロトコル構成オブジェクトに組み込まれた変換規則を読み取って、要求の処理方法 (同期または非同期)、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) に使用するデータ・ハンドラー、および (要求ビジネス・オブジェクトであるプロトコル構成 MO の Destination 属性から) 宛先を決定します。同期トランザクションの場合、プロトコル・ハンドラーは、応答メッセージを処理して、応答ビジネス・オブジェクトに変換し、コラボレーションに戻します。

プロトコル・ハンドラーについて詳しくは、62 ページの『プロトコル・ハンドリング』を参照してください。

SOAP データ・ハンドラー

SOAP データ・ハンドラーは、SOAP ビジネス・オブジェクトから SOAP メッセージへの変換およびその逆の変換を行います。SOAP データ・ハンドラーについて詳しくは、83 ページの『第 5 章 SOAP データ・ハンドラー』を参照してください。

詳しくは、83 ページの『第 5 章 SOAP データ・ハンドラー』を参照してください。

構成ツール

SAP XI Adatper を配置するには、SAP XI を呼び出すコラボレーション、または SAP XI の宛先として公開されているコラボレーションを使用します。

要求処理のためにコラボレーションを使用可能にする場合は、BIA_XIWSDLUtil 変換プログラム・ユーティリティと SAP XI Object Discovery Agent (ODA) を使用して TLO を生成します。要求処理および SAP XI ODA について詳しくは、125 ページの『第 6 章 要求処理のためのコラボレーションの有効化』を参照してください。

コラボレーションを Web サービスとして公開する場合は、WSDL 構成ウィザードを使用します。このウィザードは、後で、例えば、UDDI レジストリーを介して公開する、コラボレーション用の WSDL 文書の生成に役立ちます。コネクタは、この情報を公開するためのツールを備えていません。コラボレーションを SAP XI

メッセージの宛先として公開する方法については、127ページの『第7章 Web サービスとしてのコラボレーションの公開』を参照してください。

コネクターの配置

SAP XI コネクターには、外部クライアントからの着信メッセージ、または外部クライアントへの発信メッセージを管理するためのゲートウェイまたはフロントエンドが組み込まれていません。ユーザー独自のゲートウェイを構成および配置する必要があります。コネクターは、**DMZ** 内またはファイアウォールの外側ではなく、**エンタープライズ内でのみ配置する必要があります。**

Connector for SAP XI のアーキテクチャー

上位レベルのコンポーネントのアーキテクチャーを示すため、このセクションでは、2つのデータ・フローについて説明します。図2に2つのシナリオを示します。これら2つのコンポーネントについて、以下で説明します。

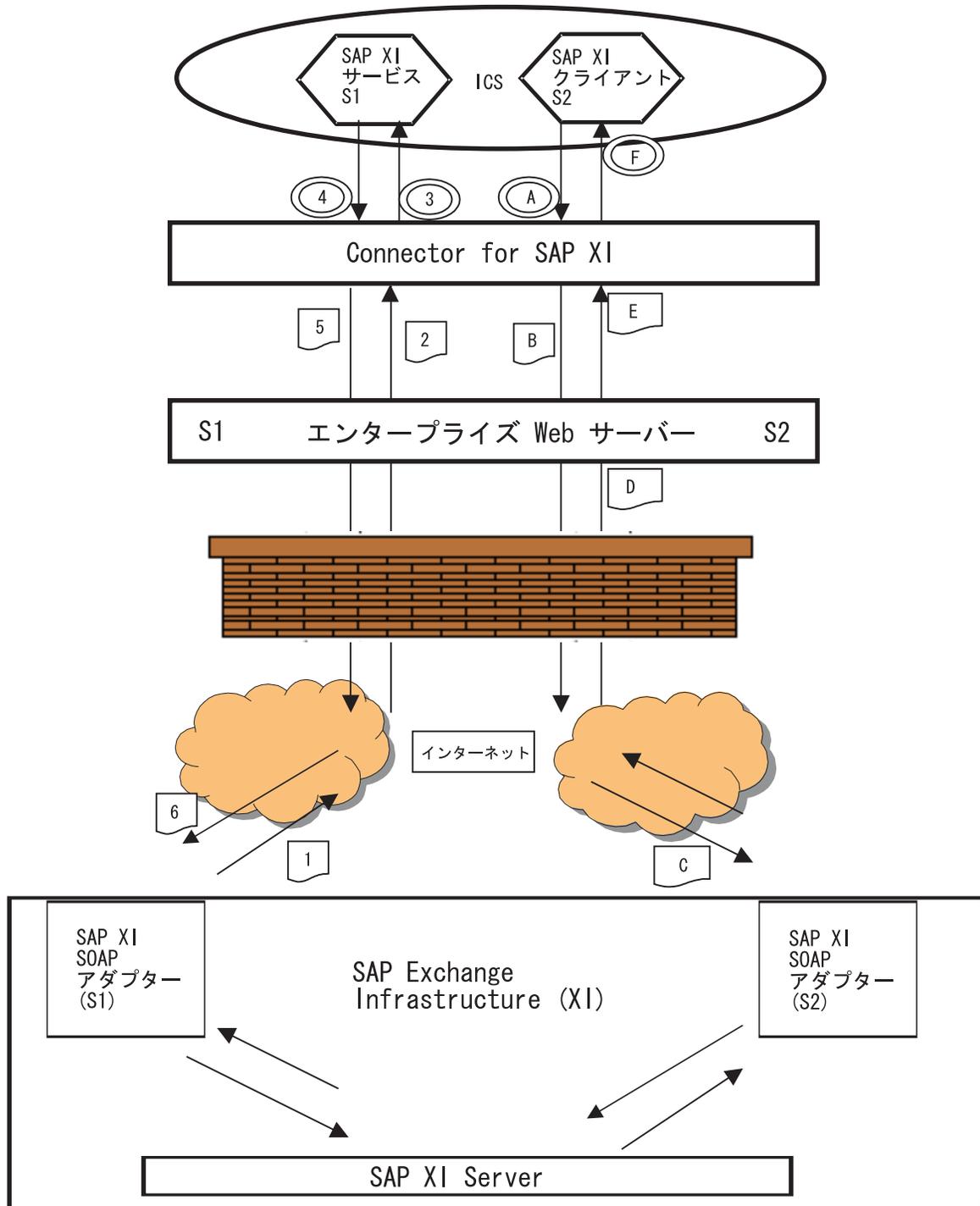


図2. SAP XI メッセージの流れ

要求処理は、コラボレーションがコネクタに対してサービス呼び出し要求を実行したときに発生するイベントの順序を表しています。このシナリオでは、コラボレーションの果たす役割はクライアントであり、要求をサーバーに送信しています。

- A コラボレーションはサービス呼び出し要求をコネクタに送信し、コネクタは SOAP データ・ハンドラーを呼び出してビジネス・オブジェクトを要求メッセージに変換します。
- B コネクタは要求メッセージを送信して、エンタープライズ Web サーバーの URL を呼び出します。
- C エンタープライズ Web サーバーは、SAP XI SOAP アダプターの URL を呼び出します。これにより、要求メッセージは SAP XI サーバー (S2) に転送されます。
- D SAP XI サーバー S2 は、要求を処理して応答を戻します。応答は、同じ接続の一部として戻されます。
- E エンタープライズ Web サーバーは、応答メッセージをアダプターに戻します。
- F コネクタは応答 (または障害) メッセージを受信すると、SOAP データ・ハンドラーを呼び出して、メッセージをビジネス・オブジェクトに変換し、コラボレーションに戻します。

イベント処理は、コラボレーションが SAP XI クライアントに呼び出されたときに発生する、イベントの順序を表しています。このシナリオでは、コラボレーションはサーバーの役割を果たし、クライアント (外部または内部) からの要求を受け入れて、必要に応じて応答します。

- 1 SAP XI SOAP アダプター (クライアント) (S1) は、宛先 (コラボレーション) に要求メッセージを送信します。
- 2 ゲートウェイは、メッセージを受信してコネクタに転送します。
- 3 コネクタはメッセージをビジネス・オブジェクトに変換するために、SOAP データ・ハンドラーに送信します。コネクタはコラボレーションを呼び出します。
- 4 コラボレーションは、応答 (または障害) ビジネス・オブジェクトを戻します。
- 5 コネクタは、SOAP データ・ハンドラーを呼び出して、応答 (または障害) ビジネス・オブジェクトを応答メッセージに変換します。コネクタは、応答をゲートウェイに戻します。
- 6 ゲートウェイは応答メッセージを SAP XI SOAP アダプター (S1) に転送します。

インストール、構成、および設計のチェックリスト

このセクションでは、SAP XI Adapter のインストール、構成、および設計時に実行が必要な作業について要約します。各セクションでは、作業を簡単に説明したうえで、その作業の実行方法や背景情報を説明する、本書中の該当セクションへのリンク (および関連資料への相互参照) を提供します。

アダプターのインストール

何をどこにインストールする必要があるのかについては、13 ページの『第 2 章 インストールおよび始動』を参照してください。

コネクタ・プロパティの構成

コネクタには、2 種類の構成プロパティがあります。標準の構成プロパティと、コネクタ固有の構成プロパティです。これらのプロパティの一部には、デフォルト値が設定されていて、その値を変更する必要はありません。また、コネクタを実行する前に値を設定しなければならないプロパティもあります。詳しくは、51 ページの『第 4 章 SAP XI コネクタ』を参照してください。

プロトコル・ハンドラーおよびプロトコル・リスナーの構成

プロトコル・ハンドラーおよびプロトコル・リスナーの構成は、これらのコンポーネントの振る舞いを制御するコネクタ構成プロパティに値を割り当てるときに行います。詳しくは、51 ページの『第 4 章 SAP XI コネクタ』を参照してください。

SAP XI サービス向けコラボレーションの使用可能化

コラボレーションを SAP XI サービスで使用可能にするには、SAP XI サービスを呼び出すことのできるコラボレーション、あるいは SAP XI サービスとして公開することのできるコラボレーションを作成してください。また、ビジネス・オブジェクトの作成あるいは改造も行ってください。関連する作業の概要については、7 ページの『構成ツール』を参照してください。

SAP XI サービスとしてのコラボレーションの公開

手順の説明については、127 ページの『第 7 章 Web サービスとしてのコラボレーションの公開』を参照してください。

SAP XI サービスを呼び出すコラボレーションの使用可能化

手順の説明については、125 ページの『第 6 章 要求処理のためのコラボレーションの有効化』を参照してください。

SOAP データ・ハンドラーの構成

データ・ハンドラー・メタオブジェクト内の情報は、製品ファイルをインストールしてから始動するまでの間に構成してください。カスタム名ハンドラーを追加しないのであれば、デフォルトの SOAP データ・ハンドラー構成を使用して時間を節約することができます。ただし、それぞれのデータ・ハンドラーを変換するたびに特定のメタオブジェクト情報を構成しなければなりません。この情報は SOAP 構成 MO に収納されます。SOAP 構成 MO は、ビジネス・オブジェクトの作成時に指定します。この作業の大半は、SAP XI サービス (要求処理) を呼び出すコラボレーションを開発するとき自動的に行われます。SOAP メッセージ用のビジネス・オブジェクトを生成するために BIA_XIWSDLUtil と SAPXI ODA を使用すると、SOAP 構成 MO が自動的に生成されます。

データ・ハンドラーの構成に関する詳細については、83 ページの『第 5 章 SOAP データ・ハンドラー』を参照してください。

第 2 章 インストールおよび始動

- 『インストール作業の概要』
- 『コネクタおよび関連ファイルのインストール』
- 16 ページの『構成作業の概要』
- 17 ページの『アダプターの複数インスタンスの実行』
- 18 ページの『コネクタの始動と停止』

この章では、Connector for Web Services をインプリメントするためのコンポーネントのインストール方法について説明します。ICS システムの一般的なインストール方法については、使用プラットフォームの「システム・インストール・ガイド」を参照してください。

インストール作業の概要

ブローカーの互換性、アダプター・フレームワーク、ソフトウェア前提条件、依存関係、および標準と API については、1 ページの『Adapter for SAP XI の環境』を参照してください。

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

ICS のインストール

この作業にはシステムのインストールと ICS の始動が含まれますが、これについては、「システム・インストール・ガイド」に記述されています。ICS は、バージョン 4.2 をインストールする必要があります。

リポジトリにファイルをロードするには、「*WebSphere InterChange Server* システム・インプリメンテーション・ガイド」を参照してください。

コネクタおよび関連ファイルのインストール

この作業の内容は、ソフトウェア・パッケージからコネクタ（および関連コンポーネント）のファイルを使用システムにインストールすることです。『コネクタおよび関連ファイルのインストール』を参照してください。

コネクタ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

このセクションの表では、インストール済みファイルの構造を示します。

Windows コネクターのファイル構造

インストーラーにより、コネクターに関連付けられている標準のファイルがシステムにコピーされます。

ユーティリティーにより、コネクターがインストールされ、コネクター・エージェントのショートカットが「スタート」メニューに追加されます。

表 1 には、コネクターが使用する Windows ファイルの構造が説明されており、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルが示されています。

表 1. アダプター用のインストール済み Windows ファイルの構造

ProductDir のサブディレクトリー	説明
%connectors%SAPXI%start_SAPXI.bat	コネクター始動ファイル
%ODA%SAPXI%start_SAPXIODA.bat	ODA 始動ファイル
connectors%SAPXI%CWSAPXI.jar	SAP XI コネクター
bin\Data\App%SAPXIConnectorTemplate	SAP XI コネクターのテンプレート
connectors%SAPXI%dependencies%soap.jar	SAP XI コネクター、SOAP データ・ハンドラー、WSDL 構成ウィザード、および SAP XI ODA に必要な Apache SOAP API
connectors%messages%SAPXIConnector.txt	コネクター・メッセージ・ファイル
ODA%SAPXI%SAPXIODA.jar	SAP XI および SAP XI ODA に関連したファイル
ODA%messages%SAPXIODAAgent.txt	SAPXI ODA のメッセージ・ファイル
DataHandlers\CwSAPXIDataHandler.jar	SOAP データ・ハンドラー
repository\DataHandlers\MQ_DataHandler_SOAP.txt	SOAP データ・ハンドラーの関連ファイル
connectors%SAPXI%dependencies%LICENSE	Apache ライセンス・ファイル
connectors%SAPXI%dependencies%mail.jar	JavaMail API
connectors%SAPXI%dependencies%activation.jar	Java Activation Framework
connectors%SAPXI%dependencies%ibmjse.jar	IBM 製の JSSE (Java Secure Socket Extension) API
connectors%SAPXI%dependencies%jms.jar	Java Messaging Service
connectors%SAPXI%dependencies%uddi4j-wsdl.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%uddi4jv2.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%IPL10.txt	SAP XI ODA に必要なライセンス・ファイル
connectors%SAPXI%dependencies%wsdl4j.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%CPL10.txt	SAP XI ODA に必要なライセンス・ファイル
connectors%SAPXI%dependencies%qname.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%j2ee.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%wsweb2.1.1%common.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%wsweb2.1.1%core.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%wsweb2.1.1%xercesImpl.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%wsweb2.1.1%xmlParserAPIs.jar	SAP XI ODA に必要
connectors%SAPXI%dependencies%wsweb2.1.1%xsd.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/wsweb2.1.1%xsd.resources.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/IBMReadme.txt	使用権
connectors/SAPXI/dependencies/Notices.txt	サード・パーティーのライセンス
connectors/SAPXI/dependencies/wsweb2.1.3/AL20.txt	サード・パーティーのライセンス
connectors%SAPXI%BIA_XIWSDLTransformer.jar	BIA_XIWSDL 変換プログラム・ツール
connectors%SAPXI%BIA_XIWSDLUtil.bat	BIA_XIWSDL 変換プログラム・ツールの始動スクリプト
ODA%SAPXI%SAPHeader.xsd	SAP XI ODA ヘッダー・スキーマ
ODA%SAPXI%SAPXISystemError.xsd	SAP XI ODA エラー・メッセージ
ODA%SAPXI%SAPXIRFCHeaders.xsd	SAP XI RFC ヘッダー・スキーマ

注: すべての製品パス名は、製品がシステムにインストールされているディレクトリーを基準にした相対パス名です。

UNIX コネクターのファイル構造

インストーラーにより、コネクターに関連付けられている標準のファイルがシステムにコピーされます。

表 2 には、コネクターが使用する UNIX ファイルの構造が説明されており、インストーラーによるコネクターのインストールを選択したときに自動的にインストールされるファイルが示されています。

表 2. アダプター用のインストール済み UNIX ファイルの構造

ProductDir のサブディレクトリー	説明
/connectors/SAPXI/start_SAPXI.sh	コネクター始動ファイル
/ODA/SAPXI/start_SAPXIODA.bat	ODA 始動ファイル
connectors/SAPXI/CWSAPXI.jar	SAP XI コネクター
bin/Data/App/SAPXIConnectorTemplate	SAP XI コネクターのテンプレート
connectors/SAPXI/dependencies/soap.jar	SAP XI コネクター、SOAP データ・ハンドラー、WSDL 構成ウィザード、および SAP XI ODA に必要な Apache SOAP API
connectors/messages/SAPXIConnector.txt	コネクター・メッセージ・ファイル
ODA/SAPXI/SAPXIODA.jar	SAP XI および SAP XI ODA に関連したファイル
ODA/messages/SAPXIODAAgent.txt	SAPXI ODA のメッセージ・ファイル
DataHandlers/CwSAPXIDataHandler.jar	SOAP データ・ハンドラー
repository/DataHandlers/MO_DataHandler_SOAP.txt	SOAP データ・ハンドラーの関連ファイル
connectors/SAPXI/dependencies/LICENSE	Apache ライセンス・ファイル
connectors/SAPXI/dependencies/mail.jar	JavaMail API
connectors/SAPXI/dependencies/activation.jar	Java Activation Framework
connectors/SAPXI/dependencies/ibmjsse.jar	IBM 製の JSSE (Java Secure Socket Extension) API
connectors/SAPXI/dependencies/jms.jar	Java Messaging Service
connectors/SAPXI/dependencies/uddi4j-wsd1.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/uddi4jv2.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/IPL10.txt	SAP XI ODA に必要なライセンス・ファイル
connectors/SAPXI/dependencies/wsd14j.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/CPL10.txt	SAP XI ODA に必要なライセンス・ファイル
connectors/SAPXI/dependencies/qname.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/j2ee.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/wswb2.1.1/common.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/wswb2.1.1/ecore.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/wswb2.1.1/xercesImpl.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/wswb2.1.1/xmlParserAPIs.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/wswb2.1.1/xsd.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/wswb2.1.1/xsd.resources.jar	SAP XI ODA に必要
connectors/SAPXI/dependencies/IBMReadme.txt	使用権
connectors/SAPXI/dependencies/Notices.txt	サード・パーティーのライセンス
connectors/SAPXI/dependencies/wswb2.1.3/AL20.txt	サード・パーティーのライセンス
connectors/SAPXI/BIA_XIWSDLTransformer.jar	BIA_XIWSDL 変換プログラム・ツール
connectors/SAPXI/BIA_XIWSDLUtil.sh	BIA_XIWSDL 変換プログラム・ツールの始動スクリプト
ODA/SAPXI/SAPHeader.xsd	SAP XI ODA ヘッダー・スキーマ
ODA/SAPXI/SAPXISystemError.xsd	SAP XI ODA エラー・メッセージ
ODA/SAPXI/SAPXIRFCHeaders.xsd	SAP XI RFC ヘッダー・スキーマ

注: すべての製品パス名は、製品がシステムにインストールされているディレクトリーを基準にした相対パス名です。

構成作業の概要

インストールしたら、始動する前に、コンポーネントを以下のように構成する必要があります。

コネクターの構成

この作業の内容は、コネクターのセットアップと構成です。69 ページの『コネクターの構成』を参照してください。

ビジネス・オブジェクトの構成

ビジネス・オブジェクトの構成手順は、製品スイートのインプリメント方法をどのように選択するかによって異なります。

- **要求処理** 以下の内容に対応するビジネス・オブジェクトを作成する必要があります。

- 各 SAP XI Web サービスへ送信される要求メッセージ
- 要求に対して考えられる応答 (障害の場合も含む)

詳しくは、21 ページの『第 3 章 ビジネス・オブジェクトの要件』を検討してから、125 ページの『第 6 章 要求処理のためのコラボレーションの有効化』を参照してください。

- **イベント処理** TLO または非 TLO ビジネス・オブジェクトを使用できます。
詳しくは、21 ページの『第 3 章 ビジネス・オブジェクトの要件』を検討してから、127 ページの『第 7 章 Web サービスとしてのコラボレーションの公開』を参照してください。

データ・ハンドラーの構成

インストールが終了したら、SOAP データ・ハンドラーのメタオブジェクトを構成する必要があります。さらに、それぞれの SOAP ビジネス・オブジェクトごとに、SOAP 構成 MO を構成する必要があります。データ・ハンドラーを構成するには、83 ページの『第 5 章 SOAP データ・ハンドラー』を参照してください。

コラボレーションの構成

- **要求処理** 処理の一部として Web サービスを呼び出すコラボレーションの場合には、SAP XI ODA を使用してビジネス・オブジェクトを生成してから、コラボレーション・オブジェクトのポートをそのコネクターにバインドします。ステップごとの手順などの、詳しい説明については、125 ページの『第 6 章 要求処理のためのコラボレーションの有効化』を参照してください。

- **イベント処理** SAP XI クライアントの宛先として公開されるコラボレーションの場合は、WSDL 構成ウィザードを使用して WSDL 文書を生成し、潜在的な顧客がその文書を利用できるようにしてから、顧客がコラボレーションを呼び出せるようにコラボレーション・オブジェクトのポートを構成する必要があります。ステップごとの手順などの、詳しい説明については、127 ページの『第 7 章 Web サービスとしてのコラボレーションの公開』を参照してください。

アダプターの複数インスタンスの実行

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。

- 必要に応じて、コネクタ・プロパティをカスタマイズします。

始動スクリプトの作成

始動スクリプトは以下のように作成します。

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

`dirname`

- この始動スクリプトを、17 ページの『新規ディレクトリの作成』で作成したコネクタ・ディレクトリに格納します。
- 始動スクリプトのショートカットを作成します (Windows のみ)。
- 初期コネクタのショートカット・テキストをコピーし、新規コネクタ・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・コネクタ作成の詳細については、「コネクタ開発ガイド (C++ 用)」または「コネクタ開発ガイド (Java 用)」を参照してください。

コネクタの始動と停止

重要: この章で前述したように、コネクタ、ビジネス・オブジェクト、SOAP データ・ハンドラーのメタオブジェクト、およびコラボレーションは、適正な動作を保証するために、インストールしたらコネクタを始動する前に構成しておく必要があります。これらの作業の要約については、16 ページの『構成作業の概要』を参照してください。さらに、絶対にコネクタのポーリングを使用不可にしないでください (コネクタ・ポーリングはデフォルトで使用可能になっています)。

コネクタは、**コネクタ始動スクリプト**を使用して明示的に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のランタイム・ディレクトリに存在していなければなりません。

`ProductDir%\connectors%\connName`

ここで、`connName` はコネクタを示します。始動スクリプトの名前は、表 3 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 3. コネクタの始動スクリプト

オペレーティング・システム	始動スクリプト
UNIX ベースのシステム	<code>connector_manager_connName</code>
Windows	<code>start_connName.bat</code>

コネクタ始動スクリプトは、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。

第 3 章 ビジネス・オブジェクトの要件

- 『ビジネス・オブジェクトのメタデータ』
- 『コネクター・ビジネス・オブジェクトの構造』
- 22 ページの『同期イベント処理 TLO』
- 32 ページの『非同期イベント処理 TLO』
- 35 ページの『イベント処理の非 TLO』
- 36 ページの『同期要求処理 TLO』
- 36 ページの『同期要求処理 TLO』
- 47 ページの『非同期要求処理 TLO』
- 50 ページの『ビジネス・オブジェクトの開発』

この章では、コネクター・ビジネス・オブジェクトの構造、要件、および属性について説明します。

ビジネス・オブジェクトのメタデータ

Connector for SAP XI は、メタデータ主導型のコネクターです。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に関するデータのことです。このデータはビジネス・オブジェクト定義に格納されており、コネクターとアプリケーションとの対話に役立ちます。メタデータ主導型のコネクターは、コネクター内にハードコーディングされている命令ではなく、ビジネス・オブジェクト定義内にエンコードされているメタデータに基づいて、コネクター自身がサポートしている各ビジネス・オブジェクトを処理します。

ビジネス・オブジェクトのメタデータには、ビジネス・オブジェクトの構造、その属性プロパティの設定値、およびそのアプリケーション固有情報の内容が含まれています。コネクターは、メタデータ主導型なので、新規や変更後のビジネス・オブジェクトを処理する場合にコネクター・コードを変更する必要がありません。ただし、コネクターの構成済みデータ・ハンドラーでは、そのビジネス・オブジェクトの構造、オブジェクトの基数、アプリケーション固有の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現について前提事項が存在します。そのため、Web サービスのビジネス・オブジェクトを作成または変更する場合、その変更内容は、コネクターが従うべきルールに適合している必要があります。適合していないと、コネクターは新規または変更済みのビジネス・オブジェクトを正しく処理できません。

メタデータ、メタオブジェクト、ならびにこれらの構成およびビジネス・オブジェクトや SOAP メッセージとの相互作用については、83 ページの『第 5 章 SOAP データ・ハンドラー』を参照してください。

コネクター・ビジネス・オブジェクトの構造

コネクターは、以下の 2 種類のビジネス・オブジェクトを処理します。

- **TLO SOAP XI** のトップレベル・ビジネス・オブジェクト (TLO) には、要求ビジネス・オブジェクトと、オプションとして応答ビジネス・オブジェクトおよび障害ビジネス・オブジェクトが含まれます。これらの子オブジェクトには、内容データおよび SOAP 構成 MO、およびオプションでプロトコル構成 MO も含まれています。TLO、要求オブジェクト、応答オブジェクト、および障害オブジェクトのほかに、アプリケーション固有情報、属性、および要求とイベント処理に関する要件について、以下のセクションで説明および図解します。

注: TLO は、要求処理およびイベント処理のために使用されます。

- **非 TLO** これらは、TLO 以外の汎用ビジネス・オブジェクト (GBO) およびアプリケーション固有ビジネス・オブジェクト (ASBO) ですが、WSDL 生成時の WSDL 構成ウィザードで使用されています。コネクタは、イベント処理中に、非 TLO を処理できます。これらのオブジェクトについては、以下の 35 ページの『イベント処理の非 TLO』で説明します。詳しくは、130 ページの『WSDL 構成ウィザード』を参照してください。

注: 非 TLO は、イベント処理のみに使用されます。

注: SOAP ヘッダー・コンテナ、ならびに要求、応答、および障害の各ビジネス・オブジェクトに含まれるヘッダー・ビジネス・オブジェクトは、この章では説明しません。SOAP ヘッダー・コンテナおよびヘッダー・ビジネス・オブジェクトについての詳細は、83 ページの『第 5 章 SOAP データ・ハンドラー』を参照してください。

同期イベント処理 TLO

イベント処理のために、コネクタでは、同期および非同期の 2 種類の TLO を使用することが可能です。このセクションでは、同期イベント処理 TLO について説明します。

23 ページの図 3 は、同期イベント処理のためのビジネス・オブジェクト階層を示しています。要求オブジェクトおよび応答オブジェクトは必須であり、障害オブジェクトはオプションです。

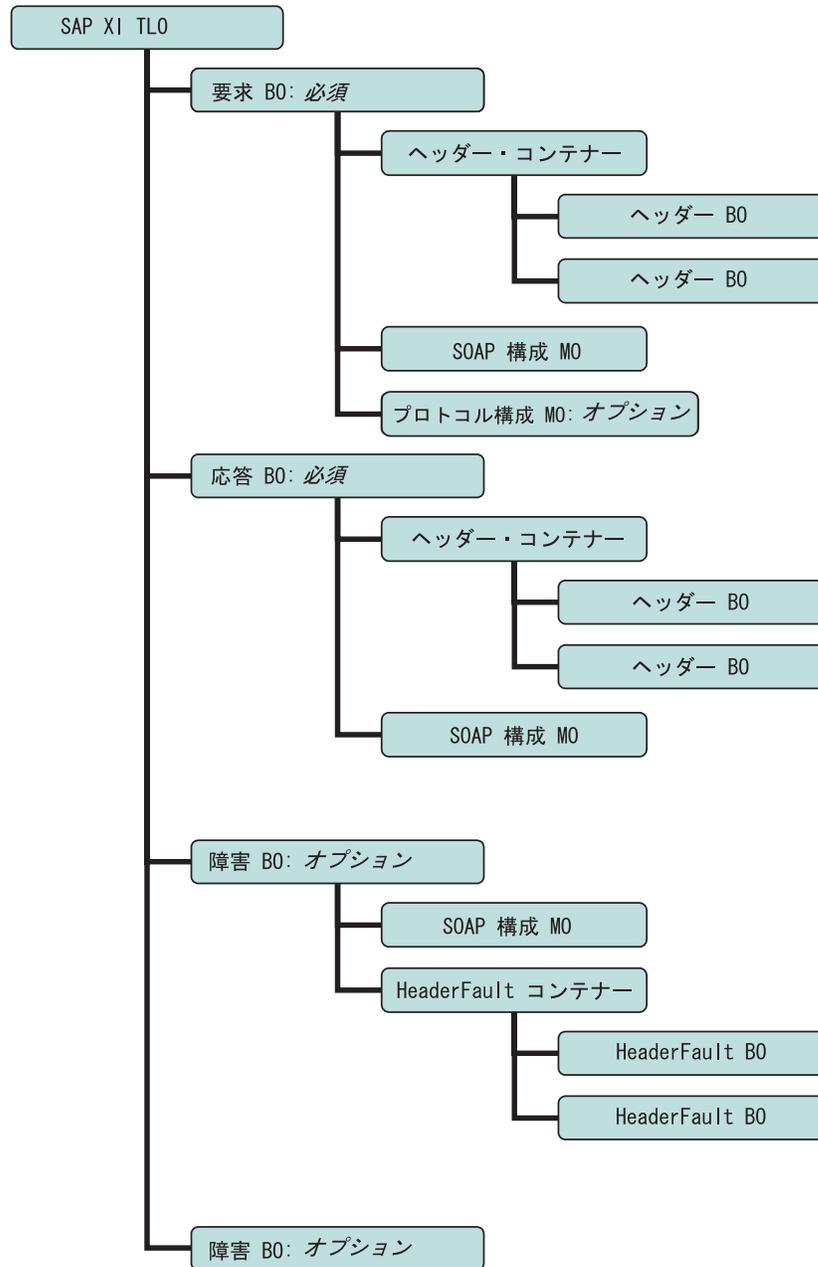


図3. 同期イベント処理のためのビジネス・オブジェクト階層

TLO には、オブジェクト・レベルの ASI のほか、属性レベルの ASI を持った属性が含まれています。両方の種類の ASI について、以下で説明します。

同期イベント処理 TLO のオブジェクト・レベルの ASI

オブジェクト・レベルの ASI は、TLO の性質、および TLO に含まれるオブジェクトについての基本的情報を提供します。図4 は、同期イベント処理のためのサンプル TLO である SERVICE_SYNCH_OrderStatus のオブジェクト・レベルの ASI を表しています。

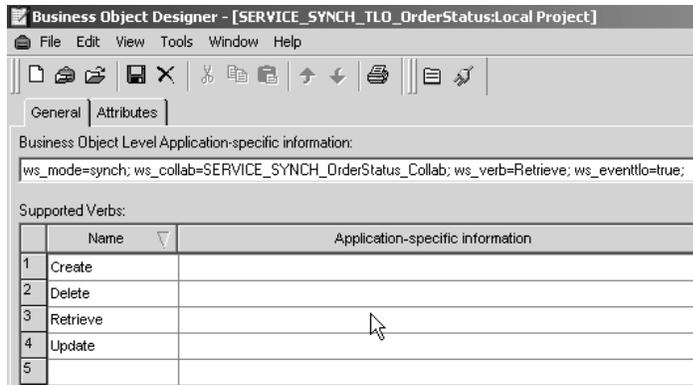


図4. 同期イベント処理のためのトップレベル・ビジネス・オブジェクト

以下の表4では、同期イベント処理 TLO のためのオブジェクト・レベルの ASI について説明します。

表4. 同期イベント処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
<code>ws_eventtlo=true</code>	この ASI プロパティが true に設定されている場合には、コネクタはこのオブジェクトをイベント処理専用の TLO として扱います。 WSDL 構成ウィザードがこの ASI を使用して、ビジネス・オブジェクトが TLO であるかどうかを判別することに注意してください。これについての詳細は、130 ページの『WSDL 構成ウィザード』を参照してください。
<code>ws_collab=collabname</code>	この ASI は、どのコラボレーションを呼び出すのかをコネクタに指示します。この値はコラボレーションの名前です。(この ASI は、WSDL の生成時にコラボレーション用の TLO を判別するためにも使用されます。詳しくは、130 ページの『WSDL 構成ウィザード』を参照してください。) 図4 に示されているサンプルでは、コラボレーション名は <code>SERVICE_SYNCH_OrderStatus_Collab</code> です。
<code>ws_verb=verb</code>	コネクタは、TLO をコラボレーションに引き渡す前に、TLO で動詞を設定するためにこの ASI を使用します。図4 に示されているサンプルでは、動詞は <code>Retrieve</code> です。
<code>ws_mode=synch</code>	イベント通知の際に、コネクタは、この ASI プロパティを使用して、コラボレーションを同期 (<code>synch</code>) で呼び出すのか非同期 (<code>asynch</code>) で呼び出すのかを決定します。同期処理の場合は、この ASI を <code>synch</code> に設定する必要があります。 デフォルトは <code>asynch</code> です。

同期イベント処理 TLO の属性レベルの ASI

それぞれの同期イベント処理 TLO は、属性および属性レベルの ASI を保有しています。図 5 は、サンプル TLO である SERVICE_SYNCH_OrderStatus の属性を表しています。また、この図には、属性レベルの ASI が「アプリケーション固有の情報 (App Spec Info)」列に表示されています。

	Pos	Name	Type	Key	Foreign	Required	Card	App Spec Info
1	1	Request	SERVICE_SYNCH_OrderStatus_Request	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	ws_botype=request
2	2	Response	SERVICE_SYNCH_OrderStatus_Response	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	ws_botype=response
3	3	Fault	SERVICE_SYNCH_OrderStatus_Fault	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	ws_botype=fault
4	4	ObjectEventId	String					
5	5			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

図 5. 同期イベント処理のための TLO 属性

表 5 は、同期イベント処理 TLO の Request、Response、Fault、MimeType、および Charset の各属性に対する属性レベルの ASI を要約したものです。

表 5. 同期イベント処理 TLO の属性 ASI

TLO 属性	属性レベルの ASI	説明
MimeType		オプションの属性: 指定されていると、その値は、同期応答のために起動するデータ・ハンドラーの MIME タイプとして使用されます。String 型で、デフォルトは xml/soap です。
Charset		このオプション・パラメーター (String 型) は、発信ビジネス・オブジェクトをメッセージに変換するときにデータ・ハンドラーに設定される charset を示します。注: この属性に指定された charset 値は、応答メッセージの Content-Type プロトコル・ヘッダーでは伝搬されません。

表 5. 同期イベント処理 TLO の属性 ASI (続き)

TLO 属性	属性レベルの ASI	説明
Request	ws_botype=request	<p>この属性は、要求に対応します。コネクタはこの ASI を使用して、この TLO 属性のタイプが SOAP 要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクタは最初の要求属性の ASI を使用します。</p> <p>この属性は、同期イベント処理 TLO の場合には必須です。</p>
Response	ws_botype=response	<p>この属性は、Web サービスによって戻される応答に対応します。コネクタはこの ASI を使用して、この TLO 属性のタイプが SOAP 応答 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の応答属性がある場合は、コネクタは最初の応答属性の ASI を使用します。</p> <p>この属性は、同期イベント処理 TLO の場合には必須です。</p>
Fault	ws_botype=fault ws_botype=defaultfault	<p>この属性は、同期イベント処理の場合のオプションであり、応答を正常に取り込むことができないときにコラボレーションから戻される障害メッセージに対応します。コネクタは、属性名ではなくこの ASI を使用して、属性のタイプが SOAP 障害 BO であるかどうかを判別します。ws_botype=defaultfault である場合は、WSDL 構成ウィザードはこの障害ビジネス・オブジェクトをヘッダー処理に使用します。詳しくは、97 ページの『ヘッダー障害の処理』を参照してください。</p>

同期イベント処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、同期イベント処理の場合には必須です。要求ビジネス・オブジェクトは、オブジェクト・レベルの ASI を保有しています。例えば、Business Object Designer で SERVICE_SYNCH_OrderStatus_Request をオープンし、「一般」タブをクリックすると、図 6 に示すように、オブジェクト・レベルの ASI が表示されます。

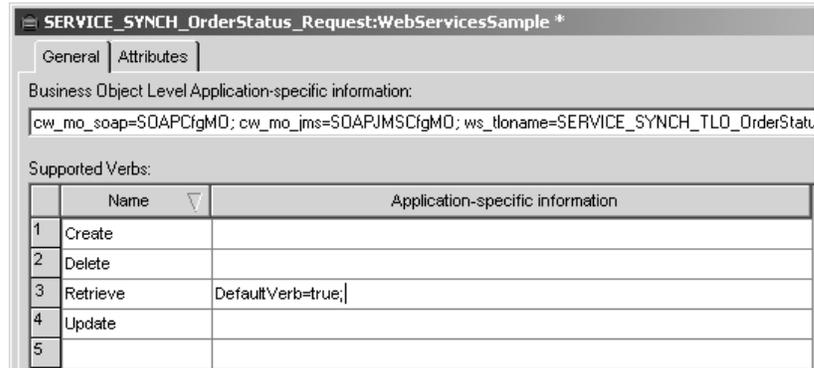


図 6. 同期イベント処理要求オブジェクトに関するオブジェクト・レベルの ASI

同期イベント処理のための要求ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 6 に説明されています。図 6 に示すように、要求ビジネス・オブジェクトに対して、デフォルトの動詞を指定できます。これを行うには、トップレベルの要求ビジネス・オブジェクトの「サポートされている動詞 (Supported Verbs)」リストにある動詞の ASI フィールドに、

DefaultVerb=true;

と指定します。DefaultVerb ASI が指定されず、データ・ハンドラーが動詞の設定されていないビジネス・オブジェクトを処理する場合、そのビジネス・オブジェクトは、動詞なしで戻されます。

表 6. 同期イベント処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_soap=SOAPCfMO	この ASI の値は、SOAP 構成 MO に対応する属性の名前と一致しなければなりません。これは、要求ビジネス・オブジェクトのためのデータ・ハンドラー変換を定義するメタオブジェクトです。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。
cw_mo_http=SOAPHTTPCfMO	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。ASI は、SOAP/HTTP または SOAP/HTTPS のプロトコル・リスナーを指定します。ASI およびプロトコル構成 MO は、両方ともオプションです。詳しくは、29 ページの『プロトコル構成 MO』を参照してください。

表 6. 同期イベント処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI (続き)

オブジェクト・レベルの ASI	説明
ws_tloname=tloname	この ASI は、このオブジェクトが属する SAP XI TLO の名前を指定します。イベント処理の際に、コネクタは、この ASI を使用して、データ・ハンドラーによって引き渡された要求ビジネス・オブジェクトが TLO の子であるかどうかを判別します。引き渡された要求ビジネス・オブジェクトが TLO の子である場合は、コネクタは、指定された TLO を作成し、要求ビジネス・オブジェクトをその TLO の子として設定し、TLO のオブジェクト・レベルの ASI を使用して、この要求ビジネス・オブジェクトを、サブスクライブしているコラボレーションに引き渡します。

同期イベント処理のための応答ビジネス・オブジェクト

応答ビジネス・オブジェクトは、TLO の子であり、同期イベント処理の場合には必須です。同期イベント処理のための応答ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 7 に説明されています。

表 7. 同期イベント処理: 応答ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_soap=SOAPCfgMO	この ASI の値は、SOAP 構成 MO に対応する属性の名前と一致しなければなりません。これは、応答ビジネス・オブジェクトに関するデータ・ハンドラー変換を定義する SOAP 構成 MO です。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。

注: オプションで、要求 BO にプロトコル構成 MO のオブジェクト・レベルの ASI を組み込むことができます。

同期イベント処理のための障害ビジネス・オブジェクト

障害ビジネス・オブジェクトは、TLO の子であり、同期イベント処理の場合にはオプションです。同期イベント処理のための障害ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 8 に説明されています。

表 8. 同期イベント処理: 障害ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_soap=SOAPCfgMO	この ASI の値は、SOAP 構成 MO に対応する属性の名前と一致しなければなりません。これは、障害ビジネス・オブジェクトに関するデータ・ハンドラー変換を定義する SOAP 構成 MO です。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。

注: オプションで、障害 BO にプロトコル構成 MO のオブジェクト・レベルの ASI を組み込むことができます。

SOAP 構成 MO

SOAP 構成 MO は、あるデータ・ハンドラー変換 (SOAP メッセージからビジネス・オブジェクトへの変換、またはビジネス・オブジェクトから SOAP メッセージへの変換) で行われるフォーマットの振る舞いを定義します。それぞれの要求、応答、または障害属性に SOAP 構成 MO が割り当てられます。その属性である BodyName、BodyNS、Style、Use、TypeInfo、TypeCheck、および BOverb の型は、常に String になっています。これらの属性は、SOAP メッセージ要素に対応し、それらの値は、SOAP データ・ハンドラーがメッセージおよびオブジェクトをどのように読み取り、検証するのかを決定します。SOAP 構成 MO および属性についての詳細は、85 ページの『SOAP 構成メタオブジェクト: 各 SOAP ビジネス・オブジェクトの子』を参照してください。すべての SOAP 構成 MO は、要求、応答、または障害オブジェクトのいずれの場合にも、BodyName および BodyNS のデフォルト値に関する固有の項目を含んでいなければなりません。

プロトコル構成 MO

この MO は、オプションで、イベント処理の要求、応答、または障害ビジネス・オブジェクトの子として組み込まれます。通常、プロトコル・ヘッダーとカスタム・プロパティの (要求メッセージからの) 読み取りまたは (応答または障害メッセージへの) 伝搬を行う必要のある場合に指定します。前述したように、要求ビジネス・オブジェクトは、必要に応じて、プロトコル構成 MO の名前をビジネス・オブジェクト・レベルの ASI として `cw_mo_http=HTTPProtocolListenerConfigMOAttribute` と宣言します。

イベント処理中に、コネクタはプロトコル・リスナー (SOAP/HTTP、SOAP/HTTPS) を使用してトランスポートからイベントを検索します。これらのイベントは、SAP XI サービスとして公開されているコラボレーションからのサービスを要求する、内部または外部 Web サービス・クライアントから送られるメッセージです。それぞれのトランスポートには、固有のヘッダー要件があります。コネクタは、プロトコル構成 MO を使用して、プロトコル固有のヘッダー情報をプロトコル・リスナーからコラボレーションへと転送します。

SOAP/HTTP(S) プロトコルの場合、プロトコル構成 MO の属性は次のとおりです。

表9. イベント処理用 HTTP/HTTPS プロトコル構成 MO 属性

属性	必須	型	説明
Content-Type	いいえ	String	この属性の値によって、発信メッセージの Content-Type ヘッダーが定義されます (ヘッダーには、発信メッセージのメッセージ ContentType および 0 個以上のパラメーター (charset) が組み込まれます)。構文は、HTTP プロトコルの Content-Type ヘッダーと同じです (例: text/html; charset=ISO-8859-4)。 Content-Type 属性が定義されていない場合、コネクタは、応答/障害メッセージの ContentType として、要求の ContentType を使用します。
UserDefinedProperties	いいえ	ビジネス・オブジェクト	この属性は、ユーザー定義のプロトコル・プロパティのビジネス・オブジェクトを保持します。
1 つ以上の HTTP ヘッダー	いいえ	String	この属性を使用すれば、ハンドラーは、指定された HTTP ヘッダーの値の受け渡しまたは検索ができます。
Authorization_UserID	いいえ	String	この属性は HTTP 基本認証の userID に対応します。
Authorization_Password	いいえ	String	この属性は HTTP 基本認証のパスワードに対応します。

これらの属性を以下に示します。

- 『イベント処理のためのユーザー定義のプロパティ』
- 31 ページの『イベント処理用の HTTP 証明書伝搬』

プロトコル・リスナーの詳細については、54 ページの『プロトコル・リスナー』を参照してください (要求処理用のプロトコル構成 MO については、36 ページの『同期要求処理 TLO』を参照してください)。

イベント処理のためのユーザー定義のプロパティ: オプションで、HTTP(S) プロトコル構成 MO にカスタム・プロパティを指定できます。この指定を行うには、UserDefinedProperties 属性を組み込みます。この属性は、1 つ以上の子属性とプロパティ値を持つビジネス・オブジェクトに対応します。このビジネス・オブジェクトのすべての属性は、メッセージ・ヘッダーの変数部分で読み取られる (同期応答の場合は書き込まれる) 単一プロパティを以下のように定義する必要があります。

- 属性の型は、プロトコル・プロパティ・タイプにかかわらず、常に String です。属性のアプリケーション固有情報には、属性がマップされるプロトコル・メッセージ・プロパティ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入れることができます。

表 10 は、これらの属性のアプリケーション固有の情報を要約したものです。

表 10. ユーザー定義のプロトコル・プロパティ属性のアプリケーション固有情報: 名前 = 値のペアの内容

名前	値	説明
ws_prop_name (大文字小文字を区別しない。指定されなければ、属性名がプロパティ名として使用される)	任意の有効なプロトコル・プロパティ名	これはプロトコル・プロパティの名前です。ベンダーによっては、いくつかのプロパティが拡張機能に予約されています。
ws_prop_type	String	プロトコル・プロパティのタイプ。

指定されたカスタム・プロパティ ASI (ws_prop_name または ws_prop_type) が無効で、このヘッダーを処理する論理的方法 (HTTP 処理でプロパティ・タイプを無視するなど) がない場合、コネクタはログに警告を記録し、このプロパティを無視します。ws_prop_name または ws_prop_type に対しての必要な検査は実行されたが、カスタム・プロパティの値を設定することも検索することもできない場合、コネクタはログにエラーを記録し、イベントは失敗します。

UserDefinedProperties 属性が指定された場合、コネクタは、UserDefinedProperties ビジネス・オブジェクトのインスタンスを作成します。次に、コネクタはメッセージからプロパティ値を抽出し、ビジネス・オブジェクトに保管します。少なくとも 1 つのプロパティ値が正常に検索されれば、コネクタは、変更された UserDefinedProperties 属性をプロトコル構成 MO に設定します。

同期イベント処理の場合、UserDefinedProperties 属性が指定されて、そのビジネス・オブジェクトのインスタンスが作成されると、コネクタは、この子ビジネス・オブジェクトの各属性を処理し、それに対応するメッセージ・プロパティ値を設定します。

イベント処理用の HTTP 証明書伝搬: 証明書伝搬のために、コネクタは HTTP プロトコル構成 MO の Authorization_UserID 属性および Authorization_Password 属性をサポートします。このサポートは、HTTP 基本認証スキームの一部としてのこれらの証明書の伝搬に限られます。

SOAP/HTTP または SOAP/HTTPS プロトコル・リスナーが許可ヘッダーを含む SOAP/HTTP Web サービス要求を処理する場合は、リスナーはヘッダーを解析して HTTP 基本認証に従うかどうか判別します。従う場合は、リスナーはユーザー名およびパスワードを抽出し、Base64 を使用してデコードします。このデコードされたストリングは、コロンで区切られたユーザー名およびパスワードから構成されません。プロトコル・リスナーがプロトコル構成 MO で Authorization_UserID 属性および Authorization_Password 属性を検出した場合、リスナーは、イベント許可ヘッダーから抽出した値をこれらの属性に設定します。

ヘッダー・コンテナ・ビジネス・オブジェクト

ヘッダー・コンテナ属性は SOAP ヘッダー属性とも呼ばれ、子ビジネス・オブジェクトのみを含むビジネス・オブジェクトに対応します。それぞれの子は、SOAP メッセージのヘッダー項目を表します。SOAP ヘッダー属性には、SOAP データ・ハンドラーが必要とするアプリケーション固有情報 (ASI) が含まれます。例えば、ヘッダー・コンテナ・ビジネス・オブジェクトは、その ASI によって、`soap_location=SOAPHeader` のように識別されます。ヘッダー処理についての詳細は、91 ページの『SOAP データ・ハンドラーの処理』を参照してください。

すべての SOAP ビジネス・オブジェクトは、要求オブジェクトであるのか、応答オブジェクトであるのか、あるいは障害オブジェクトであるのかに関係なく、ヘッダー・コンテナを 1 つだけ含んでいます。

ヘッダー子ビジネス・オブジェクト

それぞれのヘッダー子ビジネス・オブジェクトは、SOAP メッセージ内の 1 つのヘッダー要素を表します。ヘッダー要素は、SOAP メッセージの SOAP-Env:Header 要素の直接の子です。ヘッダー子ビジネス・オブジェクトには、`actor` および `mustUnderstand` 属性が割り当てられることがあります。これらの属性は、SOAP ヘッダー要素の `actor` および `mustUnderstand` 属性に対応しています。ヘッダー処理についての詳細は、91 ページの『SOAP データ・ハンドラーの処理』を参照してください。

SOAP ヘッダー・メッセージ要素を表すために必要な、任意の数のヘッダー子オブジェクトを含めることができます。

非同期イベント処理 TLO

図7 は、非同期イベント処理のためのビジネス・オブジェクト階層を示しています。要求オブジェクトのみが必須です。

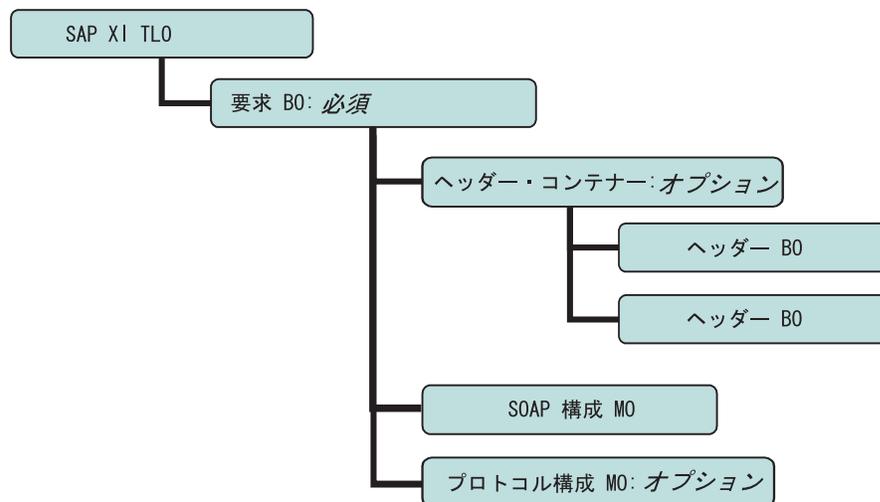


図7. 非同期イベント処理のためのビジネス・オブジェクト階層

TLO には、オブジェクト・レベルの ASI のほか、属性レベルの ASI を持った属性が含まれています。両方の種類の ASI について、以下で説明します。ヘッダー・コ

コンテナおよびヘッダー子ビジネス・オブジェクトについての詳細は、32ページの『ヘッダー・コンテナ・ビジネス・オブジェクト』を参照してください。

非同期イベント処理 TLO のためのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI は、TLO の性質、および TLO に含まれるオブジェクトについての基本的情報を提供します。図 8 は、非同期イベント処理のためのサンプル TLO である SERVICE_ASYNC_TLO_Order のオブジェクト・レベルの ASI を表しています。

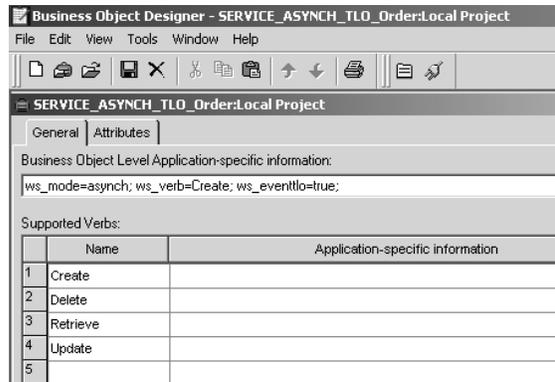


図 8. 非同期イベント処理のためのトップレベル・ビジネス・オブジェクト

以下の表 4 では、非同期イベント処理 TLO のためのオブジェクト・レベルの ASI について説明します。

表 11. 非同期イベント処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
ws_eventtlo=true	この ASI プロパティが true に設定されている場合には、コネクタはこのオブジェクトをイベント処理用の TLO として扱います。 WSDL 構成ウィザードがこの ASI を使用して、ビジネス・オブジェクトが TLO であるかどうかを判別することに注意してください。これについての詳細は、130 ページの『WSDL 構成ウィザード』を参照してください。
ws_verb=verb	コネクタは、TLO をコラボレーションに引き渡す前に、TLO で動詞を設定するためにこの ASI を使用します。図 8 に示されているサンプルでは、動詞は Create です。
ws_mode=asynch	イベント通知の際に、コネクタは、この ASI プロパティを使用して、コラボレーションを同期 (synch) で呼び出すのか非同期 (asynch) で呼び出すのかを決定します。非同期処理の場合は、この ASI を asynch に設定する必要があります。 デフォルトは asynch です。

注: 同期イベント処理の場合とは異なり、非同期イベント処理では、TLO レベルの
 コラボレーション名 ASI は必要ありません。その代わりに、統合ブローカー
 は、サブスクライブしているすべてのコラボレーションにアプリケーション・
 イベントが到達するものと想定します。

非同期イベント処理 TLO のための属性レベルの ASI

それぞれの非同期イベント処理 TLO には、要求ビジネス・オブジェクトに対応す
 る単一の属性が含まれています。図 9 は、サンプル TLO の
 SERVICE_ASYNC_TLO_Order の要求属性、および属性の ASI を表しています。

	Pos	Name	Type	Key	Foreign	Required	Card	Maximum	App Spec Info
1	1	Request	SERVICE_ASYNC_Order	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1		ws_botype=request
2	2	ObjectEventId	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
3	3			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255	

図 9. 非同期イベント処理のための TLO 属性

表 12 は、非同期イベント処理 TLO の要求属性に対する属性レベルの ASI を要約
 したものです。

表 12. 非同期イベント処理 TLO の属性 ASI

TLO 属性	属性レベルの ASI	説明
Request	ws_botype=request	<p>この属性は、Web サービス要求に対応します。コネクタはこの ASI を使用して、この TLO 属性のタイプが SOAP 要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクタは最初の要求属性の ASI を使用します。</p> <p>この属性は、同期イベント処理 TLO の場合には必須です。</p>

非同期イベント処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、非同期イベント処理の場合には
 必須です。要求ビジネス・オブジェクトに対して、デフォルトの動詞を指定できま
 す。これを行うには、トップレベルの要求ビジネス・オブジェクトの「サポートさ
 れている動詞 (Supported Verbs)」リストにある動詞の ASI フィールドに、

DefaultVerb=true;

と指定します。DefaultVerb ASI が指定されず、データ・ハンドラーが動詞の設定されていないビジネス・オブジェクトを処理する場合、そのビジネス・オブジェクトは、動詞なしで戻されます。非同期イベント処理のための要求ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 13 に説明されています。

表 13. 非同期イベント処理: 要求ビジネス・オブジェクトのためのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
<code>cw_mo_soap=SOAPCfgMO</code>	この ASI の値は、SOAP 構成 MO に対応する属性の名前と一致しなければなりません。これは、要求ビジネス・オブジェクトに対するデータ・ハンドラー変換を定義する SOAP 構成 MO です。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。
<code>cw_mo_http=SOAPHTTPCfgMO</code>	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。ここに示す ASI は、SOAP/HTTP または SOAP/HTTPS のプロトコル・リスナーを指定します。ASI およびプロトコル構成 MO は、両方ともオプションです。詳しくは、29 ページの『プロトコル構成 MO』を参照してください。
<code>ws_tloname=tloname</code>	この ASI は、このオブジェクトが属する Web サービス TLO の名前を指定します。イベント処理の際に、コネクタは、この ASI を使用して、データ・ハンドラーによって引き渡された要求ビジネス・オブジェクトが TLO の子であるかどうかを判別します。引き渡された要求ビジネス・オブジェクトが TLO の子である場合は、コネクタは、指定された TLO を作成し、要求ビジネス・オブジェクトをその TLO の子として設定し、TLO のオブジェクト・レベルの ASI を使用して、この要求ビジネス・オブジェクトを、サブスクライブしているコラボレーションに引き渡します。

非同期イベント処理の場合の SOAP 構成 MO、プロトコル構成 MO、SOAP ヘッダー・コンテナー、およびヘッダー子ビジネス・オブジェクトの要件および特性は、同期イベント処理の場合と同じです。詳しくは、22 ページの『同期イベント処理 TLO』における該当のトピックを参照してください。

イベント処理の非 TLO

オブジェクト・レベルの ASI である `ws_eventtlo=true` がビジネス・オブジェクト内に存在しない場合、コネクタはそのオブジェクトが TLO ではないと判断します。イベント処理の際に、コネクタは、非 TLO、すなわち、汎用ビジネス・オブジェクトおよびアプリケーション固有ビジネス・オブジェクトを処理できます。非 TLO の場合、同じビジネス・オブジェクトが、要求ビジネス・オブジェクトと応答ビジネス・オブジェクトを表します。

非 TLO には、SOAP 構成 MO が含まれていません。コラボレーションを SAP XI サービスとして公開すると、WSDL 構成ウィザードはコネクターの WSCollaborations プロパティを構成します。コネクターは、WSCollaborations プロパティを使用して、要求メッセージの BodyName および BodyNS を判別します。非 TLO の場合、WSCollaborations プロパティがビジネス・オブジェクトの解決のために使用されるということに注意してください。

非 TLO を使用する利点は、SAP XI ソリューションで使用する TLO 構造のビジネス・オブジェクトを新規に開発する必要がないということです。しかし、TLO を使用すると、データ、すなわち、顧客や会社などをより正確で経済的に公開することができます。また、TLO ビジネス・オブジェクトは、非 TLO よりもカスタマイズに適しています。

非 TLO を WSDL 構成ウィザードへの入力データとして使用する場合は、128 ページの『ビジネス・オブジェクトの識別または開発』を参照してください。

同期要求処理 TLO

要求処理のために、コネクターでは、同期および非同期の 2 種類の TLO を使用することができます。このセクションでは、同期要求処理 TLO について説明します。

図 10 は、同期要求処理のための TLO ビジネス・オブジェクト階層を示しています。要求オブジェクトおよび応答オブジェクトは必須であり、障害オブジェクトはオプションです。イベント処理とは異なり、プロトコル構成 MO は、要求オブジェクトの場合には必須であり、応答および障害オブジェクトの場合には、オプションです。ヘッダー・コンテナおよびヘッダー子ビジネス・オブジェクトについての詳細は、32 ページの『ヘッダー・コンテナ・ビジネス・オブジェクト』を参照してください。

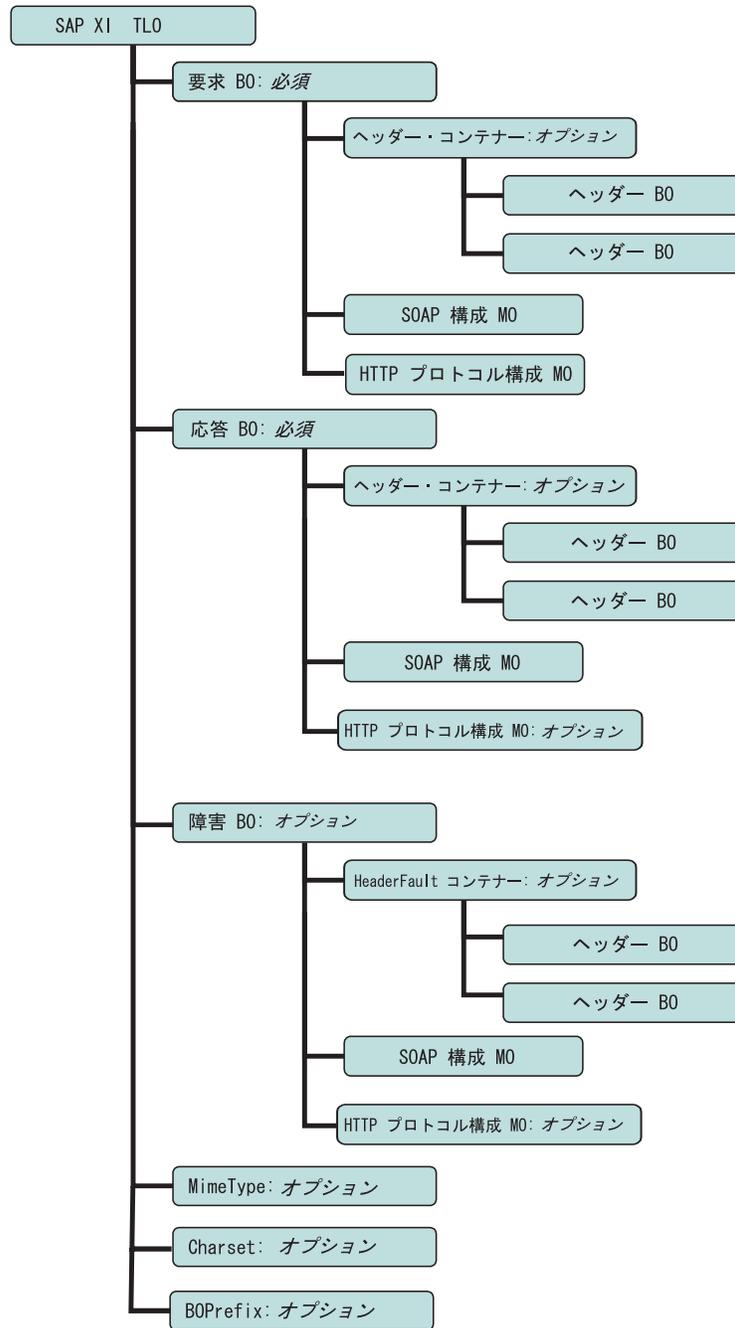


図 10. 同期要求処理のためのビジネス・オブジェクト階層

同期要求処理 TLO のオブジェクト・レベルの ASI

オブジェクト・レベルの ASI は、TLO の性質、および TLO に含まれるオブジェクトについての重要情報を提供します。図 11 は、同期要求処理のためのサンプル TLO である CLIENT_SYNCH_TLO_OrderStatus を表しています。

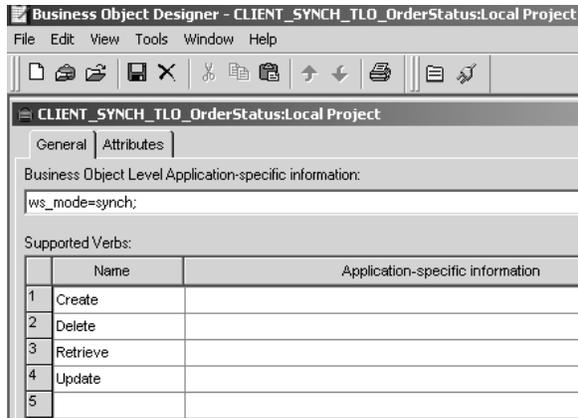


図 11. 同期要求処理のためのトップレベル・ビジネス・オブジェクト

表 14 では、同期要求処理 TLO のためのオブジェクト・レベルの ASI について説明します。同期イベント処理 TLO のための ASI とは異なり、ws_collab、ws_verb、または ws_eventtlo ASI は、このレベルでの要求処理には必須ではありません。

表 14. 同期要求処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
ws_mode=synch	<p>要求処理の際に、コネクタは、この ASI プロパティを使用して、Web サービスを同期 (synch) で呼び出すか非同期 (asynch) で呼び出すかを決定します。synch が指定された場合、コネクタは応答を予期します。したがって、TLO には、要求および応答ビジネス・オブジェクト、およびオプションとして 1 つまたは複数の障害オブジェクトを組み込む必要があります。</p> <p>デフォルトは asynch です。</p>

同期要求処理 TLO のための属性レベルの ASI

図 12 は、CLIENT_SYNCH_TLO_OrderStatus TLO の属性、および属性レベルの ASI を示しています。

	Pos	Name	Type	Key	Card	Maximum Length	Default	App Spec Info
1	7	ObjectEventId	String					
2	2	MimeType	String	<input type="checkbox"/>		255	xml/soap	
3	3	BOPrefix	String	<input type="checkbox"/>		255		
4	1	Handler	String	<input type="checkbox"/>		255	soap/http	
5	6	田 Fault	CLIENT_SYNCH_OrderStatus_Fault	<input type="checkbox"/>	1			ws_botype=fault
6	4	田 Request	CLIENT_SYNCH_OrderStatus_Request	<input checked="" type="checkbox"/>	1			ws_botype=request
7	5	田 Response	CLIENT_SYNCH_OrderStatus_Response	<input type="checkbox"/>	1			ws_botype=response

図 12. 同期要求処理のための TLO 属性

図 12 に表示された属性および ASI について、表 15 で説明します。

表 15. 要求処理の TLO 属性

TLO 属性	属性レベルの ASI	説明
MimeType	なし	この属性は、要求ビジネス・オブジェクトを要求メッセージに変換するためにコネクタが起動するデータ・ハンドラーの MIME タイプを指定します。この値は、メッセージ変換規則構成に応じて、同期応答/障害メッセージをビジネス・オブジェクトに変換するために使用される場合があります。
BOPrefix	なし	型 String のこの属性は、データ・ハンドラーに渡されます。
Handler	なし	この属性は、Web サービス要求の処理に使用するプロトコル・ハンドラーを指定もので、要求処理専用です。設定できる値は soap/http です。コネクタは SOAP/HTTP または SOAP/HTTPS プロトコル・ハンドラーを使用してこの要求を処理します。デフォルトは soap/http です。
Charset		型 String のこのオプション・パラメーターは、要求ビジネス・オブジェクトをメッセージに変換するときにデータ・ハンドラーに設定される charset を指定します。注: この属性に指定された charset 値は、要求メッセージの Content-Type プロトコル・ヘッダーでは伝搬されません。

表 15. 要求処理の TLO 属性 (続き)

TLO 属性	属性レベルの ASI	説明
Request	ws_botype=request	この属性は、Web サービス要求ビジネス・オブジェクトに対応します。コネクタはこの属性 ASI を使用して、この TLO 属性のタイプが SOAP 要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクタは最初に取り込まれた属性の ASI を使用します。
Response	ws_botype=response	この属性は、コラボレーションによって戻される応答に対応し、同期要求処理の場合には必須です。コネクタはこの属性 ASI を使用して、この TLO 属性のタイプが SOAP 応答 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。
Fault	ws_botype=fault または ws_botype=defaultfault	この属性は同期要求処理の場合のオプションであり、応答を正常に取り込むことができないときに Web サービスから戻される障害メッセージに対応するものです。 コネクタはこの ASI を使用して、TLO の属性のタイプが SOAP 障害 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。障害メッセージが詳細要素である場合は、defaultfault ビジネス・オブジェクトが戻されます。defaultfault は、デフォルトのビジネス・オブジェクトの解決に使用されます。詳しくは、83 ページの『第 5 章 SOAP データ・ハンドラー』を参照してください。

同期要求処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、同期要求処理の場合には必須です。要求ビジネス・オブジェクトは、オブジェクト・レベルの ASI を保有していません。

表 16 では、同期要求処理のための要求ビジネス・オブジェクトのオブジェクト・レベルの ASI について説明します。

表 16. 同期要求処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
<code>cw_mo_soap=SOAPCfgMO</code>	この ASI の値は、SOAP 構成 MO に対応する属性の名前と一致しなければなりません。これは、要求ビジネス・オブジェクトに対するデータ・ハンドラー変換を定義する SOAP 構成 MO です。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。
<code>cw_mo_http=SOAPHTTPCfgMO</code>	このオプション ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。これは、SOAP/HTTP-HTTPS プロトコル・ハンドラーの宛先を指定する、別のプロトコル構成 MO です。この ASI は、SOAP/HTTP および SOAP/HTTPS プロトコル・ハンドラーによって使用されます。要求処理のためには、TLO 要求属性に HTTP プロトコル構成 MO が必要です。詳しくは、42 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。
<code>SOAPAction=SOAPActionURI</code>	コネクタはこの ASI を使用して、要求メッセージに SOAPAction ヘッダーを設定するかどうかを判別します。この ASI は、宛先 Web サービスが SOAPAction ヘッダーを必要とする場合にのみ指定してください。この ASI は要求処理に使用されるものであり、イベント通知には使用されません。

同期要求処理のための応答ビジネス・オブジェクト

応答ビジネス・オブジェクトは、TLO の子であり、同期要求処理の場合には必須です。同期要求処理のための応答ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 17 に説明されています。

表 17. 同期要求処理: 応答ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
<code>cw_mo_soap=SOAPCfgMO</code>	この ASI の値は、SOAP プロトコル構成 MO に対応する属性の名前と一致しなければなりません。これは、応答ビジネス・オブジェクトに関するデータ・ハンドラー変換を定義する SOAP 構成 MO です。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。
<code>cw_mo_http=SOAPHTTPCfgMO</code>	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。これは、HTTP プロトコル・ハンドラーの応答 SOAP メッセージ内のヘッダーを指定するプロトコル構成 MO であり、応答ビジネス・オブジェクトの場合にはオプションです。詳しくは、29 ページの『プロトコル構成 MO』を参照してください。

応答ビジネス・オブジェクトに対して、デフォルトの動詞を指定できます。これを行うには、トップレベルの要求ビジネス・オブジェクトの「サポートされている動詞 (Supported Verbs)」リストにある動詞の ASI フィールドに、

```
DefaultVerb=true;
```

と指定します。DefaultVerb ASI が指定されず、データ・ハンドラーが動詞の設定されていないビジネス・オブジェクトを処理する場合、応答ビジネス・オブジェクトが、動詞なしで戻されます。

同期要求処理のための障害ビジネス・オブジェクト

障害ビジネス・オブジェクトは、TLO の子であり、同期要求処理の場合にはオプションです。同期要求処理のための障害ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 8 に説明されています。

表 18. 同期要求処理: 障害ビジネス・オブジェクトのためのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_soap=SOAPCfgMO	この ASI の値は、SOAP プロトコル構成 MO に対応する属性の名前と一致しなければなりません。これは、障害ビジネス・オブジェクトに関するデータ・ハンドラー変換を定義する SOAP 構成 MO です。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。
cw_mo_http=SOAPHTTPCfgMO	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。これは、プロトコル・ハンドラーの応答 SOAP メッセージ内のヘッダーを指定するプロトコル構成 MO であり、障害ビジネス・オブジェクトの場合にはオプションです。詳しくは、29 ページの『プロトコル構成 MO』を参照してください。

SOAP 構成 MO

SOAP 構成 MO (SOAPCfgMO) の属性は、イベント処理用の SOAP 構成 MO の場合と同じです。詳しくは、29 ページの『SOAP 構成 MO』および 85 ページの『SOAP 構成メタオブジェクト: 各 SOAP ビジネス・オブジェクトの子』を参照してください。

要求処理のための HTTP プロトコル構成 MO

要求処理の際に、SOAP/HTTP-HTTPS プロトコル・ハンドラーは、ターゲット SAP XI サービスの宛先を判別するために、HTTP プロトコル構成 MO を使用します。このプロトコル構成 MO は、要求ビジネス・オブジェクトの場合には必須です。SOAP/HTTP-HTTPS プロトコル・ハンドラーは、HTTP 1.0 POST 要求のみをサポートします。43 ページの表 19 に示すように、唯一の必須属性 (Destination) はターゲット Web サービスの完全 URL です。オプションの許可属性については、以下のセクションで説明します。

表 19. 要求処理のための HTTP プロトコル構成 MO 属性

属性	必須	型	説明
Destination	はい	String	ターゲット SAP XI サービスの宛先 URL。SOAP/HTTP-HTTPS プロトコル・ハンドラーは、この属性を使用して、SAP XI サービスの宛先を判別します。
Content-Type	要求ビジネス・オブジェクトでは必須。それ以外ではオプション。	String	この属性の値は、発信メッセージの Content-Type ヘッダー (発信メッセージのメッセージ ContentType、およびオプションで charset が含まれる) を定義します。構文は、HTTP プロトコルの Content-Type ヘッダーと同じです (例: text/html; charset=ISO-8859-4)。Content-Type 属性が定義されていない場合、コネクタは、メッセージの ContentType として text/xml を使用します。
Authorization_UserID	いいえ	String	この属性は HTTP 基本認証の userID に対応します。詳しくは、46 ページの『要求処理用の HTTP 証明書伝搬』を参照してください。
Authorization_Password	いいえ	String	この属性は HTTP 基本認証のパスワードに対応します。詳しくは、46 ページの『要求処理用の HTTP 証明書伝搬』を参照してください。
1 つ以上の HTTP ヘッダー	いいえ	String	この属性を使用すれば、ハンドラーは、指定された HTTP ヘッダーの値の受け渡しまたは検索ができます。
UserDefinedProperties	いいえ	ビジネス・オブジェクト	この属性は、ユーザー定義のプロトコル・プロパティのビジネス・オブジェクトを保持します。詳しくは、44 ページの『要求処理のためのユーザー定義のプロパティ』を参照してください。
MessageTransformationMap	いいえ	単一カーディナリティー・ビジネス・オブジェクト	この属性は、0 個またはそれ以上のメッセージ変換規則を保持するビジネス・オブジェクトを指します。この変換規則には、規則に指定された着信メッセージに適用する MIME タイプおよび charset に関する情報が入っています。詳しくは、45 ページの『メッセージ変換マップ』を参照してください。

図 13 に、Business Object Designer における HTTP プロトコル構成 MO 属性の一部を示します。

General		Attributes								
	Pos	Name	Type	Key	Foreign	Required	Card	Maximum	Default	App Spec Info
1	1	OrderId	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255		
2	2	OrderHeader	CLIENT_SYNCH_OrderStatus_Request_Header	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1			
3	3	HTTPCfMo	CLIENT_SYNCH_OrderStatus_HTTPCfMo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1			
3.1	3.1	Date	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
3.2	3.2	Content-Type	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
3.3	3.3	MessageTransformationMap	HTTP_CfMo_MsgTrnstfMap	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1			
3.3.1	3.3.1	TransformationRule	HTTP_CfMo_MsgTrnstfRule	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	N			
3.3.1.1	3.3.1.1	Content-Type	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		255	/*	
3.3.1.2	3.3.1.2	MimeType	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
3.3.1.3	3.3.1.3	Charset	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
3.3.1.4	3.3.1.4	ObjectEventId	String							
3.3.2	3.3.2	ObjectEventId	String							
3.4	3.4	UserDefinedProperties	HTTP_CfMo_CustomProperties	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1			
3.4.1	3.4.1	CustomProperty1	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		
3.4.2	3.4.2	CustomProperty2	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ws_prop_type=Integer
3.4.3	3.4.3	CustomPropertyN	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		ws_prop_type=Boolean
3.4.4	3.4.4	ObjectEventId	String							
3.5	3.5	ObjectEventId	String							
4	4	ObjectEventId	String							
5	5			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		255		

図 13. 要求処理のための HTTP プロトコル構成 MO 属性

HTTP プロトコル構成 MO の属性を以下に示します。

- 『要求処理のためのユーザー定義のプロパティ』
- 45 ページの『メッセージ変換マップ』
- 46 ページの『要求処理用の HTTP 証明書伝搬』

要求処理のためのユーザー定義のプロパティ: オプションで、HTTP プロトコル構成 MO にカスタム・プロパティを指定できます。この指定を行うには、UserDefinedProperties 属性を組み込みます。この属性は、1 つ以上の子属性とプロパティ値を持つビジネス・オブジェクトに対応します。このビジネス・オブジェクトのすべての属性は、メッセージ・ヘッダーの変数部分で読み取られる (同期応答の場合は書き込まれる) 単一プロパティを以下のように定義する必要があります。

- 属性の型は、プロトコル・プロパティ・タイプにかかわらず、常に String です。属性のアプリケーション固有情報には、属性がマップされるプロトコル・メッセージ・プロパティ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入れることができます。

表 20 は、これらの属性のアプリケーション固有の情報を要約したものです。

表 20. ユーザー定義のプロトコル・プロパティ属性のアプリケーション固有情報: 名前 = 値のペアの内容

名前	値	説明
ws_prop_name (大文字小文字を区別しない。指定されなければ、属性名がプロパティ名として使用される)	任意の有効なプロトコル・プロパティ名	これはプロトコル・プロパティの名前です。ベンダーによっては、いくつかのプロパティが拡張機能に予約されています。
ws_prop_type	String	プロトコル・プロパティのタイプ。

指定されたカスタム・プロパティ ASI (ws_prop_name または ws_prop_type) が無効で、このヘッダーを処理する論理的方法 (HTTP 処理でプロパティ・タイプを無視するなど) がない場合、コネクタはログに警告を記録し、このプロパティを無視します。ws_prop_name または ws_prop_type に対しての必要な検査は実行されたが、カスタム・プロパティの値を設定することも検索することもできない場合、コネクタはログにエラーを記録し、イベントは失敗します。

UserDefinedProperties 属性が指定され、そのビジネス・オブジェクトのインスタンスが作成されると、コネクタは、この子ビジネス・オブジェクトの各属性を処理し、それに対応するメッセージ・プロパティ値を設定します。

同期要求処理の場合は、Web サービス/URL から応答メッセージを受信したときに、UserDefinedProperties 属性が指定されていると、コネクタは UserDefinedProperties ビジネス・オブジェクトのインスタンスを作成します。次に、メッセージからプロパティ値を抽出して、それを新規ビジネス・オブジェクトに保管します。少なくとも 1 つのプロパティ値が正常に検索されれば、コネクタは、変更された UserDefinedProperties ビジネス・オブジェクトをプロトコル構成 MO に設定します。

メッセージ変換マップ: メッセージ変換マップ (MTM) 機能は、要求処理 HTTP(S) プロトコル・ハンドラーでのみサポートされます。MessageTransformationMap はプロトコル構成 MO のオプションの属性で、ビジネス・オブジェクトを指します。このビジネス・オブジェクトには、規則に指定された MIME タイプおよび charset を持つメッセージを変換する規則が格納されています。属性名が MessageTransformationMap (大文字小文字を区別する) で、この属性がビジネス・オブジェクト・オブジェクト・タイプの場合 (図 13 を参照)、コネクタは、そのオブジェクト内の規則を使用して、メッセージを変換します。

図 13 に示すように、MTM 属性には、カーディナリティー N の子ビジネス・オブジェクト属性が 1 つあり、その属性名は TransformationRule です。メッセージの TransformationRule を検索するときに、SOAP/HTTP(S) プロトコル・ハンドラーはまず、そのメッセージが、すべての TransformationRule に指定された ContentType と正確に一致するかを確認します。一致しない場合、コネクタは、複数のメッセージ・タイプに適用される規則を検索します。プロトコル・ハンドラーの処理について詳しくは、62 ページの『HTTP-HTTPS プロトコル・ハンドラー処理』を参照してください。

TransformationRule ビジネス・オブジェクトの各インスタンスでは、表 21 に示すとおりに属性を指定する必要があります。

表 21. HTTP プロトコル構成 MO における MessageTransformationMap の TransformationRule 属性

属性名	必須	型	デフォルト値	説明
TransformationRule	いいえ	ビジネス・オブジェクト、カーディナリティー N		この属性は、メッセージ変換に関する規則を 1 つ保持します。MessageTransformationMap 属性の下にあるこの属性のインスタンスは 0 個の場合も、それ以上の場合もあります。

表 21. HTTP プロトコル構成 MO における MessageTransformationMap の TransformationRule 属性 (続き)

属性名	必須	型	デフォルト値	説明
+ContentType	はい	String	*/*	このプロパティの値は、この変換規則が適用されるメッセージの HTTP ContentType を示します。この属性にデフォルト値 */* が指定された場合、コネクタは、この規則を任意の ContentType に適用できます。プロトコル・ハンドラーの処理の詳細については、62 ページの『HTTP-HTTPS プロトコル・ハンドラー処理』を参照してください。他の規則と同じ ContentType を持つ規則を複数検出した場合、プロトコル・ハンドラーはログに警告を記録し、重複する規則をすべて無視しますが、固有の規則は使用します。
+MimeType	いいえ			このビジネス・オブジェクトに指定された ContentType のメッセージの処理でデータ・ハンドラーを呼び出すときに使用する MIME タイプ。
+Charset	いいえ			このビジネス・オブジェクトに指定された ContentType の要求を変換するときに使用する charset。

要求処理用の HTTP 証明書伝搬: 証明書伝搬のために、コネクタは HTTP プロトコル構成 MO の Authorization_UserID 属性および Authorization_Password 属性をサポートします。このサポートは、HTTP 基本認証スキームの一部としてのこれらの証明書の伝搬に限られます。

要求処理中に証明書を伝搬する必要がある場合は、SAP XI ODA によって生成されたプロトコル構成 MO に Authorization_UserID 属性および Authorization_Password 属性を手動で追加する必要があります。この操作は、ビジネス・オブジェクト定義およびメタ・オブジェクト定義を生成した後に Business Object Designer で行います。(SAP XI ODA の詳細については、125 ページの『第 6 章 要求処理のためのコラボレーションの有効化』を参照してください。)

コラボレーションは、プロトコル構成 MO の Authorization_UserID 属性および Authorization_Password 属性の値を設定します。これらの属性が null でも空でもない場合は、コネクタは、ターゲット Web サービスに送信する要求に許可ヘッダーを作成します。SOAP HTTP/HTTPS プロトコル・ハンドラーは、許可ヘッダーの作成時に *HTTP Authentication: Basic and Digest Access Authentication (RFC 2617)* に従います。

注: RFC2617 で規定されている HTTP 認証のダイジェスト認証方式も、オプションのチャレンジ・レスポンス機構もサポートされていません。信任状を要求する

サーバーを HTTP(S) プロトコル・ハンドラーが呼び出した場合、コネクターはサーバーからのチャレンジ・レスポンスを待ちません。代わりに、信任状を連続して送信します。

非同期要求処理 TLO

図 14 は、非同期要求処理のためのビジネス・オブジェクト階層を示しています。要求オブジェクトのみが必要ですが、このオブジェクトには、SOAP データ・ハンドラーのための SOAP 構成 MO と、SOAP/HTTP-HTTPS プロトコル・ハンドラーのためのプロトコル構成 MO が格納されています。これらについて、以下のセクションで説明します。

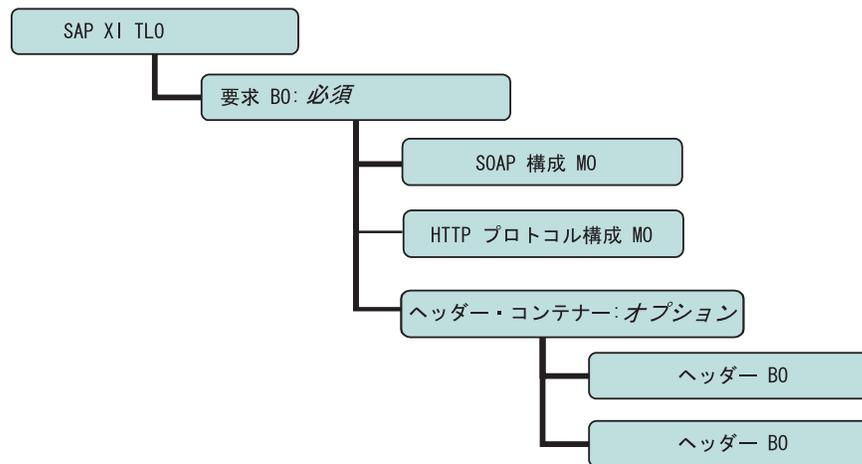


図 14. 非同期要求処理のためのビジネス・オブジェクト階層

TLO には、オブジェクト・レベルの ASI のほか、属性レベルの ASI を持つ属性が含まれています。両方の種類の ASI について、以下で説明します。ヘッダー・コンテナーおよびヘッダー子ビジネス・オブジェクトについての詳細は、32 ページの『ヘッダー・コンテナー・ビジネス・オブジェクト』を参照してください。

非同期イベント処理 TLO のオブジェクト・レベルの ASI

図 15 は、非同期要求処理のためのサンプル TLO である CLIENT_ASYNCH_Order_TLO を表しています。

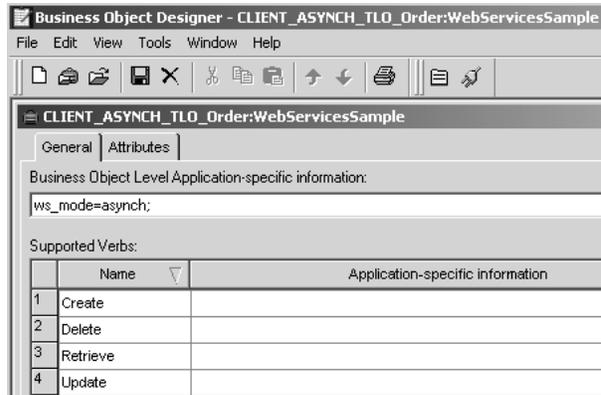


図 15. 非同期要求処理のためのトップレベル・ビジネス・オブジェクト

以下の表 22 では、非同期要求処理 TLO のオブジェクト・レベルの ASI について説明します。

表 22. 非同期要求処理 TLO のオブジェクト ASI

オブジェクト・レベルの ASI	説明
ws_mode=asynch	<p>要求処理の際に、コネクタは、この ASI プロパティを使用して、コラボレーションを同期 (synch) で呼び出すのか非同期 (asynch) で呼び出すのかを決定します。非同期要求処理の場合は、この ASI を asynch に設定する必要があります。</p> <p>デフォルトは asynch です。</p>

非同期要求処理 TLO のための属性レベルの ASI

図 16 は、要求処理 TLO のサンプルである CLIENT_ASYNC_TLO_Order の属性を表しています。

Pos	Name	Type	Key	Required	Card	Maximum Length	Default	App Spec Info
1	Handler	String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		255		soap/http
2	MimeType	String	<input type="checkbox"/>	<input type="checkbox"/>		255		xml/soap
3	BOPrefix	String	<input type="checkbox"/>	<input type="checkbox"/>		255		
4	Request	CLIENT_ASYNC_TLO_Order	<input type="checkbox"/>	<input type="checkbox"/>	1			ws_botype=request
5	ObjectEventId	String						

図 16. 非同期要求処理のための TLO 属性

表 23 は、非同期要求処理 TLO の要求属性に対する属性レベルの ASI を要約したものです。

表 23. 非同期要求処理 TLO の属性

TLO 属性	属性レベルの ASI	説明
MimeType	なし	この属性は、コネクタが呼び出すデータ・ハンドラーの MIME タイプを指定します。この属性は要求処理のみに使用されます。(イベント処理の場合、プロトコル・リスナーは SOAPDHMimeType コネクタ固有構成プロパティを使用します。) デフォルトは xml/soap です。
BOPrefix	なし	String タイプのこの属性は、今後の開発のために予約されているもので、必須ではありません。
Handler	なし	この属性は、Web サービス要求の処理に使用するプロトコル・ハンドラーを指定もので、要求処理専用です。この値は soap/http になります。コネクタは、SOAP/HTTP-HTTPS プロトコル・ハンドラーを使用して、この要求を処理します。デフォルトは soap/http です。
Request	ws_botype=request	この属性は、要求ビジネス・オブジェクトに対応します。コネクタはこの属性 ASI を使用して、この TLO 属性のタイプが SOAP 要求 BO であるかどうかを判別します。属性名ではなく、この ASI が属性タイプを判別します。複数の要求属性がある場合は、コネクタは最初の要求属性の ASI を使用します。

非同期要求処理のための要求ビジネス・オブジェクト

要求ビジネス・オブジェクトは、TLO の子であり、非同期要求処理の場合には必須です。非同期要求処理のための要求ビジネス・オブジェクトに対するオブジェクト・レベルの ASI は、表 24 で説明します。

表 24. 非同期要求処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI

オブジェクト・レベルの ASI	説明
cw_mo_soap=SOAPCfgMO	この ASI の値は、SOAP 構成 MO に対応する属性の名前と一致しなければなりません。これは、要求ビジネス・オブジェクトに対するデータ・ハンドラー変換を定義する SOAP 構成 MO です。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。

表 24. 非同期要求処理: 要求ビジネス・オブジェクトのオブジェクト・レベルの ASI (続き)

オブジェクト・レベルの ASI	説明
<code>cw_mo_http=SOAPHTTPCfgMO</code>	この ASI の値は、プロトコル構成 MO に対応する属性の名前と一致しなければなりません。これは、SOAP/HTTP-HTTPS プロトコル・ハンドラーの宛先を指定する、別のプロトコル構成 MO です。この ASI は、SOAP/HTTP-HTTPS プロトコル・ハンドラーによって使用されます。詳しくは、42 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。
<code>SOAPAction=SOAPActionURI</code>	コネクタはこの ASI を使用して、要求メッセージに SOAPAction ヘッダーを設定するかどうかを判別します。この ASI は、宛先 Web サービスが SOAPAction ヘッダーを必要とする場合にのみ指定してください。この ASI は要求処理に使用されるものであり、イベント通知には使用されません。

非同期要求処理の場合の SOAP 構成 MO、プロトコル構成 MO、SOAP ヘッダー・コンテナ、およびヘッダー子ビジネス・オブジェクトの要件および特性は、同期要求処理の場合と同じです。詳しくは、36 ページの『同期要求処理 TLO』における該当のトピックを参照してください。

非同期要求処理のための構成 MO

SOAP 構成 MO (SOAPCfgMO) の属性は、イベント処理用の SOAP 構成 MO の場合と同じです。詳しくは、29 ページの『SOAP 構成 MO』を参照してください。また、85 ページの『SOAP 構成メタオブジェクト: 各 SOAP ビジネス・オブジェクトの子』も参照してください。

要求処理の際に、SOAP/HTTP-HTTPS プロトコル・ハンドラーは、ターゲット SAP XI サービスの宛先を判別するために、HTTP プロトコル構成 MO を使用します。このプロトコル構成 MO は、要求ビジネス・オブジェクトの場合には必須です。詳しくは、42 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。

ビジネス・オブジェクトの開発

ビジネス・オブジェクトを作成する場合は、SAP XI ODA ツール、BIA_XIWSDLUtil ツール、および Business Object Designer を使用します。コネクタを構成してビジネス・オブジェクトをサポートするようにする場合は、Connector Configurator を使用します。Business Object Designer ツールの詳細については、「ビジネス・オブジェクト開発ガイド」および 127 ページの『第 7 章 Web サービスとしてのコラボレーションの公開』を参照してください。Connector Configurator の詳細については、173 ページの『付録 B. Connector Configurator』を参照してください。

第 4 章 SAP XI コネクタ

- 『コネクタ処理』
- 53 ページの『HTTP(S) サービス』
- 54 ページの『イベント処理』
- 61 ページの『要求処理』
- 67 ページの『SSL』
- 69 ページの『コネクタの構成』
- 79 ページの『始動時のコネクタ』
- 80 ページの『ロギング』
- 81 ページの『トレース』

この章では、SAP XI コネクタとその構成方法について説明します。

すべての WebSphere Business Integration コネクタは、統合ブローカーと連動して動作します。SAP XI コネクタは、IBM WebSphere InterChange Server 統合ブローカーと連動します。これについては、「テクニカル入門 (IBM WebSphere InterChange Server)」で説明されています。

コネクタは、アダプターのランタイム・コンポーネントです。コネクタは、アプリケーション固有のコンポーネントとコネクタ・フレームワークからなります。アプリケーション固有のコンポーネントには、特定のアプリケーションに合わせて調整されたコードが含まれています。コネクタ・フレームワーク (このコードは、すべてのコネクタで共通です) 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を果たします。コネクタ・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サービスを提供します。

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

本書には、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・フレームワークに関する情報が記載されています。本書では、これら両方のコンポーネントをコネクタと呼んでいます。

統合ブローカーとコネクタの関係に関する詳細については、「システム管理ガイド」を参照してください。

コネクタ処理

コネクタには、イベント処理を行うプロトコル・リスナー・フレームワークと要求処理を行うプロトコル・ハンドラー・フレームワークがあります。コネクタ・フレームワークでは、この双方向の機能を利用して以下のことが可能になります。

- SAP XI クライアントからの呼び出しを処理する (イベント処理)
- SAP XI サービスを呼び出すコラボレーションによる要求を処理する (要求処理)

イベント処理の概要

コネクター・イベント処理 (またはイベント通知) は、クライアントとして動作する SAP XI SOAP アダプターからの要求を処理するために使用します。このイベント処理機能は、以下のコンポーネントを含む、プロトコル・リスナー・フレームワークを採用しています。これらのコンポーネントについては、この章の後半でさらに詳しく説明します。

- HTTP プロトコル・リスナー
- HTTPS プロトコル・リスナー

コネクターはこれらのコンポーネントを使用して、クライアントからコラボレーションを呼び出すためのトランスポートを `listen` します。

クライアントからの要求が到着すると、リスナーは、要求メッセージをビジネス・オブジェクトに変換し、コラボレーションを呼び出します。これが同期要求である場合、コネクターは同一タイプの応答ビジネス・オブジェクトを、要求ビジネス・オブジェクトとして受信します。リスナーは、応答ビジネス・オブジェクトを応答メッセージに変換します。リスナーは、次に応答メッセージをクライアントにトランスポートします。イベントの順序は、このコネクターの要件ではないので注意してください。つまり、コネクターはイベントをどのような順序でもデリバリーできます。

HTTP コネクターは、構成済みのデータ・ハンドラーを使用して、着信した要求メッセージをビジネス・オブジェクトに変換します。着信した要求メッセージをどのビジネス・オブジェクトで解決すべきか、データ・ハンドラーが判別できるように、コネクターは、サポートされているビジネス・オブジェクトに関するメタ情報をデータ・ハンドラーに提供します。コネクターはまず最初に、サポートされているビジネス・オブジェクトの中から変換の候補となるすべてのビジネス・オブジェクトのリストを作成します。このリストは、サポートされている TLO のみで構成されます。サポートされている TLO ビジネス・オブジェクトは、オブジェクト・レベル ASI が `ws_eventtlo=true` のビジネス・オブジェクトです。

プロトコル・リスナーは、次のようにして TLO のオブジェクト・レベル ASI を読み取ります。

- `ws_collab=` これは、どのコラボレーションを呼び出すのかを決定します。
- `ws_mode=` これは、コラボレーションを呼び出す方法 (同期 (`synch`) または非同期 (`asynch`)) を決定します。

コネクターはデータ・ハンドラーから戻された要求ビジネス・オブジェクトを検査します。コネクターは、このビジネス・オブジェクトの `ws_tloname` ASI を使用して、親 TLO の名前を抽出します。この TLO のインスタンスが生成され、要求ビジネス・オブジェクトがこの TLO に設定されます。最後に、この構成済み TLO を使用して、コラボレーションを呼び出します。

コラボレーションを同期実行する場合、コネクターはデータ・ハンドラーを使用して、クライアントに戻す応答または障害メッセージを作成します。この場合、コネクターは、ビジネス・オブジェクト (TLO の子) をデータ・ハンドラーに渡すだけです。データ・ハンドラーは、渡されたビジネス・オブジェクトに基づいてメッセージを戻します。

要求処理の概要

コネクタは、コラボレーションのために、HTTP(S) を介して SAP XI Web サービスを呼び出すことができます。この要求処理機能は、プロトコル・ハンドラー・フレームワークによってサポートされています。プロトコル・ハンドラー・フレームワークは、HTTP-HTTPS プロトコル・ハンドラーから成る、構成可能なランタイム・モジュールです。これらのコンポーネントについては、この章の後半でさらに詳しく説明します。

プロトコル・ハンドラー・フレームワークは、常に TLO に設定されているコラボレーション要求ビジネス・オブジェクトを受信すると、プロトコル・ハンドラーをロードします。プロトコル・ハンドラーは、SAP XI サービスの呼び出しおよび (オプションで) 応答の保護に必要なトランスポート・レベルの詳細を管理し、コラボレーション要求ビジネス・オブジェクトの要求メッセージへの変換、要求メッセージによる SAP XI サービスの呼び出し、および応答メッセージのビジネス・オブジェクトへの変換ならびにそのオブジェクトのコラボレーションへのリターン (要求/応答 (同期) モードの場合)、という 3 つの主要なタスクを実行します。

SAP XI サービス・コネクタは常に TLO を使用してコラボレーションから呼び出されます。コネクタは、TLO から要求ビジネス・オブジェクトを判別して、このビジネス・オブジェクトによりデータ・ハンドラーを呼び出します。データ・ハンドラーは、コネクタにより送信された要求メッセージを SAP XI サービスに戻します。

同期実行する場合、コネクタはデータ・ハンドラーを使用して、応答および障害メッセージをそれぞれ 応答ビジネス・オブジェクトおよび障害ビジネス・オブジェクトに変換します。データ・ハンドラーが、これらの応答/障害をビジネス・オブジェクトに変換する際に、どのビジネス・オブジェクトにより解決すべきかを判別できるようにするために、コネクタは、特定のメタ情報をデータ・ハンドラーに提供します。具体的には、コネクタは、呼び出し側の TLO の子であるすべての応答および障害ビジネス・オブジェクトのリストを作成します。応答ビジネス・オブジェクトは 1 つだけ存在している必要があり、オプションで多数の障害ビジネス・オブジェクトが存在します。defaultfault ビジネス・オブジェクトが 1 つだけ存在している場合もあります。defaultfault ビジネス・オブジェクトの場合、コネクタは、データ・ハンドラーに defaultfault ビジネス・オブジェクトの名前を通知するだけです。この変換で解決する障害ビジネス・オブジェクトが他にない場合、最後の手段として、データ・ハンドラーによって defaultfault ビジネス・オブジェクトを解決する必要があります。

HTTP(S) サービス

SAP XI サービスは HTTP トランスポート・プロトコルをサポートします。HTTP は、HTTP クライアントが接続を開き、要求メッセージを SAP XI サーバーに送信するというクライアント/サーバー・モデルを実現します。クライアント要求メッセージは SAP XI サービスを呼び出します。SAP XI サーバーは、呼び出しを含むメッセージをディスパッチして、接続を閉じます。

コネクタの HTTP および HTTPS プロトコル・リスナーは、コラボレーションに対するクライアント要求を処理する際に HTTP クライアント/サーバーおよび要求/応答モデルを利用します。ただし、HTTP リスナーは、HTTP サーバー (プロキシ

一、仲介、またはそれ以外)として機能するようになっていません。HTTP リスナーは、どちらかといえば、企業内およびファイアウォールの内側で使用するエンドポイントとして機能します。したがって、クライアント要求をリスナーに送るには、ファイアウォール内に別の Web サーバーまたはゲートウェイを配置する必要があります。詳しくは、1 ページの『第 1 章 アダプターの概要』を参照してください。

同期 HTTP(S) サービス

コネクター処理の観点からは、同期 HTTP サービスは、要求/応答の流れをたどるサービスです。HTTP または HTTPS プロトコル・リスナーにより HTTP 要求メッセージが正常に処理された場合、メッセージ本文には応答および HTTP 状況コード 200 OK が記載されます。障害が戻された場合、本文には障害メッセージおよび状況コード 500 が記載されます。

非同期 HTTP(S) サービス

コネクター処理の観点からは、非同期 HTTP サービスは、要求専用の流れをたどるサービスです。HTTP または HTTPS プロトコル・リスナーにより、要求専用操作が正常に受信および処理された場合、HTTP 状況コード 202 Accepted が生成されます。HTTP 状況コード 200 OK が生成されるようにコネクターを構成することもできます。詳細については、表 32 の HTTPAsyncResponseCode プロパティを参照してください。障害が発生すると、HTTP 状況コード 500 が生成されます。応答はありませんが、障害の本文が戻されることがあります。

イベント処理

イベント処理中に、コネクターはプロトコル・リスナーと構成済みのデータ・ハンドラーを使用して、SAP XI サービス・クライアントの要求メッセージを、コラボレーションで操作できるビジネス・オブジェクトに変換します。プロトコル・リスナーは、イベント処理において極めて重要な役割を果たしています。

プロトコル・リスナー

SAP XI 要求は、HTTP トランスポートまたは HTTPS トランスポートを介して到着する場合があります。リスナーは、トランスポート・チャンネルにこのような要求が到着するのをモニターします。プロトコル・リスナーには次の 2 種類があり、それぞれに対応するチャンネルがあります。

- HTTP プロトコル・リスナー
- HTTPS プロトコル・リスナー

これらの各リスナーは、トランスポートで `listen` するスレッドで構成されます。クライアントから要求メッセージを受け取ると、リスナーは、プロトコル・リスナー・フレームワークにそのイベントを登録します。

プロトコル・リスナー・フレームワークは、プロトコル・リスナーを管理し、リソースが使用可能になったときに要求を処理するようにスケジューリングします。コネクター固有のプロパティに値を設定するときに、プロトコル・リスナー・フレームワークのリスナーおよび性質を構成してください。構成可能なプロトコル・リスナー・フレームワークのプロパティには、以下のものが含まれます。

- **WorkerThreadCount** プロトコル・リスナー・フレームワークが使用することのできるスレッドの合計数。これは、プロトコル・リスナー・フレームワークが同時に処理できる要求の数です。
- **RequestPoolSize** プロトコル・リスナー・フレームワークに登録できる要求の最大数。この最大数を超える要求を受け取ると、新規要求は登録されなくなります。

これら 2 種類のコネクター固有のプロパティは、プロトコル・リスナーが際限なくイベントを発生させてコネクターをふさいでしまわないように、メモリー割り振りを制御します。この割り振りアルゴリズムは、「コネクターは、 $WorkerThreadCount + RequestPoolSize$ に等しいイベントの合計数を常に受信できる」というものです。ここでは、*WorkerThreadCount* 数の要求を並行して処理できます。

追加のプロトコル・リスナーをプロトコル・リスナー・フレームワークに付け加えることができます。詳しくは、79 ページの『複数のプロトコル・リスナーの作成』および 70 ページの『コネクター固有の構成プロパティ』を参照してください。

HTTP および HTTPS プロトコル・リスナー処理

HTTP(S) プロトコル・リスナーは、クライアントの HTTP(S) 要求について継続的に *listen* するスレッドから構成されています。リスナー・スレッドは、コネクター固有の構成 (リスナー) プロパティ *Host* および *Port* で指定されているホストとポートをバインドします。別の構成プロパティ (*RequestWaitTimeout*) は、コネクターがシャットダウンされたかどうか確認するまでリスナーが要求を待つインターバルを定義します。

図 17 は、同期操作のための HTTP プロトコル・リスナー処理を表しています。

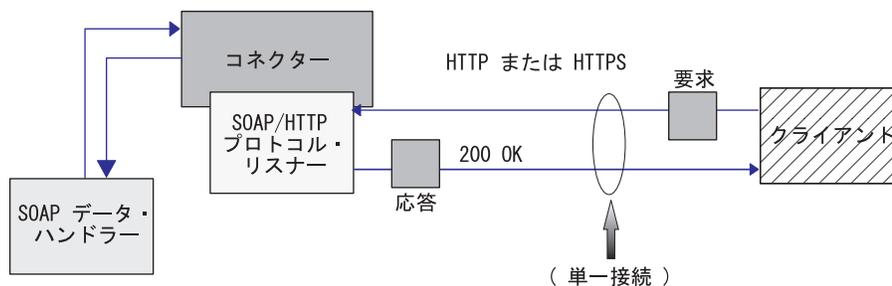


図 17. HTTP プロトコル・リスナー: 同期イベント処理

図 18 は、非同期操作のための HTTP プロトコル・リスナー処理を表しています。

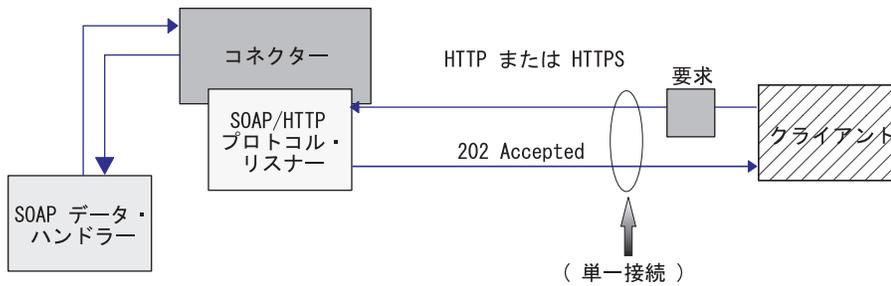


図 18. HTTP プロトコル・リスナー: 非同期イベント処理

クライアントは、HTTP 要求または HTTPS 要求を開始すると、要求メッセージを HTTP リスナーまたは HTTPS リスナーに送信します。プロトコル・リスナー URL を呼び出すためには、クライアントは HTTP POST メソッドを使用する必要があります。

HTTP(S) 要求が到着すると、リスナーは、要求をプロトコル・リスナー・フレームワークに登録し、リソースが使用可能になったときにイベントを処理するようにスケジュールします。リスナーは、次に、要求からプロトコル・ヘッダーとペイロードを抽出します。

表 25 は、インバウンド・メッセージの Charset、MimeType、ContentType、および Content-Type ヘッダーを決定する場合にリスナーが使用する規則の優先順位を要約したものです。

表 25. インバウンド・メッセージに対する HTTP(S) プロトコル・リスナーの処理規則

優先順位	Charset	MimeType	ContentType	Content-Type ヘッダー
1	着信 HTTP メッセージの Content-Type ヘッダー値から得られる Charset パラメーター値	このリスナーの URLsConfiguration コネクター・プロパティ値	Content-Type ヘッダー値から得られる着信 HTTP メッセージ・タイプ/サブタイプ値	着信 HTTP メッセージの Content-Type ヘッダー
2	このリスナーの URLsConfiguration プロパティ値			
3	要求メッセージ ContentType のタイプが、任意のサブタイプ (例えば、text/xml、text/plain など) を持つ text の場合は、ISO-8859-1 にデフォルト設定されます。それ以外の場合は、charset を使用しません。	ContentType (デフォルト)		

表 25 に示した内容から、次の項目を指定します。

- プロトコル・リスナーは、次の規則に従って、インバウンド・メッセージの Charset を決定します。
 1. プロトコル・リスナーは、HTTP メッセージ Content-Type ヘッダー値の charset パラメーターから Charset の抽出を試みます。
 2. Charset 値が、Content-Type ヘッダーから取得されない場合、プロトコル・リスナーは、そのリスナー用に URLsConfiguration プロパティー値を読み取ろうとします。
 3. Charset 値が、上記のステップで説明されている方法を使用しても取得されず、また、メッセージ ContentType のタイプが、任意のサブタイプ (例えば、text/xml、text/plain など) を持つ text の場合、リスナーは、デフォルト Charset 値の ISO-8859-1 を使用します。それ以外の場合は、Charset 値を使用しません。
- リスナーは、次の規則に従って、応答メッセージの MIMEType を決定します。
 1. ユーザーが、着信要求メッセージの使用する URL に対する TransformationRules の構成を終了し、要求 ContentType が TransformationRule の ContentType に一致する場合、リスナーは、TransformationRule を使用して、要求メッセージを要求ビジネス・オブジェクトに変換するために MIMEType 抽出します。リスナーは、要求された URL に対して、URLsConfiguration プロパティーの中で、ContentType 値 (例えば、text/soap) に基づき、正確に一致する TransformationRule を検索しようとしています。
 2. 検索が失敗すると、リスナーは、要求 URL (例えば、*/*) のもとで複数の ContentType に適用される TransformationRule を検索しようとしています。
 3. 上記のステップがすべて失敗し、MIMEType が決定できない場合は、データ・ハンドラーを起動し、要求メッセージを、要求ビジネス・オブジェクトに変換するために、ContentType の値が MIMEType として使用されます。
- リスナーは、着信 HTTP メッセージ Content-Type ヘッダーからタイプ/サブタイプを抽出して、ContentType を決定します。
- リスナーは、着信 HTTP メッセージ Content-Type ヘッダーのそれから、Content-Type ヘッダーを決定します。

コラボレーションを非同期に呼び出す場合、リスナーは、要求ビジネス・オブジェクトを統合ブローカーにデリバリーし、HTTP 状況コード 202 Accepted でクライアントに応答します。これでリスナー処理が完結します。

同期呼び出しの場合は、リスナーは同期させてコラボレーションを呼び出します。コラボレーションは応答ビジネス・オブジェクトを使用して応答します。

表 26 は、応答メッセージの Charset、MIMEType、ContentType、Content-Type ヘッダーを決定する際にリスナーが使用する規則の優先順位を要約したものです。

表 26. アウトバウンド同期応答メッセージに対する HTTP(s) プロトコル・リスナーの処理規則

優先順位	Charset	MIMEType	ContentType	Content-Type ヘッダー
1	プロトコル構成 MO の Content-Type ヘッ ダー	TLO の MIMEType プ ロパティー	プロトコル構成 MO の Content-Type ヘッ ダー	プロトコル構成 MO の Content-Type ヘッ ダー

表 26. アウトバウンド同期応答メッセージに対する HTTP(s) プロトコル・リスナーの処理規則 (続き)

2	TLO の Charset プロパティ値	要求メッセージの MIMEType。ただし要求と応答の ContentType が一致する場合のみ。	要求メッセージの ContentType	ContentType および Charset を使用して、Content-Type ヘッダーを構成します。
3	要求メッセージの Charset。ただし要求と応答の ContentType が一致する場合のみ。	MIMEType として ContentType 値を使用します。		
4	ContentType が text/* の場合、デフォルトは ISO-8859-1 です。それ以外の場合は、charset を使用しません。			

表 26 に示した内容から、次のことを指定します。

- リスナーは、次の規則に従って、応答メッセージの Charset を決定します。
 1. 応答ビジネス・オブジェクトのプロトコル構成 MO に Charset が指定されている場合は、その値を使用します。
 2. 応答ビジネス・オブジェクトのプロトコル構成 MO ヘッダーに Charset 値が指定されていない場合、リスナーは、TLO に Charset が指定されていることを検査します。
 3. TLO に Charset 値が指定されていない場合、応答と要求の ContentType が同じときは、要求の Charset が応答に使用されます。
 4. 上記のステップが失敗し、応答 Charset 値が決定できない場合に、メッセージ ContentType のタイプ部分が、任意のサブタイプ (例えば、text/soap、text/plain など) を持つ text であれば、リスナーは、デフォルト Charset 値の ISO-8859-1 を使用します。それ以外の場合は、Charset 値を使用しません。
- リスナーは、次の規則に従って、応答メッセージの MIMEType を決定します。
 1. TLO の MIMEType 属性を使用します。
 2. TLO MIMEType 属性が欠落し、要求と応答の ContentType が一致する場合、リスナーは、応答メッセージに要求の MIMEType を使用します。
 3. 上記以外の場合、リスナーは、MIMEType として、ContentType 値を使用します。
- リスナーは、次の規則に従って、応答メッセージの ContentType を決定します。
 1. Content-Type ヘッダーが、応答ビジネス・オブジェクトのプロトコル構成 MO に指定されている場合は、Content-Type ヘッダーのタイプ/サブタイプ部分が、ContentType として使用されます。
 2. Content-Type ヘッダーが、応答ビジネス・オブジェクトのプロトコル構成 MO に指定されていない場合、リスナーは、(Charset が応答メッセージに対して決定されている場合) 決定されている ContentType と Charset を使用して、Content-Type ヘッダーを構成します。

リスナーは HTTP プロトコル構成 MO を処理します。HTTP プロトコル構成 MO で渡されたヘッダー値を、この要求応答イベント環境に合った正しい値にすることは、コラボレーション側の責任となります。リスナーは、次の規則に従って、標準ヘッダーおよびカスタム・プロパティーを取り込みます。

1. リスナーは、特殊な属性 (ObjectEventId など) を無視するため、HTTP プロトコル構成 MO の各項目を調べます。
2. 空でない各ヘッダーが、発信メッセージに書き込まれ、追加処理 (例えば、Content-Type ヘッダーに対する処理) が行われる場合があります。
3. 上記の方法では、メッセージに標準でないヘッダーが設定される可能性があるにもかかわらず、リスナーは、メッセージが論理的または意味的に正しいかをチェックしないことに注意してください。
4. HTTP プロトコル構成 MO UserDefinedProperties 属性に 1 つ以上のカスタム・プロパティーがある場合、リスナーは、それらをエンティティー・ヘッダー・セクション (最後のヘッダー・セクション) に追加します。カスタム・プロパティーの詳細については、30 ページの『イベント処理のためのユーザー定義のプロパティー』を参照してください。

注: HTTP プロトコル構成 MO のヘッダー Connection、Trailer、Transfer-Encoding、Content-Encoding、Content-Length、Content-MD5、Content-Range の中からいずれを指定しても、ほとんどの場合、正しくない HTTP メッセージが作成されます。

リスナーは、次に、データ・ハンドラーを呼び出して、コラボレーションによって戻された応答ビジネス・オブジェクトを応答メッセージに変換します。

リスナーは、応答メッセージをクライアントにデリバリーし、200 OK HTTP 状況コードを組み込みます。コラボレーションにより障害ビジネス・オブジェクトが戻された場合は、障害メッセージに変換されます。この障害メッセージは、500 Internal Server Error HTTP コードと共にクライアントにデリバリーされます。

リスナーは、次に、接続を閉じ、イベントを処理したスレッドは使用可能になります。

HTTP プロトコル・リスナーでサポートされない処理機能

HTTP プロトコル・リスナーでは、以下のサポートをしていません。

- キャッシュ: プロトコル・リスナーは、HTTP 仕様 (RFC2616) で規定されているキャッシュ機能を実行しません。
- プロキシ: プロトコル・リスナーは、HTTP 仕様 (RFC2616) で規定されているプロキシ機能を実行しません。
- 持続接続: プロトコル・リスナーは、HTTP 仕様 (RFC2616) で規定されている持続接続をサポートしていません。その代わりに、プロトコル・リスナーでは、各 HTTP 接続のスコープは単一のクライアント要求であると想定されています。したがって、サービス要求が完了すると、接続は閉じられます。プロトコル・リスナーは、複数のサービス呼び出しの間で接続を再利用することはありません。
- リダイレクト: プロトコル・リスナーはリダイレクトをサポートしていません。
- 大容量ファイル転送: プロトコル・リスナーは、大容量ファイルの転送には使用できません。その代わりに、参照による大容量ファイルの受け渡しは可能です。

- 状態管理: プロトコル・リスナーは、RFC2965 で示されている HTTP 状態管理機構をサポートしていません。
- Cookies: プロトコル・リスナーは Cookies をサポートしていません。

セキュア・ソケットを使用した HTTPS リスナー処理

HTTPS プロトコル・リスナー処理は HTTP プロトコル・リスナー処理のセクションで説明されているとおりですが、HTTPS ではセキュア・ソケットを使用するという点が異なります。詳しくは、67 ページの『SSL』を参照してください。

イベントの永続性とデリバリー

イベントの永続性は、プロトコルによって決まります。

- HTTP プロトコル・リスナー 永続性がなく、デリバリーは保証されません
- HTTPS プロトコル・リスナー 永続性がなく、デリバリーは保証されません

イベントの順序付け

コネクタは、任意の順序でイベントをデリバリーすることができます。

イベントのトリガー

イベント・トリガーのメカニズムは、プロトコル・リスナーの構成方法によって異なります。

- HTTP プロトコル・リスナー HTTP 接続要求の場合、listen は ServerSocket を介して行われます。
- HTTPS プロトコル・リスナー HTTPS 接続要求の場合、listen は ServerSocket 層を介して行われます。

注: コネクタは、Create (作成)、Update (更新)、Retrieve (検索) または Delete (削除) の区別を行いません。これらのイベントは、すべて同じ方法で扱われます。

イベントの検出

イベントの検出は、それぞれのプロトコル・リスナーによって行われます。イベント検出のメカニズムは、トランスポート、および各リスナーごとのコネクタ固有プロパティの構成方法に、完全に依存しています。これらのプロパティの詳細については、70 ページの『コネクタ固有の構成プロパティ』を参照してください。

イベント状況

イベント状況はプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- HTTP プロトコル・リスナー HTTPS は本来、非永続的で同期的なものです。したがって、イベント状況は維持されません。
- HTTPS プロトコル・リスナー HTTP は本来、非永続的で同期的なものです。したがって、イベント状況は維持されません。

イベントの検索

イベントの検索はプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー**・イベントの検索は、ソケットから HTTP 要求を取り出すことによって行われます。
- **HTTPS プロトコル・リスナー**・イベントの検索は、ソケットから HTTP 要求を取り出すことによって行われます。

イベントのアーカイブ

イベントのアーカイブはプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー**・トランスポートが非永続的であり、同期的であるため、アーカイブは行われません。
- **HTTPS プロトコル・リスナー**・トランスポートが非永続的であり、同期的であるため、アーカイブは行われません。

イベントのリカバリー

イベントのリカバリーはプロトコル・リスナーによって管理され、トランスポート、およびリスナーの構成方法によって異なります。

- **HTTP プロトコル・リスナー**・トランスポートが非永続的であるため、イベント・リカバリーは行われません。
- **HTTPS プロトコル・リスナー**・トランスポートが非永続的であるため、イベント・リカバリーは行われません。

要求処理

コネクターの要求処理機能を使用して、コラボレーションから HTTP サービスを呼び出すことができます。コネクターおよびその要求処理コンポーネント (プロトコル・ハンドラー・フレームワークおよびプロトコル・ハンドラー) を構成する必要があります。

コネクターは、実行時にビジネス・オブジェクトの形でコラボレーションから要求を受け取ります。ビジネス・オブジェクト (要求、およびオプションで応答および障害ビジネス・オブジェクト) は、TLO に格納され、SAP XI サービスを使用するように構成されているコラボレーションによって発行されます。TLO およびその子ビジネス・オブジェクトには、処理モード (同期または非同期) を指定する属性および ASI、データ・ハンドラー MIME タイプ、使用するプロトコル・ハンドラーの種類、およびターゲットのアドレスが入っています。プロトコル・ハンドラーは、この情報を使用して、データ・ハンドラーのインスタンスを呼び出し、要求ビジネス・オブジェクトを要求メッセージに変換し、ターゲット SAP XI サービスを呼び出します。同期モードの場合、プロトコル・ハンドラーは、データ・ハンドラーを再度呼び出し、応答メッセージを応答ビジネス・オブジェクトに変換して、これをコラボレーションに返送します。

コネクターは、要求メッセージに対する応答として、リモート側の取引先から次のいずれかを受け取ることができます。

- データを含んでいる応答メッセージ
- 障害情報を含んでいる応答メッセージ

プロトコル・ハンドラーは、要求処理において重要な役割を果たしています。

プロトコル・ハンドリング

コラボレーションは、HTTP トランスポートまたは HTTPS トランスポートを介して SAP XI サービスを呼び出すことができます。コネクターには、1 つのプロトコル・ハンドラーとこれに対応するチャンネルがあります。それは、HTTP サービスと HTTPS サービスを呼び出すための HTTP-HTTPS プロトコル・ハンドラーです。

プロトコル・ハンドラー・フレームワークはプロトコル・ハンドラーを管理し、起動時にプロトコル・ハンドラーをロードします。コネクターが要求ビジネス・オブジェクトを受け取ると、要求スレッド (それぞれのコラボレーション要求は、独自のスレッドで送られてきます) は、プロトコル・ハンドラー・フレームワークを呼び出して、要求を処理します。

プロトコル・ハンドラー・フレームワークは、TLOs Handler 属性 ASI を読み取り、使用するプロトコル・ハンドラーを決定します。一連の規則 (『HTTP-HTTPS プロトコル・ハンドラー処理』を参照) を適用すると、プロトコル・ハンドラーはデータ・ハンドラーを呼び出して、要求ビジネス・オブジェクトを要求メッセージに変換します。プロトコル・ハンドラーは、要求メッセージをトランスポート (HTTP(S)) メッセージにパッケージします。

次に、プロトコル・ハンドラーは、要求ビジネス・オブジェクトのプロトコル構成 MO の Destination 属性を読み取り、ターゲット・アドレスを判別します。プロトコル・ハンドラーは、次に、要求メッセージを使用してターゲット SAP XI サービスを呼び出します。

プロトコル・ハンドラーは、ws_mode TLO ASI を読み取り、処理モードが同期または非同期のいずれであるかを判別します。この ASI が asynch に設定されていると、プロトコル・ハンドラー処理は完了します。このように設定されていない場合、プロトコル・ハンドラーは応答メッセージを待ちます。応答メッセージが到着すると、プロトコル・ハンドラーは、プロトコル・ヘッダーとペイロードを抽出します。次にデータ・ハンドラー (MimeType TLO 属性によって指示される) を呼び出して、メッセージを応答または障害ビジネス・オブジェクトに変換します。プロトコル・ハンドラーは、プロトコル構成 MO を再度使用して、プロトコル・ヘッダーをビジネス・オブジェクトに設定します。プロトコル・ハンドラーは、次に、応答または障害ビジネス・オブジェクトをコラボレーションに戻します。

コネクターの構成によっては、1 つまたは複数のプロトコル・ハンドラーがコネクターにプラグされている場合があります。コネクター固有のプロパティを指定することにより、プロトコル・ハンドラーを構成することができます。

HTTP-HTTPS プロトコル・ハンドラー処理

HTTP-HTTPS プロトコル・ハンドラーは、このセクションに記載している点を除き、『プロトコル・ハンドリング』で説明しているように動作します。図 19 は、同期操作のための HTTP-HTTPS プロトコル・ハンドラーを表しています。

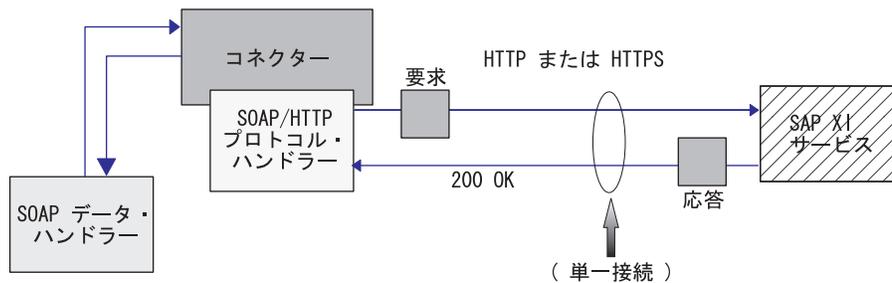


図 19. HTTP-HTTPS プロトコル・ハンドラー: 同期要求処理

図 20 は、非同期要求処理のための HTTP-HTTPS プロトコル・ハンドラーを表しています。

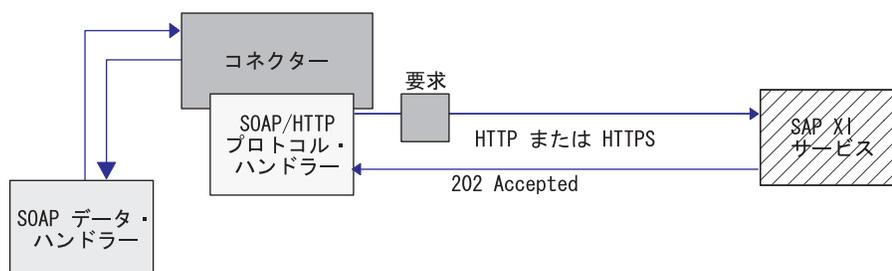


図 20. HTTP-HTTPS プロトコル・ハンドラー: 非同期要求処理

注: このセクションでは、HTTP プロトコル処理についてのみ説明します。

HTTP-HTTPS プロトコル・ハンドラーは、要求ビジネス・オブジェクトのオブジェクト・レベル ASI (cw_mo_http) を使用して、プロトコル構成 MO を決定します。HTTP-HTTPS プロトコル・ハンドラーは、HTTP プロトコル構成 MO の Destination 属性を読み取り、ターゲット SAP XI サービスの URL を決定します。URL が欠落している、あるいは不完全であれば、プロトコル・ハンドラーはサービス呼び出しで失敗します。HTTP プロトコル構成 MO およびその属性の詳細については、42 ページの『要求処理のための HTTP プロトコル構成 MO』を参照してください。

HTTP-HTTPS プロトコル・ハンドラーは、データ・ハンドラーによって戻される要求メッセージを使用して、SAP XI サービスを呼び出します。HTTP プロキシ・コネクタの構成プロパティが指定されている場合、HTTP-HTTPS プロトコル・ハンドラーは、それに応じた振る舞いをします。応答が戻されると、HTTP-HTTPS プロトコル・ハンドラーはそれを読み取ります。

64 ページの表 27 は、発信要求メッセージの Charset、MimeType、ContentType、および Content-Type ヘッダーを決定する際に HTTP-HTTPS プロトコル・ハンドラーが使用する規則の優先順位を要約したものです。

表 27. アウトバウンド・メッセージに対する HTTP-HTTPS プロトコル・ハンドラーの処理規則

優先順位	Charset	MimeType	ContentType	Content-Type ヘッダー
1	プロトコル構成 MO の Content-Type ヘッダー	TLO 属性の MimeType プロパティー	プロトコル構成 MO の Content-Type ヘッダー	プロトコル構成 MO の Content-Type ヘッダー
2	TLO 属性の Charset プロパティー	ContentType (デフォルト)		
3	ContentType が text/* の場合、デフォルトは ISO-8859-1 です。それ以外の場合は、charset を使用しません。			

表 27 に示したように、以下を決定します。

- HTTP-HTTPS プロトコル・ハンドラーは、次の規則に従って、応答メッセージの Charset を決定します。
 1. 要求ビジネス・オブジェクトのプロトコル構成 MO ヘッダーに Charset 値が指定されている場合は、その値を使用します。
 2. 上記のステップで Charset が決定しなかった場合、プロトコル・ハンドラーは TLO 属性から Charset を抽出しようとします。
 3. 上記のステップに示した操作が失敗した場合は、次の表を使用して Charset を決定します。

表 28. Charset のデフォルト要求処理

ContentType	デフォルトの Charset
text/*	ISO-8859-1 詳しくは、RFC2616 を参照してください。
application/*	デフォルトなし
その他	デフォルトなし

4. 上記のステップで Charset が決定した場合は、その Charset がデータ・ハンドラーに設定されます。
 5. データ・ハンドラーは、ストリームまたはバイト配列 API (要求の作成に必要なデータ構造に依存する) を使って呼び出されます。
- HTTP-HTTPS プロトコル・ハンドラーは、次の規則に従って、要求の MimeType を決定します。
 1. TLO MimeType 属性を使用します。
 2. TLO の MimeType 属性がない場合、プロトコル・ハンドラーは ContentType を使用して MimeType を決定します。
 - HTTP-HTTPS プロトコル・ハンドラーは、次の規則に従って、要求メッセージの ContentType を決定します。
 - Content-Type ヘッダーが、要求ビジネス・オブジェクトのプロトコル構成 MO に指定されている場合は、Content-Type ヘッダーのタイプ/サブタイプが、ContentType として使用されます。

- HTTP-HTTPS プロトコル・ハンドラーは、次の規則に従って、要求メッセージの Content-Type ヘッダーを決定します。
 - 要求ビジネス・オブジェクトのプロトコル構成 MO に Content-Type ヘッダーが指定されている場合は、その値を発信メッセージに設定します。

表 29 は、応答メッセージの Charset、MimeType、ContentType、および Content-Type ヘッダーを決定する際にハンドラーが使用する規則の優先順位を要約したものです。

表 29. インバウンド同期応答メッセージに対する HTTP(S) プロトコル・ハンドラーの処理規則

優先順位	Charset	MimeType	ContentType	Content-Type ヘッダー
1	着信 HTTP メッセージの Content-Type ヘッダー値から得られる Charset パラメーター値	要求ビジネス・オブジェクトのプロトコル構成 MO の MessageTransformationMap 子ビジネス・オブジェクト	Content-Type ヘッダー値から得られる着信 HTTP メッセージ・タイプ/サブタイプ値	着信 HTTP メッセージの Content-Type ヘッダー
2	要求ビジネス・オブジェクトのプロトコル構成 MO の MessageTransformationMap 子ビジネス・オブジェクト	要求メッセージの MimeType。ただし要求と応答の ContentType が一致する場合のみ。		
3	要求メッセージの Charset。ただし要求と応答の ContentType が一致する場合のみ。	TLO の MimeType プロパティ		
4	TLO の Charset プロパティ	ContentType (デフォルト)		
5	Content-Type が text/* の場合、デフォルトは ISO-8859-1 です。それ以外の場合は、Charset を使用しません。			

表 29 に示したように、以下を決定します。

- プロトコル・ハンドラーは、次の規則に従って、同期応答メッセージの Charset を決定します。
 1. Charset パラメーターが着信応答メッセージの Content-Type ヘッダーに設定されている場合、プロトコル・ハンドラーは、データ・ハンドラーに設定されている Charset 値を使用します。
 2. 応答メッセージ・ヘッダーに Charset 値がない場合、プロトコル・ハンドラーは、TLO 要求プロトコル構成 MO の MessageTransformationMap から、コラレーションが定義した Charset を読み取ろうとします。
 3. ある要求の MessageTransformationMap に Charset 値が指定されていない場合、応答と要求の ContentType が同じときは、要求の Charset が応答に使用されます。

4. 上記のステップで Charset 値を決定できない場合、プロトコル・ハンドラーは TLO の Charset 属性を読み取ろうとします。
 5. Charset 値が、上記のステップで説明されている方法を使用しても取得されず、また、メッセージ ContentType のタイプが、任意のサブタイプ (例えば、text/xml、text/plain など) を持つ text の場合は、ISO-8859-1 にデフォルト設定されます。それ以外の場合は、Charset 値を使用しません。
- プロトコル・ハンドラーは、次の規則に従って、同期応答メッセージの MIMEType を決定します。
 1. プロトコル・ハンドラーはまず、TLO 要求プロトコル構成 MO の MessageTransformationMap から MIMEType を抽出しようとしています。特に、プロトコル・ハンドラーは、MTM の中で正確に一致する ContentType を検索して、MessageTransformationRule を抽出した後、その中の MIMEType プロパティ値を使用しようとしています。検索できない場合、プロトコル・ハンドラーは、複数の ContentType に適用される MessageTransformationRule (ContentType が */*) を検索します。
 2. MessageTransformationMap を使用しても MIMEType が決定しない場合は、プロトコル・ハンドラーは、要求と応答の ContentType が一致する場合に限り、要求の MIMEType を応答に使用します。
 3. 上記のステップを使用しても MIMEType を抽出できない場合、プロトコル・ハンドラーは、TLO の MIMEType 属性を使用します。
 4. 上記のすべてのステップが失敗した場合、プロトコル・ハンドラーは、ContentType を使用して MIMEType を設定します。
 - ハンドラーは、着信 HTTP メッセージ Content-Type ヘッダーからタイプ/サブタイプを抽出して、ContentType を決定します。

ハンドラーは HTTP プロトコル構成 MO を処理します。HTTP プロトコル構成 MO で渡されたヘッダー値を、この要求応答イベント環境に合った正しい値にすることは、コラボレーション側の責任となります。ハンドラーは、次の規則に従って、標準ヘッダーおよびカスタム・プロパティを取り込みます。

1. ハンドラーは、特殊な属性 (ObjectEventId など) を無視するため、HTTP プロトコル構成 MO の各項目を調べます。
2. 空でない各ヘッダーが、発信メッセージに書き込まれ、追加処理 (例えば、Content-Type ヘッダーに対する処理) が行われる場合があります。
3. 上記の方法では、メッセージに標準でないヘッダーが設定される可能性があるにもかかわらず、ハンドラーは、メッセージの論理的または意味的な正しさを保証しないことに注意してください。
4. HTTP プロトコル構成 MO UserDefinedProperties 属性に 1 つ以上のカスタム・プロパティがある場合、ハンドラーは、それらをエンティティ・ヘッダー・セクション (最後のヘッダー・セクション) に追加します。カスタム・プロパティの詳細については、44 ページの『要求処理のためのユーザー定義のプロパティ』を参照してください。

注: HTTP プロトコル構成 MO のヘッダー Connection、Trailer、Transfer-Encoding、Content-Encoding、Content-Length、Content-MD5、Content-Range の中からいずれを指定しても、ほとんどの場合、正しくない HTTP メッセージが作成されます。

SSL

このセクションでは、コネクターの SSL 機能のインプリメント方法について説明します。背景情報については SSL の資料を参照してください。このセクションでは、SSL テクノロジーについて十分に理解していることを前提としています。

JSSE

コネクターは、JSSE を使用して HTTPS および SSL をサポートします。IBM JSSE はコネクターと共に出荷されます。この機能を使用可能にするには、`java.security` ファイルに次の項目があることを確認してください。このファイルは、コネクターと一緒にインストールされるファイルの 1 つです。

```
security.provider.5=com.ibm.jsse.IBMJSSEProvider
```

`java.security` は、コネクターのインストール先システムの `$ProductDir\lib\security` ディレクトリーにあります。コネクターは、`JavaProtocolHandlerPackages` コネクター・プロパティの値を使用して、システム・プロパティ `java.protocol.handler.pkgs` を設定します。コネクターと共に出荷された IBM JSSE の場合、このプロパティの値は `com.ibm.net.ssl.internal.www.protocol` に設定する必要があることに注意してください。

`JavaProtocolHandlerPackages` 構成プロパティのデフォルト値は、この値になります。ただし、このシステム・プロパティに別の値を指定すると、コネクターは初期化時にその値を使用します。`java.protocol.handler.pkgs` の値を指定しなかった場合、コネクターは初期化時にデフォルト値を使用します。

コネクターは、初期化中に、JSSE でサポートされる無名の暗号スイートをすべて使用不可にします。

KeyStore および TrustStore

コネクターで SSL を使用するには、鍵ストアとトラストストアをセットアップする必要があります。鍵ストア、証明書、および鍵生成のセットアップ用ツールは提供されていません。これらの作業を完了するには、サード・パーティーのソフトウェア・ツールを使用する必要があります。

SSL プロパティ

コネクター固有の SSL プロパティとして、以下のプロパティを指定することができます。

- `SSLVersion`
- `SSLDebug`
- `KeyStore`
- `KeyStoreAlias`
- `KeyStorePassword`
- `TrustStore`
- `TrustStorePassword`

これらのプロパティは、コネクタ・インスタンスに適用されることに注目してください。コネクタにプラグインされるすべての HTTPS プロトコル・リスナーと、コネクタ・インスタンスごとの HTTP-HTTPS プロトコル・ハンドラーにより、同じ SSL プロパティ値のセットが使用されます。HTTPS/SSL セットアップの詳細については、193 ページの『付録 D. HTTPS/SSL の構成』を参照してください。

SSL および HTTPS プロトコル・リスナー

HTTPS プロトコル・リスナーを使用するには、コネクタ固有の SSL プロパティを指定する必要があります。これらのプロパティに割り当てる値は、以下の SSL 要件を満たしている必要があります。

- **SSLVersion** 使用する SSLVersion が JSSE によってサポートされていることを確認してください。
- **KeyStore** HTTPS プロトコル・リスナーは、SSL 通信のサーバーとして動作するので、鍵ストアを指定する必要があります。リスナーは、SSL->KeyStore 構成プロパティに指定されている鍵ストアを使用します。このプロパティの値は、鍵ストア・ファイルの完全パスでなければなりません。鍵ストアにはコネクタ用の鍵のペア（秘密鍵と公開鍵）があることを確認してください。秘密鍵の別名を、SSL->KeyStoreAlias プロパティで指定する必要があります。鍵ストアにアクセスする際に必要なパスワードは、SSL-> KeyStorePassword プロパティで指定しなければなりません。鍵ストアにアクセスする際に必要なパスワードと秘密鍵（鍵ストアにある）が同じであることも確認してください。最後に、コネクタのデジタル証明書をクライアントに配布して、コネクタの認証ができるようにする必要があります。
- **TrustStore** HTTPS プロトコル・リスナーでクライアントの認証を行うようにする場合は、クライアント認証をアクティブしておく必要があります。このためには、SSL -> UseClientAuth プロパティを true に設定します。以下も指定する必要があります。
 - トラストストアのロケーション (SSL -> TrustStore 構成プロパティの値として)
 - トラストストアにアクセスする際に必要なパスワード (SSL -> TrustStorePassword プロパティの値として)

トラストストアには、クライアントのデジタル証明書が含まれていることを確認してください。クライアントで使用されるデジタル証明書は、自己署名するか、または CA から発行されます。トラストストアが CA のルート証明書を信頼すると、JSSE は、その CA によって発行されたすべてのデジタル証明書を認証することになるので注意してください。

HTTPS/SSL セットアップの詳細については、193 ページの『付録 D. HTTPS/SSL の構成』を参照してください。

SSL および HTTP-HTTPS プロトコル・ハンドラー

HTTP-HTTPS プロトコル・ハンドラーで SSL を使用する場合は、コネクタ固有の SSL プロパティを指定する必要があります。これらのプロパティに割り当てる値は、HTTP プロバイダーの以下の HTTPS/SSL 要件を満たしている必要があります。

- **SSLVersion** 使用する SSLVersion がプロバイダーおよび JSSE によってサポートされていることを確認してください。
- **TrustStore** HTTP-HTTPS プロトコル・ハンドラーは SSL 通信ではクライアントとして動作するため、トラストストアをセットアップする必要があります。ハンドラーは、SSL -> Truststore 構成プロパティに指定されているトラストストアを使用します。このプロパティの値は、トラストストア・ファイルの完全パスでなければなりません。トラストストアにアクセスする際に必要なパスワードは、SSL -> TrustStorePassword プロパティに指定しなければなりません。トラストストアには、プロバイダーのデジタル証明書が含まれていることを確認してください。プロバイダーで使用されるデジタル証明書は、自己署名するか、または CA から発行されます。トラストストアが CA のルート証明書を信頼すると、JSSE は、その CA によって発行されたすべてのデジタル証明書を認証することになるので注意してください。
- **KeyStore** HTTP サービス・プロバイダーでクライアント認証が必要な場合は、鍵ストアをセットアップする必要があります。HTTP-HTTPS プロトコル・ハンドラーは、SSL -> KeyStore 構成プロパティに指定されている鍵ストアを使用します。この値は、鍵ストア・ファイルの完全パスでなければなりません。鍵ストアにはコネクター用に構成された鍵のペア (秘密鍵と公開鍵) があることを確認してください。秘密鍵の別名を、SSL -> KeyStoreAlias プロパティに指定する必要があります。鍵ストアにアクセスする際に必要なパスワードは、SSL -> KeyStorePassword プロパティに指定しなければなりません。最後に、鍵ストアにアクセスする際に必要なパスワードと秘密鍵 (鍵ストアにある) が同じであることを確認してください。認証のためにコネクターのデジタル証明書を HTTP サービス・プロバイダーに配布する必要があります。

HTTPS/SSL セットアップの詳細については、193 ページの『付録 D. HTTPS/SSL の構成』を参照してください。

コネクターの構成

Installer を使用してコネクター・ファイルをシステムにインストールした後で、標準およびアプリケーション固有のコネクター構成プロパティを設定する必要があります。

構成プロパティの設定

コネクターには、2 種類の構成プロパティがあります。標準の構成プロパティと、コネクター固有の構成プロパティです。コネクターを稼働させる前に、System Manager (SM) を使用して、これらのプロパティの値を設定する必要があります。

標準構成プロパティ

標準構成プロパティは、すべてのコネクターによって使用される情報を提供します。これらのプロパティの説明については、153 ページの『付録 A. コネクターの標準構成プロパティ』を参照してください。下の表には、この付録で説明する構成プロパティに関する、このコネクターに固有の情報が示されています。

プロパティ	説明
CharacterEncoding	このコネクターはこのプロパティを使用しません。

プロパティ	説明
Locale	このコネクタは国際化に対応していないため、このプロパティの値は変更できません。現在サポートされているロケールについては、コネクタのリリース情報を参照してください。

このコネクタが統合ブローカーとしてサポートするのは **InterChange Server (ICS)** のみであるため、関連する構成プロパティは **ICS** 用のプロパティのみです。

少なくとも、以下の標準コネクタ構成プロパティを設定する必要があります。

- AgentTraceLevel
- ApplicationName
- ControllerTraceLevel
- DeliveryTransport

コネクタ固有の構成プロパティ

コネクタ固有の構成プロパティは、実行時にコネクタ・エージェントが必要とする情報を提供します。また、コネクタ固有のプロパティを使用することにより、エージェントの再コーディングおよび再ビルドを行わずに、コネクタ・エージェント内の静的な情報または論理を変更することができます。

表 30 には、コネクタ固有の構成プロパティがリストされています。プロパティの説明については、以下の各セクションを参照してください。プロパティの中には、別のプロパティを含んでいるものもあります。+ 文字は、プロパティ階層内の項目の位置を示しています。

表 30. コネクタ固有の構成プロパティ

名前	指定可能な値	デフォルト値	必須
DataHandlerMetaObjectName	データ・ハンドラーのメタオブジェクト名	MO_DataHandler_Default	はい
JavaProtocolHandlerPackages	有効な Java プロトコル・ハンドラー・パッケージ	com.ibm.net.ssl. internal.www.protocol	いいえ
ProtocolHandlerFramework	これは階層プロパティであり、値はありません。	なし	いいえ
+ProtocolHandlers	これは階層プロパティであり、値はありません。		いいえ
++Handler1	これは階層プロパティです。このプロパティのサブプロパティについては、72 ページの『Handler1』を参照してください。		はい
ProtocolListenerFramework	これは階層プロパティであり、値はありません。		いいえ
+WorkerThreadCount	使用可能なリスナー・スレッド数を指定する 1 以上の整数。	10	いいえ
+RequestPoolSize	リソース・プール・サイズを指定する WorkerThreadCount より大きい整数	20	いいえ
+ProtocolListeners	これは階層プロパティであり、値はありません。		

表 30. コネクタ固有の構成プロパティ (続き)

名前	指定可能な値	デフォルト値	必須
++Listenerl	固有の名前が指定されたプロトコル・リスナー		はい
+++Protocol	http または https		はい
+++ListenerSpecific	リスナーで固有または必要なプロパティ。73 ページの『ListenerSpecific』を参照してください。		
ProxyServer	これは階層プロパティであり、値はありません。		いいえ
+HttpProxyHost	HTTP プロキシ・サーバーのホスト名		いいえ
+HttpProxyPort	HTTP プロキシ・サーバーのポート番号	80	いいえ
+HttpNonProxyHosts	直接接続が必要な HTTP ホスト		いいえ
+HttpsProxyHost	HTTPS プロキシ・サーバーのホスト名		いいえ
+HttpsProxyPort	HTTPS プロキシ・サーバーのポート番号	443	いいえ
+HttpsNonProxyHosts	直接接続が必要な HTTPS ホスト		いいえ
+SocksProxyHost	Socks プロキシ・サーバーの名前		いいえ
+SocksProxyPort	Socks プロキシ・サーバーのポート		いいえ
+HttpProxyUsername	HTTP プロキシ・サーバー・ユーザー名		いいえ
+HttpProxyPassword	HTTP プロキシ・サーバー・パスワード		いいえ
+HttpsProxyUsername	HTTPS プロキシ・サーバー・ユーザー名		いいえ
+HttpsProxyPassword	HTTPS プロキシ・サーバー・パスワード		いいえ
SSL	これは階層プロパティであり、値はありません。		いいえ
+SSLVersion	SSL、SSLv2、SSLv3、TLS、TLSv1	SSL	いいえ
+SSLDebug	true、false	false	いいえ
+KeyStoreType	任意の有効な鍵ストア・タイプ	JKS	いいえ
+KeyStore	KeyStore ファイルへのパス		いいえ
+KeyStorePassword	KeyStore における秘密鍵のパスワード		いいえ
+KeyStoreAlias	KeyStore における鍵ペアの別名		いいえ
+TrustStore	TrustStore ファイルへのパス		いいえ
+TrustStorePassword	TrustStore のパスワード		いいえ
+UseClientAuth	true、false	false	いいえ

DataHandlerMetaObjectName: これは、データ・ハンドラーが構成プロパティを設定するために使用する、メタオブジェクトの名前です。

デフォルト値は `MO_DataHandler_Default` です。

JavaProtocolHandlerPackages: このプロパティの値は、Java プロトコル・ハンドラー・パッケージを指定します。コネクタは、このプロパティの値を使用して、システム・プロパティ `java.protocol.handler.pkgs` を設定します。

デフォルトは `com.ibm.net.ssl.internal.www.protocol` です。

ProtocolHandlerFramework: プロトコル・ハンドラー・フレームワークは、そのプロトコル・ハンドラーをロードおよび構成するために、このプロパティを使用します。これは階層プロパティであり、値はありません。

デフォルト = なし。

ProtocolHandlers: この階層プロパティには、値がありません。この第 1 レベルの子は、別個のプロトコル・ハンドラーを表します。

デフォルト = なし。

Handler1: HTTP-HTTPS プロトコル・ハンドラーの名前。これは階層プロパティです。リスナーとは異なり、プロトコル・ハンドラーは重複しない場合もあるので、それぞれのプロトコルごとにハンドラーは 1 つだけしかないことがあります。次の、表 31 は HTTP-HTTPS プロトコル・ハンドラーのサブプロパティを表しています。+ 文字は、プロパティ階層内の項目の位置を示しています。

表 31. HTTP-HTTPS プロトコル・ハンドラー構成プロパティ

名前	指定可能な値	デフォルト値	必須
++HTTPHandler	これは階層プロパティであり、値はありません。		はい
+++Protocol	ハンドラーがインプリメントしようとしているプロトコルの種類。HTTP および HTTPS の場合の値は http です。 注: このプロパティの値を指定しなければ、コネクタはプロトコル・ハンドラーを初期化しません。	http	はい
+++HTTPReadTimeout	リモート・ホストから読み取る際のタイムアウト間隔を (ミリ秒単位で) 指定する HTTP 固有のプロパティ。このプロパティが指定されていないか、または 0 に設定されている場合、HTTP-HTTPS プロトコル・ハンドラーは、リモート・ホストからの読み取り中、無制限に妨害します。	0	いいえ

ProtocolListenerFramework: プロトコル・リスナー・フレームワークは、このプロパティを使用してプロトコル・リスナーをロードします。これは階層プロパティであり、値はありません。

WorkerThreadCount: このプロパティは 1 以上の整数である必要がありますが、このプロパティによって、プロトコル・リスナー・フレームワークで使用できるプロトコル・リスナー・ワーカー・スレッドの数が設定されます。詳細については、54 ページの『プロトコル・リスナー』を参照してください。デフォルト値は 10 です。

RequestPoolSize: このプロパティ (これは、WorkerThreadCount より大きい整数でなければなりません) は、プロトコル・リスナー・フレームワークのリソース

ス・プール・サイズを設定します。このフレームワークは、最大で、WorkerThreadCount 要求と RequestPoolSize 要求の合計数を並行して処理することができます。

デフォルト値は 20 です。

ProtocolListeners: これは階層プロパティーであり、値はありません。このプロパティーの第 1 レベルの子は、それぞれ、別のプロトコル・リスナーを表します。

Listener1: プロトコル・リスナーの名前。複数のプロトコル・リスナーが存在する場合があります。これは階層プロパティーです。このプロパティーのインスタンスを複数作成して、固有の名前を持つ、追加のリスナーを作成することができます。このようにする場合、リスナー固有のプロパティーは変更できますが、プロトコル・プロパティーは変更できません。複数のリスナーの名前は固有でなければなりません。指定可能な名前 (値ではありません) は、HTTPListener1、HTTPSListener1 のいずれかです。

Protocol: このプロパティーは、このリスナーがインプリメントしようとしているプロトコルを指定します。指定可能な値は、http、https のいずれかです。

注: このプロパティーの値を指定しなければ、コネクタはプロトコル・リスナーを初期化しません。

ListenerSpecific: リスナー固有のプロパティーは、指定されたプロトコル・リスナーに固有の、あるいは指定されたプロトコル・リスナーにとって必須のプロパティーです。例えば、HTTP リスナーにはリスナー固有のプロパティー Port が設定されますが、これは、リスナーが要求をモニターするポート番号を表します。表 32 は、HTTP/HTTPS リスナーに固有のプロパティーを要約したものです。+ 文字は、プロパティー階層内の項目の位置を示しています。

表 32. HTTP および HTTPS プロトコル・リスナーに固有の構成プロパティー

名前	指定可能な値	デフォルト値	必須
+++HTTPListener1	HTTP プロトコル・リスナーの固有の名前。これは、ProtocolListenerFramework -> ProtocolListeners 階層プロパティーの子です。リスナーは複数存在する場合があります。このプロパティーおよびその階層の別のインスタンスを作成することにより、追加の HTTP リスナーを付け加えることができます。		はい
++++Protocol	http (HTTP プロトコル・リスナーの場合) https (HTTPS プロトコル・リスナーの場合) 注: このプロパティーの値を指定しなければ、コネクタはプロトコル・リスナーを初期化しません。		はい
++++BOPrefix	このプロパティーの値は、データ・ハンドラーに渡されます。		いいえ
++++Host	リスナーは、このプロパティーの値によって指定された IP アドレスで要求を listen します。Host が指定されない場合、デフォルト値として localhost が使用されます。リスナーを実行しているマシンのホスト名 (DNS 名) または IP アドレスのどちらを指定しても構いません。1 台のマシンで複数の IP アドレスまたは複数の名前を持つこともできます。	localhost	いいえ

表 32. HTTP および HTTPS プロトコル・リスナーに固有の構成プロパティ (続き)

名前	指定可能な値	デフォルト値	必須
++++Port	リスナーが要求を <i>listen</i> するポート。ポートが指定されていない場合、ポートのデフォルトとして、 <i>HTTP</i> の場合は 80、 <i>HTTPS</i> の場合は 443 が使用されます。コネクタ内でリスナーを複製した場合、 <i>Host</i> プロパティと <i>Port</i> プロパティの組み合わせは固有にします。そうしないと、リスナーは、要求を受け入れるためにポートにバインドすることができなくなる可能性があります。	<i>HTTP</i> リスナーの場合は 80 <i>HTTPS</i> リスナーの場合は 443	いいえ
++++SocketQueueLength	着信接続要求のためのキュー (ソケット・キュー) の長さ。ホストに接続を拒否されずに、一度に保管できる着信接続の数を指定します。キューの最大長は、オペレーティング・システムに依存します。	5	いいえ
++++RequestWaitTimeout	要求の到着を待つ際にリスナー・スレッドがホストおよびポートでブロックを行う時間間隔を、ミリ秒単位で指定します。この間隔以内に要求を受け取った場合、リスナーはその要求を処理します。それ以外の場合、リスナー・スレッドは、コネクタ・シャットダウン・フラグが設定されているかどうかを調べます。このフラグが設定されている場合、コネクタは終了します。このフラグが設定されていない場合、 <i>RequestWaitTimeout</i> 間隔の間、ブロックが継続します。このプロパティを 0 に設定した場合、ブロックはいつまでも継続します。このプロパティを指定しないと、デフォルトとして 60000 ミリ秒が使用されます。	60000 (ms)	いいえ
++++HTTPReadTimeout	クライアントからの要求を読み取っているときにリスナーがブロックされる時間間隔を、ミリ秒単位で指定します。このパラメーターを 0 に設定すると、リスナーは、要求メッセージ全体を受け取るまで、いつまでもブロックされます。	0	いいえ
++++HttpAsyncResponseCode	リスナーへの非同期要求に対する <i>HTTP</i> 応答コード。 200 (OK) 202 (ACCEPTED)	202 (ACCEPTED)	いいえ
++++URLsConfiguration	これは階層プロパティであり、値はありません。ここには、このリスナーがサポートする <i>URL</i> の構成が 1 つ以上格納され、さらにオプションで、 <i>MIME</i> タイプと <i>charset</i> の値が格納されます。これは、 <i>ProtocolListenerFramework</i> -> <i>ProtocolListeners</i> -> <i>HTTPListener1</i> 階層プロパティの子プロパティです。このプロパティが指定されない場合、リスナーはデフォルト値を想定します。	<i>ContextPath</i> : / <i>Enabled</i> : true データ・ハンドラー <i>MimeType</i> : 要求の <i>ContentType</i> と同じ <i>Charset</i> : NONE。詳細については、55 ページの『 <i>HTTP</i> および <i>HTTPS</i> プロトコル・リスナー処理』を参照。	いいえ

表 32. HTTP および HTTPS プロトコル・リスナーに固有の構成プロパティー (続き)

名前	指定可能な値	デフォルト値	必須
+++++URL1	これは階層プロパティーであり、値はありません。 この子には、このリスナーがサポートする URL の名前があります。サポートされる URL は、複数存在する場合があります。このプロパティーおよびその階層を複製すれば、追加の URL を付け加えることができます。		いいえ
+++++ContextPath	リスナーが受け取る HTTP 要求の URI。この値は、URLsConfiguration プロパティーの下の ContextPath 値の中で固有でなければなりません。そうでないと、コネクタはログにエラーを記録し、始動に失敗します。ContextPath の値については、大文字小文字の区別が行われます。ただし、大文字小文字が区別されないプロトコル、ホスト名およびポートを含めることができます。プロトコルを ContextPath に指定する場合は、http にする必要があります。ホストを指定する場合には、Host リスナー・プロパティーの値と同じにする必要があります。ポートを指定する場合には、Port リスナー・プロパティーの値と同じにする必要があります。		いいえ
+++++Enabled	このプロパティーの値によって、親 URL 階層プロパティーがコネクタで使用可能かどうかが決まります。	True	いいえ
+++++TransformationRules	これは階層プロパティーであり、値はありません。1 つ以上の変換規則が保持されます。		
+++++++TransformationRule1	これは階層プロパティーであり、値はありません。ここには変換規則が保持されます。		いいえ
+++++++ContentType	このプロパティーの値は、特殊処理 (データ・ハンドラー MIME タイプまたは charset) が適用される着信要求の ContentType を示します。 TransformationRuleN 階層プロパティーによって ContentType が指定されていない場合、コネクタは、ログに警告メッセージを記録して、その TransformationRuleN プロパティーを無視します。このプロパティーに特殊値 */* が指定された場合、プロトコル・リスナーは、この規則を任意の ContentType に適用できます。同じコンテキスト・パスで、同じ ContentType を持つ規則が複数検出された場合、リスナーはログにエラーを記録し、初期化に失敗します。		いいえ
+++++++MimeType	指定された ContentType の要求を処理するためにデータ・ハンドラーを呼び出すときに使用する MIME タイプ。		いいえ
+++++++Charset	指定された ContentType の要求をビジネス・オブジェクトに変換するときに使用する Charset。		いいえ

ProxyServer: ネットワークがプロキシー・サーバーを使用する場合には、このプロパティー以下の値を構成してください。これは階層プロパティーであり、値はありません。このプロパティー以下で指定された値は、HTTP-HTTPS プロトコル・ハンドラーによって使用されます。

図 21 は、Connector Configurator で表示された ProxyServer プロパティを表しています。

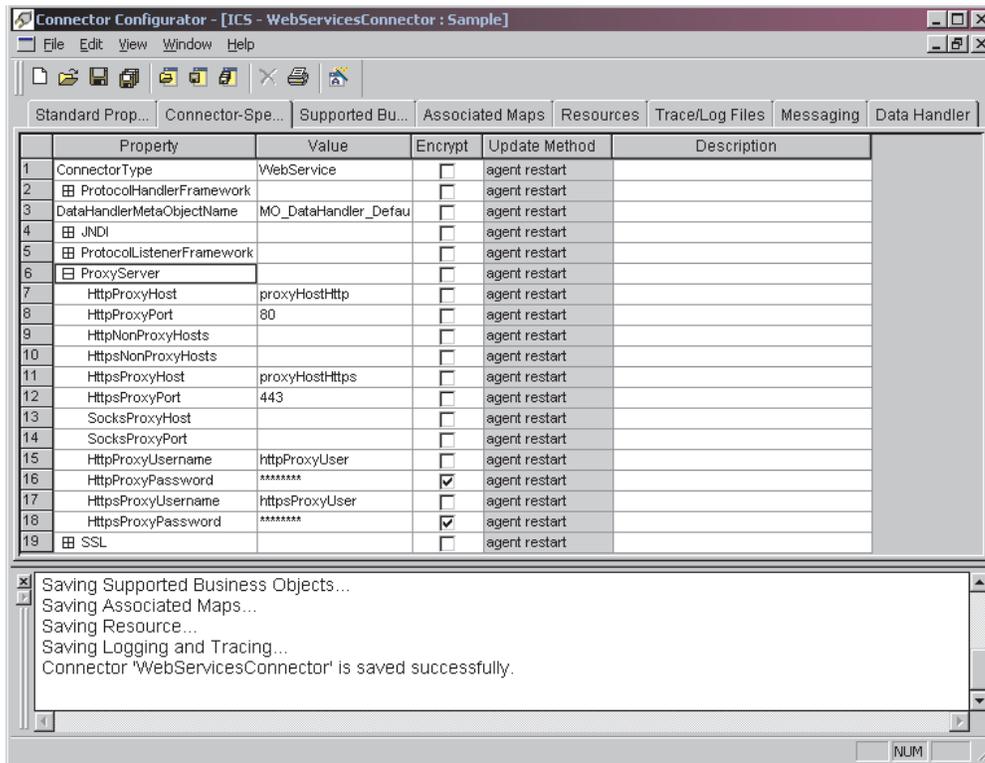


図 21. ProxyServer プロパティ

HttpProxyHost: HTTP プロキシ・サーバーのホスト名。ネットワークが HTTP プロトコルでプロキシ・サーバーを使用する場合には、このプロパティを指定してください。

デフォルト値はありません。

HttpProxyPort: HTTP プロキシ・サーバーに接続するためにコネクタが使用するポート番号。

デフォルト値は 80 です。

HttpNonProxyHosts: このプロパティの値は、プロキシ・サーバー経由ではなく直接接続する必要のある、1 つまたは複数のホスト (HTTP の場合) を表します。この値は、各ホストを "|" で区切った、ホストのリストの形で指定することができます。

デフォルト値はありません。

HttpsProxyHost: HTTPS プロキシ・サーバーの場合のホスト名。

デフォルト値はありません。

HttpsProxyPort: HTTPS プロキシ・サーバーに接続するためにコネクタが使用するポート番号。

デフォルト値は 443 です。

HttpsNonProxyHosts: このプロパティの値は、プロキシ・サーバー経由ではなく直接接続する必要のある、1 つまたは複数のホスト (HTTPS の場合) を表します。この値は、各ホストを "|" で区切った、ホストのリストの形で指定することができます。

デフォルト値はありません。

SocksProxyHost: Socks Proxy サーバーの場合のホスト名。ネットワークが Socks プロキシを使用する場合には、このプロパティを指定してください。

注: 基礎となる JDK は Socks をサポートしていなければなりません。

デフォルト値はありません。

SocksProxyPort: Socks Proxy サーバーに接続するためのポート番号。ネットワークが Socks プロキシを使用する場合には、このプロパティを指定してください。

デフォルト値はありません。

HttpProxyUsername: HTTP プロキシ・サーバーのユーザー名。要求の宛先が HTTP URL であり、ProxyServer -> HttpProxyUsername が指定された場合、HTTP-HTTPS プロトコル・ハンドラーは、このプロキシでの認証のときに Proxy-Authorization ヘッダーを作成します。ハンドラーは、認証に CONNECT メソッドを使用します。

このプロキシ認証ヘッダーは Base64 方式でエンコードされており、以下の構造になっています。

```
Proxy-Authorization: Basic  
Base64EncodedString
```

ハンドラーは、ユーザー名とパスワードのプロパティ値を連結し、その間をコロン (:) で区切って、Base64 エンコードされたストリングを作成します。

デフォルト値はありません。

HttpProxyPassword: HTTP プロキシ・サーバーのパスワード。これらの値の使用に関する詳細については、『HttpProxyUsername』を参照してください。

デフォルト値はありません。

HttpsProxyUsername: HTTPS プロキシ・サーバーのユーザー名。要求の宛先が HTTPS URL であり、ProxyServer -> HttpsProxyUsername が指定された場合、HTTP-HTTPS プロトコル・ハンドラーは、このプロキシでの認証に対して Proxy-Authorization ヘッダーを作成します。ハンドラーは、HttpsProxyUsername と HttpsProxyPassword 構成プロパティ値を連結し、その間をコロン (:) で区切って、Base64 エンコードされたストリングを作成します。

デフォルト値はありません。

HttpsProxyPassword: HTTPS プロキシ・サーバーのパスワード。これらの値の使用に関する詳細については、77 ページの『HttpsProxyUsername』を参照してください。

デフォルト値はありません。

SSL: コネクタのための SSL を構成するためには、このプロパティ以下の値を指定してください。これは階層プロパティであり、値はありません。

SSLVersion: コネクタによって使用される SSL のバージョン。詳しくは、IBM JSSE 資料に記載された、サポートされる SSL バージョンに関する説明を参照してください。

デフォルト値は SSL です。

SSLDebug: このプロパティの値が true に設定されている場合、コネクタは `javax.net.debug` システム・プロパティの値を true に設定します。IBM JSSE はこのプロパティを使用して、トレース機能をオンにします。詳しくは IBM JSSE の資料を参照してください。

デフォルト値は false です。

KeyStoreType: このプロパティの値は、KeyStore および TrustStore のタイプを表します。詳しくは、IBM JSSE 資料に記載された、有効な鍵ストア・タイプに関する説明を参照してください。

デフォルト値は JKS です。

KeyStore: このプロパティでは、鍵ストア・ファイルへの完全パスを指定します。KeyStore または KeyStoreAlias プロパティ、あるいはその両方が指定されていない場合には、KeyStorePassword、KeyStoreAlias、TrustStore、および TrustStorePassword の各プロパティは無視されます。コネクタは、このプロパティで指定されたパスを使用して鍵ストアをロードすることができない場合、始動に失敗します。このパスは、鍵ストア・ファイルの完全パスでなければなりません。

デフォルト値はありません。

KeyStorePassword: このプロパティでは、Keystore 内の秘密鍵のパスワードを指定します。

デフォルト値はありません。

KeyStoreAlias: このプロパティでは、KeyStore 内の鍵ペアの別名を指定します。HTTPS リスナーは、KeyStore からこの秘密鍵を入手して使用します。また、HTTP-HTTPS プロトコル・ハンドラーは、クライアント認証を必要とするサービスを呼び出すときに、KeyStore からこの別名を入手して使用します。このプロパティは、有効な JSSE 別名に設定しなければなりません。

デフォルト値はありません。

TrustStore: このプロパティでは、TrustStore への完全パスを指定します。TrustStore は、コネクタに信頼される証明書を保管するために使用されます。

TrustStore のタイプは KeyStore と同じでなければなりません。TrustStore ファイルの完全パスを指定しなければなりません。

デフォルト値はありません。

TrustStorePassword: このプロパティーは、Truststore のパスワードを指定します。

デフォルト値はありません。

UseClientAuth: このプロパティーでは、SSL クライアント認証を使用するかどうかを指定します。このプロパティーが true に設定されている場合、HTTPS リスナーはクライアント認証を使用します。

デフォルト値は false です。

複数のプロトコル・リスナーの作成

プロトコル・リスナーのインスタンスを複数作成することができます。プロトコル・リスナーは、ProtocolListenerFramework> ProtocolListeners コネクター・プロパティーの子プロパティーとして構成されます。(ProtocolListenerFramework -> ProtocolListeners の) それぞれの子プロパティーは、コネクターのプロトコル・リスナーを明確に識別します。したがって、ProtocolListeners プロパティーの下で、新しい子プロパティーを構成することにより、追加のプロトコル・リスナーを作成することができます。新規に作成したリスナー・プロパティーの子プロパティーをすべて指定していることを、確認してください。各リスナーの名前は固有でなければなりません。ただし、リスナーのプロトコル・プロパティー (http または https) は変更しません。これらのプロパティーは、リスナーの複数のインスタンスに対して同じ名前のままとなります。

注: このプロトコル・プロパティーはスイッチの役目をしているため非常に重要です。リスナーまたはハンドラーを使用したくない場合は、このプロパティーを空のままにします。

HTTP リスナーまたは HTTPS リスナーの複数インスタンスを作成している場合は、それぞれのインスタンスに別々のポートおよびホスト・プロパティーを必ず指定してください。

1 つのハンドラーの複数のインスタンスは作成できません。プロトコルごとにハンドラーは 1 つしか存在できません。

始動時のコネクター

コネクターを始動すると、init() メソッドは、System Manager の Connector Configurator を使用して設定された構成プロパティーを読み取ります。正常に機能させるために、絶対にコネクターのポーリングを使用不可にしないでください (コネクターのポーリングはデフォルトで使用可能になっています)。この際にどのようなことが行われるのかについて、以下のセクションで説明します。

プロキシのセットアップ

コネクタ固有の ProxyServer プロパティを指定すると、コネクタはプロキシ・システム・プロパティをセットアップします。プロキシ・サーバーは、要求処理の場合のみ、HTTP-HTTPS プロトコル・ハンドラーとともに使用されます。コネクタはまた、セットアップしたそれぞれのシステム・プロパティをトレースします。ProxyServer プロパティの詳細については、70 ページの『コネクタ固有の構成プロパティ』を参照してください。

プロトコル・リスナー・フレームワークの初期化

コネクタは、始動時にプロトコル・リスナー・フレームワークのインスタンスを作成し、それを初期化します。このフレームワークがコネクタ固有の ProtocolListenerFramework プロパティを読み取り、さらにコネクタがコネクタ・プロパティ WorkerThreads および RequestPoolSize の値を読み取ります。ProtocolListenerFramework プロパティが指定されていないかあるいは欠落している場合、コネクタはクライアントから要求を受け取ることができず、警告をログに記録します。

コネクタは、次に ProtocolListenerFramework -> ProtocolListeners プロパティを読み取ります。ProtocolListeners プロパティのすべての第 1 レベル・プロパティは、プロトコル・リスナーを表します。プロトコル・リスナー・フレームワークは、それぞれのリスナーのロードと初期化を試み、それらをトレースします。永続的イベントに対応している場合には、リスナーはイベント・リカバリーを試みます。

プロトコル・ハンドラー・フレームワークの初期化

コネクタは、コネクタ固有の ProtocolHandlerFramework プロパティを読み取り、プロトコル・ハンドラー・フレームワークのインスタンスを作成してそれを初期化します。このプロパティが欠落しているか、あるいは正しく設定されていない場合、コネクタは要求を処理することができず、警告をログに記録します。次に、コネクタはすべての ProtocolHandlerFramework -> ProtocolHandlers プロパティ（これは、プロトコル・ハンドラーに対応します）を読み取り、それらのプロパティのロード、初期化、およびトレースを試みます。プロトコル・ハンドラーはコネクタの初期化中にロードされ、コラボレーションがサービス要求を行うときにはインスタンス化されません。プロトコル・ハンドラーはマルチスレッドに対応しています。

ロギング

コネクタは、以下の場合にログに警告を記録します。

- ProtocolListenerFramework プロパティが指定されていない場合。コネクタは、イベント通知を行えないことを警告します。
- ProtocolHandlerFramework プロパティが指定されていない場合。コネクタは、(コラボレーション) 要求処理を行えないことを警告します。

トレース

トレースは、コネクタの振る舞いを綿密にたどるために使用することのできる、オプションのデバッグ機能です。デフォルトでは、トレース・メッセージは `STDOUT` に書き込まれるようになっています。トレース・メッセージの構成に関する詳細については、コネクタ構成プロパティを参照してください。トレースを使用可能にして設定する方法などのトレースの詳細については、「コネクタ開発ガイド (Java 用)」を参照してください。

コネクタのトレース・レベルは次のとおりです。

- レベル 0 このレベルは、コネクタのバージョンを識別するトレース・メッセージに使用されます。
- レベル 1 `pollForEvents` メソッドが呼び出されるたびにトレースを行います。ICS へのデリバリーのためにリスナーによって作成された TLO 名をトレースします。要求ビジネス・オブジェクト名、およびそれに対応する TLO 内の属性名をトレースします。
- レベル 2 このレベルは、`gotAppEvent()` または `executeCollaboration()` から `InterChange Server` にビジネス・オブジェクトがポストされるたびにトレース・メッセージをログに記録する場合に使用してください。また、どのプロトコル・ハンドラーが要求を処理しているのかもトレースします。
- レベル 3 処理中のビジネス・オブジェクトの ASI をトレースします。処理中のビジネス・オブジェクトの属性をトレースします。イベント通知中に要求ビジネス・オブジェクトの TLO をトレースします。データ・ハンドラーによって戻されたビジネス・オブジェクトをトレースします。
- レベル 4 以下のものと関連したトランスポート・ヘッダーをトレースします。
- プロトコル・リスナーによってトランスポートから検索された要求メッセージ
 - プロトコル・リスナーによってクライアントに送信された応答メッセージ
- スレッドの作成、処理されるすべての ASI、および重要な機能を持つすべての入り口および出口をトレースします。
- レベル 5 以下のものをトレースします。
- それぞれの重要なメソッドごとの入り口および出口
 - 構成に固有のすべてのプロパティ
 - それぞれのプロトコル・リスナーのロード
 - プロトコル・リスナーによってトランスポートから検索された要求メッセージ
 - プロトコル・リスナーによってトランスポートでクライアントに送信された応答メッセージ
 - それぞれのプロトコル・ハンドラーのロード
 - データ・ハンドラーによって戻されたメッセージ

- コラボレーションに送られた TLO のビジネス・オブジェクト・
ダンプ
- データ・ハンドラーによって戻されたビジネス・オブジェクトの
ダンプ

第 5 章 SOAP データ・ハンドラー

- 『SOAP データ・ハンドラーの構成』
- 91 ページの『SOAP データ・ハンドラーの処理』
- 121 ページの『SOAP の Style および Use に関するガイドライン』
- 123 ページの『XML の制約事項』

SOAP データ・ハンドラーは、ビジネス・オブジェクトから SOAP メッセージへ、および SOAP メッセージからビジネス・オブジェクトへの変換を主な役割とするデータ変換モジュールです。SOAP データ・ハンドラーは以下の機能を実行します。

- 要求処理
 - SOAP 要求ビジネス・オブジェクトから SOAP 要求メッセージへ
 - SOAP 応答メッセージから SOAP 応答ビジネス・オブジェクトへ
 - SOAP 障害メッセージから SOAP 障害ビジネス・オブジェクトへ
- イベント処理
 - SOAP 要求メッセージから SOAP 要求ビジネス・オブジェクトへ
 - SOAP 応答ビジネス・オブジェクトから SOAP 応答メッセージへ
 - SOAP 障害ビジネス・オブジェクトから SOAP 障害メッセージへ

この章では、SOAP データ・ハンドラーの構成方法、SOAP データ・ハンドラーによるメッセージおよびオブジェクトの処理方法、およびデータ・ハンドラーのカスタマイズ方法について説明します。

SOAP データ・ハンドラーの構成

SOAP データ・ハンドラーは、Connector for SAP XI の中核コンポーネントです。コネクタは、ビジネス・オブジェクトを SAP XI に対応する SOAP メッセージに変換するため、SOAP データ・ハンドラーを呼び出します。

コラボレーションが SAP XI サービスとして公開されている場合には、コネクタも SOAP データ・ハンドラーを呼び出します。次にデータ・ハンドラーは、リモート取引先 (または内部クライアント) から送信された SOAP メッセージをビジネス・オブジェクトに変換します。コネクタは、変換されたビジネス・オブジェクトを、SAP XI 用に構成されたコラボレーションに渡します。

上記の変換の中で、データ・ハンドラー・メタオブジェクトに格納された情報が重要な役割を果たします。この情報の構成は、製品ファイルのインストール後、ただし始動前に実行してください。カスタム名ハンドラーを追加しないのであれば、デフォルトの SOAP データ・ハンドラー構成を使用して時間を節約することができます。ただし、それぞれのデータ・ハンドラーを変換するたびに特定のメタオブジェクト情報を構成しなければなりません。データ・ハンドラーのメタオブジェクトについて、以下のセクションで説明します。

メタオブジェクトの要件

メタオブジェクトは、構成情報を格納しているビジネス・オブジェクトです。コネクタは、実行時にメタオブジェクトを使用してデータ・ハンドラーを構成し、そのインスタンスを作成します。SOAP データ・ハンドラーもメタオブジェクトを使用することにより、SOAP メッセージの本文を見つけ、本文に対応するビジネス・オブジェクトおよび動詞を決定し、SOAP メッセージ内のビジネス・オブジェクトをエンコードし、またこの章で説明する他のいくつかのタスクを実行します。ここでは、これらのメタオブジェクトの要件について説明します。

メタオブジェクトの階層および用語

図 19 は、SAP XI サービス製品用アダプターに対応するメタオブジェクト構造を示しています。メタオブジェクトは図の中で太字で示され、図の後で説明されています。

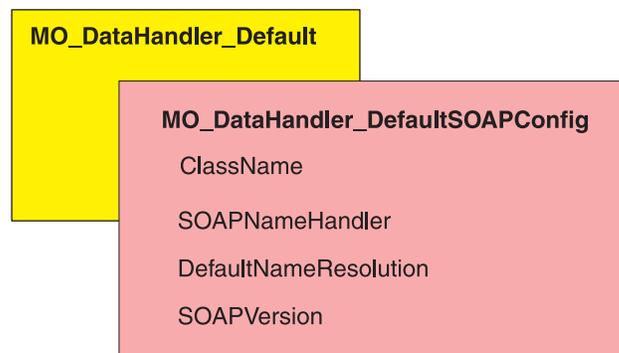


図 22. メタオブジェクトの構造

本書では、メタオブジェクトについて述べる時次の用語を使用します。

- **MO_DataHandler_Default**: コネクタ・エージェントが、どのデータ・ハンドラーのインスタンスを生成するかを決定するために使用するデータ・ハンドラー・メタオブジェクトです。これは、コネクタの `DataHandlerMetaObjectName` プロパティで指定されます。
- **MO_DataHandler_DefaultSOAPConfig**: SOAP データ・ハンドラー固有の子データ・ハンドラー・メタオブジェクトです。
- **SOAP 構成メタオブジェクト (SOAP 構成 MO)**: それぞれの SOAP ビジネス・オブジェクトの子として指定されるメタオブジェクトであり、ビジネス・オブジェクトから SOAP メッセージへの、またその逆方向への単一の変換を行うための構成情報を含みます。

MO_DataHandler_Default

MO_DataHandler_Default は、コネクタから呼び出されるすべてのデータ・ハンドラーに関する、トップレベル・メタオブジェクトです。これらのメタオブジェクトに設定されている MIME 型により、使用されるデータ・ハンドラーが決まります。コネクタ・エージェントは、このメタオブジェクトを使用して、SOAP データ・ハンドラーのインスタンスを作成します。したがって、**MO_DataHandler_Default** オブジェクトには、`xml_soap` という名前でも **MO_DataHandler_DefaultSOAPConfig** という型の属性が設定されている必要があります。

MO_DataHandler_Default オブジェクトは、インストール後に構成できます。
MO_DataHandler_DefaultSOAPConfig タイプの xml_soap を追加する必要があります。

MO_DataHandler_DefaultSOAPConfig

コネクタ・エージェントは、このメタオブジェクトを使用して、実行時に SOAP データ・ハンドラーを作成および構成します。MO_DataHandler_DefaultSOAPConfig には、string 型の属性が 2 つあり、以下を指定します。

- SOAP データ・ハンドラーのクラス名
- SOAP 名ハンドラー
- カスタム名ハンドラーに障害が発生した場合のデフォルトのネーム解決
- SOAP のバージョン (1.1 または 1.2)

これらの属性を、表 33 に示します。

カスタム名ハンドラー (これについては、この章の後半で説明します) をインプリメントしたい場合を除き、MO_DataHandler_DefaultSOAPConfig は、配布およびインストールされたままの状態で使用することができます。構成は不要です。

表 33. MO_DataHandler_DefaultSOAPConfig のメタオブジェクト属性

名前	型	デフォルト値	説明
ClassName	String	com.ibm.adapters .dataHandlers.xml_soap	createHandler メソッドに渡された MIME 型に基づいて、データ・ハンドラー基本クラスがクラス名を見つけるために使用する標準属性。
SOAPName Handler	String		使用する SOAP 名ハンドラーの名前。
DefaultName Resolution	String	false	カスタム名ハンドラーに障害が発生した場合に、デフォルトのネーム解決を使用するかどうかを指定します。
SOAPVersion	String	1.1	データ・ハンドラーが SOAP メッセージの読み取りおよび書き込みに使用する SOAP 規格 (1.1 または 1.2) を指定します。

SOAP 構成メタオブジェクト: 各 SOAP ビジネス・オブジェクトの子

SOAP 構成 MO は、あるデータ・ハンドラー変換 (SOAP メッセージからビジネス・オブジェクトへの変換、またはビジネス・オブジェクトから SOAP メッセージへの変換) で行われるデータ・フォーマットの振る舞いを定義します。SOAP 構成 MO は SOAP ビジネス・オブジェクトの子です。これらの子 SOAP 構成 MO は、デフォルトのビジネス・オブジェクトの解決には不可欠なものです。デフォルトのビジネス・オブジェクト解決を使用する場合、すべての子 SOAP 構成 MO は、要求、応答、または障害オブジェクトのいずれの場合にも、BodyName および BodyNS のデフォルト値に関する固有の項目を含んでいなければなりません。表 34 には、これらの属性および SOAP 構成 MO のその他の属性が示されています。

表 34. SOAP 構成 MO の属性

名前	必須	説明
BodyNS	はい	SOAP 本文に使用されるネーム・スペース。
BodyName	はい	SOAP メッセージの本文の名前。SOAP 障害に対して、デフォルト値を <code>soap:fault</code> に設定します。
BOVerb	はい	SOAP 構成 MO を含むビジネス・オブジェクトの動詞。
TypeInfo	いいえ	SOAP 要素で型情報 (<code>xsi:type</code>) の書き込みおよび読み取りが行われるかどうかを示す、 <code>true</code> または <code>false</code> 属性。デフォルト値は <code>false</code> です。
TypeCheck	いいえ	このプロパティは、TypeInfo が <code>true</code> に設定されている場合にのみ読み取られます。指定可能な値は <code>none</code> および <code>strict</code> です。 <code>none</code> を指定した場合、このビジネス・オブジェクトに SOAP メッセージを読み込むときに型の検証が省略されます。 <code>strict</code> を指定した場合、データ・ハンドラーは、すべての SOAP 型名およびネーム・スペースを、ビジネス・オブジェクトのアプリケーション固有情報と対比して、厳密に検証します。デフォルト値は <code>none</code> です。
Style	いいえ	このプロパティは SOAP メッセージ・スタイルを指定するものであり、BodyName や BodyNS などの他の属性と密接に関連しています。この属性として指定可能な値は、 <code>rpc</code> および <code>document</code> です。デフォルト値は <code>rpc</code> です。
Use	いいえ	このプロパティは SOAP メッセージの使用を指定するものであり、ビジネス・オブジェクトから SOAP 本文を構成する方法に影響を及ぼします。指定可能な値は <code>literal</code> および <code>encoded</code> です。デフォルト値は <code>literal</code> です。

図 23 は、SOAP ビジネス・オブジェクトと SOAP 構成 MO の関係を示しています。

SOAP ビジネス・オブジェクト

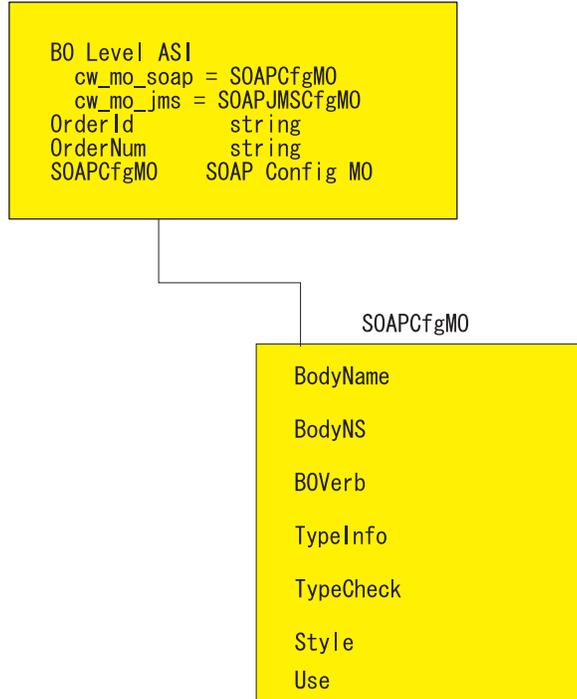


図 23. SOAP 構成のメタオブジェクト

図 23 は、SOAP 応答ビジネス・オブジェクトとその子ビジネス・オブジェクトを示したものです。子ビジネス・オブジェクト SOAPCfgMO は、ビジネス・オブジェクト応答から SOAP 応答メッセージへの変換を行う際の SOAP データ・ハンドラーの振る舞いを指定する、SOAP 構成 MO です。子 SOAP 構成 MO を指定する属性では、cw_mo_soap で始まる名前と値のペアを使用しなければなりません。

規約により、データ・ハンドラーは、cw_mo_ で始まるビジネス・オブジェクト・レベルのアプリケーション固有情報を読み取る際には、名前と値のペアで指定された子オブジェクトには変換メタオブジェクト情報が含まれているものと認識するため、変換するメッセージの本文の内容としてこの子を含みません。この例で、名前と値のペア cw_mo_jms と cw_mo_soap で指定されている子オブジェクトは、メタオブジェクトとして認識され、SOAP 応答メッセージには書き込まれません。また、SOAP データ・ハンドラーは、cw_mo_soap を除き、cw_mo_ で始まるすべてのビジネス・オブジェクト・レベルのアプリケーション固有の情報を無視します。したがって、SOAP データ・ハンドラーは、cw_mo_tpi のようなアプリケーション固有の情報は無視します。ただし、SOAP データ・ハンドラーは、ビジネス・オブジェクトから SOAP メッセージへの SOAP 応答変換を実行するために、cw_mo_soap で指定された SOAP 構成 MO については、読み取って使用します。

すべての SOAP ビジネス・オブジェクトには子 SOAP 構成 MO が必要であり、これらの SOAP 構成 MO は、ビジネス・オブジェクト・レベルでアプリケーション固有情報として指定されていなければなりません。この作業の大半は、自動的に行われます。SOAP メッセージ用のビジネス・オブジェクトを生成するために SAP XI ODA を使用すると、SOAP 構成 MO が自動的に生成されます。

SOAP メッセージに対する Style と Use の影響

SOAP 構成 MO オプション・プロパティ Style および Use は、SOAP メッセージの作成方法に影響を及ぼします。Style に指定可能な値は `rpc` および `document` です。Use に指定可能な値は `literal` および `encoded` です。以下のセクションでは、Style および Use の組み合わせが SOAP メッセージの作成に及ぼす影響について説明します。

rpc/literal: Style プロパティを `rpc` に設定し、Use プロパティを `literal` に設定した場合、SOAP メッセージの Body Name および Body Namespace は、それぞれ SOAP 構成 MO の BodyName プロパティおよび BodyNS プロパティから読み取られます。

Body Name および Body Namespace がそれぞれ `getOrderStatus` および `OrderStatusNS` に解決された `rpc/literal` スタイル・メッセージの例を以下に示します。

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:getOrderStatus xmlns:ns1="http://www.ibm.com/">
      <Part1>
        <ns2:Elem1 xmlns:ns2="http://www.ibm.com/element">
          <Child1>1</Child1>
          <Child2>2</Child2>
        </ns2:Elem1>
        <ns3:Elem1 xmlns:ns3="http://www.ibm.com/element">
          <Child1>3</Child1>
          <Child2>4</Child2>
        </ns3:Elem1>
        <Elem2>10</Elem2>
      </Part1>
    </ns1:getOrderStatus>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 24 は、この `rpc/literal` メッセージに対応するビジネス・オブジェクトを示しています。

Name	Type	Key	Card	Default	App Spec Info
☐ Part1	SOAP_Part1Type	<input checked="" type="checkbox"/>	1		
☐ Elem1	SOAP_MaxType	<input checked="" type="checkbox"/>	N		maxoccurs=5;elem_ns=http://www.ibm.com/elem1
Child1	String	<input checked="" type="checkbox"/>			
Child2	String	<input type="checkbox"/>			
ObjectEventId	String				
Elem2	String	<input type="checkbox"/>			
ObjectEventId	String				
☐ SOAPConfigMO	SOAP_Req_Cfg_MO	<input type="checkbox"/>	1		
BodyName	String	<input checked="" type="checkbox"/>		getOrderStatus	
BodyNS	String	<input type="checkbox"/>		http://www.ibm.com	
BOVerb	String	<input type="checkbox"/>		Retrieve	
TypeInfo	String	<input type="checkbox"/>		false	
TypeCheck	String	<input type="checkbox"/>		none	
Style	String	<input type="checkbox"/>		rpc	
Use	String	<input type="checkbox"/>		literal	

図 24. rpc/literal SOAP 構成 MO

注: 対応する SOAP メッセージが作成されるように、これらのプロパティおよびビジネス・オブジェクト属性を適切に構成してください。

rpc/encoded: Style プロパティを rpc に設定し、Use プロパティを encoded に設定した場合、SOAP メッセージの Body Name および Body Namespace は、それぞれ子構成 MO の BodyName プロパティおよび BodyNS プロパティから読み取られます。また、

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" 属性も Body タグに追加されます。

Body Name および Body Namespace がそれぞれ getOrderStatus および OrderStatusNS に解決された rpc/encoded メッセージの例を以下に示します。

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
    <ns1:getOrderStatus xmlns:ns1="http://www.ibm.com/">
      <Part1 xsi:type="ns1:SOAP_Part1Type">
        <ns2:Elem1 SOAP-ENC:arrayType="ns2:SOAP_MaxType[2]"
          xsi:type="SOAP-ENC:Array" xmlns:ns2="http://www.ibm.com/elem1">
          <item>
            <Child1 xsi:type="xsd:string">1</Child1>
          <Child2 xsi:type="xsd:string">2</Child2>
          </item>
          <item>
            <Child1 xsi:type="xsd:string">3</Child1>
            <Child2 xsi:type="xsd:string">4</Child2>
          </item>
        </ns2:Elem1>
        <Elem2 xsi:type="xsd:string">10</Elem2>
      </Part1>
    </ns1:getOrderStatus>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 25 に、この rpc/encoded メッセージに対応するビジネス・オブジェクトを示します。

Name	Type	Key	Card	Default	App Spec Info
☐ Part1	SOAP_Part1Type	<input checked="" type="checkbox"/>	1		
☐ Elem1	SOAP_MaxType	<input checked="" type="checkbox"/>	N		elem_ns=http://www.ibm.com/elem1
Child1	String	<input checked="" type="checkbox"/>			
Child2	String	<input type="checkbox"/>			
ObjectEventId	String				
Elem2	String	<input type="checkbox"/>			
ObjectEventId	String				
☐ SOAPConfigMO	SOAP_Req_Cfg_MO	<input type="checkbox"/>	1		
BodyName	String	<input checked="" type="checkbox"/>		getOrderStatus	
BodyNS	String	<input type="checkbox"/>		http://www.ibm.com	
BOVerb	String	<input type="checkbox"/>		Retrieve	
TypeInfo	String	<input type="checkbox"/>		true	
TypeCheck	String	<input type="checkbox"/>		none	
Style	String	<input type="checkbox"/>		rpc	
Use	String	<input type="checkbox"/>		encoded	

図 25. rpc/encoded SOAP 構成 MO

document/literal: Style プロパティを document に設定し、Use プロパティを literal に設定した場合、Body Name タグを囲むものは一切存在しません。次に示すのは、上のビジネス・オブジェクトに基づいた、document スタイルの SOAP メッセージの例です。

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:Elem1 xmlns:ns1="http://www.ibm.com/elem1">
      <Child1>1</Child1>
      <Child2>2</Child2>
    </ns1:Elem1>
    <ns2:Elem1 xmlns:ns2="http://www.ibm.com/elem1">
      <Child1>3</Child1>
      <Child2>4</Child2>
    </ns2:Elem1>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 26 に、この document/literal メッセージに対応するビジネス・オブジェクトを示します。

Name	Type	Key	Card	Default	App Spec Info
☐ Elem1	SOAP_Elem1	<input checked="" type="checkbox"/>	1		maxoccurs=3;elem_ns=http://www.ibm.com/Elem1
Child1	String	<input checked="" type="checkbox"/>			
Child2	String	<input type="checkbox"/>			
ObjectEventId	String				
☐ SOAPConfigMO	SOAP_Req_Cfg_MO	<input type="checkbox"/>	1		
BodyName	String	<input checked="" type="checkbox"/>		getOrderStatus	
BodyNS	String	<input type="checkbox"/>		http://www.ibm.com	
BOVerb	String	<input type="checkbox"/>		Retrieve	
TypeInfo	String	<input type="checkbox"/>		false	
TypeCheck	String	<input type="checkbox"/>		none	
Style	String	<input type="checkbox"/>		document	
Use	String	<input type="checkbox"/>		literal	

図 26. document/literal SOAP 構成 MO

この XML コード・フラグメントでは encodingStyle 属性が設定されていません。

document/encoded: この Style/Use の組み合わせはサポートされません。データ・ハンドラーは、Style を document に設定し、Use を encoded に設定した SOAP 構成 MO を検出すると失敗します。

SOAP データ・ハンドラーの処理

SOAP データ・ハンドラーは、SOAP メッセージとビジネス・オブジェクトの間の変換を次の方法で実行します。

• SOAP メッセージからビジネス・オブジェクトへの処理

- 要求メッセージから SOAP 要求ビジネス・オブジェクトへのデータ処理は、イベント処理の際に、SAP XI として公開されているコラボレーションを SAP XI クライアントが呼び出すときに行われます。
- 応答メッセージから SOAP 応答ビジネス・オブジェクトへのデータ処理は、要求処理の際に、SAP XI サービスを呼び出したコラボレーションに、SAP XI サービスが SOAP 応答メッセージを戻すときに行われます。あるいは、この段階で障害メッセージから SOAP ビジネス・オブジェクトへのデータ処理が行われる可能性もあります。

この処理について詳しくは、このセクションで後述される 92 ページの『SOAP 本文メッセージからビジネス・オブジェクトへの処理』を参照してください。

• ビジネス・オブジェクトから SOAP メッセージへの処理

- ビジネス・オブジェクトから SOAP 応答メッセージへのデータ処理は、イベント処理の際に、SAP XI サービスとして公開されているコラボレーションによって応答ビジネス・オブジェクトが戻されたときに行われます。あるいは、この段階で障害ビジネス・オブジェクトから SOAP 障害メッセージへのデータ処理が行われる可能性もあります。
- ビジネス・オブジェクトから SOAP 要求メッセージへのデータ処理は、要求処理の際に、ビジネス・オブジェクトを SOAP 要求メッセージに変換するために、コラボレーションがコネクタに対してサービス呼び出しを行ったときに行われます。

この処理について詳しくは、このセクションで後述される 94 ページの『ビジネス・オブジェクトから SOAP メッセージ本文への処理』を参照してください。

SOAP 本文メッセージからビジネス・オブジェクトへの処理

このセクションでは、SOAP 本文メッセージからビジネス・オブジェクトへの変換について、段階を追って説明します。

1. SOAP データ・ハンドラーが SOAP メッセージを受け取ります。
2. データ・ハンドラーは、Apache SOAP API を使用して、SOAP メッセージを解析します。
3. データ・ハンドラーは、SOAP メッセージのコンポーネントであるエンベロープ、ヘッダー、および本文を抽出します。
4. **ヘッダーの処理** 詳しくは、93 ページの『SOAP ヘッダー・メッセージからビジネス・オブジェクトへの処理』を参照してください。
5. **本文の処理** データ・ハンドラーは、SOAP 本文が障害に関するものか、データの通知かを確認するため、SOAP 本文の最初の要素を読み取ります。本文の内容が障害でない場合には、データ・ハンドラーは次の処理を実行します。
 - a. どのビジネス・オブジェクトが変換に使用されるのかを判別するために、ビジネス・オブジェクト解決を実行します。カスタム名前ハンドラーが構成されている場合には、下記のデフォルトのビジネス・オブジェクト解決は適用されません。プラグ可能な名前ハンドラーの指定については、119 ページの『プラグ可能な名前ハンドラーの指定』を参照してください。
 - b. データ・ハンドラーは、変換に使用される SOAP 構成 MO (データ・ハンドラーが作成している SOAP ビジネス・オブジェクトの子) も解決します。SOAP 構成 MO のインスタンスが存在しない場合、データ・ハンドラーはインスタンスを作成し、そのデフォルト値を読み取ります。データ・ハンドラーは、ConfigMO の属性値からビジネス・オブジェクトの動詞を読み取ります。データ・ハンドラーは SOAP ビジネス・オブジェクトのインスタンスを生成し、該当の動詞を設定します。このビジネス・オブジェクトは、データ・ハンドラーが SOAP メッセージの書き込みを試みる対象となります。
 - c. データ・ハンドラーは、一度に 1 要素ずつ SOAP メッセージの構文解析を続行します。rpc の場合、データ・ハンドラーは、先頭の要素を親と判断しません。
 - d. データ・ハンドラーは、ビジネス・オブジェクト (またはそのアプリケーション固有情報、詳しくは 100 ページの『ビジネス・オブジェクトから SOAP メッセージへの変換における ASI』を参照) の属性の名前が、子要素と同じであるものと想定します。その属性がビジネス・オブジェクトから見つからない場合、データ・ハンドラーは例外をスローします。子要素は、単純タイプの場合と複合タイプ場合があります。複合要素とは、子要素を持つ要素のことです。
 - e. **単純要素** 子要素が単純要素の場合、デフォルトでは、データ・ハンドラーは、ビジネス・オブジェクト属性が単純要素のビジネス・オブジェクト属性と同じ名前 (または ASI) であることを想定します。データ・ハンドラーは単純要素の値を読み取り、ビジネス・オブジェクトにその値を設定します。
 - f. **複合要素** 子要素が複合タイプの要素である場合、データ・ハンドラーは、ビジネス・オブジェクトの属性の名前 (または ASI) および型が子ビジネス・オ

プロジェクトのものと同じであるものと想定します。この属性は、複合 SOAP 要素が存在するか、それとも SOAP 配列が存在するかによって、単一カーディナリティーの場合と複数カーディナリティーの場合があります。次に、データ・ハンドラーは子ビジネス・オブジェクトのインスタンスを作成し (デフォルトでは、属性の型に応じて子ビジネス・オブジェクトの名前が決まります)、この複合要素のすべての子要素を読み取って、子ビジネス・オブジェクトのそれらの値を設定します。データ・ハンドラーは、親ビジネス・オブジェクト属性のカーディナリティーを検査した後で、この子ビジネス・オブジェクトを親ビジネス・オブジェクト属性に組み込みます。属性がカーディナリティー n の場合、データ・ハンドラーは、このビジネス・オブジェクトをコンテナーに追加します。複合要素の子要素は、単純要素の場合と複合要素の場合とがあります。これらも、同じように扱われます。単純要素である場合には、データ・ハンドラーは値を子ビジネス・オブジェクトで設定し、複合要素である場合には、データ・ハンドラーは子ビジネス・オブジェクトのインスタンスを作成します。

6. **障害の処理** データ・ハンドラーは、SOAP 本文の最初の要素の名前を読み取り、SOAP 本文が障害に関するものかどうかを判別します。最初の要素の名前が `Fault` の場合、データ・ハンドラーはこれが障害メッセージであると認識します。この障害メッセージをどのビジネス・オブジェクトに変換するのかを決定するために、障害ビジネス・オブジェクトの解決が行われます。その後データ・ハンドラーは、本文の処理の場合と同じ処理を行います。データ・ハンドラーは、子ビジネス・オブジェクトで指定されたビジネス・オブジェクトが以下の属性を備えているものと想定します。
 - a. `faultcode`: 必須。String 属性
 - b. `faultstring`: 必須。String 属性
 - c. `faultactor`: 任意。String 属性
 - d. `detail`: 任意。子ビジネス・オブジェクト
7. なんらかの理由で障害処理が失敗した場合、スローされる例外には、SOAP 障害メッセージの `faultcode`、`faultstring`、および `faultactor` 要素から得られたテキストが含まれます。

注: 障害メッセージに関する SOAP 仕様によると、`faultcode`、`faultstring`、および `faultactor` は単純要素であり、`detail` は複合要素 (子要素を含む要素) です。また、`faultcode`、`faultstring`、`faultactor`、および `detail` は SOAP エンベロープ・ネーム・スペースに属しますが、`detail` 子要素はユーザー定義のネーム・スペースに属することもあります。

SOAP ヘッダー・メッセージからビジネス・オブジェクトへの処理

ここでは、データ・ハンドラーによる SOAP メッセージのヘッダーからビジネス・オブジェクトへの変換の方法について説明します。

1. SOAP データ・ハンドラーは SOAP メッセージの本文を処理します。本文の処理により SOAP ビジネス・オブジェクトが作成されます。
2. SOAP メッセージに SOAP ヘッダー要素があると、SOAP データ・ハンドラーは、本文の処理の結果生成されたビジネス・オブジェクト内に SOAP ヘッダー属性が存在すると想定します。SOAPHeader 属性は、ビジネス・オブジェクトの

子属性であり、soap_location=SOAPHeader をそのアプリケーション固有の情報として持っています。このような属性が存在しない場合、SOAP データ・ハンドラーはエラーをスローします。SOAPHeader 属性は、SOAP ヘッダー・コンテナー・ビジネス・オブジェクトの型であることが必要です。SOAP データ・ハンドラーは、ステップ 1 で取得した SOAP ビジネス・オブジェクト内に、この属性のインスタンスを作成します。

3. SOAP-Env:Header 要素の直接の子それぞれに対して、次の処理が実行されます。
 - a. データ・ハンドラーは、SOAP ヘッダー・コンテナー・ビジネス・オブジェクト内に子属性が存在すると想定します。この属性の名前は、ヘッダー要素の名前と同じであること、および SOAP ヘッダー子ビジネス・オブジェクトに準拠していることが必要です。このような属性を見つけない場合、データ・ハンドラーはエラーをスローします。さらに、ネーム・スペースこの要素のネーム・スペースは、この属性の elem_ns アプリケーション固有情報に指定されたネーム・スペースと同じです。同じでない場合、データ・ハンドラーはエラーをスローします。
 - b. データ・ハンドラーは SOAP ヘッダー子ビジネス・オブジェクトのインスタンスを作成し、ステップ 2 で作成された SOAP ヘッダー・コンテナー・ビジネス・オブジェクトのインスタンスにこれを格納します。
 - c. このヘッダー要素に actor 属性がある場合、データ・ハンドラーは、actor 属性が上記で作成された子ビジネス・オブジェクト内に存在すると想定します。actor 属性が見つからない場合、データ・ハンドラーはエラーをスローします。

注: actor 属性を追加する必要がある場合は、104 ページの『SOAP 属性の指定』を参照してください。

- d. このヘッダー要素に mustUnderstand 属性がある場合、データ・ハンドラーは、上記で作成された子ビジネス・オブジェクト内に mustUnderstand 属性が存在すると想定します。mustUnderstand 属性が見つからないと、データ・ハンドラーはエラーをスローします。

注: mustUnderstand 属性を追加する必要がある場合は、104 ページの『SOAP 属性の指定』を参照してください。

- e. このヘッダー要素の各子要素について、データ・ハンドラーは、子ビジネス・オブジェクト内に同じ名前の属性が存在すると想定します。これらの要素は、SOAP-Env:Body 要素の子要素と同じ方法で処理されます。

ビジネス・オブジェクトから SOAP メッセージ本文への処理

ビジネス・オブジェクトから SOAP メッセージ本文への変換は、次のような手順を経て実行されます。アプリケーション固有情報が使用される特殊なケースについては、100 ページの『ビジネス・オブジェクトから SOAP メッセージへの変換における ASI』を参照してください。

1. SOAP データ・ハンドラーは、変換対象の SOAP ビジネス・オブジェクトに対応する SOAP 構成 MO を探します。
2. データ・ハンドラーは、SOAP メッセージのエンベロープとヘッダーを作成します。

3. データ・ハンドラーは SOAP 構成 MO を解決します。SOAP 構成 MO のインスタンスが存在しない場合、データ・ハンドラーはインスタンスを作成し、デフォルト値を読み取ります。デフォルトでは、データ・ハンドラーは、SOAP 構成 MO の BodyName 属性の値を読み取って、処理しているビジネス・オブジェクトが障害ビジネス・オブジェクトであるかどうかを判別します。この値が soap:fault に設定されている場合には、そのビジネス・オブジェクトは SOAP 障害ビジネス・オブジェクトであるとみなされます。それが障害ビジネス・オブジェクトでない場合、データ・ハンドラーは、下記の『本文の作成』で述べられた処理を行い、それ以外の場合には『障害メッセージの作成』で述べられた処理を行います。
4. **本文の作成** データ・ハンドラーがビジネス・オブジェクトから SOAP メッセージの本文を作成するために行う処理について、以下のステップで詳しく説明します。
 - データ・ハンドラーは、SOAP 構成 MO 属性から BodyName および BodyNS を入手してから、SOAP メッセージ本文の最初の (親) 要素を作成します。デフォルトでは、最初の要素の名前は BodyName の値になります。本書では、これを本文要素とも呼びます。本文要素のネーム・スペースは、デフォルトでは BodyNS について決められた値となります。SOAP 構成 MO の Style 属性が document に設定されている場合、このステップ (最初の本文要素の作成) はスキップされます。
 - データ・ハンドラーはビジネス・オブジェクトの属性を読み取り、それら属性を型ごとに処理します。各型の属性の処理について以下で説明します。
 - **単純属性** 属性が単純型の場合、データ・ハンドラーが本文要素から子要素を作成し、名前は属性と同じになります (特別なアプリケーション固有情報で別の値が指定されている場合を除きます)。データ・ハンドラーは、この要素の値を、ビジネス・オブジェクト属性の値に設定します。
 - **カーディナリティー 1 の子ビジネス・オブジェクト属性**

属性が単一カーディナリティーの子ビジネス・オブジェクトの場合、データ・ハンドラーは本文要素の子要素を作成します。これは、子ビジネス・オブジェクト要素と呼ばれます。作成された子要素の名前は、属性の名前と同じになります (ただし、特別な ASI プロパティーで別の名前が指定されている場合を除きます)。次にデータ・ハンドラーは、子ビジネス・オブジェクトの属性をすべて探索し、着信ビジネス・オブジェクトの属性を処理する場合と同じ方法で、それらの属性の子要素を作成します。ただし、それらの子要素は、本文要素の子ではなく、子ビジネス・オブジェクト要素の子になります。
 - **カーディナリティー n の子ビジネス・オブジェクト属性** 属性がカーディナリティー n の子ビジネス・オブジェクトである場合、データ・ハンドラーは SOAP 配列を作成します。それぞれの属性は、単一カーディナリティーの子ビジネス・オブジェクトと同じ方法で処理されます。
5. **障害メッセージの作成** 以下のセクションでは、データ・ハンドラーが障害メッセージを作成するプロセスについて、順に説明します。
 - データ・ハンドラーは、ビジネス・オブジェクト内に以下の属性が存在するものと想定します。
 - faultcode: 必須。String 属性

- faultstring: 必須。String 属性
- faultactor: 任意。String 属性
- detail: 任意。子ビジネス・オブジェクト属性

必須属性が欠落している場合には、データ・ハンドラーはエラー終了します。

- データ・ハンドラーは faultcode の要素を作成します。そして、ビジネス・オブジェクトの faultcode 属性により指定された値を設定します。
 - データ・ハンドラーは faultstring の要素を作成します。そして、ビジネス・オブジェクトの faultstring 属性により指定された値を設定します。
 - データ・ハンドラーは faultactor を作成します。そして、ビジネス・オブジェクトの faultactor 属性により指定された値を設定します。
 - ビジネス・オブジェクトに detail 属性が存在する場合、この属性は、子ビジネス・オブジェクトと同じ型になっている必要があります。そのようになっている場合、データ・ハンドラーはエラー終了します。データ・ハンドラーは各 detail ビジネス・オブジェクトの属性を、上記の『本文の作成』で述べたように処理します。
6. **CxIgnore の処理** データ・ハンドラーは、属性の値が CxIgnore に設定されていることを検出した場合には、この属性に対応する要素を作成しません。
 7. **CxBlank の処理** データ・ハンドラーは、属性の値が CxBlank に設定されていることを判別した場合には、この属性に対応する要素を作成しますが、その値を設定しません。

ビジネス・オブジェクトから SOAP メッセージ・ヘッダーへの処理

このセクションでは SOAP ヘッダー属性の処理についてのみ説明します。他のすべての属性は、94 ページの『ビジネス・オブジェクトから SOAP メッセージ本文への処理』で説明されているとおりに処理されます。

1. SOAP データ・ハンドラーは、ビジネス・オブジェクトから SOAPHeader 属性を取得します。この属性には、そのアプリケーション固有情報として、soap_location=SOAPHeader があります。SOAP データ・ハンドラーは、この属性の値がヌル以外のときに限って、SOAP-Env:Header 要素を作成します。ビジネス・オブジェクトに複数の SOAPHeader 属性が格納されている場合、最初の属性のみが処理され、残りは本文の一部として扱われます。
2. SOAP データ・ハンドラーは、SOAPHeader 属性が、SOAP Header Container ビジネス・オブジェクトを表す単一カーディナリティーの子であると想定します。データ・ハンドラーは、SOAP Header Container ビジネス・オブジェクトの属性の中で、SOAP Header Child ビジネス・オブジェクトの型の属性を処理します。
3. SOAP Header Container ビジネス・オブジェクトの各属性に対して、データ・ハンドラーは次の処理を実行します。
 - a. カーディナリティーをチェックし、この属性がカーディナリティー 1 の子オブジェクトでもカーディナリティー n の子オブジェクトでもなければ、その属性を無視します。
 - b. 値をチェックし、この属性の値が NULL であれば、その属性を無視します。
 - c. 属性がカーディナリティー 1 または n の子オブジェクトの場合、SOAP データ・ハンドラーは、ステップ 1 で作成された SOAP-Env:Header 要素の直

接の子であるヘッダー要素を作成します。このヘッダー要素の名前は属性の名前と同じです。この要素のネーム・スペースは、この属性の `elem_ns` アプリケーション固有情報により指定されます。

- d. 属性が SOAP Header Child ビジネス・オブジェクトであれば、このビジネス・オブジェクトのすべての属性が処理されます。この属性は `actor` および `mustUnderstand` 属性を持つ場合があります。

注: `mustUnderstand` 属性または `actor` 属性を追加する必要がある場合は、104 ページの『SOAP 属性の指定』を参照してください。

- e. SOAP Header Child ビジネス・オブジェクトにヌル以外の `actor` 属性が存在する場合、データ・ハンドラーは、ステップ c で作成されたヘッダー要素内に `actor` 属性を作成します。
- f. SOAP Header Child ビジネス・オブジェクトにヌル以外の `mustUnderstand` 属性が存在する場合、データ・ハンドラーは、ステップ c で作成されたヘッダー要素内に `mustUnderstand` 属性を作成します。
- g. SOAP Header Child ビジネス・オブジェクトのヌル以外のその他すべての属性は、このヘッダー要素の子要素となります。これらの要素は、SOAP-Env:Body 要素の子要素と同じ方法で構成されます。

ヘッダー障害の処理

SOAP の仕様では、ヘッダーに関するエラーはヘッダーで戻さなければならないと規定されています。このようなヘッダーは、SOAP 障害メッセージで戻します。メッセージ・ヘッダーが要求および応答のビジネス・オブジェクトの SOAPHeader 属性で指定されるように、障害ヘッダーは、障害ビジネス・オブジェクトの SOAPHeader 属性で指定されます。

要求ビジネス・オブジェクトまたは応答ビジネス・オブジェクトに付くどのヘッダーも、エラーの発生を招くことがあります。このようなエラーは、障害メッセージのヘッダーに報告されます。

WSDL 文書には SOAP バインディング・ヘッダー障害要素があり、これを使用することにより、障害ヘッダーを指定することができます。詳しくは、第 1 章に表で示した SOAP 仕様および WSDL 仕様を参照してください。

`headerfault` のアプリケーション固有情報により、各ヘッダーの障害を指定することができます。SOAP Header Container ビジネス・オブジェクトの各属性について、`headerfault` アプリケーション固有情報を指定できます。障害ビジネス・オブジェクトに対する SOAP Header Container ビジネス・オブジェクト内の属性リストは次のとおりです。

`headerfault=attr1, attr2, attr3...`

WSDL 構成ウィザードにより、要求または応答オブジェクトの SOAP Header Child ビジネス・オブジェクト内に `headerfault` アプリケーション固有情報が検出されると、このユーティリティーは、これらのヘッダーのために生成された WSDL の中に `headerfault` 要素を作成します。WSDL では、要求 (入力) ヘッダーおよび応答 (出力) ヘッダーのそれぞれに対して、複数のヘッダー障害を指定することができます。したがって、アプリケーション固有情報の値は、コンマ区切りの属性リストです。

アプリケーション固有情報機能の使用

オブジェクト・レベルおよび属性レベルのアプリケーション固有情報 (ASI) を指定して、SOAP データ・ハンドラー機能を拡張および強化することができます。表 35 に、これらの属性を示します。各属性については、以下のセクションで説明します。表の項目はすべて、特に明記されていない限り、属性レベルの ASI です。

表 35. SOAP オブジェクト ASI の要約

ASI	指定可能な値	説明
soap_location	SOAPHeader	このビジネス・オブジェクト属性をヘッダー属性として指定します。
headerfault	String	障害ビジネス・オブジェクト内の対応する SOAP ヘッダーのビジネス・オブジェクト属性名を指定します。
elem_name	String	このビジネス・オブジェクト属性に対応する SOAP 要素の名前を指定します。
elem_ns	String	このビジネス・オブジェクト属性に対応する SOAP 要素のネーム・スペースを指定します。
type_name	String	このビジネス・オブジェクト属性に対応する SOAP 要素の型を指定します。
type_ns	String	このビジネス・オブジェクト属性に対応する要素の型ネーム・スペースを指定します。
xsdtype	true	このビジネス・オブジェクト属性に対応する要素の型ネーム・スペースとして xsd を指定し、古いバージョンの xsd (1999、2000 など) を最新バージョンの xsd (2001 など) でオーバーライドします。
attr_name	String	このビジネス・オブジェクト属性に対応する SOAP 属性の名前を指定します。
attr_ns	String	このビジネス・オブジェクト属性に対応する SOAP 属性のネーム・スペースを指定します。

表 35. SOAP オブジェクト ASI の要約 (続き)

ASI	指定可能な値	説明
arrayof	String	単純型の配列項目のプレースホルダーとして使用する必要がある、カーディナリティーが n の子ビジネス・オブジェクト属性の名前を指定します。
dh_mimetype	String	この複合型の属性を変換するために使用される、データ・ハンドラーの mimeType を指定します。
cw_mo_*	String	このビジネス・オブジェクト・レベルの ASI では、SOAP データ・ハンドラーによって内容ではなくメタデータとして解釈される、子構成 MO の名前を指定します。メタデータとして処理される子構成 MO を指定する値は、cw_mo_soap のみです。それ以外のすべての cw_mo_* は、別のコンポーネントを表すため、SOAP データ・ハンドラーの処理対象から除外されます。他の cw_mo* はすべて無視されます。
cw_mo_soap	String	このビジネス・オブジェクト・レベルの ASI では、このビジネス・オブジェクトを変換するとき使用する必要がある、子構成 MO 属性の名前を指定します。
cw_mo_jms	String	このビジネス・オブジェクト・レベルの ASI では、使用する JMS プロトコル構成 MO の名前を指定します。
cw_mo_http	String	このビジネス・オブジェクト・レベルの ASI では、使用する HTTP プロトコル構成 MO の名前を指定します。
wrapper	true	このビジネス・オブジェクト内のラッパー・オブジェクトの属性名を指定します。ラッパー・オブジェクトは特定のスキーマ標識で使用され、直列化してはなりません。

表 35. SOAP オブジェクト ASI の要約 (続き)

ASI	指定可能な値	説明
maxoccurs	Integer	このビジネス・オブジェクト属性について出現可能な、最大オカレンス件数を指定します。maxoccurs の値によって、ビジネス・オブジェクトにラッパーを含められる場合と、含められない場合があります。
minoccurs	Integer	このビジネス・オブジェクト属性について可能な、最小オカレンス件数を指定します。minoccurs の値によって、ビジネス・オブジェクトにラッパーを含められる場合と、含められない場合があります。
all	String	スキーマの all 標識を表す子属性を指定します。
choice	String	スキーマの choice 標識を表す子属性を指定します。

ビジネス・オブジェクトから SOAP メッセージへの変換における ASI

SOAP データ・ハンドラーは、SOAP メッセージの構成方法を決定するために、ビジネス・オブジェクトの ASI を使用します。別の指示が示されていないかぎり、以下のセクションで述べるすべての ASI は属性レベルの ASI を表し、ストリング・ベースでの比較はすべて、大文字小文字を区別しないで行われます。

elem_name および elem_ns の処理

このセクションで示す例では、属性名が OrderId であって、SOAP 要素ネーム・スペースの接頭部が ns0 であると想定されています。

1. elem_name も elem_ns も指定されていない場合、elem_name のデフォルト値としては属性名が使用され、elem_ns のデフォルト値としては要素の親のネーム・スペースが使用されます。ASI は指定されません。

```
<OrderId>1</OrderId>
```

2. elem_name が指定され、elem_ns が指定されていない場合、elem_name は ASI の elem_name 値に設定され、elem_ns についてはデフォルト値として SOAP 本文のネーム・スペースが使用されます。ASI は以下ようになります。

```
elem_name=CustOrderId  
<CustOrderId>2</CustOrderId>
```

3. elem_ns が指定され、elem_name が指定されていない場合、elem_name についてはデフォルト値として属性名が使用され、elem_ns は ASI の elem_ns 値に設定されます。xmlns 属性が明示的に書き込まれるのは、この要素の範囲内における他のどの個所にも要素ネーム・スペースが見付からない場合のみです。要素ネーム・スペースが検出された場合には、すでに定義されているネーム・スペース

スの接頭部が使用されます。それ以外の場合 (要素ネーム・スペースが見付からない場合)、`elem_ns` のための固有の接頭部が生成されます。次の例では、スコープ内で接頭部がすでに定義されているものと想定しています (`ns1` は、この要素のスコープ内ですでに定義されているネーム・スペースに対応する接頭部を表します)。ASI は以下のようにになります。

```
elem_ns= http://www.w3.org/2001/XMLSchema
<ns1:OrderId>3</ns1:OrderId>
```

次の例では、接頭部が見付からない場合が想定されています (`ns2` は固有の接頭部を表します)。ASI は以下のようにになります。

```
elem_ns=CustOrderIdNamespace
<ns2:OrderId xmlns:ns2="CustOrderIdNamespace">3</ns2:OrderId>
```

4. `elem_name` と `elem_ns` の両方が指定されている場合、`elem_name` および `elem_ns` は、ASI の値に設定されます。すでに定義されているネーム・スペースに関しては、上記 3 の場合と同じ検査が行われます。3 の場合と同様に、ネーム・スペースがまだ定義されていない場合には、`elem_ns` のために固有の接頭部が生成されます。ASI は以下のようにになります。

```
elem_name=CustOrderId;elem_ns=CustOrderIdNamespace
<ns2:CustOrderId xmlns:ns2="CustOrderIdNamespace">1</ns2:OrderId>
```

単純属性の場合の `type_name` および `type_ns` の処理

このセクションの例では、属性名は `OrderId`、SOAP 要素ネーム・スペースの接頭部は `ns0`、そして属性タイプは `String` になっています。

注: `type_name` および `type_ns` の処理は、構成 MO の属性 `TypeInfo` が `true` になっている場合にのみ行われます。

1. `type_name` も `type_ns` も指定されていない場合、`type_name` のデフォルト値として単純型が使用され、`type_ns` のデフォルト値として、xml スキーマで定義されたネーム・スペース (`xsd`) が使用されます。ASI は指定されません。

```
<OrderId xsi:type="xsd:string">1</OrderId>
```

2. `type_name` が指定され、`type_ns` が指定されていない場合、`type_name` は ASI の `type_name` 値に設定され、`type_ns` についてはデフォルト値として要素のネーム・スペースが使用されます。ASI は以下のようにになります。

```
type_name=CustString
<OrderId xsi:type="ns0:CustString">2</OrderId>
```

3. `type_ns` が指定され、`type_name` が指定されていない場合、`type_ns` のデフォルト値として単純型名が使用され、`type_name` は ASI の `type_ns` 値に設定されます。この接頭部は、`elem_ns` の作成に準じた方法で処理されます。要素スコープ内にその型ネーム・スペースがすでに存在している場合を除き、型ネーム・スペースのために固有の接頭部が生成されます。ASI は以下のようにになります。

```
type_ns=CustStringNamespace
<OrderId xmlns:ns2="CustStringNamespace" xsi:type="ns2:String">3</OrderId>
```

4. `type_name` と `type_ns` の両方が指定されている場合、これらは、割り当てられた ASI 値に設定されます。型ネーム・スペースのために固有の接頭部が生成されません。ASI は以下のようにになります。

```
type_name=CustString;type_ns=CustStringNamespace
<OrderId xmlns:ns2="CustStringNamespace" xsi:type="ns2:CustString">1</OrderId>
```

単一カーディナリティー属性の場合の type_name および type_ns の処理

このセクションの例では、属性名は OrderStaus、SOAP 要素ネーム・スペースの接頭部は ns0、そして属性タイプは OrderStatus になっています。

注: type_name および type_ns の処理は、構成 MO の属性 TypeInfo が true になっている場合にのみ行われます。

1. type_name も type_ns も指定されていない場合、type_name のデフォルト値としてビジネス・オブジェクト名が使用され、型ネーム・スペースのデフォルト値として要素のネーム・スペースが使用されます。ASI は指定されません。

```
<OrderStatus xsi:type="ns0:OrderStatus">1</OrderStatus>
```

2. type_name が指定され、type_ns が指定されていない場合、type_name は割り当てられた ASI の値に設定され、type_ns については、デフォルト値として要素のネーム・スペースが使用されます。ASI は以下のようになります。

```
type_name=CustOrderStatus
<OrderStatus xsi:type="ns0:CustOrderStatus">1</OrderStatus>
```

3. type_ns が指定され、type_name が指定されていない場合、type_name のデフォルト値としてビジネス・オブジェクト名が使用され、type_ns は割り当てられた type_ns 値に設定されます。型ネーム・スペースのために固有の接頭部が生成されます。ASI は以下のようになります。

```
type_ns=CustTypeNS
<OrderStatus xsi:type="ns2:SOAP_OrderStatusLine
" xmlns:ns2="CustTypeNS">1</OrderStatus>
```

4. type_name と type_ns の両方が指定されている場合、これらは、割り当てられた ASI 値に設定されます。型ネーム・スペースのために固有の接頭部が生成されます。ASI は以下のようになります。

```
type_name=CustOrderStatus;type_ns=CustTypeNS
<OrderStatus
xsi:type="ns2:CustOrderStatus" xmlns:ns2="CustTypeNS">1</OrderStatus>
```

複数カーディナリティー属性の場合の type_name および type_ns の処理

このセクションで示すすべての例では、属性名が MultiLines であって、SOAP 要素ネーム・スペースの接頭部が ns0 であると想定されています。属性タイプは OrderStatus であるものとします。

注: type_name および type_ns の処理は、構成 MO の属性 TypeInfo が true になっている場合にのみ行われます。

1. type_name も type_ns も指定されていない場合、type_name のデフォルト値としてビジネス・オブジェクト名が使用され、type_ns のデフォルト値として要素のネーム・スペースが使用されます。ASI は以下のようになります。

```
<MultiLines SOAP-ENC:arrayType="ns0:OrderStatus[2]"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="SOAP-ENC:Array">
```

2. type_name が指定され、type_ns が指定されていない場合、type_name は、割り当てられた ASI の type_name 値に設定され、type_ns については、デフォルト値として要素のネーム・スペースが使用されます。ASI は以下のようになります。

```
type_name=CustOrderStatus
<MultiLines SOAP-ENC:arrayType="ns0:CustOrderStatus[2]"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="SOAP-ENC:Array">
```

3. `type_ns` が指定され、`type_name` が指定されていない場合、`type_name` のデフォルト値としてビジネス・オブジェクト名が使用され、`type_ns` は、割り当てられた ASI の `type_ns` 値に設定されます。型ネーム・スペースのために固有の接頭部が生成されます。ASI は以下ようになります。

```
type_ns=CustTypeNS
<MultiLines SOAP-ENC:arrayType="ns2:OrderStatus[2]"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="CustTypeNS" xsi:type="SOAP-ENC:Array">
```

4. `type_name` と `type_ns` の両方が指定されている場合、これらは、割り当てられた ASI 値に設定されます。型ネーム・スペースのために固有の接頭部が生成されず。ASI は以下ようになります。

```
type_name=CustOrderStatus;type_ns=CustTypeNS
<MultiLines SOAP-ENC:arrayType="ns2:CustOrderStatus[2]"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="CustTypeNS" xsi:type="SOAP-ENC:Array">
```

注: 各 Array 要素の親を表す項目要素の型およびネーム・スペースは、`arrayType` と同じです。

単純型、単一カーディナリティー型、および複数カーディナリティー型の場合の `xsdtype`

単純型、単一カーディナリティー型、および複数カーディナリティー型の場合、型名が SOAP メッセージの現行 XSD に従うときには、`xsdtype` ASI 属性を `true` に設定してください。`xsdtype` プロパティーが読み取られるのは、`type_name` と `type_ns` の両方のプロパティーが設定されている場合のみです。`type_name` および `type_ns` が指定された場合、SOAP データ・ハンドラーは、最初に、SOAP API を使用してそれらのペアを Java 型にマップすることを試みます。データ・ハンドラーは、その後で、SOAP メッセージのための現行 XSD を使用して、Java 型を SOAP 要素型に変換することを試みます。例えば、現行 XSD が以下のとおりであって

```
http://www.w3.org/2001/XMLSchema
```

ASI が以下のとおりである場合、

```
type_name=timeInstant;type_ns=http://www.w3.org/1999/XMLSchema;xsdtype=true
```

SOAP メッセージ型名は次のように書かれます。

```
<OrderDate xsi:type="xsd:dateTime">
```

これは、2001 XSD における `dateTime` が、1999 XSD における `timeInstant` と同等であるためです。

`xsdtype` および単純型の配列

複数カーディナリティー・オブジェクトの場合、次のような単純型の配列を作成することができます。

```
<MultiLines SOAP-ENC:arrayType="xsd:string[4]"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="SOAP-ENC:Array">
```

そのためには、type_name プロパティを、希望する単純型 (例えば、string) に設定し、type_ns プロパティを適切な XSD 仕様に設定してください。その後、xsdtype プロパティを true に設定して、この型が現行の XSD 型に変換されるようにしてください。最後に、arrayOf プロパティを、単純型値を保持すべきコンテナ内の属性の名前に設定する必要があります。ストリング配列の場合の ASI の例を次に示します。

```
arrayof=size;type_name=string;type_ns=http://www.w3.org/2001/XMLSchema;xsdtype=true
```

障害処理における ASI の効果

faultcode、faultfactor、faultstring、および detail の各要素は、以下の規則に従います。

1. これらの属性における elem_name、elem_ns、type_name、および type_ns ASI は無視されます。
2. detail 要素のすべての子は、本文処理の説明で述べたとおりに書かれます。

ヘッダー処理における ASI の効果

ヘッダーの子オブジェクトのレベルおよびそれ以下では、すべての ASI プロパティ (表 35 を参照) を使用することができます。

SOAP 属性の指定

単純型の場合の attr_name の処理

simpleContent 拡張子または制約事項のある complexTypes に値と属性の両方が存在する XML スキーマの事例があります。例えば、次の SOAP タグを検討してください。

```
<size system="us">10</size>
```

これは、次のスキーマが基本になっています。

```
<complexType name="SizeType">
  <simpleContent>
    <extension base="int">
      <attribute name="system" type="string"/>
    </extension>
  </simpleContent>
</complexType>
<element name="size" type="ns:SizeType"/>
```

単純型の拡張子または制約事項があり、複合型に対応するビジネス・オブジェクトには、複合型に対応する属性のほかに、追加属性が 1 つ格納されている必要があります。追加の属性には、単純なコンテンツ値 (前述の例では、要素サイズの値である 10) が入っている必要があります。ビジネス・オブジェクト属性は、ビジネス・オブジェクトをその型としてこのような複合型に対応付けるため、ビジネス・オブジェクト属性には、その属性レベルの ASI として、elem_value=simpleContentValue が保持されます。

図 27 は、これに対応するビジネス・オブジェクトを示しています。

Name	Type	Key	Card	Application Specific Information
Request	SOAP_getQuote_N09218329332_Request	<input type="checkbox"/>	1	ws_botype=request
size	SOAP_getQuote_C09218329332_SizeType	<input type="checkbox"/>	1	elem_value=simpleContentValue; type_name=SizeType
simpleContentValue	String	<input checked="" type="checkbox"/>		
system	String	<input type="checkbox"/>		attr_name=system

図 27. 単純型の attr_name ビジネス・オブジェクト

単一カーディナリティー型および複数カーディナリティー型の場合の attr_name の処理

ビジネス・オブジェクト属性を SOAP 要素ではなく SOAP 属性に変換する ASI を指定することができます。データ・ハンドラーは、単一および n カーディナリティーの複合型に対してのみ SOAP 属性の追加をサポートしています。次の例について考えてみましょう。

```
<CustInfo City="4" State="5" Street="2" Zip="6">
  <Name xsi:type="xsd:string">1</Name>
  <Street2 xsi:type="xsd:string">3</Street2>
</CustInfo>
```

このビジネス・オブジェクト定義構造 (図 28 の各属性の右側に属性レベル ASI が指定されています) の場合、データ・ハンドラーは以下の処理ステップに従います。

Name	Type	App Spec Info
CustInfo	CustomerInfo	
Name	String	
Street1	String	attr_name=Street
Street2	String	
City	String	attr_name=City
State	String	attr_name=State
Zip	String	attr_name=Zip

図 28. attr_name ビジネス・オブジェクト

1. データ・ハンドラーは、複合属性全体を探索する際に、まず、この複合ビジネス・オブジェクト属性に対応するタグを生成します。この例では、CustInfo が複合ビジネス・オブジェクト属性を表します。
2. データ・ハンドラーは、複合ビジネス・オブジェクトの複数の子について繰り返し処理を行います。属性を作成する際に考慮されるのは、単純型属性のみです。ある単純型の ASI プロパティの名前が attr_name である場合、データ・ハンドラーはこの単純型を SOAP 要素の属性として記述します。この例では、要素 (CustInfo) には Street、City、State、および Zip の 4 つの属性があります。
3. ビジネス・オブジェクトのそれ以外の属性は、標準的な BODY 処理を使用して記述されます。これはつまり、関係のあるすべての ASI が、attr_name ASI を持たないビジネス・オブジェクト属性に対しても評価されるということです。

複数カーディナリティー型を処理する際のロジックは、単一カーディナリティー型を処理するロジックと同じです。具体的には、それぞれの <item> タグは複数カーディナリティー・オブジェクトにおける各ビジネス・オブジェクト・インスタンスに対応し、ASI を使用して処理されます。例えば、この複数カーディナリティー・ビジネス・オブジェクト定義構造に、対応する ASI があるとします。

Name	Type	Card	App Spec Info
☐ CustInfo	CustomerInfo	N	
Name	String		
Street1	String		attr_name=Street
Street2	String		
City	String		attr_name=City
State	String		attr_name=State
Zip	String		attr_name=Zip

図 29. attr_name 複数カーディナリティー・ビジネス・オブジェクト

データ・ハンドラーに送られたイベントに、この複数カーディナリティー・オブジェクトのインスタンスが 2 つ含まれている場合、次のような SOAP メッセージが作成されます。

```
<CustInfo>
  <item City="Armonk" Street="Main Street">
    <Name>IBM</Name>
    <Street2>None</Street2>
  </item>
  <item City="Burlingame" State="Ca" Street="577 Airport Blvd" Zip="94010">
    <Name>Burlingame Labs</Name>
    <Street2>Suite 600</Street2>
  </item>
</CustInfo>
```

item タグが複合要素型として扱われることに注意してください。ビジネス・オブジェクト定義に含まれている属性は、対応する item タグの SOAP 属性になります。

単純型配列の arrayof 処理

arrayof ASI プロパティーは、SOAP でエンコードされた単純型配列の場合にのみ使用します。例えば、次のような直列化の場合を考えます。

```
<CustomerNames SOAP-ENC:arrayType="xsd:string[4]"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="SOAP-ENC:Array">
<item xsi:type="xsd:string">value1</item>
<item xsi:type="xsd:string">value2</item>
<item xsi:type="xsd:string">value3</item>
<item xsi:type="xsd:string">value4</item>
</CustomerNames>
```

この場合には、次の図 30 に示すようなビジネス・オブジェクト定義が必要になります。

Request	SOAP_echoStringArray_N0562488530_Request	1	ws_botype=request
CustomerNames	SOAP_echoStringArray_N0562488530_N1185926546_ArrayOfstring	n	arrayof=item,type_name=string,type_ns=http://www.w3.org/2001/XMLSchema
Item	String		type_name=string,type_ns=http://www.w3.org/2001/XMLSchema

図 30. arrayof ビジネス・オブジェクト

(分かりやすくするため、ビジネス・オブジェクトは Request レベルから表示します。)

注: ここには表示されていませんが、この例では、SOAP 構成 MO の TypeInfo プロパティを true に設定して、前述の SOAP 直列化をビジネス・オブジェクト構造から導き出す必要があります。

さらに、arrayof プロパティを使用すると、item 以外の名前の配列項目も作成できます。前記の例を使用すると、BO 属性名と「arrayof」ASI プロパティ値の両方が名前の場合、<item> タグは <name> タグに置き換えることができます。この結果、次のような直列化になります。

```
<CustomerNames SOAP-ENC:arrayType="xsd:string[4]"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="SOAP-ENC:Array">
<item xsi:type="xsd:string">value1</name>
<item xsi:type="xsd:string">value2</name>
<item xsi:type="xsd:string">value3</name>
<item xsi:type="xsd:string">value4</name>
</CustomerNames>
```

attr_name 処理および attr_ns 処理

作成される SOAP 属性に対応したネーム・スペースを指定することをお勧めします。そのためには、単純型の場合には ASI プロパティ attr_ns を指定します。データ・ハンドラーが attr_ns プロパティを処理するのは、同じ属性の ASI に attr_name が存在する場合のみです。attr_name および attr_ns には、以下の規則が適用されます。

1. attr_name も attr_ns も設定されていない場合には、ビジネス・オブジェクト属性は SOAP 要素に変換されます。
2. attr_name のみが設定されている場合には、SOAP 属性のネーム・スペースのデフォルト値として、その要素のネーム・スペースが使用されます。

```
<CustInfo Street="577 Airport"></CustomerInfo>
```

3. attr_ns のみが設定されている場合には、このプロパティは無視され、ビジネス・オブジェクト属性は SOAP 要素に変換されます。

4. attr_name と attr_ns の両方が設定されている場合には、次のような SOAP 属性が作成されます。

```
<CustInfo ns2:Street="577 Airport" xmlns:ns2=
"AttrNS"></CustomerInfo>
```

dh_mimetype: データ・ハンドラーの呼び出し

SOAP データ・ハンドラーを使用して別のデータ・ハンドラーを呼び出し、ビジネス・オブジェクトを、対応するデータ・ハンドラーが存在する任意の形式に書き出

することができます。これは、SOAP 子ビジネス・オブジェクトを SOAP ストリングに変換するときに、エンコードされたテキストを SOAP メッセージに追加することによって行います。

SOAP 要素の値をエンコードすることのできるフォーマットの 1 つに、RNIF 文書があります。この機能を使用するためには、任意のレベルの SOAP 子ビジネス・オブジェクトに RNIF BO を追加してください。この RNIF ビジネス・オブジェクトをストリングに変換するときに別のデータ・ハンドラーを呼び出すように SOAP データ・ハンドラーに通知するためには、その属性の ASI に dh_mimetype プロパティを追加します。ASI プロパティ dh_mimetype の値は、MO_DataHandler_Default メタオブジェクトで指定された適格な mimeType でなければなりません。この mimeType は、ビジネス・オブジェクトを処理するために呼び出すデータ・ハンドラーを判別するために使用されます。

図 31 に示す SOAP 子ビジネス・オブジェクトでは、CustomerInfo は複合子であり、RNET_Pip3A2PriceAndAvailabilityQuery は RNIF ビジネス・オブジェクトになっています。

Name	Type	App Spec Info
CustomerInfo	CustomerInfo	
Name	String	
CustID	String	
RNIFMsg	RNET_Pip3A2PriceAndAvailabilityQuery	elem_name=RNIFexample;dh_mimetype=application/x_rossettanet_agent;type_name=base64Binary;type_ns=http://www.w3.org/2001/XMLSchema;xsdtype=true

図 31. dh_mimetype を有する RNIF ビジネス・オブジェクト

このビジネス・オブジェクトから作成される SOAP メッセージは、次のようになります。

```
<CustomerInfo>
<Name>IBM Corporation</Name>
<CustID>95626</CustID>
<RNIFexample
xsi:type="xsd:base64Binary">1AWERYER238W98EYR9238728374871892787ASRJK23423
JKAWERJ234AWERIJHI423488R4HASF1AWERYER238W98EYR9238728374871892787ASRJK234
34JKAWERJ234AWERIJHI423488R4HASF1AWERYER238W98EYR9238728374871892787ASRJK2
4234JKAWERJ234AWERIJHI423488R4HASF1AWERYER238W98EYR9238728374871892787ASRJ
234234JKAWERJ234AWERIJHI423488R4HASFWR234
</RNIFexample>
</CustomerInfo>
```

RNIF サンプル要素には RNIF エンコード・ストリングが含まれていますが、これは、その要素値としてエンコードされた base64 バイナリーだったものです。また、ASI プロパティ elem_name、elem_ns、type_name、type_ns、および xsdtype は、このビジネス・オブジェクト属性との関係を維持します。この例では、メッセージが作成されると、指定された elem_name が SOAP 要素の名前になります。

注: 呼び出し先のデータ・ハンドラーによって戻される要素値がエンコード・テキストである場合には、type_name プロパティは base64Binary に設定されていなければならない、type_ns は xsd ネーム・スペースに対応していなければならない、また xsdtype は true に設定されていなければならない。

xsd:base64Binary: type_name および type_ns を xsd:base64Binary に解決されるように設定すると、SOAP データ・ハンドラーは、対応する要素の値を設定する前に、ビジネス・オブジェクトから得られた値をエンコードします。データ・ハンドラーは Apache API を使用してレジストリーに対して base64Binary シリアライザーの照会を行い、呼び出し先データ・ハンドラーから戻されたストリングをシリアライズし、要素の値を設定します。

スキーマ complexType の標識

以下のセクションでは、スキーマ complexType の標識がビジネス・オブジェクトに与える影響について説明します。標識には以下のものがあります。

- maxOccurs
- minOccurs
- all
- sequence
- choice

単純型の場合の maxOccurs および minOccurs 標識: maxOccurs 標識は、1 つの複合型の中である要素が出現することのできる、最大回数を表します。minOccurs 標識は、1 つの複合型の中である要素が出現する必要がある、最小回数を表します。

次のスキーマを検討してください。

```
<xs:element name="Address" type="Address">
<xs:complexType name="Address">
  <xs:sequence>
    <xs:element name="AddressLine" type="xsd:string" maxOccurs="10"/>
    <xs:element name="SuiteNumber" type="xsd:string" minOccurs="3"
      maxOccurs="unbounded"/>
    <xs:element name="City" type="xsd:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

上の例では、Address 要素内で AddressLine 要素が最大 10 回出現することができ、SuiteNumber 要素が少なくとも 3 回出現することを示しています。このスキーマに対応するビジネス・オブジェクトでは、次の ASI を備えたそれぞれの maxOccurs/minOccurs 標識ごとに、N カーディナリティーのラッパー・オブジェクトが 1 つ存在しなければなりません。

```
maxOccurs=N;wrapper=true
```

または

```
minOccurs=3;wrapper=true;
```

wrapper=true ASI は、このオブジェクトがラッパーであること、およびその理由から、SOAP メッセージに明示的に書き込まれないことを示しています。明示的な書き込みが行われなため、このラッパー・オブジェクトには、単純型の子が 1 つ存在しなければなりません。実行時に SOAP からビジネス・オブジェクトへの変換を行うために、データ・ハンドラーは、ラッパーの子オブジェクトを N 個読み取り、それぞれの子オブジェクトごとに対応する要素を作成します。ビジネス・オブジェ

クトから SOAP メッセージへの変換を行う際には、データ・ハンドラーは、検出されたそれぞれの要素ごとに、N カーディナリティー・ラッパー内に子オブジェクトを作成します。

これに対応する SOAP ビジネス・オブジェクトは、図 32 のようなものとなります。

Pos.	Name	Type	Key	Card	App Spec Info
1	☐ Address	Address	<input checked="" type="checkbox"/>	1	
1.1	☐ AddressLine	AddressLine_wrap	<input checked="" type="checkbox"/>	N	maxoccurs=10;wrapper=true
1.1.1	AddressLine	String	<input checked="" type="checkbox"/>		
1.1.2	ObjectEventId	String			
1.2	☐ SuiteNumber	SuiteNumber_wrap	<input checked="" type="checkbox"/>	N	minoccurs=3;wrapper=true
1.2.1	SuiteNumber	String	<input checked="" type="checkbox"/>		
1.2.2	ObjectEventId	String			
1.3	City	String	<input type="checkbox"/>		
1.4	ObjectEventId	String			
2	ObjectEventId	String			

図 32. SOAP ビジネス・オブジェクトにおける単純型 ASI の *minOccurs* および *maxOccurs*

図 32 に示したビジネス・オブジェクトに対応する SOAP メッセージは、次のようになります。

```
<Address xsi:type="ns0:Address">
  <AddressLine xsi:type="xsd:string">Line1</AddressLine>
  <AddressLine xsi:type="xsd:string">Line2</AddressLine>
  <SuiteNumber xsi:type="xsd:string">600</SuiteNumber>
  <SuiteNumber xsi:type="xsd:string">650</SuiteNumber>
  <SuiteNumber xsi:type="xsd:string">700</SuiteNumber>
  <City xsi:type="xsd:string">San Francisco</City>
</Address>
```

注: SOAP データ・ハンドラーは、*maxOccurs* 標識を処理する場合にも *minOccurs* 標識を処理する場合にも、要素の最大および最小出現回数を検証しません。データ・ハンドラーは、*maxOccurs* および *minOccurs* 標識を備えた特定の要素の複数のインスタンスを保持するためのコンテナ構造を提供するだけです。この方法は、単純型および複合型に適用されます。

複合型の場合の *maxOccurs* および *minOccurs* 標識: *<maxOccurs>* 標識は、1 つの複合型の中である要素が出現することのできる、最大回数を表します。

<minOccurs> 標識は、1 つの複合型の中である要素が出現する必要のある、最小回数を表します。次のスキーマにおける *maxOccurs* 標識について検討してください。

```
<xs:element name="Address" type="Address">
<xs:complexType name="Address">
  <xs:sequence>
    <xs:element name="AddressInfo" type="AddressInfo" maxOccurs="3"/>
    <xs:element name="City" type="xsd:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>

<xs:complexType name="AddressInfo">
  <xs:sequence>
```

```

        <xs:element name="StreetLine" type="xsd:string"/>
    </xs:sequence>
</xs:complexType>
</xs:element>

```

上の例では、Address 要素内で AddressInfo 複合型要素が最大 3 回出現することができることを示しています。このスキーマに対応するビジネス・オブジェクトには、ラッパー・オブジェクトがありません。complexType の AddressInfo 自体が N カーディナリティーの要素になりうるためです。N カーディナリティーの属性には、maxOccurs=3 という ASI が含まれます。

図 33 は、これに対応する SOAP ビジネス・オブジェクトを示しています。

Pos	Name	Type	Key	Card	App Spec Info
1	☐ Address	Address	<input checked="" type="checkbox"/>	1	
1.1	☐ AddressInfo	AddressInfo	<input checked="" type="checkbox"/>	N	maxOccurs=3
1.1.1	StreetLine	String	<input checked="" type="checkbox"/>		
1.1.2	ObjectEventId	String			
1.2	City	String	<input type="checkbox"/>		
1.3	ObjectEventId	String			
2	ObjectEventId	String			

図 33. SOAP ビジネス・オブジェクトにおける複合型 ASI の minOccurs および maxOccurs

図 33 に示したビジネス・オブジェクトに対応する SOAP メッセージは、次のようになります。

```

<Address xsi:type="ns0:Address">
  <AddressInfo xsi:type="ns0:AddressInfo">
    <StreetLine xsi:type="xsd:string">100 Market St.</ StreetLine>
    <StreetLine xsi:type="xsd:string">Apt 15</ StreetLine>
  </AddressInfo>
  <City xsi:type="xsd:string">San Francisco</City>
</Address>

```

all 標識: all 標識は、デフォルトでは、この complexType に関する子要素を任意の順序で指定できること、およびそれぞれの子要素がゼロまたは 1 回出現しなければならないことを指定します。次のスキーマについて考えてみましょう。

```

<complexType name="Item">
  <all>
    <element name="quantity" type="xsd:int"/>
    <element name="product" type="xsd:string"/>
  </all>
</complexType>

```

上記の例は、SOAP メッセージ内で要素 quantity および product を任意の順序で表示できることを示しています。quantity 要素を最初に表示して、product 要素を 2 番目に表示することも、逆の順序で表示することもできます。

図 34 は、このスキーマに対応するビジネス・オブジェクトを表しています。

Pos	Name	Type	Card	App Spec Info
1	Item	Item	1	all=Item_wrapper
1.1	Item_wrapper	Item_wrapper	N	wrapper=true
1.1.1	quantity	String		
1.1.2	product	String		
1.1.3	ObjectEventId	String		
1.2	ObjectEventId	String		
2	ObjectEventId	String		

図 34. SOAP ビジネス・オブジェクトにおける all 標識 ASI

対応する SOAP メッセージは次のようになります。

```
<Item xsi:type="ns0:Item">
  <quantity xsi:type="xsd:string">12</quantity>
  <product xsi:type="xsd:string">2</product>
</Item>
```

「all」コンテンツ・モデルを備えた配列コンテンツの処理: SOAP データ・ハンドラーは、このセクションで説明した「all」コンテンツ・モデルを備えた複合型の配列コンテンツを処理します。次の例では、ArrayOfSOAPStruct に SOAPStruct が格納されていますが、ここには「all」コンテンツ・モデルが存在します。

```
<complexType name="SOAPStruct">
  <all>
    <element name="varString" type="string" />
    <element name="varInt" type="int" />
    <element name="varFloat" type="float" />
  </all>
</complexType>
<complexType name="ArrayOfSOAPStruct">
  <complexContent>
    <restriction base="SOAP-ENC:Array">
      <attribute ref="SOAP-ENC:arrayType"
        wsdl:arrayType="typens:SOAPStruct[]" />
    </restriction>
  </complexContent>
</complexType>
```

SOAP データ・ハンドラーは、直列化において次の SOAP データを生成する必要があります。

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
  <ns0:echoStructArray xmlns:ns0="http://soapinterop.org/">
    <inputStructArray SOAP-ENC:arrayType="ns1:SOAPStruct[2]"
      xmlns:ns1="http://soapinterop.org/xsd" xsi:type="SOAP-ENC:Array">
      <item>
        <ns1:varFloat xsi:type="xsd:string">1.1</ns1:varFloat>
        <ns1:varInt xsi:type="xsd:string">1</ns1:varInt>
        <ns1:varString xsi:type="xsd:string">hi</ns1:varString>
      </item>
      <item>
        <ns1:varString xsi:type="xsd:string">hello</ns1:varString>
        <ns1:varInt xsi:type="xsd:string">1</ns1:varInt>
        <ns1:varFloat xsi:type="xsd:string">1.1</ns1:varFloat>
      </item>
    </inputStructArray>
  </ns0:echoStructArray>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```

    </inputStructArray>
  </ns0:echoStructArray>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

この例では、echoStructArray は操作の名前であり、inputStructArray は ArrayOfSOAPStruct という型を持つパラメーターの名前です。

sequence 標識: sequence 標識は、子要素が、complexType で指定されたとおりの順序で現れなければならないことを示します。

```

<complexType name="Item">
  <sequence>
    <element name="quantity" type="int"/>
    <element name="product" type="string"/>
  </sequence>
</complexType>

```

SOAP データ・ハンドラーは、この標識のための特別な ASI またはラッパー・オブジェクトを必要としません。デフォルトでは、データ・ハンドラーは、ビジネス・オブジェクトで指定されたとおりの順序で SOAP 要素の読み取りおよび書き込みを行います。

choice 標識: choice 標識は、complexType 内の要素のうちの 1 つだけを SOAP メッセージに含めることができることを示します。次のスキーマについて考えてみましょう。

```

<complexType name="Item">
  <choice>
    <element name="quantity" type="int"/>
    <element name="product" type="string"/>
  </choice>
</complexType>

```

SOAP データ・ハンドラーは、この標識のための特別な ASI またはラッパー・オブジェクトを必要としません。ビジネス・オブジェクトを SOAP メッセージに変換するとき、データ・ハンドラーは処理を中断し、SOAP メッセージにどの要素を含めるのかをユーザーが選択できるようにします。SOAP メッセージをビジネス・オブジェクトに変換するときには、データ・ハンドラーは既存の要素を読み取り、それに対応する属性を取り込みます。

sequence、choice、group および all における maxOccurs 標識: モデル・グループ (sequence、choice、group、および all) には、minOccurs 属性と maxOccurs 属性があります。minOccurs と maxOccurs のデフォルト値は、1 です。all グループの場合、maxOccurs に 1 以外の値は設定できません。SAP XI ODA および SOAP データ・ハンドラーでは、sequence、choice および group の maxOccurs について、考えられるすべての値をサポートしています。

SOAP からビジネス・オブジェクトへの変換における ASI

SOAP データ・ハンドラーは、受け取った SOAP メッセージを読み取って検証するために、ビジネス・オブジェクトの ASI を使用します。SOAP データ・ハンドラーによる ASI の検証には、以下の規則が適用されます。

- ヘッダーの処理と本文の処理は同じ方法で行われます。

- データ・ハンドラーに下記のセクションで述べる検証を行わせるためには、SOAP 構成 MO プロパティ `TypeCheck` を `strict` に、また `TypeInfo` を `true` に設定しなければなりません。
- 型検証は一般に `type_name` プロパティと `type_ns` プロパティの両方に依存するため、これらのプロパティの検証は並行して行われます。

注: 別の指示が示されていないかぎり、以下のセクションで述べるすべての ASI は属性レベルの ASI です。

elem_name の検証

単純、カーディナリティー 1、およびカーディナリティー n の属性の検証には、以下の規則が適用されます。

1. SOAP メッセージの構文解析中に要素が検出された場合、データ・ハンドラーは最初にビジネス・オブジェクト・レベルのすべての ASI を検索し、その要素の名前と `elem_name` 値の突き合わせを行います。
2. 要素の名前と `elem_name` 値が一致しない場合、データ・ハンドラーは、要素の名前を、そのビジネス・オブジェクト・レベルのそれぞれの属性名と突き合わせます。
3. どちらの検索も成功しなかった場合、データ・ハンドラーは失敗します。

elem_ns の検証

単純、カーディナリティー 1、およびカーディナリティー n の属性の検証には、以下のケースが当てはまります。

1. `elem_ns` ASI も、この要素に関する SOAP メッセージから得られる `xmlns` も存在しない場合、この要素が適切に検証されます。
2. `elem_ns` ASI が存在せず、それに対応する SOAP メッセージ内の要素で `xmlns` が指定されている場合、データ・ハンドラーはデフォルトの `elem_ns` として、ビジネス・オブジェクトから最後に読み取った、スコープ内にある `elem_ns` を使用します。データ・ハンドラーはこの値を、SOAP メッセージから得られた `xmlns` と比較します。両者が一致しない場合、検証は失敗します。
3. `elem_ns` ASI が存在し、それに対応する SOAP メッセージ内の要素で `xmlns` が指定されていない場合、データ・ハンドラーは、ASI で指定されている `elem_ns` が、SOAP メッセージの現行スコープ内のいずれかのネーム・スペースと一致するかどうかを調べます。両者が一致しない場合、検証は失敗します。

type_name および type_ns の検証

以下のセクションでは、`type_name` および `type_ns` の検証について説明します。

単純属性: `xsdType` が `true` の場合、`type_name` および `type_ns` の検証には以下の規則が適用されます。

- **type_name と type_ns の両方が指定されている場合** データ・ハンドラーは、`type_name` と `type_ns` のペアを使用して、対応する java Class オブジェクトを作成します。受け取った SOAP メッセージの `typename` と `typenamespace` を使用して、別の java Class オブジェクトが照会されます。2 つの java Class オブジェクトが一致した場合には、検証が成功します。それ以外の場合、検証は失敗します。

- **type_name も type_ns も指定されていない場合** データ・ハンドラーは、単純ビジネス・オブジェクト属性を java Class オブジェクトにマップします。受け取った SOAP メッセージの typename と typenamespace を使用して、別の java Class オブジェクトが照会されます。2 つの java Class オブジェクトが一致した場合には、検証が成功します。それ以外の場合、検証は失敗します。
- **type_name のみが指定されている場合** 単純型検証は失敗します。xsdType が true の場合には、type_name と type_ns がともに指定されているか、あるいはそのいずれも指定されていない必要があります。
- **type_ns のみが指定されている場合** 単純型検証は失敗します。xsdType が true の場合には、type_name と type_ns がともに指定されているか、あるいはそのいずれも指定されていない必要があります。

xsdType が false の場合、type_name および type_ns の検証には以下の規則が適用されます。

- **type_name と type_ns の両方が指定されている場合** データ・ハンドラーは SOAP メッセージの typename および typenamespace のペアと、ASI で指定された type_name 値および type_ns 値のペアを直接に比較します。両方のペアが正確に一致した場合、検証は成功します。それ以外の場合、検証は失敗します。
- **type_name も type_ns も指定されていない場合** データ・ハンドラーは、単純ビジネス・オブジェクト属性を java Class オブジェクトにマップします。受け取った SOAP メッセージの typename と typenamespace を使用して、別の java Class オブジェクトが照会されます。2 つの java Class オブジェクトが一致した場合には、検証が成功します。それ以外の場合、検証は失敗します。
- **type_name のみが指定されている場合** type_ns 値のデフォルト値として、ビジネス・オブジェクトの ASI から検出された要素ネーム・スペースが使用されます。データ・ハンドラーは、このデフォルト type_ns と ASI で指定された type_name を使用して、これらの値のペアと SOAP メッセージの typename および typenamespace のペアを、直接比較します。両方のペアが正確に一致した場合、検証は成功します。それ以外の場合、検証は失敗します。
- **type_ns のみが指定されている場合** type_name のデフォルト値として、ビジネス・オブジェクトの属性タイプが使用されます。データ・ハンドラーは、このデフォルト type_name と ASI で指定された type_ns を使用して、これらの値のペアと SOAP メッセージの typename および typenamespace のペアを、直接比較します。両方のペアが正確に一致した場合、検証は成功します。それ以外の場合、検証は失敗します。

複合属性 (カーディナリティー 1 および n): xsdType が true の場合、type_name および type_ns の検証には以下の規則が適用されます。

- **type_name と type_ns の両方が指定されている場合** xsdType は無視されます。データ・ハンドラーは、xsdType が false である場合と同様に処理を行います。
- **type_name も type_ns も指定されていない場合** xsdType は無視されます。データ・ハンドラーは、xsdType が false である場合と同様に処理を行います。
- **type_name のみが指定されている場合** xsdType は無視されます。データ・ハンドラーは、xsdType が false である場合と同様に処理を行います。
- **type_ns のみが指定されている場合** xsdType は無視されます。データ・ハンドラーは、xsdType が false である場合と同様に処理を行います。

xsdType が false の場合、type_name および type_ns の検証には以下の規則が適用されます。

- **type_name と type_ns の両方が指定されている場合** データ・ハンドラーは SOAP メッセージの typename および typenamespace のペアと、ASI で指定された type_name 値および type_ns 値のペアを直接に比較します。両方のペアが正確に一致した場合、検証は成功します。それ以外の場合、検証は失敗します。
- **type_name も type_ns も指定されていない場合** type_name のデフォルト値として、ビジネス属性タイプが使用されます。type_ns 値のデフォルト値として、ビジネス・オブジェクトの ASI から検出された要素ネーム・スペースが使用されます。データ・ハンドラーは、このデフォルトの振る舞いを使用して、これらの値のペアと SOAP メッセージの typename および typenamespace のペアを、直接比較します。両方のペアが正確に一致した場合、検証は成功します。それ以外の場合、検証は失敗します。
- **type_name のみが指定されている場合** type_ns 値のデフォルト値として、ビジネス・オブジェクトの ASI から検出された要素ネーム・スペースが使用されます。データ・ハンドラーは、このデフォルト type_ns と ASI で指定された type_name を使用して、これらの値のペアと SOAP メッセージの typename および typenamespace のペアを、直接比較します。両方のペアが正確に一致した場合、検証は成功します。それ以外の場合、検証は失敗します。
- **type_ns のみが指定されている場合** type_name のデフォルト値として、ビジネス・オブジェクトの属性タイプが使用されます。データ・ハンドラーは、このデフォルト type_name と ASI で指定された type_ns を使用して、これらの値のペアと SOAP メッセージの typename および typenamespace のペアを、直接比較します。両方のペアが正確に一致した場合、検証は成功します。それ以外の場合、検証は失敗します。

attr_name および attr_ns の検証

SOAP メッセージをビジネス・オブジェクトに読み込む際には、SOAP 属性を探すために個々の SOAP 要素が検索されます。SOAP 属性が検出された場合は、対応するビジネス・オブジェクトから得られた attr_name プロパティ値と比較されます。例えば、次の SOAP メッセージについて考えてみましょう。

```
<CustInfo City="4" State="5" Street="2" Zip="6">  
  <Name xsi:type="xsd:string">1</Name>  
  <Street2 xsi:type="xsd:string">3</Street2>  
</CustInfo>
```

ここで、図 35 に示すビジネス・オブジェクト定義構造について考えてみます (各属性の右に属性レベル ASI が指定されています)。

Name	Type	App Spec Info
☐ CustInfo	CustomerInfo	
Name	String	
Street1	String	attr_name=Street
Street2	String	
City	String	attr_name=City
State	String	attr_name=State
Zip	String	attr_name=Zip

図 35. attr_name および attr_ns の検証

データ・ハンドラーは以下の処理ステップに従います。

1. 要素名 CustInfo を読み取ります。
2. この要素名に対応するビジネス・オブジェクト属性を解決します。
3. SOAP 要素の属性を読み取り、それらの子属性の ASI と突き合わせます。この例では、SOAP メッセージの Street とビジネス・オブジェクト属性 Street1 の突き合わせ、City とビジネス・オブジェクト属性 City の突き合わせ、さらに、以下同様の突き合わせが行われます。
4. CustInfo の子要素が、本文の残りの部分と同じ方法で読み取られ、処理されず。

注: attr_ns の検証は行われません。

データ・ハンドラーは、所定の要素の SOAP 属性について繰り返し処理を行います。属性に出会うたびに、データ・ハンドラーは、対応する属性のためのビジネス・オブジェクトを検索します。ビジネス・オブジェクトが見付かったら、そのビジネス・オブジェクト属性に SOAP 属性の値を取り込みます。対応するビジネス・オブジェクト属性が見付からない場合、データ・ハンドラーは次の SOAP 属性に移って処理を続けます。

SOAP データ・ハンドラー内部からのデータ・ハンドラーの呼び出し

SOAP データ・ハンドラーは、別のデータ・ハンドラーを使用して、エンコードされた要素値を、SOAP メッセージからビジネス・オブジェクトに読み込むことができます。例えば、SOAP 要素の値がエンコードされるフォーマットの 1 つに、RNIF 文書があります。この機能を使用するためには、RNIF ビジネス・オブジェクトを任意のレベルの SOAP 子ビジネス・オブジェクトに追加します。この RNIF エンコード・ストリングを RNIF ビジネス・オブジェクトに変換するとき別のデータ・ハンドラーを使用しなければならないことを SOAP データ・ハンドラーに通知するためには、その属性の ASI に dh_mime_type プロパティを追加する必要があります。dh_mime_type ASI の値は、MO_DataHandler_Default ビジネス・オブジェクトで指定された適格な mimeType でなければなりません。mimeType は、ストリング上で使用するデータ・ハンドラーを判別するために使用されます。例えば、次の SOAP メッセージでは、RNIFExample が SOAP 要素で、ここには RNIF エンコード・ストリングが含まれています。

```
<CustInfo>
<Name>IBM Corporation</Name>
<CustID>95626</CustID>
<RNIFExample xsi:type="xsd:base64Binary">
1AWERYER238W98EYR9238728374871892787ASRJK234234JKAWER
```

```

J234AWERIJHI423488R4HASF1AWERYER238W98EYR923872837487
1892787ASRJK234234JKAWERJ234AWERIJHI423488R4HASF1AWER
YER238W98EYR9238728374871892787ASRJK234234JKAWERJ234A
WERIJHI423488R4HASF1AWERYER238W98EYR92387283748718927
87ASRJK234234JKAWERJ234AWERIJHI423488R4HASFWR234
</RNIFexample>
</CustomerInfo>

```

SOAP ビジネス・オブジェクトは、図 36 のようなものとなります。

Name	Type	App Spec Info
☐ CustInfo	CustomerInfo	
Name	String	
Street1	String	attr_name=Street
Street2	String	
City	String	attr_name=City
State	String	attr_name=State
Zip	String	attr_name=Zip

図 36. RNIFexample ビジネス・オブジェクト

RNIFexample 要素には、その要素値として RNIF エンコード・ストリングが含まれています。また、ASI プロパティ `elem_name`、`elem_ns`、`type_name`、`type_ns`、および `xsdtype` は、このビジネス・オブジェクト属性との関係を依然として維持しています。

注: 呼び出し先のデータ・ハンドラーによって戻される要素値がエンコード・テキストである場合には、`type_name` プロパティは `base64Binary` に設定されていなければならない、`type_ns` は `xsd` ネーム・スペースに対応していなければならない、また `xsdtype` は `true` に設定されていなければならない。

デフォルトのビジネス・オブジェクト解決

SOAP からビジネス・オブジェクトへの変換では、SOAP データ・ハンドラーと SAP XI サービス・コネクタは、情報を交換してビジネス・オブジェクト名を解決する特殊な契約に準拠しています。このコネクタは、SOAP データ・ハンドラーに、`BodyName` と `BodyNamespace` のペアにマップされるビジネス・オブジェクト名のリストを提供します。さらに、TLO に `defaultfault` ビジネス・オブジェクトが設定されている場合は、この情報がデータ・ハンドラーに渡されます。この情報が渡されると、SOAP データ・ハンドラーは、以下のステップに従って処理します。

1. SOAP メッセージを受け取ります。
2. そのメッセージが SOAP 要求メッセージか応答メッセージか、あるいは障害メッセージかを判断します。
 - a. SOAP 要求メッセージまたは応答メッセージである場合は、SOAP-ENV:Body 要素の最初の子要素から、`BodyName` および `BodyNamespace` を読み取ります。

- b. SOAP 障害メッセージである場合は、その障害メッセージ内の detail 要素の最初の子要素から、BodyName および BodyNamespace を読み取ります。障害メッセージに detail 要素がない場合は、この変換用の defaultfault ビジネス・オブジェクトを使用します。
3. defaultfault ビジネス・オブジェクトがまだ選択されていない場合、データ・ハンドラーは、ステップ 2 で検出した BodyName および BodyNamespace を、コネクタが提供するリストで検出されるペアと一致させようとします。一致させられれば、ビジネス・オブジェクトの解決は成功したことになります。一致しない場合、データ・ハンドラーは失敗し、失敗を意味するエラー・メッセージが出されます。

プラグ可能な名前ハンドラーの指定

デフォルトのビジネス・オブジェクト解決を使用すると、SOAP メッセージからビジネス・オブジェクトへの変換で使用されるビジネス・オブジェクトを判別するために、プラグ可能な名前ハンドラーを指定することができます。これを実行するには、MO_DataHandler_DefaultSOAPConfig 属性を変更します。

MO_DataHandler_DefaultSOAPConfig には、特に、次の内容を指定する string 型の属性が 2 つあります。

- **ClassName** SOAP データ・ハンドラーの基本クラスのクラス名。プラグ可能な名前ハンドラーを指定する場合には、この属性値を変更しないでください。
- **SOAPNameHandler** SOAPNameHandler 属性では、どの名前ハンドラーを呼び出すのかを指定します。プラグ可能な名前ハンドラーには、値を指定できます。このプロパティの値は、クラス名でなければなりません。SOAPNameHandler クラスは、以下のシグニチャーを備えた抽象クラスです。

```
public abstract String getBOName(Envelope msgEnv, SOAPProperty prop)
```

SOAPNameHandler 属性に値が指定されている場合、SOAP データ・ハンドラーは、指定された名前ハンドラーを呼び出します。この値が存在しない場合、または指定された名前ハンドラーがビジネス・オブジェクト名を入手できなかった場合には、デフォルトのビジネス・オブジェクト解決を行うために、SOAP データ・ハンドラーがデフォルトで呼び出されます。

SOAP データ・ハンドラーは、MO に指定されている SOAPNameHandler プロパティを使用して、カスタム名ハンドラー・クラスのインスタンスを生成します。次に、getBOName を呼び出して、ビジネス・オブジェクト名を解決します。SOAP DataHandler は、コネクタから受け取った SOAPProperty オブジェクトをカスタム名ハンドラー実装クラスに渡します。

この SOAPProperty オブジェクトには、解決のための潜在的な候補 BO の構造化リストが格納されています。このリストには、BodyName、BodyNamespace、および BOName の各トリプレットが登録されています。これらのトリプレットは、SOAP 構成 MO 構成情報が基本になっています。デフォルトの名前ハンドラーは、このオブジェクトを使用して BO を解決します。カスタム名ハンドラーの開発者は、このオブジェクトを自由に使用できます。

SOAPProperty オブジェクトの使用

SOAPPropertyUtils クラスは、SOAPProperty からビジネス・オブジェクト名を抽出するときに使用します。そのためには、次のメソッドを使用します。

```
/**
 * Retrieve the business object name based on the body name and the body
 * namespace
 * .
 * @param soapProp top level SOAPProperty object that is passed by the
 * connector
 * @param bodyName body name from the SOAP message
 * @param uri body namespace from the SOAP message
 * @return business object name from the SOAPProperty object with the body
 * name and body namespace.
 */
java.lang.String findBOName(SOAPProperty soapProp, String name, String uri);
```

NameHandler のサンプル

次に示すのは、NameHandler:package のサンプルです。

```
com.ibm.adapters.datahandlers.soap.namehandlers;
// DOM and Parsers
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.InputSource;
// Apache Xerces and SOAP
import org.apache.soap.Envelope;
import org.apache.soap.Header;
import org.apache.soap.Body;
import org.apache.soap.Constants;
import org.apache.soap.util.xml.DOMUtils;
import org.apache.soap.util.xml.XMLParserUtils;
import org.apache.soap.util.xml.QName;
import org.apache.soap.encoding.soapenc.SoapEncUtils;
import org.apache.soap.encoding.soapenc.Base64;
// java
import java.util.Vector;
// SOAP data handler
import com.ibm.adapters.datahandlers.soap.*;
import com.ibm.adapters.datahandlers.soap.exceptions.*;
public class MyCustomNameHandler extends SOAPNameHandler {
    private static final String BOPREFIX = "MyCustomBOPrefix";
    private static final char UNDERSCORE = '_';
    private static final char EMPTY_STRING = "";

    public String getBOName(Envelope msgEnv, SOAPProperty prop)
        throws SOAPNameHandlerException
    {
        // Initialize a String Buffer
        StringBuffer boName = new StringBuffer();
        // Determine the "MyCustomBOPrefix" SOAP data handler
        // MO property. If it exists, and is populated append
        // this prefix to the front of the BOName.
        String pref = dh.getOption(BOPREFIX);
        if (pref != null) {
            boName.append(pref.equals(EMPTY_STRING)
                ? EMPTY_STRING : pref + UNDERSCORE);
        }
        // Begin parsing the SOAP msg envelope.
        Element bodyEl, requestEl;
        Body msgBody = msgEnv.getBody();
        Vector bodyEntries = msgBody.getBodyEntries();
        if((bodyEntries == null) || (bodyEntries.size() <= 0))
            throw new SOAPNameHandlerException("No Body Entries exist
```

```

        for this SOAP message. Cannot determine BOName to use.");
// Grab the first <SOAP-ENV:Body> Element
bodyEl = (Element) bodyEntries.elementAt(0);
// Grab the first Child Element of the <SOAP-ENV:Body>
// Element
requestEl = (Element) DOMUtils.getFirstChildElement(bodyEl);
// Read the name and namespace of this first child
String name = bodyEl.getLocalName();
String uri = bodyEl.getNamespaceURI();
if (uri == null)
    uri = Constants.NS_URI_SOAP_ENV;
// Use the SOAPPropertyUtils findBOName() method to search
// the SOAPProperty object for this messages first element
// name and namespace. If no match is found, a
// SOAPDataHandlerException will be thrown. If a match is
// found, and it's not an empty string, append to the boname.
String returnedBOName = SOAPPropertyUtils.findBOName(prop, name, uri);
if (returnedBOName != null &&
    !returnedBOName.equals(EMPTY_STRING))
    boName.append(returnedBOName);
return boName.toString()
}
}
}

```

制約事項

以下のセクションでは、データ・ハンドラーの制約事項について説明します。

SOAP の Style および Use に関するガイドライン

SOAP メッセージは、SAP XI サービスによって定義された Style および Use に基づいて作成されます。SOAP データ・ハンドラーが提供するサポートのレベルは、表 36 に示すとおりです。

表 36. Style および Use のガイドライン

Style	Use	パーツを定義するもの	データ・ハンドラーのサポート
document	literal	element	完全
document	literal	type	限定 (下記を参照)
document	encoded	element	なし
document	encoded	type	限定 (下記を参照)
rpc	literal	element	なし
rpc	literal	type	完全
rpc	encoded	element	なし
rpc	encoded	type	完全

パーツおよびパーツ要素の順序

SOAP データ・ハンドラーが SOAP メッセージをビジネス・オブジェクトに変換する際に、SOAP メッセージが document/literal/type または document/encoded/type のフォーマットに従う場合、メッセージのパーツは、WSDL で記述されたとおりの順序になっていなければなりません。例えば、次の WSDL を検討してください。

```

<operation name="GetQuote"
  style="document" ...>
<input>

```

```

    <soap:body parts="Part1 Part2 Part3 Part4" use="literal">
</input>
</operation>

<definitions
  xmlns:stns="(SchemaTNS)"
  xmlns:wtns="(WsdITNS)"
  targetNamespace="(WsdITNS)">

<schema targetNamespace="(SchemaTNS)"
  elementFormDefault="qualified">
<element name="SimpleElement" type="xsd:int"/>
<element name="CompositElement" type="stns:CompositeType"/>
<complexType name="CompositeType">
<all>
  <element name='elem_a' type="xsd:int"/>
  <element name='elem_b' type="xsd:string"/>
</all>
</complexType>
</schema>

<message...>
<part name='Part1' type="stns:CompositeType"/>
<part name='Part2' type="xsd:int"/>
<part name='Part3' element="stns:SimpleElement"/>
<part name='Part4' element="stns:CompositElement"/>
</message>
0
</definitions>

```

SOAP メッセージは、パーツによって定義された順序に従うなければなりません。下の SOAP 例では、Part1 要素が Part2、Part3、および Part4 要素に先行しています。正しいビジネス・オブジェクト解決を行うためには、この順序を維持しなければなりません。

```

<soapenv:body... xmlns:mns="(MessageNS)"
  xmlns:stns="(SchemaTNS)">
  <stns:elem_a>123</stns:elem_a>
  <stns:elem_b>hello</stns:elem_b>
  <soapenc:int>123</soapenc:int>123</soapenc:int>
  <stns:SimpleElement>123</stns:SimpleElement>
  <stns:CompositElement>
    <stns:elem_a>123</stns:elem_a>
    <stns:elem_b>hello</stns:elem_b>
  </stns:CompositElement>
</soapenv:body>

```

SOAP メッセージが document/literal/type フォーマットまたは document/encoded/type フォーマットに従う場合、パーツ要素も定義された順序に従う必要があります。上の例の Part1 で、elem_a タグは elem_b タグの前になければなりません。この制約は、データ・ハンドラーのビジネス・オブジェクト解決プロセスで指示されています。document スタイルの場合のデフォルトのビジネス・オブジェクト解決には、最初の要素の本文の名前およびネーム・スペースが使用されるため、これらを、特定の要求、応答、または障害に関するすべての SOAP メッセージで同じ要素になるようにして、要求、応答、または障害のいずれの場合にも同じビジネス・オブジェクトに解決されるようにする必要があります。

注: SOAP メッセージが document/literal/type フォーマットまたは document/encoded/type フォーマットに従う場合、要素をオプションにしているではありません。

XML の制約事項

以下の XML 構造、機能、および表記はサポートされません。

- 多次元配列
- 部分的に伝送される配列
- 疎配列
- 混合コンテンツ
- `maxOccurs` が 1 より大きいシーケンス、グループ、および選択モデル・グループ・コンポーネント

第 6 章 要求処理のためのコラボレーションの有効化

- 『要求処理コラボレーションのチェックリスト』

この章では、要求処理のためのコラボレーションを使用可能にする場合に実行する必要のある手順について説明します。コラボレーションは、コネクタを使用して SAP XI サービスを呼び出します。

要求処理コラボレーションのチェックリスト

Business Object Designer を使用してビジネス・オブジェクトを生成することは、コラボレーション開発のプロセスの一部として行われます。コラボレーションが SAP XI サービスを呼び出すために使用するビジネス・オブジェクトを生成するには、以下の作業を行う必要があります (これらについては、以下のセクションで説明します)。

1. URL、UDDI、ファイル・システムのいずれかから、WSDL 文書を識別する。このタスクにはサード・パーティー製のツールを使用してください。SAP XI コネクタではこのタスクのためのツールは提供されません。
2. SAP XI Integration Developer WSDL ツールを使用して、WSDL 文書を生成する。
3. BIA_XIWSDLUtil ツールを使用して、SAP XI 文書を標準に準拠した WSDL 文書に変換する。
4. Business Object Designer を開いて SAP XI ODA を立ち上げる。詳細については、141 ページの『SAP XI ODA の始動』を参照してください。
5. ODA を構成する。
6. 選択内容を確認する。
7. Business Object Designer を使用して、要求ビジネス・オブジェクト、および (同期要求の場合に) 応答ビジネス・オブジェクトと障害ビジネス・オブジェクト、SOAP 構成 MO、プロトコル構成 MO、ヘッダー・コンテナ、およびそれぞれのオブジェクトと属性に適した子オブジェクトとアプリケーション固有情報を含む、トップレベルのビジネス・オブジェクトを生成する。このプロセスは、WSDL ODA によって自動化されます。

ビジネス・オブジェクトを生成した後で、コラボレーションがコネクタおよび SOAP データ・ハンドラーを使用して SAP XI サービスを呼び出せるようにするために、いくつかの作業を行う必要があります。コラボレーション・テンプレートやオブジェクトの作成、そのポートのバインディングを含む、コラボレーションの開発手順については、「*IBM WebSphere InterChange Server コラボレーション開発ガイド*」を参照してください。汎用ビジネス・オブジェクトと SAP XI ODA によって生成されるアプリケーション固有のビジネス・オブジェクトとの間のマップを作成するための詳細については、「*IBM WebSphere InterChange Server マップ開発ガイド*」を参照してください。

第 7 章 Web サービスとしてのコラボレーションの公開

- 『手順のチェックリスト』
- 128 ページの『ビジネス・オブジェクトの識別または開発』
- 128 ページの『コラボレーション・テンプレートの選択または開発』
- 129 ページの『新規コラボレーション・オブジェクトのポートのバインディング』
- 130 ページの『WSDL 構成ウィザード』
- 132 ページの『TLO フォーマットのビジネス・オブジェクトの WSDL 構成ウィザード処理』
- 135 ページの『処理の要件および例外』

この章では、コラボレーションを SAP XI サービスとして公開するための、設計段階における手順について説明します。これにより、Web サービス・クライアントがコラボレーションを呼び出したときにコネクタがイベントを処理できるようになります。

設計ツールが統合されているため、コラボレーションを SAP XI サービスとして公開する作業を容易に行うことができます。SAP XI サービス用にコラボレーションとビジネス・オブジェクトを構成した後で、WSDL 構成ウィザードを使用します。このウィザードは、コラボレーションを SAP XI サービスとして表現する WSDL 文書および XML スキーマを作成します。WSDL 出力は、コラボレーションを記述するだけでなく、SAP XI クライアントがコラボレーションを呼び出すための基礎を構成します。

手順のチェックリスト

コラボレーションを SAP XI サービスとして公開するには、以下のセクションで説明する作業を行う必要があります。

1. 要求 SOAP メッセージ、およびオプションで (同期イベント処理のために) 応答および障害 SOAP メッセージとして使用されるビジネス・オブジェクトを識別するか、あるいは必要に応じて開発します。それらのオブジェクトを生成するには、以下の 2 通りの方法があります。1) Business Object Designer を使用して手動で生成する方法。2) SAP XI サービスに対して WSDL インターフェース・ファイルが存在する場合には、SAP XI ODA を使用して、要求ビジネス・オブジェクト、その他の (応答または障害) ビジネス・オブジェクトを生成する方法。上記の 2 番目の方法を使用する場合は、以下の手順を実行します。
 - a. Collaboration WSDL ODA 構成プロパティで、コラボレーションの名前を指定します。この値は、TLO の ws_collab ASI を示します。
 - b. WSDL インターフェース・ファイルに対して、WSDL_URL または UDDI_InquiryAPI_URL WSDL ODA 構成プロパティを指定します (使用しているネットワークまたはローカルに WSDL インターフェース・ファイルが常駐する場合は、そのファイルに対するディレクトリー・パスを指定することもできます)。

詳細については、141 ページの『SAP XI ODA の始動』を参照してください。

2. ビジネス・オブジェクトを使用するには、コラボレーションのテンプレートを開発するか、あるいは既存のテンプレートを選択します。
3. SAP XI サービス用のコラボレーション・オブジェクトとそのポートを作成します。

最初に、コラボレーション・オブジェクトがビジネス・オブジェクトを正しく取り込むことを確認しておく必要があります。コラボレーション・オブジェクトを作成するための詳しい情報およびステップごとの手順については、「*WebSphere InterChange Server システム・インプリメンテーション・ガイド*」を参照してください。

注: コラボレーション・オブジェクトは、そのマップが適切な変換を実行できるように構成されていることが必要です。マップにより、SOAP 要求メッセージの形で受け取ったビジネス・オブジェクトは、コラボレーションが使用するビジネス・オブジェクトに変換されます。コラボレーションが返すビジネス・オブジェクトを SOAP 応答メッセージに組み込まれたビジネス・オブジェクトに変換するときにも、マップが使用されます。マッピングおよびマッピング手順について詳しくは、「*マップ開発ガイド*」を参照してください。

4. WSDL 構成ウィザードを使用して、WSDL 文書を作成します。このユーティリティは、Web サービス・コネクタの構成も行います。

注: WSDL 構成ウィザードは、インプリメンテーション・ファイル、インターフェース・ファイル、および 1 つ以上のスキーマ・ファイルを作成します。本書では、これらの出力を一括して WSDL 文書として参照します。

5. 必要に応じて WSDL 文書を公開します。

注: コネクタは、WSDL 文書を公開するためのツールを提供しておらず、また公開のサポートもしません。

ビジネス・オブジェクトの識別または開発

Business Object Designer を使用してビジネス・オブジェクトを作成し、また、Connector Configurator を使用して、ビジネス・オブジェクトをサポートするようにコネクタを構成します。

Business Object Designer の詳細については、「*Business Object Designer*」を参照してください。Web サービス・ビジネス・オブジェクトについて詳しくは、21 ページの『第 3 章 ビジネス・オブジェクトの要件』を参照してください。

コラボレーション・テンプレートの選択または開発

選択または開発するコラボレーション・テンプレートは、SAP XI サービスとして公開するためのシナリオを 1 つまたは複数備えていなければなりません。コラボレーション・テンプレートの詳細については、「*コラボレーション開発ガイド*」を参照してください。

新規コラボレーション・オブジェクトのポートのバインディング

ビジネス・オブジェクト・タイプ用にコラボレーション・テンプレートのポートを構成した後で、コラボレーション・オブジェクトを作成し、そのポートを SAP XI コネクタのインスタンスにバインドする必要があります。

新規コラボレーション・オブジェクトを作成してそのポートを SAP XI コネクタのインスタンスにバインドするには、次のようにしてください。

1. 「コラボレーション・オブジェクト (Collaboration Objects)」フォルダーを右マウス・ボタンでクリックし、「新規コラボレーション・オブジェクトの作成 (Create New Collaboration Object)」を選択します。これによって「新規コラボレーションの作成 (Create New Collaboration)」ウィンドウが現れ、テンプレートのリストが示されます (図 37 を参照)。

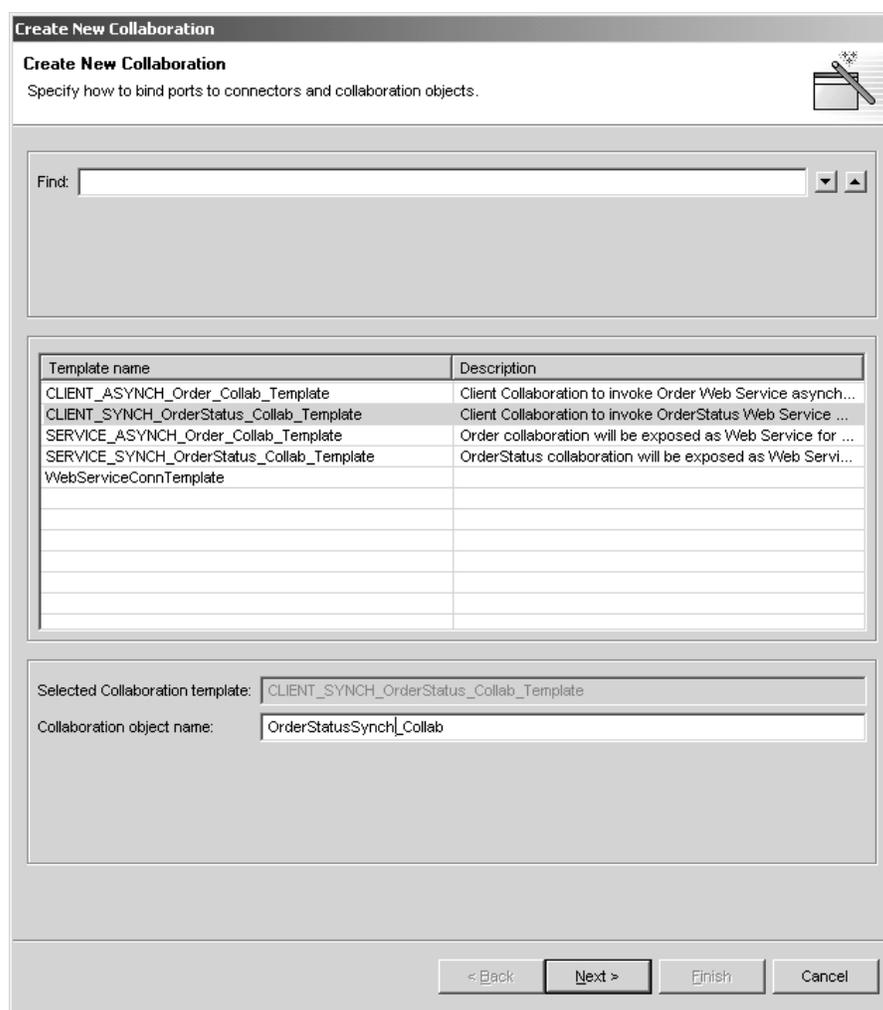


図 37. 「新規コラボレーションの作成 (Create New Collaboration)」ウィンドウ

2. 「テンプレート名 (Template Name)」からコラボレーション・テンプレートを選択し、「コラボレーション・オブジェクト名 (Collaboration object name)」フィールドにコラボレーション・オブジェクトの名前を入力します。これにより、図 38 に示すような「ポートのバインド (Bind Ports)」ウィンドウが表示されます。

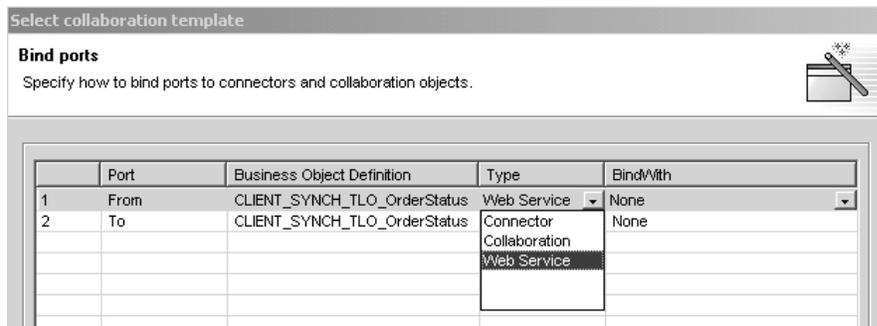


図 38. 「ポートのバインド (Bind Ports)」 ウィンドウ

3. ポートを選択し、「タイプ (Type)」の矢印をクリックしてポートのプルダウン・メニューを表示して、「Web サービス (WebService)」を選択します (図 38 を参照)。

Web サービス・コネクタのすべてのインスタンスは、アプリケーション固有プロパティ ConnectorType を備えています。デフォルトでは、このプロパティは WebService に設定されています。System Manager の「コラボレーション・ポートのバインド (Bind Collaborations Port)」ウィンドウでは、どのコネクタが Web サービス・コネクタかを判別するために、ConnectorType プロパティの値が使用されます。

4. 「バインド先 (BindWith)」の矢印をクリックしてコネクタ・インスタンスのリストを表示します。System Manager は、ConnectorType プロパティの値が WebService に設定されているコネクタのインスタンスを表示します。SAP XI サービス・コネクタのインスタンスを選択してください。
5. 「完了」をクリックします。

これで、WSDL 構成ウィザードを実行する準備ができました。

WSDL 構成ウィザード

コラボレーション・オブジェクトを作成して、そのトリガー・ポートを SAP XI コネクタのインスタンスにバインドした後は、いつでも WSDL 構成ウィザードを使用することができます。このユーティリティーは、ユーザーがコラボレーション、ビジネス・オブジェクト定義、およびコネクタに関して指定した、バインディング、ポート名、操作、およびその他のデータを使用して、WSDL インプリメンテーション・ファイル (*.impl.wsdl)、WSDL インターフェース・ファイル (*.wsdl)、および xml スキーマ・ファイル (*.xsd) を作成します。これらのファイルは、1 つの SAP XI サービスとして公開されるコラボレーション複合体であり、このユーティリティーでは、これらを複数の別個のファイルとして生成するのか、1 つのファイルとして生成するのかを指定することができます。このユーティリティーは SOAP over HTTP および HTTPS プロトコルをサポートします。プロトコル・リスナー・フレームワークの構成情報は、コネクタ固有プロパティ ProtocolListenerFramework から検索されます。このプロパティはまた、リスナーのリストを使用できるようにします。

2. 必要に応じて以下の値を入力します。
 - **サービス名** デフォルトでは、このコラボレーション・オブジェクトを記述するためにユーザーが使用した名前。
 - **ディレクトリー名** SAP XI のためのアダプターとコラボレーションのテンプレートおよびオブジェクトが入るディレクトリー。
 - **ターゲット・ネーム・スペース** SAP XI サービスとして公開されるコラボレーションの URL。
 - **コラボレーション・ポート** これらのフィールド内の情報は、コラボレーション・オブジェクト構成手順の「ポートのバインド (Bind Ports)」ウィンドウで指定されます。
 - **TLO 以外の場合のコラボレーション・モード** これは、TLO を使用している場合には適用されません。あるいは、非 TLO オブジェクトを入力用に使用している場合には、synchronous または asynchronous を指定しなければなりません。
 - **スキーマおよび WSDL** これらの出力を単一ファイルに収めるのか、別個のファイルに収めるのかを指定してください。
3. 「完了」をクリックします。このユーティリティーは、ユーザーが入力した入力データおよび指定内容に基づいて出力を生成します。次のセクションに、これらすべての要約を示します。

TLO フォーマットのビジネス・オブジェクトの WSDL 構成ウィザード処理

構成ウィザードは、SAP XI コネクターにバインドされるコラボレーション・オブジェクトの各トリガー・ポート用に、WSDL オペレーションを作成します。このオペレーションは、このコラボレーションの呼び出しに関連付けられているビジネス・オブジェクトに基づいて作成されます。

構成ウィザードは、オブジェクト・レベル ASI の ws_eventtlo を読み取って、そのビジネス・オブジェクトが TLO フォーマットであるかどうかを判断します。ASI プロパティーが true に設定されていれば、そのビジネス・オブジェクトは TLO です。TLO によって、以下の WSDL プロパティーが検出されます。

- **Operation Name および BodyNS** ウィザードは、TLO フォーマットのビジネス・オブジェクトを検出すると、その TLO の SOAP 要求ビジネス・オブジェクト内の SOAP 構成 MO の BodyName プロパティーを使用して、オペレーション名を作成します。同様に、ウィザードは、同じ SOAP 構成 MO 内の BodyNS プロパティーをメッセージのネーム・スペースであると判断します。
- **Execution Mode** TLO のビジネス・オブジェクト・レベル ASI から ws_mode プロパティーをインスペクションすることにより、ウィザードは、実行モードが同期か非同期かを判断して、それぞれ REQUEST_RESPONSE または ONE_WAY WSDL を作成します。

TLO に基づいて WSDL オペレーションを作成する場合、コラボレーションの構成には、マップを使用する方法と使用しない方法の 2 つの方法があります。

マップを使用する TLO: コラボレーションは、一般に、汎用ビジネス・オブジェクト (GBO) 要求を受け入れるように構成されています。つまり、コラボレーション・テンプレートのトリガー・ポートは、GBO にサブスクライブしています。この

ような状況で TLO を使用するには、コラボレーションが SAP XI コネクターにバインドされていることと、コネクターがマップを介した GBO の TLO への変換をサポートしていることが必要です。図 40 に、この例を示します。

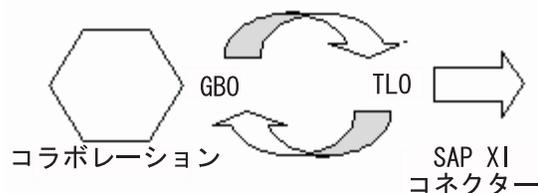


図 40. マップを使用する TLO

コラボレーションとコネクターがこの方法で構成されている場合、ウィザードは、WSDL 文書で記述されているオペレーションの作成に TLO ビジネス・オブジェクトが使用されると判断します。この判断は、コネクターでサポートしているビジネス・オブジェクトとそれに関連したマップをインスペクションすることによって行われます。SAP XI コネクターのランタイム処理にとっては、構成されたマップが常にコラボレーションの GBO を 1 つの TLO だけに変換することが重要です。また、インバウンド・マップのソース・ビジネス・オブジェクトと宛先ビジネス・オブジェクトが、それぞれ、アウトバウンド・マップの宛先ビジネス・オブジェクトとソース・ビジネス・オブジェクトに変換されることも重要です。

マップを使用しない TLO: ウィザードは、マップを使用しない TLO の処理もサポートしています。この場合、コラボレーション・テンプレートのトリガー・ポートは TLO に直接サブスクライブします。SAP XI コネクターは TLO をサポートしているため、マップは必要ありません。図 41 は、この例を示したものです。

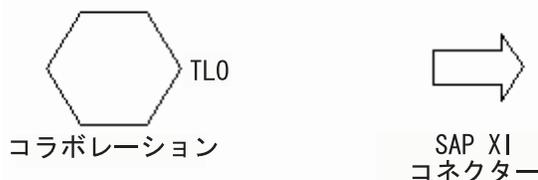


図 41. マップを使用しない TLO

コラボレーションとコネクターがこの方法で構成されている場合、ウィザードは、コラボレーションで検出される TLO ビジネス・オブジェクトを使用して、WSDL 文書で記述されているオペレーションを作成します。ウィザードは、このポート用に構成されているマップは存在しないと判断します。

非 TLO フォーマットのビジネス・オブジェクトの WSDL 構成ウィザード処理

非 TLO のビジネス・オブジェクトがサポートされていれば、既存のコラボレーションおよびマップを使用して SAP XI サービスとして公開することができます。このため、ウィザードは、TLO フォーマット以外のビジネス・オブジェクトの使用による WSDL オペレーションの作成もサポートしています。

TLO プロセスと同様、ウィザードは、オブジェクト・レベル ASI の ws_eventtlo を読み取って、そのビジネス・オブジェクトが非 TLO フォーマットであると判断します。ASI プロパティが存在しないか、存在しても true 以外の値に設定されている場合は、このビジネス・オブジェクトは非 TLO です。非 TLO は、Web サービスの TLO 構造に準拠していない任意のビジネス・オブジェクトです。ウィザードは、非 TLO を使用して以下のプロパティを検出します。

- **Operation Name および BodyNS** ウィザードは、非 TLO フォーマットのビジネス・オブジェクトを検出すると、コラボレーション名、ビジネス・オブジェクト名、およびポート名の組み合わせを使用して、オペレーション名を作成します。WSDL オペレーションの Body Namespace は、WSDL 構成ウィザードの「ターゲット・ネーム・スペース (Target Namespace)」項目を使用して構成されます。
- **WSCollaborations** ウィザードは、SAP XI コネクタのプロパティの階層を作成します。この階層には、BO Name、SOAP Body Name、SOAP Body Namespace、および SAP XI サービスとして公開されるコラボレーションのポートにおける各 WSDL オペレーションの Mode などのプロパティが含まれます。図 42 に、WSCollaborations プロパティのサンプルを示します。

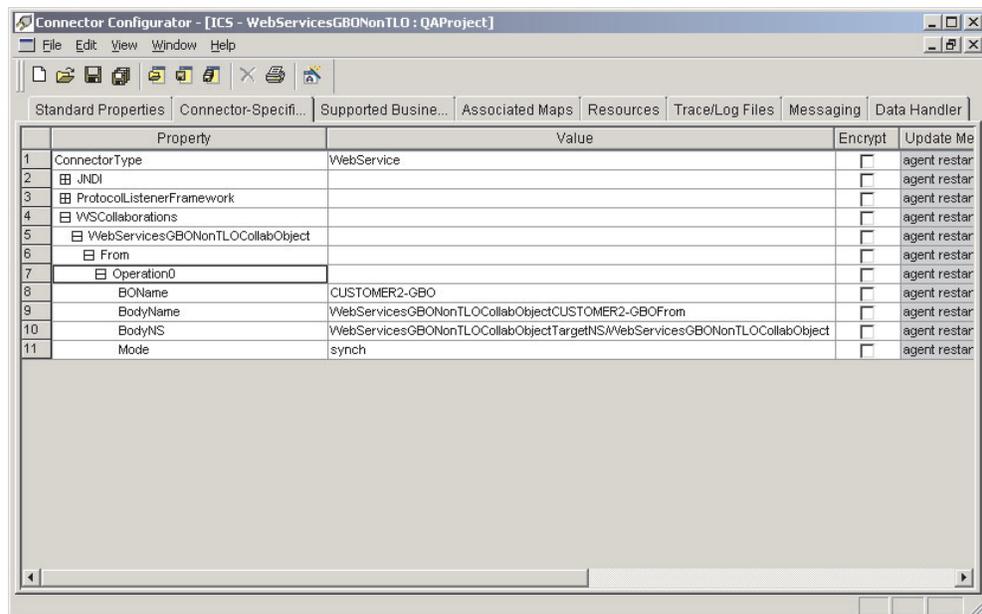


図 42. WSCollaborations

- **Execution Mode** WSDL オペレーションの Execution Mode は、WSDL 構成ウィザードの非 TLO 選択ボタンの「コラボレーション・モード (Collab Mode)」を使用して構成されます。

非 TLO に基づいて WSDL オペレーションを作成する場合、コラボレーションの構成には、マップを使用する方法と使用しない方法の 2 つの方法があります。

マップを使用する非 TLO: コラボレーションは、一般に、汎用ビジネス・オブジェクト (GBO) 要求を受け入れるように構成されています。また、GBO をコラボレ

ーションから非 TLO ビジネス・オブジェクトに変換する既存のマップが存在する場合があります。図 43 に、この例を示します。

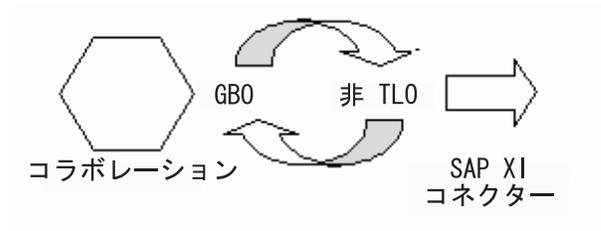


図 43. マップを使用する非 TLO

この場合、ウィザードは非 TLO のビジネス・オブジェクトを使用して、WSDL 文書に記述されている WSDL オペレーションを作成します。SAP XI コネクターのランタイム処理にとっては、構成されたマップが常にコラボレーションの GBO を 1 つの非 TLO だけに変換することが重要です。また、インバウンド・マップのソース・ビジネス・オブジェクトと宛先ビジネス・オブジェクトが、それぞれ、アウトバウンド・マップの宛先ビジネス・オブジェクトとソース・ビジネス・オブジェクトに正確に変換されることも重要です。

マップを使用しない非 TLO: 高度に専門化されたケースでは、コラボレーションが、GBO 以外のビジネス・オブジェクトからの要求を受け入れるように構成されている場合があります。この場合、非 TLO はコラボレーション用の直接のビジネス・オブジェクトであり、マップは存在しません。図 44 に、この例を示します。



図 44. マップを使用しない非 TLO

この場合、ウィザードは、このポートではマップが構成されていないと判断し、非 TLO のビジネス・オブジェクトを使用して、WSDL 文書に記述されている WSDL オペレーションを作成します。

処理の要件および例外

以下のセクションで説明する WSDL 構成ウィザードの要件は、異なる指示が明記されていないかぎり、すべてのタイプのオブジェクト (TLO および非 TLO) に適用されます。SAP XI サービス TLO に関するビジネス・オブジェクト要件の詳細については、21 ページの『第 3 章 ビジネス・オブジェクトの要件』を参照してください。

注: WSDL ツールが読み取るビジネス・オブジェクト ASI のうち、国際化対応した文字を使用できるのは以下のもののみです。

- elem_name

- elem_ns
- attr_name
- attr_ns
- BodyName
- BodyNS
- type_name
- type_ns

SOAP 構成 MO での Use プロパティのサポート: WSDL 構成ウィザードは、SOAP 構成 MO の Use プロパティをサポートしますが、SOAP 要求 BO と対応する SOAP 応答 BO の Use 値が異なる場合はエラーをスローします。Use 値を literal または encoded に設定して、WSDL 文書を生成できます。Use プロパティとその値に関する詳細については、88 ページの『SOAP メッセージに対する Style と Use の影響』を参照してください。

SOAP 構成 MO でのスタイルのサポート: コラボレーションを SAP XI サービスとして公開する場合は、rpc スタイルのみがサポートされます。Style を SOAP 構成 MO の文書として指定すると、ウィザードはエラーをスローします。

障害処理: SOAP 障害ビジネス・オブジェクト内の details 属性は、子属性を 1 つだけ備えることができます。それ以外の場合、ユーティリティーはエラーを生成します。

ユーティリティーは、障害ビジネス・オブジェクトを受け入れます。複数の障害ビジネス・オブジェクトを検出した場合、ユーティリティーは、最初の、あるいはデフォルトの障害ビジネス・オブジェクトのヘッダー・コンテナを処理します。処理は次のように行われます。

- バインディング・セクション内の soap:fault 要素については、Namespace は指定されません。
- Fault は常に、document スタイルおよび用途 literal を使用して指定されます。
- Message 部分は、element 属性を使用して指定されます。

ヘッダー障害の処理: ヘッダー障害は、WSDL 文書のバインディング・セクションで、soap:header の子要素である soap:headerfault として処理されます。ヘッダー障害の処理は、次のように、ヘッダー子ビジネス・オブジェクト内で指定された headerfault ASI を使用して行われます。

- soap:headerfault 要素については、Namespace は指定されません。
- ヘッダー障害は常に、document スタイルおよび用途 literal を使用して指定されます。
- Message 部分は、type 属性ではなく element 属性を使用して指定されます。

ヘッダーの処理: SOAP ヘッダー・コンテナ・ビジネス・オブジェクト内では、SOAP ヘッダー子ビジネス・オブジェクトとして複数のヘッダー属性が指定されます。ヘッダー・コンテナ・ビジネス・オブジェクトは、その ASI によって、soap_location=SOAPHeader のように識別されます。ユーティリティーによる処理の際に、バインディング・セクション内で、ヘッダー・コンテナ・ビジネス・オブジェクト内のそれぞれの属性ごとに soap:header 要素が作成されます。この場合、以下の規則が適用されます。

- ヘッダーは常に、document スタイルおよび用途 literal を使用して指定されます。
- Message 部分は、type 属性ではなく element 属性を使用して指定されます。
- elem_ns が指定されていない場合、ヘッダーは Body Namespace に書き込まれます。

注: ヘッダー・コンテナー・ビジネス・オブジェクトは、SOAP 要求、応答、または障害ビジネス・オブジェクトの子です。soap:header 要素については、namespace 属性は指定されません。

elem_ns ASI の処理: このユーティリティーは、メッセージ・パーツ・レベルの elem_ns ASI を無視します。その代わりに、elem_ns は 2 次レベルおよび低レベルの属性で使用されます。elem_ns が指定される場合、2 次レベルのビジネス・オブジェクト属性を別個のネーム・スペース内で定義することができます。

HTTP プロトコルの処理: WSDL 文書のサンプル・ポート・セクションを以下に示します。

```
<service name="StockQuoteWebService">
  <port name="StockQuoteWebServicePort" binding="intf:StockQuoteBinding">
    <soap:address location="http://localhost:8080/wbia/webservices/stockquoteservice"/>
  </port>
</service>
```

WSDL 構成ウィザードは、コンテキスト・パス内のホスト名とポートの値を使用します。コンテキスト・パスに含まれているのがホスト名もポートもない相対パスのみである場合は、Listener_HTTP 構成プロパティ下にあるホスト名およびポート・プロパティの値が、soap:address xml 要素内で location 属性を指定するのに使用されます。

第 8 章 SAP XI ODA の使用

- 140 ページの『BIA_XIWSDLUtil ツールの実行』
- 141 ページの『SAP XI ODA の始動』
- 141 ページの『SAP XI ODA の実行』
- 142 ページの『エージェントの構成』
- 145 ページの『WSDL 文書の指定』
- 145 ページの『選択内容の確認』
- 146 ページの『オブジェクトの生成』
- 147 ページの『制約事項』

注: SAP XI Object Discovery Agent (ODA) は、要求処理用のビジネス・オブジェクトおよび、WSDL インターフェース・ファイルが使用可能な場合に、イベント処理用のビジネス・オブジェクトを生成するために使用されます。

コラボレーションは、コネクターを使用して SAP XI サービスを呼び出します。あるいは、コラボレーションは、SAP XI サービスとして公開することができます。SAP XI サービスは WSDL (Web Services Description Language) を使用して記述されています。この章では、SAP XI Object Discovery Agent (ODA) を使用してビジネス・オブジェクトを生成する方法について説明します。コラボレーションが SAP XI サービスを呼び出すとき、およびコラボレーションを SAP XI サービスとして公開するときに、コネクターおよび SOAP データ・ハンドラーは、これらのビジネス・オブジェクトを使用します。

以下の 2 つの目的のために、SAP XI ODA を使用して、ビジネス・オブジェクトを生成します。

1. SAP XI ODA は、WSDL インプリメンテーション・ファイルを使用して、コラボレーション用にビジネス・オブジェクトを生成して、外部 Web サービスを起動することができます。
2. SAP XI ODA は、WSDL インターフェース・ファイルを使用して、SAP XI サービスとして公開されるコラボレーション用にビジネス・オブジェクトを生成することができます。

SAP XI ODA を使用してビジネス・オブジェクトを生成する手順は、次のとおりです。

1. ネイティブの SAP XI Integration Developer-Design ツールを使用して、WSDL を生成します。

注: この手順は SAP の資料に説明されています。その資料を参照してください。この手順の概要説明については、191 ページの『付録 C. クイック・ステップ』も参照してください。

2. ネイティブの SAP によって生成された WSDL を変換するには、SAP XI アダプターと共にインストールされた BIA_XIWSDLUtil ツールを使用します。この出力は、SAP XI ODA と組み合わせて使用できる、標準に準拠した WSDL 文書になります。詳細については、140 ページの『BIA_XIWSDLUtil ツールの実行』を参照してください。

3. SAP XI アダプターによる配置に適した TLO を生成するには、SAP XI ODA を使用します。詳細については、141 ページの『SAP XI ODA の始動』を参照してください。

SAP XI ODA は、Business Object Designer を使用するときには立ち上げることができます。SAP XI ODA は、BIA_XIWSDLUtil ツールによって変換された WSDL 文書を読み取って、コネクタおよび SOAP データ・ハンドラーが必要とするビジネス・オブジェクトを作成します。SAP XI ODA を使用すると、ビジネス・オブジェクト開発の作業が簡略化されます。

注: SAP XI ODA は、WSDL 内の SOAP/HTTP バインディングを処理します。

BIA_XIWSDLUtil ツールの実行

BIA_XIWSDLUtil は、SAP XI によって生成された WSDL ファイルを標準に準拠した WSDL ファイルに変換するコマンド行ツールです。出力は WSDL ファイルで、このファイルを SAP XI ODA が使用すると、アダプターと組み合わせて使用するための TLO を生成できます。

注: この手順を進める前に、SAP XI ツールを使用して WSDL ファイルを生成する必要があります。詳細については、SAP XI の資料を参照してください。さらに、WBI Adapter Framework と SAP XI アダプターもインストールしておく必要があります。

BIA_XIWSDLUtil ツールを開始する手順は、次のとおりです。

1. コマンド・プロンプトのウィンドウまたはシェルを開きます。
2. SAP XI アダプターが格納されているディレクトリーに移動します。ディレクトリーは、通常、次のとおりです (14 ページの『インストール済みファイルの構造』を参照)。
 - **Windows** C:\¥WebSphereAdapters¥Connectors¥SAPXI
 - **UNIX** /home/WebSphereAdapters/Connectors/SAPXI
3. このディレクトリーに次のファイルが存在することを確認します。
 - BIA_XIWSDLUtil.bat (UNIX の場合は .sh)
 - SAPXIHeaders.xsd
 - SAPXIRFCHeaders.xsd
 - SAPXISystemError.xsd
4. SAP XI Integration Developer - Design ツールを使用して生成した WSDL ファイル (SalesOrder_GetList.wsdl など) をこのフォルダーにコピーします。
5. SAP XI コネクタが要求を listen する URL (例: http://sapxi:4444/mysevice/ws) を指定します。
6. 出力ファイル名 (例: SalesOrder_GetList_out.wsdl) を選択します。SAP インターフェース関数名を実際の SAP BAPI/IDOC 呼び出しと明確に関連付けるため、その関数名を指定することもできます。
7. コマンド行プロンプトで、次のコマンドを入力します。

```
BIA_XIWSDLUtil <input_filename> <SAPXI_URL> <output_filename>
```

ここで、以下のように説明されます。

- `<input_filename>` は、SAP XI Integration Developer-Design ツールを使用して生成した WSDL ファイルの名前を表します。
- `<SAPXI_URL>` は、SAP XI コネクタが要求を listen する URL を表します。
- `<output_filename>` は、SAP XI ODA と組み合わせて使用する準備が整っている WSDL ファイルに対して選択した名前を表します。
例えば、次のようにします。

```
BIA_XIWSDLUtil SalesOrder_GetList.wsdl http://sapxi:4444/myservice/ws
SalesOrder_GetList_out.wsdl
```

プログラムは、新規の WSDL ファイルが正常に作成されると、コマンド・プロンプトに戻ります。

8. 出力ファイルを SAP XI ODA ディレクトリーにコピーします。このディレクトリーは、通常、WBIAdapters Home/ODA/SAPXI となります。

SAP XI ODA の始動

SAP XI ODA は、以下のスクリプトのいずれかを使用して始動することができます。

- Windows
 - start_SAPXIODA.bat

注: SAP XI ODA は、インストーラーが Windows 環境用に自動的に作成するショートカットを使用して始動することもできます。

- UNIX
 - start_SAPXIODA.sh

SAP XI ODA の選択、構成、および実行は、Business Object Designer を使用して行います。Business Object Designer は、個々の ODA を、各スクリプトまたはバッチ・ファイルの AGENTNAME 変数で指定された名前で探し出します。

SAP XI ODA の実行

Object Discovery Agent (ODA) は、要求処理のためにビジネス・オブジェクトを構築する作業を簡略化します。Business Object Designer は、使用可能なすべての ODA に対するグラフィカル・インターフェースを備えているため、ユーザーは、必要なエージェントを見付けやすくなります。SAP XI ODA は、デフォルトでは SAPXIODA という名前になっています。WSDL Wizard で表示される名前は、start_SAPXIODA.bat または start_SAPXIODA.sh ファイル内の AGENTNAME 変数の値によって決まります。ODA およびビジネス・オブジェクト定義の詳細、および ODA を構成、開始、および使用する方法については、「*IBM WebSphere* ビジネス・オブジェクト開発ガイド」を参照してください。下記の手順を実行する際に、必要に応じてこの資料を参照することをお勧めします。

Object Discovery Agent を始動した後で、以下のステップに従って SAP XI ODA を立ち上げてください。

1. Business Object Designer を開きます。

2. 「ファイル」メニューから「ODA を使用して新規作成...」サブメニューを選択します。Business Object Designer が Business Object Wizard で「エージェントの選択」ダイアログ・ボックスを表示します。図 45 はこのウィンドウを示しています。
3. 「エージェントの検索」ボタンをクリックして実行中のすべてのエージェントを表示し、WSDL ODA を選択します。

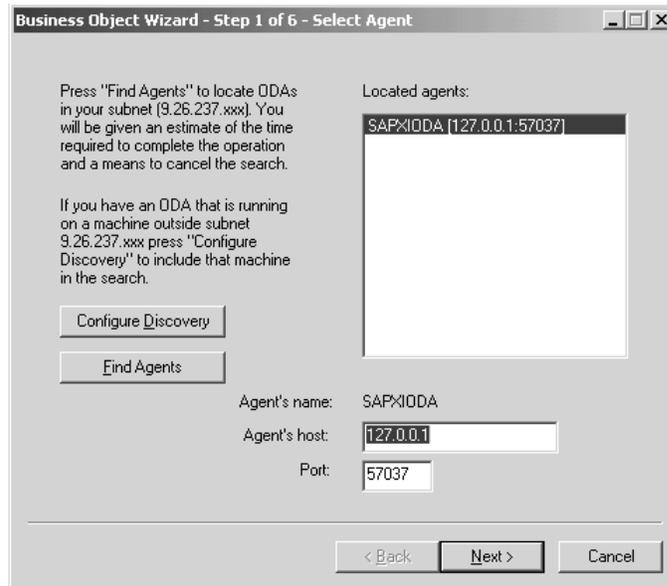


図 45. 「エージェントの選択」ウィンドウ

Business Object Designer で SAP XI ODA が見付からない場合には、ODA のセットアップを見直してください。

4. 「検索されたエージェント」ペインのリストから SAP XI ODA を選択し、「次へ」をクリックします。

これによって「エージェントの構成」ウィザード・ウィンドウが表示され、ユーザーが指定する必要がある構成プロパティが示されます。

エージェントの構成

図 46 は、SAP XI ODA Business Object Wizard の「エージェントの構成」ウィンドウを示しています。

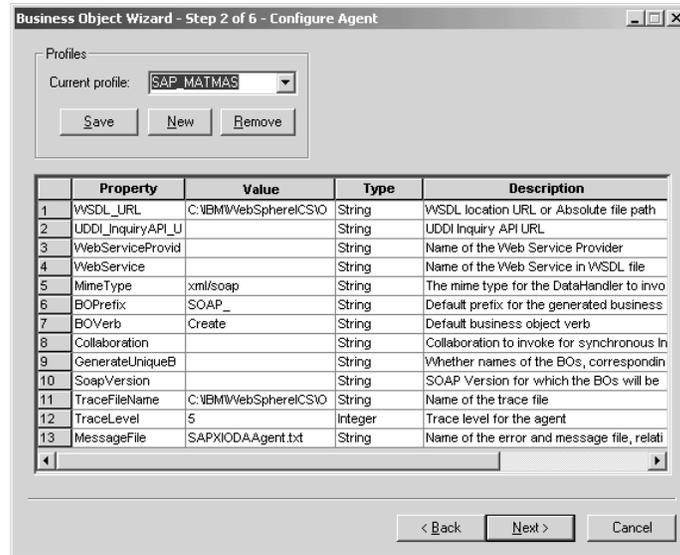


図 46. 「エージェントの構成」ウィンドウ

表 37 には、SAP XI ODA のために構成する必要があるプロパティがリストされています。

注: 初めて SAP XI ODA を使用する場合には、それぞれの各構成プロパティごとに値を指定する必要があります。それらの値を指定した後で、「保管」ボタンをクリックして、プロパティ値をプロファイルに保管することができます。次回に SAP XI ODA を使用するときには、「プロファイルを選択」ボックスから保管済みプロファイルを選択することができます。

表 37. WSDL ODA 構成プロパティ

プロパティ	型	必須	デフォルト	説明
WSDL_URL	String	UDDI を指定していないときには必須	なし	SAP XI 文書の URL。この値として、ローカル WSDL ファイルへの絶対パスを指定することもできます。URL はネイティブ言語で指定することができます。
UDDI_InquiryAPI_URL	String	UDDI の場合には必須	なし	UDDI 照会 API の URL。
WebServiceProvider	String	UDDI の場合には必須	なし	ターゲット Web サービス・プロバイダーの名前。これは通常、UDDI レジストリーで公開された Business 名です。この項目では大文字小文字の区別があり、英字のみを使用することができます。

表 37. WSDL ODA 構成プロパティ (続き)

プロパティ	型	必須	デフォルト	説明
WebService	String	UDDI の場合には必須		SAP XI サービスの名前。この項目では大文字小文字の区別があり、英字のみを使用することができます。
MimeType	String	いいえ	xml/soap	コネクタが呼び出すデータ・ハンドラーの MIME タイプ。これはビジネス・オブジェクト TLO でデフォルト値として設定される値であり、英字のみで指定しなければなりません。
BOPrefix	String	いいえ	SOAP_	これは、作成されるすべてのビジネス・オブジェクトの前に付加されます。ユーザー構成可能 (英字のみ)。8 文字以下で指定します。
BOVerb	String	はい	Create	要求ビジネス・オブジェクト (およびオプションにより応答ビジネス・オブジェクトと障害ビジネス・オブジェクト) の SOAP 構成 MO で設定される動詞。
Collaboration	String	いいえ	なし	この値は、TLO の ws_collab ASI を示し、イベント処理用にオブジェクトを生成する場合には必須となります。
GenerateUniqueBOs	String	いいえ	true	このプロパティが true の場合、ビジネス・オブジェクト名はすべての Web サービスの中で固有になります。このプロパティが false の場合、同じパーツ型を持つ操作の中でこのビジネス・オブジェクトを再使用できます。
SOAPVersion	String	いいえ	1.1	BO を生成するために使用する SOAP 標準を指定します。指定可能な値は 1.1 および 1.2 です。

次のセクションでは、「エージェントの構成」ウィンドウで WSDL 文書を指定する方法を説明します。

WSDL 文書の指定

SAP XI ビジネス・オブジェクトは、(BIA_XIWSDLUtil ツールで処理された) WSDL 文書を基にして生成されます。このセクションでは、ODA の「エージェントの構成」ウィンドウで WSDL 文書のソースを選択および指定する方法を示します。

WSDL 文書は、ローカル・ファイル・システムに存在します。WSDL 文書が存在する場所を指定すると、この WSDL 文書は、SAP XI ODA によって検索されます。

WSDL 文書を指定する手順は、次のとおりです。

1. BIA_XIWSDLUtil ツールを使用して生成した WSDL ファイルの場所を、構成プロパティ WSDL_URL の値として入力します。この文書はローカル・ファイル・システム上にありますが、URL 構文 (例: file://C:/test/wsdl) または絶対パス (例: C:%test%wsdl) を使用できます。この文書とその依存ディレクトリへのアクセス権、および WSDL ファイルのファイル名へのアクセス権が ODA にあることを確認する必要があります。その後 ODA は、WSDL 文書から SAP XI サービスのリストを検索します。

2. 「次へ」をクリックします。

ODA は、Web サービス・プロバイダーの URL を照会し、WSDL で定義されたサービスのリストをこの URL の場所で検索して、このリストを表示します。

注: WSDL ODA は、SOAP/JMS または SOAP/HTTP バインディングを備えたポートのみを表示し、それ以外のタイプのバインディングは除外します。

3. ポートのリストから、1 つだけオペレーションを選択します (選択可能なオペレーションが強調表示されています)。サービスまたはポート・ノードを選択することはできません。これらは表示専用です。WSDL オペレーションのタイプとしては、ONE_WAY、REQUEST_RESPONSE、SOLICIT_RESPONSE、および NOTIFICATION があります。WSDL ODA がサポートおよび表示するのは、REQUEST_RESPONSE オペレーションと ONE_WAY オペレーションのみです。

4. 「次へ」をクリックして『選択内容の確認』に進みます。

選択内容の確認

SAP XI 操作のソースを選択すると、「SAP XI ODA ビジネス・オブジェクト・ウィザード (SAP XI ODA Business Object Wizard)」により、次のような確認の画面が表示されます。

1. 選択内容を確認します。
2. 「次へ」をクリックして 146 ページの『オブジェクトの生成』に進みます。

オブジェクトの生成

ユーザーが、WSDL 文書ソースを確認すると、起動する必要がある SAP XI サービスまたは SAP XI サービスとして公開する必要があるコラボレーション用に、ビジネス・オブジェクトとメタオブジェクトが、SAP XI ODA によって生成されます。

注: SAP XI ODA は、トップレベルのビジネス・オブジェクトのキー属性を自動的に選択することはできません。それ以外のレベルのビジネス・オブジェクトでは、SAP XI ODA は最初の属性をキーに設定します。したがって、Business Object Designer で SAP XI ODA が生成したオブジェクトを保管すると、トップレベルのオブジェクトにキー属性が欠落していることを知らせるエラー・メッセージが出されます。ビジネス・データとビジネス・オブジェクトの要件を満たすキー属性を割り当ててから、オブジェクトを再度保管してください。キー属性を選択する際は注意が必要です。キー属性はイベントの順序付けに使用されるので、慎重に選択しないと、パフォーマンス上の問題が発生します。

1. 「ビジネス・オブジェクトをファイルに保管」、または「別のウィンドウで新規ビジネス・オブジェクトを開く」にチェックマークを付けます。後者を選択すると、Business Object Designer が立ち上がり、そのアプリケーションでビジネス・オブジェクトが開きます。
2. 「ODA をシャットダウン」にチェックマークを付けて「終了」をクリックします。

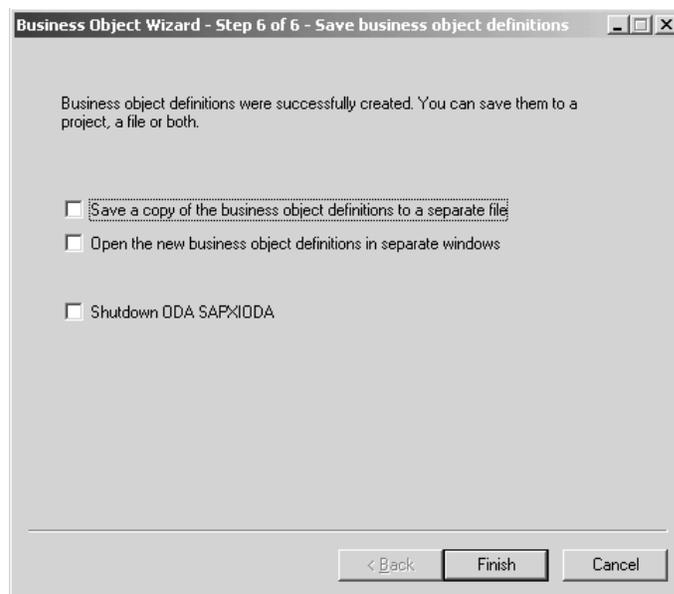


図 47. 「保管」ウィンドウ

要求処理の場合、Web サービスに対する呼び出しには、要求を設定する必要があります。また、同期処理では、応答メッセージと障害メッセージも設定する必要があります。イベント処理の場合、公開されるコラボレーションには要求を設定する必要があります。SAP XI ODA は、これらのそれぞれについて、各レベルのアプリケーション固有情報 (ASI)、SOAP データ・ハンドラー、およびプロトコル構成 MO を含む、ビジネス・オブジェクトを生成します。WSDL 文書における SOAP バインデ

イングにより、SOAP メッセージの構造が決定します。ビジネス・オブジェクト構造の詳細については、21 ページの『第 3 章 ビジネス・オブジェクトの要件』を参照してください。

制約事項

表 38 は、属性のスタイル、使用法、および型と要素を使用したパーツ定義をさまざまに組み合わせたものに対する、SAP XI ODA のサポート状況を示しています。

表 38. SAP XI ODA の制約事項

スタイル/用途/パーツ定義に使用	説明
rpc/encoded/type	サポートあり
rpc/encoded/element	サポートあり
rpc/literal/type	サポートあり
rpc/literal/element	サポートあり
doc/encoded/type	サポートなし
doc/encoded/element	サポートなし
doc/literal/type	サポートあり
doc/literal/element	サポートあり

SAP XI ODA は (1 つのファイルで) 完全に自己完結した WSDL ファイルか、または、サービス要素を含むインプリメンテーション・ファイル、他のすべての WSDL 要素 (型、メッセージ、ポート・タイプ、バインディング) を含むインターフェース・ファイル、およびスキーマ用の 1 つ以上のファイルに分離された WSDL ファイルを検索できます。SAP XI ODA は、複数のインターフェース・ファイル (例えば、メッセージとポート・タイプが 1 つのファイルにあり、バインディングが別のファイルにある場合など) を含む WSDL ファイルを正常に検索できません。

WSDL 文書内のスキーマは、ネーム・スペース・プレフィックスに関しては自己完結型でなければなりません。<types> 要素の子である <schema> 要素内の WSDL 文書の <definitions>/<types> 要素に定義されたネーム・スペース・プレフィックスを使用することはできません。ネーム・スペース・プレフィックスを <schema> 要素のサブエレメントで使用する場合は、<schema> 要素に対してネーム・スペース・プレフィックスを再定義する必要があります。自己完結型でないスキーマの例を以下に示します。

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:NS="NS">
  <types>
    <schema xmlns="http://www.w3.org/1999/XMLSchema">
      <element name="NSElem" type="NS:NSType"/>
    </schema>
  </types>
</definitions>
```

ネーム・スペース・プレフィックス NS が <definitions> 要素に定義され、<schema> 要素に再定義されることなく使用されます。したがって、SAP XI ODA はエラーをスローします。この制限を回避するには、ネーム・スペース・プレフィックス NS を <schema> 要素に再定義します。以下に例を示します。

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:NS="NS">
  <types>
    <schema xmlns="http://www.w3.org/1999/XMLSchema" xmlns:NS="NS">
```

```
        <element name="NSElem" type="NS:NSType"/>
    </schema>
</types>
</definitions>
```

第 9 章 トラブルシューティング

この章では、コネクターの始動または実行時に検出される問題について説明します。

始動時の問題

問題

アルゴリズムがサポートされていない/アルゴリズム「SSL」が使用できない。

可能なソリューション/説明

このエラーは、Connector Configurator で指定された SSL バージョンが JSSE プロバイダーによってサポートされない場合に発生します。ソリューション: サポートされる SSL バージョンについての JSSE プロバイダーの資料を調べます。IBM JSSE の場合は、*ProductDir/lib/security* ディレクトリー内の *java.security* ファイルに、以下の項目が含まれていることを確認します。

```
security.provider.<number>=com.ibm.jsse.  
IBMJSSEProvider
```

ここで、<number> は、セキュリティー・プロバイダーをロードする際の優先順位です。

鍵ストアのロード中にエラーが発生する。鍵ストア・ファイル・パス "<path>" が誤って指定されている。鍵ストアが見つからない。

このエラーは、鍵ストアおよび/またはトラストストア・ファイルに間違ったパスを指定した場合に発生します。ソリューション: Connector Configurator 内の「SSL」->「鍵ストア (KeyStore)」プロパティーで指定された鍵ストア・ファイル・パスを調べます。また、トラストストアを使用している場合は、Connector Configurator 内の「SSL」->「トラストストア (TrustStore)」プロパティーで指定されたトラストストア・ファイル・パスを調べます。

KeyManagementError。鍵ストアが改ざんされ、鍵管理エラーが出る。

このエラーは、鍵ストアおよび/またはトラストストアが改ざんされたか、あるいは破壊された場合に発生します。また、このエラーは、パスワードに間違った値を指定した場合にも発生します。ソリューション: 鍵ストアが改ざんされていないことを確認します。鍵ストアの再作成を試みます。また、正しいパスワードを

「SSL」->「KeyStorePassword」および

「SSL」->「TrustStorePassword」コネクター・プロパティーに入力したことを確認します。

鍵ストアから証明書のロード中にエラーが発生する。

このエラーは、証明書および/または鍵ストア、トラストストアが改ざんされた場合に発生します。また、このエラーは、パスワードに間違った値を指定した場合にも発生します。ソリューション: 証明書、鍵ストア、またはトラストストアが改ざんされているかどうかを調べます。また、正しいパスワードを「SSL」->「KeyStorePassword」および「SSL」->「TruststorePassword」コネクター・プロパティーに入力したことを確認します。

問題	可能なソリューション/説明
サーバー・ソケットの作成中にエラーが発生し、終了する: エラー。	このエラーは、SOAP/HTTP または SOAP/HTTPS プロトコル・リスナーが、コネクタ・プロパティで指定されたポートにバインドできない場合に発生します。ソリューション: すべての SOAP/HTTP および SOAP/HTTPS プロトコル・リスナーに対して指定したポートを調べます。同一のポートが複数のリスナーに対して指定されている場合は、リスナーの 1 つだけが始動できます。さらに、そのポート上で実行中のほかのサービスがあるかどうか調べます。そのポート上で実行中のほかのサービスがある場合には、別のポートをプロトコル・リスナー用に選択することもできます。
KeyManagementError:UnrecoverableKeyException、鍵が回復できなかった。	このエラーは、鍵ストアまたはトラストストアを使用できない場合に発生します。ソリューション: 鍵ストアを新規に作成します。
SSL ハンドシェイク例外: 不明なCA。	これは、CA 証明書がトラストストア内にない場合に発生します。ソリューション: CA の証明書のほか、その自己署名証明書がトラストストア内に存在しているかどうかを調べます。さらに、証明書の DN にホスト名 (なるべく IP アドレス) が含まれていることを確認します。基盤となる JSSE の詳細をすべてコンソール上に表示する必要がない場合は、Connector Configurator 内の「SSL」->「SSLDebug」プロパティの値を false に設定します。
ログ・ファイル内の過剰な JSSE ログに気付く。	コネクタが、プロトコル・リスナーの「プロトコル (Protocol)」プロパティに対して有効な値を取り出せませんでした。有効な値は、soap/http、soap/https、または soap/jms です。ソリューション: これはエラー状態ではありません。しかし、コネクタにこのリスナーを使用させる場合は、有効な値を「プロトコル (Protocol)」プロパティに指定します。
プロトコル・リスナーを指定したが、そのリスナーが初期化されないで、以下の警告メッセージがコネクタに表示される。	コネクタが、ハンドラーの「プロトコル (Protocol)」プロパティに対して有効な値を取り出せませんでした。有効な値は、soap/http、soap/https、または soap/jms です。ソリューション: これはエラー状態ではありません。しかし、コネクタにこのハンドラーを使用させる場合は、有効な値を「プロトコル (Protocol)」プロパティに指定します。
<pre>Skipping Protocol Listener Property Set "SOME_LISTENER_NAME" with protocol property "": unable to determine the protocol listener class.]</pre>	コネクタが、ハンドラーの「プロトコル (Protocol)」プロパティに対して有効な値を取り出せませんでした。有効な値は、soap/http、soap/https、または soap/jms です。ソリューション: これはエラー状態ではありません。しかし、コネクタにこのハンドラーを使用させる場合は、有効な値を「プロトコル (Protocol)」プロパティに指定します。
プロトコル・ハンドラーを指定したが、そのハンドラーが初期化されないで、以下の警告メッセージがコネクタに表示される。	コネクタが jms.jar を検出できません。ソリューション: jms.jar がコネクタのクラスパス内にあることを確認します。
<pre>Unable to determine the type of the handler; skipping initializing of current handler. Handler property details: Name: <Handler Name>; Value: Name: Protocol; Value: Name: ResponseWaitTimeout; Value: Name: ReplyToQueue; Value: .]</pre>	
<pre>java.lang.NoClassDefFoundError: Javax/jms/JMSEException...</pre>	

問題

検索に失敗しました: キュー "InProgressQueue"
指定したキュー名: "<queue name>"
JNDI を使用するキュー "<queue name>"
javax.naming.NameNotFoundException: <queue name>
(Fail to lookup, queue: "InProgressQueue"
for specified queue name: "<queue name>"
queue using JNDI "<queue name>"
javax.naming.NameNotFoundException: <queue name>)

初期化中にエラーが発生する。JNDI コンテキストが初期化されておらず、ユーザーは JMS プロトコルを使用できない。

初期コンテキストの取得中にエラーが発生した。

可能なソリューション/説明

SOAP/JMS Web サービスをコネクタとともに使用している場合は、キューを作成しないと、この問題が発生します。また、このエラーは、

「JNDI」->「LookupQueuesUsingJNDI」を true に設定していて、コネクタが JNDI を使用してキューを検索できない場合にも発生することがあります。ソリューション: コネクタに必要なキューを作成します。

「JNDI」->「LookupQueuesUsingJNDI」が true に設定されている場合は、コネクタに必要なキューを、JNDI を使用して検索できることを確認します。

SOAP/JMS プロトコル・リスナーまたは SOAP/JMS プロトコル・ハンドラーを使用するようにコネクタを構成した場合は、JNDI プロパティを指定する必要があります。ソリューション: 必要な JNDI コネクタ固有のプロパティを指定したことを確認します。JNDI プロバイダーの資料を参照して、JNDI プロバイダーに接続するために必要なライブラリーおよび JAR ファイルを確認します。必要な JAR ファイルがすべてコネクタのクラスパス内にあることを確認します。さらに、必要なライブラリーがすべてコネクタのパス内にあることを確認します。

SOAP/JMS プロトコル・リスナーまたは SOAP/JMS プロトコル・ハンドラーを使用するようにコネクタを構成した場合は、JNDI プロパティを指定する必要があります。また、このエラーは、JNDI プロパティを正しく指定しなかった場合にも発生します。ソリューション: JNDI プロパティを調べます。JNDI が適切に構成されていることを確認します。JNDI プロバイダーの資料を参照して、JNDI プロバイダーに接続するために必要なライブラリーおよび JAR ファイルを確認します。必要な JAR ファイルがすべてコネクタのクラスパス内にあることを確認します。さらに、必要なライブラリーがすべてコネクタのパス内にあることを確認します。

ランタイム・エラー

問題

HTTP 応答の解析中にエラーが発生する。HTTP 応答ヘッダーの読み取り中にストリームの終わりに到達した。

指定された URL でエラーが発生した。ホストおよびポートの詳細を抽出できない。宛先が誤っている。

<destination URL>

可能なソリューション/説明

このエラーは、コネクタが SOAP/HTTP Web サービスを呼び出すときに発生します。これは、ターゲット Web サービスが、間違った HTTP 応答を送信したために発生します。ソリューション: ターゲット SOAP/HTTP Web サービスのエンドポイント・アドレスが正しいことを確認します。

このエラーは、コネクタが SOAP/HTTP Web サービスを呼び出すときに発生します。これは、SOAP/HTTP Web サービスに間違ったエンドポイント・アドレスを指定したために発生します。ソリューション: Web サービスに正しいエンドポイント・アドレスを指定したことを確認します。

問題

動詞 <Verb> でイベント・ビジネス・オブジェクト <BO Name> をブローカーに送信中に失敗した。実行状況「-1」と以下のエラー・メッセージを受け取った。

MapException: コネクター WebServicesConnector のマップ・ビジネス・オブジェクト <BO Name> へのマップが見つかりません。(MapException: Unable to find the map to map business objects <BO Name> for the connector controller WebServicesConnector)

SOAP要求の要求ビジネス・オブジェクトへの変換に失敗する。 SOAP 障害:

要求オブジェクトの生成中に障害が発生しました。要求ビジネス・オブジェクトに動詞を設定できませんでした。(Failure in generating request object - no verb could be set on the request bo)

可能なソリューション/説明

このエラーが発生するのは、コネクターによるイベントの同期送信先のコラボレーションが存在していないか、あるいはビジネス・オブジェクトの動詞を受け入れないかのいずれかであるため、統合ブローカーがそのイベントの処理に失敗する場合です。ソリューション: イベント通知のために Web サービス TLO を使用している場合は、TLO の ws_collab オブジェクト・レベルの ASI を調べます。(TLO の名前は、エラー・メッセージに示されます。) ws_collab ASI の値を調べます。このコラボレーションが存在して、稼働していることを確認します。ws_mode BO レベルの ASI が synch に設定されている場合は、ws_collab ASI が必要です。ws_verb オブジェクト・レベルの ASI の値を調べます。ws_collab ASI で指定したコラボレーションを、ws_verb ASI で指定した動詞によって起動できることを確認します。イベント通知のために非 TLO を使用している場合は、WSCollaborations コネクター・プロパティを調べます。このビジネス・オブジェクトによって同期的に呼び出されるコラボレーションを見つけます。このコラボレーションが存在して、稼働していることを確認します。

このエラーは、コネクターが統合ブローカーに送信しようとしているビジネス・オブジェクトの動詞を判別できない場合に、イベント通知の際に発生します。ソリューション: イベント通知のために Web サービス TLO を使用している場合は、この TLO に対して ws_verb オブジェクト・レベルの ASI を指定したことを確認します。動詞をこの ASI の値として指定します。イベント通知のために非 TLO を使用している場合は、Web サービス・クライアントによって送信された SOAP メッセージに動詞要素が含まれていなければなりません。SOAP データ・ハンドラーは、SOAP メッセージ内の動詞要素の値を使用して、ビジネス・オブジェクトの動詞を設定します。Web サービス・クライアントが SOAP メッセージ内の動詞を送信しない場合、SOAP データ・ハンドラーは、ビジネス・オブジェクト上で動詞を設定できません。この場合、コネクターはビジネス・オブジェクトを統合ブローカーに渡すことができません。Web サービス・クライアントで SOAP メッセージ内に動詞要素が含まれていない可能性がある場合は、このビジネス・オブジェクトに対して DefaultVerb 動詞レベルの ASI を指定できます。そのようにする場合、コネクターは、この動詞をビジネス・オブジェクト上で設定してから、それを統合ブローカーに送信します。

付録 A. コネクターの標準構成プロパティ

この付録では、WebSphere Business Integration Adapter のコネクタ・コンポーネントの標準構成プロパティについて説明します。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)

コネクタによっては、一部の標準プロパティが使用されないことがあります。Connector Configurator 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のリストが表示されます。

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

注: 本書では、ディレクトリー・パスの規則として円記号 (¥) 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。

新規プロパティと削除されたプロパティ

以下の標準プロパティは、本リリースで追加されました。

新規プロパティ

- XMLNamespaceFormat

削除されたプロパティ

- RestartCount

標準コネクタ・プロパティの構成

アダプター・コネクタには 2 つのタイプの構成プロパティがあります。

- 標準構成プロパティ
- コネクタ固有の構成プロパティ

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

Connector Configurator の使用

Connector Configurator からコネクタ・プロパティを構成します。Connector Configurator には、System Manager からアクセスします。Connector Configurator の使用法の詳細については、本書の Connector Configurator に関する付録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。コネクタを UNIX システム上で稼働している場合でも、これらのツールがインストールされた Windows マシンが必要です。UNIX 上で動作するコネクタのコネクタ・プロパティを設定する場合は、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、コネクタ用の Connector Configurator を開く必要があります。

プロパティ値の設定と更新

プロパティ・フィールドのデフォルトの長さは 255 文字です。

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

1. デフォルト
2. リポジトリ (WebSphere InterChange Server が統合ブローカーである場合のみ)
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・プロパティの値を変更する場合は、プロパティの**更新メソッド**によって、変更を有効にする方法が決定されます。標準コネクタ・プロパティには、以下の 4 種類の更新メソッドがあります。

• 動的

変更を System Manager に保管すると、変更が即時に有効になります。コネクタが 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を変更できます。この場合、動的更新は実行できません。

• 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。

• コンポーネント再始動

System Manager でコネクタ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。

特定のプロパティ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 155 ページの表 39 の「更新メソッド」列を参照してください。

標準プロパティの要約

表 39 は、標準コネクタ構成プロパティの早見表です。標準プロパティの依存関係は RepositoryDirectory に基づいているため、コネクタによっては使用されないプロパティがあり、使用する統合ブローカーによってプロパティの設定が異なる可能性があります。

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

注: 表 39 の「注」列にある「Repository Directory は REMOTE」という句は、ブローカーが InterChange Server であることを示します。ブローカーが WMQI または WAS の場合には、リポジトリ・ディレクトリは LOCAL に設定されます。

表 39. 標準構成プロパティの要約

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
AdminInQueue	有効な JMS キュー名	CONNECTORNAME /ADMININQUEUE	コンポーネント再始動	Delivery Transport は JMS
AdminOutQueue	有効な JMS キュー名	CONNECTORNAME /ADMINOUTQUEUE	コンポーネント再始動	Delivery Transport は JMS
AgentConnections	1 から 4	1	コンポーネント再始動	Delivery Transport は MQ および IDL: Repository Directory は <REMOTE> (ブローカーは ICS)
AgentTraceLevel	0 から 5	0	動的	
ApplicationName	アプリケーション名	コネクタの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		コンポーネント再始動	
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> (ブローカーは ICS)
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

表 39. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
ControllerStoreAndForwardMode	true または false	true	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
ControllerTraceLevel	0 から 5	0	動的	Repository Directory は <REMOTE> (ブローカーは ICS)
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、値は JMS のみ
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

表 39. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
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> (ブローカーは ICS)
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> (ブローカーは ICS)
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> (ブローカーは ICS)
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_DEit_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 (ブローカーは ICS)
MaxEventCapacity	1 から 2147483647	2147483647	動的	Repository Directory は <REMOTE> でなければならない (ブローカーは ICS)
MessageFileName	パスまたはファイル名	CONNECTORNAMEConnector.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

表 39. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
OADAutoRestartAgent	true または false	false	動的	Repository Directory は <REMOTE> でなければなりません (ブローカーは ICS)
OADMaxNumRetry	正数	1000	動的	Repository Directory は <REMOTE> でなければなりません (ブローカーは ICS)
OADRetryTimeInterval	正数 (単位: 分)	10	動的	Repository Directory は <REMOTE> でなければなりません (ブローカーは ICS)
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 39. 標準構成プロパティの要約 (続き)

プロパティ名	指定可能な値	デフォルト値	更新メソッド	注
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
WsifSynchronousRequestTimeout	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

このセクションでは、各標準コネクタ構成プロパティの定義を示します。

AdminInQueue

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

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

AdminOutQueue

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

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

AgentConnections

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

`AgentConnections` プロパティは、`orb.init[]` により開かれる ORB (オブジェクト・リクエスト・ブローカー) 接続の数を制御します。

このプロパティのデフォルト値は 1 に設定されます。必要に応じてこの値を変更できます。

AgentTraceLevel

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

ApplicationName

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

BrokerType

使用する統合ブローカー・タイプを指定します。オプションとしては `ICS`、`WebSphere Message Brokers (WMQI、WMQIB、または WBIMB)`、または `WAS` があります。

CharacterEncoding

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

注: Java ベースのコネクターでは、このプロパティは使用しません。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、ドロップダウン・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。ドロップダウン・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、本書の `Connector Configurator` 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。

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

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

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

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

ContainerManagedEvents

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

デフォルト値はありません。

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 = /SOURCEQUEUE

また、MimeType、DHClass (データ・ハンドラー・クラス)、および DataHandlerConfigMOName (オプションのメタオブジェクト名) 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。

これらのプロパティーはアダプター固有ですが、例の値は次のようになります。

- MimeType = text/xml
- DHClass = com.crossworlds.DataHandlers.text.xml
- DataHandlerConfigMOName = M0_DataHandler_Default

「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・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。

- RepositoryDirectory がリモートの場合は、DeliveryTransport プロパティの指定可能な値は 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 プロパティに指定されている値が、MQ または IDL である場合、コネクタは、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・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・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・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、コネクタとクライアント・コネクタ・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です。

重要: 以下の環境では、コネクタに 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・コントローラーと (クライアント側の) コネクタの両方を始動するのは困難な場合があります。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` プロパティの値を 11 または 12 に設定する。このプロパティの詳細については、「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

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

このプロパティは、`false` に設定することもできます。

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

FaultQueue

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

デフォルト値は CONNECTORNAME/FAULTQUEUE です。

JvmMaxHeapSize

エージェントの最大ヒープ・サイズ (メガバイト単位)。このプロパティは、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・プロパティを必ず 設定してください。

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

jms.MessageBrokerName

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

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

リモート・メッセージ・ブローカーに接続すると、このプロパティは次の (必須) 値をとります。

QueueMgrName:<Channel>:<HostName>:<PortNumber>

各変数の意味は以下のとおりです。

QueueMgrName: キュー・マネージャー名です。
Channel: クライアントが使用するチャンネルです。
HostName: キュー・マネージャーの配置先のマシン名です。
PortNumber: キュー・マネージャーが listen に使用するポートの番号です。

例えば、次のようにします。

```
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456
```

jms.NumConcurrentRequests

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

デフォルト値は 10 です。

jms.Password

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

デフォルトはありません。

jms.UserName

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

デフォルトはありません。

ListenerConcurrency

このプロパティは、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。

このプロパティは、MQ トランスポートを使用するコネクタにのみ適用されません。DeliveryTransport プロパティには 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、ドロップダウン・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。ドロップダウン・リストに、サポートされる他の値を追加するには、製品ディレクトリーにある `¥Data¥Std¥stdConnProps.xml` ファイルを手動で変更する必要があります。詳細については、本書の `Connector Configurator` 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。コネクタがグローバル化に対応していない場合、このプロパティの有効な値は `en_US` のみです。特定のコネクタ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・バッファ内のイベントの最大数。このプロパティはフロー制御が使用し、`RepositoryDirectory` プロパティの値が `<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・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、製品ディレクトリーの `¥connectors¥messages` です。メッセージ・ファイルが標準位置に格納されていない場合は、メッセージ・ファイル名を絶対パスで指定します。

コネクタ・メッセージ・ファイルが存在しない場合は、コネクタは `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: 特定のコネクタについて、コネクタ独自のメッセージ・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ズ・ガイドを参照してください。

MonitorQueue

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

デフォルト値は CONNECTORNAME/MONITORQUEUE です。

OADAutoRestartAgent

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

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

自動再始動機能およびリモート再始動機能を使用可能にするには、このプロパティを true に設定する必要があります。MQ によりトリガーされる 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 によりトリガーされる OAD がコネクタの再始動を自動的に試行する回数の最大数を指定します。このプロパティを有効にするためには、OADAutoRestartAgent プロパティを true に設定する必要があります。

デフォルト値は 1000 です。

OADRetryTimeInterval

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

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

デフォルト値は 10 です。

PollEndTime

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

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

PollFrequency

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

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

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

PollQuantity

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がある場合、標準プロパティの値は、このコネクター固有のプロパティの設定値によりオーバーライドされます。

電子メール・メッセージもイベントと見なされます。コネクターは、電子メールに関するポーリングを受けたときには次のように動作します。

コネクターは、1 回目のポーリングを受けると、メッセージの本文を選出します。これは、本文が添付とも見なされるからです。本文の MIME タイプにはデータ・ハンドラーが指定されていないので、コネクターは本文を無視します。コネクターは PO の最初の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。

2 回目のポーリングを受けると、コネクターは PO の 2 番目の添付を処理します。この添付の MIME タイプには対応する DH があるので、コネクターはビジネス・オブジェクトを Visual Test Connector に送信します。これが受け入れられると、PO の 3 番目の添付が届きます。

PollStartTime

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

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

RequestQueue

統合ブローカーが、ビジネス・オブジェクトをコネクタ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

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

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、コネクタが InterChange Server リポジトリ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 の場合には、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・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

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

デフォルト値は 3 です。

RestartRetryInterval

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

デフォルト値は 1 です。

RHF2MessageDomain

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

このプロパティにより、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、このプロパティには `xml` も設定できます。このプロパティは、`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 対応コネクタでの保証付きイベント・デリバリーをサポートするコネクタ・フレームワークに、JMS ソース・キューを指定します。詳細については、161 ページの『`ContainerManagedEvents`』を参照してください。

デフォルト値は `CONNECTOR/SOURCEQUEUE` です。

SynchronousRequestQueue

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

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

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

SynchronousResponseQueue

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

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

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

SynchronousRequestTimeout

DeliveryTransport が JMS の場合のみ適用されます。

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

デフォルト値は 0 です。

WireFormat

トランスポートのメッセージ・フォーマットです。

- 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。

WsifSynchronousRequestTimeout

WAS 統合ブローカーでのみ使用されます。

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

デフォルト値は 0 です。

XMLNameSpaceFormat

WebSphere Message Broker 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。

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

注:

本書では、ディレクトリー・パスの規則として円記号 (¥) 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』
- 174 ページの 『Connector Configurator の始動』
- 175 ページの 『コネクタ固有のプロパティ・テンプレートの作成』
- 178 ページの 『新規構成ファイルの作成』
- 181 ページの 『構成ファイル・プロパティの設定』
- 190 ページの 『グローバル化環境における Connector Configurator の使用』

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・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。

- コネクタを構成するためのコネクタ固有のプロパティ・テンプレート を作成する。
- **コネクタ構成ファイル**を作成する。インストールするコネクタごとに構成ファイルを 1 つ作成する必要があります。
- 構成ファイル内のプロパティを設定する。
場合によっては、コネクタ・テンプレートでプロパティに対して設定されているデフォルト値を変更する必要があります。また、サポートされるビジネス・オブジェクト定義と、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)。

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

標準プロパティはすべてのコネクタにより使用されるので、標準プロパティ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の値を設定する必要があります。

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

ただし**コネクタ固有プロパティ**の場合は、最初にプロパティを定義し、その値を設定する必要があります。このため、特定のアダプターのコネクタ固有プロパティのテンプレートを作成します。システム内で既にテンプレートが作成されている場合には、作成されているテンプレートを使用します。システム内でまだテンプレートが作成されていない場合には、176 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 Tools」>「Connector Configurator」をクリックします。
- 「ファイル」>「新規」>「コネクタ構成」を選択します。

- 「システム接続: Integration Broker」の隣のプルダウン・メニューをクリックします。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 (181 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」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」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。

- 「System Manager」ウィンドウの「コネクタ」フォルダーでいずれかの構成ファイルを選択し、右クリックします。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。
- Connector Configurator で「ファイル」>「開く」を選択します。プロジェクトまたはプロジェクトが保管されているディレクトリーからコネクタ構成ファイルを選択します。
- 「標準のプロパティ」タブをクリックし、この構成ファイルに含まれているプロパティを確認します。

コネクタ固有のプロパティ・テンプレートの作成

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

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

- テンプレートの新規作成については、176 ページの『新規テンプレートの作成』を参照してください。
- 既存のファイルを使用する場合には、既存のテンプレートを変更し、新しい名前でのこのテンプレートを保管します。既存のテンプレートは `¥WebSphereAdapters¥bin¥Data¥App` ディレクトリーにあります。

新規テンプレートの作成

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

Connector Configurator でテンプレートを作成するには、以下のステップを実行します。

1. 「ファイル」>「新規」>「コネクタ固有プロパティ・テンプレート」をクリックします。
2. 「コネクタ固有プロパティ・テンプレート」ダイアログ・ボックスが表示されます。
 - 「新規テンプレート名を入力してください」の下の「名前」フィールドに、新規テンプレートの名前を入力します。テンプレートから新規構成ファイルを作成するためのダイアログ・ボックスを開くと、この名前が再度表示されます。
 - テンプレートに含まれているコネクタ固有のプロパティ定義を調べるには、「テンプレート名」表示でそのテンプレートの名前を選択します。そのテンプレートに含まれているプロパティ定義のリストが「テンプレートのプレビュー」表示に表示されます。
3. テンプレートを作成するときには、ご使用のコネクタに必要なプロパティ定義に類似したプロパティ定義が含まれている既存のテンプレートを使用できます。ご使用のコネクタで使用するコネクタ固有のプロパティが表示されるテンプレートが見つからない場合は、自分で作成する必要があります。
 - 既存のテンプレートを変更する場合には、「変更する既存のテンプレートを選択してください: 検索テンプレート」の下の「テンプレート名」テーブルのリストから、テンプレート名を選択します。
 - このテーブルには、現在使用可能なすべてのテンプレートの名前が表示されます。テンプレートを検索することもできます。

一般特性の指定

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

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

プロパティの一般特性の選択を終えたら、「値」タブをクリックします。

値の指定

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

1. 「値」タブをクリックします。「一般」のパネルに代わって「値」の表示パネルが表示されます。
2. 「プロパティを編集」表示でプロパティの名前を選択します。
3. 「最大長」および「最大複数値」のフィールドに値を入力します。

新規プロパティ値を作成するには、以下のステップを実行します。

1. 「プロパティを編集」リストでプロパティを選択し、右マウス・ボタンでクリックします。
2. ダイアログ・ボックスから「追加」を選択します。
3. 新規プロパティ値の名前を入力し、「OK」をクリックします。右側の「値」パネルに値が表示されます。

「値」パネルには、3つの列からなるテーブルが表示されます。

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

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

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

値が作成されて、グリッドに表示されると、そのテーブルの表示内から編集できるようになります。

テーブルにある既存の値の変更を行うには、その行の行番号をクリックして行全体を選択します。次に「値」フィールドを右マウス・ボタンでクリックし、「値の編集 (Edit Value)」をクリックします。

依存関係の設定

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

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

1. 「使用可能なプロパティ」表示で、依存プロパティとして指定するプロパティを選択します。

2. 「プロパティを選択」フィールドで、ドロップダウン・メニューを使用して、条件値を持たせるプロパティを選択します。
3. 「条件演算子」フィールドで以下のいずれかを選択します。

== (等しい)

!= (等しくない)

> (より大)

< (より小)

>= (より大か等しい)

<= (より小か等しい)

4. 「条件値」フィールドで、依存プロパティをテンプレートに組み込むために必要な値を入力します。
5. 「使用可能なプロパティ」表示で依存プロパティを強調表示させて矢印をクリックし、「依存プロパティ」表示に移動させます。
6. 「完了」をクリックします。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。

新規構成ファイルの作成

構成ファイルを新規に作成するには、構成ファイルの名前を指定し、統合ブローカーを選択する必要があります。

- 「System Manager」ウィンドウで「コネクタ」フォルダーを右クリックし、「新規コネクタの作成」を選択します。Connector Configurator が開き、「新規コネクタ」ダイアログ・ボックスが表示されます。
- スタンドアロン・モードの場合は、Connector Configurator で「ファイル」>「新規」>「コネクタ構成」を選択します。「新規コネクタ」ウィンドウで、新規コネクタの名前を入力します。

また、統合ブローカーも選択する必要があります。選択したブローカーによって、構成ファイルに記述されるプロパティが決まります。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。

- 「Integration Broker」フィールドで、ICS 接続、WebSphere Message Broker 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」ウィンドウの残りのフィールドに入力します。

コネクタ固有のテンプレートからの構成ファイルの作成

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

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

- **名前**

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

重要: Connector Configurator 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。

- **「コネクタ固有プロパティ・テンプレート」を選択します。**

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

使用するテンプレートを選択し、「**OK**」をクリックします。

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

重要: ここで設定するディレクトリー・パスおよび名前は、コネクタの始動ファイルで指定するコネクタ構成ファイルのパスおよび名前に一致している必要があります。

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

既存ファイルの使用

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。

- **コネクタ定義ファイル。**

コネクタ定義ファイルは、特定のコネクタのプロパティと、適用可能なデフォルト値がリストされたテキスト・ファイルです。コネクタの配布パッケージ

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 コネクタの場合は `CN_XML.txt` です)。

- ICS リポジトリー・ファイル。
コネクタの以前の 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。
- コネクタの以前の構成ファイル。
これらのファイルの拡張子は、通常 `*.cfg` です。

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

既存ファイルを使用してコネクタを構成するには、Connector Configurator 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」をクリックします。
2. 「ファイル・コネクタを開く」ダイアログ・ボックス内で、以下のいずれかのファイル・タイプを選択して、使用可能なファイルを調べます。
 - 構成 (`*.cfg`)
 - ICS リポジトリー (`*.in`、`*.out`)

ICS 環境でのコネクタ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せん。

- すべてのファイル (`*.*`)

コネクタのアダプター・パッケージに `*.txt` ファイルが付属していた場合、または別の拡張子で定義ファイルが使用可能である場合は、このオプションを選択します。

3. ディレクトリー表示内で、適切なコネクタ定義ファイルへ移動し、ファイルを選択し、「開く」をクリックします。

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

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

構成ファイルの完成

構成ファイルを開くか、プロジェクトからコネクターを開くと、「Connector Configurator」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」タブで、BrokerType プロパティ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が「標準のプロパティ」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 184 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「すべてファイルに保管」を選択し、コネクター構成をすべて System Manager プロジェクトに保管するには「すべてプロジェクトに保管」をクリックします。

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すべてに値が設定されているかどうかを確認されます。必須の標準プロパティ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
- コネクター固有のプロパティ
- 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の場合、データをビジネス・オブジェクトに変換するデータ・ハンドラーの構成に関して追加のカテゴリ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の場合、以下のプロパティの値も設定されている必要があります。

- 関連付けられたマップ
- 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値も設定可能です。ただし、標準のプロパティおよびコネクタ固有プロパティ、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。

標準プロパティとコネクタ固有プロパティの違いは、以下のとおりです。

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

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

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

標準コネクタ・プロパティの設定

標準のプロパティの値を変更するには、以下の手順を実行します。

1. 値を設定するフィールド内でクリックします。

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

アプリケーション固有の構成プロパティの設定

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

1. グリッドの左上端の部分で右マウス・ボタンをクリックします。ポップアップ・メニュー・バーが表示されます。プロパティを追加するときは「追加」をクリックします。子プロパティを追加するには、親の行番号で右マウス・ボタンをクリックし、「子を追加」をクリックします。
2. プロパティまたは子プロパティの値を入力します。
3. プロパティを暗号化するには、「暗号化」ボックスを選択します。
4. 182 ページの『標準コネクタ・プロパティの設定』の説明に従い、変更内容を保管するかまたは破棄するかを選択します。

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

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

コネクタ・プロパティの暗号化

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

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

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

更新メソッド

付録 A『コネクターの標準構成プロパティ』の 154 ページの『プロパティ値の設定と更新』にある更新メソッドの説明を参照してください。

サポートされるビジネス・オブジェクト定義の指定

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

注: コネクターによっては、アプリケーションでイベント通知や (メタオブジェクトを使用した) 追加の構成を実行するために、特定のビジネス・オブジェクトをサポートされているものとして指定することが必要な場合もあります。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」リストで空のフィールドをクリックします。
System Manager プロジェクトに存在するすべてのビジネス・オブジェクト定義を示すドロップダウン・リストが表示されます。
2. 追加するビジネス・オブジェクトをクリックします。
3. ビジネス・オブジェクトの「エージェント・サポート」(以下で説明) を設定します。
4. 「Connector Configurator」ウィンドウの「ファイル」メニューで、「プロジェクトに保管」をクリックします。追加したビジネス・オブジェクト定義に指定されたサポートを含む、変更されたコネクター定義が、System Manager の ICL (Integration Component Library) プロジェクトに保管されます。

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

1. ビジネス・オブジェクト・フィールドを選択するため、そのビジネス・オブジェクトの左側の番号をクリックします。
2. 「Connector Configurator」ウィンドウの「編集」メニューから、「行を削除」をクリックします。リスト表示からビジネス・オブジェクトが除去されます。
3. 「ファイル」メニューから、「プロジェクトの保管」をクリックします。

サポートされるリストからビジネス・オブジェクトを削除すると、コネクタ定義が変更され、削除されたビジネス・オブジェクトはコネクタのこのインプリメンテーションで使用不可になります。コネクタのコードに影響したり、そのビジネス・オブジェクト定義そのものが System Manager から削除されることはありません。

エージェント・サポート: ビジネス・オブジェクトがエージェント・サポートを備えている場合、システムは、コネクタ・エージェントを介してアプリケーションにデータを配布する際にそのビジネス・オブジェクトの使用を試みます。

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

ビジネス・オブジェクトがコネクタ・エージェントによってサポートされるよう指定するには、「エージェント・サポート」ボックスにチェックマークを付けます。「Connector Configurator」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: コネクタの最大トランザクション・レベルは、そのコネクタがサポートする最大のトランザクション・レベルです。

ほとんどのコネクタの場合、選択可能な項目は「最大限の努力」のみです。

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

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・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 を選択した場合、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可能なビジネス・オブジェクトのリストが示されます。このリストから必要なビジネス・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付けられた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リプション・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、どのマップを使用するかが決定されます。

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

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

「関連付けられたマップ」タブには以下のフィールドが表示されます。

- **ビジネス・オブジェクト名**

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

- **関連付けられたマップ**

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

- **明示的**

場合によっては、関連付けられたマップを明示的にバインドすることが必要になります。

明示的バインディングが必要なのは、特定のサポートされるビジネス・オブジェクトに複数のマップが存在する場合のみです。ICS は、ブート時、各コネクターでサポートされるそれぞれのビジネス・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 つのマップを見つけて、バインドしようとします。

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

以下の手順を実行して、マップを明示的にバインドします。

1. 「**明示的 (Explicit)**」列で、バインドするマップのチェック・ボックスにチェックマークを付けます。
2. ビジネス・オブジェクトに関連付けるマップを選択します。
3. 「Connector Configurator」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 に配置します。
5. 変更を有効にするため、サーバーをリブートします。

リソース (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できます。

すべてのコネクターがこの機能をサポートしているわけではありません。複数のプロセスを使用するよりも複数のスレッドを使用する方が通常は効率的であるため、Java でマルチスレッドとして設計されたコネクター・エージェントを実行している場合、この機能を使用することはお勧めできません。

メッセージング (ICS)

メッセージング・プロパティは、DeliveryTransport 標準プロパティの値として 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

コネクタ構成ファイルまたはコネクタ定義ファイルを開くと、Connector Configurator 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、指定する保管場所へ移動し、ファイル名を指定し、「保管」をクリックします。ログ・メッセージまたはトレース・メッセージは、指定した場所の指定したファイルに書き込まれます。

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

データ・ハンドラー

データ・ハンドラー・セクションの構成が使用可能となるのは、DeliveryTransport の値に JMS を、また ContainerManagedEvents の値に JMS を指定した場合のみです。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に使用する値については、付録 A 『コネクタの標準構成プロパティ』にある ContainerManagedEvents の下の説明を参照してください。その他の詳細は、「コネクタ開発ガイド (C++ 用)」または「コネクタ開発ガイド (Java 用)」を参照してください。

構成ファイルの保管

コネクタの構成が完了したら、コネクタ構成ファイルを保管します。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。
- 指定したディレクトリーに保管します。
- スタンドアロン・モードで、ディレクトリー・フォルダーに *.cfg 拡張子付きファイルとして保管します。デフォルトでは、このファイルは %WebSphereAdapters%bin%Data%App に保管されます。
- WebSphere Application Server プロジェクトをセットアップしている場合には、このファイルを WebSphere Application Server プロジェクトに保管することもできます。

System Manager でのプロジェクトの使用法、および配置の詳細については、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* システム・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「**BrokerType**」フィールドで、ご使用のブローカーに合った値を選択します。
現行値を変更すると、プロパティー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變更され、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 Locale 標準構成プロパティのドロップ・リストに表示されるのは、サポートされる値の一部のみです。ドロップ・リストに、サポートされる他の値を追加するには、製品ディレクトリーの %Data%Std%stdConnProps.xml ファイルを手動で変更する必要があります。

例えば、Locale プロパティの値のリストにロケール en_GB を追加するには、stdConnProps.xml ファイルを開き、以下に太文字で示した行を追加してください。

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

付録 C. クイック・ステップ

この付録では、SAP XI 用アダプターを要求処理およびイベント処理用に構成するためのクイック・ステップを示します。

要求処理

SAP XI 用アダプターを要求処理用に構成するには、以下の手順を行います。

1. 次の手順に従って、SAP XI システムから WSDL ファイルを生成します。
 - a. SAPXI (2.0) システムの GUI にログオンします。
 - b. 「リポジトリ (設計) (Repository(Design))」 を選択します。
 - c. オブジェクト (例: BAPI_SALESORDER_GETLIST) をインポートします。
 - d. インポートしたオブジェクト (例: BAPI_SALESORDER_GETLIST) をダブルクリックします。
 - e. 「ツール (Tools)」 -> 「WSDL のエクスポート (Export WSDL)」 を選択して、ローカル・ファイル・システムに WSDL ファイルを保管します。
2. BIA_XIWSDLUtil ツール、SAP XI ODA の順に使用して、ビジネス・オブジェクトを生成します。これらのツールは、WebSphere Business Integration Adapter for SAP XI に付属しています。詳細については、125 ページの『第 6 章 要求処理のためのコラボレーションの有効化』を参照してください。
3. Connector Configurator を使用して Adapter を構成し、生成されたオブジェクトをサポートするようにします。
4. SAP XI が未構成の場合は、以下のように構成します。
 - a. WebSphere Business Integration Adapter for SAP XI のソフトウェア・コンポーネントをシステム・ランドスケープ・ディレクトリー (SLD) で定義します。
 - b. Connector Configurator を使用して、HTTP(S) プロトコル・ハンドラーおよびリスナーを構成します。
 - c. ディレクトリーを保守します。
5. エンド・アプリケーションからの要求オブジェクトをテストします。あるいは、テスト要求を送信する場合は Test Connector を使用します。

イベント処理

SAP XI 用アダプターをイベント処理用に構成するには、以下の手順を行います。

1. 次の手順に従って、SAP XI システムから WSDL ファイルを生成します。
 - a. SAPXI (2.0) システムの GUI にログオンします。
 - b. 「リポジトリ (設計) (Repository(Design))」 を選択します。
 - c. オブジェクト (例: BAPI_SALESORDER_GETLIST) をインポートします。
 - d. インポートしたオブジェクト (例: BAPI_SALESORDER_GETLIST) をダブルクリックします。

- e. 「ツール (Tools)」 -> 「WSDL のエクスポート (Export WSDL)」を選択して、ローカル・ファイル・システムに WSDL ファイルを保管します。
2. BIA_XIWSDLUtil ツール、SAP XI ODA の順に使用して、ビジネス・オブジェクトを生成します。これらのツールは、WebSphere Business Integration Adapter for SAP XI に付属しています。詳細については、125 ページの『第 6 章 要求処理のためのコラボレーションの有効化』を参照してください。
3. WebSphere Business Integration Adapter for SAP XI を以下のように構成します。
 - a. Connector Configurator を使用して、生成されたオブジェクトへのサポートを追加します。
 - b. 必要に応じて、ビジネス・プロセス (コラボレーション) およびマッピングを定義します。
 - c. テストのため、エンド・アプリケーションを模擬するように Test Connector を構成します。
4. SAP XI を以下のように構成します。
 - a. WebSphere Business Integration Adapter for SAP XI のソフトウェア・コンポーネントをシステム・ランドスケープ・ディレクトリー (SLD) で定義します。
 - b. Connector Configurator を使用して、HTTP(S) プロトコル・ハンドラーおよびリスナーを構成します。
 - c. ディレクトリーを保守します。
5. IDoc またはその他のタイプのメッセージを起動するように SAP アプリケーションを構成します。
6. SAP アプリケーションでイベントを起動して、エンドツーエンドの接続性をテストします。

付録 D. HTTPS/SSL の構成

- 『鍵ストアのセットアップ』
- 194 ページの『トラストストアのセットアップ』
- 195 ページの『公開鍵証明書用の証明書署名要求 (CSR) の生成』

SSL の使用を計画している場合は、鍵ストア、証明書、および鍵生成を管理するために、サード・パーティーのソフトウェアを使用する必要があります。SAP XI コネクターにはこれらの作業用のツールは備わっていません。ただし、IBM JRE に同梱の Keytool の使用を選択して、自己署名証明書を作成し、鍵ストアを管理することもできます。

鍵および証明書管理ユーティリティー、すなわち鍵ツールにより自己所有の公開鍵/秘密鍵の鍵ペアおよび関連証明書の管理が可能です。これらは、自己認証 (他のユーザーまたはサービスに対して自分自身を認証させる)、またはデータ保全性もしくはデジタル署名を使用する認証サービス用に使用されます。さらに鍵ツール・ユーティリティーでは、通信相手の公開鍵を証明書の形態で保管することが可能です。

この付録では、鍵ツールを使用して鍵ストアのセットアップを行う方法について説明します。この付録は、具体的な例の説明のみを意図しており、鍵ツールまたは関連製品の資料を置き換えるものではありませんので、ご注意ください。鍵ストアのセットアップでツールを使用する際は、必ずソース資料を参照してください。鍵ツールの詳細については、次の web サイトを参照してください。

- <http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html#security>

鍵ストアのセットアップ

鍵ツールを使用して鍵ストアを作成するには、まず最初に鍵ストアに鍵ペアを作成する必要があります。例えば、コマンド行に次のように入力します。

```
keytool -genkey -alias wsadapter -keystore c:%security%keystore
```

これにより、鍵ツールからパスワードを入力するよう求めるプロンプトが出されます。鍵ツール・パラメーター内で選択したパスワードを入力することも可能ですが、鍵ツールで入力したパスワードは、SSL ” KeyStorePassword コネクター・プロパティーの値として指定する必要があります。詳しくは、78 ページの『KeyStorePassword』を参照してください。

このコマンド例は、c:%security%keystore ディレクトリーで命名された keystore の鍵ストアを作成します。したがって、c:%security%keystore を SSL ” KeyStore コネクター階層プロパティーの値として入力することができます。また上記例のように、コマンド行から -alias wsadapter を SSL ” KeyStoreAlias コネクター階層プロパティーの値として入力することもできます。次に、鍵ツール・ユーティリティーから、証明書の詳細についてプロンプトが出されます。以下は、各プロンプトとその入力例を示しています。(鍵ツール文書を参照してください。)

```
What is your first and last name?  
[Unknown]: HostName  
What is the name of your organizational unit?  
[Unknown]: wbi  
What is the name of your organization?  
[Unknown]: IBM  
What is the name of your City or Locality?  
[Unknown]: Burlingame  
What is the name of your State or Province?  
[Unknown]: CA  
What is the two-letter country code for this unit?  
[Unknown]: US  
Is <CN=HostName, OU=wbi, O=IBM, L=Burlingame,  
ST=CA, C=US> correct?  
[no]: yes
```

鍵ツールからパスワードを入力するよう求める次のプロンプトが出されます。

```
Enter key password for <wsadapter> (RETURN if same as keystore password):
```

同じパスワードを使用するには「Return」を選択します。自己署名証明書を使用する場合は、上記で作成済みの証明書のエクスポートが可能です。その場合は、コマンド行で次のように入力します。

```
keytool -export -alias wsadapter -keystore c:%security%keystore -file wsadapter.cer
```

ここで、鍵ツールから、鍵ストア・パスワードを入力するよう求めるプロンプトが出されます。上記で入力したパスワードを入力します。

トラストストアのセットアップ

トラストストアを次のようにセットアップしたい場合について説明します。SOAP/HTTPS プロトコル・リスナーを web サービス・クライアントに認証したい場合は、SSL ” UseClientAuth コネクターの構成プロパティを true に設定します。このケースでは、SOAP/HTTPS プロトコル・リスナーは、すべてのトラステッド web サービス・クライアントに対応した証明書を、トラストストアが収容するよう要求します。コネクターは、JSSE デフォルト・メカニズムを使用してクライアントを信頼することに注意してください。SOAP/HTTPS SAP XI サービスを呼び出す場合は、SOAP/HTTP-HTTPS プロトコル・ハンドラーはトラストストアが SAP XI サービスを信頼することを要求します。これは、トラストストアがトラステッド SAP XI サービスの証明書をすべて収容していなければならないことを意味しています。コネクターは、JSSE デフォルト・メカニズムを使用してクライアントを信頼することに注意してください。トラストストアにトラステッド証明書をインポートするには、次のようにコマンドを入力します。

```
keytool -import -alias trusted1 -keystore c:%security%truststore -file  
c:%security%trusted1.cer
```

ここで、鍵ツールから、鍵ストア・パスワードを入力するよう求めるプロンプトが出されます。-keystore c:%security%truststore と入力する場合は、SSL ” TrustStore 階層プロパティが c:%security%truststore に設定されていることを必ず確認してください。また、SSL ” TrustStorePassword 階層プロパティの値を、前に入力したパスワードに設定する必要があります。

公開鍵証明書用の証明書署名要求 (CSR) の生成

ユーザーの身元を信頼しているトラステッド・パートナーの中での SSL データ交換では、自己署名証明書は適切と考えられます。ただし、証明書が証明機関 (CA) により署名されていれば、より他者から信頼されやすくなります。

鍵ツール・ユーティリティーを使用して CA で署名された証明書を取得するには、まず最初に Certificate Signing Request (CSR) を生成し、その CSR を CA に渡します。次に、CA は証明書に署名してユーザーに戻します。

次のコマンドの入力により CSR が生成されます。

```
keytool -certreq -alias wsadapter -file wsadapter.csr  
-keystore c:%security%keystore
```

このコマンドで、`alias` は、秘密鍵用に作成した鍵ストアの別名です。鍵ツール・ユーティリティーは CA に提供する CSR ファイルを生成します。次に CA より署名済みの証明書が提供されます。この証明書を鍵ストアにインポートする必要があります。そのためには、次のコマンドを入力します。

```
keytool -import -alias wsadapter -keystore c:%security%keystore -trustcacerts  
-file casignedcertificate.cer
```

一度インポートすると、鍵ストアの自己署名証明書は CA 署名済み証明書で置き換えられます。

特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

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

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

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

著作権使用許諾

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

プログラミング・インターフェース情報

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

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

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

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

商標

以下は、IBM Corporation の商標です。

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

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

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

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。



IBM WebSphere Business Integration Adapter Framework V2.4.0



Printed in Japan